The DeVIAS Letter

The RSX Multi-Tasker

June 1984 Issue

Printed in the U.S.A.

IAS

Did not submit material for this issue

Send your submissions for the next issue to:

IAS NEWSLETTER EDITOR

DECUS
249 Northboro Road
BPO2
Marlboro, MA 01752

RSX MULTITASKER

Table of Contents

FROM THE EDITORS

The supply of new articles seems to be dwindling again as the June Symposium approaches and people get busy getting ready for it. Those of you who are doing any sessions: if you have handouts, whether or not they made it into Session Notes, please send me a machine readable copy.

My RSX11M-Plus system is going away in two weeks, traded in for a VAX. We will still be running RSX on two other 11/70's but it is a closed system based on M V3.1. I used to spend lots of time sorting through the RSX SIG tapes, finding all the goodies, and documenting them for our programming staff, but now I will no longer be doing so. I am sure that there are many of you out there who have been through the latest RSX tapes and found things that you liked and that you use. If you have written anything for your own use about them, or if you could write something, please send it along so other users can be spared the searching through so many directories. Even just a paragraph on what something does and why you like it (or why you don't). I would even appreciate your sending me documentation files right off the SIG tapes for stuff you think everyone should know about; neither Dominic nor I has any occasion to even read the RSX tapes anymore, sad to say.

Just think: you can get your name in the Multi-Tasker as a submitter just by copying someone else's stuff off the tape and mailing it to us!

Another possible article source: if any of you publish LUG newsletters, please send us copies. The North Texas LUG dials in every month and sends their newsletter directly into my VAX, for which we thank them profusely. We have used several pieces from their publication; there is one in this issue.

1.0 ARTICLE SUBMISSIONS

All submissions for articles should be sent in machine-readable form. Magnetic tape at 800, 1600, or 6250 BPI is acceptable. VMS BACKUP, RSX BRU, PIP, FLX ... we're flexible. RX01 or RX02 floppies are fine, too. The contents of the files should be either in Runoff source format or straight text; please don't send only Runoff output because we may need to re-format. We have no time whatsoever to type in material sent in hard copy; if you send it, we will just fire back a letter asking for a tape or floppy, so please send machine-readable stuff, even if it is just one page.

We intend to submit the machine-readable files to the RSX SIG tape so that you can simulate getting back issues and reprints. Neither we or DECUS has any ability currently to provide back issues other than for past years, which are available from DECUS on microfiche. If you can find someone who will copy an issue and ignore the copyright notice, the decision is up to you...DECUS is working on the legal implications now but the results probably won't be implemented until 1986. Until then, strictly speaking, it is illegal to make copies of the Multi-Tasker.

If you have the ability to dial in to another computer you can send the material over the phone line at 1200 baud into my VAX. Please call me at (201) 646-4111 for the dial-up number and password.

Articles can be mailed to DECUS c/o "Multi-Tasker Editor" or sent directly to us at:

Allen A. Watson
THE RECORD
150 River Street
Hackensack, NJ 07602

Dominic J. DiNollo
Loral Electronic Systems
Engineering Computer Center
Ridge Hill
Yonkers, NY 10710

Speak Out

A Reply to Bruce Mitchell's article, "Whither Goest the SIG?"

Roger Jenkins
Wycliffe Bible Translators
19891 Beach Blvd.
Huntington Beach, CA 91648

Bruce brought up many excellent points in his article for which he is to be commended: The enduring architecture of the PDP-11, the flexibility and success of RSX, the fact that RSX can't rest on its past laurels, the fact that we aren't going to get any help from P/OS or the micros, the growing apathy of RSX users and

suggestions of future directions and projects. Bruce stated these points very articulately and I doubt I could add anything significant to them.

However I would like to bring up the other side of the "VAX issue". We run a data processing shop and for the past 8 years or so have used RSX on an 11/70. In recent years, our applications have grown both in number and size. Our user community has doubled and system response has degraded. We added more memory (1 1/2 Megs) to eliminate swapping only to find out that there was not enough pool space to support all of the tasks we wanted to run. It was common for PMT to poke its head up 5 or more times a day. RSX 11M+ might have solved this problem had it been the only problem, but read on.

In the program development world things were equally unacceptable. Our programmers spent 10% to 25% of their time managing overlays (we have large COBOL programs with FMS and RMS). Compiles could take up to 10 minutes and taskbuilds up to 20 minutes. If several compiles and/or taskbuilds hit the system at once, RSX would come to its knees.

RSX timeshares, but I would not say that it does it well. In a timesharing environment one would want a high priority job to get more of the CPU than a low priority job, but not ALL of it as RSX does.

We were at the point where we recognized that we needed to upgrade and replace much of our software because our buisness had changed and the programs were old. We looked around for packages that we could buy and found that there were no buisness packages that would suit our needs available on RSX. We expanded our search to RSTS and still found nothing. We finally found what we wanted and you would probably never guess what it ran on: a VAX.

We purchased a 750 and have been extreemly pleased with it. We have never had a system crash due to low pool. Overlays are a thing of the past. COBOL compiles are 10 times faster on the VAX than on RSX as are taskbuilds (links). A low priority job can be consuming vast quantities of machine resources and the interactive users can hardly tell that it is running due to what I think are superior scheduling techniques on the VAX. The VAX hardware is far more reliable than the 11/70; all computers should run so well. Oh yes, one more: The VAX has a backup program that really works. (Side note: Another SIG project would be to migrate BACKUP to RSX!) Yes, the VAX is expensive, but it does what we need; RSX did not.

I think I have gone astray from my original intention, but Bruce would probably have been disappointed had I not said something about VAXes. My purpose is not to throw rocks at RSX. I still think it is a fine system, but I do not think it is a buisness system. To use an analogy, we as RSX users need to

recognize that someone has invented a new tool that does some of what our old tool used to do, and does it better. We have been pounding in screws with our hammer for a long time and don't understand why anyone would want to put screws in any other way. But someone has invented a screwdriver. That is not to say that the hammer is no longer needed. It is still desperately needed for pounding nails. The screwdriver is never going to pound nails like our hammer will. But neither does that mean that the screwdriver is a useless tool.

Regarding the future of the SIG: There is still a big need for it and Bruce has mentioned some excellent needs that the SIG can fulfill. Money talks. If DEC does not give us what we want then perhaps someone out there is smart enough to start his own company to supply the upgrades to RSX that we seem to want so badly. Or do we want them bad enough to pay for them? Maybe we only want them if DEC will give them to us for "free".

Now here is the really important part: It is my opinion that the VAX will not kill the RSX SIG. If the RSX SIG dies, it will be because we kill it ourselves. We only have so much energy to expend on the SIG. If all we do is complain and cry about the VAX then it is all over for us. No one wants to go to RSX symposia sessions or LUG meetings and listen to what is wrong with the VAX. Would it not be better to accept that fact that like it or not, the VAX is here and emphasize what we (RSX and PDPs) do best? I think therein lies our hope.

HELP YOURSELF

Virtual Program Section Problem

R. E. Beverly III, President
R. E. Beverly III and Associates
1891 Fishinger Road
Columbus, Ohio 43221
(614) 457-1242

Dear sir:

We recently migrated to RSX-11M and FORTRAN-77 from a RT-11 environment and were naturally anxious to utilize those features of RSX-11M which permit management of large tasks (Overlay Descriptor Language, etc.). The use of Virstul Program Sections (VSECTs) would be particularly helpful and is described on pages 5-53 to 5-61 of the Task Builder Manual (AA-L680B-TC). As a starting point, we tried the sample program VSECT.FTN given on pages 5-59 and 5-60, resulting in ERROR =-85 (IE.WOV, address window allocation overflow) in the call to CRAW. We could not identify any obvious error in the coding or task building. The program ran, however, by declating two extra address windows (WNDWS = 2) during task building instead of one. The reason for this is unclear. Your help or that of the Multi-Tasker readers in solvintg this problem would be greatly appreciated. A tutorial for high-level language programmers on the use of VSECTs with region creation, address windows and window mapping would be most welcome.

We are running RSX-11M Version 4.1, Update A on PDP-11/23 and PDP-11/60 systems, both having 128KW memory.

Sincerely yours,
R. E. Beverly III, Ph.D.

Power Failure Recovery -- a comment

D. J. Armstrong
Bell Canada
HQ Engineering (Network Technology)
Room 1855, 160 Elgin Street
Ottawa, Ontario K1G 3J4
(613) 239-4183

We had the same problem reported by Jim Neeland in the RSX Question and Answer Session as reported in the November 1983 Multi-Tasker (Question No. 45). The problem is that after power fail recovery, RL02 disks must be re-mounted under RSX11M Plus V2.0.

Attached is DEC's response to our SPR.

NOTE

NOTE FROM EDITOR This patch applies to RSX-11M-PLUS Version 2.0, and should not be applied to any other version.

RESPONSE:

Thank you for bringing this problem to our attention. It will be fixed for the next release of RSX-11M-PLUS. I have enclosed a solution to the problem which is a little bit different from your solution. To correct the problem, please create the following file and name it DLDRV.ZAP.

```
[1,54]DLDRV.TSK
2:120000;0R
0,456/
1022V
1014
X
```

To apply the patch, copy the DLDRV.TSK:

```
>PIP [1,54]DLDRV.TSK;2=[1,54]DLDRV.TSK;1
```

And invoke ZAP to patch it:

```
>ZAP @DLDRV.ZAP
```

Then in VMR, remove the old RL02 driver and load the new one:

```
>SET /UIC=[1,54]
>VMR
Enter filename:RSX11M
VMR>UNL DL:
VMR>LOA DL:
VMR>^Z
```

And finally reboot the system.

Driver for MAP Array Processor

Elizabeth B. Uptegrove
XYBION Corporation
240 Cedar Knolls Road
Cedar Knolls, NJ 07927-1195
(201) 538-5111

We have a MAP array processor from CSP, Inc., with a driver for RSX11M. We are converting to RSX11M-PLUS and CSPI is not able to supply us with an RSX11M-PLUS driver.

Does anyone have one or know where I can get one?

Please contact me at the above address or telephone number.

Response from Editor

If you can get the source for the RSX11M driver from CSPI, there is a short description of how to convert a driver from M to M+ in the RSX-11M-PLUS GUIDE TO WRITING AN I/O DRIVER, Appendix B. It is really fairly simple to do; I converted a driver that came on a SIG tape from M to M+ with no previous experience at writing drivers.

HOUSTON LUG SYMPOSIUM AUG. 16-17

The Houston LUG will be having a LUG symposium; readers nearby may be interested. The Symposim will be August 16-17 1984 at the Adam's Mark in Houston Texas.

Anyone interested in more information should contact:
Mike Sievers
c/o Computer Services Unlimited
8830 South Gessner Mail Stop 314
Houston, Texas 77036

At the last meeting they had 300 people and ran four parallel sessions.

It's on the SIG Tape

TAT - Task Attribute Utility

Contribution from the North Texas LUG Newsletter

Editor's Note: We assume that this program has appeared on a recent SIG Tape or will appear on a SIG Tape in the near future.

TAT allows for the display and update of task parameters in an uninstalled task image, eliminating the need to re-taskbuild a task to change a task parameter. It is specifically useful with tasks which are not normally left installed.

USAGE:

TAT is invoked in the same way as other RSX system tasks, i.e. by typing 'TAT' or 'RUN $TAT'. TAT will then ask for a command line by typing the prompt 'TAT>'. The user then enters a command line in the following format:

filespec[/sw[:val:val]]

where:

filspec - is any legal RSX file specifier, excluding wildcards. Device SY: and file extension .TSK are defaults.

[/sw[:val:val] - are optional switches and optional switch values from the following list:

- /TASK:taskname - where 'taskname' is a legal RSX task name
- /PAR:parname - where 'parname' is a legal RSX partition name

- /UIC:grp:mem - where 'grp' and 'mem' are group and member, respectively, of the task's UIC. (the colon separating group and member is required syntax)

- /PRI:pri - where 'pri' is the task's default priority in octal. A decimal number may be specified by following it with a decimal point.

- /INC:inc - where 'inc' is the task extend size, in octal words. A decimal number may be specified by following it with a decimal point.

- /ASG:ddnn:lun - where 'ddnn:' is a device or pseudo device name (e.g. SY0:), and 'lun' is a logical unit number. The lun specified must be less than or equal to the total number of luns specified at task build by TKB's UNITS option. Up to 5 ddnn:lun pairs may be specified with the /ASG switch. Each pair is separated by a colon. (see examples)

- /CKP - enables task checkpointability

- /LI - lists on the invoking terminal the current values of the above task parameters (including all assigned luns). The /LI function is performed after any task parameters specified on the command line have been updated, therefore, the new values are displayed. /LI is the default, thus a parameter listing is always provided, except when /-LI is specified.

- /NOLI or /-LI - suppresses the listing

VTM -- Video Terminal Message Program

Allen A. Watson
Multi-Tasker Editor

This article is a very quick adaptation of the VTM HELP file to document form. VTM is a program which has appeared on a couple of past RSX SIG tapes which I have found to be very useful in generating fancy-looking screen displays quickly. I know for certain that it was on the Fall 1982 tape in [351,74], submitted by Vincent Graham. No doubt most of the document that follows was written by Mr. Graham.

At THE RECORD we have continued to find VTM useful on our VAX systems. It is very easy to use and I recommend it highly.

2.0 PURPOSE OF VTM

VTM is a program used to format messages for a VT100 terminal. Various options are available and are listed below. If no options are specified, the default is to clear the screen and write the message in double height characters centered in the middle of the screen.

3.0 USING VTM

VTM is an installed task, so it may be used directly from MCR. It accepts its commands either from the command line or from an indirect command file.

3.1 STARTUP OF VTM

There are three methods of starting VTM; they are:

1.
```
$ VTM
VTM>
```

   This is the interactive mode. VTM continues to prompt for input until you type CTRL/Z to exit. You must use either this form or the indirect command file form to pass any commands other than a simple message to VTM.

2. $ VTM message_text

Clears the screen, writes the message in the middle of the screen, and then exits. Because MCR translates case of parameters, you cannot use any of the special VTM commands in this mode.

3. $ VTM @command_file

Using this method, command lines are read from the specified command file. The length of the command lines can be up to a maximum of 256 characters. The continuation character "-" is allowed at the end of a line. If the extension isn't specified the default is .VTM.

3.2 COMMAND LINE FORMAT

Command lines are of the form:

{option, option, ...} message_text {option, option, ...} etc.

Where message_text is the text to display. If contained in a command file, upper and lowercase characters are accepted. The command line interpreters (MCR and DCL) convert lowercase to uppercase. The message is truncated if the length is too long.

The braces MUST be specified around the options. Multiple options must be separated with commas. The options can be abbreviated to one or two characters.

3.3 AVAILABLE VTM COMMANDS

ALL	ACSET	ANSI	BLINE	BELL	BLINK	BOLD	BOX
CLEAR	COLUMN	CENTER	CBOL	CEOL	CLIN	CBOS	CEOS
DATE	DELAY	DHEIGHT	DWIDTH	Format	JUMP	HOME	ITERATION
KEYPAD	LINE	LED	LJUSTIFY	LMARGIN	NARROW	OUTPUT	OFF
PLINE	REVERSE	REGION	RESET	RJUSTIFY	RMARGIN	RULER	TLINE
SKIP	SCREEN	SGRAPHICS	SMOOTH	Startup	TIME	TEST	TERMINAL
TTY	UNDERSCORE		UKSET	USSET	WIDE		

4.0 VTM COMMANDS

Each of the following commands may be given inside the enclosing braces ({}). See the second above on COMMAND LINE FORMAT for more information.

4.1 ALL

Turns on all attributes except underscore. The attributes turned on are BOLD, BLINK, and REVERSE.

4.2 ACSET

Selects the alternate character set. Our terminals do not have the alternate character sets but they are useful for special affects such as boxing a message.

4.3 ANSI

Sets the ANSI mode. The VT100 MUST be in this mode for VTM to work properly.

4.4 BLINE

BLINE=n

Specifies the bottom line number. This is the line number where wraparound to the top line occurs. The default bottom line number is 24.

4.5 BELL

Inserts a bell in the message. Multiply bells can be inserted in each message by using the command: {BELL, BELL}

4.6 BLINK

Turns on the blink attribute.

4.7 BOLD

Turns on the bold attribute.

4.8 BOX

BOX=topline:bottomline[:leftcolumn:rightcolumn]

Draws a box on the screen using special graphics. The default line and column numbers are {BOX=1:22:1:80} for VAX/VMS and {BOX=1:23:1:80} for RSX-11M. If just {BOX} is specified, the previous parameters are used.

4.9 CLEAR

CLEAR or -CLEAR

Enables or disables clearing the entire screen. The default is to clear the screen once before the first message is written.

4.10 COLUMN

COLUMN=n

Use column number 'n' for the next message.

4.11 CENTER

CENTER or -CENTER

Enables or disables centering messages in the middle of the screen. The default is to center messages.

4.12 CBOL

Clears to the beginning of the line.

4.13 CEOL

Clears to the end of the line.

4.14 CLIN

Clears the entire line.

4.15 CBOS

Clears to the beginning of the screen.

4.16 CEOS

Clears to the end of the screen.

4.17 DATE

Outputs the current date in the format DD-MMM-YY.

4.18 DELAY

DELAY=n

Delays for 'n' seconds. The delay is immediate. If it is specified on a line with a message, the message is displayed after the delay. The delay has no effect on output to a file. It's useful only in VTM command files.

4.19 DHEIGHT

DHEIGHT or -DHEIGHT

Enables or disables double height messages. Once specified, the next and subsequent messages are written in double height.

4.20 DWIDTH

DWIDTH or -DWIDTH

Enables or disables double width messages. Once specified, the next and subsequent messages are written in double width.

4.21 JUMP

Sets the jump scroll mode.

4.22 HOME

Writes the message starting in the top left corner of the screen. This is equivalent to command {L=1, C=1}. The screen is not cleared.

4.23 ITERATION

ITERATION=n

Specifies an iteration count for the next or current command line. If you specify a count of 65535, you will loop infinitly and you must type CTRL/C to exit from VTM.

4.24 KEYPAD

KEYPAD=ON

The ON state is known as the application mode. In this mode, the keys on the keypad transmit an escape sequence.

KEYPAD=OFF

The OFF state is known as the numeric mode. In this mode, the keys on the keypad transmit the number on the key. Your keypad is normally in the numeric mode.

4.25 LINE

LINE=nn

Specifies the line number for the next message. On startup, the default line number is 12.

4.26 LED

LED=n or -LED

Turns on the specified LED (L1 - L4) on the terminal. The The range of 'n' is 1-4. If 'n' isn't specified, all the LEDs are turned on. A {-LED} command turns all LEDS off.

4.27 LJUSTIFY

Left justify the next message. Same as the command {C=1}.

4.28 LMARGIN

LMARGIN=n

Sets the left margin to 'n'. The default is 1.

4.29 NARROW

Sets the screen to 80 column mode. The VT100 automatically clears the screen when changing column modes.

4.30 OUTPUT

OUTPUT=file.ext

Outputs to the specified file. If running interactive, messages are output to both the terminal and the file. The -TTY command will inhibit output to the terminal. If no output file is specified (i.e., {OUT=}), the default file name of VTM.DAT is used. If a second {OUT=...} is specified, the current file is closed and the new file is opened. This mode can be useful to create data files which can then be printed with a single "TYPE" command.

4.31 OFF

Turns off all the attributes (BLINK/BOLD/REVERSE/UNDERSCORE). All attributes are turned off because the VT100 does not have the capability to turn off a single attribute. This command is executed automatically at the end of each message output if you havn't turned the attributes off.

4.32 PLINE

PLINE=n

Specifies the prompt line number. The default prompt line is 24 for RSX-11M and line 23 for VAX/VMS. This is also the last line output to a file when the file is closed.

4.33 REVERSE

Turns on the reverse video attribute.

4.34 REGION

REGION=topline:bottomline or -REGION

Enables or disables the scrolling region. When specifying a scrolling region, the top line must be greater than the bottom line. To disable the scroll region, use {-REGION} or {REGION=1:24}.

4.35 RESET

This does a power-up reset. It is useful in command files being aborted by CTRL/C or CTRL/Y when the state of the VT100 is unknown.

4.36 RJUSTIFY

Right justify the next message. The message will be output so the last character lines up with right margin.

4.37 RMARGIN

RMARGIN=n

Sets the right margin to 'n'. The default is 80.

4.38 RULER

RULER=n

Write a ruler on the VT100 to aid in message formatting. The default line number (n) is 1.

4.39 TLINE

TLINE=n

Specifies the top line number. This is the line started at after wrapping around from the bottom. The default top line number is 1.

4.40 SKIP

SKIP=n

Skips 'n' lines on the screen. After displaying a message, VTM automatically adjusts the line number to the next line. This option is used for additional spacing.

4.41 SCREEN

SCREEN=BLACK or WHITE

The BLACK state causes the screen to be black with white characters. The WHITE state causes the screen to be white with black characters.

4.42 SGRAPHICS

Turns on special graphics. In this mode, the lowercase character set is used to display special graphic characters. Use either {UKSET} or {USSET} to exit special graphics mode.

4.43 SMOOTH

Sets smooth scroll mode.

4.44 TEST

Fills the screen with "Es".

4.45 TERMINAL

TERMINAL=ddnn:

Specifies the terminal to write the message to. On VAX/VMS, you can only write to terminals which are not logged on. Logical names are permitted of the form ddnn: (RSX standard). The double underscore on terminal names on VAX are stripped.

4.46 TIME

Outputs the current time in the format HH:MM:SS.

4.47 TTY

TTY or -TTY

Enables or disables output to the local terminal. The default is to always output to the local terminal. On RSX-11M, the output device is TI0:. On VAX/VMS, the output device is wherever SYS$OUTPUT is directed (normally the terminal).

4.48 UNDERSCORE

Turns the underscore attribute on.

4.49 UKSET

Selects the United Kingdom character set.

4.50 USSET

Selects the USASCII character set.

4.51 WIDE

Sets the screen to 132 column mode. The VT100 automatically clears the screen when changing column modes.

Working Group News

Jeff Hamilton
Working Group Coordinator
(214)457-4175

Date of this report: 11MAY84
The working group chairmen are as follows:
RSX-11M Unsupported Versions:
  Bill Burton
  Texas Research Institute
  1300 Moursand
  Houston, Texas 77030
System Performance and Accounting
  Paul Sorenson
  AEP, Interactive Graphics
  1 Riverside Plaza
  Columbus, Ohio 43215
DECUS Library
  Bruce Zielinski
  RCA
  Marne Highway M/S 138-2
  Moorestown, N. J. 08057
SIG Tape Collection
  Glen Everhart
  RCA Government Systems Division
  Route 38
  Cherry Hill, New Jersey 08358
SRD
  Bob Turkelson
  NASA/Goddard Space Flight Center
  Mail Code 935
  Greenbelt, Maryland 20771
Data Acquisition, Simulation and Process Control (DASPC)

RSX MULTITASKER

Allen J. Bennett
Clark Equipment Co.
P.O.Box 3000
Battle Creek, Mich. 49016

Runoff

Chuck Spalding
Adept Technology Inc.
1202 Charleston Rd.
Mountain View, Calif. 94043

The Unsupported Version working group is currently planning sessions for the Spring Symposium for past unsupported versions of 11M/11M+. An article is being submitted to the Multitasker about the current work being done by the working group.

The System Performance and Accounting working group is continueing its work in preparing the index of the past RSXSIG tapes in regards to System performance and accounting. There is a session being planned for the Spring Symposium for the System Performance and Accounting working group to discuss further work to be done. Anyone interested in system performance and accounting please attend the working group session. Paul reports that as of April 30 his job responsibilities will be changing such that he will have less day-to-day contact with RSX. If there is someone who would be interested in taking over the chair of the working group please get in touch with Paul.

The DECUS Library working group have continued efforts to construct a tape to provide to the DECUS library of the best software off of the past RSXSIG tapes. A form letter was sent out to the 60 members of the working group and 30 replies were returned. A restructoring of the tree for distribution of this tape is being done. These people are being used in the evaluation of a sample 2400' 800 bpi tape that will be submitted to the DECUS library. The RSX European branch will also be used to evaluate the software. They will be sent a tape as soon as a small correction is made to the tape. It seems that one of the programs has been marked not for export, thus limiting its shipment overseas.

According to Glenn Everhart the SIG Tape working group has not done much as most of the work has been done for the Fall RSXtape. Copies of the tape were to be sent to DECUS UK and DECUS Germany to the RSX people there. Also tapes were being made available to those regional people from the National LUG organization interested in distributing the tape in parallel with the tree. This will be continued, and LUGS will be able to get the tape using either (or both) path. Since forms will go out with the tapes in either case to allow them to be identified (the forms should be returned to Glenn Everhart upon receipt of the tape so the Working Group can account for who has the tape), a LUG will be accounted for no matter how it gets the tapes. LUGS will vote by electing to be part of the tree or not on how big the tree will be. In the meanwhile we will not break the tree and will use the LUG organization in PARALLEL to it. (The VAX SIG is going to use this

philosophy too, a change from earlier plans.)

A site has been lined up for the tape copy operation in Cincinnati. Thus The tape copy operation should thus go rather smoothly. A preliminary tree structure for distribution of the tapes has been made up from the participation cards submitted by LUG chairmen. This tree structure is preliminary and will undoubtly change as time progresses.

Glenn wants to remind everyone that Kitty Bethe, the Eastern Tree Coordinator, has many of the Sig tape packages running under P/OS. All the FCS stuff and most all privileged code for RSX runs unchanged under P/OS with simple relinking. Accordingly, a question has been placed in the form letter that goes out with the tree asking which LUGS have people interested in packaging SIG tape software onto RX50 locally. An article will appear in the Multitasker about this. Contributions are also being solicited to Tape Copy on RX50 or RX01/RX02 (or even TU58). We will only be able to copy tapes at the symposium, but other media will be returned later if submitted. They may be submitted BEFORE the symposium if possible too. This is the preferred course. Floppy contributions will be isolated somehow to allow their identities to be preserved for redistribution at LUGS from the tapes. The availability on the PRO 350 of the RSX SIG collection has made it important for PRO owners running P/OS to get involved in the RSX SIG; they will be able to learn tricks of getting software running on their PROs, and participate in one of the greatest software exchanges ever begun.

The SRD working group has continued work on the improved SRD supplied to the Fall 83 RSXSIG tape. Enhancements and continued bug fixes are being considered.

The DASPC working group has continued its efforts to lobby DEC for more real time development into RSX systems. This will continue into the future. Representatives from the LABS SIG, VAX Realtime Working Group and DASPC continued in their efforts to form a new SIG, but the status is unknown as of this writing. A seperate session is also planned for this working group and should provide for an open forum in this area. Nothing else is being done in this area, efforts being shelved till the Spring Symposium.

The Runoff group has continued its effort to consolidate desirable features of several versions of Runoff into an "official" version. Chuck, due to work efforts, is finding the time to upkeep the working group more and more difficult. If there is anyone willing to help in this position, please get in touch with Chuck. No sessions other than the general session is planned for the Spring Symposium.

If you are interested in providing information to a special working group concerning problems or ideas in that area, please get in touch with the working group chairman of that group.

Hints & Things

Installing QUE under another name

B. Z. Lederman Bankers Trust Co.

Like most users, we like to have the queue manager package running so we can spool print jobs out to various printers (most of which are attatched to terminal lines), and have batch jobs. The problem is that on most of our systems, there was a user application which installs under the task name ...QUE so that it may be invoked as QUE command: this was done many years before DEC came out with their package, and the users don't want to change. This means we can't install the QMGCLI task under the normal name of ...QUE, and this in turn means we can't do any normal QUE functions. The work around was to install QMGCLI at startup, start the queues, and then remove it an put in the user task. This allows queues, but you can't look at them or delete jobs off of them while the user task is installed. The obvious solution is to install QMGCLI under another name, and invoke it with that other name, but this won't work. QMGCLI is used for QUE and SUBmit and PRInt commands, so it checks itself to see what name it was invoked under: if you install it under some other name, it won't recognize any command you give it, as the task name it is invoked under is not one of the three it knows.

I finally decided to do something about it, by ZAPping the task image so it would accept itself under a different name. This turned out to be fairly simple. First, I made a copy of the task image under a different name: I decided to use QUX. Then I used the DMP utility to dump the task image, first in RAD50 and then in ASCII. I found the section of code where the three commands appear in ASCII: this may differ from system to system, but mine were in virtual block 11 (octal). If you dump the image, in this block you will see a file string with the UIC of [1,7], and then the string PRIQUESUBXXX followed by some more ascii characters and error messages. This can be accessed with ZAP as absolute or task relative. Again, this may vary from system to system, but mine looked like this:

```
-----------------------------------------------
11:614              2:6614          PR
11:616              2:6616          IQ
11:620              2:6620          UE      42525
11:622              2:6622          SU
```

11:624 2:6624 BX

After looking at this section with ZAP in ASCII, I replaced word 620 with 54125, which corresponds to the ASCII letters UX, so the command will change from QUE to QUX. I then looked at the section again in ASCII, to be sure I did not accidentally change something else, then exit ZAP and the job is done. The task is installed with a command like

INS $QUX/TASK=...QUX

and then any commands which used to use QUE will now use QUX: for example, QUE /LI becomes QUX /LI. Now it is possible to have the user written QUE and the DEC QUE (as QUX) in the system at the same time. ZAPping task images is not by first choice for solving problems, but without the source code, this is the easiest way out. Obviously, this could also be done to avoid any similar conflicts if someone was already using tasks installed as ...SUB or ...PRI.

I did run into a problem which is not directly related to this change, but which should be mentioned. When I first tried to invoke my new QUX, it told me that it could not access the queue file. I found that sometimes when QMG exits it leaves the file [1,7]QUEUE.SYS with file protection codes of [,,,] which means that no one can read the file. I get around this by having all of my queue start command files do a PIP [1,7]QUEUE.SYS;*/PR:0/NM so that the protection code is re-set to allow everyone to read the file. I also found that on occasion, if there is garbage in the beginning of the QUEUE.SYS file, then when you invoke QUE to create a queue, instead of creating it it appears to hang: if you look at QUEUE.SYS the number of allocated blocks will be getting larger and larger. The only way out I found was to abort QUE, abort QMG, delete QUEUE.SYS, and start over. I have not yet determined if this is a bug, or if QUEUE.SYS got corrupted during my experiments, or during an abnormal system shutdown, but I thought I'd mention it in case someone else has the same problem.

RSX Disk Optimization Guidelines Panel

Terry Medlin
GEJAC, Inc.
Riverdale, Maryland

Tony Lekas
Digital Equipment Corporation
Nashua, New Hampshire

Reported by Paul E. Triulzi

The intent of this session was to review the ODS-1 file structure, ACP's and to describe guidelines to improve disk performance. A discussion of disk I/O optimization was also presented.

The static structure of an ODS-1 Volume consists of an Index file, a Storage Bitmap, a Master File Directory (MFD), User File Directories (UFD), and a Bad Block File. The Index file points to the MFD, UFD's, the Boot block, Home block, Index bitmap, and File headers.

A file is uniquely identified by its File ID. The File ID consists of a file number,a file sequence number, and structure level. The structure level number is not used in ODS-1. As an example, the File ID 221,34,0 says that the file uses the 221st header in the Index file and that the header has been used 34 times previously.

A directory is just another file that happens to contain a cross-reference between File ID's and file names. A single file may be known by several names in several directories.

Files-11 currently has low level support for hierarchical directories. Neither FCS nor RMS support them.

The file header contains, among other things, a list of retrieval pointers (max of 102). Each pointer can map up to 256 contiguous blocks and uses a 24 bit address for the first block. Files-11 can map devices which have up to 8590 Mbytes but because of Storage Bitmap size ODS-1 is limited to 500 Mbytes.

One of the primary methods for improving disk performance is the INI command. The various switches available in this command allow the user to optimize disk access.

1.

The /EXT switch specifies how much space will be allocated every time you extend a file. The default is 5. Increasing this number could reduce the number of disk references that are needed to access a file.

2.

The /INDX switch specifies the location on the disk where you want to put the Index File. The Index File should be positioned to optimize the head movement of the disk. Sometimes, the center of the disk might be more optimal than the edges, depending upon the application.

3.

The /LRU switch specifies the approximate amount of directories that will be accessed concurrently. If the /LRU number is increased, the amount of disk I/O required for finding a directory may be decreased. The default is 3 which may not be sufficient for large systems with many users.

4.

The /WIN switch specifies the number of retrieval pointers the system uses for file windows. Increasing this number can speed up your disk access but it will also decrease the pool.

Another command that can optimize your disk performance is the MOUNT command. This command has a /ACP switch which specifies the name of the ACP to be used as the file processor for the volume being mounted. Multiple ACP's should be considered for performance reasons and pool space. Each ACP can store information about each disk, e.g. MFD, UFDs, and FCBs, which should speed access to the disk. If you use multiple ACP's, you may save some pool space since each ACP has its own buffer space. It should be noted, on balance, that another ACP represents another active task which will use pool space and that memory must be available to allow it to run without impacting the checkpoint rate.

The MOUNT command also lets the user re-specify some of the INI switches when a disk is mounted.

It is recommended that the file system ACP not share a user partition with tasks that perform file-structured I/O, especially in systems with minimal user memory.

From a another point of view, disk I/O can be optimized through the use of:

1. Overlapped seeks
2. Seek optimization
3. Multiple controllers
4. Dual ported controllers

Seek optimization has three algorithms:

1. nearest cylinder
2. elevator
3. c-scan

In the nearest cylinder algorithm, the disk picks the I/O packet that is closest to the current head position. In the elevator algorithm, the head moves from the center to the edge and back doing the I/O packets it traverses. The c-scan algorithm is the same as the elevator except that the movement of the head is in only one direction during the read/write operations.

Multiple controllers will definitely speed up disk I/O as transfers can occur at the same time. Disks that are accessed at the same time should be on different controllers. With RSX-11M, dual porting controllers is possible.

Finally, since ACP's are single threaded they can cause a bottleneck. It is a good idea to have an ACP for each disk.

A Comparison of Sorting Routines

B. Z. Lederman
Bankers Trust Co.

In the Spring 1981 DECUS symposia proceedings, there was an article comparing various sorting routines. This article estimated the time various routines would take by theoretical calculations. Not long after, we had an application which called for sorting a large array in core, and I was interested in finding out just how long each routine would take, so I wrote all of the routines in fortran, and ran them on a lightly loaded PDP-11/70 with RSX-11M (at that time, V3.1 or V3.2).

In order to test the timing, all routines were compiled with the Fortran-IV-Plus compiler with no trace. All of the programs use the same base:

```
        PROGRAM SORT
        IMPLICIT INTEGER ( A - Z )
        PARAMETER MAX = 2000
        REAL SECNDS, A, DELTA, RAN
        INTEGER*2 D(MAX)
C
        DO 50 I = 1, MAX
        A = 100. * RAN( 1, 1)
50      D(I) = NINT(A)
C
        A = SECNDS( 0.0)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        (one of the sorting modules goes here)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        DELTA = SECNDS(A)
        TYPE 60, DELTA
60      FORMAT( 1X / ' TIME  =  ', F14.7 / 1X)
C
        DO 80 J = 25, MAX, 25
        I = J - 24
        TYPE 10, (D(K), K = I, J)
10      FORMAT(1X, 25(1X, I2))
80      CONTINUE
C
        STOP
        END
```

The program is very simple. The loop ending at 50 puts random numbers between 1 and 99 into an array whose size is determined by the parameter MAX, and the time is obtained from the system. The chosen routine then sorts the array, and the elapsed time is obtained from the system and printed at the terminal. The loop ending at 80 prints out the data so I can see if the routine worked properly. As the random number generator always generates the same sequence of pseudo-random numbers, all of the routines work on the same data, so the timings are valid. The sorting routines used are:

BUBBLE SORT

```
        DO 500 K = 1, MAX - 1
        DO 300 L = 1, MAX - 1
        IF (D(L) .LE. D(L + 1)) GOTO 300
        X = D(L)
        D(L) = D(L + 1)
        D(L + 1) = X
300     CONTINUE
500     CONTINUE
```

INSERT SORT

```
        DO 500 K = 3, MAX + 1
        DO 300 I = K, 3, -1
        X = D(I)
        D(1) = X
        J = I - 1
        IF (X .GE. D(J)) GOTO 300
        D(I) = D(J)
        D(J) = X
        J = J - 1
300     CONTINUE
500     CONTINUE
```

SELECTION SORT

```
        DO 500 I = 1, MAX - 1
        D = D(I)
        DO 300 J = (I + 1), MAX
        IF (D(J) .GE. X) GOTO 300
        X = D(J)
        D(J) = D(I)
        D(I) = X
300     CONTINUE
500     CONTINUE
```

SHELL SORT

```
          M = 1
20        IF ((2**M) .GT. MAX) GOTO 30
          M = M + 1
          GOTO 20
30        M = M - 1
          M = 2**M
C
100       K = MAX - M
          DO 500 J = 1, K
          DO 300 I = J, 1, -(M)
          IF (D(I + M) .GE. D(I)) GOTO 500
          X = D(I)
          D(I) = D(I + M)
          D(I + M) = X
300     CONTINUE
500     CONTINUE
          M = M/2
          IF (M .GT. 0) GOTO 100
```

TREE SORT

```
          L = MAX
          DO 100 I = (MAX/2), 2, -1
100       CALL UPTREE
          I = 1
          DO 200 L = MAX, 2, -1
          CALL UPTREE
          X = D(1)
          D(1) = D(L)
200       D(L) = X
```

The following subroutine is also required for the tree sort.

```
          SUBROUTINE UPTREE
          IMPLICIT INTEGER (A-Z)
          PARAMETER MAX = 2000
          DIMENSION D(MAX)
          COMMON /TR/ D, I, L
C
          A = I
          X = D(A)
300       F = 2 * A
          IF (F .GT. L) GOTO 500
          IF (F .EQ. L) GOTO 400
          IF (D(F + 1) .GT. D(F)) F = F + 1
400       IF (D(F) .LE. X) GOTO 500
          D(A) = D(F)
          A = F
          GOTO 300
500       D(A) = X
          RETURN
          END
```

As the article mentioned explains the sorts I will not do so here except for a few particulars about the implementation. In the Shell sort, the selection of the jump parameter is important if the sort is to work. The method I used is to set M to be the largest power of two which is not greater than the number of items to be sorted (MAX). The routine in fortran looks complicated, but it is not. When we implemented this sort later in assembly language, it was very easy, as one simply finds the most significant bit set in the number of items, and this, taken as an integer number, is your starting value of M. For each pass, the jump M is reduced by a factor of two: in assembly language, a simple right shift one place on each pass until M is zero is all that is required (this is an interesting case where math is actually easier in assembler than in a high level language). In the tree sort, a common region was set up to pass the parameters from the main routine to the subroutine: if the parameters were passed as subroutine arguments (i.e., CALL UPTREE(D, I, L)), then the sort would take longer as the argument list would have to be set up for each call.

The final test of any sort is how long it takes, and how large it is. I determined the code size from the $CODE psect in the fortran compiler listings (the tree sort includes both the main routine and the subroutine). The speed of the sort was determined for MAX equal to 2000 and 6000 items. The results were:

SORT	MAX = 2000	MAX = 6000	$CODE
BUBBLE	39 Seconds	358 Seconds	132
INSERT	38	355	145
SELECT	17	155	143
SHELL	0.63	2.3	184
TREE	0.47	1.6	191

I believe the above figures speak for themselves. The only sorts worth considering are the Shell and tree sorts. As I find the Shell sort easier to understand, I have been using it exclusively in all of my programming since I first ran these tests.

I must point out that there is a very subtle and elusive bug in the INSERT sort routine, which causes exactly one data item to be incorrect. Because the insert sort showed no real advantage over the bubble sort, I did not persue this fault. Every indication shows that the Shell and tree sorts are so much faster that there is nothing to be gained by fixing the insert sort. And just in case anyone is wondering, Shell sort is capitalized as is named after a person.

History of the RSX SIG

Mark Lewis
Sally Shlaer
Raymond French
George Hamma
H. Legare Coleman

Former chairs of RSX SIG

Reported by Steven B. Dimick

The RSX SIG celebrated its tenth anniversary at the DECUS symposium held in Las Vegas. In conjunction with the event, the history of the RSX SIG as remembered by past chairmen and the present chairman was presented at one session.

Mark Lewis founded the RSX SIG at the San Francisco DECUS symposium in the fall of 1973. The SIG began with about 30 people about one-third of which were DEC employees. The event that spawned the SIG was the introduction of the RSX-11D.

Not a lot was accomplished that first year, said Lewis. Eric Pollik presented the first user paper in San Francisco. Hank Krache conducted an impromptu question and answer session that ended up lasting three hours. A wishlist also came out of the first meeting, and Lewis promised to begin a newsletter. The newsletter was started in December and the first issue appeared in January.

Lewis attributed part of the SIG's success in the first year to having a prompt newsletter. And not only a prompt newsletter but a controversial one. Lewis went out of his way to print any information he could get his hands on no matter what the source. The column was called "Notes from the RSX-11D Underground." A picture of the 1170 appeared in the publication six months before the machine entered the market.

Lewis said the SIG really got rolling at the spring 1974 DECUS meeting held in Boston. The SIG held its first night-time session, and many more people were interested in the RSX SIG. Topics at the Boston meeting included a plea for a system documentation manual, discussion of Fortran-4-Plus, and RSX-11M which had been formally announced at that DECUS. Sally Shlaer was also drafted at that meeting to write a column in the newsletter called "Help for New Users."

"All hell broke loose" at the fall 1974 convention in San Diego. A session was held and the demise of Mumps Fortran was announced. There were many people mighty angry over this, said Lewis.

RSX-11S was introduced at the Miami DECUS symposium held in spring 1975. The nucleus of a steering committee was formed with four people. The air conditioning system in the meeting rooms in Miami broke down, so the sessions were very sticky. Information on version two of 11M was available, and the first mail survey of RSX users was conducted. The issue of compatibility between 11M and 11D became hot at that meeting.

The fall 1975 symposium in Los Angeles was another good meeting for the SIG. The steering committee was larger, and many traditions were started in Los Angeles. Mini-tutorials were held, a formal question and answer session was held, and the first, though an informal one, campground was established.

Formal organization was the main topic at the spring 1976 meeting in Atlanta. June Baker joined the steering committee and volunteered to draft some by-laws, and a session was held to get some ideas on this subject.

Sally Shlaer and Jeff ??? presented a compatibility report on 11M and 11D at the DECUS convention held in Las Vegas in fall 1976. A formal reply to this from DEC was presented at the spring 1977 meeting in Boston.

The RSX program at the fall 1977 San Diego DECUS symposium included the first formal RSX campground. Lewis also submitted his resignation on the last day of the San Diego meeting and informed Sally Shlaer by telephone that he had elected her to succeed him.

Before that meeting adjourned, the position of newsletter editor was created and Mike Blake-Knox was selected to fill this position. Initial plans for the next symposium were also made.

The next meeting was in Chicago in the spring of 1978. By the Chicago symposium, the steering committee was very large and virtually all LUGs sent representatives to the steering committee. The by-laws were still being formulated, and Ralph Stamerjohn conducted a very successful ACP tutorial. That was the start of the pre-symposium workshops. There was also a software engineering panel at Chicago.

The fall 1978 symposium was again held in San Francisco. It was announced at this meeting that the up-until-then-free newsletter was costing money and would have to be paid for. The campground in San Francisco was the SIG's most successful event. San Francisco also marked the SIG's fifth anniversary, and a very brief history session was held. At five years, the SIG had established campgrounds and other DECUS SIGs were picking up on

these, help sessions and Q & A sessions had been established, and the SIG's organization was in place. The RSX SIG was, in Shlaer's words, "a very serious, very responsible organization."

Other commitments forced Shlaer to resign after only one year and Raymond French became the SIG's chairman. Then spring 1979 symposium was held in New Orleans. Two all-day pre-symposium seminars were held in New Orleans, the SIG's first. The SIG was also becoming more administratively organized, and the steering committee made some changes, mostly cosmetic, to the by-laws. The first steering committee election was conducted at the that symposium, but the election did not actually materialize as there were only as many nominees as there were positions on the committee. A very large wishlist was also collected.

DECUS went to San Diego for the fall 1979 symposium. RSX-11M+ was out by then and going strong. The first steering committee woods meeting was held to work on planning issues. One thing that came out of that was a proposed reorganization of the steering committee. This was presented to the membership at the spring 1980 symposium held in Chicago. Mike Knox resigned as newsletter editor shortly after Chicago, and Ralph Stamerjohn became the present newsletter editor.

DECUS returned to San Diego for the fall 1980 symposium, and the SIG held an software clinic. The clinic was experimental and did not work well because the clinic was not defined and everyone wanting help showed up at once.

The spring 1981 symposium was held in Miami Beach, and most attendees seem to agree that it is memorable most because of the horrible lunches. The subject of SPRs was hot at the Miami Beach convention, because DEC was not publishing SPRs as they had in the past. This was causing problems in tracking down bugs. A resolution came out of that meeting requesting DEC to publish all raw SPRs. DEC did not respond to the request. Another software clinic was held, but the San Diego experience proved useful as this one was well organized and very successful.

A genuine steering committee election was held between the spring and fall 1981 symposia, and at a special mid-year meeting George Hamma was selected by the committee to be the chair.

Los Angeles was the site of the fall 1981 DECUS meeting. The SIG experimented with session video taping at the symposium for the benefit of the many SIG members not able to attend. Video taping of special sessions is now common. A woods meeting was held in Dallas between the fall 1981 and spring 1982 symposia.

The spring 1982 DECUS convention was held in Atlanta, but no really significant RSX events took place, said Hamma. Version four came out at about that time, but not quite enough people were upset to make any changes. H. Legare Coleman succeeded Hamma as

chairman of the SIG in July 1982 after Hamma was elected to the executive board.

By this time the SIG was running pretty well, said Coleman. The Anaheim meeting in the fall a firsts. The magic session was split into a fun session and a technical session. Ralph Stamerjohn also resigned his position as newsletter editor. A woods meeting was held in Marlboro, Massachusetts later in the spring.

The spring 1983 meeting was held in St. Louis and a subscription service for the newsletter was announced. New operating procedures were discussed as the operating procedures then in use had been formulated early in the SIG's history. Those operating procedures have only recently been approved. Two other SIGs were also spun off the RSX SIG inn St. Louis: IAS and PC.

This brings RSX SIG history up to the fall 1983 symposium held in Las Vegas. Coleman believes it to have been one of the SIG's more successful meetings, though the campgrounds did not work as well as they had in the past. There was also a problem with space for meetings, but Coleman considered this to be a welcome problem because it shows the size and responsiveness of the SIG.

VMS to RSX Migration

Joseph Sventek
Lawrence Berkley Laboratory

Reported by Steven B. Dimick

"We've all seen RSX to VMS migration and RSTS to VMS migration, but I thought it would be interesting to look at the problems in going the other way: from VMS to RSX," said Joe Sventek at the DECUS symposium in Las Vegas. "No, I'm serious."

First, to consider, of course, is motivation. Why VMS to RSX? The J-11 chip makes the PDP-11 a nice cheap, fast processor. There is a distinct lack of cheap, fast VAX processors or even cheap processors. "I'm not convinced that the MicroVAX is so fast. You shouldn't ask a person who's been running on a really fast 11/34 to take a cut in performance to run on a VAX." But the best is poor coding practices encouraged by virtual memory.

Of course there are going to be some major sources of incompatibility. One is process virtual address space. If your programmers are trained correctly they won't need that address space. But sometimes they think it's easier; ODL files are fun to create. Logical names are actually really nice. Hierarchical file structure. The biggest source of incompatibility between VMS and RSX is that RSX has a very small documentation set.

Some kind of compatibility mode has to be provided. We have to come up with a VMS AME and emulate VMS system calls. There would be features of such an AME. It would provide true emulation for a set of system calls:

1. $EXIT

2. $WAITFR

3. $SETEF

4. $CLREF

With those four primitives you can do anything. Obviously VMS is over specified if those four are sufficient. So, for all other services you get the message "insufficient privilege" or "quota exceeded."

Certain compatibility images have to be provided. A program called SYSGEN has to be provided. We will do true emulation of SYSGEN, and you will be able to change, in exactly the same way as you do on a VMS system, all the parameters specified in SYSGEN. And as in VMS, changing those parameters will have absolutely nothing to do with how the system runs. The other compatibility image is DCL; we have to have a standard command language which is non-standard everywhere.

Of course we must keep in mind the user's perception of the system. The first perception of the user will be that the most heavily used utilities, the software tools, are unchanged. Next they will notice that DCL is just different enough to be a pain. Terminal response will typically be a little faster than for VMS, so two courses of action will be open to the system manager: he can tell his users that they have installed a speed-of-light accelerator on the 11 or there is now is a new conditional when you build a terminal driver: T$$BVE which stands for" brain damaged VMS simulation" in which you set it to the number of no-op instructions to execute after each successfully transmitted or received character. They will also notice the documentation kit is too small. The suggested solution to this is to purchase extra orange binders and fill them with whatever magazines they would like to read while they're waiting for the taskbuilder to finish.

There is an optional enhancement for this package. We want to make this look like VMS as much as we possibly can. We've already slowed down the terminal driver. Next, you've probably noticed the error messages on VMS are rich and voluminous. To help provide that support we will have an error ACP, but we will do better than VMS; we expand to a whole screenful, and it will be designed so that it pushes your valid results right off the top of the screen, and to be compatible it must put out absolute garbage that has absolutely nothing to do with what went wrong.

An important aid to help users migrate their programs will be a "foreign terminal package." This will provide users with an important placebo, so you can think you are telling the system that you have a different terminal. Of course, EDT will not use this package.

Abort AST's in Fortran

(and other high level languages)

B. Z. Lederman

Bankers Trust Co.

Some time previously an article was published showing how to do such things as AST's from Fortran programs, and in particular the Abort AST CALL SREA identified as calling and EXTERNAL function address. This is really nice when you want a program to clean itself up before exiting when it is aborted by command (or ran over the time limit in a Batch job, or some similar contition). I followed the instructions in the example, and it worked very well on RSX-11M-Plus, but I began to run into problems when I tried to move the same program to a PRO-350 with P/OS. The system documentation (both M-Plus and P/OS) clearly state that there are severe limitations on doing AST servicing in Fortran for various reasons, including the fact that the OTS impure regions can get very messed up. You are not supposed to do any Fortran I/O or file services in AST state, which limits the clean-up you can do. My program was doing some I/O, and got away with it on M-Plus, but it worked only some of the time on P/OS. Because I wanted my programs to work every time, I developed the following simple subroutine which does all of the AST servicing in a macro module: because it exits AST state when done, your high level language program can do what it wants when cleaning up.

The module is very simple:

```
   .TITLE ABORT
   .IDENT /V2.0/
;
;  Abort AST function for Fortran programs
;
;  B. Z. Lederman          Bankers Trust Co.
;
;  Use of this module is entirely at the users' risk. No liability
;          of any kind is assumed.
;
   .MCALL  ASTX$S, SREX$S
;
;  The following PSECT matches the attributes set by Fortran for
;          common areas.
;
   .PSECT  ABOCOM, RW, D, GBL, REL, OVR
ABOFLG::
   .WORD   0                          ; LOGICAL*2 data type
;
   .PSECT  $ABRTC, RO, I, GBL, CON, LCL
;
ABOINI::
   CLR ABOFLG                         ; set abort to .FALSE.
   SREX$S   ABOAST                    ; specify AST point
   RETURN                             ; return to main program
;
ABOAST::
   ADD (SP), SP                       ; clean up stack
   DEC ABOFLG                         ; set abort to .TRUE.
   ASTX$S                             ; leave AST state
;
   .END
```

The PSECTs have been chosen to be compatible with those produced by the Fortran-77 (and Fortran-IV-Plus) compilers, and can be seperated into I and D space. It is an ABSOLUTE requirement that this module be in the ROOT of your program (both the code and the data common PSECTS), otherwise you may not receive AST notification properly. An example of how the module is used follows.

```
   PROGRAM ABOTST
C
C  Test Abort-AST action
C
C  B. Z. Lederman
C
   LOGICAL*2 ABOFLG
C
   COMMON /ABOCOM/ ABOFLG
C
C  Initialize abort
```

```
C
   CALL ABOINI
C
   OPEN (UNIT = 1, FILE = 'ABOTST.DAT', STATUS = 'NEW')
C
C  Begin dummy loop
C
   DO 100 I = 1, 10000
   DO 100 J = 1, 10000
   DO 100 K = 1, 10000
   IF (ABOFLG) GOTO 900
   WRITE (1, 920) I, J, K
100        CONTINUE
   STOP ' Reached end of all loops'
C
900        WRITE (5, 920) I, J, K
920        FORMAT( 3( 1X, I5) )
   CLOSE (UNIT = 1)
   CALL EXIT
C
   END
```

The declaration of ABOFLG as LOGICAL*2 and ABOCOM are used to match up the variable and the area where it is stored. The call to ABOINI should occur at the beginning of your program to set ABOFLG and the abort AST entry points to the correct values. The task build would be something like this:

```
TKB ABOTST = ABOTST, ABORT
```

Or if overlayed, the ODL file should have something like:

```
        .ROOT   ABOTST-ABORT-*( tree1, tree2, etc.)
```

Both of which are intended to force the ABORT module into the root of the program.

The benefit of abort AST's may be seen by running the above program: when aborted, the program will close the file ABOTST.DAT, and print out the current values of I, J, and K on the terminal. If you run the same program without the AST routines and abort it, the file ABOTST.DAT will be locked, and you will not learn the state the program was in when aborted. I am sure that nearly every reader can think of an application where it is desireable for a program to do clean-up work when aborted, or at least to leave files properly closed rather than locked.

To derive the greatest benefit from this abort AST routine, you must test ABOFLG fairly often: the macro module only sets the value of ABOFLG, and it is up to you to test it's status and shutup when you are supposed to. I generally put it at the beginning of any long loops, or before reading new data to process, or afte writing out a data item, or in similar "breaks" in processing. In

the above example, the test IF (ABOFLG) GOTO 900 will occur on every pass through the loop, so notification that an abort AST has been received will be quick.

I would hope the following information is superfluous, but in case it isn't: do not use the variable ABOFLG for anything else in your program, and don't put any other variables into ABOCOM.

This method should work just as well for other languages. The requirements are that you be able to declare some kind of common area for ABOFLG, and that you be able to declare a one word integer and test it's state. It will be zero (all bits off) if false (no abort yet), and a minus one (all bits on) if true (time to abort). Fortran-77 tests only the sign bit, but any test that can distinguish these two states is equally valid. The calling convention to ABOINI is the usual DEC convention (JSR PC, ABOFLG), with the argument list on R5: in this case, there are no arguments, so no registers are tested or altered. The RETURN statement translates to an RTS PC.

An alternative I considered was to have the macro module do a subroutine call to an ENTRY in the main program from the AST service routine. The problem with this would be that the Fortran program would have to do servicing at AST time, which is exactly what needs to be avoided. However, the restrictions on AST service might not apply to some other high level languages, and if declaring a common region is difficult in that other language, then it might be easier to have the macro module do a call to the other language module which will do the abort clean up and exit. If you do try this, remember that when the AST service state is entered, the stack has additional information placed on it, and that other registers may not be preserved.