RSX-11/IAS SIG DECUS U.S. CHAPTER

THE MULTI-TASKER

Volume 15, Number 7 February, 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

Columns

Articles

READ THIS FIRST

The "Software Performance Reports" column has a list of various problems in Autopatch E. This list was obtained from a variety of sources and **HAS NOT BEEN VERIFIED.** Please use the information with care, and report any errors or new problems to the Multi-Tasker.

This issue has a couple of key corrections for SIG tape software. Most important is a bug recently discovered in the Virtual Disk package that can cause a 22-bit system to crash. See the report from the Virtual Disk working group for a fix. Also see "Hints and Things" for a correction to the version of INDEX on the Los Angeles SIG tape.

FROM THE EDITOR

This issue has something for everyone. There are excellent articles for all levels of readers, from the transcript of the Los Angeles Symposium RSX-11M/M Plus Question and Answer Session to a note from Eric Levy on how to convert RSX-11M's scheduler into a Unix-style timesharing algorithm!

The issue is also missing two key articles I had planned for it. At the Los Angeles symposium, I asked attendees to copy the Multi-Tasker in their trip reports. I have received a two-inch stack of reports and am in the process of distilling this information down to a master trip report for the symposium. This will be the lead article in the next issue. Also, I am starting to receive many SPR's, per the request at the Symposium and last issue. Next month I will begin sorting through them and putting as many into print as possible.

The Multi-Tasker staff is slowly being formed. The lack of speed is all my fault, however four people are already contributing: Elizabeth Bailey, Gail Green, Jim Oberlin, and Wendy Cockerline. Elizabeth and Jim entered some of the articles in the this issue. Gail has the title of Multi-Tasker Historian and will be writing the "From Five Years Ago" column in the future. Wendy volunteered to start a new column especially for new users of RSX/IAS and her lead column is in this issue.

The column (temporarily named "New Users" until someone comes up with something snappier) has me very excited. One short-coming of the Multi-Tasker is the lack of material written directly for the brand new person to RSX/IAS. There are many reasons for this. One is that the best people to write for new users are new users themselves. Wendy and I hope that this column will be a focal point for your input, whether it is a question on what does SYSGEN question 28 (Disk Writecheck) mean, a note on how you managed to get a PLAS region to work, or suggestions on topics you would like to see discussed in the Multi-Tasker. I and other experienced RSX/IAS SIG members will back Wendy up to handle any technical areas and she will make sure our none of our cryptic acronyms make it into print.

Finally, Phil Stephensen-Payne sent me a note on the XDT hint in device drivers published in the November 1981 issue of the Multi-Tasker. While purusing DECUS tapes and other non-proprietary software, he encountered the idea and put it in his master collection, which he later submitted to the Miami RSX/IAS SIG tape. However, he is not the original author and did not want to take credit for it. Unfortunately, we have lost who the original author was; so contact the Multi-Tasker to get the applause you deserve.

Ralph Stamerjohn
Multi-Tasker Editor

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

JUST FOR FUN

FROM THE EDITOR

Announcing the first, periodical Multi-Tasker "Just for Fun" contest. Send your entries in now and see your **NAME IN PRINT.** The grand prize winners will be selected in a completely arbitrary fashion by the editor and will win something he can obtain cheaply. Entries will be published in the May 1982 issue and a new contest started then.

Computers are very frustrating. They take things to literally and are the ultimate device for finding the impossible case. Machines seem to take pride in executing faster than a speeding bullet when things are going wrong and like a snail when you have a deadline. All of us have imagined ways we would like to take our frustrations out on them.

The contest this month is to fill in the statement "Just once I would like to ...". Some examples of entries are:

...call in an air strike on my CPU.
...feed my backup tapes through a paper shredder.

FROM FIVE YEARS AGO

Five years ago (Vol.7, No.2), an announcement was made that the SIG would no longer be publishing SPRs in the Multi-Tasker. Initially, SPRs were printed in the Multi-Tasker because Digital did not print raw SPRs in the Software Dispatch. The Multi-Tasker continued to carry SPRs even after Digital changed its policy and started publishing raw SPRs -- Digital was not publishing all raw SPRs received. As a result of a meeting with DEC at Fall DECUS '76, Digital agreed to review their to-publish or not-to-publish screening process of raw SPRs. Although SPRs would no longer be published in the Multi-Tasker, the SIG requested that a copy of every SPR submitted to Digital also be submitted to the

SIG, to help track Digital's performance. The Multi-Tasker's no-SPR policy lasted seven months. Digital's screening of SPRs for publication remains an issue today. (See "Speak Out," Vol.15, No.8.)

The rest of the February '77 issue contained:

* A method of implementing overlapped disk seeks for RSX-11M RK05's. This was done by declaring a seperate SCB for each disk drive unit and modifying DKDRV to do an overlapped seek if the driver is already doing a transfer request. Seek interrupts are ignored and when transfers finish, DKDRV exmaines its own internal table to find the next transfer request.

* User response to the DECUS/Atlanta wish list. The full duplex terminal driver was a popular item.

* A program authored by Ray French which reconstructs files corrupted by an aborted task.

* DEC's reply to the European SIG's 1976 resolutions:

  - Digital reported that internal documentation for RSX would be available within the next year.

  - Assurance that new features were being added to RSX-11D (Version 6.2), and that there was no plan at present to drop support of RSX-11D.

  - A promise to improve on the time between announced availability and delivery of products to customers. Digital explained that the problem was due in part to its desire to get new products in the hands of live users as soon as possible. This zeal, in combination with optimism about "everything going well," sometimes causes underestimates of the time required to get the product deliverable to customers.

* From the Fall '77 DECUS Symposium in Las Vegas, came resolutions asking for machine-readable software update packages, and for expansion of the RSX-11D hot-line to include calls on IAS, RSX-11S, and RSX-11M.

* A report from the I/D Space Working Group discussed the desirability of separate I/D space in RSX/IAS operating systems, and possible means of implementing the required changes to the operating system.

Gail Green
Multi-Tasker Historian

NEW USERS

As a first timer to DECUS symposia as well as a newcomer to RSX-11M, I was not in Los Angeles long before I found my mind as enveloped as my body in a seemingly impenetrable haze. Each successive session contributed another layer of confusion - filled with meaningless acronyms - rather than bring order to the technical chaos.

To offset my growing sense of impotence arising from this mental disarray, I began to think my frustrations were the same as many others and the problem was not me. As I voiced this concern to more experienced members of the SIG, I learned that they too wondered whether the needs of novice users were adequately addressed by the symposium presentations. Encouraged, my focal point enlarged to include all novice SIG members and whether the Multi-Tasker or the symposia was meeting their needs.

Not long thereafter, I found myself expressing these thoughts to the Multi-Tasker editor, Ralph Stamerjohn. He encouraged me to edit a column specifically directed toward this group. This introductory article thus represents my first attempt to do exactly that. The remainder of it consist of outlining what I perceive as the needs of novice users as well as my conception of how this column may best serve those needs.

Almost by definition, the foremost need of a novice is access to information that he or she can understand. Thus, the primary goal of this column will be to provide articles addressing fundamental topics in application and system programming under RSX-11/IAS. These articles, which I envision to be written by both experienced users as well as novices as they gain mastery over a given area, will be conceptually based and therefore minimize the amount of detail presented in favor of concentrating on the basic concepts underlying the topic. In addition, general interest as well as hardware oriented submissions of this same flavor will also be welcomed.

As a second thrust, this column will hopefully serve also as a forum which novices may feel free to ask questions, provide answers, and make suggestions directed toward how the SIG, especially the Multi-Tasker and symposia, can better serve your needs.

Submissions to this column should be made in the same form as any other contribution to the Multi-Tasker with the exception that they be mailed to the attention of this column rather than to that of the editor. To start with, I would like ideas on what to name this column. Submit you ideas and you will get to see your name in print! Also, one of the first subjects will be RSX-11M system generations. This is always a confusing area for the new user. Send me any questions you have on system generation, problems you had with your first system generation, or hints that now make the process easier.

Wendy Cockerline
New User Columnist

Phone: (415) 494-7400 x4921
(9-10:30 am, PST)

WORKING GROUP NEWS

VIRTUAL DISK WORKING GROUP

At the Fall, 1981 DECUS Symposium in Los Angeles, a Virtual Disk Working Group was formed. This group intends to standardize the various virtual disk packages into one master package, maintain this package for future releases of RSX/IAS, and examine and implement various proposed extensions to the package.

One serious problem has been discovered with some of the current distributions. The virtual disk driver (VDDRV) will crash your system if your CPU is a 22-bit machine VDDRV detects an error, i.e. a write to a readonly disk.

```
>AVD VD:=file/RO          ;Assign disk read-only
>MOU VD:label             ;Mount the disk
>UFD VD:[1,1]             ;UFD does write, causes crash
```

The fix is to locate the CALLR $IOALT instruction in VDDRV.MAC and change to a CALL $IOALT and a RETURN, i.e:

```
        CALLR   $IOALT          ;Finish I/O packet and return
to:
        CALL    $IOALT          ;Finish I/O packet
        RETURN                  ; and return to caller
```

If you have any suggestions for activites the working group should be involved with or would like to join the working group, please contact:

Mike Fraser
Armed Forces Radiobiology Research Institute
Computer Science Department
Mail Stop: CSD
Bethesda, MD 20814

RETIRED VERSIONS OF RSX-11 WORKING GROUP

At the Spring, 1982 DECUS Symposium in Miami, a working group was formed to provide a user-to-user mechanism for support of RSX-11M. This group will serve as a focal point for those sites which, for whatever reason, are unable or unwilling to upgrade to new RSX-11M releases. This includes sites running RSX-11M V3.2 who are not moving to V4.0.

Some of the services working group members have expressed an interest in providing include:

* A central list of known problems and a library of known fixes (if any).
* A mechanism for identifying SIG tape and DECUS library software which is usable on old versions.
* A collection of device drivers for new devices not supported by old versions.

If you are among those sites running old versions of RSX-11M or are not converting to RSX-11M V4.0, please send your name, address, configuration, and thoughts to:

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

HINTS & 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.

MEASURING FORTRAN PERFORMANCE

R.N. Stillwell
Baylor College of Medicine
Texas Medical Center
Houston, TX 77030

There are two jobs involved in writing a program: getting it to run and getting it to run well. A good performance monitor is a great help in optimizing a program. All you need to know is the half-dozen most active areas of your program in order to know where to put you effort. There is a nice, compact, Fortran monitor on the Spring 1980 DECUS tape (Chicago) in [307,25]TIMER.MAC and [307,25]TIMERX.FTN (this is the RATFOR account). The monitor is initialized by ca call from the program to be monitored and uses the USEREX subroutine to print when the program terminates a listing of subroutine names and the percent of time spent in each routine. I have modified to the routines to work with FOR V2.5 and to trap statement numbers as well as subroutines. I would be glad to supply a listing of the modified version to any user who would like it. Please write to me at the address above.

READING RSX-11M BRU TAPES UNDER IAS

Frank Borger
Micheal Reese Hospital
Chicago, IL

BRU as distributed with IAS V3.1 will not read RSX/IAS SIG tapes as produced on RSX-11M systems. Anyone who wants to read BRU format SIG tapes, and who doesn't, can patch BRU by one of the following:

1. Rebuild BRU using the TKB command file [11,30]BRUBLD.CMD, after editing the command file to enable the conditional documented therein. The existance of the patch was NOT documented anywhere in the IAS V3.1 manual, only in the release notes, page 13.

2. Since it is a one-word patch, it can also be entered by using the following dialogue.

```
MCR>SET /UIC=[1,1]                          !Write access to task
MCR>PIP [11,1]BRU11M.TSK=[11,1]BRU.TSK      !Create new copy of BRU
MCR>ZAP
ZAP>[11,1]BRU11M.TSK/LI
SEGMENT TABLE
000003: 000000-136217
000142: 136220-175433
000202: 136220-146577
_142:153072+12/
142:153104/000050
_24
_X
```

Note that the two task images are mutually exclusive, the RSX-11M version cannot read IAS BRU tapes and vice-versa. The above patch also works on version 3.0.

CORRECTIONS TO INDEX

Micheal N. LeVine
Naval Weapons Center
China Lake, CA 93555

At the Fall 1981 DECUS Symposium, the most current version of INDEX-FORTRAN Cross Referencer (V05.6) was submitted to the RSX SIG Tape (UIC [370,130]), RT SIG Tape and to the DECUS Library as an up date (11-229). Since that submittal, two problems have been found:

1. A feature of FORTRAN-77 that I was not aware of untill the symposium was the ability to express octal and hex numbers in the VAX-FORTRAN compatable form '17732'O or '1AF3'X . INDEX would see the O or X characters as variable names not as operators signaling the base of the number in quotes.

2. A programming error in the entry point cross reference that cause a variable to be mistaken for an entry point name if their existed in one of the modules an entry point of the same name (I thought I had fixed this problem with version 5.6).

The fix for both problems is to edit the module 'GET.MAC' to the form as shown below.

```
                .
                .
QTHAL:  INC     LINPT                   ; INC PTR TO NEXT CHR
        INC     SUEXP
        JSR     PC,MOVCHR
        TSTB    R0
        BEQ     OT                      ; NULL-DONE
        CMPB    #47,R0                  ; FOUND CLOSEING "'" YET
        BNE     QTHAL                   ; NOT YET
        CLR     SUEXP
;*********************************************************************
;***    change to handle F77 octal and hex constants starts here    ***
;*********************************************************************
        INC     LINPT
        CALL    MOVCHR                  ; GET NEXT CHAR
        CMPB    #'O,R0                  ; IS IT AN 'O' FOR F77 OCTAL CONSTANT
        BEQ     NTLG                    ; YES-SKIP IT AND SET UP FOR NEXT CHAR
        CMPB    #'X,R0                  ; HOW ABOUT AN 'X' FOR HEX F77 CONST.
        BNE     NOBIN                   ; IF YES-INC CHAR-ELSE JUST CLEAN UP
NTLG:   INC     LINPT                   ; SET PTR TO CHR AFTER HOL
NOBIN:  CLR     PNMFLG
;*********************************************************************
;***    end of change to handle F77 octal and hex constants         ***
;*********************************************************************
        RTS     PC
                .
                .
NGTVAR:
GTVAR:  TST     SKIP
        BNE     GTDATA
        CLR     PRNFLG                  ;SEE IF IT HAS A TRAILING '('
        CMP     #"CL,VARTYP
        BEQ     20$
        CMPB    R0,#'(
;*********************************************************************
;***    the next instruction is the one that needs to be changed to fix ***
;***    the entry point cross reference problem                      ***
;*********************************************************************
        BNE     21$                     ;NO-SKIP ARRAY/FUNCTION CHECK
;*********************************************************************
;***                    end of change                                ***
;*********************************************************************
        MOV     #LIST,R0                ;CHECK TO SEE IF IT IS A FUNCTION CALL
        MOV     #NLIST,R1
10$:    CMP     (R0)+,VARTYP            ;IS THIS A MATCH
        BEQ     20$                     ;YES-THIS IS AN ENTRY POINT DEFEINED ELSEWA
        DEC     R1                      ;LIST DONE
        BGT     10$                     ;NO-CHECK NEXT
        INC     PRNFLG                  ;ARRAY/FUNCTION-SET FLAG AND LET STORE
                                        ;FIGURE OUT WHICH
        BR      21$                     ;GO STORE IT
20$:    MOV     #100000,PRNFLG          ;TELL STORE THIS IS DEFINED ENTRY
                                        ;POINT-ENTERED ELSEWARE
21$:
        JSR     PC,STOVAR
GTDATA: CLR     PRNFLG                  ;RESET PAREN FLAG
                .
                .
```

The above modification updates INDEX to version 5.7. All occourances of the string '5.6' should be changed to '5.7' in the following modules:

INDEX.MAC
IOLINE.MAC
IDXGEN.FTN
INDEX.HLP
INDEX.DOC

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 potentally dangerous: incorrect usage could result in system crashes and other incorrect system operations.

NEW RSX-11M DIRECTIVES

Alex Yovanovich
Alcan Smelters and Chemicals Ltd.
P.O. Box 1800
Kitimat, British Columbia
Canada, V8C 2H2

Our PDP-11 systems migrated from IAS/RSX-11D to RSX-11M V3.2 in eary 1981. In doing so, it was necessary to overcome the lack of variable send/recieve directives in RSX-11M. It turned out to be trivial to rework the fixed RSX-11M send/recieve directive module to implement the variable forms to work exactly as documented in the RSX-11M Plus sections of the executive module. As a bonus, the SYSLIB.OLB library for RSX-11M includes the Fortran callable variable send/recieve modules intended for RSX-11M Plus, so the new directives could be

put to use immediately. The system macro library also has the macro forms of the directives.

In addition, it was necessary to implement a directive to set local event flags by task name and a directive to check pool fragmentation and availabilty. Both of these were very straight forward. Once these changes were in place, conversion of application systems required only two weeks.

I would be pleased to mail a copy of these directives to any seriously interested IAS or RSX-11M user. Please send a card or letter to the address above.

NON-PROTECTED MULTI-USER HELLO TASK

Alan Silverman
CERN
Geneva, Switzerland

This is a alternate technique to build a non-protected multi-user RSX-11M system then the one suggested by Margaret Knox in the November 1981 issue of the Multi-Tasker that we use at CERN.

First build a normal, protected multi-user system. When finished, edit the source of [12,10]HELLO.MAC as follows: between labels 170$ and 175$ there is an instruction BLOS 180$. Change this to a BR 180$. This leaves the terminal as privileged, no matter what group number the user uses. An alternative is to edit the compare before the BLOS to alter the RSX default of only allowing groups 1 to 7 to be privileged. You then assemble HELLO using the command line below, replace the module in [1,24]MCR.OLB, and use the TKB command file to build the new version.

```
>MAC HELLO=LB:[1,1]EXEMC/ML,[200,200]RSXMC/PA:1,[12,10]HELLO
```

While you are inside HELLO, there are two other mildly interesting suggestions. First, you can alter the default logon messages by changing the text strings at labels MS3:, MS4:, MS5:, and MS6:. For example, you may want a foreign language version of "Good Morning".

Also, one often receives the message "LOGINS ARE DISABLED" without knowing why or how long. We have devised a method of having HELLO type out a text file in such conditions. Define a standard file in which the system manager who disables logins describes why and how long the system will be unavailable. We use file LB:[1,2]NOLOG.TXT. In the introductary declaration of HELLO.MAC, add the following:

```
NOLDSP: .WORD   4
        .WORD   DEVNAM
        .WORD   5
        .WORD   DIRNAM
        .WORD   9
        .WORD   NLGNAM
NLGNAM: .ASCII  /NOLOG.TXT/
```


At the label ERR7: in the error processing section of HELLO, you will find the following code:

```
ERR7:   MOV     #ER7,R0
        MOV     #EX$ERR,EXSTAT
        BR      ERROR

ERR8:   ...
```

Replace the BR ERROR with a JMP LOGDSB. The add the new subroutine at some appropriate point, for example, just before the section on HELP subroutines.

```
LOGDSB: CALL    ERROUT                  ;Report message
        MOV     #ODPB,R4                ;Output DPB
        MOV     #IO.ATT,2(R4)           ;Attach terminal function
        CALL    QIO                     ;Attach user terminal
        MOV     #100000,FRCFLG          ;Force message
        MOV     #IO.CCO,ODPB+Q.IOFN     ;Set function code
        MOV     #NOLDSP,R1              ;Get filename block to output
        CALL    DSPFIL                  ;Type LB:[1,2]NOLOG.TXT
        CALL    DETACH                  ;Free terminal
        JMP     ERROR1                  ; and contine as normal
```

I will submit the SLP file we use at CERN to a forthcoming RSX SIG tape. Hopefully, this will be updated for RSX-11M V4.0 by then.

SOFTWARE PERFORMANCE REPORTS

This section contains SPR's submitted to the Multi-Tasker by users. SPR's should always be sent to DIGITAL. However, if you feel that a report should be published in the Multi-Tasker, you may send a duplicate copy to the editor at the addresses listed on the cover. Publication of an SPR in the Multi-Tasker does not imply endorsement by the SIG. Implementation of suggested fixes must be at the reader's own risk. The SPR's published in this column may be abstracts of the original submission and have not been checked for accuracy.

AUTOPATCH E PROBLEMS

The following information on problems with the Autopatch E distribution was gathered from a variety of sources and has not been totally verified. If you find an error with this information or other problems with Autopatch E, please get in contact with the Multi-Tasker.

1. The [31,60]P2POP.PAT file is bad and will cause the patched TKB to sometimes abort with an odd address trap or illegal instruction. The following lines in the file should be changed:

```
        .PSECT
$PC=.
APR:
.=$PC+2
APRMP:
.=$PC+12
HGHAD:
.=$PC+446
        CALL    PAT1
        NOP
.=$PC+1412
CVAPR:
.=$PC
```

The correct lines are as follows: Note, the new correction checksum for P2POP.POB will be 21461.

```
        .PSECT
$PC=.
APR:
.=$PC+2
APRMP:
.=$PC+12
HGHAD:
.=$PC+436
        CALL    PAT1
        NOP
        NOP
.=$PC+1154
CVAPR:
.=$PC
```

2. The distributed patch for the DXDRV is incorrect and if applied, could case a system crash. This patch is found in [11,60]CONCAT.ULB in entry DRV9. The incorrect lines read as follows:

```
-476,479,/; DS002/
        MOV     R0,S.CSR(R4)    ; FAKE CSR ADR WITH ERR BLOCK ADR
        CALL    @(SP)+          ; LOG THE ERROR
        MOV     R2,S.CSR(R4)    ; RESTORE THE CSR ADR IN SCB
```

The incorrect line is the SLP command, which should range from lines 476 to 480. The corrected lines read as follows:

```
-476,480,/; DS002/
        MOV     R0,S.CSR(R4)    ; FAKE CSR ADR WITH ERR BLOCK ADR
        CALL    @(SP)+          ; LOG THE ERROR
        MOV     R2,S.CSR(R4)    ; RESTORE THE CSR ADR IN SCB
```

3. While doing an ISgen, you will get an undefined symbol error. Also loading the IC driver following the ICgen loads the driver but not the vectors. The following is the updated source correction file to


```
replace [200,60]SGNICIS.COR.

OU:SGNICIS.CMD;2/-AU/-BF=IN:[200,200]SGNICIS.CMD;1
.;
.; MLH012 -- Allow # of DRS-11 modules to be zero
.;
.; TL025 -- Initialize the fork interlock flag to nonzero
.;
.; HB005 -- Fix interrupt vector mung for DRS/DSS
.;
-443,.,/;TL025/
        .BYTE   5*2,1
-457,.,/;TL025/
        .BYTE   4*2,1
-580,.,/;MLH012/
.IS10: .ASKN [0:16.] ZN1 19.'$SF1' Number of DRS-11 modules
-582,,/;HB005/
.CLOSE
.OPEN ['$UIC',10]VECTOR.TMP
-598,598,/;HB005/
.ENABLE DATA #0
-605,605,/;HB005/
.DISABLE DATA #0
-629,629,/;HB005/
.ENABLE DATA #0
-635,635,/;HB005/
.DISABLE DATA #0
-690,690,/;HB005/
.CLOSE
-783,,/;HB005/
.IF DEV EQ "IS" .GOTO SCBIS
.797,,/;HB005/
.GOTO SCBFN
SCBIS:
.CLOSE #0
.CLOSE #2
PIP ['$UIC',10]SYSTB.MAC/AP/NM=['$UIC',10]VECTOR.TMP
.OPENA #2 ['$UIC',10]SYSTB.MAC
PIP ['$UIC',10]VECTOR.TMP;*/DE/NM
.IFT $SAVE .OPENA ['$DFUIC']SYSSAVED.DAT
.SCBFN:
/
```

4. The [15,60]SHUTUP.COR file has a .ASCII statement which should be a .ASCIZ. This causes the SHUTUP task to overprint messages. The lines which read as follows:

```
-93,93,/;MLG110
MSG3A: .ASCII  /XXX MINUTES BEFORE SHUTDOWN/
```

should be changed on the autopatch kit to be the following:

```
-93,93,/;MLG110
MSG3A: .ASCIZ  /XXX MINUTES BEFORE SHUTDOWN/
```

5. The [230,40]NETGEN.COR is missing a period for a .IFT statement. The last three lines currently read as follows:

```
-1978,1979
IFT FTS .DATA #3 'TAB'.IFINS PIP PIP LB:[0,0]001004.DIR/PR:0
/
```

A period needs to be added to the second line as follows:

```
-1978,1979
.IFT FTS .DATA #3 'TAB'.IFINS PIP PIP LB:[0,0]001004.DIR/PR:0
/
```

RSX-11M QUESTION AND ANSWER SESSION

Bob Denny

Creative System Design
Pasadena, California

The RSX-11M Question and Answer Session was held at the Los Angeles Symposium on Thursday night. The Digital developers answered questions from the floor:

Steve Paavola	RSX Product Manager
John Covert	Magtapes, MOU/DMO, file system
Tim Martin	RSX-11M Plus executive, terminal driver
Mike Fox	RSX-11M Plus executive, MCR, DCL
Steve Rusich	VFY, FCS, CODRV, file system
Jim Salman	Queue manager, despooler, batch
Mike Harvey	RSX-11M executive

Opening comments from Steve Paavola included an annoucement that a RSX-11M logic manual for version 4.0 is funded but they are having problems finding a person to write the manual. The project will be starting soon.

From the Editor

This part of the session was transcribed by Brady Reed, Systems Control, Inc., Palo Alto, California.

Ralph Stamerjohn, Monsanto (M,M+)

Q: RSX-11M was billed as the real-time system, while M+ was the "big-hammer solution". However, RSX-11M Plus's latest releases have had all real-time performance features (notably disk seek optimization and overlapped seeks), and RSX-11M's changes have to do with development features not related to performance. My application is real-time and heavily disk-based, and I feel that DEC is turning its back on the real-time user who made the decision to stay with RSX-11M given the initial product presentations.
A: These features were not put into RSX-11M because of the pool required for the additional data structures.

Louis Stoll, Zia Company (M)

Q: I notice that the sources for F11ACP have been dropped from the distribution kit as of V3.2. Will other sources be dropped in the V4.0 release?
A: There isn't enough room on the small disk kits for F11ACP sources. Much of the contents of the release kits is determined by the packaging of the small disk (RK05, RL01) distribution kits. With the additional software for V3.2, something had to be removed and the F11ACP sources were chosen because they are not needed for system generation. No sources currently on a V3.2 kit will be removed from V4.0 and some new, unsepecified sources are added.

David Birkmeyer, Clark Equipment Co. (M+)

Q: How do I determine which installed tasks map to a specific common region?
A: Search the installed task list and for each TCB, go through the T.PCBV list to find attached regions.

Bruce Mitchell, 3M (M,M+)

Q: RSX-11M and RSX-11M Plus should support levels of privilege for privileged users. In particular, install, create UFD, and create account should be separate privileges.
A: DEC does not wish to add another word to the terminal data structures, but will do so, if necessary. They will take the question into consideration. A work around was suggested. Write a LOGIN.CMD file that slaves the terminal, inputs commands from the user, and then filters which commands it will execute. The command file would need to disable control-Z's. For example, this would allow you to setup a privilege account for operators that could only execute a limited set of commands and programs.

Ben Burch, Fermilab (M)

Q: "PRI filename" deletes the file even if protection is [R,R,R,R].
A: Fixed next release, Digital believed the problem also fixed in Autopatch A and an early V3.2 Software Dispatch.

Q: "QUE TT5:jobname/DEL" gives error "Jobname does not exist", whereas "QUE jobname/DEL" removes job from queue.
A: You might not have specified it right. If job was in the default queue this might not have found it.

Alan Ainsworth, Tektronix (M)

Q: Indirect Command File Processor -- A subroutine with a "begin-end" block containing a "goto" statement seems to wipe out all other labels after the "goto" is executed and a "return" is effected.

A: Submit an SPR.

Keith E. Gorlen, National Institute of Health (M)

Q: Is there a method for linking a resident FCS library to alternative RQCB/RLCB (request/release core block) modules to allow individual tasks to dynamically manage buffer pool using different pool managers but still link to one resident FCS library? This is needed because OMSI Pascal uses modules with the same name and this currently prevents Pascal task from being linked to FCSRES.
A: It may be possible using method similar to that used by cluster libraries, however, technique is fairly complicated.

Hugh Kennedy, Oakweald, Ltd. (M)

Q: Will there be full (official) documentation for interface to the queue manager for both spool requests and despoolers?
A: There is despooler documentation available through DECUS, but there is nothing on the CLI/PRINT request. In general, if users stick to named offsets, it should be OK.

Alex Yovanovich, ALCAN (M)

Q: What is the future of the RSX-11M and RSX-11M Plus products? I note that all current "11" processors have 22-bit architecture. Does this imply stabilization of RSX-11M (like IAS) or will RSX-11M and RSX-11M Plus merge, say in version 5.0.
A: We are obviously making an attempt to make RSX-11M Plus run on all our mainline hardware. On the other hand, RSX-11M is shipping is vastly larger volumes than RSX-11M Plus. We'd like to see just one RSX-11M product in the longer term. In the short term, we don't know how to do this. We don't know by what means a single long term RSX-11M product will be achieved. (Poll of audience indicated that given equal pricing, most RSX-11M users with 22-bit machines would convert to M+).

Steve Sylvester, C.A.C.I. (M)

Q: Problem with Print Queue Manager. We have 2 printers. We attempt to stop the printer, change the form number, then start the printer. Printer will not start, does not accept any output. The only way to restart it is to boot system. What is the proper syntax to do this. We think we are following the manual correctly.
A: No definite answer given in session.

From the Editor

This part of the session was transcribed by Billy Justice, ESD Corporation, Santa Clara, California.

Dennis Cook, Fisher & Porter (M)

Q: When will the TU58 driver be available which supports the modified TU58 protocol?
A: The first Autopatch after version 4.0 is released.

James A. Demange, Systeme Corporation (M)

Q: I have been talking to alot of the Digital people and they say we have a unique opportunity at this symposium because you are right at the start of the next release. The RSX menu seems to be a good way for us to get input to you. Can Digital make a menu of items to be in the next release, and solicit DECUS membership for comments so when you need to cut out items you cut out the lower items?
A: No. Due to a legal problem to do with predisclosure.

Philip Cannon, Science Applications, Inc. (M)

Q: When my line printer (Versatec) runs out of paper or the device goes not ready, LPP0 seems to grab my CPU and run in a tight loop at priority 70. Why?
A: There is a bug in the line printer despooler. The line printer despooler, instead of waiting on an event flag for the last buffer, loops for I/O completion. Therefore, if the printer runs out of paper while printing the last record of a file, LPP0 will loop. There will be an SPR published for RSX-11M V3.2. The bug is fixed in version 4.0.

John Vilandre, University of Minnesota (M+)

Q: What changes have been made to the queue manager in RSX-11M Plus. Specifically, do files which have been explicitly requested NOT to be printed still go into queue on hold? Examples are:

```
SUBMIT/NOPRINT
PRINT/PAGE:2
```

A: SUB/NOPRINT will not produce a log file at all on version 2.0 [M+].

Tom Zak, Illinois Bell Telephone Co. (S)

Q: I have two duplicate RSX-11S systems on which I am receiving 'EXEC PARITY ERRORS' whenever I open any location within a 60 (octal) byte range in upper memory. I've had memory and memory management checked with diagnostics with no problems shown. Can software in some way be responsible for these errors?
A: 1) Check power supply 2) Check programmable clock 3) You might have two devices responding to the same UNIBUS address. 4) You might have a problem with sizing-memory algorithm in tertiary bootstrap in DECnet.

Al Tywill, Digital Consulting (M)

Q: My system is an 11/60. What steps can I, as a system manager, take to speed up the task builder?
A: You can; separate ACPs, [build] file on less busy disk, change file buffer size, unfold overlays, etc. [ED: Next months Multi-Tasker will have an article on how to make TKB faster.]

Jim Preciado, Advanced Tech Systems (M)

Q: SPR responses concerning user written drivers are unacceptable. What can be done? In my case, the error logger was destroying my user-written device driver data base, although the device does not use error logging.
A: See developer in RSX SIG campground.

Roger Jenkins, Wycliffe Bible Translators (M)

Q: Will RSX-11M V3.2 programs have to be reTKBed when we move to version 4.0?
A: Non-privileged tasks will not rebuilt, privilege tasks must.

Michael Antin, Polaroid Corp. (M+)

Q: BRU has had a problem with large files. Has this problem been solved? And, can we expect a solution before release date of (M+) V2.0 [May '82] ? Or, at least with the release of V2.0?
A: The problems with BRU should be fixed by software dispatches. Some dispatches have only been listed in RSX-11M and not in RSX-11M Plus. All RSX-11M patches should also be applyed to RSX-11M Plus. The versions of BRU are identical for both systems.

John E. Crider, Shell Oil Co. (M)

Q: RSX-11M V4.0 is said to include the use of the machine instructions MUL and DIV within the Executive on those processors that have those instructions. Is this also true for the machine instructions SOB, MTPS, and MFPS? These are now (in version 3.2 for almost all processors) redefined as macros in file RSXMC.MAC.
A: Yes, for EIS instructions (including SOB); but, no for MTPS, MFPS. Digital says speed improvement should not be significant, the primary use is in disk drivers caculating the track, sector, cylinder from the logical block number.

From the Editor

This part of the session was transcribed by Bob Turkelson of NASA/Goddard Space Flight Center, Greenbelt, Maryland.

Larry Baker, U.S. Geological Survey (M/M+)

Q: A FORTRAN OPEN statement can specify both a RECORDSIZE and a BLOCKSIZE for doing I/O to either a tape or a disk file. When PIP transfers a file from disk to tape, it places the blocksize specified for the transfer in the recordsize field - it does not pick up the FCS maximum record size to write into the tape label field. A processing program is not able to read a file which was originally written to disk and then transferred to tape by PIP, unless the program has a record buffer size which is as big as the block size. This is not the case if the original program had written the file directly to the tape. Is PIP going to be fixed so that it properly writes the FCS maximum record size field into the record size field when it creates ANSI tapes?
A: When PIP copies from disk to ANSI mag tape, it does so a record at a time. PIP uses the same code to copy to tape from either a disk file or a terminal, so at file creation time it does not know the maximum record size. PIP will not be modified to handle a disk input file as a special case in the manner desired. Also, FORTRAN OTS developers made the decision that when a FORTRAN file is opened for input, an error will be returned if the record buffer is not big enough for the largest record in the file (disk or tape). Some time in the future we want the OTS to allow the OPEN to occur and to return an error only when reading a particular record which is too large.

Rodger S. Miles, TELEMED (M)

Q: I have been requesting a patch for COT to keep it from rejecting a QIO. This is very important because the problem keeps us from going into production with one of our systems. This problem is reported to be fixed in Version 4.0. Can this feature be made to work in Version 3.2?
A: In version 3.2 when a QIO is sent to the driver, it copies the data into an internal buffer and then notifies the COT task which gets it out. When the buffer fills up, further I/O is rejected. We rewrote the driver for version 4.0 so that it manages its own queue. When there is a lot of I/O, there will be many I/O packets, but no rejected QIO's. This change is too significant to issue a patch for the vesrion 3.2 driver.

Tony Scandora, Science Applications Inc. (M)

Q: Will RMS-11K and Digital Standard Runoff (DSR) be bundled with RSX-11M Version 4.0?
A: RMS-11K will be bundled with RSX-11M Version 4.0 when Version 2 of RMS-11K is available - but not with the initial release of Version 4.0. It will also be available as an update to RSX-11M for all customers in warranty or on subscription. DSR is not available for RSX-11M.

James K. Neeland, Hughes Research Labs (M)

Q: Are we allowed to publish all previously unpublished SPR responses in the Multi-Tasker? Or will Digital now publish previously unpublished patches which meet the new patch publication criteria?
A: Digital will not publish them, but we may do so. [ED: The discussion

following lead to a call for all SPR's submitted on RSX-11M V3.2 to be sent to the Multi-Tasker, see the editorial in the last issue]

Jerry B. Williams, Computer Science and Applications, Inc. (M)

Q: Is there a way to make the full duplex terminal driver on RSX-11M Version 3.2 do DMA output to a DH-11 without copying the user buffer to the terminal driver buffer before starting the DMA transfer? I would like to DMA out to the DH-11 straight from my user buffer.
A: No, there would be a problem allocating UMR's with 22-bit addressing on the 11/70. The terminal driver also performs other processing, such as expansion of tabs. For a write-pass-all on 18-bit systems, some fix may be possible.

Mark King, Colorado Interstate Gas Co. (M)

Q: Under RSX-11M Version 3.1 you could have multiuser MCR without selecting multiuser protection. Digital did not make this a valid combination during a Version 3.2 SYSGEN. Will this option be available in Version 4.0?
A: You may try to implement this combination by editing RSXMC.MAC to comment out the definition of M$$MUP during SYSGEN before assembling the executive. The resulting system would be unsupported in 4.0 - we are not sure it would work.

Gary Nelson, Logicon, Inc. (M)

Q: Under RSX-11M on an 11/44, a cache parity error resulted in a system halt with EXEC PARITY ERROR STOP. How does RSX-11M handle memory parity errors versus cache parity errors?
A: The operating system should disable cache upon a burst of cache errors. The EXEC PARITY ERROR STOP results from a fatal memory error while the system is in kernel mode. A cache error should not cause this. Submit an SPR.

Randy Girard, Caterpillar Tractor Co. (M+)

Q: What was the logic behind not supporting the RM80 and RP07 devices on the 11/70 running RSX-11M-PLUS?
A: RM80's and RP07's are in short supply - there are not enough to supply the demand for them on the VAX 32-bit processors. This was a management decision to try to minimize the number of frustrated customers.

Bruce R. Mitchell, 3M Company (M/M+)

Q: Global deletes of the form [*,*] and/or *.*;* are extremely dangerous. PIP should request the user to verify that he really wants to do this operation. A switch could be used to override verification for use in indirect command files [Appluase].
A: This is not supported in Version 4.0. [The audience pointed out that RT-11 and the PDP-8 operating system support this feature.]

Michael Antin, Polaroid Corporation (M+)

Q: The online formatter does not work with the /BAD switch for our RM03 disk. When BAD starts it returns with Device Not Ready. Why? [Other members of the audience reported seeing the same problem or similar circumstances on other disk drives.]
A: Submit an SPR.

From the Editor

This part of the session was transcribed by Dan McCoy, Informatics Inc., Palo Alto, California.

David Birkenmeyer, Clark Equipment (M+)

Q: Will there be a way of using the new cluster library mechanism to implement cluster commons? My system has many more resident commons than resident libraries.
A: No.

Stan Dicken, Conoco Inc. (M)

Q: What are the major and minor known problems with Autopatch E? Will there be any more Autopatch tapes for RSX-11M V3.2?
A: Will publish problems with Autopatch version E, possibly in the Software Dispatch. There may possibly be one more Autopatch for Version 3.2. [ED: See article in this issue on the rumored problems with Autopatch E].

Rod Pettinato, Corroon and Black (M+)

Q: Are the layered product people who market COBOL, RMS, FORTRAN, DATATRIEVE, etc. going to make use of I/D space and/or supervisor mode libraries for the RMS library, COBOL/FORTRAN OTS, or DATATRIEVE data structures? If not, why, and what kind of influence does the M+ group have.
A: We are right now in the process of twisting arms to get layered products to support cluster libraries. As of the day that V4.0 ships, the only library we will have tested is FCS and RMS V2.0. We are also working on getting more RSX-11M Plus support into layered products.

Pam Vavra, KMS Fusion, Inc. (M)

Q: What are the plans for RGL (REGIS Graphics Library) availability? I want to put VT125's on an RSX-11M system and want RGL. Moreover, the VMS price for the RGL is outrageous.
A: It is available under the MINC system upgrade to RSX. No known plans to support it. (A head count showed near 50% interested.)

Leo F. Hinterlang, Technicare Inc. (M)

Q: Are there any plans to introduce any new PDP-11's with additional machine instructions to enhance performance of DEC's operating systems? Such as save/restore registers, block moves, context switching, link/unlink, etc. In general, do the software groups talk to the hardware people?
A: Not in current plans. Software groups do have inputs to hardware groups.

Ralph Stamerjohn, Monsanto (M)

Q: Are SPR patches covered by Digital licensing (i.e. copyrighted) or can users exchange patches?
A: Legal issues are unsure.

James Demange, Systeme Corporation (M)

Q: What standard utilities use the Get Partition, Get Task and Get Lun Characteristics directives?
A: They are needed in various utilities. The utilities use the directives to find out how much space is available to them for to use as dynamic space. One utility which is known to use them is TKB. The reason for the options is because RSX-11M was intended, and still can be, to be generated as a very lean, run-time system.

Louis Stoll, Zia Company (M)

Q: Are there any plans to add batch queues to RSX-11M?
A: Not planned for next release. Will keep it open, but doesn't look possible.

John Vilandre, University of Minnesota (M,M+)

Q: Has anything been, or will anything be done to allow BRU/APPEND to a multi-volume backup set on tape? Often 2000 feet are wasted when the set won't quite fit on one tape.
A: Nobody knew for sure, but panel believed answer was no.

Philip Cannon, Science Applications, Inc. (M)

Q: I ran a BRU (autopatch E) backup on a large disk that took two reels of tape. I used the verify switch and the verify failed on the second reel. I then ran compare and it said the tape was good. Is the tape really good?
A: The maintainer is looking at this problem. The consensus of the DEC panel was that the patch will appear in the Software Dispatch soon. Some in the audience said they already have unpublished SPR's that correct the problem.

(A heated discussion ensued concerning DEC's policy on which SPR's get published and which don't. The general consensus of the audience was that all SPR's that can be published, should be, since the users spend time and money to isolate bugs for which fixes are known.)

From the Editor

This part of the session was transcribed by Bob Mearns, NASA/Ames Moffit AFB, California.

C. H. Stockley, Sandia National Labs (M)

Q: BRU with /VER fails on start of last reel of 2 reel set. Also, BRU with /COM fails on start of last reel of 5 reel set.
A: Patch coming.

Kreigh Tomaszewski, Amway (M+)

Q: When a user on a disk other than the system disk submits a batch job, the log file goes to [1,7] on the system disk (unless the user's UIC also exists on the system disk). Will the batch system be changed to return log files to the user's UIC?
A: Yes, fixed in version 2.

Roger Jenkins, Wycliffe Bible Translators (M)

Q: Is there a way to determine that RSX has shut down a bad cache memory, given that when the system does so, it does so on boot, before error logging has been started?
A: Good point. We will look into it.

Frank J. DeLalla, US Department of the Air Force (M)

Q: Tasks are installed, run, and left installed. All goes OK. Some time later, 12-24 hours typically, an attempt is made to run the same tasks. Either they abort immediately, or crash the system. Note that these tasks are installed from an RK05, we run version 3.1, and that this problem does NOT occur with tasks installed by VMR (PIP, TKB, etc).
A: Could be a problem with disk alignment changing over time. Disks should be checked out. No definite answer.

Gary Nelson, Logicon (M)

Q: How many files can I put on an RP06? BRU let me specify a maximum file size of 50,000. Am I going to have problems?
A: 24,000 files maximum, because of single header index file. Under version 4, this will be increased to 65,000, by using multiheader index file and bitmap. Check you disk to find out what BRU did.

David Springhetti, MPI (M)

Q: BRU failed to move all of my 2400+ files from UIC [6,10] to disk. It moved about 1250 files, then quit. This was the only UIC I was backing up.
A: SPR the problem.

George Hamma, Synergistic Technology (M)

Q: Building an overlayed task compiled by FOR. When I used BIGTKB to build it, using an indirect command file, the resulting task runs fine. Using same indirect command file and same input with SLOTKB, the resulting task aborts with error 2 (initialization failure).
A: None.

Art Kocsis, McDonnell Douglas (M)

Q: I want limited source licenses, i.e., for selected utilities such as print spooler.
A: We are still looking into it.

Alan Frisbie, Flying Disk Systems (M,M+)

Q: Assuming the terminal is set /BUF=TI:132., the PIP command A.*,B.*,C.*/FU:80. will result in all lines but the last being 132 characters wide. The last is 80.
A: Supposed to be fixed in version 4. However, a check of the demonstration system showed the problem still happened.

Bruce R. Mitchell, 3M (M,M+)

Q: RSTS supports an option which allows forced logout of terminals after an idle period of 5 minutes. In RSX, this would greatly improve system security, particularly for privileged terminals.
A: DEC has looked into the matter and received mixed response. There is a (remote) possibility of having this as a SYSGEN option in a future release. (See TT.KGB on SIG tape).

From the Editor

This part of the session was transcribed by Mike Drabicky, Rockwell International, Richardson, Texas.

Larry Ward, AT&T (M)

Q: Some of our systems (3.1) have hung with open RMS files. When I try to CNV or DMP these files, CNV and DMP hang after dumping out a few blocks. What's the problem, and is it hardware, software or both? DMP and CNV are in a wait-for state (TAL shows WFR). Nothing showed up in the error log.
A: No known answer.

Hugh Kennedy, Oakweald, Ltd. (M)

Q: Please could you prvide a linkage between ODT (and your new symbolic debugger) and the overlay system. I know about $ALBP1 and $ALBP2. Why can't ODT have a command to place the two breakpoints and remove them.

A: Confer in campground.

Ralph Stamerjohn, Monsanto (M)

Q: The current pricing for a VS11 controller and memory (color graphics system) is less than the ML11 (non-rotating electronic disk). Because the VS11 is a DMA device with block addressable memory, will RSX-11M support the VS11 with a disk so we can checkpoint in color? Also if you have the display and checkpoint to the VS11, you could use the light pen to selected who comes back into memory.
A: Laughter.

Pam Vavra, KMS Fusion (M)

Q: I intend to use VAX FORTRAN for developing programs for use on PDP 11/45 that must undergo revisions using RSX FORTRAN 77 (which is wonderful, thanks). I would like a switch on VAX FORTRAN to limit me to the F77 subset (like NOF77 on RSX FORTRAN).
A: Use F77 under AME and look at error messages at cost of symbolic debug facility of VAX FORTRAN. Will pass request to appropriate people.

Larry Baker, U.S. Geological Survey (M)

Q: I was surprised at the number of users of 22-bit machines still running RSX-11M. It would relieve a lot of pressure to make RSX-11M the kind of development machine RSX-11M Plus has become if you would make it VERY attractive for RSX-11M users to migrate to RSX-11M Plus -- ala nearly free. Then users of 18-bit machines that should migrate could use the dollars towards hardware for an upgrade as well. Why the big hurdle for M to M+ migration.
A: Marketing decision are made by the product lines. Go beat on your salesperson.

C. H. Stockley, Sandia National Lab (M)

C: The Autopatch-E file HELLO.COR (IDENT 3.1 2.2.1.14 M OCT-80) is not in the module in MCR.OLB file supplied.

Philip Cannon, Science Applications, Inc. (M)

Q: How many patches does DEC know about that are not published that would be published under the policy that was announced tonight? I think I have "discovered" 2 on my own.

Addressed to the audience: I didn't pay $200,000 for my system and $300/month for support so that DEC can hide solutions to my problems. Will the members of the SIG please copy and send all of their SPR's to the Multi-Tasker? The SIG will sort out the unpublished ones and address the remaining ones in the Multi-Tasker. At best, the SIG can "rediscover" the bugs and get them published under the current guidelines.

A: [Editor's note: General approval from the audience although no specific answer was given.]

Kreigh Tomaszewski, Amway Corporation (M+)

Q: When a fatal device check occurs on an RM05, the volume invalid indicator is set blocking all access to the disk until it is dismounted and remounted. However, error recovery issues a "drive clear" following this condition that resets all hardware error indicators making it very difficult for the service people to isolate the problem. Can error recovery be fixed (so that I can improve my uptime) to not issue the clear which wipes the necessary diagnostic information in this fatal situation?
A: No answer.

Mark Goldsmith, Johnson Controls, Inc. (M)

Q: BRU doesn't support some of DEC's tape drives. Specifically, the TS11 or MS:. We can read from tape but cannot do any BRU with a verify that works (on a 11/34 M3.2). Do you know if and when BRU will support MS:?
A: Should have been fixed in Rev E.

Joseph Sventek, Lawrence Berkeley Lab (M)

Q: Limitation of autobaud to 1200 baud eliminates its use for terminal switches.
A: TTDRV does autobaud to 9600 baud.

From the Editor

This part of the session was transcribed by Louis Stoll of Zia Company, Los Alamos, New Mexico.

Art Kocsis, McDonnell Douglas (M)

Q: When will the undocumented incompatibilities between RSX and VAX DECnet, (such as blank fill vs. zero fill in node names), be documented.
A: We, (the RSX group) just got responsibility and will publish a list.

Robert Bismuth, Redkite Software LTD. (M,M+)

Q: Can I use the reserved word in the file I.D. (third word) for status bits in adversity? This word is also stored in directories and would be useful to use.
A: Playing with the reserved word interferes with the F11ACP's maintenance of the on disc directory structure.

Bruce R. Mitchell, 3M (M,M+)

Q: Many sites have expensive collections of RSX, often to the point of one set of the system manuals per user. Will documentation sets be made available to update V3.2 to V4.0 for RSX-11M, and V1.0 to V2.0 for RSX-11M Plus?
A: The only manuals which have not been changed are RMS and ODT. An update is only possible for system library routines manual. While it might be cheaper to produce manual updates, it would entail a prohibitive amount of work for Digital.

Bill Hay, Dataline, Inc. (M)

Q: Can a standalone, I/D separated, image be built with new TKB under RSX-11M, or RSX-11M/PLUS?
A: Yes it can but no operating system support. Discussion from floor was that it can be done and made into an executable task by using VMR after task build to create a bootable copy. However, a loader for the stand-alone system would have to setup the I/D memory management registers, i.e., the starting address could not being using both I/D space because the memory management unit is not initialized.

Ralph Stamerjohn, Monsanto (M)

Q: We have a DECsysten-10 in our DECnet network, and therefore, cannot use DECnet Phase III. Who do we contact to find out how to get DECnet Phase II to run under RSX-11M V4.0?
A: The DECnet development group.

Alan Frisbie, Flying Disk Systems (M)

C: I submitted an SPR on BASIC-11 in October 1979 and followed up several months later with calls and letters. In October 1980, I sent the SPR a birthday card. The reply, (and fix), was received in early 1981. Naturally, it still hasn't been published.

Jim Neeland, Hughes Research Labs. (M)

Q: Patch to MT: driver to not make data overruns fatal. It now doesn't retry. Condition also true for parity errors. Is this an error or a feature.
A: Fixed, SPR unpublished.

Wade Scannel, Athena Systems, Inc. (M)

Q: Our IND <SYSTEM> variable returns a value of '6', indicating the system is an M-PLUS system. We have included stop bit support as described in the sysgen manual.
A: The wrong conditional may have been uncommented in the patch.

Joseph Sventek, Lawrence Berkley Lab (M)

Q: A perennial problem with ODS-1 & BRU is the replication of files with multiple directory entries. Would it be possible to have a field in the file header to keep track of the number of links to the file & make BRU cognizant of it?
A: No intention of doing it, will put it on the list.

Art Kocsis, McDonnell Douglas (M)

Q: Would it be possible/feasible to display the volume label on a PIP directory listing? Also to put date, UIC, & label on one line to get more file names on a 24 line CRT?
A: Noted as a suggestion. Violates volume security philosophy.

Pieter Botman, Bell Canada (M)

Q: Any plans for an RSX-11M Plus driver for a DT07 unibus switch, and how difficult would you see it being, to convert the RSX-11M driver to RSX-11M Plus.
A: RSX-11M driver for DT07 from CSS is out now. RSX-11M Plus driver for DT07 from RSX group through CSS. The RSX-11M Plus Guide to Writing an I/O Driver manual has an appendix on converting RSX-11M drivers. A suggestion was made to include this information in the RSX-11M manual so RSX-11M users writing new drivers could build in prior knowledge of RSX-11M Plus.

Gregory Moulton, New England Medical Center (M)

Q: Will FLX be modified to read RT-11 magtape.
A: No. RT-11 write ANSI tapes that can be written with RSX-11M V4.0 ANSI PIP. May still have problem with RT-11 stream ASCII files.

Bill Hay, Dataline Inc. (M)

Q: Has anyone had any problems with the AST processing initiated by data ready, (IO.ATA) on V2.0 of the 3271 PE?
A: No one.

Bruce R. Mitchell, 3M Co., Engr Systems (S)

Q: Often it is desirable to collect data on an RSX-11S system and analyze it on a larger system, later. For those without the benefit of DECnet, will RSX-11S optionally support a simple file system, (possibly RT), on TU58 cartridge?
A: DEC has examined this and might have implemented it, save there were manpower problems. (Mike Lynch, 3M) It is also necessary to ensure line integrity when a line goes down. DEC will examine this and possibly implement a limited Files-11 in the next release of 11S.

Vickie Griffith, W.R. Grace, AG. Chem. (M)

Q: How can I change 'AT.' to run with the same UCB of the spawning task.
A: Fall '81 SIG tape RECUR program. Also, set console logging and log on CO:. This allows spawning 'AT.' directly.

RSX-11M V4.0 FIELD TEST REPORT

Margaret H. Knox

Uinversity of Texas Computation Center
Austin, Texas

The University of Texas at Austin Computation Center participated in the RSX-11M version 4.0 field test and I presented our results at the Fall 1981 DECUS Symposium Field Test Panel. Here are some brief notes from my portion of that panel:

* Overall, the field test went very smoothly, very few problems.

* SYSGEN now includes answer files for the first two sections and that similifies doing resysgens.

* Our system generation was performed by a student operator, Thomas, who is without previous SYSGEN experience and was done while the programming staff was at a conference. Yes, he got it to work (98%) just by following the instructions.

* Auto-configure was not a big help for our site because it does not include Digital's more "exotic" equipment such as K-series modules.

* Thomas did fall for the old trap of not believing SYSGEN's proposed CSR for the RK05, our only device where the CSR is not the first word. I believe the release kit will have improved documentation.

* Thomas crashed his SYSGEN once. He software booted the distribution kit from our V3.2 system. Sometime later someone hit <return> on one of our DZ11 lines, causing a crash. A software boot does not reset the DZ11's and the baseline systems vector address to not cover the DZ11 vector. This has always been true, so do a hardware boot of the basline system to reset all CSR's, rather than a software boot from a running V3.2 system.

* We essentially select all options except XDT, and this time also asked for DCL and alternate CLI support. Our system grew approximately 2.5K words over the V3.2 size. It is no longer possible for us to have a 16K executive and we will either choose a 20K executive or executive common blocks to fix this problem. The 2.5K words is general growth. I think DCL and alternate CLI support accounted for less than 1K words.

* DCL felt a little awkard at first to use. You sit at your terminal knowing what the correct MCR command is and try to imagine the DCL command. There are help files for DCL and of course, you can execute an MCR command from DCL (and vice-versa). We are most fond of DIR, TYPE, DEL, and PRINT.

* DCL is a little slower than MCR since it is a translator. It seemed to bog down if you forked and gave it lots of commands from one terminal.

* MCR supports a DCL command and DCL supports an MCR command. However MCR does not support a MCR command, i.e. it will not accept a command such as:

```
>MCR PIP /LI
```

  I found this unfortunate because we supply a large number of command files to our users and would prefer to always prefix the commands with the correct CLI name rather than changing the CLI on the unsuspecting user.

* The queue manager will now automatically detach form the printer, if you have set it up that way. For printer/plotter users of equipment such as Versatecs, this simplifies running your plot jobs which do not use the queue manager.

* Unlabeled magtapes are now supported. From Fortran you can use standard READ and WRITE statements to access unlabeled fixed block tapes. PIP can do this also. You can specify the blocking factor and translation. ASCII and EBCDIC are available and at SYSGEN, you can add other byte oriented translation tables of your own. You can also position an unlabeled magtape by referring to it with a peculiar name:

```
>PIP MM0:"POS=R"
```

  rewinds the tape!

* We have always run a multi-user, non-protected system. In RSX-11M V3.1, this was an option. In V3.2, there was an easy hack to get around the restriction (November 1981 Multi-Tasker). This hack will not work in V4.0 if you want DCL. What did work was to modify SYSGEN (SGNEXEC.CMD) to ignore the M$$MUP symbol at system assembly time. You tell SYSGEN you want multi-user protection, which allows you to get DCL, but then build an unprotected executive. Only two problems are known with this hack: the CLI symbol in Indirect MCR always responds MCR and ERRLOG did not build successfully.

In summary, RSX-11M V4.0 was remarkably solid. It seemed more like a release than a field test!

REDUCING THE SIZE OF A FORTRAN PROGRAM

Larry Baker

U.S. Geological Survey
Menlo Park, California

1.0 ABSTRACT

This paper gives programming suggestions for reducing the size of Fortran IV-Plus task images. With the exception of linking to shared system libraries, all of these suggestions will reduce the amount of virtual address space required by your programs, which will help you shoe-horn them into the address space limitations of PDP-11's. All of the suggestions will reduce the amount of physical memory your programs require, so there will be more room in your total system for more tasks, before the system performance becomes degraded due to checkpointing activity.

Familiarity is assumed with the PDP-11 Fortran IV-Plus language, the Fortran IV-Plus compiler and Object Time System (OTS), and RSX-11M/M-Plus program development procedures. Putting these hints to work can result in substantial memory savings with relatively little effort.

NOTE

The information pertaining to buffer size reduction and shared I/O libraries applies to the FCS version of Fortran IV-Plus OTS only. Some information in this paper has appeared in previous editions of the IAS/RSX Fortran IV-Plus User's Guide.

2.0 REDUCING STATEMENT NUMBER STORAGE

The Fortran IV-Plus compiler supports four levels on run-time error traceback:

1. Full traceback of every Fortran statement (/TR:ALL, /TR:LINES, or /TR).
2. Traceback by groups of statements, or blocks (/TR:BLOCKS).

3. Traceback by entry name only (/TR:NAMES).

4. No traceback information (/TR:NONE, /NOTR, or /-TR).

The amount of overhead (execution time and storage) decreases as you go down the list of options. In general, /TR:BLOCKS (the default as the compiler is shipped from Digital) is sufficient for debugged code, while /TR:ALL (the default on our system) is most useful during program development. Check with you own system manager if you are not sure of your own system defaults.

To illustrate the effects of the various options, I have compiled all the routines in one of our larger subroutine libraries and a magtape utility using the four different options. The sizes for the subroutines were taken from the size summary at the end of the compiler listings (see Figures 1 and 2). The sizes for the magtape utility in the table below were taken from the task builder map, and more properly reflect the total effect of the different options on task size, since there may be differences in the Fortran OTS support required for the different levels of tracing.

/TR:ALL	26430 words	7.1% larger
/TR:BLOCKS	25470 words	3.2% larger
/TR:NAMES	24928 words	1.0% larger
/TR:NONE	24676 words	base value

Those of you that use structured Fortran pre-processors, such as Ratfor, may not be aware of the penality you pay in additional storage for statement traceback information, even if you use the /TR:BLOCKS option. Any potential branch point starts a new block, and the Ratfor-type preprocessors tend to generate lots of extra labels and CONTINUE statements. According to Joe Sventek at LBL, he observed a reduction of roughly 30% in the storage required for code segments in the LBL Software Tools by changing the compiler option from /TR:BLOCKS to /TR:NONE.

3.0 USING THE SHORT ERROR MESSAGE FILE

Fortran IV-Plus provides two error messaeg modules: one containing the full text of the error messages, and one containing on the error numbers, whose descriptions may be found in the User's Guide, section C.3.2. The is built with the full error message module by default. The full message module occupies 1128 words of memory that can be recovered by requesting the short message module.

To include the short module in you task image, enter the module name $SHORT as shown in the following sample Task Builder command string:

```
TKB>TRANS,TRANS/-SP=TRANS,LB:[1,1]SYSLIB/LB:$SHORT
```

where LB:[1,1]SYSLIB is the file specification for the default system library containing the Fortran IV-Plus OTS. The installation guide for Fortran IV-Plus documents how to install the short error message module as the default module for the system.

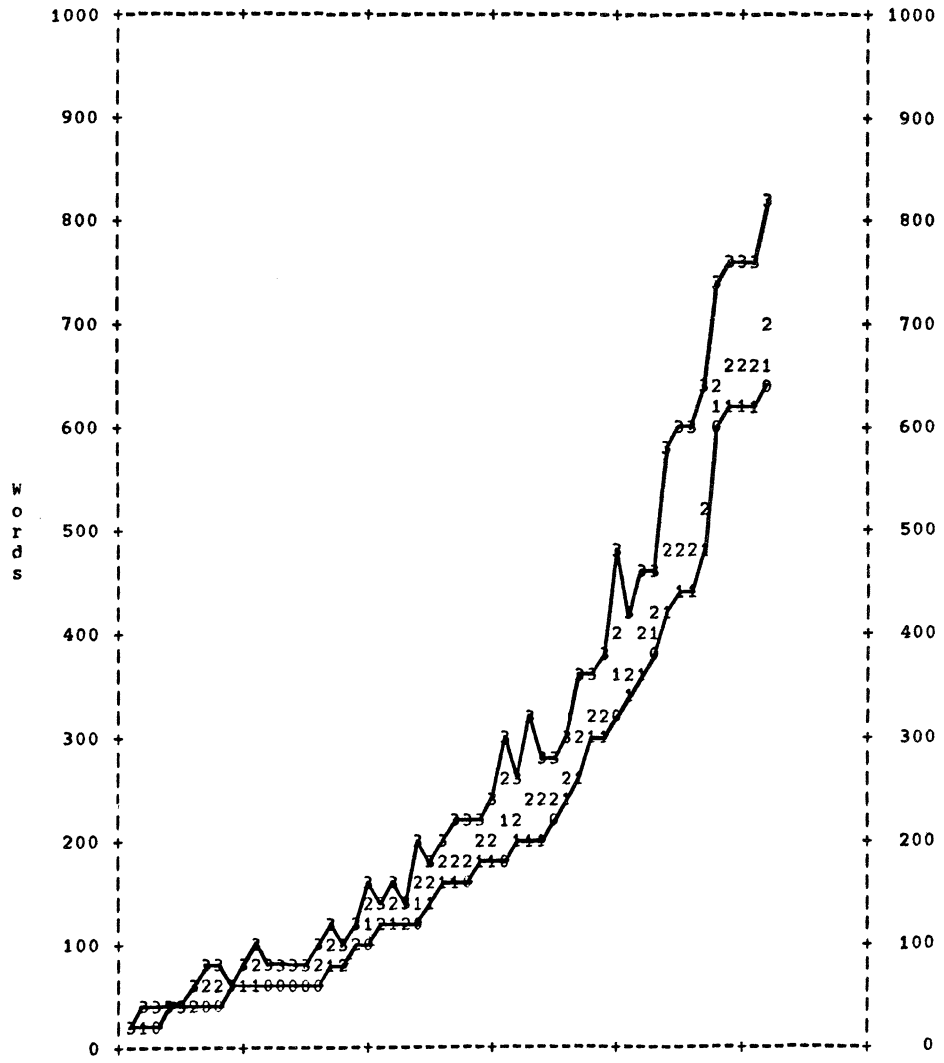

Figure 1. Module size, in words, as a function of tracing level: 0=/TR:NONE, 1=/TR:NAMES, 2=/TR:BLOCKS, 3=/TR:ALL.

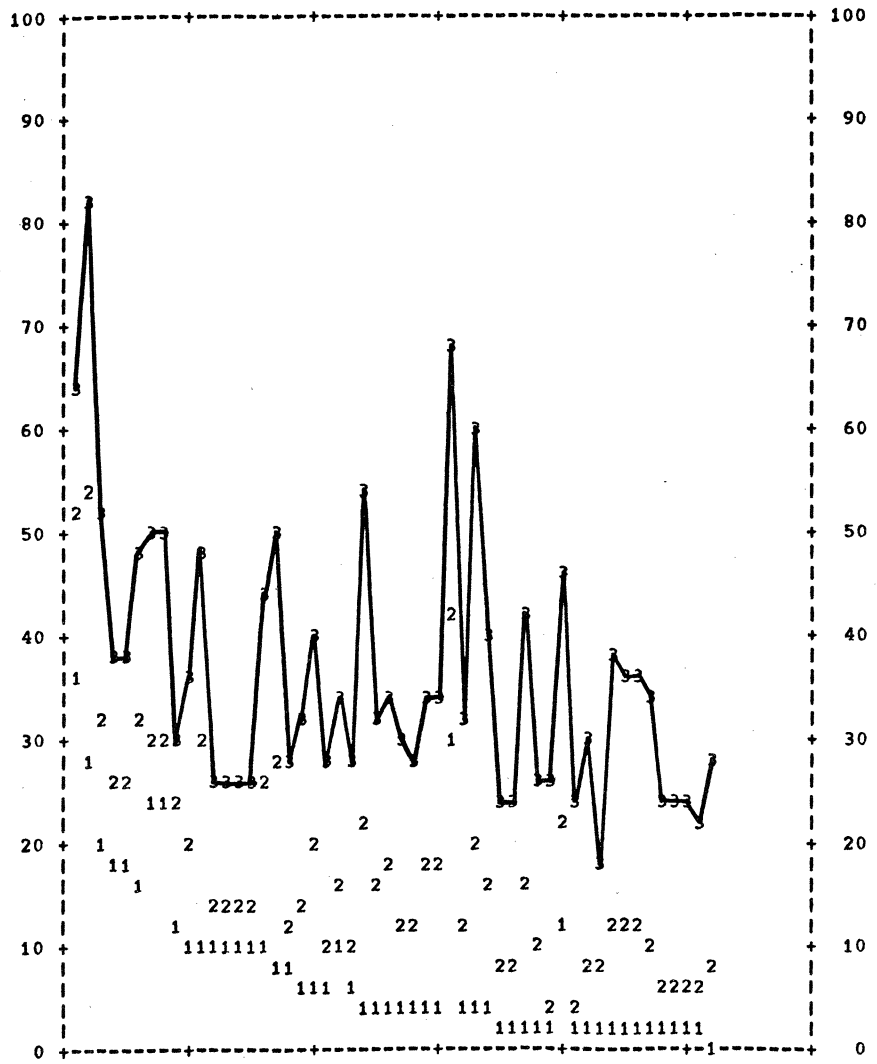

Figure 2. Module size, in percent over /TR:NONE, as a function of tracing level: 1=/TR:NAMES, 2=/TR:BLOCKS, 3=/TR:ALL.

4.0 REDUCING THE NUMBER OF LOGICAL UNITS AND OPEN FILES

By default, the Task Builder assigns six logical units and four open files to a Fortran program, regardless of the number the program actually uses. Every unit number requires 56 words of storage and every active file requires 264 words of storage. The Task Builder options UNITS= and ACTFIL= can significantly reduce the task size if you specify the actual number of logical units (UNITS=) and simultaeously open (or active) files (ACTFIL=). Of course, this only works if the actual requirements are less than the default assumptions.

You must provide enough unit numbers as the highest unit number your program uses, not the simultaneous unit numbers in use. Requesting a smaller number of units than the largest unit in the program causes an error at execution time. So it is good programming practice on a RSX system to use small, sequential values for your program unit numbers.

To use these options, enter the keyword followed by the decimal number of units or active files. For example, the TKB commands:

```
ENTER OPTIONS:
TKB>UNITS=3
TKB>ACTFIL=3
```

request three logical units and three active files. Note, however, that the Task Builder default assignments still apply, i.e., units 1-4 default to SY0:, unit 5 to TI0:, and unit 6 to CL0:. There are no default assignments for units 7 and up. You must use the Task Builder ASG= option to override the default assignment or else specify the device explicitly in your program at run time.

5.0 RECUCING FCS BUFFER SIZE

The task builder allocates a 528 byte FCS buffer area (16 bytes for the buffer header and 512 bytes for the buffer) by default for each simutaneously open file (ACTFIL=). This is sufficient to contain one disk block or one default size block from an ANSI magtape. However, it is almost four times the size required for terminal and direct lineprinter I/O (allowing for a 136 byte buffer size). The amount of memory occupied by this default buffer can be reduced by specifying an explicit size for the FCS block buffer with the EXTSCT= option.

For example, if your program has three simultaneously active files, the Task Builder buffer allocation is 1584 bytes (3 x 528 or 3060 octal). If one of these units is connected to a terminal, the allocation may be reduced to 1508 bytes ((3x16) + (2x512) + 132 or 2270 octal). To use the option, enter the command EXTSCT=$$FSR1:size, where size is the buffer size required by your program, in octal, as shown below for the above example:

```
ENTER OPTIONS:
TKB>EXTSCT=$$FCS1:2270
```

The following table contains the appropriate octal values for use with the EXTSCT=$$FSR1:size command where at least one of the program's units is

connected to a terminal. Note, you lose a certain amount of device independence with this feature. If you shorten $$FCS1 to assume one open file is a terminal, you cannot suddenly use a disk file instead without rebuilding the program.

```
ACTFIL  EXTSCT=$$FSR1:size

  1              230
  2             1250
  3             2270
  4             3310
  5             4330
  6             5350
  7             6370
```

6.0 REDUCING FORTRAN IV-PLUS OTS SUPPORT CODE

The Fortran IV-Plus OTS code for certain I/O operations is included in your task image only if your program makes use of the language facilities. Programs that use TYPE, ACCEPT, or list-directed I/O statements incur additional space overhead for the OTS modules that process them. If your program already uses formatted READ and WRITE statements, space may be saved by converting these alternate forms to the standard form of Fortran formatted I/O. List-directed I/O is most useful for Fortran debugging statements used to print quick diagnostics and values, especially were you do not want to struggle with format statements. Confining such statements to debugging statements, i.e., statements with a "D" in column 1, eliminates their overhead when the compiler's debugging option (/DE) is turned off.

7.0 USING THE NON-ANSI FCS OTS MODULES

By default, the Task Builder searches for unreolved references from the default system library, LB:[1,1]SYSLIB.OLB. If your system supports ANSI magtapes, chances are that the system manager put the ANSI version of FCS routines into this library to allow programs to access ANSI magtape files as well as disk files for sequential I/O operations. However, the FCS ANSI routines occupy more space that the equivalent non-ANSI versions. If your program will never require magtape files, you can save space by linking to the non-ANSI system library. Check with your system manager to find out if your system library has the ANSI FCS modules and what alternate library to use if you do not want this support.

To inform the Task Builder to use a different library as your default system library, specify the /DL switch in your Task Build command string. For example, assume LB:[1,1]SYSLIB.NMT is the version without ANSI FCS modules:

```
TKB>TRANS,TRANS/-SP=TRANS,LB:[1,1]SYSLIB.NMT/DL
```


8.0 REDUCING STACK SIZE

By default, the Task Builder includes a 256 word stack for your program to use at run time for subroutine linkage and temporary storage. The STACK= option may be used to reduce the stack size. For example, the following sets the stack size to 128 words:

```
ENTER OPTIONS:
TKB>STACK=128
```

It is extremely difficult to recognize when the stack size is too small. This option is only recommended for those cases when you need a small amount of space to make a program fit into memory. Some Fortran programs will run with as little as 64 words of stack with no problems. However, at least 128 words is required if any OPEN statements are used. The usual symptom of the stack size being too small is a failure in OPEN statement processing that goes away as soon as the stack size in increased.

9.0 ELIMINATING REDUNDANT FCS CODE

Programs that perform bot sequential and random I/O operations cause the Task Builder to load both the FCS sequential I/O routines (GETSQ and/or PUTSQ) and the random I/O routines (GET and/or PUT). The random routines, although longer than their sequential counterparts, contain all of the features of the sequential routines and can, in fact, process sequential files. Thus any program performing both types of I/O contains functionally redundant code. The GETSQ module use 286 words and the PUTSQ module uses 399 words.

To exclude the sequential routines, insert the following module Macro-11 module, JUMPER.MAC, in the root segment of you task. This module eliminates the unnecessary sequential I/O routines by simply passing control from the entry points .GETSQ/.PUTSQ to .GET/.PUT.

```
        .TITLE  JUMPER
        .PSECT  JUMPER,RO,I

.GETSQ::JMP     .GET            ;Pass control to .GET
.PUTSQ::JMP     .PUT            ;Pass control to .PUT

        .END
```

The following commands will assemble JUMPER and include it as a part of your task.

```
>MAC JUMPER,JUMPER/-SP=JUMPER
>TKB TRANS,TRANS/-SP=TRANS,JUMPER
```

10.0 USING THE TASK BUILDER OVERLAY FEATURE

The Task Builder's overlay feature can further reduce the task size by segmenting and overlaying portions of your program that are mutually independent of each other, such as initialization (once-only code), input routines, calculations, and output routines.

The minimum size for a particular task can only be attained though careful analysis of the build maps, planning, trial, and error. However, many programs go though a typical sequence of program initialization, some command recognition and file opening, and a loop of reading data, calculation, writing data, and finally some cleanup and task exit. If you are careful in the initial design stages, then these operations can all be isolated in different subroutines with a clean calling interface. If overlays become necessary, then it is almost trivial to construct an overlay description file that overlays the separate phases your program goes though.

As a simple example, I took the magtape utility used previously in section 2.0 and overlaid just the command parsing subroutine with the rest of the program:

```
.ROOT   MAIN-*(COMMAND,allelse)
.END
```

Unoverlaid, the program uses 24660 words; the overlaid version uses only 21390 words -- over 3000 words saved without having to change a single line of code!

For further information on using overlays, see the RSX-11M/M-Plus Task Builder manual and the Fortran IV-Plus User's Guide.

11.0 USING RESIDENT SYSTEM LIBRARIES

You can save disk space, physical memory space, link time, and load/swap time by including one or more resident libraries in your program's virtual address space. The resident library mechanism on RSX systems allows all programs that have included the library to share a single copy of the code in core.

Note, if your objective is to increase task address space only, a resident library cannot help and typically will cost you some space. This is because entire Address Page Registers are needed to map resident libraries and even if you used all modules in the library, you are not increasing task address space. However, the other savings can be substantial (see example below).

Many systems have at least one library containing the FCS I/O modules, typically called FCSRES. RSX-11M Plus systems may also have a supervisor-mode FCS library, usually named FCSFSL. Systems may also have a Fortran IV-Plus resident library named F4PRES. Check with your system manager to find out what shared libraries exist on your system.

The Task Builder options LIBR= or SUPLIB= are used to reference resident libraries:


```
        ENTER OPTIONS:
        TKB>LIBR=FCSRES:RO[:n]
or
        ENTER OPTIONS:
        TKB>SUPLIB=FCSFSL:[-]SV[:n]
```

where RO informs the Task Builder that you want read-only access to the library and [-]SV specifies whether or not to replace calls to the supervisor-mode library with context swithc vectors. For more information on supervisor-mode lirbaries, read the RSX-11M/M-Plus Task Builder Manual, section 3.1.8.

You only need to specify the segment number n (between 1 and 7) if you do not want the standard library mapped to APR 7 (virtual address 160000 to 177777) or the supervisor-mode library starting a supervisor APR 0. If you program uses VIRTUAL arrays, you must specify LIBR=FCSRES:RO:6 to avoid conflicts with the address mapping for the VIRTUAL array. Unpredictable program behaviour can result if the default mapping to APR 7 is used.

RSX-11M Plus V1.0 supplied an alternate, faster supervisor call mechanism that has no known problems and will probably become the default call mechanism in future versions. It is documented on an on-line help file. Type HELP/UNS SML for details.

The following table issustrates the amount of memory saved in a production, real-time PDP-11/34 system when programs were all linked to an F4PRES that was also linked to FCSRES:

	w/o LIBR=	w/ LIBR=	Savings
BUFMGR	5562	1910	3652
DELAY	5824	2172	3652
MSGMGR	9290	2382	6908
SAMPLER	10126	3236	6890
STARTUP	9740	2540	7200
TRIGGER	6674	3022	3652
WRITR	11586	4170	7416
FCSRES	0	4096	-4096
F4PRES	0	4096	-4096
	58802	27624	31178

LAST-DITCH METHOD FOR FILE RECOVERY

Chris Doran

Sira Institute Limited
Chislehurst, England

The program below extends Helman's set of disk recovery programs ("Multi-Tasker" Vol 11 No 2, pp 71-77, August 1979) to the case where file headers have been corrupted. The only remedy I can think of in this situation is to manually inspect the contents of each block of the disk. Obviously, this is only really practicable for ASCII files, and is unlikely to help recover task or object modules.

Based on Helman's LOCATE program, it used a logical block QIO to fetch 512 bytes and display them on the terminal. It then asks which file (if any) you want the text output to. Valid answers are: 0 or carriage- return for no output, -number to STOP, or number n>=3 to output to file FOR00n.DAT. The maximum value of n is set by PARAMETER NMAX, and UNITS etc. statements in the build command file. Assuming all record information has been lost, there is no way to insert newlines automatically. The program therefore takes any non-printing characters as parts of byte counts and uses them as line delimiters. This seems to work well in practice.

As in Helman's program, <*>'s mark possibly system dependent values (ok for RSX-11M V3.2). Statements 70 and 71 send escape sequences to reset a VT52 from any strange mode the control characters in a binary file may have set it to. Put in escape sequences appropriate to your terminal. Typing of the program as supplied can be verified with the SLP checksum facility:

```
>SLP
SLP>=NL:/CS:042606
@ASCII.FTN
/
```

This program should be considered as a last-ditch method. It takes a long time to go through a whole disk, and the recovered files may be jumbled, and need a lot of editing. However if (as happeded to me!) the only alternative is to start again without even a listing, almost anything goes.

```
      PROGRAM ASCII
C
C     LOCATE AND RECOVER CORRUPTED ASCII FILES
C
C     C J DORAN, SIRA INSTITUTE LTD., SOUTH HILL, CHISLEHURST
C                KENT, BR7 5EH, ENGLAND.
C
C     THIS IS A LAST-DITCH PROGRAM TO RECOVER FILES FROM A CORRUPTED
C     DISK, IN WHICH EVEN THE FILE HEADERS HAVE BEEN LOST.  IT WILL
C     READ THE DISK BLOCK BY BLOCK AND DISPLAY EACH BLOCK IN ASCII
```



```
C     ON THE TERMINAL. IT WILL THEN ASK WHETHER YOU WANT THAT BLOCK
C     WRITTEN TO AN OUTPUT FILE. ANSWER 0 IF NOT, >2 TO WRITE IT TO
C     FILE FOR00N.DAT, WHERE N IS THE NUMBER YOU SPECIFY.
C
C     TASK BUILD AS:
C
C     >TKB
C     TKB>ASCII,ASCII/-SP=ASCII
C     TKB>/
C     ENTER OPTIONS:
C     TKB>UNITS=<NMAX+2>
C     TKB>ACTFIL=<NMAX+2>
C     TKB>ASG=SY:1,TI:2,SY:3:4:5:6:7:8:9:10
C     TKB>ASG=SY:11:12:13:14...:<NMAX+2>
C     TKB>MAXBUF=512
C     TKB>GBLPAT=ASCII:$TYPE:2        ; RE-ASSIGN TYPE AND
C     TKB>GBLPAT=ASCII:$ACCPT:2       ; ACCEPT TO LUN 2
C     TKB>//
C
C     BASED ON HEADER PROGRAM BY W P HELMAN,
C           "THE MULTI=TASKER", V11 NO2 PP71-77
C
      PARAMETER NMAX=12                          ! MAX NO OF FILES+2
      BYTE B(512),NAME(9),EXT(3)
      INTEGER*2 A(256),IPR(6),ISB(2),NBLK(2)
      INTEGER*4 NBLOCK
      EQUIVALENCE (NBLOCK,NBLK),(A,B)
C
      DATA IPR(2)/512/
      DATA IFNAM,IFTYP,IFVER/"30,"33,"34/        ! FILE ID POINTERS <*>
      DATA JUIC1,JUIC2,JCRDT/"12,"11,"110/       ! MAY BE SYSTEM DEPENDENT <*>
      DATA ICHH,IESQN,IUSE/7,"57,"63/            ! <*>
C
      CALL GETADR(IPR,A)
      TYPE 1
 1    FORMAT('$DEVICE: ')
      ACCEPT 2,IDEV
 2    FORMAT(1A2)
      TYPE 3
 3    FORMAT('$UNIT = ')
      ACCEPT 72,IUNIT
      CALL ASNLUN(1,IDEV,IUNIT)
      TYPE 4
 4    FORMAT('$START AT BLOCK = ')
      ACCEPT 72,NBLOCK
      IF(NBLOCK.NE.0) NBLOCK=NBLOCK-1
C
 10   NBLOCK=NBLOCK+1                            ! GET NEXT LOGICAL BLOCK
      IPR(5)=NBLK(1)
      IPR(4)=NBLK(2)
      A(1)=0
      A(4)=0
      CALL WTQIO(512,1,6,,ISB,IPR,IDS)           ! READ BY LOGICAL BLOCK
      IF(ISB(1).EQ."354) STOP 'END OF DISK'
      IF(ISB(1).EQ.1) TYPE 61,NBLOCK,ISB,IDS     ! TEST FOR ERROR
```

```
        IF(ISB(1).NE.1) GO TO 10                ! AND TRY NEXT BLOCK
        IF((A(1).NE."27027).OR.(A(4).NE."401)) GOTO 50 ! TEST FOR HEADER <*>
        CALL R50ASC(9,A(IFNAM),NAME)
        CALL R50ASC(3,A(IFTYP),EXT)
        IVERS=A(IFVER)
        TYPE 11, NBLOCK,NAME,EXT,IVERS,(B(I),I=JCRDT,JCRDT+"14),
     1          B(JUIC1),B(JUIC2),A(ICHH)
        GOTO 10
C
 11     FORMAT('0BLOCK: ',I8,',  HEADER FOR FILE:',/,
     1          X,9A1,'.',3A1,';',O5,2X,2A1,'-',3A1,'-',2A1,X,2A1,
     2          ':',2A1,':',2A1,' [',O3,',',O3,']   ',O7)
C
 61     FORMAT(' QIO ERROR, BLOCK:',I8,' ISB(1),ISB(2),$DSW =',3O7)
C
C NOT HEADER, TREAT AS FILE BLOCK
 50     TYPE 51,NBLOCK,(B(I),I=1,512)
 51     FORMAT('0BLOCK ',I8,/,8(/,X,64A1),//)
C  INCLUDES ESCAPE SEQUENCES TO RESET TERMINAL CHARACTERISTICS.
C  HERE FOR VT-52: NOHOLD, NO ALT KEYPAD, NO GRAPHICS
 70     TYPE 71,"33,"33,"33,NBLOCK
 71     FORMAT('$',A1,'\',A1,'>',A1,'GCopy block ',I8,' to file No = ')
        ACCEPT 72,N
 72     FORMAT(I7)
        IF(N.EQ.0) GOTO 10                      ! NOT WANTED
        IF(N.LT.0) STOP
        IF(N.LE.2 .OR. N.GT.NMAX) GOTO 70       ! ILLEGAL LUN
C WRITE OUT BLOCK AS A SERIES OF LINES, TREATING ALL NON-PRINTING CHARS
C AS LINE TERMINATORS TO SAVE A LOT OF EDITING OUT OF BYTE COUNTS.
        I=0
 80     I=I+1
        IF(I.GT.512) GOTO 10
        IF(B(I).LT."40 .AND. B(I).NE."11) GOTO 80
        J=I
 90     IF(J.EQ.512 .OR. (J-I+1).EQ.131) GOTO 100
        J=J+1
        IF(B(J).GE."40 .OR. B(J).EQ."11) GOTO 90
        J=J-1
 100    WRITE(N,101) (B(K),K=I,J)
 101    FORMAT(X,<J-I+1>A1)
D       TYPE 101,(B(K),K=I,J)
        I=J
        GOTO 80
        END
```


AN INDIRECT FILE FOR TAPE OPERATIONS

Ira L. Caplan

Institute for Environmental Medicine
University of Pennsylvania Medical School
Philadelphia, PA 19104

Magnetic tape offers an inexpensive means of file backup. It is often useful for programmers to maintain their most important files on tapes. Maintaining tape files, however, can be a time consuming and multi-step process. For example, because files in mid-tape cannot be changed, it is difficult to update them with newer versions. The user must first transfer the entire tape to disk, replace the disk file, and transfer everything back to tape, as well as do whatever operations are necesary to prepare and disengage the disk.

This paper presents an indirect command file, TAPE, which allows one to maintain tape files almost as easily as disk files. It lets the user replace, delete, add and transfer files by entering a few simple commands. The actual operation involves copying tape files to a scratch disk, where the operations are performed, and copying the results back to tape; however, this is virtually transparent to the naive user. Furthermore, with the exception of physically loading the tape and scratch disk, all device preparation and disengagement is done by the indirect command file.

1.0 COMMANDS AND OPTIONS AVAILABLE IN TAPE.CMD

1.1 COMMANDS

(Each is initiated by giving the first letter of its name in reponse to a prompt)

REPLACE....Replace a tape file with a more recent version.
DELETE.....Delete a file from tape.
ADD........Add a file to tape.
TRANSFER...Copy tape file to the user's account.
CREATE.....Create tape (i. e., write over whatever is on
           the tape).

1.2 OPTIONS

CHANGE.....Change one or more operations previously specified.
           Initiated by giving an appropriate response to
           a prompt following a display of the user's choice.
LOGOFF.....Logoff when finished.

DELAY.....Delay the execution of the tape operations for
number of hours specified.
DIRECTORY..Give a directory of the tape when finished.

2.0 SUMMARY OF TAPE.CMD FLOW

TAPE first asks the user the name of the tape, and allocates the tape drive. It asks the user which device the scratch disk is loaded on, tests whether the disk is labelled "SCRATCH" (aborting if it is not), and allocates and mounts the disk. It displays the operations codes, asks the user to specify an operation, and, if appropriate, it asks him to specify a file, repeating this procedure until the user indicates there are no more operations. Similarly, it displays the operations (and files) chosen, and asks him to verify these, repeating this until all choices are verified. The use is asked if he wants to use logoff, delay or directory options. Following any specified delay, TAPE mounts the tape, and carries out the operations all at once, in order. It makes a tape directory if requested, disconnects and deallocates tape and scratch disk, and, if requested, logs the user off.

3.0 DETAILED DESCRIPTION OF TAPE.CMD FLOW

3.1 Prerequisites For Use

The user must have the tape loaded and on-line, and have loaded a disk labelled SCRATCH.

3.2 Device Preparation

1. The user is asked to specify the name of the tape. TAPE allocates the tape drive (MT0:).

2. The user is asked to specify the name of the device on which a scratch disk is loaded. TAPE allocates the device, and mounts the disk SCRATCH on it. As a safeguard (since the disk will be initialized), TAPE attempts to open a file on the disk. On error -- if for instance the disk is not labelled SCRATCH -- it prints an error message and stops.

3. The scratch disk is initialized.

3.3 Operations Entry

1. Before making each set of entries, the user is presented with a table of operation codes, the codes being single letters (C, A, D, R, and T) corresponding to the commands shown in 1.1.

2. The user is asked to enter an operation. If he enter A, D, R, or T ("substantive" operations), he is then asked to enter a file, that is, the file he wants added, deleted, replaced or transferred. Wild cards can be used, except for the version number of a file to be deleted - this is assumed to be a wild card.

   Standard defaults apply. C means that a tape is being created; that is, that its present contents should be overwritten. When C is used, it makes sense only to add files or delete from those that have been added (although TAPE does not recognize this). Following the entry of a file name, or of operation C, TAPE starts operations entry again (3.3.1).

3. A list of the substantive operations chosen and the file associated with each is presented to the user. If he specified a tape was being created, this feature is also shown.

4. The user is asked to verify his choices, by typing either the number of an entry he wishes to change, typing 999 to change all entries, or hitting carriage return if all entries are O.K. If he changes a single entry, TAPE makes the change and begins verification again (3.3.3). If he changes all entries, TAPE redirects its flow from 3.3.1.

3.4 Options Information Entry

1. The user is asked if he wants a directory of the tape when finished, how many hours he wants to delay execution (a carriage return indicating none), and if he wants to be logged off when finished.

2. TAPE delays for the period requested.

3.5 Operations Execution

1. If the tape is being created, TAPE initializes the tape. It mounts the tape in any case.

2. If the tape is not being created (that is, operation C was not specified) TAPE transfers the contents of the tape to the scratch disk. If C was specified, no transfer is made.

3. TAPE carries out the operations on the newly transferred scratch disk files in the order requested. Replacing a file deletes it from the scratch disk, and replaces it with the latest version from the (default) disk. Deleting a file deletes all of its versions from the scratch disk. Adding a file copies it from the default disk to the scratch disk. Transferring a file copies it from the scratch to default disk.

4. TAPE transfers the contents of the scratch disk to the tape, writing over whatever is there.

3.6 Disengagement

1. If the user had requested a directory, TAPE places a tape directory in the file DIR.LST, and displays it at the terminal.

2. TAPE dismounts and deallocates both the tape and scratch disk.

3. If the user had requested to be logged off, TAPE logs him off.

4.0 TAPE.CMD CODE

The code follows a programming technique making use of defined macrostructures, developed by Roger Sessions of the Institute for Environmental Medicine. "If", "if-else", "repeat", and "delayed repeat" structures are recognizable by headings. Statement labels are placed toward the right so they do not obscure the flow. Code associated with a particular structure is indented two spaces from its headings.

NOTE

Although the file presented here uses PIP, it could be easily modified to use FLX.

```
.SETS NULL ""
.SETS CARAT "
.SETS C "C"
.SETS A "A"
.SETS R "R"
.SETS D "D"
.SETS T "T"
.SETS N "N"
.ENABLE SUBSTITUTION
```



```
.;
.;INTRODUCE FILE
.;--------- ----
.OPEN TI:
.ENABLE DATA
```

TAPE OPERATIONS

THIS FILE ALLOWS YOU TO ADD FILES TO A TAPE, DELETE FILES FROM A TAPE, REPLACE TAPE FILES WITH THEIR LATEST VERSIONS, OR TRANSFER FILES FROM TAPE TO DISK. YOU CAN SPECIFY AS MANY OPERATIONS AS YOU WANT, AND THEY WILL THEN BE CARRIED OUT IN THE ORDER THAT YOU ENTERED THEM.

PLEASE LOAD A DISK LABELLED "SCRATCH", AND NOTE WHICH DRIVE IT IS ON. MAKE SURE YOUR TAPE IS WRITE ENABLED AND ON LINE.

```
.DISABLE DATA
.CLOSE
.;
.;GET TAPE NAME; ALLOCATE TAPE
.;--- ---- ----  -------- ----
.ASKS TAPE ENTER TAPE NAME
ALL MT0:
.;
.;GET DEVICE NAME; TEST IF DISK IS SCRATCH
.;--- ------ ----  ---- -- ---- -- -------
.ASKS DEVICE ENTER DISK DEVICE
.GOSUB TEST
.;
.;INITIALIZE DISK AND CONNECT TO SYSTEM
.;---------- ---- --- ------- -- ------
INI 'DEVICE'SCRATCH
MOU 'DEVICE'SCRATCH
UFD 'DEVICE''<UIC>'
.;
.;GET OPERATIONS INFORMATION
.;--- ---------- -----------
.OPEN #2 TI:
.ENABLE DATA #2
```

--OPERATIONS ENTRY BEGUN--

```
.DISABLE DATA
.;REPEAT UNTIL (OPERATIONS AS A WHOLE VERIFIED)
                                                .10:
.;GET OPERATIONS CHOICES
.;--- ---------- -------
.SETF NUTAPE
.OPEN #1 TAPE.DAT
.GOSUB OPRGET
.;
.; SHOW OPERATION CHOICES TO USER
.; ---- --------- ------- -- ----
.; REPEAT UNTIL (NO SINGLE OPERATION NOT VERIFIED)
                                                .20:
```

```
    .GOSUB OPRSHO
.;
.;    HAVE USER VERIFY CHOICES
.;    ---- ---- ------ -------
    .GOSUB VERIFY
.; END (REPEAT)
  .IFF SOPOK .GOTO 20
.;END (REPEAT)
.IFF AOPOK .GOTO 10
.ENABLE DATA #2

                        ---OPERATIONS ENTRY COMPLETE---

.DISABLE DATA
.CLOSE #2
.;
.;GET OPTIONS INFORMATION
.;--- ------- -----------
.ASK DIREC DO YOU WANT A DIRECTORY (DIR.LST) WHEN FINISHED
.ASKS DELAY ENTER NUMBER OF HOURS DELAY (FOR NO DELAY, HIT RETURN)
.ASK LOGOFF DO YOU WANT TO BE LOGGED OFF WHEN FINISHED
.;
.;CARRY OUT DELAY IF REQUESTED
.;----- --- ----- -- ---------
.IF DELAY NE NULL .DELAY 'DELAY' .H
.;
.;CARRY OUT OPERATIONS
.;----- --- ----------
.OPEN TI:
.ENABLE DATA

                        ---OPERATIONS EXECUTION BEGUN---

.DISABLE DATA
.CLOSE
.IFT NUTAPE INI MT0:'TAPE'
MOU MT0:'TAPE'
.IFF NUTAPE PIP 'DEVICE'=MT0:*.*;*
OPENR TAPE.DAT
.;
.;REPEAT UNTIL (EOF)
                                                .50:
  .READ OPREC
  .IFT <EOF> .GOTO XEQDUN
  .GOSUB XEQ
.;
.;END (REPEAT)
.GOTO 50
.XEQDUN:
.CLOSE
PIP MT0:/RW='DEVICE'*.*;*
PIP TAPE.DAT/PU
.IFT DIREC PIP DIR.LST=MT0:*.*;*/LI
.OPEN TI:
```



```
.ENABLE DATA

                        ---OPERATIONS EXECUTION COMPLETE---

.DISABLE DATA
.CLOSE
.;
.;GIVE TAPE DIRECTORY IF REQUESTED
.;---- ---- --------- -- ---------
.IFT DIREC PIP TI:=DIR.LST
.;
.;DISCONNECT DEVICES
.;---------- -------
DMO 'DEVICE'
DEA 'DEVICE'
DMO MT0:
DEA MT0:
.OPEN TI:
.ENABLE DATA

                        ---END OF TAPE OPERATIONS---

.DISABLE DATA
.CLOSE
.;
.;LOG USER OFF IF REQUESTED
.;--- ---- --- -- ---------
.IFT LOGOFF BYE
.STOP
.;
.;
.;SUBROUTINE TEST
.;---------- ----
.TEST:
ALL 'DEVICE'
MOU 'DEVICE'SCRATCH
UFD 'DEVICE''<UIC>'
.ONERR DSKERR
.OPEN 'DEVICE'TEST.FIL
.CLOSE
DMO 'DEVICE'
.OPEN TI:
.ENABLE DATA
        *****NOTE: IGNORE THIS DISMOUNT MESSAGE****


.DISABLE DATA
.CLOSE
.RETURN
.DSKERR:
.OPEN TI:
.ENABLE DATA

        ************************************
        *        WRONG DISK LOADED         *
```

```
        *  PLEASE LOAD DISK LABELED "SCRATCH" *
        *          AND TRY AGAIN              *
        ***************************************

.CLOSE
.STOP
.;
.;SUBROUTINE OPRGET
.;---------- ------
.OPRGET:
.;LIMITS OF REPEAT
                                                .GOTO 60
                                                .70:
   .ASKS FILE ENTER FILE NAME
   .DATA #1 'OPER' FILE'
                                                .60:

.; BEGIN REPEAT
.;
   .ENABLE DATA #2

          ***********************************
          *          OPERATION CODES        *
          *          --------- -----        *
          * CREATING TAPE.................C *
          * ADD...........................A *
          * DELETE........................D *
          * REPLACE WITH LATEST VERSION...R *
          * TRANSFER TO SYSTEM DISK.......T *
          * NO MORE OPERATIONS............N *
          ***********************************


.DISABLE DATA
   .ASKS OPER ENTER OPERATION
   .IF OPER EQ C                                .GOTO 80
                                                .GOTO 90
                                                .80:

     .SETT NUTAPE
     .GOTO 60
                                                .90:
.; END (IF)
.;
.;END (REPEAT)
.IF OPER NE N .GOTO 70
.CLOSE #1
.RETURN
.;
.;SUBROUTINE OPRSHO
.;---------- ------
.OPRSHO:
.SETN NUM 0.
.OPENR #1 TAPE.DAT
.ENABLE DATA #2
```



```
                        OPERATIONS SPECIFIED
                        ---------- ---------

.DISABLE DATA
.IFT NUTAPE                                     .GOTO 100
                                                .GOTO 110
                                                .100:
    .DATA #2                    1. CREATING TAPE
    .INC NUM
                                                .110:
.;END(IF)
.;
.;REPEAT UNTIL (EOF)
                                                .115:
   .READ #1 OPREC
   .IFT <EOF> .GOTO SHODUN
   .INC NUM
   .PARSE OPREC CARAT OPER FILE
   .IF OPER EQ A .SETS OPSHO "ADD"
   .IF OPER EQ D .SETS OPSHO "DELETE"
   .IF OPER EQ R .SETS OPSHO "REPLACE"
   .IF OPER EQ T .SETS OPSHO "TRANSFER"
   .DATA #2                             'NUM'. 'OPSHO' 'FILE'
.;END (REPEAT)
.GOTO 115
.SHODUN:
.CLOSE #1
.RETURN
.;
.;SUBROUTINE VERIFY
.;---------- ------
.VERIFY:
.SETT AOPOK
.SETT SOPOK
.ENABLE DATA #2

IF ALL ENTRIES CORRECT, HIT RETURN.
TO REDO ALL ENTRIES, TYPE 999.
.DISABLE DATA
.ASKN [::0.] OPFIX TO CHANGE A SINGLE ENTRY, TYPE IN ITS NUMBER.
.;
.IF OPFIX GT 0                                  .GOTO 120
                                                .GOTO 200
                                                .120:
.; MAKE APPROPRIATE ENTRIES CHANGE
.; ---- ----------- ------- ------
  .IF OPFIX EQ 999.                             .GOTO 130
                                                .GOTO 140
                                                .130:
.;   CHANGE ALL ENTRIES
.;   ------ --- -------
     .SETF AOPOK
                                                .GOTO 180
.; ELSE
                                                .140:
```

```
      .SETF SOPOK
.;
.;    CHANGE "NEW TAPE" FEATURE
.;    ------ --- ---- -------
      .IF OPFIX EQ 1 .AND .IFT NUTAPE          .GOTO 150
                                               .GOTO 160
                                               .150:
        .SETF NUTAPE
                                               .GOTO 170
.;    ELSE
.;
.;    MAKE SPECIFIC OPERATION CHANGE
.;    ---- -------- --------- ------
                                               .160:
.;      GET NEW FEATURES
.;      --- --- --------
        .ASKS NUOP ENTER OPERATION
        .ASKS NUFIL ENTER FILE
.;
.;      MAKE NEW VERSION OF TAPE.DAT WITH NEW FEATURES
.;      ---- --- ------- -- -------- ---- --- --------
        .OPENR #1 TAPE.DAT
        .OPEN #3 TAPE.DAT
        .SETN FIXNUM 0.
        .IFT NUTAPE .DEC OPFIX
.;
.;      REPEAT UNTIL (EOF)
                                               .190:
          .INC FIXNUM
          .READ #1 OPREC
          .IFT <EOF> .GOTO FIXDUN
          .IF FIXNUM EQ OPFIX .SETS OPREC NUOP+" +NUFIL
          .DATA #3 'OPREC'
.;      END (REPEAT)
        .GOTO 190
        .FIXDUN:
        .CLOSE #1
        .CLOSE #3
                                               .170:
.;    END (IF-ELSE)
                                               .180:
.; END (IF-ELSE)
                                               .200:
.;END (IF)
.;
.RETURN
.;
.;SUBROUTINE XEQ
  ---------- ---
.XEQ:
.PARSE OPREC CARAT OPER FILE
.IF OPER EQ A                                  .GOTO 210
                                               .GOTO 220
                                               .210:
.;  ADD
```



```
.;    ---
      PIP 'DEVICE'='FILE'
                                               .220:
.;END (IF)
.IF OPER EQ D                                  .GOTO 230
                                               .GOTO 240
                                               .230:
.;    DELETE
      ------
      PIP 'DEVICE''FILE';*/DE
                                               .240:
.;END (IF)
.IF OPER EQ R                                  .GOTO 250
                                               .GOTO 260
                                               .250:
.;    REPLACE
.;    -------
      PIP 'DEVICE''FILE';*/DE
      PIP 'DEVICE'='FILE'
                                               .260:
.;END (IF)
.IF OPER EQ T                                  .GOTO 270
                                               .GOTO 280
                                               .270:
.;    TRANSFER
      --------
      PIP ='DEVICE''FILE'
                                               .280:
.;END (IF)
.RETURN
```

RSX-11M POOL FRAGMENTATION

Ian Webb

Swiss Institute for Nuclear Physics
5234 Villigen, Switzerland

While using a PDP 11/34 to implement a gerneral purpose data acquisition system at our institute we encountered a severe pool problem. This problem became acute enough for investigation when the machine crashed several times a day. The first thing that was noticed was that over a period of time the pool fragments, i.e. the largest available block of contiguous pool space, decreased as a function of time.

To investigate why RSX-11M pool space fragments, we wrote a primitive program to printout a map of the pool. This map indicated that I/O packets were spread throughout the pool. This appeared to be the primary source of our pool fragmentation problem. This was rather puzzling in that we had specified 10 pre-allocated I/O packets during SYSGEN, and tacitally assumed that these I/O packets would occupy contiguous space. As it turned out this assumption was totally wrong.

Using RMD to monitor the performace of our data acquisition system, it was difficult to ascertain where all the I/O packets disapeared to. Although a fairly large number of tasks (approximately 10) where installed as the system was brought up; very few I/O packets should have been used by the system. The I/O packet available counter of RMD decreased as the tasks were subsequently removed. We later associated this phenomena with the fact that a PCB is required for each installed task and these PCB's utilize I/O packet space.

Our first thought was that if we can increase the number of pre-allocated packets then the fragmentation problem should be reduced. The module SYSCM contains the data structures used to handle pre-allocated I/O packets under RSX:

```
        $PKAVL::.word 0         ;pointer to first packet in list
        $PKNUM::.byte 0         ;number of packets currently in list
        $PKMAX::.byte Q$$OPT    ;maximum number allowed in list
```

The first solution adopted involved patching $PKMAX directly, increasing the number of allocated packets and then re-booting the system. This had the desired effect that more I/O packets were in fact allocated, i.e. one didn't have to perform another SYSGEN. This solution didn't seem to cure the problem. I/O packets where still splattered all over our dynamic pool space.

The obvious solution to this problem was that pre-allocated I/O packets were not actually created using a contiguous block of pool. As far as we can determine this is in fact the case. The module $DEACB checks to see if a pool fragment of the same size as an I/O packet is being returned and if all the pre-allocated packets are in fact allocated. $DEACB is responsible for ensuring that the number of pre-allocated packets specified by $PKMAX actually get linked into the list. Code does not exist in INITL to pre-allocate contiguous space for the I/O packets and the only reference that we could discover to the variables involved, exists in $DEACB.

How can one overcome this problem? We decided to write a small task that would allocate a number of I/O packets using $ALPKT and then de-allocate them using $DEPKT. Having written the program it was inserted into the start-up command file before any other tasks were installed and/or run. This was done in the belief that the contiguous space should be allocated in a block before the system using I/O packets. We struck our next problem when the start-up command file finished, some of our contiguous I/O packet space had already been allocated. Which tasks were using this space? This was when it was discovered that each installed task needed a PCB and an I/O packet was being used to contain it.

Our solution to this problem was to call our packet fix program at the end of the start-up command file thus ensuring that a contiguous block of I/O packets are available to support the system once it is running.

None of these solutions can be considered ideal. Once the system runs out of pre-allocated I/O packets you are very likely to start fragmenting the pool as any returned I/O packet is considered as a candidate for being retured to the pre-allocated list. If the pre-allocated I/O packet list becomes exhausted and pool subsequently fragments then even removing all the installed tasks does not return the contiguous pool space, as this is being held by pre-allocated I/O packets. A better solution would be to allocate I/O packets at system start-up time and ascertain the limits of the contiguous space used. Then if the

pre-allocated I/O packet list becomes exhausted one can still ensure that only the correct I/O packets are returned to the contiguous list.

As it turned out our system required approximately 20 I/O packets before the fragmentation problem disappeared. Having a greater number of pre-allocated I/O packets than the maximum specified by SYSGEN (15) does not seem to cause any problems. Due to the fact that other people utilize our PDP 11/34 even when our data acquisition system is running we struck pool space problems with 20 pre-allocated I/O packets and need to move to a 20k Executive; you can't win them all!

```
        .ident  /V01A/
        .title pktfix  Allocate a block of I/O packets
        .enable LC
count:  .word   0               ;Counter
        .mcall  exit$s
pktfix::call    $swstk,p30      ;Switch to system state and
                                ; allocate packets
        movb    nopkt,$PKMAX    ;Set up maximum number of packets
        mov     nopkt,count     ;Use count as a loop counter
                                ;Nb.  $ALPKT and $DEPKT destroy
                                ;a lot of registers
p10:    call    $ALPKT          ;Allocate an I/O packet
        mov     r0,-(sp)        ;Save address of packet
        dec     count           ;Count number of packets
        bgt     p10             ;Loop through packets
;
        mov     nopkt,count     ;Use count as a loop counter
p20:    mov     (sp)+,r0        ;Retrieve packet address
        call    $DEPKT          ;Deallocate I/O packet
        dec     count           ;count number of packets
        bgt     p20             ;loop through packets
        return                  ;Switch back to user state
p30:    exit$s                  ;Finished
        .end    pktfix

TKB>pktfix.tsk;1/pr:5=pktfix,lb:[1,54]rsxllm.stb/ss
TKB>/
TKB>GBLDEF=NOPKT:24
TKB>//
```

THE CASE FOR STRUCTURED MACRO-11: SUPERMAC

Robert Bismuth

Redkite Software Limited
Swansea, Wales, U.K.

The editor of the Multi-Tasker recently learned that I have started coding in SUPERMAC as opposed to ordinary MACRO-11 and, largely due to the fact that he wishes to be talked into using it himself, he has pressed me into writing this article concerning why I use it. I shall do this by first talking briefly about MACRO-11, then providing the contrast of SUPERMAC.

MACRO-11 is the assembly/macro language available under RSX11M. Naturally, it requires thinking "one machine instruction at a time" so yielding the ultimate control of program logic. MACRO-11 code must be well commented or it will become un-readable very quickly (ie. without comments it is largely "write only code"). These two 'features' of assembly languages combine to cause horror in the minds of most 'high-level-language' programmers, at the very least they threaten colleagues with being shot when the suggestion is made that they should code in assembler. Usually they point fingers and make comments about productivity, which mainly reduce to statements about the difficulty and general verboseness of assembly coding together with shouts of non-portability of the code produced.

SUPERMAC offers an alternative for those who need the speed of assembly code, but do not wish to sacrifice the 'structured approach'. It is an "extension" of MACRO-11: a structured language composed of MACRO-11 macros, which when assembled by the MACRO assembler expand to give the appropriate PDP/LSI assembly instructions. Once the syntax is understood, a programmer has only to understand the addressing modes of PDP/LSI 11s and it becomes possible to write code in SUPERMAC which is 99% as efficient as straight assembly code. Note that as it is really the MACRO assembler which is 'compiling' SUPERMAC, given MACRO-11 cross compilers for microprocessors, SUPERMAC becomes a 'transportable' language. Of course, VMS posesses a MACRO-11 like assembler, need I say more?

Conditional structures such as IF ... THEN ... ELSE ... END, REPEAT , UNTILL , WHILE , FOR , CASE , etc. all exist in SUPERMAC, as well as ON.ERROR for carry condition detection. For those who need their crutches, there is even a GOTO statement and for those who need even less subtle crutches, a JUMPTO statement. Finally for those who really cannot get away from thinking one machine instruction at a time, as SUPERMAC is assembled by the MACRO assembler, it is possible to switch at any time into pure MACRO-11/assembly language when writing a program in SUPERMAC.

Control statements exist for PROCEDURE definition, with two calling conventions: call with parameters in general registers, and Fortran compatible calls. Where symbols need to be defined for alternate entry points, there is an ENTRYPOINT statement. These are complemented by an easy error/noerror return statement. This control of module structure is further enhanced by the BEGINMODULE statement which allows for the insertion of standard copyright or disclaimer

notices as well as revision and version information.

Writing in SUPERMAC is much easier for the "high level language" type and code may be debugged via ODT on a "logical block" level. The first time a new user debugs a SUPERMAC program with its extremely readable listing, the advantages over ordinary MACRO-11 code become obvious. Generally, in my experience, SUPERMAC greatly reduces program coding time and gives an enormous reduction in debugging time (for those of us who cannot write bug free code). Furthermore SUPERMAC is available free on various SIG tapes or from the DECUS library: note that it is actually maintained (after a fashion) as several RSX products (such as RMS11K and the mag tape ACP) are written in it.

One final note, as SUPERMAC relies on macro expansion, IT TAKES A LONG TIME TO ASSEMBLE A SUPERMAC PROGRAM! - in general this makes for more efficient programming: programmers who know that they must wait a long time for assembly (let alone taskbuilding) are much more careful in the program design and entry, also the extra time for coffee allows for a more relaxed approach and philosophy to life.

I have included an example written in SUPERMAC to give a feel for its appearance and readability. Please note that I make no claims that it is written in any sort of reasonable style or as well commented as it could be. This example took one and a half hours to write code and debug - that's how fast SUPERMAC can be to use! It is concerned with connecting a logged in terminal to a modem line and so provide switched communication to another computer system. The assumption is made that the communication will be full duplex, with the remote system providing character echo and the remote line speed being lower than the terminal line speed. Unsolicited input ASTs and I/O completion ASTs are used: note that the program must be installed as non-checkpointable otherwise the terminal driver will stop the task for terminal I/O and so ruin the responsiveness of the task. This example is perhaps not the simplest if all that were required were an illustration, but I felt that people under-utilize ASTs for this sort of application and so used it.

```
  .MCALL SMACIT,QIOW$S,EXIT$S,WTSE$S,ASTX$S,SETF$S
  SMACIT          ; DEFINE ALL SUPERMAC MACROS.

  BEGINMODULE RMTERM,RB01,10-JAN-82

;+
;
; RMTERM :    REMOTE TERMINAL ACCESS.
;
;             THIS PROGRAM PERMITS CONNECTION TO A REMOTE COMPUTER
;             VIA THE TERMINAL DRIVER OF THE HOST RSX11M SYSTEM.
;
;-

; SYMBOLIC VALUES :-

RMTLUN=4          ; REMOTE SYSTEM LUN.
TRMLUN=5          ; TI LUN.
TRMMSK=1          ; MASK FOR TERMINAL ACTIVITY.
RMTMSK=2          ; MASK FOR REMOTE ACTIVITY.
```

```
WTFLG=7          ; FLAG FOR WAITING.
STPCHR=0         ; THE EXIT CHARACTER.

; THE MAIN CONTROL BLOCK: -

   PROCEDURE MAIN,GLOBAL

   $CALL ATTACH <#TRMLUN,#TRMAST>    ; ATTACH TO TI:
   $CALL ATTACH <#RMTLUN,#RMTAST>    ; ATTACH TO REMOTE LINE.
   WTSE$S #WTFLG                     ; WAIT FOR EXIT FUNCTION.
   EXIT$S                            ; EXIT.

; THE ATTACH PROCEDURE: -

   PROCEDURE ATTACH,GLOBAL

   QIOW$S   #IO.ATA,R0,R0,,,,<R1>    ; GO DO IT.
   $RETURN                           ; AND GO BACK.

; THE TWO UNSOLICITED AST ROUTINES: -

   PROCEDURE TRMAST,GLOBAL

   LET TRMTMP := (SP)+               ; GET THE CHARACTER.
   LET TRMTMP := TRMTMP CLEARED.BY #177600 ; CLEAN OFF THE PARITY.
   IFB TRMTMP EQ #STPCHR             ; THE EXIT CHARACTER??
      SETF$S   #WTFLG                ; SET THE FLAG.
    ELSE
      PUSH  R0,R1,R2,R3,R4,R5        ; SAVE REGISTERS.
      $CALL RCVR <#TRMBLK>           ; CALL THE RECEIVE FCN WITH RIGHT PARAMS.
      POP   R5,R4,R3,R2,R1,R0        ; UNSAVE REGISTERS.
   END
   ASTX$S                            ; RETURN FROM AST STATE.

   PROCEDURE RMTAST,GLOBAL

   LET RMTTMP := (SP)+               ; GET THE CHARACTER.
   LET RMTTMP := RMTTMP CLEARED.BY #177600 ; CLEAN OFF THE PARITY.
   PUSH  R0,R1,R2,R3,R4,R5           ; SAVE REGISTERS.
   $CALL RCVR <#RMTBLK>              ; CALL THE RECEIVE FCN WITH RIGHT PARAMS.
   POP   R5,R4,R3,R2,R1,R0           ; UNSAVE REGISTERS.
   ASTX$S                            ; RETURN FROM AST STATE.

; THE CHARACTER RECEIVING PROCEDURE: -

   PROCEDURE RCVR,GLOBAL

   LET R1 := (R0) + 4(R0)            ; GET THE CIRCULAR BUFFER ADDR.
   LETB (R1) :=  10(R0)              ; PICK UP CHARACTER FROM TEMPORARY LOC.
   LET 4(R0) := 4(R0) + #1           ; SHOW GOT ONE.
   IF 4(R0) GT #131.                 ; OVER THE END??
      LET 4(R0) := #0                ; IF SO, RESET IT.
   END
   IF 12(R0) OFF.IN MASK             ; HAVE WE  I/O ACTIVE??
      $CALL  XMITR                   ; IF NOT, THEN SEND SOMETING
```



```
   END
   $RETURN

; THE TRANSMIT PROCEDURE: -

   PROCEDURE XMITR,GLOBAL

   LET MASK := MASK CLEARED.BY 12(R0) ; SHOW NO ACTIVITY IN CASE AST COMPLETE
   IF 2(R0) NE 4(R0)                 ; ANYTHING TO BE DONE??
      LET R2 := 20(R0)               ; PICK UP LINEAR BUFFER ADDRESS.
      LET R3 := #0                   ; SET CHAR COUNT.
      WHILE 2(R0) NE 4(R0)           ; UNTIL WE GET THERE
         LET R4 := 2(R0) + (R0)      ; GET NEXT CHAR LOC.
         LETB (R2)+ := (R4)          ; COPY IT OVER.
         LET R3 := R3 + #1           ; SHOW GOT ONE.
         LET 2(R0) := 2(R0) + #1     ; AND REMOVE FROM CIRC BUFFER.
         IF 2(R0) GT #131.           ; OVER THE TOP??
            LET 2(R0) := #0          ; RESET IT.
         END
      END                            ; END OF THE WHILE LOOP.
      LET MASK := MASK SET.BY 12(R0) ; SHOW I/O ACTIVE IN BIT MASK.
      QIOW$S   #IO.WAL,14(R0),14(R0),,,16(R0),<20(R0),R3,#0>  ; DO I/O .
   END
   $RETURN

; THE I/O COMPLETION AST ROUTINES: -

   PROCEDURE TRXAST,GLOBAL

   POP   TOP                         ; CLEAN THE STACK.
   PUSH  R0,R1,R2,R3,R4,R5           ; SAVE REGISTERS.
   $CALL XMITR <#TRMBLK>             ; CALL XMITR TO SEE IF THERE ARE ANY MORE.
   POP   R5,R4,R3,R2,R1,R0           ; UNSAVE REGISTERS.
   ASTX$S

   PROCEDURE RMXAST,GLOBAL

   POP   TOP                         ; CLEAN THE STACK.
   PUSH  R0,R1,R2,R3,R4,R5           ; SAVE REGISTERS.
   $CALL XMITR <#RMTBLK>             ; CALL XMITR TO SEE IF THERE ARE ANY MORE.
   POP   R5,R4,R3,R2,R1,R0           ; UNSAVE REGISTERS.
   ASTX$S


; THE XMITR AND RCVR PARAMETER BLOCKS: -

TRMBLK:
        .WORD TRMBUF                 ; 132. BYTE CIRC. BUFFER PTR.      0
        .WORD 0                      ; XMITR OUTPUT POINTER.            2
        .WORD 0                      ; RCVR INPUT POINTER.              4
        .WORD 0                      ; SPARE.                           6
TRMTMP:
        .WORD 0                      ; TEMPORARY CHAR STORAGE.          10
        .WORD RMTMSK                 ; BIT MASK VALUE.                  12
        .WORD RMTLUN                 ; LUN NUMBER.                      14
```

```
        .WORD TRXAST            ; COMPLETION ROUTINE.           16
        .WORD TRXBUF            ; LINEAR OUTPUT BUFFER.         20
RMTBLK:
        .WORD RMTBUF            ; 132. BYTE CIRC. BUFFER PTR.   0
        .WORD 0                 ; XMITR OUTPUT POINTER.         2
        .WORD 0                 ; RCVR INPUT POINTER.           4
        .WORD 0                 ; SPARE.                        6
RMTTMP:
        .WORD 0                 ; TEMPORARY CHAR STORAGE.       10
        .WORD TRMMSK            ; BIT MASK VALUE.               12
        .WORD TRMLUN            ; LUN NUMBER.                   14
        .WORD RMXAST            ; COMPLETION ROUTINE.           16
        .WORD RMXBUF            ; LINEAR OUTPUT BUFFER.         20

; THE ACTIVITY MASK:-

MASK:   .WORD 0

; THE BUFFER AREAS :-

TRMBUF: .BLKB 132.              ; CIRCULAR BUFFER.
TRXBUF: .BLKB 132.              ; LINEAR BUFFER.

RMTBUF: .BLKB 132.              ; CIRCULAR BUFFER.
RMXBUF: .BLKB 132.              ; LINEAR BUFFER.


  .END  MAIN                    ; END OF MODULE.
```


Many definitions of an ideal scheduler can be given. The one that I use is derived by my own system's particular needs. With that in mind: The job of a scheduler is to choose the process that has the greatest need for the CPU and is ready to use it. The difficulties are two-fold. First, how does one determine the process with the greatest need. Second, as an algorithm becomes more sophisticated it usually approaches being too costly to use. This is the ''Catch-22'' of scheduler design.

The decision making with respect to ''need'' has always been installation dependent and quite political. One early system is reported to have set the priority of incoming jobs proportional to the amount of business the user gave the computing center each month! On our system many interactive processes are run where poor response is defined as: If I can tell there are other users on the system, then it's too slow. This may be unrealistic on most systems, but we have an almost dedicated system for interactive graphics research with only two or three remote terminals doing program development. Interactive text editing is an example of an application that requires very good real-time response when tracking the cursor but occasionally requires a lot of the CPU when it does a global search on a complicated pattern or some word processing function such as filling and justifying an entire file. Most important to us is that our painting programs must be highly responsive to any ''brush movement'' (simulated via a tablet and a pen). In order to get this response, we need to run these processes at higher priority, but these programs also burn the CPU when the user selects a transformation function. What is needed is a scheduler that can determine if a process with high priority has just begun to take over the system.

The RSX11M round-robin scheduler attempts to distinguish between compute-bound and interactive processes by leaving blocked tasks at the top of the round-robin queue while rotating computing tasks to the bottom. However, if you have an interactive task that happens to be computing (no matter how slightly) at the time of a schedule interval, then it is placed at the bottom of the queue and must percolate back up through all compute-bound processes.

While the DEC supplied scheduler algorithm produces very good results for the amount of code involved, it has a few drawbacks. First, in order to get good response, the scheduler must be run very often. On our system, the scheduler works every 5 ticks. Second, interactive processes still run with a jerky motion. Third, there is no command to tune the parameters of the algorithm interactively. If DEC had seen fit to isolate the parameters in one place in exec common, it would have been trivial to implement such a process. As it exists now, one must patch the immediate operands of various instructions to acquire this capability.

Looking through the literature on scheduler algorithms one finds three basic designs. At the low end (efficient and simple) there is pure

round-robin. At the top end are algorithms that manipulate queues of different levels with quanta assigned to each level and rules for promotions and demotions between levels. Round-robin of some sort is usually implemented for the processes on each level. Tenex and IAS are examples of systems that use this method. It is interesting to note that, under Tenex [1] any user (who is willing to pay) can be guaranteed a fraction of the CPU.

In between are algorithms that use a priority driven dispatcher and a separate procedure or process to assign priorities dynamically. Unix [2,3] and VMS [4] are examples of this kind of system.

VMS has 16 priorities for normal processes and juggles these priorities based on an ad hoc procedure. A process gets its priority raised by a factor associated with different events, such as I/O completion, and lowers it every time it gets scheduled. I have not found VMS to be very responsive to interactive processes either and it has almost no way to tune the scheduling process other than by assigning different base priorities.

The algorithm used in Unix is quite simple. It assigns priorities based on a recent ratio of the amount of compute time to wall clock time used by the process. It then simply picks the process with the highest priority. The compute ratio is updated every second. Thus for example, if processes are simply looping, they will be scheduled round-robin each with a 1 second quantum. Any higher priority process can awaken and preempt a lower priority process. Thus the heavy users end up at the bottom and the interactive processes stay at the top (provided they continue to be light CPU users).

Since our system is becoming more of a production tool there is not a lot of time to experiment; therefore, I decided to implement a Unix-style algorithm in a privileged task that juggles the order of processes within a single priority. Then there would exist three classes of processes: above, below, and timesharing priority. The only information needed that is currently not available in RSX11M is the recent behavior of all the active processes. A simple approach which some accounting systems for RSX use is to increment a counter in the task control block (tcb) for the currently active task every time the clock interrupts. In version 3.2 there just happens to be a spare byte. Thus one can get the information desired by a small patch to the clock interrupt routine. Since I have a partition JPLCOM that looks like:

```
JPLCOM 117400 000400 MAIN COM
LDRPAR 120000 002500 MAIN TASK
TTPAR  122500 022000 MAIN TASK
         ....
GEN    252100 425700 MAIN SYS
```

I always have a little patch space at a fixed location below 20K. This makes it very easy to load some code into a known spot in exec space and make a small one instruction patch. Below is the clock interrupt code in RSX and the patch made to module tdsch.mac at the location shown below.


```
$CKINT::CALL    $INTSV,PR6
        .IF DF  K$$W11
        MOV     #3,K$$W11
        .ENDC
        MOV     #$INTCT,R4      <- modify to a jsr pc,@#patch
        INC     (R4)
        BEQ     10$
        RETURN
10$:    CALL    $FORK0

patch:
        mov @#$tktcb,r4         ;;; inc count for current task
        if t.actl(r4) ne 0      ;;; but check for null task
           incb t.off+2(r4)     ;;;  even though we never get
        fi                      ;;;  there (fire wall)
        mov #$intct,r4          ;;; do the patched instruction
        rts pc                  ;;; ret to main line interrupt
```

It is also necessary to disable the DEC supplied round-robin scheduler. This is rather easy; the conditional branch instruction as shown below is simply made a br (unconditional). The code for this is found in the same module (tdsch.mac).

```
ROBIN:                          ;REF LABEL
        .IF DF  R$$NDC
        DEC     RNDCT           ;TIME TO SCHEDULE?
 ----->  BNE     SWAP            ;IF NE NO
        MOV     #R$$NDC,RNDCT   ;RESET CLOCK TICKS TO NEXT INTERVAL
        CALL    $DRDSE          ;CAUSE A REDISPATCH OF PROCESSOR
```

One item I discovered during implementation was that the null task's tcb is not placed into the cell $tktcb which otherwise is always pointing to the currently running task. Therefore, when no task is busy, this cell erroneously points to whatever task was last active. I suppose the reason is to avoid a context switch back to that task should it become the next to schedule, but this is just a guess. This problem is easily overcome by running a priority 1 process that just loops. In our case we have a small program that reads the front panel switches and taking that as an address in either the lower 20k (exec) or the I/O page, displays on the lights the contents of the toggled location. I usually leave it set to the cylinder register on our disk controller. Of course it only runs if nothing is looping, and thus it can be used to monitor activity much the same as the idle pattern. (As an aside - I have a patch in the tt driver to increment a cell in JPLCOM every time it allocates a buffer from either pool or its partition area and a decrement on deallocate. Using this program I was able to set the size of the tt driver partition efficiently by monitoring this location dynamically.)

The process which runs periodically to adjust task priorities sorts the tasks having priority 50 according to the value of t.off+2 resulting in CPU bound tasks dropping to the bottom. Since there are only have 8 bits to play with, one has to keep the value of this cell below 255 or overflow will invalidate the counts. The first thing tried was simply

to clear this cell after each sort. This works fine until there are two looping processes. In this situation the first one will get the entire quantum while the other gets nothing. On the next sort they switch places (which is fine), but if there is an interactive process in the middle that uses only a small percentage of the CPU it will appear to be more compute-bound than the looping process that got none of the CPU during the last interval. This results in the round-robin of all three processes which is not very desirable.

What would be desirable is another 5 or so words in each tcb to store a moving average. However, short of this one can use a function that will retain a bit of history in a single value. One such function is:

newvalue := oldvalue/constant1 - constant2

Starting with this, I ended up making constant2 zero (that is I decided not to use it) and the value of constant1 about 1.2 (using scaled arithmetic). In addition, I experimented with a second set of constants which are additionally applied every m'th sort. The thought here was to have a way of making sure that the values never get too large and risk overflow by lowering the counters every so often. Setting m to a very large value will of course suppress this capability.

Taking an idea from the DEC round-robin scheduler, I also decided to check each task for a blocked status. When the task is in such a state another constant is used as follows:

newvalue := oldvalue/constant3

Here one can set constant3 to a very large number which causes all blocked tasks to get a new value of zero, placing them at the head of the priority 50 list. Other values are also possible in the event one does not wish to give special treatment to blocked tasks. In particular, a value of 1 will cancel this effect. All the constants are of course stored in the JPLCOM area so they can be modified easily using the open command.

Since the sort process is run every second, it should be rather efficient. There didn't seem to be much choice, however, since the active task list is just a single thread through the task control block list. Looking through Knuth's sorting algorithms [5] the best choice for small N is list insertion. However, his algorithm is for sorting arrays that have a thread through them. Nevertheless, I feel that the idea of sort by insertion in a list is so simple that with very small N (under 16) the algorithm chosen is efficient. Experience has shown no adverse affects using this algorithm.

The results were neither as bad nor as good as I had expected. The cost of having the scheduler as a separate process proved not to be a factor since it only had to do a single context switch every 1/2 second or so. The size of the scheduler process (2700 octal bytes) was not a particularily important factor either. As a small test I fired up 6 copies of a compute-bound program, except that after a few 100 million loops of no-ops they print the time. After about 5 minutes with the round-robin and timesharing scheduler, the counters in each of the 6

processes were not significantly different. Either the schedulers were equally efficient in this test or the ratio of scheduler to user state is so low that it is not worth considering. However, I did not try this with a large number of tasks which would have tested the sort algorithm for larger N.

Other tests were made subjectively. These had to do with how it felt to edit interactively at priority 50 while the 6 CPU-bound processes were running and 3 PIPs were doing [*,*]/li's to terminals. The editing was noticably smoother with the timesharing scheduler in use, provided that it did not get swapped out. If it did get swapped, then response was so terrible that it became evident that the swapping algorithm would need modification or that the editor would have to be run at a higher priority to defeat the swapping. This would also, of course, defeat the scheduler. The action of the PIPs on the other terminals were smoother with the timesharing scheduler in this test since the editor and all three PIPs almost always stayed at the top of the priority queue. (I had to cheat a little to determine this by running yet another higher priority monitoring program). Our interactive paint programs also ran more smoothly. These programs poll the tablet 30 times a second with 2 clock tick mark times to do non-busy waits. A user can now paint without the brush getting ''stuck'' every time the program does a litte computing while at the same time not locking out all other processes every time a lot of computing is necessary. Since the timesharing scheduler did not violate any conservation laws, the compute-bound tasks had to suffer some and they did in fact suffer the effects (about 5-10 percent) of not running when an interactive process was ready. However, the algorithm was fair in assigning time to the compute-bound tasks which all got the same amount of work done.

In conclusion, a user-written timesharing scheduler is a useful investment for a system that has a particular kind of need. However, if your system is not experiencing very bad performance and you do not have a compelling reason to implement this type of scheduler, I suggest you stick with DEC's modified round-robin. However, the ability to tune the parameters interactively is essential. If I could have done this easily to the round-robin scheduler, the results might have been different. Since the overriding issue on our system is still limited space, swapping is more of a problem than anything else and with our gradual migration to the VAX, it is not worth having still another operating system patch to worry about; therefore, we are not going to install this scheduler. Fortunately, I had a good systems programming language (C) to use: hence this experiment only took a couple of weekends to implement - less time in fact than it took to write this paper. Had I been forced to use MACRO-11, I would have had to spend considerably more than two weekends. It would have then been a very costly experiment and it would have been difficult to make the decision to bail out.

References

1. Bobrow D. G., Burchfiel J. D., Murphy D L. and Tomlinson R. S. "Tenex, a paged time sharing system for the pdp-10" Cacm March 1972 Volume 15 Number 3 page 135.

2. Ritchie D. M. and Thompson K. "The Unix time-sharing system" Cacm July 1974, Volume 17, Number 7. pp 365-375.

3. Thompson K. "Unix Implementation" Bell System Technical Journal 57(6) 1978 pp 1935-1937.

4. "Vax software hand book" Digital Equipment Corporation. 1981. Chapter 14.

5. D. Knuth "The Art of Computer Programming Vol 3" 1973 Section 5.5 pp 379-381.

Below is the Sort program. It is written in C with a little help from the macro processer MP. The addresses of the locations containing the tunable constants are set up with DEFINE statements. There is another program that inserts the patches and sets up the values I ended up using. Only an extract of that program is shown.

```
/*
        Timesharing Scheduler for RSX11M

        Build instructions:

           (Vd14: is where the rsx objects are kept.)

           sched/cp/pr = sched,schedm
           vd14:[1,24]mcr/lb:lklst
           vd14:[1,1]exelib/lb
           vd14:[1,54]rsx11m.stb/ss
           [1,1]jplclib/lb
           /
           stack=200
           task=sched
           pri=75
           //


        This process does no I/O, so don't include standard
        definitions; but we need max, min, and abs:

*/

#define max(x, y)       (((x) < (y)) ? (y) : (x))
#define min(x, y)       (((x) < (y)) ? (x) : (y))
#define abs(x)          ((x < 0) ? -(x) : (x))


#define SCALE    100            /* factor for scaled arithmetic */
#define ADJUST   0
#define CLEAR    1

                                /* The ADDRESSES of the constants */
#define WAITA    0117710        /* interval between sorts         */
#define THA      0117712        /* Threshold - constant 2         */
#define DIVIDEA  0117714        /* divide by - constant 1         */

#define TIMESB   0117720        /* times before using 2nd set     */
#define THB      0117722        /* Threshold - constant 2'        */
#define DIVIDEB  0117724        /* divide    - constant 1'        */

#define DIVIDEC  0117726        /* blocking divide constant 3     */

/*
        macro to get a tunable constant
*/
#define Valof(x) ((unsigned int)  *(ppp = x) )
```

```
char **ppp = 0;

#include "eric.h"
/* eric.h is because I have my own preferences on syntax */

 I   +----
 n   :  #define function  /*  this is just used by the indenter  */
 c   :                    /*  and to make finding functions easy */
 l   :  #define then      {
 u   :  #define elsif     } else if
 d   :  #define els       } else  {
 e   :  #define fi        }
 d   :
     :  #define forever   for (;;) {
     :  #define rof       }
 f   :
 i   :  #define or ||
 l   :  #define and &&
 e   +----

#define z if(1==0)
#define NULL 0
#define Count(a) (( (int) a->t_count) & 0377)
#define NextTask(a)  (a->t_actl)
#define Priority(a) (    ((int) (a->t_pri))  & 0377    )
#define Intof(a)    (    ((int) (a))  & 0377    )

typedef char Byte;
typedef short int Word;
struct atl {                          /*  Active task list node  */
        struct atl *t_lnk;
        Byte t_pri, t_ioc;
        Word t_cpcb, t_nam[2], t_rcvl[2], t_astl[2], t_eflg[2],
             t_ucb, t_tcbl, t_stat, t_st2, t_st3;
        Byte t_dpri, t_lbn[3];
        Word t_ldv, t_pcb, t_mxsz;
        struct atl *t_actl;           /*  active task list thread  */
        Word t_sast, t_att[2], t_off;
        Byte t_count, t_srct;         /*  count is US  */
        }
unsigned int bmask = 0;

typedef struct atl *Ptr;    /*  create new type "pointer to atl node"  */

   /*
        ------------------------------------------------
        Main - Get some assembly constants and drive  the
        sorting,  reset,  and  nonbusy  wait  procedures.
        Place lock and unlock around what WE do.
        ------------------------------------------------
    */
function main(argc,argv) int argc; char *argv[]; {
        char buf[10];
        register int n;
        register unsigned int cnt;
```



```
        Ptr lh,p,q,a,b,fst,getlh();

        lh = getlh();
        bmask = getmsk();       /* get task blocking bits */
        p = lh;
        cnt = -1;
        forever
             wait(Valof(WAITA));
             lock();
                reset(lh,ADJUST);
                if (cnt >= Valof(TIMESB)) then
                   reset(lh,CLEAR);
                   cnt = -1;
                fi
                cnt++;
                n = find50(lh,&fst);
                SortAtl(fst,n);
             unlock();
        rof
}

   /*
        ------------------------------------------------
        Find50 - Find  the  first  non  blocked  task  of
        priority 50 that has a non zero count (or blocked
        with  count  greater than 5). Return value is the
        number of tasks to be  sorted,  also  return  the
        pointer to the previous of first (in fst).
        ------------------------------------------------
    */

function find50(lh,fst)  Ptr lh,*fst; {

        register Ptr p,q;
        register int n;
        q = p = lh;

        forever
            q = p;    /*  previous task (back pointer)  */
            p = NextTask(p);

            if (p == NULL) then
                return 0;    /*  nada to sort if we reach null task */
            fi

            if (Priority(p) == 50 and  Count(p) > 0 ) then
              if (p->t_stat == 0  or (((unsigned)(p->t_st2)& bmask)==0
                        or Count(p) > 5)) then
                   break;    /*  keep looking till first nonzero pri=50  */
              fi             /*  that is not blocked or count > 5        */
            fi
        rof

        *fst = q; n = 1;
```

```
        forever
            p = NextTask(p);
            if (p == NULL or Priority(p) != 50) then
                break;
            fi
            n++;    /*  keep counting till below 50 or null task  */
        rof
        return n;
}
    /*
        ------------------------------------------------
        SortAtl  -  Sort the Active task list for n tasks
        in task list using lfst as a listhead. Its a node
        that has a pointer to the first task to sort.
        ------------------------------------------------
     */
function SortAtl(lfst,n)    Ptr lfst; int n; {
    /*
        a,b,c   are  pointers  to  3  consecutive  tasks;
        necessary because the thread  is  singly  linked.
        aa,bb are similar.
     */

        register Ptr a,b,c,aa,bb;
        int cb,i,j;

        if (n < 2) then          /* must be enough work or just leave */
            return;
        fi
        a = NextTask(lfst);
        for (i=2 ; i <= n ; i++) {    /*  for the 2 thru the nth task  */

            b = NextTask(a) ; cb = Count(b); c = NextTask(b);
            aa = lfst;
            for (j = 2 ; j <= i ; j++) {    /*  find its slot up front  */
                bb = NextTask(aa);
                if ( cb < Count(bb) ) then
                    a-> t_actl = c;     /*  it needs to be moved  */
                    aa-> t_actl = b;    /*  to a higher pri spot  */
                    b-> t_actl = bb;    /*  nearer the front      */
                    b  = a;
                    break;
                fi
                aa = bb;
            rof
            a = b;
        rof
}
```



```
    /*
        ------------------------------------------------
        Reset  -  reset  counters  in  each  active  task
        according  to  our formula and whether or not the
        task is blocked. We use 1 of 2 formulas depending
        on ztype.
        ------------------------------------------------
     */
function reset(lh,ztype) Ptr lh; int ztype; {
        register Ptr p;
        register int div,thr,divc;

        if (ztype == ADJUST) then    /* get constants   */
            thr = Valof(THA);        /* regular   set   */
            div = Valof(DIVIDEA);
        els
            thr = Valof(THB);        /* alternate set   */
            div = Valof(DIVIDEB);
        fi
        divc = Valof(DIVIDEC);       /* blocking factor */
        p = lh;                      /* start at list head */
        forever
            p = NextTask(p);
            if (NextTask(p) == NULL) then
                break;
            fi
    /*
        Apply  adjustments  to our count field. First the
        regular factors then see if the task  is  blocked
        and apply that as well.
     */

            p->t_count = max(((Intof(p->t_count)*SCALE)/div - thr),0);

            if (p->t_stat != 0  or
                    (((unsigned)(p->t_st2)& bmask) != 0) ) then
                p->t_count = (Intof(p->t_count)*SCALE)/ divc;
            fi

        rof
}
```

Since some things can only be done in Macro-11 under Rsx, I had to have some help; my macro processor was designed to aid the writing of C programs, the macros function and ret should be self explanatory.

```
        program atlhelp

        .mcall mrkt$s,wtse$s,exit$s


function lock
```

```
        call $lockl
        ret

function unlock
        call $unlkl
        ret

function getlh
        mov $tskhd,r0           ;get the task list header
        ret


function getmsk
        mov #t2.spn!t2.wfr!t2.stp!t2.sef!t2.tio,r0
        ret             ;return blocking mask for t.st2


function wait[n]
        mrkt$s #25. n(r5) #1.   ;n ticks
        wtse$s #25.
        ret

start:                          ;use a tiny main program since we
        call main               ;want this process to be as small
        exit$s                  ;as possible

        .end    start
```

The following is an extract from the program that loads the patches. It also disables round-robin scheduling.

```
        patch=117500            ;patch space starting addr
        timer=22364             ;timer patch location
        patch3=23112            ;disable round-robin location
        data3=22352             ;round-robin data cell
        code3=463               ;br swap
        code3a=3063             ;bgt swap  (dec used bne)
        parms=117710            ;parameter area
code:
        mov @#$tktcb,r4         ;;;incr counter for current task
        if t.actl(r4) ne 0      ;;;but check for null task
            incb   t.off+2(r4)  ;;; even though we never get
        fi                      ;;; there (fire wall)
        mov #$intct,r4          ;;;support patched location
        rts pc                  ;;;return to main line interrupt
code1=.-code

code2:  jsr pc,@#patch

def:                            ;default parameter values
        .word 32.               ;0117410 =  sched interval in ticks
        .word 0                 ;0117412 =  thresh a
        .word 105.              ;0117414 =  divide a
        .word 16.               ;0117416 =  sch display wait in ticks
        .word 5.                ;0117420 =  b parm count down
        .word 0.                ;0117422 =  thresh b
        .word 140.              ;0117424 =  divide b
        .word 1000.             ;0117426 =  divide blocked
def1=.-def

load:
        moveb[#code,#patch,#code1]
        printf "Loaded"
        return
dopatch:
        mov @#code2,@#timer
        mov @#code2+2,@#timer+2
        mov #code3,@#patch3
        printf "Patched"
        return
unpatch:
        mov #5,@#data3
        mov @#code,@#timer
        mov @#code+2,@#timer+2
        mov #code3a,@#patch3
        printf "Unpatched"
        return
initit:
        moveb[#def,#parms,#def1]
        printf "Inited"
        return
```

Since the display program we run at priority 1 is useful in its own right, I list it as well. Most of the code is to handle bad switch settings without the program aborting.

```
        .enabl lc
        .title disp             ;display lights
        .ident /v1/
;
;       directive macros.
;
        .mcall  dir$            ;issue directive
        .mcall  svtk$s          ;setup sst vector table
        .mcall exit$s   .

        .psect disp
start:
        svtk$s  #ssttbl,#sstsiz ;setup sst vector table
go:     mov @#177570,r0         ;get addr from switch register
        mov (r0),@#177570       ;and display contents on leds
        br go                   ;need a few of these since a
        br go                   ;trap may restart us at different
```

```
        br go                   ;places.
        br go
;
;       stt vector table.
;
ssttbl: .word   err0            ;odd address
        .word   err6            ;memory protection
        .word   err0            ;t-bit trap or bpt
        .word   err0            ;iot trap
        .word   err0            ;reserved instruction
        .word   err2            ;non-rsx emt instruction
        .word   err2            ;trap instruction

sstsiz  = <.-ssttbl>/2

err6:                           ;add 6 to clear stack
        add #2,sp
err4:                           ;    4
        add #2,sp
err2:                           ;    2
        add #2,sp
err0:
        rti                     ;return


        .end start
```

The task build commands:

```
        disp/pr=disp
        /
        task=disp
        pri=1
        units=0
        stack=64
        /
```