RSX-11/IAS SIG DECUS US CHAPTER

THE MULTI-TASKER

Volume 15, Number 2 August 1981

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

FROM THE EDITOR

To start with, I must make a correction. In the May 1981 issue, I inexplicably left off the name of two co-authors to the article "The Best of ICR". Bill Cael and William Wood are also authors along with Robert Stodola and deserve the thanks of the SIG for their very excellent submissions. I have put up some of their programs and the "Best of ICR" is a gold mine.

This issue has two articles on resident libraries and the overheads from the SPM-11M presentation given at the Spring Symposium in Miami. For RSX-11M Plus sites, you should not miss Stanley Rose's description on how to link a F4P resident library to a supervisor-mode FCS library. SPM-11M is the new performance monitor, available from Software Services, for RSX-11M. We are currently using it at Monsanto and I have included some comments on our initial reaction. Also, there are the regular columns. Of special note is the repeat of Jim Downward's description of the undocumented features to IND. Because of the demand, this is reprinted from two years ago. If you know of some article that should be reprinted, drop me a line.

One of my submissions to the Miami RSX/IAS Spring Decus tape was a file called TAPESB.DOC. This is my attempt to abstract and index the thousands of submissions to the RSX SIG tapes. The Multi-Tasker is budgetted for 78 pages per issue, something we never seem to reach. Starting with this issue and continuing though the next several months, I will be using TAPESB.DOC as filler. If you cannot wait to the exciting climax, you can find the whole file in UIC [346,100]. Note, the copy printed here is updated to include the Miami tape and has a better index.

Once TAPESB.RNO is exhausted, I would like to print something else in serial form. If you have something suitable or can recommend some documentation from the SIG tapes, please get in touch with me. One recommendation already made was the ACP manual from the 1980 Spring SIG tape.

Finally a note about submissions. If you prepare your articles using computers, I would appreciate machine-readable form. This would greatly help me and my wife prepare the Multi-Tasker. I can handle magtapes (either 800 or 1600 BPI), RX01 floppies, RK05 disks, or TU58 cartridges. All media will be returned. And don't despair if you cannot do this. I still want your article, even if it is scrawled in crayon on tissue paper.

The September issue is up in the air. I am teaching the ACP Seminar at the European Decus and taking vacation at the same time. Esther and I leave August 21 and return September 15. If possible, we will try to publish the next issue before we leave. Otherwise, there will be a combined September and October issue.

Ralph Stamerjohn
Multi-Tasker Editor

HELP YOURSELF

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

We would also like to publish the answers to questions. If you can help someone, send a letter to the Multi-Tasker or call Ralph Stamerjohn at (314) 694-4252. Your answer will be sent directly to the person in need and published in the next edition of the Multi-Tasker.

ANSWERS TO PREVIOUS QUESTIONS

DX11-B UNIBUS TO IBM CHANNEL INTERFACE

V.R. Robson replied to John Seddon's request for help for a RSX-11M driver for the DX11-B. Mr. Robson company, Scicon Consultancy International, markets software packages which drive the DX11-B under RSX-11M and emulate IBM's magnetic tape controllers, 2701, or an IBM SNA sub-area. Scicon has a great deal of expertise with the DX11-B. For further information, conact V.R. Robson at Scicon Consultancy International, Sanderson House, 49 Berners Street, London, Great Britian, W1P 4AQ. Phone 01-580 5599. Scicon operates throughout the world, including North America.

THIS MONTH'S QUESTIONS

/PRM PATCH TO INSTALL COMMAND

I am very interested in obtaining the PATCH to the INSTALL command to allow use of the /PRM switch I know it exists on one (or several) SIG tapes, but have been frustrated in my search. Would someone please publish this patch in the "Multi-Tasker" so that I and users like myself could implement it?

Hal C. Kelley, GIANT FOOD INC., Box 1804, Washington, DC. 20013. (301) 799-6762.

FROM THE EDITOR

My search of the SIG tapes shows the patches can be found on the Fall 1979 tape in UIC [344,40]. The necessary files are INSHD.SLP, INSPS.SLP, and INSROT.SLP. Also, on the Spring 1980 tape in the same UIC is another version of INSPS.SLP and a patch to RPSOV.SLP which is needed for /PRM support. I have not used these patches, so if anyone has more up-to-date information, please pass it along.

ODT AND EVENT FLAGS

While debugging Macro-11 tasks with ODT, we at times would like to be able to look at the special event flags. Perhaps someone out there could tell us how to do this.

Greg Marshall, United Grain Growers Limited, 433 Main Street, P.O. Box 6600, Winnipeg, Canada, R3C 3A7. (204) 944-5411.

FROM THE EDITOR

A not-so-clean way to do this is as follows:

* The ATL command will display the local event flags (1-32).

* The OPE command can be used for the global event flags (33-64). Open and examine locations $COMEF+0 and $COMEF+2 in the executive. The value of $COMEF can be found from the executive map.

* The FLA command will display the group global event flags (65-96).

HINTS AND THINGS

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

SRD PROBLEM

The following article is from Hal C. Kelley, GIANT FOOD INC., Washington, D.C. 20013.

For those who have tried to use the SRD in account [300,57] on the Fall 1980 SIG tape and did not generate support for read after prompt in their terminal driver, the /SD (selective delete) switch, and possibly others, will not work. However, replacing the two QIO's that use IO.RPR's with two pair of QIO's using IO.WVB, IO.RVB solves the problem. The IO.RPR's appear in module SRDLST.MAC.

UNDOCUMENTED IND FEATURES

The following article was submitted by Jim Downward, KMS Fusion, 3941 Research Park Drive, Ann Arbor, Michigan, 48104. This article was originally published in the October, 1979 issue of the Multi-Tasker (volume 11, number 4) and is repeated here by popular demand. Also included is a follow-on patch for the .TESTFILE command.

IND now supports:

1. PARAMETER PASSING. If a command of the form

```
@cmdfile  Parm1 Parm2 ....Parm9
```

   is entered, the parameter strings will appear in variables P1 through P9. P0 will contain '@cmdfile' and COMMAN will contain the entire command as it was typed in.

2. COMMAND LINE PARSING via the .PARSE directive. The .PARSE directive is perhaps the most complicated and powerful ...AT directive. It is used to break (parse) a command string (input from a .ASKS or any string variable) into a series of substrings. The command syntax is shown below:

```
.PARSE ISTRING CONTROLSTRING  OS1 OS2 ..... OSn
```

   ISTRING is the input string which is to be parsed. CONTROLSTRING consists of from 1 to (n-1) characters which are to be uses as delimiters for parsing ISTRING into the output strings OS1 OS2 .... etc. If there are fewer characters in CONTROLSTRING than (n-1) the last control character is used for all remaining delimiters. Typically, a control character is a punctuation character but it may be anything(even a space). The control string may be a string variable, a string expression, or a doubled quoted string. String expressions may include forms such as 'STRING[N1:N2]'. No string may be longer than 80 characters. The contents of ISTRING up to the first delimiter will be placed in OS1, the contents of ISTRING from past the first delimiter up to the second delimiter will be placed in OS2, and so on until the contents of ISTRING from past the last delimiter to the end of ISTRING are placed in OSn.

   Consider the following example of using .PARSE.

```
.SETS STRING "DK0:[1,2]STARTUP.CMD;4 ;PARSE THIS STRING"
.PARSE STRING ":].;" DEV UIC FILNAM EXT VERSN COMENT
; DEV='DEV':
; UIC='UIC']
; FILENAME='FILENAM'
; EXTENSION='EXT'
; VERSION='VERNS'
; COMMENT='COMENT'
```

3. .TESTFILE. The .TESTFILE directive is used to find out if a file exists or to discover what physical device is currently assigned to a logical device. The syntax is as below:

```
.TESTFILE filespec
```

   When a .TESTFILE is executed, an FCS .PARSE is attempted on the file as specified. The FCS error code is returned to the special numeric symbol <FILERR>. If <FILERR> = 1, the file exists. Other error codes can be found by looking in the back of the I/O Operations manual. It should be noted that <FILERR> is set by .TESTFILE, .OPEN, .OPENA, and .OPENR. When a .TESTFILE or a .OPEN is executed, the expanded file specifier is placed in the special string symbol <FILSPC>. By examining this string symbol one can discover the physical device assigned to a logical device. For example, one can execute a <.testfile TI:. The string TTn: will then be placed in <FILSPC> where it can be examined and used.

4. .OPENR. The .OPENR directive is used to open a command file for reading via the .READ directive. The syntax is identical to the other .OPEN directives, i.e.

```
.OPENR [#N] filspec
```

5. .READ. The .READ directive is used to read a data file, one ASCII line per .READ, into a target string variable. The syntax is below:

```
.READ [#N] stringvariable
```

   The special logical symbol <EOF> is set true if a .READ directive hits the end of file.

The only known restriction, at this time, is that parameter passing does not work correctly through a .CHAIN directive. With this minor exception all the features work well and make life much easier. While these features are 'unsupported", I am quite sure the RSX group would like to get feedback on them and information on any bugs related to them.

In using the .TESTFILE command I located a bug which luckily has nothing to do with the .TESTFILE command but surfaces if two .TESTFILE commands (or two .OPEN's) are executed without any intervening commands. The enclosed SLP correction file (supplied by DEC) will correct this problem. DEC will probably not issue this correction in the SOFTWARE DISPATCH as the bug was discovered while using an unsupported feature. It may however be on the Auto Patch tape.

The problem is caused by ...AT. storing the terminal's UIC in the word following the internal FDB in the .OPEN overlay. Unfortunately, the word for storing the UIC was omitted and the UIC was overwriting code in an adjacent module. If two .TESTFILE's or .OPEN's were executed, the second time around...AT. would try to execute the UIC as an instruction with sometimes disastrous results (depends on the UIC). If, however other commands existed between the two commands using the .OPEN overlay, additional overlays would come in and overwrite the bad data.

```
INDOPN.MAC;2/AU=[12,10]INDOPN.MAC;1
\
-2,2
        .IDENT  /4.20/
-47
;
;       JGD     17-AUG-79       FIX SNEEKY PROBLEM WITH TWO .TESTFILES
;                               BACK-TO-BACK, OVERWRITING THE FDB AND FIRST
;                               INSTRUCTION OF THE NEXT MODULE. FIX RECEIVED
;                               VIA PHONE FROM DIGITAL. WITHOUT THIS FIX, THE
;                               SEVERITY OF THE PROBLEM VARIES FROM IOT ABENDS
;                               TO SYSTEM CRASHES, DEPENDING ON ONE'S CURRENT
;                               UIC. THIS IS A MANDATORY PATCH IF .TESTFILES
;                               ARE TO BE USED.
;
%
-306,306/; JGD/
        .BLKW   1               ;LEAVE SPACE FOR OUR UIC(OR ELSE!!!!!)
```

FORCING RSX-11M OR IAS CRASH DUMPS

The following article is from Greg Merrell, a Digital Software Specialist from Cleveland, Ohio. He describes how to force crash dumps for both RSX-11M and IAS.

Forcing a crash dump on RSX-11M is very straightforward (see RSX-11M/M-PLUS Crash Dump Analyzer Reference Manual, page 1-3, end paragraph):

1. Halt the processor (if it isn't already).
2. Load address 000040 and start. (Location 40 contains a JMP $CRASH instruction, and is the route taken by the normal system crash routine).

For an IAS system, the idea is the same (except the address is 000044 instead of 000040), but the virtual memory configuration is different from RSX-11M. When RSX maps the executive into physical memory, virtual address 0 is also physical address 0. This is not the case on IAS.

The result of this is that if memory management is on (the normal case) then the crash routine may be executed properly. If for some reason, memory management has been turned off (such as an INIT or a START from the front panel would do) you must re-enable it.

On IAS V3.0, virtual 0 is USUALLY physical 400; this value is USUALLY 500 on V3.1. The offset from physical 0 is actually a function of the top vector specified to SGN1, so if you have lots of devices in the floating space, these numbers may very well be higher.)

You must also verify that the contents of Kernel APR 0/1 registers are proper. Note that all of the following operations are performed from the front panel of the processor, and all numbers are in octal. If your processor does not have a switch register (such as early 11/34's) you cannot use this technique.

IMPORTANT

You should verify the value of KIPAR0 while your system is running before you need to invoke this procedure as it may vary from system to system. Use an OPEn command from a priv account as follows:

```
MCR>OPE 172340/KNL
00172340 000005/ <esc>  -or-

PDS>EXA/KER 172340
00172340 000005/ <esc>
```

If you neglected to do this before you needed the value, look in location 544 for a 000167 . If you find it, KIPAR0 most likely is a 5. Try other values (444,644) until you find a match for the 000167. When you do, take the value of the address without the last two digits, and use that for the KIPAR0 value.

1. Halt the processor (if it isn't already).
2. Examine the PSW to be sure that you are in Kernel mode:
   * Examine location 777776 (Processor Status Word).
   * If the high two bits PSW<15:14> are not both 0, zero them.
3. Verify that memory management is enabled:
   * Examine location 777572 (MMR0).
   * If the low order bit (MMR0<0>) is not set, set it.
4. Check that the Kernel APR's are still properly mapped:
   * Examine location 772300 and 772302 (KIPDR0/1).
   * If they are not already set to 077406 or 077506, set them to 077406.
   * Examine location 772340 (KIPAR0).
   * If it is not 000004 (IASV3.0) or 0000005 (IASV3.1) set it appropriately.
   * Set location 772342 (KIPAR1) to be 200 higher than KIPAR0, i.e. if KIPAR0 is 000005, then set KIPAR1 to 000205 .

5. Load the program counter with 44:

   * Load address 777707.

   * Deposit value 000044.

6. Hit CONTINUE (NOT START!!!!!!!).

7. The system should put out the message:

```
CRASH - UNDEFINED SYSTEM ERROR - CONTINUE WITH etc...
```

8. At this point continue as you would with any normal crash dump.

Be aware that when you use this method, the top of the system stack will not contain the telltale 000006 as with a normal dump. The system stack pointer will, however, still point to the top of the stack when the system was halted.

CAVEATS

Since the crash dump routines run strictly out of memory, there is no reason to have any disks write enabled, much less spun up. It is highly recommended that you spin down any disks (except for the crash pack/tape) or at least write protect them before continuing any crash dump. Remember- the system has died in a horrible way, don't let the same happen to your disks!

It would also be a nice idea to indicate the dump procedure used when submitting an SPR to Digital to save any possible confusion over the state of the system stack.

Yes, there are other ways to get into the IAS crash module without using the crash vector in 44, but they are somewhat more involved. The method described above works, and is relatively simple. I have always found that when I use a cookbook approach, I have less chance of making a mistake. Feel free to take whatever shortcuts you wish; it helps to read the microfiche source listings for another method which ends up being as safe.

A LAUDATION FOR RESIDENT LIBRARIES

W. Sander

FINNIGAN MAT
Bremen, West Germany

What is a Resident Library?

This is well known. It is a piece of code, which is usually permanently resident in memory and which may be used by several tasks. This code therefore has to be reentrant. Such code is supplied to the RSX user in the normal SYSLIB and consists of the modules which link together in the PSECT $$RESL. These modules contain FCS-related software, so we call the resident library FCSRES.

Well, resident libraries are easy to use but difficult to tailor, but be convinced, the work is worthwhile. Tasks are shorter in memory, use less disk space, and if they are overlayed, they run even faster. Doesn't that sound exciting?

How do we use the FCSRES? Quite easy! We add only one line to the ENTER OPTIONS during taskbuilding: TKB>LIBR=FCSRES:RO. Optionally we can add the APR number to use, usually 7. Unfortunately the library option is only available for the normal taskbuilder, not for FTB! That is why FTB is rather useless, because TKB time also shrinks significantly when using resident libraries. For my measurements a typical Fortran task links with the BIGTKB when using F4PRES and FCSRES as fast as with the FTB! I claim again, FTB is useless in most cases.

Now to the harder part. How to tailor a Resident Library. Doing this you should have in mind that each LIBR option uses 1 APR for a library which is 0-4 KW, 2 APRs for a library which is between 4 and 8 KW long, etc. As a RSX11M task has only 8 APRs we must consider carefully what to put into the library. We have to put as much in it as fits into 4KW boundary.

Unfortunately there is little help from DEC. A long time ago, until release 3.0, it was comparatively easy to tailor it. We only had to put all those modules into it that link in PSECT $$RESL. But that worked only if we did not select ANSI support. At that time we were happy, because there was a tiny note in the Multitasker, telling how all could fit together for ANSI support by amalgamating three OPEN routines into one. Due to software changes from that time which resulted in an increase of code, it is no longer possible to put all together in 4KW.

But don't give up, there is still hope for the patient. If you selected ANSI support, you can link a 5.5KW memory resident overlayed library which uses only 1 APR. The command for that is supplied in the normal distribution kit. The code is roughly divided into 2 parts, one with the OPEN/CLOSE Routines, the other one with the I/O routines.

If you have the normal SYSLIB, you can either

- o use the above mentioned patch described actually for ANSI support. This also works fine for the normal SYSLIB, and shrinks the code significantly under 4KW. Unfortunately there is a "Magic Number" in the code, or,

- o put as much in FCSRES as fits into 4KW. Simply comment out the rest. At the moment (Patchlevel D) this is .XCTRL and .RENAM. Then link the Library and then !!! simply change the name of PSECT $$RESL to, lets say, $$RESK !!! (Words 24 and 26 in FCSRES.STB.)

PSECTs have several purposes. Usually modules are named to a PSECT in order to link them together for some reason. But obviously not all modules that link per default into PSECT $$RESL have to be there. That is why we can change the name of the PSECT. I must admit, that I am not really happy about that. Who dares to alter things from the system software? We would be better off if we could safely link all corresponding code, with no tricks, into one APR. So, when will DEC satisfy this need? When will DEC supply the RSX community with a standard FCSRES and with the build files for all RSX tasks, privileged and nonprivileged?

For, be aware, now comes the hardest part! After we created the library successfully, we would probably want to link not only our tasks to that, but also all the RSX tasks that use FCS. (This saves about 500 blocks on disk.) But which ones are they? There is no note anywhere in the RSX documentation. Well, we can find out nevertheless. If we look into the build files, we find lots of references to those modules we have jut put into the library. So, the simplest solution should be to remove all those references. What a lot of work! Shouldn't we write some TECO Macros for that? This work has already been done! Look onto the Fall 79 DECUS tape, many thanks for that.

Apparently the ODL files for the system tasks are optimized for the case that they do not link to resident libraries. They can really be improved for FCSRES. In order to be able to do that, we need the following information:

Some tasks extend their workspace at runtime or get it larger when INCremented at INStall. Which tasks are they? In which branches are the corresponding modules? Do they get their workspace only from the end of the task, or also within one branch of the overlay?

Is there any expert in the RSX community who can answer that? Or is it true what is rumoured, that DEC will support FCSRES in the next release?

BUILDING A F4P RESIDENT LIBRARY LINKED TO FCSFSL

Stanley M. Rose

Bankers Trust Company
New York, New York

This note is a description of the technique used at Bankers Trust to build a Fortran-IV-PLUS resident library linked to the FCS Supervisor Mode library (FCSFSL). The resultant library is not itself in supervisor mode but has all its FCS references resolved from FCSFSL. It has almost all the F4P runtime system, including the long error text ($ERTXT) included, and is 8KW (2 APR). At Bankers Trust we found that programs are typically 10KB smaller built with F4PFSL versus F4PRES. Tasks built with F4PFSL use one additional window block which takes 25 words of primary pool for each task.

This method is based on Richard Kirkman's note in the July 1980 (Vol 13, #1) MULTI-TASKER and acknowledgment is given to him for developing the method.

It should also be noted that by the time this is published sufficient patches may have been issued by DEC to $SUPL and the Task-Builder to allow the building of the library without any tricks.

TECHNIQUE

The Task Builder does not want to allow a non-supervisor mode library to be cross linked to a supervisor mode library; this method does the cross linking manually. What is done is that an assembly language module (FCSFSLVEC.MAC) is created that contains one entry point for each FCSCSL entry point and a CMS (Change to Supervisor Mode) instruction to call the appropriate module in FCSFSL. This object module is linked in with a tailored version of F4PRES.MAC, resolving all FCS references. The label block of F4PFSL is ZAPped in its task header to look as if it is linked to FCSFSL so that at run time the system will actually map FCSFSL with the appropriate Supervisor mode APRs of the user task. When a Fortran-IV-Plus task is task built to F4PFSL, any run-time module in F4PFSL that needs an FCS service issues its normal subroutine call but calls the transfer vector in FSCFSLVEC which issues the CSM instruction and transfers to the appropriate module in FCSFSL.

One additional "kludge" that is necessary is to fake out the task builder when building a task against F4PFSL. When building the user task, the task builder opens the FCSFSL.STB since it finds that this library is linked (indirectly) to the task. Because the modules in FCSFSLVEC.MAC have been built with the same global names as the modules in FCSFSL, many MULTIPLY DEFINED symbol error messages are printed. Luckily it appears that TKB uses the same UFD for FCSFSL.STB as it uses for F4PFSL.STB so the last step is to put F4PFSL in an account with a phony copy of FCSFSL.STB that has been built with no symbols at all.

STEP 1

Build any task with FCSFSL, use ZAP to examine the 14 word library descriptor block at addresses 40(8) to 72(8) of the Task File. Save the values for use at the end to make the library we will build look like it was built against FCSFSL. For example, follow the procedure below:

```
                .TITLE TEST
        START:  NOP
                .END START

        >MAC TEST=TEST

        >TKB
        TKB>TEST=TEST
        TKB>/
        ENTER OPTIONS:
        TKB>SUPLIB=FCSFSL:R0
        TKB>//

        >ZAP
        ZAP>TEST/AB
        _0:40/
        000:000014/XXXXXX
           .
           .
        _0:72/
        000:000072/WWWWWW
        _X
```

STEP 2

Create an entry point table (FCSFSLVEC.MAC) that will be task-built with an F4PRES that resolves any calls to FCS modules that are in FCSFSL. The start of FCSFSLVEC.MAC contains a SV macro definition and the SUPL subroutine which results in a transfer into FCSFSL at the correct entry point. It should look like the version used to build our library (attached). This table will be hardcoded with the absolute addresses (in SUPERVISOR MODE address space) of the equivalent entry point names in FCSFSL. The most reliable way to produce this table is to create a CROSS REFERENCED MAP of FCSFSL from [1,1]FCSFSL.STB, and then to edit the CROSS REFERENCE portion (which is in a one-line-per-symbol format) to match the MACRO of the table. In other words, if the listing has the lines:

```
        $CMPCS  000014  # CMPCS
        $DSPAT  000652  # CMPCS
```

then after editing these two lines will look like:

```
                SV      $CMPCS  000014
                SV      $DSPAT  000652
```

STEP 3

The library (F4PFSL) should now be built using the following command lines to

TKB:

```
        F4PFSL/-HD,F4PFSL/-SP/CR/MA,F4PFSL=
        F4PFSL,FCSFSLVEC,[1,1]SYSLIB/LB:$ERTXT,SYSLIB/LB
        /
        STACK=0
        PAR=GEN:100000:40000
        //
```

STEP 4

Use ZAP to add the 14 words from step 1 to the library block of F4PFSL.TSK.

STEP 5

It is necessary to put the library in an account with a dummy copy of FCSFSL.STB that has been built with no symbols. This is done as follows:

```
        .TITLE  DUMMY
        .PSECT  BTDUMM
        .END

        >MAC DUMMY=DUMMY
        >TKB
        TKB>NL:/-HD,,DUMMY=DUMMY
        TKB>/
        ENTER OPTIONS:
        TKB>STACK=0
        TKB>PAR=GEN:0:40000
        TKB>//
```

USING THE LIBRARY

We use [2,2] to hold the necessary files, F4PFSL.TSK, F4PFSL.STB. Also, DUMMY.STB should be moved to [2,2] and named FCSFSL.STB. Tasks can be linked to the library by specifying the following option when task building a Fortran task.

```
        OPTION>LIBR=[2,2]F4PFSL/RO
```

F4PFSL.BLD

The following is a listing of F4PFSL.BLD. This is the command file we use to automate this procedure.

```
        .ENABLE SUBSTITUTION
;
; Setup devices and UIC.
;
        ASN SY:=TK:
        ASN SY:=MP:
        ASN SY:=IN:
        ASN SY:=LB:
        .SETS OLDUIC "'<UIC>'"
;
```

```
; Assemble F4PFSL.MAC
;
        MAC F4PFSL,F4PFSL=F4PFSL
;
; Assemble FCSFSLVEC - the module that resolves the
; the FSCFSL entry points.
;
        MAC FCSFSLVEC,FCSFSLVEC=FCSFSLVEC
;
; Task build F4PFSL.TSK and F4PFSL.STB
;
        TKB @F4PFSLBLD
;
; ZAP F4PFSL Task Header
;
        .IFNINS ZAP INS $ZAPFSL
        ZAP @F4PFSLZAP
;
; Create the dummy FCSFSL.STB
;
        MAC DUMMY=DUMMY
        TKB @DUMMYBLD
;
; Move the files into [2,2].
;
        SET /UIC=[2,2]
        UFD SY:[2,2]
        PIP /NV/CD='OLDUIC'F4PFSL.TSK,F4PFSL.STB
        PIP FCSFSL.STB/NV/CD='OLDUIC'DUMMY.STB
;
; Install F4PFSL - this also must be done in system startup.
;
        INS [2,2]F4PFSL/PAR=GEN
;
; All done, clean up.
;
        SET /UIC='OLDUIC'
        ASN =
```

F4PFSL.MAC

This is the listing of F4PFSL.MAC we use. It is the standard Digital F4PRES that is tailored to our needs. It is shown here as it builds to closes to 8 KW's as possible. Use the comments shown for tailoring F4PRES but follow the rules above for building a F4PFSL.

```
        .TITLE  F4PFSL  F4P V3 SUP. MODE LINK LIBRARY
        .IDENT  /F4P013/
;
;       EDITED  14-JAN-81       SMR     INITIAL
;
; Entry points are grouped according to use, but not all of a group
; need to be included. Individual modeules may be excluded by commenting
; out the .GLOBL xxxxx statement.
;
```

```
;       The groups defined are:
;
        SFIO =1         ;SEQUENTIAL FORMATTED I/O
;       SUIO=1          ;SEQUENTIAL UNFORMATTED I/O
;       DFIO=1          ;DIRECT ACCESS FORMATTED I/O
        DUIO=1          ;DIRECT ACCESS UNFORMATTED I/O
;       LIO=1           ;LIST-DIRECTED SEQUENTIAL I/O
        ECIO=1          ;ENCODE/DECODE STATEMENTS
;       OFMT=1          ;RUN-TIME FORMAT COMPILATION
        CODSP=1         ;STACK-ORIENTED CODE SUPPORT
;       VIRT=1          ;VIRTUAL ARRAY SUPPORT
        I2MTH=1         ;INTEGER*2 FUNCTIONS AND SUPPORT ROUTINES
        I4MTH=1         ;INTEGER*4 FUNCTIONS AND SUPPORT ROUTINES
        SPMTH=1         ;REAL FUNCTIONS AND SUPPORT ROUTINES
        DPMTH=1         ;DOUBLE PRECISION FUNCTIONS AND SUPPORT ROUTINES
;       CPXMTH=1        ;COMPLEX FUNCTIONS AND SUPPORT ROUTINES
;
; Addition miscellaneous I/O routines, support routines, etc., are listed
; separately.
;
;******* NOTE *******
;
; If formatted or list-directed I/O processing routine are to be included,
; all of the following must be included or they cannot be used!!!
;
;               $FIO    ;FORMATTED I/O PROCESSOR
;               $LSTI   ;LIST-DIRECTED INPUT PROCESSOR
;               $LSTO   ;LIST-DIRECTED OUTPUT PROCESSOR
;               ICI$    ;INTEGER FORMAT CONVERSIONS
;               LCI$    ;LOGICAL FORMAT CONVERSIONS
;               RCI$    ;FLOATING POINT FORMAT CONVERSIONS
;
; 1. Math functions marked: (INLINE) are normally performed inline
;    and no external reference is required.
;
; 2. Some math functions contain both the single and double precision
;    versions in a single module (eg. $EXP).
;
; 3. Direct access I/O requires the definefile or open statement.
;
;*****  DO NOT INCLUDE A REFERENCE TO ANY OF:
;*****          $OTSVA
;*****          $SEQC
;*****          ORGSQ$
;*****          ORGRL$
;*****          ORGIX$
;
;-
; Modules always required (or almost always).
;
        .GLOBL  OTI$    ;MAIN PROGRAM INITIALIZATION
        .GLOBL  NAM$    ;TRACEBACK CHAIN PROCESSING (/TR COMPILER SWITCH)
;       .GLOBL  ARYCK$  ;ARRAY SUBSCRIPT CHECKING (/CK COMPILER SWITCH)
        .GLOBL  MAK1$   ;ONE-DIMENSION ADJUSTABLE ARRAYS
        .GLOBL  MAK2$   ;TWO-DIMENSION ADJUSTABLE ARRAYS
```

```
;       .GLOBL  MAKN$    ;MULTI-DIMENSION ADJUSTABLE ARRAYS
;
; Miscellansous I/O, compile-code, and statement support routines. These
; should be selected on a per statement basis.
;
        .GLOBL  STOP$    ;STOP/PAUSE STATEMENTS
        .GLOBL  REWI$    ;REWIND STATEMENT
        .GLOBL  ENDF$    ;ENDFILE STATEMENT
        .GLOBL  BKSP$    ;BACKSPACE STATEMENT
        .GLOBL  DEFF$    ;DEFINEFILE STATEMENT
        .GLOBL  CLOS$    ;CLOSE STATEMENT
        .GLOBL  OPEN$    ;OPEN STATEMENT
        .GLOBL  CGO$     ;COMPUTED GOTO STATEMENT
        .GLOBL  AGO$     ;ASSIGNED GOTO STATEMENT
;
; I/O processing routines.
;
        .IF DF SFIO      ;SEQUENTIAL FORMATTED I/O
        .GLOBL  ISF$     ;FORMATTED SEQUENTIAL INPUT
        .GLOBL  OSF$     ;FORMATTED SEQUENTIAL OUTPUT
        FMT=1            ;FORMATTING IN USE
        IO=1             ;I/O IN USE
        .ENDC

        .IF DF SUIO      ;SEQUENTIAL UNFORMATTED I/O
        .GLOBL  ISU$     ;UNFORMATTED SEQUENTIAL INPUT
        .GLOBL  OSU$     ;UNFORMATTED SEQUENTIAL OUTPUT
        IO=1             ;I/O IN USE
        .ENDC

        .IF DF LIO       ;LIST-DIRECTED I/O
        .GLOBL  ISL$     ;LIST-DIRECTED SEQUENTIAL INPUT
        .GLOBL  OSL$     ;LIST-DIRECTED SEQUENTIAL OUTPUT
        LFMT=1           ;LIST-DIRECTED FORMATTING IN USE
        IO=1             ;I/O IN USE
        .ENDC

        .IF DF DFIO      ;DIRECT ACCESS FORMATTED I/O
        .GLOBL  IRF$     ;FORMATTED DIRECT ACCESS INPUT
        .GLOBL  ORF$     ;FORMATTED DIRECT ACCESS OUTPUT
        FMT=1            ;FORMATTING IN USE
        IO=1             ;I/O IN USE
        .ENDC

        .IF DF DUIO      ;DIRECT ACCESS UNFORMATTED I/O
        .GLOBL  IRU$     ;UNFORMATTED DIRECT ACCESS INPUT
        .GLOBL  ORU$     ;UNFORMATTED DIRECT ACCESS OUTPUT
        IO=1             ;I/O IN USE
        .ENDC

        .IF DF ECIO      ;ENCODE/DECODE
        .GLOBL  ENF$     ;ENCODE/DECODE STATEMENTS
        FMT=1            ;FORMATTING IN USE
        IO=1             ;I/O IN USE
        .ENDC
```

```
;
; Common I/O support routines.
;
        .IF DF IO
        .GLOBL  IOAI$    ;I/O VARIABLE TRANSMISSION
        .GLOBL  IOAA$    ;I/O ARRAY TRANSMISSION
        .ENDC
;
; Formatting capabilities.
;
        .IF DF FMT
        .GLOBL  $FIO     ;FORMATTED I/O PROCESSOR
        .ENDC

        .IF DF OFMT
        .GLOBL  FMTCV$   ;RUN-TIME FORMAT COMPILATION
        .ENDC

        .IF DF LFMT
        .GLOBL  $LSTI    ;LIST-DIRECTED INPUT PROCESSOR
        .GLOBL  $LSTO    ;LIST-DIRECTED OUTPUT PROCESSOR
        .ENDC

        .IF DF FMT!LFMT
        .GLOBL  ICI$     ;INTEGER FORMAT CONVERSIONS
        .GLOBL  LCI$     ;LOGICAL FORMAT CONVERSIONS
        .GLOBL  RCI$     ;FLOATING POINT FORMAT CONVERSIONS
        .ENDC
;
; Stack-oriented compiled code support routines.
;
        .IF DF CODSP

        .GLOBL  PWIIS$   ;INTEGER*2 EXPONENTIATION
        .GLOBL  PWRIS$   ;REAL/DOUBLE ** INTEGER
        .GLOBL  PWRRS$   ;REAL ** REAL
;       .GLOBL  PWRDS$   ;REAL/DOUBLE ** REAL/DOUBLE

        .IF DF I4MTH
        .GLOBL  MLJS$    ;INTEGER*4 MULTIPLICATION
        .GLOBL  DVJS$    ;INTEGER*4 DIVISION
;       .GLOBL  PWIJS$   ;INTEGER*4 EXPONENTIATION
        .ENDC

        .IF DF CPXMTH
        .GLOBL  PWCIS$   ;COMPLEX ** INTEGER
        .GLOBL  ADCS$    ;COMPLEX ADD/SUBTRACT
        .GLOBL  MLCS$    ;COMPLEX MULTIPLICATION
        .GLOBL  DVCS$    ;COMPLEX DIVISION
        .GLOBL  CMCS$    ;COMPLEX COMPARE
        .GLOBL  TSCS$    ;COMPLEX TEST FOR ZERO
        .GLOBL  NGCS$    ;COMPLEX NEGATE
        .ENDC

; If any of these are selected, also include the stack swap.
```

```
        .GLOBL  SWP11$  ;STACK SWAP

        .ENDC
;
; Virtual array support
;
        .IF DF VIRT
        .GLOBL  MAKV$   ;ADJUSTABLE VIRTUAL ARRAY INITIALIZE
        .GLOBL  IOAVA$  ;VIRTUAL ARRAY I/O TRANSMISSION
        .GLOBL  VRTB$   ;VIRTUAL ADDRESS TRANSLATION
        .ENDC
;
; Math library functions.
;
        .IF DF I2MTH
        .GLOBL  $IABS   ;INTEGER*2 ABSOLUTE VALUE (INLINE)
        .GLOBL  $MAX0   ;INTEGER*2 MAX OF INTEGER*2 (INLINE)
        .GLOBL  $AMAX0  ;REAL MAX/MIN OF INTEGER*2 (INLINE)
        .GLOBL  $MIN0   ;INTEGER*2 MIN OF INTEGER*2 (INLINE)
        .GLOBL  $NINT   ;INTEGER NEAREST INTEGER OF REAL/DOUBLE
        .GLOBL  $IMOD   ;INTEGER*2 MOD (INLINE)
        .GLOBL  $ISIGN  ;INTEGER*2 SIGN TRANSFER
        .GLOBL  $IDIM   ;INTEGER*2 DIM
        .GLOBL  $IAND   ;INTEGER*2 AND (INLINE)
        .GLOBL  $IOR    ;INTEGER*2 OR (INLINE)
        .GLOBL  $IEOR   ;INTEGER*2 EXCLUSIVE OR (INLINE)
        .GLOBL  $INOT   ;INTEGER*2 NOT (INLINE)
        .GLOBL  $ISHFT  ;INTEGER*2 SHIFT
        .ENDC

        .IF DF I4MTH
        .GLOBL  $JABS   ;INTEGER*4 ABSOLUTE VALUE (INLINE)
        .GLOBL  $JMAX0  ;REAL/INTEGER*4 MAX/MIN OF INTEGER*4
        .GLOBL  $JMOD   ;INTEGER*4 MOD
        .GLOBL  $JSIGN  ;INTEGER*4 SIGN TRANSFER
        .GLOBL  $JDIM   ;INTEGER*4 DIM
        .GLOBL  $JAND   ;INTEGER*4 AND (INLINE)
        .GLOBL  $JOR    ;INTEGER*4 OR (INLINE)
        .GLOBL  $JEOR   ;INTEGER*4 EXCLUSIVE OR (INLINE)
        .GLOBL  $JNOT   ;INTEGER*4 NOT (INLINE)
        .GLOBL  $JSHFT  ;INTEGER*4 SHIFT
        .ENDC

        .IF DF SPMTH
        .GLOBL  $ABS    ;REAL ABSOLUTE VALUE (INLINE)
;       .GLOBL  $EXP    ;REAL/DOUBLE EXPONENTIAL
;       .GLOBL  $ALOG   ;REAL NATURAL/COMMON LOGARITHM
;       .GLOBL  $SIN    ;REAL SINE/COSINE
;       .GLOBL  $SQRT   ;REAL/DOUBLE SQUARE ROOT
;       .GLOBL  $TAN    ;REAL TANGENT
;       .GLOBL  $ACOS   ;REAL ARC COSINE
;       .GLOBL  $ASIN   ;REAL ARC SINE
;       .GLOBL  $ATAN   ;REAL ARC TANGENT
;       .GLOBL  $COSH   ;REAL HYPERBOLIC COSINE
;       .GLOBL  $SINH   ;REAL HYPERBOLIC SINE
```

```
;       .GLOBL  $TANH   ;REAL HYPERBOLIC TANGENT
        .GLOBL  $AMAX1  ;REAL/INTEGER*2 MAX/MIN OF REAL (INLINE)
        .GLOBL  $JMAX1  ;INTEGER*4 MAX/MIN OF REAL (INLINE)
        .GLOBL  $ANINT  ;REAL/DOUBLE NEAREST INTEGER OF REAL/DOUBLE
        .GLOBL  $AINT   ;REAL TO REAL TRUNCATION
        .GLOBL  $AMOD   ;REAL MOD
        .GLOBL  $SIGN   ;REAL SIGN TRANSFER
        .GLOBL  $FLOAT  ;INTEGER*2 TO REAL CONVERSION (INLINE)
        .GLOBL  $FLOTJ  ;INTEGER TO REAL/DOUBLE CONVERSION (INLINE)
        .GLOBL  $IFIX   ;REAL/DOUBLE TO INTEGER*2 CONVERSION (INLINE)
        .GLOBL  $JFIX   ;REAL/DOUBLE TO INTEGER*4 CONVERSION (INLINE)
        .ENDC

        .IF DF DPMTH
        .GLOBL  $DABS   ;DOUBLE ABSOLUTE VALUE (INLINE)
;       .GLOBL  $DLOG   ;DOUBLE NATURAL/COMMON LOGARITHM
;       .GLOBL  $DSIN   ;DOUBLE SINE/COSINE
;       .GLOBL  $DTAN   ;DOUBLE TANGENT
;       .GLOBL  $DACOS  ;DOUBLE ARC COSINE
;       .GLOBL  $DASIN  ;DOUBLE ARC SINE
;       .GLOBL  $DATAN  ;DOUBLE ARC TANGENT
;       .GLOBL  $DCOSH  ;DOUBLE HYPERBOLIC COSINE
;       .GLOBL  $DSINH  ;DOUBLE HYPERBOLIC SINE
;       .GLOBL  $DTANH  ;DOUBLE HYPERBOLIC TANGENT
        .GLOBL  $DMAX1  ;DOUBLE MAX/MIN OF DOUBLE (INLINE)
        .GLOBL  $DINT   ;DOUBLE TO DOUBLE TRUNCATION
        .GLOBL  $DMOD   ;DOUBLE MOD
        .GLOBL  $DSIGN  ;DOUBLE SIGN TRANSFER
        .GLOBL  $DDIM   ;DOUBLE DIM
        .GLOBL  $SNGL   ;DOUBLE TO REAL CONVERSION (INLINE)
        .GLOBL  $DBLE   ;REAL TO DOUBLE CONVERSION (INLINE)
        .GLOBL  $DPROD  ;DOUBLE PRODUCT OF 2 REALS
        .ENDC

        .IF DF CPXMTH
        .GLOBL  $CABS   ;COMPLEX ABSOLUTE VALUE
        .GLOBL  $CEXP   ;COMPLEX EXPONENTIAL
        .GLOBL  $CLOG   ;COMPLEX LOGARITHM
        .GLOBL  $CSIN   ;COMPLEX SINE/COSINE
        .GLOBL  $CSQRT  ;COMPLEX SQUARE ROOT
;       .GLOBL  $REAL   ;COMPLEX TO REAL CONVERSION (INLINE)
;       .GLOBL  $AIMAG  ;COMPLEX IMAGINARY TO REAL CONVERSION
;       .GLOBL  $CMPLX  ;REAL TO COMPLEX CONVERSION
;       .GLOBL  $CONJG  ;COMPLEX CONJUGATE
        .ENDC
;
; System subroutines, see appendix D of the Fortran-IV-Plus Users Guide.
;
        .GLOBL  ASSIGN
        .GLOBL  DATE
        .GLOBL  ERRSET
        .GLOBL  ERRSNS
        .GLOBL  ERRTST
        .GLOBL  FDBSET
        .GLOBL  IDATE
        .GLOBL  IRAD50
        .GLOBL  RAD50
;       .GLOBL  RAN
;       .GLOBL  RANDU
        .GLOBL  R50ASC
        .GLOBL  SECNDS
        .GLOBL  TIME
        .GLOBL  USEREX
        .GLOBL  CLOSE
        .GLOBL  EXIT
        .END
```

FCSFSLVSC.MAC

This is the listing of FCSFSLVEC we use to map the FCSFSL entry points.

```
        .TITLE  FCSFSLVEC
        .IDENT  /V01/

; FCSFSLVEC.MAC must have addresses which are the internal addresses
; of FCSFSL entry points.  To get these produce a map of FCSFSL
;
;       TKB NL:A/-HD,MP:FCSFSL/CR/-SP=[1,1]FCSFSL.STB
;
; The cross reference portion of the MAP can be edited (USE EDI) to
; be in the appropriate form.  (Try it)
; The form is:
;               SV      SYMBOL VALUE
        .PAGE
        .SBTTL  MACRO DEF'S AND CALLING STUFF
;
; This MACRO (SV) creates a subroutine call to the entry point in
; the FCSFSL supervisor mode library, one call for each entry point.

; LAB is the symbol name of an entry point in FCSFSL.
; ADDR is the absolute address of the symbol from the FCSFSL MAP.
        .MACRO  SV LAB ADDR
LAB::   JSR     PC,@ SUPL               ; CALL APPROPRIATE LABEL
        .WORD   ADDR
        .ENDM
;
;
        .PSECT $$SUPL,RO,I
SUPL:   MOV     @(SP)+,-(SP)            ; PUSH ADDRESS ON STACK
;       CSM     (SP)+                   ; CALL SUPR MODE
        7026                            ; NOT IN ASSEMBLER TABLE
        RTS     PC
;
        .PAGE
        .SBTTL  VECTORS
        .PSECT  $$SVLC,RO,D
        SV      $CMPCS  000014
        SV      $DSPAT  000652
        SV      $ULA    000032
        SV      $ULAFD  000076
```

```
SV      $ULAIN  000360
SV      .ASCPP  022536
SV      .ASLUN  014016
SV      .CLOSE  001064
SV      .CTRL   001412
SV      .DCCVT  023424
SV      .DELET  001444
SV      .DLFNB  001536
SV      .ENTER  001622
SV      .EXTND  001714
SV      .FATAL  012766
SV      .FIND   001662
SV      .FINIT  017262
SV      .GET    002020
SV      .LGETSQ 001060
SV      .GTDID  003120
SV      .GTDIR  003136
SV      .MARK   003226
SV      .MRKDL  003254
SV      .ODCVT  023420
SV      .OPEN   003326
SV      .OPFID  001050
SV      .OPFNB  005540
SV      .PARSE  013762
SV      .POINT  003166
SV      .POSIT  021660
SV      .POSRC  015176
SV      .PPASC  022672
SV      .PPR50  023734
SV      .PRINT  000460
SV      .PRSDI  020446
SV      .PRSDV  020710
SV      .PRSFN  021110
SV      .PUT    007760
SV      .PUTSQ  001054
SV      .RDFDR  011452
SV      .RDFFP  011526
SV      .RDFUI  011606
SV      .READ   011650
SV      .REMOV  001642
SV      .RENAM  011716
SV      .RFOWN  011556
SV      .TRNCL  012210
SV      .VRCVT  023420
SV      .WAIT   012414
SV      .WDFDR  011476
SV      .WDFFP  011540
SV      .WDFUI  011620
SV      .WFOWN  011570
SV      .WRITE  012436
SV      .XQIO   012562

        .END
```

SPM-11M

Ralph Stamerjohn
Monsanto
St. Louis, Missouri

The following slides are from the Software Performance Monitor (SPM-11M) presentation made by Robert Evans of Digital at the Spring, 1981 Miami DECUS Symposium. SPM-11M is a new service offered by Digital Software Support. We are currently using it at Monsanto and are very favorably impressed with it.

As the slides explain, SPM-11M is a event driven performance monitor. Patches are made to various modules in the executive to add what are called hook points. Whenever a hook is reached, a dispatch is made to the monitor dispatch routine and the event is logged into a core buffer. When the buffer is filled, it is written either to disk or magtape. A data reduction and report task is provided to process the data.

To me, SPM-11M is one of the most exciting things Digital has done with RSX-11M in a long time. At last, I can see exactly what is happening to my data systems and accurately measure throughput. Some of our initial impressions are:

1. The package installed very easily. While a system generation is required, no problems where encountered. The package has been used for over a month and no bugs have been encountered in either data collection or reduction.

2. Because the package is event driven and a KW11-P 100KHZ clock (10 microseconds resolution), the data collected is very exact and the resolution is superb.

3. The package measures just about everything I would want measured (see below).

SPM-11M is a good system performance analysis tool. It is not an RSX-11M accounting package (to begin with, it collects far too much data for accounting purposes). In addition, SPM-11M is only an adequate task performance analysis tool. The distinction between system and task is critical. SPM-11M will tell you what tasks in your system are using what resources of the system. However, the system-type granularity of the data is not adequate for good task analysis. For example, SPM-11M reports QIO's at the device level (DB0:, DB1:, etc.). So I could learn that task FOO is issuing 100,000 disk QIO's to accomplish its work. However, for me to improve task FOO's performance, what I would really like to know is a breakdown of QIO's on a per logical unit number basis. Then I would have a much better idea of where to go to improve FOO's performance.

In summary, SPM-11M works as Digital claims. While there are many improvements I could suggest, the package is a quantum step forward in Digital's offerings for RSX-11M performance analysis. If you want further details, contact your local software office.

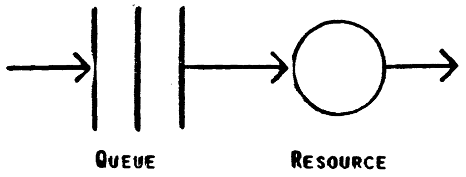
SPM GOALS

- EVENT DRIVEN
- PRECISE TIMING
- ONLY MAPPED SYSTEMS
- RECORD TO DISK OR TAPE
- SELECTIVITY

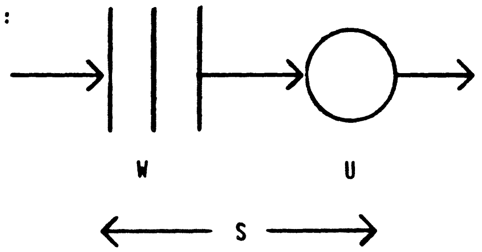
RSX-11M RESOURCE MODEL

RESOURCES:
1. CPU
2. MAIN MEMORY
3. I/O DEVICES
4. FILE SYSTEM
5. TASK LOADER

RESOURCE MODEL:

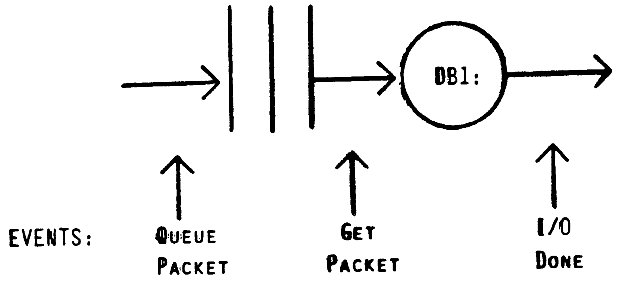


INTERVALS:



U = USAGE TIME
W = WAIT TIME
S = SERVICE TIME (U+W)
I = INTER-ARRIVAL TIME

I/O DEVICE MODEL:



U = I/O DONE - GET PACKET
W = GET PACKET - QUEUE PACKET
S = I/O DONE - QUEUE PACKET
I = QUEUE PACKET (2) - QUEUE PACKET (1)

TASK METRICS

1. Count

2. Total Time

3. % Resident Time

4. Intervals
   - A. Minimum
   - B. Mean
   - C. Maximum
   - D. Coefficient of Variation

SYSTEM METRICS

- Sample Interval in SECONDS

- Resources:

  1. CPU
     - A. % User Mode
     - B. % Kernel Mode
     - C. % Idle

  2. Memory
     % ~~Time~~ SPACE Used

  3. Device
     % Time Used

SPM-11M

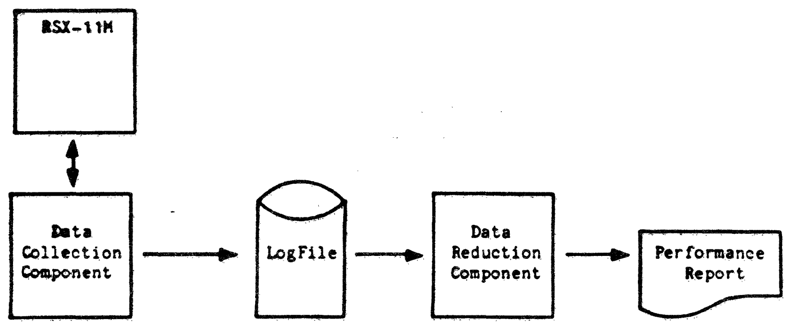


Figure 1

DATA COLLECTION COMPONENT

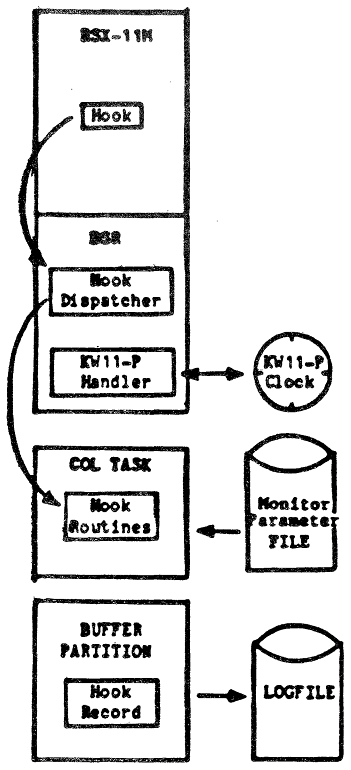


Figure 2

MODIFICATIONS TO RSX-11M EXECUTIVE

- CODE CONDITIONAL ON P$$MON

```
RSXMC - SELECT PERFORMANCE MONITOR SUPPORT

        P$$MON = 0

LOWCR - EXTRA STACK SPACE

        .BLKW   16.

XXXXX - HOOK POINTS

        CALL @$HKDSP     ; CALL PERFORMANCE MONITOR
        .WORD H$XXXX     ; HOOK ID

SYSCM - HOOK DISPATCHER POINTER

  $HKDSP::  .WORD  $HKRET

SYSXT - DUMMY HOOK RETURN

$HKRET::  MFPS          -(SP)     ; SAVE PSW
          ADD       #2,2(SP)      ; SKIP OVER HOOK ID
          MTPS          (SP)+     ; RESTORE PSW
          RETURN
```

COL TASK

- TASK LEVEL CODE
  - NON-AST: 1. INITIALIZATION

    2. START DATA COLLECTION

    3. WRITE TO LOGFILE

    4. STOP DATA COLLECTION

    5. CLEANUP AND EXIT
  - AST: SYSTEM METRICS

- EXECUTIVE LEVEL CODE
  - HOOKS CALL ROUTINES RESIDENT IN COL
  - PSW AND R0 $\longrightarrow$ R5 SAVED/RESTORED
  - EXTRA 16 WORDS ADDED TO SYSTEM STACK
  - HOOK ROUTINES EXECUTE IN SAME CONTEXT AS HOOK

DATA COLLECTION MODES

- MANUAL
- AUTOSTOP
- REPEAT

DATA COLLECTION SELECTIVITY

- EVENT CLASS

  --CPU HOOKS

  --MEMORY/LOADER HOOKS

  --DEVICE/ACP HOOKS
- NAMED TASKS
- DEVICE CLASS

  --DIRECTORY (DISK/DECTAPE)

  --SEQUENTIAL (MAGTAPE/CASSETTE)

  --TERMINAL

  --OTHER (PRINTER, REAL-TIME)

RSX-11M MONITOR DATA REDUCTION

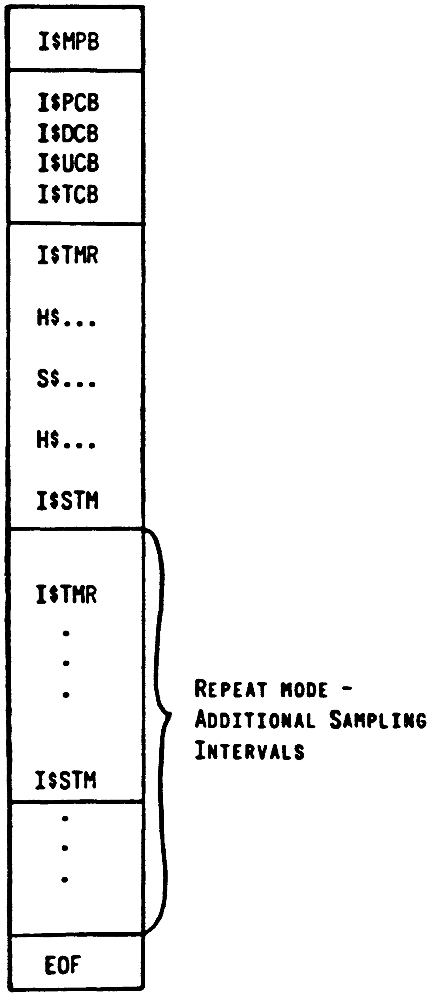
- NON-PRIVILEGED UTILITY TO READ LOGFILE.

  PRODUCING PERFORMANCE REPORT

- MULTIPLE TYPES OF OUTPUT

- HIGH DEGREE OF PARAMETERIZATION

LOG FILE STRUCTURE



PARAMETERIZATION OF DATA REDUCTION

- SELECTIVE TRACE DUMP
- DETAIL LEVEL OF TASK-BY-TASK REPORT
  - ALL INVOCATIONS
  - COMPLETE INVOCATIONS
  - TASK + TI SUMMARY
  - TASK SUMMARY
  - SYSTEM-WIDE SUMMARY
- TIME WINDOW FILTER
- TASKNAME FILTER
- TI FILTER
- ELIMINATION OF INCOMPLETE TASK INVOCATIONS
- ALL REPORTS OPTIONAL EXCEPT MPB

SAMPLE HOOK TRACE

```
rec H$LCTX 000004 027351 ...RMD TT20: 000000 047225
rec H$SCTX 000005 126003 ...RMD TT20: 000001 125503
rec H$LCTX 000005 132617 ...RMD TT20: 000001 131402
rec H$SCTX 000005 151426 ...RMD TT20: 000001 133117
rec H$LCTX 000006 022162 ...RMD TT20: 000002 002610
rec H$SCTX 000026 004544 ...RMD TT20: 000020 126401
rec H$LCTX 000026 004600 QMG... C00: 000020 126401
rec H$SCTX 000026 077271 QMG... C00: 000021 017472
rec H$LCTX 000026 077325 ...RMD TT20: 000021 017472
rec H$SCTX 000027 146332 ...RMD TT20: 000022 047225
rec H$RTSK 000027 147163 ...TKB TT15: 002402
rec H$QTRP 000027 147212 ...TKB TT15:
rec H$QLDR 000027 147243 ...TKB TT15:
rec H$GLDR 000027 147425 ...TKB TT15: 055062 116110 001074 000400
rec H$FLDR 000027 167041 ...TKB TT15:
rec H$LCTX 000027 167551 ...TKB TT15: 000022 065745
rec H$QDRV 000027 167726 ...TKB TT15: 051044 001010 056724
rec H$GPKT 000027 167765 ...TKB TT15: 056724
rec H$IODN 000027 171011 ...TKB TT15: 056724
rec H$QDRV 000027 171162 ...TKB TT15: 051044 001010 056724
rec H$GPKT 000027 171221 ...TKB TT15: 056724
rec H$IODN 000027 173544 ...TKB TT15: 056724
rec H$QDRV 000027 173744 ...TKB TT15: 051044 001010 056724
rec H$GPKT 000027 174003 ...TKB TT15: 056724
```

```
RSX-11M Performance Monitor Data Reduction V01.000          30-MAR-81 09:50:

                                       No Noise Tasks

 rec H$IODN 000030 000114 ...TKB TT15: 056724
 rec H$EXTK 000030 000575 ...TKB TT15: 116110 001074
 rec H$QDRV 000030 001117 ...TKB TT15: 051044 001010 056724
 rec H$GPKT 000030 001157 ...TKB TT15: 056724
 rec H$SCTX 000030 002540 ...TKB TT15: 000022 075727
 rec H$LCTX 000030 002574 ...RMD TT20: 000022 075727
 rec H$IODN 000030 004720 ...TKB TT15: 056724
 rec H$SCTX 000030 023765 ...RMD TT20: 000022 075727
 rec H$LCTX 000030 024022 ...TKB TT15: 000022 075727
 rec H$QDRV 000030 024777 ...TKB TT15: 051044 001010 057210
 rec H$GPKT 000030 025037 ...TKB TT15: 057210
 rec H$IODN 000030 027022 ...TKB TT15: 057210
 rec H$QAC1 000030 027413 ...TKB TT15: 051044 012000 057210
 rec H$SCTX 000030 027456 ...TKB TT15: 000022 077502
 rec H$IOFN 000030 051213 ...TKB TT15: 057210
```

```
RSX-11M Performance Monitor Data Reduction V01.000        15-MAY-81 08:37:10          Page 2          Sampling Intervals
                                        Base level 1 10-FEB-81

Data Collection Started 28-FEB-81 14:08:14.8 000004 040758
               0         +        20         +        40         +        60         +        80         +       100  (% utilization)

000137 147751  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU                       $$$$$$$
000137 147751  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
000137 147751  ddddddddddddddddddddddddddd

000273 060477  UUUUUUUUUUUUUU                                                                                   $$$
000273 060477  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
000273 060477  ddddddddd

000426 171454                                                                                                   $$$
000426 171454  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
000426 171454  d

000562 103352  UUUUUUUUUUUU                                                                                 $$$$$$$$
000562 103352  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
000562 103352  dddddddddddddddddddddddddddddddddddddddddddddd

000716 015072  UUUUUUUUUUUUUUUUUUUUUU                                                                    $$$$$$$$$$$
000716 015072  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
000716 015072  dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd

001051 126157  UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU                                                        $$$$$$$$$$$
001051 126157  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
001051 126157  dddddddddddddddddddddddddddddddddddddddddddddddddddddddddd

001205 037750  U                                                                                               $$$$
001205 037750  mmmmmmmmmmmmmmmmmmmmmmmmmmmmm
001205 037750  ddddd

001340 150237  UUUUU                                                                                            $$$
001340 150237  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
001340 150237  d

001474 060120  UUUUUUUU                                                                                        $$$$
001474 060120  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
001474 060120

001627 170450                                                                                                    $$
001627 170450  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
001627 170450

001763 102041                                                                                                    $$
001763 102041  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
001763 102041

002117 013570                                                                                                    $$
002117 013570  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
002117 013570

002252 125054                                                                                                    $$
002252 125054  mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
002252 125054
```

```
RSX-11M Performance Monitor Data Reduction V01.000        15-MAY-81 09:03:22          Page 1          Monitor Collection Parameters
                                        Sample MDR Report


               P E R F O R M A N C E     M O N I T O R     D A T A     C O L L E C T I O N     P A R A M E T E R S

Mode: MANUAL

Buffering in a DYNAMIC region called TRCPAR in Partition GEN    , of 2 Buffers with 12 Blocks per Buffer

System-wide metrics recorded every 60 seconds for the CPU, Partition GEN    , and Device LB0:

Task Workload data was limited to the TI: of TT20:

ALL DEVICES Selected for Task Workload

Task Workload Data Collection Enabled for All EXCEPT the following Task(s):
  F11ACP  ...MCR  MCR...  QMG...  TTP21   PRT...  MTAACP  F11MSG  ...LDR  ...INS

the following HOOK POINTS are ENABLED:
(Directives)
(Run/Exit)      RTSK  XTSK
(CPU Usage)     SCTX  LCTX
(Memory)        USTP  QAST  EXTK  QTRP  CREG  DREG
(Task Loader)   QLDR  QCHW  GLDR  FLDR
(In/Output)     QDRV  GPKT  IODN  IOFN  QAC1  QAC2
(System/Idle)   DRSV  INSV  INS2  SYXT  EIDL  XID1  XID2

MDR command line>TT:=TSATSB
```

```
RSX-11M Performance Monitor Data Reduction V01.000        15-MAY-81 09:37:10        Page 33        System-Wide Summary

                                  Base level 1 10-FEB-81

Elapsed Times (seconds):   0.123000E-02          68.8007            1591.53            3.31703             4747.25
                             minimum               mean              maximum           coef(var)             total

Number of Invocations = 69                   ( 60 complete )                     ( 9 incomplete )

Memory Times      seconds:  3305.18        1442.07        12.6465         % elapsed:  69.6231        30.3769        0.266396
                             in              out            wait                        in             out            wait

Memory Size in KWords:          0.50           11.28          28.09
                                min            mean           max

Space-Time Product:             134       37929.7         283.057        3.52284            10
                          residencies    total KWS        mean KWS    coef(var) KWS     incomplete

ACP Counts:          466               147                37               723             278                94                1
          access/deaccess      create/delete        read/write        directory     file attribute          extend             other

     RESOURCE                  +- - - - - - - - - - - - - - - - - USAGE  TIMES  IN  SECONDS  - - - - - - - - - - - - - -+
                         count     total time     % resident time      minimum          mean          maximum         coef(var)      % resource
Memory Residency          105      3305.18                             0.03809        31.47793      747.19274       3.14261
CPU Timeslices          11126      264.872          8.01383            0.00023         0.02381        5.24558       3.47226        16.60222
Checkpoint R + W           90      8.94475          0.270628           0.03124         0.09939        0.18310       0.385198        0.56066
Initial Loads              65      4.30589          0.130277           0.02924         0.06624        0.12759       0.361083        0.26989
DB1: Ovly Loads          2612      61.2241          1.85237            0.00169         0.02344        0.05614       0.587768        3.83754
DB1: QIOs                6127      139.030          4.20642            0.00155         0.02269        0.05614       0.570610        8.71441
DB2: QIOs                   4      0.971600E-01     0.293963E-02       0.01370         0.02429        0.04904       0.591236        0.00609
DK0: QIOs                 311      12.1721          0.368273           0.00833         0.03914        0.04030       0.519737E-01    0.76295

     RESOURCE                  +- - - - - - - - - - - - - - - - - WAIT   TIMES  IN  SECONDS  - - - - - - - - - - - - - -+
                         count     total time     % resident time      minimum          mean          maximum         coef(var)
Memory Residency          109      12.6465                             0.00021         0.11602        3.29156       3.66841
Checkpoint R + W           90      9.34998          0.282888           0.00043         0.10389        1.22032       1.77687
Initial Loads              65      0.317310         0.960038E-02       0.00044         0.00488        0.11617       3.72709
DB1: Ovly Loads          2612      14.9789          0.453195           0.00031         0.00573        1.03950       5.53750
DB1: QIOs                6127      28.3474          0.857664           0.00031         0.00463        1.03950       6.04803
DB2: QIOs                   4      0.417900E-01     0.126438E-02       0.00028         0.01045        0.02878       1.11916
DK0: QIOs                 311      47.2674          1.43010            0.00028         0.15199        0.15820       0.152853

     RESOURCE                  +- - - - - - - - - - - - - - - - SERVICE  TIMES  IN  SECONDS  - - - - - - - - - - - - - -+
                         count     total time     % resident time      minimum          mean          maximum         coef(var)
Memory Residency          104      3317.35                             0.03831        31.89758      747.36379       3.11452
Checkpoint R + W           90      18.2947          0.553517           0.03209         0.20327        1.29531       0.867839
Initial Loads              65      4.62320          0.139877           0.03053         0.07113        0.20147       0.435952
DB1: Ovly Loads          2612      76.2030          2.30556            0.00201         0.02917        1.09104       1.26214
DB1: QIOs                6127      167.377          5.06408            0.00186         0.02732        1.09104       1.19101
DB2: QIOs                   4      0.138950         0.420400E-02       0.01398         0.03474        0.04932       0.405660
DK0: QIOs                 317      59.7632          1.80817            0.00862         0.18853        0.19756       0.164077
F11ACP QIOs              1745      122.653          3.71092            0.00209         0.07029        1.91110       1.96275

     RESOURCE                          +- - - - - -  ARRIVALS  PER  SECOND  - - - - - -+
                     # arrivals              mean task rate              total rate
Memory Residency        109                  0.229607E-01               0.683214E-01
```

```
RSX-11M Performance Monitor Data Reduction V01.000        15-MAY-81 09:37:10        Page 31        Task Summary

                                  Base level 1 10-FEB-81

Generic TASK = ...TKB

Elapsed Times (seconds):     7.71086              31.1365             77.1988            1.04612             93.4096
                             minimum               mean              maximum           coef(var)             total

Number of Invocations = 3                    ( 3 complete )                      ( 0 incomplete )

Memory Times      seconds:  92.2898        1.11982        1.11913         % elapsed:  98.8012        1.19883        1.19809
                             in              out            wait                        in             out            wait

Memory Size in KWords:         17.88          17.88          17.88
                                min            mean           max

Space-Time Product:              11        1649.68         149.971        1.53714
                          residencies    total KWS        mean KWS    coef(var) KWS     incomplete

ACP Counts:          90                14                 1                20               0                 7                0
          access/deaccess      create/delete        read/write        directory     file attribute          extend             other

     RESOURCE                  +- - - - - - - - - - - - - - - - - USAGE  TIMES  IN  SECONDS  - - - - - - - - - - - - - -+
                         count     total time     % resident time      minimum          mean          maximum         coef(var)      % resource
Memory Residency            6      92.2898                             0.21040        15.38163       36.31574       0.924610
CPU Timeslices            955      29.9322          32.4328            0.00023         0.03134        0.57194       1.76877         1.87615
Checkpoint R + W            6      0.575980         0.624099           0.06007         0.09600        0.12653       0.278505        0.03610
Initial Loads               3      0.265320         0.287486           0.08701         0.08844        0.09022       0.150796E-01    0.01663
DB1: Ovly Loads           284      8.10533          8.78248            0.00245         0.02854        0.05519       0.560498        0.50804
DB1: QIOs                1633      34.2379          37.0983            0.00156         0.02097        0.05519       0.581442        2.14604

     RESOURCE                  +- - - - - - - - - - - - - - - - - WAIT   TIMES  IN  SECONDS  - - - - - - - - - - - - - -+
                         count     total time     % resident time      minimum          mean          maximum         coef(var)
Memory Residency            6      1.11913                             0.00024         0.18652        0.77645       1.48520
Checkpoint R + W            6      0.373900         0.405137           0.00043         0.06232        0.12851       0.840279
Initial Loads               3      0.337000E-02     0.365154E-02       0.00107         0.00112        0.00120       0.494758E-01
DB1: Ovly Loads           284      5.97202          6.47094            0.00031         0.02103        1.03950       3.78227
DB1: QIOs                1633      12.7224          13.7853            0.00031         0.00779        1.03950       5.46167

     RESOURCE                  +- - - - - - - - - - - - - - - - SERVICE  TIMES  IN  SECONDS  - - - - - - - - - - - - - -+
                         count     total time     % resident time      minimum          mean          maximum         coef(var)
Memory Residency            6      93.4089                             0.98685        15.56815       36.54936       0.910020
Checkpoint R + W            6      0.949880         1.02924            0.09885         0.15831        0.21888       0.266923
Initial Loads               3      0.268690         0.291137           0.08811         0.08956        0.09129       0.146549E-01
DB1: Ovly Loads           284      14.0774          15.2534            0.00276         0.04957        1.09104       1.70805
DB1: QIOs                1633      46.9603          50.8836            0.00188         0.02876        1.09104       1.63077
F11ACP QIOs               132      13.2270          14.3320            0.00214         0.10020        1.23098       1.56196

     RESOURCE                          +- - - - - -  ARRIVALS  PER  SECOND  - - - - - -+
                     # arrivals              mean task rate              total rate
Memory Residency          6                  0.642332E-01               0.376081E-02
CPU Timeslices          955                   10.2238                   0.598596
Checkpoint R + W          6                  0.642332E-01               0.376081E-02
Initial Loads             3                  0.321166E-01               0.188041E-02
DB1: Ovly Loads         284                   3.04037                   0.178012
```

RSX/IAS SIG TAPES (Part 1)

Ralph Stamerjohn
Monsanto
St. Louis, Missouri

A major activity of the RSX/IAS Special Interest Group is tape copy. At each DECUS symposium, the SIG collects and merges software from users into a tape set. The no-hassle, informal procedure works very well. An incredible potpourri of programs are on the tape sets, answering almost any need. The statistics are awesome. In the eight tapes sets since the Fall, 1977 San Diego Symposium, 13,551 files in 507 accounts have been submitted. Altogether, the tapes have 203,131 blocks of free material.

While responsible for the success of tape copy, the simple procedure is also a major problem when trying to find and use a program. There is no unified documentation on the tapes. In addition, the SIG imposes no standards on submissions and exercises no control on documentation. This makes it difficult to locate a submission to solve a particular problem.

This is my attempt to solve the problem. I started by copying each tape set to a separate virtual disk. Then, I obtained a complete directory of each tape. I edited the directories with TECO such that each line showed the filename, type, size and tape set. The eight directory files were then appended to each other. The resulting file, RSXSIG.SRD, has in it all files submitted on all tapes. The next step was to sort this file in four ways: filename, filetype, UIC by filename and UIC by filetype. The resulting four files, RSXFIL.SRD, RSXEXT.SRD, RSXDFL.SRD and RSXDEX.SRD give four different ways of locating files. The last two files are perhaps the most useful. They show the contents of each account across all tapes. This allows quick locating of the most recent copy of any particular file in an account.

Next, I briefly examined each directory and wrote an abstract for it. These abstracts form the major part of this document. For each directory, I have listed the program name(s), the abstract, and the tape sets the directory appears on. In addition, each entry is indexed by the program name and also subindexed by the category or categories it falls in. If you find a program you are interested in, consult the various SRD files for more information on what UIC's in what tapes sets the program was submitted.

NOTE

There is a lot of software on the tapes. I have tried my best to accurately describe each submission. Also, I have tried to use the most descriptive index categories I could think up. However, I am sure the abstracts and indexing could be improved. Please contact me with any comments you have so the improvements can be made.

[002,002] FALL 1979 TAPE
[002,002] SPRING 1980 TAPE

This account contains various master files for Fall 1979 (San Diego) and Spring 1980 (Chicago) tape sets. This includes the command files used to build the tapes and summaries of the contents of the tapes.

1979 - Fall - San Diego
1980 - Spring - Chicago

[070,001] ACONTR
[070,001] PRTINT
[070,001] OBJBRK
[070,001] TCLRNO
[070,001] TOWERP
[070,001] TWOCOL

The [70,*] accounts have early versions of Pascal. For more current compilers, the reader is referred to the Structured Languages SIG tapes. This account has various utilities written in Pascal and documentation on the various Pascal compilers found in the [70,*] accounts.

- ACONTR is a RSX-11D terminal logon accounting package.
- PRTINT is a program to output Pascal PASS1 internal format.
- OBJBRK is a object module disassembler.
- TCLRNO produces two-column files from RUNOFF output files.
- TOWERP is a Tower of Hanoi program.
- TWOCOL is a two column lister utility.

1977 - Fall - San Diego

[070,002] BSM PASCAL

This account contains the BSM Pascal macro code, run-time system, and library.

1977 - Fall - San Diego

[070,003] PASREF

PASREF is a cross reference utility for Pascal source files.

1978 - Spring - Chicago

[070,004] BSM PASCAL
[070,004] NBS PASCAL

This account contains the BSM Pascal PASS1 sources, objects, and build files. It also has the bootstrap code for the NBS Pascal.

1977 - Fall - San Diego
1978 - Spring - Chicago

[070,006] NBS PASCAL

This account contains the NBS Pascal PASS1 and PASS2 sources, objects, and build files.

1978 - Spring - Chicago

[070,007] BSM PASCAL

This account has RSX-11M replacement files for the BSM Pascal package. To generate a RSX-11M version of BSM Pascal, use these files to replace the RSX-11D versions found in the other [70,*] accounts.

1977 - Fall - San Diego
1978 - Spring - Chicago

[070,011] BSM PASCAL

This account contains the BSM Pascal PASS2 sources, objects, and build files.

1977 - Fall - San Diego

[070,013] BCPL

This account has the sources and runtime system for the RSX-11D version of BCPL.

1978 - Spring - Chicago

[070,016] PACK
[070,016] SWEDISH PASCAL

This account has the Swedish Pascal sources, manuals, and utility routines. Among the utility routines is PACK and its related programs LISTPACK, PACKLIB, and UNPACK. These programs provide a source-library mechanism.

1977 - Fall - San Diego
1978 - Spring - Chicago

[070,017] PASCAL-P

This account contains the PASCAL-P compiler.

1978 - Spring - Chicago

[100,100] SPRING 1978 TAPE

This account contains various master files for the Spring 1978 (Chicago) tape set. This includes a short description of the tape.

1978 - Spring - Chicago

[200,200] SPRING 1979 TAPE

This account contains a directory of the Spring 1979 (New Orleans) tape set. It also has a summary of the contents of the tape.

1979 - Spring - New Orleans

[200,211] FALL 1977 TAPE

This account has a short description of the contents of the Fall 1977 (San Diego) tape set.

1977 - Fall - San Diego

[300,001] CARTS SUBMISSIONS
[300,001] FALL 1978 TAPE
[300,001] FALL 1980 TAPE

[300,001] SPRING 1981 TAPE

This account has general information about tapes. In particular, it has descriptions of the Chicago LUG (CARTS) submissions. It also has other files used in creating the tapes. In particular, it has the concatenated README files and directories for the Spring 1981 (Miami) tape, Fall 1980 (San Diego) tape, and Fall 1978 (San Francisco) tapes.

NOTE

One key file from the Spring 1981 Miami tape is SUBMIT.DOC. This describes the prefered format and documentation for RSX SIG Tape submissions.

```
1977 - Fall   - San Diego
1978 - Spring - Chicago
1978 - Fall   - San Francisco
1979 - Spring - New Orleans
1979 - Fall   - San Diego
1980 - Spring - Chicago
1980 - Fall   - San Diego
1981 - Spring - Miami
```

[300,002] CARTS SUBMISSIONS
[300,002] BIGTPC
[300,002] TPC

This account usually has the current version of TPC. This is a program used by the RSX/IAS SIG to do fast copies of FLX format tapes. BIGTPC is the lastest version of TPC and can be used to copy FLX, DSC, or BRU tapes. TPC or BIGTPC work by coping the contents of a tape to a single file and then outputing the file to blank tape. The programs are optimized for fast disk and tape I/O. The account also has other information about the tape sets, in particular, the Chicago (CARTS) LUG submissions

```
1977 - Fall   - San Diego
1978 - Spring - Chicago
1978 - Fall   - San Francisco
1979 - Spring - New Orleans
1979 - Fall   - San Diego
1980 - Spring - Chicago
1980 - Fall   - San Diego
1981 - Spring - Miami
```

[300,010] TECO

TECO is a very versatile and powerful text editor. It is a character

oriented editor and has many possible commands, including a full range of macro-type commands that allow TECO command files to be written and executed. This is an early version of TECO (version 25).

1977 - Fall - San Diego

[300,011] RUNOFF

RUNOFF is a text preparation tool that takes free-format text and commands and produces a formatted, paginated, justified output file. RUNOFF is useful for generating documentation. This is an early version of RUNOFF that has been specially modified for Centronix printer support.

1977 - Fall - San Diego

[300,012] SRD

SRD is a directory utility with a wide range of file selection options including wild-character names and creation date. SRD outputs sorted directories and also has directory write-back and selective delete options.

1977 - Fall - San Diego

[300,013] DDT

DDT is a symbolic debugger similar to ODT but with many extensions. It knows how to output PDP-11 instructions and supports the use of STB files for global symbols definitions.

1977 - Fall - San Diego

[300,014] ISCDA

ISCDA is a modified version of the IAS core dump analyzer. It allows the memory dump to be read from a file instead of the crash device.

1977 - Fall - San Diego

[300,015] IAS IND

This is a version of the RSX-11M V3.0 indirect command file processor (IND) modified for IAS. IND allows command files to be executed from a terminal and has many commands that support indirect command symbols, string values, branches, and other useful features.

1977 - Fall - San Diego

[300,016] SUPMAC

SUPMAC is a set of macros which provide a structured programming capability for Macro-11.

1977 - Fall - San Diego

[300,017] FLECS

FLECS is a popular language that extends Fortran to include structured language constructs. FLECS is written as a pre-processor to Digital's Fortran compilers.

1977 - Fall - San Diego
1978 - Spring - Chicago

[300,020] FODT

FODT is an interactive debugging tool especially designed for Fortran IV. It has many of the same commands as Digital's ODT with some additional Fortran specific commands for data conversion and line-number breakpointing.

1977 - Fall - San Diego

[300,021] BASIC

This is a version of DOS BASIC with a number of extensions for RSX/IAS. All forms of file access are allowed including fixed and variable length records, sequential and random access, and shared access. Macro-11 subroutines are also permitted and a number of extension variables have been added. The submission is quite extensive and complete.

1977 - Fall - San Diego
1978 - Spring - Chicago
1978 - Fall - San Francisco
1979 - Spring - New Orleans
1981 - Spring - Miami

[300,022] BASIC GAMES

This account has various games written in BASIC. The games are from a variety of sources, but in general, will run on the BASIC in [300,21].

1977 - Fall - San Diego

[300,023] DISOBJ

DISOBJ is a object module disassembler. It will create a pseudo Macro-11 listing (without comments) from any object file or library. The listing can be easily edited and submitted as input to the Macro-11 assembler. A comment field is output with the ASCII and RAD50 text string for the field.

1977 - Fall - San Diego
1980 - Spring - Chicago
1981 - Spring - Miami

[300,024] XREF

XREF is a cross-reference program for Fortran sources. The input is a Fortran source and the resulting output is a cross reference by statement and line number of all symbols.

1977 - Fall - San Diego

[300,025] INFORM

INFORM is a RSX-11D or IAS program that reports on different pool structures. This includes an extended ATL listing, clock queue listing, detailed PUD information, and a partition map.

1977 - Fall - San Diego

[300,026] ATT
[300,026] CORZAP
[300,026] DCS
[300,026] MASSAGE
[300,026] PRO
[300,026] SEND
[300,026] SPD
[300,026] UICTRN
[300,026] WHO

This account holds various RSX-11D/IAS programs that perform a variety of system-type functions. The programs in this account include:

- o ATT displays and updates file header attributes. It is most useful for recovering locked files. It can also be used to change file header attributes.
- o CORZAP patches SCOM core.
- o DCS changes speed on a DZ or DH terminal.
- o MASSAGE is a modification to the IAS MCR to allow DCL type commands. It changes MCR to pass unrecognized commands to a catch-all task (...MAS) which can then operate on them.
- o PRO changes file protection. It supports wild-card specifications.
- o SEND sends a message to a specified TTY or all terminals.
- o SPD sets PUD characteristics.
- o UICTRN transfers an entire UIC, retaining header information.
- o WHO list system users and tasks by terminals.

1977 - Fall - San Diego
1981 - Spring - Miami

[300,027] DCLS

This account holds three programs which implement the Digital Command Language syntax for TYPE, DIRECTORY, and PRINT commands. The programs translate the commands into the appropriate PIP commands and pass the command to PIP on RSX-11D and IAS systems.

1977 - Fall - San Diego

[300,030] DEMO

DEMO is an enhanced version of the distributed RSX-11D or IAS DEMO program. This version supports the VT50 and VT52 as well as the VT05 and the special I/O features of the RSX-11D V6.2 TT handler.

1977 - Fall - San Diego

[300,031] INDEX

INDEX is a Fortran IV cross-reference program conditionally written for RT-11 or RSX. It outputs a line numbered source listing and a cross-reference listing of each symbol and an indication of the type of usage.

1977 - Fall - San Diego

[300,034] MARGOT

MARGOT is a set of macros for defining a command syntax and an interpreter that will process a command line into its syntax elements.

1977 - Fall - San Diego

[300,035] VAPP
[300,035] VPDRV

VAPP is a plotting package designed for use with the Versatec electrostatic printer/plotter. Included in the account is a RSX-11M device driver for the Versatec printer/plotter.

1977 - Fall - San Diego

[300,036] SYDEV

SYDEV is a collection of programs and patches to allow 'SY:' redirection on an individual terminal basis for RSX-11D V6.2 systems. The feature is similar in function to the RSX-11M ASN command.

1977 - Fall - San Diego

[300,037] ACCLOG
[300,037] DRVGEN
[300,037] FRC
[300,037] FRG
[300,037] SPQ
[300,037] WHAM

This account holds a variety of RSX-11M system-type programs. Included in the account are the following:

- o ACCLOG is a task that will produce performance statistics for specified tasks in the system. This includes the number of QIO's issued by the tasks, total run-time in tics, and percentage of CPU time.
- o DRVGEN is a command file that will generate driver sources and is intended as a tutorial and aid in writing RSX-11M device drivers.
- o FRC is a task which will force a command line on another terminal.
- o FRG is a task which will list the disk fragmentation of a volume.
- o SPQ is a task which will list the queued files to the RSX-11M print spooler (PRT...).
- o WHAM is a task which will force all other tasks in a system partition to checkpoint. This can force alignment of a partition when the shuffler is not installed or manual alignment is wanted.

1977 - Fall - San Diego
1978 - Spring - Chicago

[300,040] CSMP
[300,040] V

This account has two entries. CSMP is a continuous system modeling program. V is a video ping-pong game.

1977 - Fall - San Diego

[300,041] VOLUME
[300,041] WORLDBOOK SUBROUTINES
[300,041] WORLDBOOK TECO MACROS

This account has three entries:

- o The TECO macros include various functions:
  - * CALC.TEC performs arithmetic operations.
  - * DAR.TEC converts RAD50 (octal) words into ASCII strings.
  - * LIST.TEC lists files on lineprinter with headers and page breaks.
  - * LISTTEC.TEC list files but converts control characters for printing. It is used for outputing TECO macros.

  - * QSINUSE.TEC reports on the Q-register usage.
  - * RAD.TEC converts ASCII strings into RAD50 (octal) words.
  - * READ.TEC counts the records in a file.
  - * STATUS.TEC outputs the current status of TECO.
  - * UPCASE.TEC converts a file from lower to upper case.
- o VOLUME is a RSX-11D or IAS program to scan the PUD entries and for each mounted volume, report the volume name and number of users.
- o The WORLDBOOK subroutines are a set of Fortran callable subroutines to perform string manipulation.
  - * ISIZE returns the size of a string.
  - * INDEX searches a string for a substring and returns the starting character position.
  - * LEQ determines if two strings are lexically equal.
  - * STRING assigns one string to another.
  - * NUL assigns a null string to a string variable.
  - * SUBSTR assigns a substring from a string to another string variable.
  - * CONCAT concatenates two strings.
  - * DUPL duplicates one string the specified number of times.
  - * TRUNC truncates a string to the specified position.
  - * SDELETE deletes characters from a string, starting from the specified position.
  - * SINSRT inserts one string into another, starting at the specified position.
  - * IBREAK determines if any character in a break set is found in a string and returns a true/false indication.

1978 - Spring - Chicago

[300,042] MCRDRV

MCRDRV is a RSX-11M driver that allows a task to issue MCR command lines using the QIO mechanism. This allows tasks to perform such things as mount or dismount a disk, start indirect command files, or other MCR type operations.

1978 - Spring - Chicago

[300,043] CAMDRV

CAMDRV is a RSX-11M driver for CAMAC crates. The driver allows a task to interface with a BD-011 or JY411 parallel CAMAC branch highway.

1978 - Fall - San Francisco
1979 - Spring - New Orleans

[300,044] FALL 1977 TAPE
[300,044] SPRING 1978 TAPE

This account contains directories of the Fall 1977 (San Diego) and Spring 1978 (Chicago) SIG tapes.

1978 - Fall - San Francisco

[300,045] RSX11M
[300,045] RSXTEC
[300,045] SYSTUN

This account contains three sets of documentation of application programming for RSX-11M. RSX11M is a tutorial description of RSX features. SYSTUN is some notes on tuning RSX-11M for performance. RSXTEC is notes on some techniques to optimize RSX-11M for specific applications.

1978 - Fall - San Francisco
1980 - Fall - San Diego

[300,046] RUNOFF PATCHES

This account has SLP patch files for the RUNOFF distributed in [300,11].

1979 - Spring - New Orleans
1979 - Fall - San Diego

[300,047] ILLEGAL INSTRUCTION TRAP

This account contains documentation and code to implement the illegal instruction trap mechanism in RSX-11M. This mechanism allows sites to add executive-type services without modifing RSX-11M.

1979 - Spring - New Orleans
1979 - Fall - San Diego

[300,050] GENCOM

GENCOM is two Fortran callable routines to allow a task to create a dynamic region and load it from a file (GENCOM) and to allow other tasks to map the region (ATHNMP).

1979 - Spring - New Orleans

[300,051] DOCUMENT/SOURCE CONTROL

This account contains various programs and procedures for managing projects, particularly source control.

1979 - Spring - New Orleans
1980 - Spring - Chicago

[300,052] SAI SUBROUTINES

This account contains a set of Fortran callable subroutines to allow a Fortran program to perform FCS I/O with all the control normally available to the Macro programmer. The following routines are included in the submission:

- INIT initializes the Fortran array to be used as the FDB.
- OPEN issues a normal FCS open call.
- OFID issues a "open-by-file-ID" using the filename block setup by the PARSE and FIND calls.
- OFNB issues a "open-by-filename-block" using the filename block setup by the PARSE call.
- OPNTD opens a temporary file and marks it for delete on close.
- PARSE performs an FCS .PARSE operation.

- FIND performs an FCS .FIND operation.
- DABIO provides read/write virtual block (READ$/WRITE$) I/O.
- DARIO provides fixed length, random record (GETR$/PUTR$) I/O.
- RECIO provides record I/O (GET$/PUT$) I/O.
- FQIO issues Files-11 read/write attribute QIO's.
- RENAM renames a file.
- DELETE performs a "delete-by-filename-block" operation.
- CLOS performs a FCS CLOSE operation.
- SPEC outputs device specific I/O for magtapes.
- WAIT waits for block I/O to finish.

1979 - Spring - New Orleans
1979 - Fall - San Diego
1980 - Spring - Chicago

[300,053] PRT PATCH

This is a SLP patch to the PRT despooler sources to remove the two blank pages printed between files.

1979 - Spring - New Orleans

[300,054] F11ACT

F11ACT is a RSX-11D/IAS program to monitor F11ACP I/O and log the data to a file. It is useful for determining disk loading and structuring the ACP and application for optimal performance.

1979 - Spring - New Orleans

[300,055] EVALU8

EVALU8 is a RSX-11D program to measure resource allocation. It will measure partition holes, pool resources, and task activity.

1979 - Spring - New Orleans

[300,056] MOTOROLA TECO MACROS
[300,056] TECO PATCHES

This account has some modifications to TECO V34 and TECO macros. The modifications are to the EI command to search the user's area first and then a default area and I/O support for Hazeltine 1500 terminals. The TECO macros include the following:

- COPYNOTE.TEC sets up a file with copyright notices.
- CVL.TEC converts a RUNOFF format file in upper case to lower case and removes the RUNOFF case control flags.
- GET.TEC and STRIP.TEC edit a SRD listing into a PIP copy command file.
- IND.TEC formats assembly source programs.
- MT.TEC reads one line of input from the terminal.
- STATUS.TEC and QS.TEC report the status of TECO and the Q-registers.
- TAD.TEC creates files to put a SLP audit trail on an updated file.

1979 - Fall - San Diego

[300,057] SRD

SRD is a directory utility with a wide range of file selection options including wild-character names and creation date. SRD outputs sorted directories and also has directory write-back and selective delete options. This version has many bugs fixed and has several extensions. It is one of the versions of SRD on the tapes that should be considered if you are bring up SRD.

1979 - Fall - San Diego
1980 - Fall - San Diego
1981 - Spring - Miami

[300,062] ABSZAP
[300,062] ATT
[300,062] CORZAP
[300,062] DSKFIX
[300,062] FHD
[300,062] LUT
[300,062] PAG

This account has several utility tasks for RSX-11D/IAS. Included in the set are the following:

- o ABSZAP is a one-liner absolute core patch program.
- o ATT list or updates the attributes of a file. It is most useful for unlocking a file and reseting the EOF.
- o CORZAP is a one-liner SCOM patcher.
- o DSKFIX is a disk utility for absolute block examination or patching.
- o FHD list the file header (index entry) for a file.
- o LUT list the logical units assignments for an active task.
- o PAG produces paginated listings of source files.

1980 - Spring - Chicago

[300,063] PRXFMT

PRXFMT is a PRAXIS source formatting program. It capitalizes PRAXIS keywords and sets other words in lower case (except in literals or comments) and does block indenting.

1980 - Spring - Chicago

[300,064] CLI
[300,064] QUOTA

This account has two entries. CLI is a command line interpreter that has several fixed commands, like TYPE and DIRECTORY, and user written command files. QUOTA is a RSX-11M disk quota system based on when the user log's in.

1980 - Fall - San Diego

[300,070] FCB
[300,070] LUT
[300,070] TTPOOL
[300,070] WHE

This account has four RSX-11M system programs that display various types of system information:

- o FCB outputs the current FCB chain for a disk device.
- o LUT displays the current LUN assignments for a running program, including any open files.
- o TTPOOL displays the number of free buffers for the RSX-11M V3.2 terminal driver.
- o WHE will follow and output the current trace-back chain for an active F4P program.

1981 - Spring - Miami

[300,100] FRC
[300,100] MVI
[300,100] OPRMON
[300,100] RECUR
[300,100] RMDEMO PATCHES
[300,100] SETPRV
[300,100] TTKGB
[300,100] UFD PATCHES

This account has a variety of entries, all related to RSX-11M V3.2 system management. Included in the account are the following programs and patches:

- o FRC is a task which forces MCR commands on another terminal.
- o MVI is a utility which displays who has mounted what volumes.
- o OPRMON is a task which works with the KMS Fusion enhancements to allow certain privilege functions to non-privilege users. The account is logged in as slaved and OPRMON requested. It then fires off an indirect command file, where you implement whatever privilege functions you wish people to have. When completed, OPRMON logs the terminal off.
- o RECUR is a utility to execute MCR commands on a periodic basis. RECUR reads a command file from which it gets day-of-the-week and time-of-day scheduling in addition to the command. When the command becomes due, it is sent to MCR for execution.
- o RMDEMO PATCHES add idle-time numbers and the fix for long outputs.
- o SETPRV sets a terminal privilege if the user knows the magic word.
- o TTKGB is a task which watches for idle terminals and forces BYES on them if there is no activity after some period of time.
- o UFD PATCHES are corrections to UFD to allow a non-privilege users create accounts in their current group.

1981 - Spring - Miami

[300,101] RUNOFF

RUNOFF is a text preparation tool that takes free-format text and commands and produces a formatted, paginated, jusitified output file. RUNOFF is especially useful for generating documentation. This is a bug-fixed and enhanced version of RUNOFF. Of special note is the table-of-contents feature and the rather complete manual.

1981 - Spring - Miami

[300,102] TECO PATCHES

This account has various patches to Teco V36. The patches include the following features and fixes (note, there are other changes I did not understand, not being a TECO wizard):

o On an EI command, scan LB:[2,2] if the file is not found in the current UIC

o When creating a file with an explicit version number, reuse the version.

o Speed up normal edits by creating the initial output file to be the same size as the input file.

o If invoked as MUNG, lower priority after 30 seconds to prevent long macros from eating the CPU.

o Add new features to better support split screen mode on different terminals, like the Hazeltine 1500.

1981 - Spring - Miami

[300,103] BYE PATCHES
[300,103] HELP PATCHES
[300,103] MCR PATCHES

This account has four patches to MCR or MCR tasks:

o ATLOV is patched so the ACT command outputs the state of the task in addition to the name.

o BYE is patched so it spawns @LB:[1,5]SETTERM.CMD command file, which resets the terminal characteristics.

o HELLO is patched to output the key-words at a particular level if '?' is used.

o MCRDIS is patched to allow a scheduled task (one with a TI: of CO:) to spawn tasks via MCR.

1981 - Spring - Miami

[300,104] BROOM

BROOM is a utility to zero the free space on a disk. It allocates all free blocks and then writes zeros to them. This helps prevent the curious from allocating a file and the looking at it to see what they got.

1981 - Spring - Miami

[300,105] CKP
[300,105] CVL

This account has two RSX-11M V3.2 system utilities. CKP is a program that checkpoints all stopped tasks. It is normally used by periodically scheduling it, say once every 5 minutes. CVL is a program to display or change disk volume parameters. It will display all parameters and allow privilege users to change the volume label, LRU number, extension size, maximum files, window size, or default protection.

1981 - Spring - Miami

[300,107] TYPE

TYPE is a utility to type files at a terminal. It has special commands for that purpose, such as removing comments, expanding control characters (useful for Teco macros), and truncating lines greater than the terminal width. TYPE also has wild-character file lookup ability similar to SRD.

1981 - Spring - Miami

[300,110] TRUNC

TRUNC is a utility to delete the unused blocks allocated to a file. This version has advantages over PIP /TR as it does not modify the creation date.

1981 - Spring - Miami

[300,120] CATEST
[300,120] DNDISPLAY
[300,120] LOGLST
[300,120] NSPOOL
[300,120] SORT
[300,120] TSKBUILD
[300,120] WHEN
[300,120] VTEDITMIN
[300,120] VSDRV
[300,120] XPR

This account has a variety of submissions for RSX-11M V3.2. Included in the account are the following programs and command files:

o CATEST is a test task for device drivers which are compatible with the CAMDRV found in [300,043].

o DNDISPLAY is a program that helps interpret the registers of a Xylogics disk controler.

o LOGLST is a program used in conjunction with the KMS Fusion accounting package to output the log sorted by user name instead of UIC.

o NSPOOL is a task that allows non-privilege users to start and stop the RSX-11M V3.2 print spooler. It works by sends from other tasks. A command task (UNSPOOL) and subroutine module are included.

o SORT is a utility that sorts fixed-length records based on up to nine ASCII keys using a quicksort.

o TSKBUILD is a command file to generate task-build command files.

o WHEN is a system task that tasks a task name and MCR command. When the specified task finishes execution, WHEN passes the command to MCR. This is sort of a poor man's type-ahead.

o VTEDITMIN is a version of VTEDIT.TEC for MIME-I terminals. The MIME-I has no alternate keypad, so the macro gets the key commands from easy to remember escape and control sequences.

o VSDRV is a device driver that implements variable send and receives. The driver creates named queues which can be referenced by read and write I/O request. All storage is taken from an internal pool.

o XPR is a utility to output text files to a Versatec 1200A printer. It generates the characters using the plotting capabilities so output better fits the reverse orientation of the Versatec paper.

1981 - Spring - Miami

[300,200] INFORM

INFORM is a IAS system program that can display many different system structures including the active task list, node usage, clock queue, PUD's, partition map, device forms type, a task's ATL, STD, or pending I/O, and terminal characteristics.

1979 - Spring - New Orleans
1980 - Spring - Chicago

[300,201] UTX

UTX is a IAS system program that will display the active task list on a CRT. It will show tasks shifting from one level to another, along with the TI: for each task and the round-robin pointer, next task to swap, and next task to load names.

1979 - Spring - New Orleans
1980 - Spring - Chicago

[300,202] EM04
[300,202] EM10
[300,202] EXECSUBS

This account holds documentation on the internals of RSX-11D.

1979 - Spring - New Orleans

[300,203] DAMMIT
[300,203] MC
[300,203] XX
[300,203] XXLIST

This account has four entries for RSX-11D/IAS systems. The entires are as follows:

o DAMMIT is a frustration release. It will output some wise remark when run.

o MC is a device handler to insert entires into the MCR queue. This allows tasks to issue MCR command lines using the QIO mechanism.

o XX is a virgin handler that contains as much of the required code as possible. It can form the basis for any device handler.

- o XXLIST is a handler written from XX. It lists the contents of any QIO passed to it and returns a success status.

1979 - Spring - New Orleans
1980 - Spring - Chicago

[300,210] DIABLO
[300,210] RNP
[300,210] RUNOFF
[300,210] TWOPAGE

This account has a enhanced version of RUNOFF and various utilities designed for use with RUNOFF:

- o DIABLO is a program to output files to a Diablo printer using write-pass-all's so the terminal driver does not ruin the control codes.
- o RNP is a preprocessor to RUNOFF that allows files to be included in the output file.
- o The version of RUNOFF includes support for Diablo printers such as two-color ribbons, proportional spacing, and control-codes. It also outputs in record mode so the output can directly be used by such programs as EDI.
- o TWOPAGE is a program that separates output into 'left' and 'right' pages so front and back files can be printed.

1981 - Spring - Miami

[300,300] MICHEAL REESE SUBROUTINES

This account has various Fortran-callable subroutines. Included in the account are the following routines:

- o FLOPEN is a routine to solicit a filename and prepare for open.
- o STRCMP is a routine to compare strings for equality.
- o MOCNVT is a routine to convert the month number to ASCII (and reverse).
- o STRCON is a routine to concatenate two strings.
- o MAXINT is a routine to find maximum of an integer array.

- o AMAXFA is a routine to find maximum of a real array.
- o CHRCNT is a routine to count characters in a string.
- o FLINTP is a routine to perform a floating linear interpolation.
- o LNFILL is a routine to fill a string with a character.
- o TRNCAT is a routine to find the last non-blank character in a string.
- o LUNDLT is a routine to delete an open file.
- o ATTDET is a routine to attach and detach a device.

1979 - Spring - New Orleans

[300,375] ADVENTURE

This is a version of the game Adventure. The messages are converted to upper/lower case for better readability and a save/restore feature added so a game can be interrupted.

1981 - Spring - Miami

[300,377] MISCELLANEOUS

This account has junk in it.

1978 - Spring - Chicago

[301,001] SCLUG SUBMISSIONS

This account has general information about the [301,*] accounts. These accounts are the Southern California LUG (SCLUG) submissions. The account usually has a description of the submissions on the tapes.

1977 - Fall - San Diego
1978 - Spring - Chicago
1978 - Fall - San Francisco
1979 - Spring - New Orleans
1979 - Fall - San Diego
1980 - Spring - Chicago
1980 - Fall - San Diego
1981 - Spring - Miami

[301,002] SCLUG SUBMISSIONS

This account has general information about the SCLUG submissions found on the tapes. It also has documentation on how to report errors.

1977	Fall	San Diego
1978	Spring	Chicago
1978	Fall	San Francisco
1979	Spring	New Orleans
1979	Fall	San Diego
1980	Spring	Chicago
1980	Fall	San Diego
1981	Spring	Miami

[301,010] MTBLOK

MTBLOK is a set of Fortran callable routines to perform block I/O to magtapes.

1977	Fall	San Diego
1978	Spring	Chicago

[301,011] SELECT

SELECT is a Fortran callable routine to provide keyboard option selection from a specified list of options. The package supports defaults, prompting, recognition and string completion via the ESCAPE key and non-unique prompting for options.

1977	Fall	San Diego
1978	Spring	Chicago
1978	Fall	San Francisco
1979	Spring	New Orleans
1979	Fall	San Diego

[301,012] WHO

WHO is a RSX-11M program to allow users to see who is logged in and what task is running from each terminal. It shows a limited set of task states.

1977	Fall	San Diego
1978	Spring	Chicago
1978	Fall	San Francisco
1979	Spring	New Orleans

[301,013] ENABLE

ENABLE is a privilege task for RSX-11M systems that allows a non-privilege terminal to be set to privileged if the user knows a password.

1977	Fall	San Diego
1978	Spring	Chicago

[301,014] TERMLOG

TERMLOG is a set of MCR routines which replace the RSX-11D HELLO, BYE, and SET /UIC to account for terminal login times.

1977	Fall	San Diego
1978	Spring	Chicago

[301,015] PASREF

PASREF is a Pascal cross reference program. It is written in BSM Pascal and cross references all non-reserved indentifiers.

1977	Fall	San Diego
1978	Spring	Chicago

[301,016] SSP

This account has a modified version on the IBM/360 Scientific Subroutine Package written for Fortran IV Plus. Note, the package is totally without comments. See the IBM System/360 Scientific Subroutine Package Programmer's Manual (GH20-0205-A).

1977	Fall	San Diego
1978	Spring	Chicago

[301,017] MTM

MTM writes the volume name of all mounted magtapes to the user's terminal.

1978	Spring	Chicago
1978	Spring	Chicago

[301,020] WHO

WHO is a RSX-11D program to allow users to see who is logged in and what task is running from each terminal.

1978 - Spring - Chicago

[301,021] PASWD

PASWD is a RSX-11D program to list UIC's and their associated passwords.

1978 - Spring - Chicago

[301,022] UTP

UTP is a RSX-11M or IAS program to read UNIX format ASCII tapes.

1978 - Spring - Chicago

[301,023] VTDRV

VTDRV is a RSX-11M driver which provides a privileged logged-in terminal. The device allows such things as scheduled tasks or DECNET programs to run from and not tie up a physical terminal.

1978 - Spring - Chicago

[301,024] TYPE

TYPE is a program to perform the equivalent of a PIP TI:=file command. It is optimized for terminal I/O and has several features to support typing files to a terminal.

1978 - Fall - San Francisco

[301,025] TERCOM

This account holds a package to maintain information about each logged in terminal. In particular, the package maintains the login account name, UIC, device, and time.

1979 - Spring - New Orleans

[301,027] MATLIB

MATLIB is a set of Fortran callable subroutines to do efficient matrix and vector manipulation such as dot and cross products or matrix mutiply. Both single and double precsion reals are supported. The following routines are included in the submission. The naming convention uses a suffix of 'D' for the double-precision version of the routine. Matrix routines start with 'M' and vector routines start with 'V'. The prefix 'M3' and 'V3' is for 3x3 matrices and 3-vectors. The prefix 'MP' is for submatrices.

- o CROSS (CROSSD) gets the cross product of two 3-vectors.
- o DOT (DOTD) gets the dot product of two vectors.
- o INDENT (IDENTD) sets a array to the indentity matrix.
- o MABT (M3ABT, MPABT, MABTD, M3ABTD, MPABTD) does a matrix transpose multiply for a right hand matrix.
- o MADD (M3ADD, MPADD, MADDD, M3ADDD, MPADDD) does a matrix addition.
- o MATB (M3ATB, MPATB, MPTBD, M3ATBD, MPATBD) does a matrix transpose multiply for a left hand matrix.
- o MCLR (M3CLR, MPCLR, MCLRD, M3CLRD, MPCLRD) clears a matrix.
- o MCOM (M3COM, MPCOM, MCOMD, M3COMD, MPCOMD) does a linear combination of two matrices.
- o MMOV (M3MOV, MPMOV, MMOVD, M3MOVD, MPMOVD) moves one matrix to another.
- o MMUL (M3MUL, MPMUL, MMULD, M3MULD, MPMUD) multiplies two matrices.
- o MSCL (M3SCL, MPSCL, MSCLD, M3SCLD, MPSCLD) does a scalar multiplication of a matrix.
- o MSUB (M3SUB, MPSUB, MSUBD, M3SUBD, MPSUBD) subtracts one matrix from another.
- o MTRN (M3TRN, MPTRN, MTRND, M3TRND, MPTRND) transposes a matrix.
- o PROD (PRODD) gets the component product of two vectors.
- o VADD (V3ADD, VADDD, V3ADDD) performs a vector addition.
- o VCLR (V3CLR, VCLRD, V3CLRD) clears a vector.
- o VCOM (V3COM, VCOMD, V3COMD) performs a linear combination of two vectors.

- o VMAG (V3MAG, VMAGD, V3MAGD) gets the magnitude of a vector.
- o VMOV (V3MOV, VMOVD, V3MOVD) moves one vector to another.
- o VSCL (V3SCL, VSCLD, V3SCLD) multiplies a vector times a scalar.
- o VSUB (V3SUB, VSUBD, V3SUBD) subtracts one vector from another.

1979 - Fall - San Diego

[301,030] XMITR

XMITR is a task to allow your computer to emultate a terminal on some remote computer system. It also allows file transfers to and from the remote computer to your system.

1979 - Fall - San Diego

[301,031] POOLFL

POOFL is a RSX-11M system program to take a snapshot of pool and output a visual map of the data structures.

1979 - Fall - San Diego
1980 - Spring - Chicago
1980 - Fall - San Diego

[301,032] TSPAWN

TSPAWN is a Fortran callable routine to allow IAS time-sharing tasks to spawn other programs using the timesharing control services (TCS) rather than the real-time directives.

1980 - Spring - Chicago

[301,033] CPA

CPA is a companion to POOFL in [301,21]. It outputs a visual dump of pool taken from a crash system dump.

1980 - Spring - Chicago
1980 - Fall - San Diego