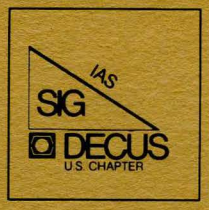

# The DeVIAS Letter





# The RSX Multi-Tasker

## JANUARY 1984 ISSUE

*Printed in the U.S.A.*


# Section One


# The RSX Multi-Tasker

# Table of Contents

## Columns



## Articles

# From The Editors

This is our first issue as co-editors of the Multitasker. From the response seen in Las Vegas, people have been really anxious to see the Multitasker re-appear. I know I was; that's how I wound up with this job. I figured if I wanted to see a Multitasker on my desk every month that one way to do it was to do it myself. Dominic DiNollo, also of the New York LUG, has agreed to co-edit with me. There will usually be some material from both of us in each issue.

## 1.0 THE FUTURE OF RSX

In the Tenth Anniversary Dinner in Las Vegas one of the speakers made the point that it was the Multitasker and its monthly appearance that really got the SIG off the ground and made it (at one time) the largest SIG in DECUS. I feel that the SIG is in a critical time right now, and the newsletter is more crucial than ever to keeping it alive and giving us all a sense of direction.

There seemed to be two conflicting attitudes among SIG members in Las Vegas: one was a nostalgic feeling of winding down to a slow extinction; the other was expectancy of massive growth due to an influx of PRO/300 and Micro-11 users. (Someone even suggested that we should have SIG floppies as well as SIG tapes...clearly impossible with the volume of software we usually distribute.)

I believe the truth lies somewhere in between. RSX, and the RSX SIG, is by no means near death! There certainly will be new people, users of P/OS and Micro-RSX (to be released in 1984), who will want to dig into things and use the vast amount of software and personal expertise that has been created by members of our SIG. M-Plus systems will also continue to be sold. The RSX group in Digital now has responsibility for the VAX/VMS AME (compatibility mode), and the odds are good that compatibility will mean more in the future than it has in the past for VAX users.

No operating system lives forever, and inevitably RSX will someday die, or at least, like the old soldier it is, just fade away. But that day is long off. RSX is the bridge system between the Professional 300s, the PDP-11s, and the VAXs. And it lives in a new incarnation in the Micro-11. There is going to be an active, demanding interest in RSX subjects for many years to come.

For as long as I can, I intend to see to it that the Multitasker is there, month after month, to serve that interest.

## 2.0 SUBMITTING MULTITASKER ARTICLES

If you share my belief in RSX, then show it by sending articles, letters, questions, and discoveries to the Multitasker. Let's make it a rewarding experience to read every issue.

### Acceptable formats for articles

We are quite flexible in the formats in which articles can be submitted. All articles should be sent in some machine readable format, accompanied by hard copy if possible. The following media are acceptable:

Magnetic tape:
800, 1600 or 6250 BPI
FLX, BRU, PIP, or VMS BACKUP

Floppy Disk:
RX01 or RX02
ODS-1 or ODS-2

In addition, for last minute submissions arrangements can be made to dial in to a special account on Allen Watson's VAX. Please phone him directly for details.

## 3.0 FORMAT OF ARTICLES

The contents of the media should be in RUNOFF format, preferably. If you don't know RUNOFF then send straight unformatted text. The RUNOFF we use is VAX Digital Standard Runoff; John Clement's version on the RSX tapes has done a good job of maintaining compatibility with DSR. We use the following formats for most articles:

```
.RIGHT MARGIN 72 .LEFT MARGIN 5 .AUTOPARAGRAPH
```

This allows you 67 characters per line. Many program examples and command files will have to be edited if they exceed 67 characters per line; it would be very helpful if you tested things through RUNOFF first using these parameters and edited any long lines into what you would prefer they be. Otherwise we will have to do it, and you will have to take our decision on where to break lines.

## 4.0 DEADLINES

Deadlines for the Multitasker are as follows, and will be strictly observed:

| | |
|---|---|
| Submissions: | Second Friday of preceding month |
| Mail to DECUS: | Third Friday of preceding month |

This means, for example, that the deadlines for the March issue are, for articles to reach us, February 10; for our camera ready copy to be mailed to DECUS, February 17. Sometimes (but not usually) material received between the second and third Fridays will be able to be included in the issue. Most often we will put the issue together the weekend after the second Friday and mail it to DECUS the following Monday or Tuesday.

Please notice that for material to reach us (the editors) by the second Friday of a month, you will have to mail it to DECUS about two weeks earlier. They will then re-mail the material to us. Although this seems inconvenient it keeps a single mailing address for Multitasker material even if editorship changes.

If you have some time-dependent material and are afraid it will not reach us in time through normal channels, phone one of us directly and we will help you get it to us.

## 5.0 PLANNED COLUMNS

In the past there have been a number of columns in the Multitasker. Since the newsletter has been absent for many months recently we felt it appropriate to list the columns we plan to publish. This may give our readers some ideas of the kind of material we are looking for. If you submit something with a particular column in mind, please indicate this when you send your submission.

All submissions should be sent to:

Multitasker Editor
c/o DECUS
One Iron Way
MR2-3/E55
Marlboro, MA 10752

### 1. Library News

If you have news about any of the DECUS library software or software from the RSX SIG tapes, send it to Multitasker Editor - Library News. Material desired would include: problems discovered or solved, patches to existing software, short notes

on library or tape submissions you have found useful, and so on. Even just a paragraph saying "I've used X and it's great" will be appreciated because it gives other users ideas of what's good and worth trying.

2. Working Group News

The Working Group Coordinator will be submitting material regularly to this column, and working group leaders should submit their material through him normally, or at least keep him informed of anything you send directly to us. If a reader has a question they would like to direct to a working group and you don't know who to contact please feel free to write the Multitasker.

3. Help Yourself

Help Yourself is the Question/Answer column. If you have a tough question or one that is tough for you, and you want to see if anyone else has encountered the problem and found a solution, write. We will publish all questions. If we know the answer or know someone who does, we will print the answer along with the question. If we don't have the answer, the question is up for grabs to the general readership; please send a letter if you see a question and know the answer.

4. It's In the Code

This column documents undocumented features of Digital's software, or clarifies things that are not obvious from the published documentation. In the past it has included articles on undocumented switches in the F4P compiler, use of the Gold/arrow keys in EDT V2.0, and how to set up SYSLOGIN and SYSLOGOUT command files. Send us your discoveries.

5. Hints and Things

This is a monthly potpouri of helpful tidbits and rumors. Any input about ways to make life easier on RSX is needed. Patches, workarounds, or user-written utility programs in any language are valid candidates. Longer submissions to this column may appear as articles in their own right.

6. Speak Out

Speak Out is a column for readers to express their opinions or to comment on a previous column or article. The articles published in this column are an individual's point of view and do not necessarily represent the opinion of DECUS or the RSX SIG. Readers are welcome to submit articles on any subject of interest to the RSX world.

7. Articles

Finally there are articles. These are generally longer submissions that expound on a single topic. We will be including material from the Symposia sessions and handouts, but we strongly encourage readers to send their own material. Articles about your experiences with installing or using some piece of software or hardware are always interesting and useful. And please note: nothing is "too elementary". If something was news to you there are probably hundreds of people out there who need to know it and don't. We still have lots of new RSX users who don't know the good old stuff.

Allen A. Watson
(201) 646-4111

Dominic DiNollo
(914) 968-2500 Ext. 2207
Multitasker Editors

# DECUS/RSX SIG Library News

Glenn C. Everhart
RCA GSD Engineering
M/S 206-1
Rt. 38
Cherry Hill, NJ. 08358
12/2/83

RSX FALL 1983 SIG TAPE

The Fall '83 Sig tape is being handled a bit differently this year from previous years. In the past, one person has always taken the preliminary master tape from the Symposium's all-nighter session, added all the necessary indexing and completed extracting any programs that couldn't be handled well that first night, and broken the tape up into pieces small enough to be distributed. He has then had to construct a tree, verify that all the people on the tree are valid recipients and able to make copies of the tape, and send cards and trees to all the nodes. This has always been a massive job, involving a lot of telephoning and time, and has slowed down getting the tape out. This year, we are having the job broken up into 4 parts. The tape is built by the Tape Copy Coordinator as before, but there are 3 Regional Tree Coordinators, corresponding to the 3 main branches of the tape copy tree we have always had, who have the job of constructing the parts of the tree in their regions. Their inputs are the previous trees and the

participation requests received for this symposium.

The coordinators are:

East: Kitty (Catherine) Bethe
Bankers' Trust Co.
130 Liberty St.
37th Floor
New York, NY 10006 212-775-4190

Midwest: Paul M. Tompkins
Dow Chemical USA
Freeport, Texas 77541 409-238-7040

West: Jim Neeland
Hughes Research Labs
3011 Malibu Canyon Rd
Malibu, California 90265 213-456-6411 x333

Where a LUG has been on the tree in the past, and where it has been a reliable node (as indicated by responses acknowledging they got the tape and lack of complaints from successor nodes), it needs to be contacted if it didn't fill in the request to participate. The tree has always been highly variable however, and cooperation from the SIG membership can be very helpful. Making these contacts is a time-eater and life is easier if, in cases where no change is made, the LUG will get the request in on time and give the correct person's name and address. A note to the tape copy coordinator or the regional tree coordinator about changes will help speed the process too, especially where there have been changes of LUG RSX librarian.

Let me go over how a LUG gets onto or off of the tree. A new LUG must make a tape copy participation request form up and say what SIG tapes are in its area of interest. The form is in a publication sent to LUG chairmen by DECUS. Once this happens, if a LUG requests to be on the RSX tree, DECUS forwards a copy of the form to the RSX SIG librarian (=tape copy coordinator) and it is then forwarded to regional tree coordinators (new this year). Without this form, the Sig doesn't know the LUG exists for this purpose. (Once a LUG is on the tree, it can be called if it hasn't filled in the form, but if its chairman or contact person changes and the phone contact fails, the LUG may be dropped from the tree. MORAL: Always fill in the form.)

The participation request has the contact address and some information about the OS and hardware at the site given as the contact. A serious problem is that many LUG chairmen give their own sites as contacts for RSX or VMS tapes where they lack the ability at those sites to make copies. Even if there are other

sites in the LUG that can do the copy work, this transfer usually introduces a couple of weeks (frequently more) delay at that stage of the tree. Therefore, unless the LUGs give an alternate contact who CAN do the work, it isn't fair to anyone downline of such nodes to have their tape blocked. It is ESSENTIAL to participation in the tree that the tape go to sites who can and will make the copies in a reasonable time. The tree nodes should also be readily available in some fashion by phone/mail/preferably both so that they can be contacted by the nodes who must get their tapes from them. Some nodes have been dropped because the contact was impossible to contact and lower nodes had to go around them. Finally, the request form asks that the node make a commitment to copy tapes for non-LUG members in the area if requested. A lot of the country isn't covered by really close-by LUGs, and we feel that tree nodes should be willing to make some copies for non LUG members as a background priority.

The above covers how a LUG gets onto the tree. Once on, it's easiest to leave a LUG on the tree. However, they get dropped sometimes because they become unreachable, because lower nodes can't get them to pass on the tape, or for other similar reasons. A frequent problem is that some nodes forget that the LOWER NODES CONTACT THE HIGHER ONES to send blank tapes for copy. They sometimes forget to call their parent nodes, and accordingly never get the tape.

NEWS ABOUT THE TAPE

The Fall '83 Sig tape has about 60,000 blocks. A few of the programs on it will be:

* Another release of the LBL Tools (UNIX - like toolkit)

* A bigger and better Rice University RUNOFF (DSR compatible superset, VERY handy for word processing) * KERMIT (though it appears the RSX Kermit may not make it this time)

* A speeded-up and more powerful PortaCalc for RSX and for VMS. A PDP11 version with 16,000 cells is now available for vanilla RSX11M with a little work on your part to bring it up (no .cmd file is there but all sources and an ODL file exist).

* A memory virtual disk for RSX11M/M+

* The latest and GREATEST version of SRD from Bob Turkleson and the SRD Working Group (merges all known enhancements into one really super program)

* A Supermac in Fortran

* An ADVENTURE compiler (!)

* Mitch Wyle's latest Desktop Calendar package

* KMSKIT (Plus CCL V9, a real winner)

The RSX83B tape itself is ready to distribute and is currently awaiting the Regional Tree Coordinators finishing up the tree. Their information was completed and sent to them a few days ago to allow them to complete the job. (I must apologize for that delay, though the regional coordinators had started hunting up people before they got the final packets I sent.) The tape will be in BRU format (again) at 1600 BPI. The BRU of RSX11M V3.2 will be used, since as far as we can find out, all later versions of BRU can read that one.

The tape will be submitted, once out, to the DECUS library also, giving people not on the tree an alternate way of obtaining the software. It has not yet been submitted to DECUS, however, and DECUS does not control when it is submitted. Therefore, please just watch DECUSCOPE for the announcement of its availability and don't bug them for it. The tree will be published to all nodes on it once it is done.

Editor's Note

If you want an old tape, rather than bothering your local LUG tape person, why not purchase it from the DECUS library? The fee is nominal and you don't have to scrounge up a blank tape and fuss with mailing it. See your library catalog or call DECUS for information. I recently had a call asking where to obtain TECO V36. It was last on the SIG tapes in 1980, but a separate distribution tape is available from the DECUS library, item #11-333 for $42, according to the 1983/1984 catalog. A separate printed manual, #11-450, is available for $15. (Prices for non-DECUS members slightly higher.)

# IT'S IN THE CODE

Allen Watson
Multitasker Editor

Recently we found a problem using RMSIFL supplied with M-Plus V2.1. Presumably this problem also exists for RMS with RSX11M V4.1. Certain sequential files would not convert properly to indexed files. The solution suggested by TSC was sorting the sequential file and using RMSCNV to convert it to indexed format. However, this also failed. In the case of RMSCNV the error reported was a duplicate primary key; when CMV encountered this record, it exited. The old version of RMSCNV used to continue running and report at the end that an undefined number of records were skipped due to duplicate primary keys.

We knew there were duplicates. Ignoring duplicates was fine; we just wanted the program to continue building the indexed file for us. Looking in the RMS Utilities manual under the error message we found a reference to a switch, "/ER". It appeared that this switch might do what we wanted, but it was not documented elsewhere in the manual.

We called TSC. They verified that such a switch exists and is undocumented. It does cause RMSCNV to ignore the error condition and continue to run. We have used it and it works. The reason it is undocumented is that it has a minor bug; at the end of the run it reports how many records were ignored due to errors, and when it does, it erroneously reports that all the records were skipped. That is, if you are converting a file with 500 records and use the /ER switch, it will say that 500 records were skipped. In fact, only the records in error were skipped. TSC says that a fixed version is currently being tested and the switch will be documented with the next release.

# Speak Out

Comments on St. Louis Q and A

Kitty Bethe
Bankers Trust, New York

I was disturbed when I read the reports on the St. Louis Q and A sessions in the November Multitasker because I felt that some of the answers given were inadequate or misleading. In particular I would like to comment on three questions found in that issue.

## 1.0 POST MORTEM DUMP FILES DELETED WHEN PRINTED

The user wanted to print several copies of his .PMD file and found that it was deleted by PRT... when he printed the first copy. The answer was "Try to modify PRT... task, or use queue manager." This is a very poor answer.

In fact, the task that needs to be modified is LPP, and modifying it is simple. Provision is made in the task build command file for LPP, called LPPBLD.CMD or LPPFSLBLD.CMD, located (on my M-Plus system) in [1,24]. The comments in this file explain quite clearly how to select by extension the files that will be automatically deleted by the spooler. By editing this file and task building LPP, any user can tailor this spooler feature to his own preference. Of course, if LPP is installed by VMR on your system you will have to re-VMR and SAVE the system image to install the new version. You can test the new version, however, by putting commands into your STARTUP.CMD file to remove the old version and install the new; these commands should precede any commands that start up the print queues.

## 2.0 CONTROL/S STATE ON TERMINAL PRINTER

The question was, "How can I tell if a terminal is in a control-S state when it is spooled as a printer?" The answer was, "No guaranteed way to tell." Hogwash.

First, unspool the printer. Second, find a program called TCR on one of the old SIG tapes (Fall, 1980, [375,2]...ed). This program shows terminal characteristics including whether or not the terminal is in a control-S state.

To get the terminal out of control-S, if you have no keyboard on the printer, you may need to plug a spare VT100 into the line and control-Q it.

### Editor's Note

See the article on "XONing an XOFFed Terminal" in this issue.

## 3.0 CLOBBERING FIRST TWO BLOCKS ON DISK

There was a question about how the first two blocks on the system disk might get clobbered when no privileged tasks were running. Amazingly no one gave the obvious answer to this old, known problem: a breakthrough write to an unassigned LUN is equivalent to a write to block zero of the system disk. A BROADCAST to a spooled terminal device used to do this, but I believe that has been fixed some time ago. Any program that does a breakthough write, however, can cause the problem if the programmer forgets to assign the LUN and the program is run in that state. Goodbye system disk.

## 4.0 OTHER TASK BUILD OPTIONS

The question about the spooler, above, prompts me to add some suggestions to people doing SYSGENs for M or M-Plus for the first time; when you do a PREPGEN, even if you do not plan to rebuild the non-privileged tasks because you don't know of any reason to, run through that part during the PREPGEN. In this way, SYSGEN will create the task build command files for both privileged and non-privileged utilities in [1,24].

Before doing the actual SYSGEN, print out all the command files in [1,24] and look through the task build options for the utilities. You will find a lot of options that are not documented anywhere except in these command files. I think you will find that many of them are options you will want. For example, besides the automatic deletion in the printer spooler discussed above, you will see options to control whether or not MACRO produces list files by default, and whether they are spooled; controlling whether or not PIP preserves creation date on a file copy; setting up a default directory for indirect command file execution; selecting whether or not DCL falls through to MCR when it gets an unrecognized command; and options to control the default switch settings for TKB. There are many more.

If you have plans to change any of the options for TKB I offer you this one suggestion: install your version of TKB under a different name and leave the old TKB undisturbed. DEC assumes these defaults during SYSGEN and does not explicitly include the switches in their task build files. If you should ever do a SYSGEN using your modified TKB the odds are the resulting system will not run. (Example: the default is that tasks are non-checkpointable in the DEC supplied TKB. If you change that and do a SYSGEN, tasks that should not be checkpointed will be built checkpointable, and your system will crash when they do.) So if you modify TKB, install it under a special name and use only the DEC-supplied TKB to do SYSGEN.

# RSX-11M-Plus System Accounting and Datatrieve

RSX-11M-Plus System Accounting and Datatrieve
B. Z. Lederman I.T.T. World Communications

Abstract

One of the features of RSX-11M-Plus is system accounting, which allows the accumulation of statistics about the system (such as the number of logins, disk I/O activity, print queue submissions, and so on). One of the methods supplied to extract this information is the conversion of the system accounting file to a form readable by Datatrieve, and a command file to produce reports with Datatrieve. This paper will show how these commands have been changed to suit our installation, with an explanation of why each change was made, and the benefits derived from the change. Some options which may be of value to other installations will also be shown. The intent is to show how easily the reports may be tailored to individual needs by using Datatrieve as the report generator.

After we had installed M-Plus and were satisfied that it was running correctly, I became interested in the system accounting feature, which was one of the features which led us to convert from RSX-11M. System accounting is started and stopped with privileged commands, and in our system it is always started by the [1,2]STARTUP.CMD file, so that our system is always accumulating statistics. These statistics become useful only when they are output in human-readable form, and the first step in this process is to convert the data from the file in which it is stored by the system (normally [1,6]ACNTRN.SYS) with the command:

```
SHOW ACCOUNTING/DATATRIEVE ACCOUNT.DAT
```

This places the data in a file (in this example named ACCOUNT.DAT) where it may be read by Datatrieve (or by other programs). The next step can be to use the command file supplied by DEC, [126,24]ACNTRN.CMD, which will produce a report for each type of transaction recorded by system accounting. It is invoked by the command:

```
DTR @[126,24]ACNTRN.CMD
```

and it will prompt the user for the name of each report.

I immediately found a number of features in this command file which I did not like. First, it creates the necessary record definitions, procedures, etc. in order to create the reports, but then immediately after using them once, it deletes them. This is inefficient, because it requires both time and computer resources (especially disk I/O) to create and delete all of these structures, as they will be needed the next time a report is desired. It did not make sense to me as I planned to obtain reports at regular intervals. There is also the problem that whenever something is defined in Datatrieve, it takes up space in the dictionary (a file usually named QUERY.DIC), and when it is deleted, the space is not re-used, so that each time the command file is invoked to obtain reports, the dictionary file will get larger and larger, with an increasing amount of space allocated on disk but not re-useable until the dictionary compression utility is used. Therefore, the first change I made was to edit the file and remove all lines beginning with the word "DELETE", so the file would create the definitions and not delete them. I also removed all lines beginning with a colon such as ":SS-REPORT", as these are commands to invoke the procedures. I thus have a file which will set up all of the data structures once, and now I can invoke the individual procedures whenever I want one particular report. After invoking the modified command file, the dictionary contents are as follows:

```
DTR> show all
Domains:
   RESOURCE-INFO    SYSTEM-INFO      USER-INFO
Records:
   RESOURCE-INFO-REC                 SYSTEM-INFO-REC USER-INFO-REC
Procedures:
   ALL-REPORT       CRH-REPORT       DEA-REPORT       DMO-REPORT
   DST-REPORT       INP-REPORT       INV-REPORT       LOG-REPORT
   MOU-REPORT       PRT-REPORT       RTP-REPORT       SAB1-REPORT
   SAB2-REPORT      SS-REPORT        TAB1-REPORT      TAB2-REPORT
   TIM-REPORT       UAB1-REPORT      UAB2-REPORT
Tables:
```

Each of the procedures generates a single report on a particular resource, and each may be invoked when wanted. For example, the LOG-REPORT looks like this:

```
                L O G I N    T R A N S A C T I O N
29-Sep-83                                                        Page 1
      NAME        UID     DEV     ACNT      UIC          LOGON       TIME
B LEDERMAN     PNA4     TT5     0003    [300,003]     6-Sep-83   13:47:13
B LEDERMAN     PNA51    TT7     0003    [300,003]    13-Sep-83   8:19:1
B LEDERMAN     PNA53    TT7     0003    [300,003]    13-Sep-83   8:20:29
B LEDERMAN     PNA55    TT7     0003    [300,003]    13-Sep-83   8:21:49
B LEDERMAN     PNA57    TT7     0003    [300,003]    13-Sep-83   8:23:56
B LEDERMAN     PNA63    TT7     0003    [300,003]    13-Sep-83   8:30:39
B SYSTEM       SYS1     TT17    9901    [001,001]    29-Aug-83   8:29:7
B SYSTEM       SYS2     TT16    9901    [001,001]    29-Aug-83   8:29:13
```

```
B SYSTEM      SYS3      TT3      9901  [001,001]   29-Aug-83  9:8:26
B SYSTEM      SYS15     TT4      9901  [001,001]    6-Sep-83  8:11:47
C SCOTT       PNA104    TT7      0011  [300,011]   13-Sep-83  10:19:5
C SCOTT       PNA105    TT7      0011  [300,011]   13-Sep-83  10:19:48
C SCOTT       PNA112    TT16     0011  [300,011]   13-Sep-83  10:35:23
```

Editor's Note

It was necessary to remove some spacing between columns in the preceding report and others in this article to make it fit in our margins.

I then started looking at the way the reports were generated, and I found an area where an immediate improvement could be made. Each report was generated with the command sequence:

```
FIND xxx WITH CODE=nn
REPORT CURRENT ON *."DEVICE OR FILE"
```

The first line finds all of the records in a domain (a collection of data) with a transaction code of nn (each transaction type has it's own code), and places it by default in a collection named "CURRENT". The second line reports this collection on a device or places it in a file, depending upon the answer received from the user; when Datatrieve sees an asterisk, it means that the user is to be prompted for an answer. This certainly works, but it requires Datatrieve to go through the data once to find the records with the proper code, and a second time to do the report. It is more efficient to do both at once with a command such as:

```
REPORT xxx WITH CODE=nn ON *"Device or File"
```

and so I changed all of the procedures to do this.

The next step was to examine the record definitions. A record describes how each data field is arranged within each record of a data file, and is the basis for all data retrieval and processing. The record definitions supplied put related data records (such as all user information) in one record definition, making three large records in all; one each for system information, user information, and general system resources. My objection to this was that it makes each record definition very large, and, in my installation, I am interested in only one information type at a time. The problem with a large record definition is that it uses up pool (it is very important to remember that this refers to Datatrieve's internal working pool, NOT RSX system pool), and running out of pool in Datatrieve is rather like running out of system pool in RSX, only you lose just one job rather than the whole system. Conserving pool space is a goal in Datatrieve which is persued with just as much fervor (and with just as much reason) as conserving system

pool in RSX, so a record definition which is unnecessarily large is something to avoid. Another reason is that, if you forget the names of the individual fields (data items), and try to do a SHOW FIELDS command to see the names, the first fields will scroll off your CRT screen long before the last fields come up, which is annoying. I decided, therefore, to set up one record definition and one domain for each transaction type, so I could deal individually with each one as I needed it. One of these cut down records would look something like this:

```
01 ACCOUNTING-TRANSACTION-REC.
   10      CODE       PIC IS 99 USAGE IS COMP.
   10      LOG.
           15  LOGNAM  PIC IS  X(16).
           15  LOGUID  PIC IS  X(10).
           15  LOGDEV  PIC IS  X(6).
           15  LOGACC  PIC IS  9(4) USAGE IS COMP.
           15  LOGUIC  PIC IS  X(10).
           15  LOGDAT  USAGE IS DATE EDIT-STRING IS DD-MMM-YY.
           15  LOGHUR  PIC IS 9(2) USAGE IS COMP.
           15  LOGMIN  PIC IS 9(2) USAGE IS COMP.
           15  LOGSEC  PIC IS 9(2) USAGE IS COMP.
;
```

and so on until all fields are defined. All records start with CODE, which is the two digit code identifying the packet type: the "USAGE IS COMP" signifies that the field is a one word binary integer (like INTEGER*2 in Fortran). Similarly, "USAGE IS DATE" is an 8 byte date format increasingly used by DEC. The record types were logically organized by the developer at DEC by naming each field with 6 characters: the first three refer to the packet type (in this case the Login Transaction Block), and the last three to a field (DEV is Device, UIC is the familiar User Identification Code, and so on). This is a very logical method of organizing the data, and there is nothing "wrong" with it, but I chose to re-organize it in a different manner, for two reasons. First, if the report procedures are examined, it may be seen that all of the fields are referenced in a manner such as:

```
PRINT                    LOGNAM("NAME"),LOGUID("UID"),LOGDEV("DEV"),
LOGACC("ACNT"),LOGUIC("UIC"),
```

The reason is that the field name must be given to identify the data, and Datatrieve, by default, uses the field name to identify that column at the top of the page, so the additional reference such as '("NAME")' must be made to override the default and have the word NAME print at the top of the column rather than LOGNAM. There is nothing wrong with this, but it requires extra working space in the report writer. This could also be done by placing the additional qualifier "QUERY-HEADER IS NAME" in the record definition of the LOGNAM field, so that Datatrieve would use the word NAME to head a column whenever LOGNAM is referenced: this would reduce the pool space used in the report, but increase the

pool space used by the record definition. My other reason for the change is that there is a name field in most of the packets, and rather than having to know a different field name for each packet, I decided to name all of the fields in all of the records uniformly: thus the device field in all records is DEVICE, the UIC field in all records is UIC, and so on. That way, I don't have to know a lot of field names, and the correct header will print out whenever I do a query or report. I also squeezed down the definitions a little by removing the word "IS", which is optional, and by removing unnecessary edit strings. I then had a set of uniform record definitions, which look like this:

```
DEFINE RECORD BASE-LOGIN-REC USING
01 BASE-LOGIN-REC.
    10 CODE PIC 99 USAGE COMP.
    10 FI PIC XX .
    10 NAME PIC X(14).
    10 DEPT PIC XXX.
    10 NBR PIC X(7).
    10 DEVICE PIC X(6).
    10 ACC PIC 9(4) USAGE COMP.
    10 UIC PIC X(10).
    10 DATE USAGE DATE.
    10 TIM.
            20 FILLER PIC 99 USAGE COMP.
            20 FILLER PIC 99 USAGE COMP.
            20 FILLER PIC 99 USAGE COMP.
    10 TIMR REDEFINES TIM.
            20 LH PIC 99 USAGE COMP.
            20 LM PIC 99 USAGE COMP.
            20 LS PIC 99 USAGE COMP.
    10 TIME PIC 9(6) COMPUTED BY
            (LH * 10000) + (LM * 100) + LS EDIT-STRING 99,99,99.
    10 FILLER PIC X(60).
;
```

This definition has some features not in the DEC supplied records, and deserves some explanation. First, the UID field of the original definition has been replaced. This field contained data of the form "BZL0013", where BZL would be the three letter mnemonic defined in the system account file for that user's account, and the number is the login sequence for that particular login. As it happens, my company has been using three letter mnemonics to designate different departments for many years, and these were used to designate the department for each user who has an account on the system. I therefore decided to access the data in two fields, the first named "DEPT" to recall the first three letters, and "NBR", which is the login number. Similarly, I have set up the name field to separate the first inital (and blank space) from the last name, which is what I will be using in my reports. The time field require some explanation. The hour, minute, and second are recorded in separate binary integers. Since the usual way to print out a time in an RSX system is "hh:mm:ss", each DEC report

concatenated the fields by printing them in a manner such as this:

```
PRINT LOGHUR||":"||LOGMIN||":"||LOGSEC("TIME")USING X(8)
```

but there is a problem: when integer fields are concatenated, leading zeroes are suppressed, and I could find no way to stop this with edit strings, so that a time of 8 hours, 14 minutes, 3 seconds would print out as "8:14:3", and not "08:14:03" as would be expected. This looked messy to me when a column of times appeared on a report. The best I could do was to combine the fields using the COMPUTED BY feature. TIM is a group name, which allows a number of fields to be referenced together. The three FILLER pictures allocate the correct amount of space in the record definition for the time data, but FILLER never prints out, nor does it appear in a SHOW FIELDS command: it is a way to allocate space in a record without using it. The next entry, "TIMR REDEFINES TIM" means I want to give Datatrieve an alternative method of accessing the space reserved in the record definition, and now I put in the three real fields containing the hour, minute, and second. When Datatrieve prints out a record, and there is a REDEFINES, the first field in the list is taken by default: in this case, the FILLER is first, so it does not print out, and neither does the TIMR fields, but they are always there, and may be accessed explicitly by name whenever they are wanted. What will print out instead is the COMPUTED BY field: this type of field does not actually take any space in the record, but is computed by Datatrieve when the information is required. The computation is to move the hour left 4 digits by multiplying by 10000, and minutes left two places by multiplying by 100, adding the hour, minute, and seconds together, and printing it out with commas (as there is not provision to put colons in an edit string. The result for the time given above would be "08,14,03", and while commas are less familiar than colons, all time fields line up one over the other when printing on a report, and the time is printed this way every time, including queries as well as reports without having to put in the concatenation operation, so I decided that I would put up with commas to gain the other advantages. I repeated this procedure for all of the transaction types, creating a record definition named BASE-xx-REC and a domain named BASE-xx, where xx corresponded to some transaction type such as LOGIN, RESET, CRASH, etc. The one exception to this is that I combined the MOUNT, DISMOUNT, ALLOCATE, and DEALLOCATE transactions into one record called BASE-AM-REC and one domain called BASE-AM: this is possible as the transaction packets are the same for these four transactions, and I will be reporting them all together, as will be shown later.

I also created a new report procedure for each transaction type, similar to this one:

```
DEFINE PROCEDURE LOGIN-REPORT
READY BASE-LOGIN
REPORT BASE-LOGIN WITH CODE=19 SORTED BY DEPT, NAME, DATE, TIMR ON
LOGIN.RPT
```

```
SET COLUMNS-PAGE=80
SET REPORT-NAME="Login   Transaction"
AT TOP OF NAME PRINT FI|NAME
PRINT DEVICE, UIC, DATE, TIME USING 99,99,99
 AT BOTTOM OF NAME PRINT SKIP, COL 1, "Number of logins for",
  SPACE 1, FI|NAME, SPACE 1, COUNT, SKIP
 AT BOTTOM OF DEPT PRINT SKIP, COL 1, "Number of logins for",
  SPACE 1, DEPT, SPACE 1, COUNT, SKIP 2
END-REPORT
FINISH
END-PROCEDURE
```

Each of these reports is set to come out on a specific file, as I know I want my reports in files, and not on TI:. Note that it is possible to sort by TIMR, the re-defined time field, even though it does not normally print out. Sorting by the group name results in all fields in the group being sorted, which is what is desired. In my case, I want to sort by DEPARTMENT and NAME, but sorting by DEVICE or any other field, is also possible. A portion of the report generated by the above procedure looks like this:

```
                    Login   Transaction
13-Sep-83                                                        Page 1
            DEPT  DEVICE          UIC             DATE          TIME
     B LEDERMAN
                  TT5          [300,003]        6-Sep-83      13,47,13
Number of logins for B LEDERMAN            1
     W ROACH
                  TT7          [300,002]        6-Sep-83      10,08,26
                  TT4          [300,002]        6-Sep-83      11,39,05
                  TT3          [300,002]        6-Sep-83      13,35,12
                  TT6          [300,002]        6-Sep-83      13,47,22
                  TT7          [300,002]        6-Sep-83      14,17,26
                  TT3          [300,002]        6-Sep-83      14,20,10
                  TT4          [300,002]        6-Sep-83      14,29,58
                  TT6          [300,002]        8-Sep-83      15,08,59
                  TT3          [300,002]        9-Sep-83      08,45,09
                  TT5          [300,002]        9-Sep-83      09,41,47
                  TT3          [300,002]        9-Sep-83      11,21,10
                  TT4          [300,002]        9-Sep-83      12,00,50
                  TT5          [300,002]        9-Sep-83      13,54,50
                  TT16         [300,002]        9-Sep-83      14,14,22
                  TT16         [300,002]        9-Sep-83      14,16,17
                  TT6          [300,002]        9-Sep-83      14,33,30
                  TT5          [300,002]        9-Sep-83      15,43,03
Number of logins for W ROACH              17
Number of logins for PNA      18
```

It is also very easy to get only the summary of the data, rather than all of the details simply by not printing the details. Where I want this, I create another procedure, which looks almost the same as the previous one:

```
DEFINE PROCEDURE LOGIN-SUMMARY
READY BASE-LOGIN
REPORT BASE-LOGIN WITH CODE=19 SORTED BY DEPT, NAME, DATE,
   TIMR ON LOGINSUM.RPT
SET COLUMNS-PAGE=80
SET REPORT-NAME = "Login Summary"
AT BOTTOM OF NAME PRINT "Number of logins for ", NAME,
   SPACE 1, COUNT
AT BOTTOM OF DEPT PRINT SKIP, "Number of logins for Department ",
   DEPT, SPACE 1, COUNT, SKIP 2
END-REPORT
FINISH
END-PROCEDURE
```

As may be seen, the only significant difference is that there is no unqualified PRINT statement, so the details do not print. It is not necessary to print details in order to have breaks on a field, or to summarize (COUNT) on a field. This report looks like this:

```
                    Login Summary                              12-Sep-83
                                                               Page 1

                                    NAME                 DEPT
  Number of logins for              LEDERMAN              1
  Number of logins for              ROACH                17
  Number of logins for Department                         PNA      18
```

So far, I consider what I did to be mostly re-shaping the DEC-supplied command file to a form I consider more acceptable in terms of efficiency or ease of use. I then started to consider making changes which would give me either reports not supplied by DEC, or which would improve performance. The first change is improving the retrieval of data from the data file. It every case, there is a selection of the CODE field, to extract those records of the type under consideration: with the existing sequential file, it is necessary to read the entire data file completely to find the records wanted for each report. It would be more efficient to put the data in an indexed file, with CODE as the primary key: then any reference to the data would quickly retrieve only those records with the correct CODE for that report. While it would be possible to read the sequential file and write to an indexed file with Datatrieve, the RMS utility IFL will do the same job, and is several orders of magnitude faster for this kind of operation. What I did was use the RMS DEF utility to define an appropriate file: the command file to create the data file is as follows.

```
;THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
; the name is ACCOUNT.DOM, and will be created
ACCOUNT.DOM
YES
;THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTRIBUTES
; an indexed file, fixed length records 120 bytes long,
; carriage control
IDX
```

```
FIX
120
YES
;THE FOLLOWING QUESTIONS DEAL WITH KEYS
; the key is an integer, two bytes long named CODE,
; and duplicates are allowed
INT
0
2
CODE
YES
NO
;THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
; no placement control, initial allocation 30 blocks, extend 5
blocks
NO
NO
30
5
NO
;THE NEXT QUESTIONS ASK ABOUT FILL SIZES FOR KEYS
; fill the areas completely (there will be no future insertions)
512
512
;THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
RWED
RWED
RWED
RWED
```

The data can then be inserted in the file with the following commands (which assume the RMS utilities are installed under their normal names):

```
SHOW ACCOUNTING/DATATRIEVE:ACNTRN.SYS ACCOUNT.SEQ
DEF @ACCOUNT.DEF            (the definition file given above)
IFL ACCOUNT.DOM=ACCOUNT.SEQ
```

It is true that it takes some time to insert the records in the indexed file, but from then on time is saved by the faster access, so if several reports are to be generated, or the data is going to be examined to find specific events, the net savings justifies creating the indexed file. No change is necessary in any of the record definitions or procedures, but the domain definition reference to ACCOUNT.DAT should be changed to ACCOUNT.DOM to access the new data file.

One of the reports I wanted to generate was for device mounts, dismounts, etc. The DEC-supplied reports give only one operation per report, and I wanted all mounts, dismounts, allocations, and de-allocations to be in a single report, and to be identified by name rather than by code number. To do this, I first created a domain with all of the operation codes.

```
DEFINE RECORD CODES-REC USING
01 CODES-REC.
   03 KEY PIC 99 USAGE COMP.
   03 TYPE PIC X(16).
;
DEFINE DOMAIN CODES USING CODES-REC ON CODES.DOM;
DEFINE FILE FOR CODES KEY=KEY;
```

Note that CODES uses an indexed file, so the data can be quickly retrieved by the KEY field. The data in CODES looks like this:

```
KEY TYPE
01  System
02  User
03  Task
09  Startup
10  Invalid Login
11  Time Change
12  Allocate
13  Deallocate
14  Mount
15  Dismount
16  Print
19  Login
20  Crash
21  Device Stats
22  Reset
27  Card Reader
```

This gives me a domain with all of the valid code types and their english translation: at the moment, I'm only using 12, 13, 14 and 15, but it was just as easy to put them all in, in case I want them later. It is also easy to ready and print CODES whenever I forget the number of a particular transaction type. Next, I defined something called a VIEW, which is simply a way to combine the information from more than one domain into what looks like a single domain.

```
DEFINE DOMAIN AM OF BASE-AM, CODES USING
01 AM OCCURS FOR BASE-AM WITH CODE=12,13,14,15.
    10 TYPES OCCURS FOR CODES WITH KEY=CODE.
           20 TYPE FROM CODES.
    10 DEPT FROM BASE-AM.
    10 NBR FROM BASE-AM.
    10 TERMINAL FROM BASE-AM.
    10 ACNT FROM BASE-AM.
    10 DATE FROM BASE-AM.
    10 TIM FROM BASE-AM.
    10 TIMR FROM BASE-AM.
    10 TIME FROM BASE-AM.
    10 DEVICE FROM BASE-AM.
;
```

The first line simply states that I am combining data from BASE-AM and CODES. The next means I only want data with a CODE of 12, 13, 14 or 15, which are the transactions I am interested in. By doing this, I will get only the proper transactions, and I don't have to remember to select the proper codes each time I access the domains, which is a great convenience. The next line is the one which matches up the code in BASE-AM with the corresponding record in CODE: thus the TYPE on the next line will contain the proper text for that transaction type (Mount, Deallocate, etc.). The rest of the view is simply the information I want to carry over from BASE-AM. Now, when AM is printed, each line will contain the proper TYPE for that transaction, and all four types of transactions, and only those transactions, will appear. This is very handy not only for reports, but also for direct query using Datatrieve: one can go in and print out the transactions for a particular user, or device, or combination. Having a view which automatically selects the proper records, so that one does not have to enter the "WITH CODE=nn" each time, is such a convenience that I created several other views for those transactions that I expect to be accessing for inquiries, such as Invalid login transactions, login transactions, print queue transactions, and user activity. Other transactions, which I expect to use only in reports, do not need a view as the "CODE=nn" can be fixed in the report procedure, and there is some extra overhead in using a view. Reporting a view is also possible:

```
DEFINE PROCEDURE AM-REPORT
READY AM
REPORT AM SORTED BY DATE, TIMR ON AM.RPT
SET COLUMNS-PAGE=80
SET REPORT-NAME="Allocate / Mount"/"De-allocate / Dismount"
PRINT TYPES, DEPT, TERMINAL, DATE,  DEVICE
END-REPORT
FINISH
END-PROCEDURE
```

A sample of the generated report follows.

```
                    Allocate / Mount                          12-Sep-83
                  De-allocate / Dismount                      Page 1
      TYPE         DEPT        TERMINAL        DATE          DEVICE
Mount              $SY          C00          8-Sep-83        DU0
Mount              $SY          C00          8-Sep-83        MT0
Mount              $SY          C00          8-Sep-83        MT1
Mount              $SY          C00          8-Sep-83        DB1
Dismount           SYS          TT7          9-Sep-83        DU0
Mount              SYS          TT7          9-Sep-83        DU0
Dismount           SYS          TT16         9-Sep-83        MT0
Dismount           SYS          TT16         9-Sep-83        MT1
Allocate           SYS          TT16         9-Sep-83        MT1
Mount              SYS          TT16         9-Sep-83        MT1
Deallocate         SYS          TT16         9-Sep-83        MT1
Dismount           SYS          TT16         9-Sep-83        MT1
```

Note that there is no "CODE=" qualifier in the REPORT command, but it is now necessary to sort by date and time, because the insertion of the data into the indexed file sorts the data by code only. It would have been very helpful if some uniformity in the record layout could have been followed by DEC, but as it is, it appears that a different team developed the packet for each different transaction, and they didn't talk to each other. Thus, fields which are common to all (or almost all) transactions, such as the date and time, appear in entirely different places in different records, and there doesn't seem to be anything the user can do about it now. It should also be noted that I am sorting on TIMR and not TIME: TIME is a COMPUTED-BY field, and Datatrieve-11 cannot sort a COMPUTED-BY field. I also like to create reports which are 80 columns wide, when possible, rather than 132 columns, as the paper is easier to handle, and it looks better than a few fields spread across a wide page with a lot of blank space between them.

My next major effort was with the device statistics. We have systems which place a heavy load on disk transactions, and balancing the load between disks appeared to be a way to improve throughput, and one of the reasons we went to M-Plus was to get the statistics to find out just what was happening on the disks. When I looked at the device report, however, I was a little disappointed to find the statistics recorded as running totals, rather than the total for the last scan interval. I can understand the reason for it, but I want the reports to show me what happened in the past hour (or other survey period), and not show me hours when nothing happened. It is possible for Datatrieve to do this through the use of an intermediate domain to store temporary data, the definitions for which are as follows:

```
DEFINE RECORD DEVICE-SUM-REC USING
01 DEVICE-SUM-REC.
    10 DATE PIC X(9).
    10 TIME PIC 9(6) EDIT-STRING 99,99,99.
    10 DEVICE PIC X(6).
    10 IO-SUM PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9
            QUERY-HEADER "I/O COUNT".
    10 WORD-SUM PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9
            QUERY-HEADER "WORD COUNT".
    10 CYL-SUM PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9
            QUERY-HEADER "CYLINDERS"/"CROSSED".
;

DEFINE DOMAIN DEVICE-SUM USING DEVICE-SUM-REC ON DEVSUM.SEQ;
```

This domain simply serves as a temporary storage place for the particular fields I want to appear in the final report. The procedure to do the actual conversion is:

```
DEFINE PROCEDURE DEVICE-SUM-REPORT
DECLARE IO PIC 9(9) USAGE COMP.
DECLARE WORD PIC 9(9) USAGE COMP.
DECLARE CYL PIC 9(9) USAGE COMP.
IO = 0
WORD = 0
CYL = 0
DEFINE FILE FOR DEVICE-SUM;
READY DEVICE-SUM WRITE
READY BASE-DEVICE
FOR BASE-DEVICE WITH CODE = 21 SORTED BY DEVICE, DATE, TIMR BEGIN
   STORE DEVICE-SUM USING BEGIN
            DATE = DATE
            TIME = TIME
            DEVICE = DEVICE
            IO-SUM = IO-COUNT - IO
            WORD-SUM = WORD-COUNT - WORD
            CYL-SUM  =  CYL-CROSSED - CYL
            END
   IO = IO-COUNT
   WORD = WORD-COUNT
   CYL = CYL-CROSSED
END
FINISH BASE-DEVICE
RELEASE IO
RELEASE WORD
RELEASE CYL
REPORT DEVICE-SUM WITH IO-SUM > 0 ON DEVSUM.RPT
SET COLUMNS-PAGE=80
SET REPORT-NAME = "Device Statistics Summary"
PRINT DEVICE-SUM-REC
AT BOTTOM OF DEVICE PRINT SKIP, "Maximums:", MAX(IO-SUM),
MAX(WORD-SUM), MAX(CYL-SUM), SKIP 2
END-REPORT
FINISH
END-PROCEDURE
```

The procedure simply goes through the device statistics data (sorted in the order I want the report, which is by device, date, and time), subtracts the current hour from the previous hour, and stores the difference. This leaves DEVICE-SUM containing the statistics for each hour sorted by the device, which is exactly what I want in the final report. The DEFINE FILE statement ensures that I start with a new data file, rather than picking up the data left over from the last time I ran the report: it is faster than deleting the old data, but it is necessary to remember to purge the old DEVSUM.SEQ files occasionally. Also note that it is necessary to declare all variables which are to be used, and that they must be initialized to zero. Once the data is converted, only the records where the activity is greater than zero will be reported to eliminate hours when nothing happened. The report looks like this (some detail lines were deleted just to save space, so the maximums reflect lines not on this page):

Device Statistics Summary 12-Sep-83
Page 1

| DATE | TIME | DEVICE | I/O COUNT | WORD COUNT | CYLINDERS CROSSED |
|---|---|---|---|---|---|
| 11-Sep-83 | 09,13,19 | DB0 | 125,060 | 22,017,672 | 1,784,389 |
| 11-Sep-83 | 10,13,19 | DB0 | 125,002 | 32,005,200 | 1,778,782 |
| 11-Sep-83 | 11,13,19 | DB0 | 124,997 | 32,001,544 | 1,778,216 |
| 11-Sep-83 | 12,13,19 | DB0 | 94,077 | 24,120,632 | 1,357,430 |
| 11-Sep-83 | 13,13,19 | DB0 | 241 | 61,696 | 26,304 |
| 11-Sep-83 | 14,13,19 | DB0 | 241 | 61,696 | 26,304 |
| 11-Sep-83 | 15,13,19 | DB0 | 241 | 61,696 | 26,304 |
| Maximums: | | | 297,604 | 22,120,108 | 6,326,075 |
| 11-Sep-83 | 07,13,19 | DB1 | 124,054 | 31,757,824 | 2,204,479 |
| 11-Sep-83 | 08,13,19 | DB1 | 124,077 | 31,763,712 | 2,215,187 |
| 11-Sep-83 | 09,13,19 | DB1 | 124,139 | 31,779,584 | 2,204,196 |
| 11-Sep-83 | 10,13,19 | DB1 | 124,060 | 31,759,360 | 2,224,216 |
| 11-Sep-83 | 11,13,19 | DB1 | 124,106 | 31,771,136 | 2,220,454 |
| 11-Sep-83 | 12,13,19 | DB1 | 93,336 | 23,894,016 | 1,657,114 |
| Maximums: | | | 124,358 | 47,766,154 | 2,951,050 |
| 11-Sep-83 | 09,13,19 | DU0 | 160,856 | 41,179,136 | 0 |
| 11-Sep-83 | 10,13,19 | DU0 | 160,811 | 41,167,616 | 0 |
| 11-Sep-83 | 11,13,19 | DU0 | 160,683 | 41,134,848 | 0 |
| 11-Sep-83 | 12,13,19 | DU0 | 120,714 | 30,902,784 | 0 |
| Maximums: | | | 160,856 | 41,179,136 | 0 |
| 11-Sep-83 | 09,13,19 | MT1 | 28,138 | 9,540,096 | 0 |
| 11-Sep-83 | 10,13,19 | MT1 | 28,122 | 9,534,464 | 0 |
| 11-Sep-83 | 11,13,19 | MT1 | 28,109 | 9,529,856 | 0 |
| Maximums: | | | 28,138 | 9,540,096 | 0 |

Users should note that this procedure will issue a warning message, "Proceeding to report unsorted records", because there is an "AT BOTTOM" statement for a field for which there is no sort parameter in the REPORT command. This is a warning message only, and may be ignored as the file being reported is a sequential file which is already sorted in the proper order for the desired report. It may also be noted that in this particular case, since DEVICE-SUM is always written in order, and reported in that same order, it can use a sequential file to store data, and no performance advantage would be obtained by making DEVSUM.SEQ into an indexed file.

I found an interesting statistic the first time I ran this report, which was that the minimum disk activity for the system disk never went below 240 QIO's, even at night when I knew no one was on the system. I concluded that this is the activity of the system accounting package itself, accumulating data. This activity was measured without task accounting, and without DECnet, with a 1 hour scan interval: changes to any of these factors would change

the base level of activity. If this activity was not wanted on the final report, the qualifier "WITH IO-SUM > 0" could be changed to one which would reject activity below a given level.

Another report not provided by DEC, but one I think will be of interest, shows a calculated charge amount for each user and department based on the system resources used. The starting point for this is the User Account transactions record, with an additional charge field:

```
DEFINE RECORD CHARGE-REC USING
01 CHARGE-REC.
    10 CODE PIC 99 USAGE COMP.
    10 DEPT PIC XXX.
    10 NBR PIC X(7).
    10 DEVICE PIC X(6).
    10 ACNT PIC 9(4) USAGE COMP.
    10 UIC PIC X(10).
    10 LOGIN-DATE USAGE DATE.
    10 LITIM.
          20 FILLER PIC X(6).
    10 LITMR REDEFINES LITIM.
          20 LIH PIC 99 USAGE COMP.
          20 LIM PIC 99 USAGE COMP.
          20 LIS PIC 99 USAGE COMP.
          20 LID PIC 9(18) COMPUTED BY (0.0000001 * LOGIN-DATE).
    10 LOGIN-TIME PIC 9(6) COMPUTED BY (LIH * 10000) +
          (LIM * 100) + LIS EDIT-STRING 99,99,99.
    10 LOGOUT-DATE USAGE DATE.
    10 LOTIM.
          20 FILLER PIC X(6).
    10 LOTMR REDEFINES LOTIM.
          20 LOH PIC 99 USAGE COMP.
          20 LOM PIC 99 USAGE COMP.
          20 LOS PIC 99 USAGE COMP.
    10 LOGOUT-TIME PIC 9(6) COMPUTED BY (LOH * 10000) +
          (LOM * 100) + LOS EDIT-STRING 99,99,99.
    10 BILLING-DATE USAGE DATE.
    10 BITIM.
          20 FILLER PIC X(6).
    10 BITMR REDEFINES BITIM.
          20 BIH PIC 99 USAGE COMP.
          20 BIM PIC 99 USAGE COMP.
          20 BIS PIC 99 USAGE COMP.
          20 BID PIC 9(18)
                COMPUTED BY (0.0000001 * BILLING-DATE).
    10 BILLING-TIME PIC 9(6) COMPUTED BY (BIH * 10000) +
          (BIM * 100) + BIS EDIT-STRING 99,99,99.
    10 ELAPSED PIC 9(12) EDIT-STRING ZZZ,ZZ9
          COMPUTED BY (BID - LID).
    10 CPU-TIME PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9.
    10 TASKS-ACTIVE PIC 999 USAGE COMP EDIT-STRING ZZ9.
    10 TASKS-RUN PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9.
```

```
    10 DIRECTIVES PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9.
    10 QIOS PIC 9(9) USAGE COMP EDIT-STRING ZZZ,ZZZ,ZZ9.
    10 FILLER PIC X(30).
    10 CHARGES PIC 99999V99 COMPUTED BY (CPU-TIME * 0.001) +
    (DIRECTIVES * 0.0003) + (QIOS * 0.0008) EDIT-STRING $$,$$$.Z9.
;
```

Most of these fields are not needed in the report I generate, but I have left them in should I want them later for a different report. The field of importance is the CHARGES, which uses the Datatrieve COMPUTED BY feature much as the time was computed earlier: in this case, I have decided to charge for use of the CPU and for all directives and QIOs generated, but the charge could also be based on the number of tasks run, etc. I have also added three other fields, LID, computed from the LOGIN-DATE, and BID, computed by the BILLING-DATE, and ELAPSED, which is the difference between the two. Normally, Datatrieve-11 puts only the date and not the time in DATE fields, but the system accounting package actually fills in the complete date and time (in clunks) when it stores the data. These can be scaled down from clunks to seconds as I have done here, and can then be used to calculate the field ELAPSED, which is the amount of time from when the user logged in to when either the user logged out or system accounting was shut down, in seconds. Using these date fields insures the proper elapsed time will be calculated even if the period a user was logged in covers more than one day: simply subtracting the logout time from the login time would not take the date change into account. If LOGOUT-DATE had been used instead of BILLING-DATE, then any one who was logged in when the system was shut down (for example, the system manager must be logged in to stop accounting or shut down the system) would have a LOGOUT-DATE of zero, and the times would not be calculated properly. This field could be used to charge for the amount of time a user was logged on the system by including it in the computation of CHARGES; I simply decided not to charge for it in my example. One other minor change which might be noted is in the LITIM, LOTIM and BITIM groups, where I used one FILLER which is 6 characters long to skip over the three time fields: it is not necessary for filler to be the same type of field at that which it replaces, nor for there to be the same number of fields; it is only necessary that the filler be the proper total length. The report generator is very much like the other reports:

```
DEFINE PROCEDURE CHARGE2-REPORT
READY CHARGE
REPORT CHARGE WITH CODE=02 SORTED BY DEPT, UIC ON CHARGE2.RPT
SET COLUMNS-PAGE=78
SET REPORT-NAME = "System Charges by Department"/"and User"
AT BOTTOM OF UIC PRINT "USER", SPACE 1, UIC(" "),
   TOTAL(CPU-TIME) USING ZZZ,ZZZ,ZZ9.,
   TOTAL(DIRECTIVES) USING ZZZ,ZZZ,ZZ9.,
   TOTAL(QIOS) USING ZZZ,ZZZ,ZZ9.,
   TOTAL(CHARGES) USING $$$,$$$,$$9.99, SKIP
AT BOTTOM OF DEPT PRINT SKIP 1, "DEPT", SPACE 1, DEPT(" "),
```

```
   TOTAL(CPU-TIME) USING ZZZ,ZZZ,ZZ9.,
   TOTAL(DIRECTIVES) USING ZZZ,ZZZ,ZZ9.,
   TOTAL(QIOS) USING ZZZ,ZZZ,ZZ9.,
   TOTAL(CHARGES) USING $$$,$$$,$$9.99, SKIP 2
END-REPORT
FINISH
END-PROCEDURE
```

The report generated is:

```
                System Charges by Department                    13-Sep-83
                          and User                              Page 1
                          CPU
                          TIME        DIRECTIVES     QIOS      CHARGES
USER    [001,001]      380,324.     1,080,404.   185,690.    $852.99
USER    [200,001]      226,423.       975,196.   142,910.    $633.30
DEPT                   606,747.     2,055,600.   328,600.  $1,486.30
USER    [000,000]    1,871,658.    17,319,746. 1,992,107.  $8,661.26
DEPT $SY             1,871,658.    17,319,746. 1,992,107.  $8,661.26
USER    [300,002]       22,600.        55,529.    31,486.     $64.44
USER    [300,003]          175.           510.       161.      $0.45
DEPT PNA                22,775.        56,039.    31,647.     $64.90
USER    [001,001]   12,987,024.   117,267,216. 30,962,109. $72,936.87
DEPT SYS            12,987,024.   117,267,216. 30,962,109. $72,936.87
```

One could easily have each department print on a seperate page, or print department totals only, or even include the details and give the charges for each login session. We don't charge users on our system, so I have not done much work on these reports, but I'm now prepared if we do start charging.

When working with the above reports, I noticed that although system accounting records all jobs on print queues, it does not record jobs on batch queues. I also became curious about what statistics could be accumulated with Datatrieve without using the system accounting package. Since the start and end of each batch job is logged on the system console, and the console logging package allows this information to also be stored in a file on disk, I decided to try to read this file with Datatrieve and see what information I could retrieve. Typically, a console log would contain information such as this:

```
08:06:17  COT -- Date is 26-SEP-83
08:06:17  ERRLOG -- Error Logging initialized
08:13:07  Login  user ROACH          [300,2] TT16:
08:15:23  Login  user SYSTEM         [1,1] TT3:
  09:03:52  *** MT0:  -- Dismount complete
09:03:52  Logout user ROACH           [300,2] TT17:
09:04:20  Logout user ROACH           [300,2] TT16:
09:19:22  Login  user ROACH          [300,2] TT17:
15:34:30  Batch job - BACKUP     started on VT1:
15:34:40  Login  user SYSTEM         [1,1] HT0:
15:35:18  Logout user SYSTEM          [1,1] HT0:
```

```
  15:45:25  *** DB1:  -- Dismount complete
15:45:27  Batch job - BACKUP    completed on VT1:
15:45:40  Login  user SYSTEM        [1,1] TT16:
15:52:19  ERRLOG -- Error Logging stopped
15:52:46  ERRLOG -- Error Logging initialized
```

Note the dismount messages: on the console, they line up with the other records, but the first two characters in the record are nulls, and these must be counted in the record definition. Because Datatrieve preferrs fixed length records, and some of the other console records can be very long (especially DECnet and task termination messages), I decided to first move the data from the log file to one with fixed length records by defining a file with RMSDEF as I did for system accounting:

```
; The first question asks for the file specification.
CONSOLE.SEQ
NO
; The next questions deal with file organization & record
attributes
FIX
56.
; The next questions deal with allocation and placement attributes
30
10
; The following questions deal with file protection
RWED
RWED
RWED
R
```

Data is moved from one file to the other with the command:

```
CNV CONSOLE.SEQ/AP/TR/PD = CONSOLE.LOG
```

The /AP places the new data at the end of the CONSOLE.SEQ file and must be present even if CONSOLE.SEQ is empty, and both the /TR (truncate) and /PD (pad) switches must be present. Now that the data is in fixed length records, I can write a record definition for the fields I want:

```
DEFINE RECORD CON-REC USING
01 CON-REC.
   10 BASE.
         20 TEXT PIC X(56).
   10 COT REDEFINES BASE.
         20 COTIME PIC X(8).
         20 FILLER PIC X(2).
         20 COTID PIC X(12).
         20 FILLER PIC X(3).
         20 COTDATE PIC X(9).
   10 LOGIN REDEFINES BASE.
         20 LITIM PIC X(8).
```

```
            20 FILLER PIC XX.
            20 LOGID PIC X(6).
            20 FILLER PIC X(6).
            20 LOGNAM PIC X(14).
      10 DMO REDEFINES BASE.
            20 FILLER PIC XX.
            20 DMOTIM PIC X(8).
            20 FILLER PIC X(6).
            20 DMODEV PIC X(4).
            20 FILLER PIC X(5).
            20 DMOID PIC X(8).
      10 BATCH REDEFINES BASE.
            20 BATIM PIC X(8).
            20 FILLER PIC XX.
            20 BATID PIC X(5).
            20 FILLER PIC X(7).
            20 BATNAM PIC X(9).
            20 FILLER PIC X.
            20 BATTYP PIC X(7).
            20 FILLER PIC X(4).
            20 DEVICE PIC X(6).
            20 D1 REDEFINES DEVICE.
                  30 DEV1 PIC X(4).
            20 D2 REDEFINES DEVICE.
                  30 FILLER PIC XX.
                  30 DEV2 PIC X(4).
;
```

This definition looks very much like one of the DEC supplied system accounting definitions. In this case, the different pages must all be in one definition as there is no way to tell one record from another except by examining the contents of a field within the record, which I will show in the following procedure. I decided the next step would have to be to read this file, and create another which would have a more uniform field layout, and which would have the date in every record. I therefore defined a second domain into which the console log data will be placed.

```
DEFINE RECORD C-REC USING
01 C-REC.
   10 DATE USAGE IS DATE.
   10 TIME PIC X(8).
   10 TYPE PIC X(12).
   10 DEVICE PIC X(4).
   10 TEXT PIC X(16) QUERY-HEADER IS " ".
;
```

This simple definition puts the fields in fixed locations, which will simplify processing it later. To get the data in, one must read the file created above, and store the data in the new domain.

```
DEFINE PROCEDURE CONVERT
READY CONSOLE READ
```

```
READY C WRITE
DECLARE S-DATE USAGE IS DATE.
FOR CONSOLE BEGIN
   IF COTID = "COT -- Date " S-DATE = COTDATE
   IF LOGID = "Login ", "Logout" STORE C USING BEGIN
            DATE = S-DATE
            TIME = LITIM
            TYPE = LOGID
            TEXT = LOGNAM
            END
   IF DMOID = "Dismount" STORE C USING BEGIN
            DATE = S-DATE
            TIME = DMOTIM
            TYPE = "Dismount"
            DEVICE = DMODEV
            END
   IF BATID = "Batch" STORE C USING BEGIN
            DATE = S-DATE
            TIME = BATIM
            IF BATTYP = "started" BEGIN
                     DEVICE = DEV1
                     TYPE = "Batch Start"
                     END
            IF BATTYP = "complet" BEGIN
                     DEVICE = DEV2
                     TYPE = "Batch Stop"
                     END
            TEXT = BATNAM
            END
END
FINISH
END-PROCEDURE
```

Note that each IF statement requires a different field to be examined to determine what type of record is involved: this is why all of the pages must be present in one record definition. The two device fields are required in the batch record as they appear in different positions for those two records. The temporary field S-DATE is used to hold the date whenever COT logs a date change, so that all records which follow will have the current date. The C domain now contains data in a uniform format.

```
   DATE        TIME        TYPE      DEVICE
21-Sep-83   09:47:12 Logout                SCOTT
21-Sep-83   09:48:25 Dismount       MT1:
21-Sep-83   09:52:15 Batch Stop     VT1:   BRU
21-Sep-83   09:53:41 Dismount       DB1:
21-Sep-83   09:53:47 Dismount       DU0:
21-Sep-83   09:57:44 Login                 SCOTT
21-Sep-83   09:57:56 Batch Start    VT1:   BRUDB1
21-Sep-83   09:59:08 Logout                SYSTEM
21-Sep-83   10:05:50 Login                 ROACH
21-Sep-83   10:10:05 Login                 SYSTEM
```

```
21-Sep-83    10:11:08 Logout                ROACH
21-Sep-83    10:14:45 Login                 ROACH
21-Sep-83    10:18:15 Dismount      DB1:
21-Sep-83    10:18:16 Dismount      DU0:
21-Sep-83    10:18:24 Batch Stop    VT1:    BRUDB1
21-Sep-83    10:22:41 Login                 ANALYSIS
21-Sep-83    10:26:52 Logout                ANALYSIS
```

With the data in this condition, it is much easier to examine it, or create reports. The only report I am interested in at the present is the batch jobs:

```
DEFINE PROCEDURE BATCH-REPORT
READY C READ
REPORT C WITH TYPE CONT "Batch" ON BATCH.RPT
SET REPORT-NAME = "Batch Jobs"
SET COLUMNS-PAGE = 80
PRINT C-REC
END-REPORT
END-PROCEDURE
```

The report is:

```
                        Batch Jobs                              23-Sep-83
                                                                Page 1
     DATE            TIME            TYPE            DEVICE
  17-Sep-83       00:00:24     Batch Start          VT1:        BRU
  17-Sep-83       00:02:53     Batch Stop           VT1:        BRU
  20-Sep-83       13:46:02     Batch Start          VT1:        BRU
  20-Sep-83       13:46:18     Batch Stop           VT1:        BRU
  20-Sep-83       13:46:44     Batch Start          VT1:        BRU
  20-Sep-83       13:56:54     Batch Stop           VT1:        BRU
  21-Sep-83       08:21:21     Batch Start          VT1:        BRU
  21-Sep-83       09:41:22     Batch Start          VT1:        BRU
  21-Sep-83       09:52:15     Batch Stop           VT1:        BRU
  21-Sep-83       09:57:56     Batch Start          VT1:        BRUDB1
  21-Sep-83       10:18:24     Batch Stop           VT1:        BRUDB1
```

By adding more pages to the record definition, it would also be possible to extract task termination messages, error log warnings that a device has exceeded it's hard error limit, or any other message which appears on the console. This technique may be useful for RSX-11M systems which do not have system accounting.

At this point, the reader will hopefully have a better idea as to the type of information and reports which may be obtained using Datatrieve to process the system accounting information. It is my belief that the information gathered is potentially of considerable value to a system manager, and that Datatrieve is the easiest way to format this information in a form where it will be of the greatest use. I have attempted to show how the reports can be re-formatted to suit the individual site, and to extract information in ways not provided for in the package supplied by

DEC, and it is hoped that readers will now be better prepared to suit the system accounting package to their own sites. The author welcomes comments or suggestions about this paper and the material presented.

# Preliminary Index To The Fall 1983 SIG Tape

From the tape

Peruse the following to whet your appetite for what is coming on the Fall 1983 tape. With Glenn Everhart in control of the SIG tape copy we can expect to see this tape shortly. If you don't know how to get a copy, contact one of the people Glenn lists in his article in this issue.

Contents of the RSX Fall 1983 SIG tape (RSX83B)
listed by UIC.

==========

[300,70] FCB program for M+ 2.0, from Frank Penner

[301,100] CRT library for VT100 control, David Truesdell

[307,50] Runoff from RSX Sig Runoff Working Group. From Chuck Spalding. (Note: See also [332,13] Rice University Runoff, this tape.)

[307,110] Multi Trek, multi user startrek game, George Whittlesey

[312,114] RSX11M/M+ Activity monitor. Written for V3.2, later version status unknown. Counts I/O, CPU, etc. Mike Drabicky.

[312,315] Large collection. COMLIB module patch to get BRU to write files from tape to Vax disks. Resubmission of old ATT task (read/set file attributes - very handy). LISTRS - multicolumn lister. DGT - read DG, Unix TAR, IBM, many more tapes; writes some too. Fixed up ORC object disassembler.
Glenn Everhart

[312,360] Resubmission of IAS virtual disks and pseudo tty drivers.

[312,345] Latest "generic" and Vax PortaCalc spreadsheet. For VAX use COMPILVMX. For RSX use SOMAKE command files. Faster/more features than 83A or 82B.

[312,366] PortaCalc-XL - RSX version of PortaCalc using virtual arrays to give large sheet.
Glenn Everhart

[312,322] Multi-file or multi-disk virtual disk package with security enhancements. Glenn Everhart

[312,346] Modified FX: memory virtual disk for M/M+ which won't hold up fork queue. Glenn Everhart

[312,365] Desk Top Calendar. Appointment/meeting scheduler and calendar display/query. Mitch Wyle.

[326,1] through [326,114] - North Texas Lug (Jeff Hamilton): Spelling checker update from F82 submission in same directory. More features, faster. Also signal processing programs, tape copier, SCCS, day-of-week subroutine and more.

[332,13] Rice University Runoff. Many word processing enhancements and letter quality output support. Syntax similar to DSR, equation handling, much much more. A "Best Buy" of the tape. John Clement

[343,30] Interr - interrogate a central DECnet node for new network control info in big networks. Bruce Mitchell

[343,36] RSX11M/M+ User Monitor. Can do logging only during certain hours also. Bruce Mitchell.

[343,37] Network Time coordinators. Allows all times on a network to agree.

[344,1] KMSKIT. Master documentation area for [344,*] here. J. Downward

[344,43] KMSKIT utilities, including updates of GREP and LIST.

[344,44] An old, SMALL RMDEMO showing KMS Accounting statistics and normal RMD things.

[344,45] Program Development Queues. Multiqueue command file despooler. Allows semi-batch operations.

[344,61] CCL V9.0 (Paul Sorenson). A MUST for any RSX11M or M+ system! Don't leave home without it! Makes RSX super friendly, flexible, and tailorable, w/o any SYSGEN.

[344,65] Complete RSX11M V4.1 System Accounting package. Oriented to performance monitoring, but useful for chargeback too. Jim Downward

[346,103] Machine-readable versions of some of the handouts. Included are Hows and Whys of ASTs in RSX; MSCAN; Cluster & Resident Libraries; and RSX Sysgen session.

[352,4] SRD Working Group SRD. This is the definitive version of SRD, a super-useful directory/file maintenance utility no RSX system should be without. From Bob Turkelson and the working group.

[352,20] Friendly - a modified Fortune Cookie program intended to be installed as a catch task after CCL or MCR which prints random "friendly" messages instead of the monotonous

"nonexistent command" or similar messages produced by unrecognized MCR or CCL commands. (Other sites install DAMMIT or COOKIE for similar purposes.)
Bob Turkelson.

[356,10] Manuscript and files for developing an ACP for RSX11M in a Higher Order Language. From Carl Mickelson.

[356,20] Supermac in Fortran. The Supermac macros give a structured and powerful assembler language, but using MACRO-11 it is painfully slow. This program translates Supermac into straight MACRO much faster. Warning: the Fortran dialect may not be that of RSX. From Tom Weslowski and Richard DiMidio, SUNY Oswego.

[356,30] Datatrieve structures needed to produce accounting reports from the built-in accounting in RSX11M+. Some reports driven off console log are also available; these work in M or M+.

[356,31] SFGL70 graphics package enhancement. Pen plotter interface added to existing Tek 4010/4014 support. A sample piechart drawer is included. From Bart Z. Lederman.

[356,40] KERMIT distribution. This is a multi-system communications package for most micros and many mainframes including many for DEC. It does what the expensive ones do, but is free and of high quality. The RSX versions are imminent, but sources in C, Pascal, and Fortran are provided in case you want to roll your own for the time being.
From Columbia University.

[356,50] RSX11M version of Empire, a screen oriented game.

[356,51] Adventure compiler and runtime for RSX, with an advanced version of the original Adventure as an example.

[356,60] Tape rewind pgm from Gordon Ross.

[356,70] 3D Plotting package for Tektronix 4014 terminals or DMPC plotters, in Fortran, from Dennis V. Jensen.

[370,60] Utilities:
CHEKIT - Check record lengths
TRUNCATE - Remove trailing blanks from files
SOLO - a UNIQ processor, removes duplicate records.
Clear - erases VT52/VT100 screen (note: built-in command in CCL.)

[370,341] TED full screen and line editor. Versions for several OSs included. From Brian Nelson.

# Introduction to RSX Indirect Command Files

Allen A. Watson
THE RECORD

From the editor

This article contains the text of the handout for both the St. Louis and Las Vegas symposia. If you have the handout you already have this.

REFERENCE DOCUMENTATION

Chapter 4 of the MCR Operations Manual is the official, complete reference. Also, on the latest releases of M and M-PLUS, there is very complete on-line help. See Appendix B for an index of on-line help.

The material presented here is based on RSX11M V4.0 or later and RSX11M-PLUS V2.0 or later. Most, but not all, applies to earlier versions as well.

## 1.0 WHAT IS AN INDIRECT COMMAND FILE?

### 1.1 A Series Of MCR Or DCL Commands

An indirect command file (henceforth abbreviated as ICF) may be a series of MCR and/or DCL commands that can be executed with a single command line. The Indirect Command Processor reads the file and responds to each line as though it were typed in from your terminal.

### 1.2 An Interpreted Computer Program

An ICF can contain "indirect commands" (direct commands to the Indirect Command Processor) that are interpreted and executed directly. By "interpreted" I mean they do not need to be compiled prior to execution, like the majority of programming languages. An interpretive language is less efficient, because the source code must be interpreted each time it is executed, but it can be written and debugged much faster because there is no compilation or task building.

The Indirect Command Processor is a rich language with capabilities of loop control, integer, string and logical variables, subroutines, parameter passing, file I/O, and easy access to system information. It is actually a programming language.

### 1.3 A Mixture Of The Above

The ICF can contain a mixture of the above two elements, with the MCR or DCL commands being conditionally executed under control of the Indirect Command Processor statements.

## 2.0 WHAT ARE ICF'S USED FOR?

What can indirect command files be used for? Some suggestions are:

### 2.1 Frequently Repeated Operations

Commands for frequently repeated operations such as compilations, task builds, disk backups , or error reports can be stored in ICF's and executed with a single command line.

The ICF [1,2]STARTUP.CMD is a good illustration. It is built by SYSGEN to perform the basic operations needed to start up your system, and can be tailored by the user to add specialized operations.

### 2.2 Temporary Repetition Of Long Command Sequences

Any time you expect to type a complicated command line more than two or three times, you can store it in an ICF and execute it with a one-word command.

### 2.3 Commands With Complex Parameters

For frequent, fairly complex operations with a multitude of options, ICF's can act as a user interface. The ICF can prompt the user, validate answers, and handle erroneous input, and then construct the complex MCR or DCL command from the input. A good example is the BRU program. (See BRUTAPE.CMD on the Spring 1983 RSX SIG tape).

## 2.4 Quick And Dirty Programs

For infrequent functions that are not worth the effort of developing a formal program, ICF's can provide a quick way to program the desired function. (See OCTAL.CMD in the handout for the "NIFTY THINGS" session.)

## 2.5 As Program Input

Certain Digital utilities accept their commands from ICF's. For example, PIP, TKB, MAC, FOR, and DTR all accept input from indirect files. The commands in these files must be commands known by the utility, not MCR commands or commands to be interpreted by the Indirect Command Processor, so this type of ICF represents a special class and is not treated further in this document.

# 3.0 THE BASICS OF ICF'S

This section will present only selected, basic information about ICF's. The full description of ICF's and the commands recognized by the processor may be found in the last section of the MCR Operations manual contained in your documentation set. This section deserves careful reading and re-reading. In preparing for this talk I re-read it and discovered several things I had overlooked the last few times I had read it.

The best way to learn how to use command files is to read other people's code. As Tom Lehrer once wrote, "Plagarize! Plagarize! Let no one else's work evade your eyes!" The SYSGEN command files are excellent sources of interesting techniques; the SIG tapes always have numerous examples.

## 3.1 How An ICF Is Used

To invoke an indirect command file, you simply type an at-sign (@) in response to the system prompt, followed by the file name. For example:

```
@MOUNT.CMD
```

Since the Indirect Command Processor will default to a file type of ".CMD", that portion of the file specification can be omitted if you have named the file with a ".CMD" file type. That is, a file called MOUNT.CMD can be invoked just by typing:

```
@MOUNT
```

If the file has another file type, for example MOUNT.TMP, then the file type must be included:

```
@MOUNT.TMP
```

## 3.2 Symbolic Variables And Data Types

You can define symbols as variables to contain one of three types of data: numeric, string, or logical (true/false). The symbols you define must contain from one to six alphanumeric characters (including also the dollar sign) and must begin with a letter or dollar sign.

Logical variables have a true or false value.

String variables have as a value a string of ASCII characters with a length between 0 and 132 decimal.

Numeric symbols have a numeric value in the range 0 to 65535 decimal. The value can be decimal or octal.

The first definition of a symbol sets its data type; that is, once a symbol has been assigned a numeric value, it cannot later be given a string value, unless you erase the symbol and redefine it.

Symbols are assigned values by some form of a "SET" command. There is a different form of command for each data type.

```
.SETN NUMBER 7  -- sets symbol "NUMBER" to 7
                   NUMBER is now a numeric symbol.
.SETT FLAG      -- sets symbol "FLAG" true.
.SETF FLAG      -- sets symbol "FLAG" false.
                   FLAG is a logical symbol.
     .SETL FLAG SWITCH -Sets symbol "FLAG" to the same
                        logical value as "SWITCH".
.SETS WORD "BOO" - sets symbol "WORD" to
                   a value of "BOO". WORD is now
                   a string symbol.
```

Notice that each Indirect Command starts with a period, e.g. ".SETN". This is true of all commands recognized by the processor.

The above commands also used constant values, the number 7 and the string "BOO". String constants are any text enclosed in double quotes, e.g. "this is a string constant". Numeric constants are, as might be expected, a sequence of digits. However, if the digits do not terminate with a decimal point the number is understood to be an octal number. Thus:

```
10      --      octal 10, equal to decimal 8.
10.     --      decimal 10, equal to octal 12.
18      --      an illegal octal constant (8 is
                an invalid octal digit).
18.     --      decimal 18, equal to octal 22.
_#22     --      octal 22 ("_#" needed only when
                DECIMAL mode has been enabled,
                see below).
```

One of the most frequent errors of novices in writing ICF's is using a numeric constant, thinking in decimal, but forgetting the decimal point. Repeating a loop "20" times will give only sixteen repetitions! You want to repeat it "20." times.

## 3.3 Special Symbols

The Indirect Processor recognizes many special symbols designated by enclosing angle brackets. For example, the symbol <DATE> always contains today's date in the usual system format, for example "05-JUN-83". <TIME> has the current time, and <UIC> contains the current user's default User File Directory -- for example "[50,100]". If the UIC is changed by a SET /UIC command, the value of <UIC> is automatically updated.

There are dozens of such symbols described in Chapter 4 of the MCR Operations manual; you should examine the list and consider how each one might be used. Some advanced uses will be discussed in another session, "Nifty Things to Do with Indirect Command Files".

## 3.4 Terminal Input

The processor provides several directives that allow direct data input from the user's terminal. These are:

```
.ASK    -- input for a logical symbol
.ASKS   -- input for a string symbol
.ASKN   -- input for a numeric symbol
```

The basic format for an ASK type directive is:

```
.ASKx symbol question-text
```

where "x" is either blank, or "S", or "N". When the command is executed, the "question-text" is typed at the user's terminal and then input is accepted. The answer typed is stored in "symbol". For example:

```
.ASK GO Do you wish to continue
```

produces the output:

```
*Do you wish to continue? [Y/N]:
```

If the user types in "Y", the symbol GO is set TRUE.
If the user types in "N", the symbol GO is set FALSE.

```
.ASKS LINE Enter some text
```

produces the output:

```
*Enter some text [S]:
```

Whatever the user types is stored as a string in LINE.

```
.ASKN NUM Enter any number
```

produces the output:

```
*Enter any number [0]:
```

Any number typed will be accepted and stored in NUM.

In the last case, the number entered must follow the same format as numeric constants, that is, decimal numbers have to be followed by a decimal point. Since this is not natural to most people, there is a way to tell the processor to interpret all numbers entered as decimal numbers. To do this you must indicate in your command either an allowable range or a default using the decimal notation. This uses a second form of the .ASKN command:

```
.ASKN [lo:hi:def] symbol question-text
```

If any one of the three values (lo, hi or default) are expressed in decimal notation with a trailing decimal point, the number entered by the user in response to the question will be assumed to be decimal. For example:

```
.ASKN[1.:100.] NUM Enter a number from 1 to 100
```

This will produce the output:

```
*Enter a number from 1 to 100 [D R:1.-100.]:
```

As you can see, the processor indicates by the "D" that decimal input is expected, and also indicates the allowed range.

If you want all the numeric input to default to decimal, you can simply put the statement ".ENABLE DECIMAL" at the beginning of your command file, and all ASKN statements will expect decimal inputs. All numeric literals in your command file will also be interpreted as decimal numbers unless you precede them with a "#". If you need to use a bunch of octal numbers you should ".DISABLE DECIMAL" first.

To summarize, by default Indirect is in octal mode: numbers will be interpreted as octal unless you terminate them with a decimal point. If you enable DECIMAL mode, then all numbers will be interpreted as decimal unless you precede them with a "#".

With all .ASK directives, you can indicate ranges and defaults. With a string variable you can accept, for example, only four to seven characters. If the users types an invalid response, he is

warned, and the question is repeated automatically. For example, this dialogue might occur at a terminal:

```
>* Enter number [D R:1.-100.]: 200
AT.T2  -- Value not in range
>* Enter number [D R:1.-100.]: 99
```

"AT.T2" is the name given by the system to the Indirect processor for Terminal number 2. "AT." is the generic name of the processor, and the terminal number is appended for each person who executes by using the at-sign to invoke it.

## 3.5 Substitution

When a value has been assigned to a symbol, that value may be substituted in any command lines that follow if the substitution mode has been enabled. To enable substitution you must include the command:

```
.ENABLE SUBSTITUTION
```

at the beginning of your command file. My experience has shown that I almost always want to substitute, and so I inevitably start all my command files with this line.

To substitute a symbolic value in a line, enclose the symbol in single quotes. The following example will ask for a file name and then will assemble, task build, and run it, assuming it is the name of a simple MACRO program with an extention of .MAC.

```
Command File                Output
--------------------        ----------------------------
.ENABLE SUBSTITUTION
.ASKS FILE File name        >*File name [S]: prog1
MAC 'FILE'='FILE'           >MAC PROG1=PROG1
TKB 'FILE'='FILE'           >TKB PROG1=PROG1
RUN 'FILE'                  >RUN PROG1
                            >@ <EOF>
```

## 3.6 Terminal Output

Outputting text to a terminal can be done in one of two ways: through comment lines, or by opening the terminal as a file.

### 3.6.1 Comment Output -

Any line beginning with a semicolon (;) is treated by the processor as a comment instead of something to execute. (This is also true of MCR and DCL.) If you wish to output a brief message to the terminal, you can do so by including it as a comment in your ICF, like this:

```
; Continuing with this process will crash the system.
.ASK DUMB Do you really want to go on
```

This will produce the output:

```
>; Continuing with this process will crash the system.
>*Do you really want to go on? [Y/N]:
```

You can substitute in comment lines just like any other. Thus, we might modify the above MAC example like this:

```
Command File                Output
--------------------        ----------------------------
.ENABLE SUBSTITUTION
.ASKS FILE File name        >*File name [S]: prog1
; 'FILE'.MAC assembly,      >; PROG1.MAC assembly,
; task build and run.       >; task build and run.
.ASK YN Continue            >*Continue? [Y/N]: Y
.IFF YN .EXIT
MAC 'FILE'='FILE'           >MAC PROG1=PROG1
TKB 'FILE'='FILE'           >TKB PROG1=PROG1
RUN 'FILE'                  >RUN PROG1
                            >@ <EOF>
```

This allows the user a chance to review what he typed and to exit if he was wrong. We have used two new directives here, .IFF and .EXIT. .EXIT does just what it sounds like; .IFF is a conditional statement used with a logical variable; it means "if false", and the statement following it is executed only if the logical variable is false.

### 3.6.2 Opening The Terminal As A File -

When you have a lot of text to output to the terminal, such as lengthly instructions, it is useful to open the terminal as a file. Three directives come into play here: .OPEN, .DATA, and .CLOSE. (You can have up to four files open at once from an ICF; see the manual for more information.)

The .OPEN directive opens a file for output. The .DATA directive tells the processor to write the remainder of the line to the open file, and .CLOSE tells the processor to close the file. Here is an example of writing data to the terminal:

```
.OPEN TI:
.DATA This text will appear on your terminal when this command
.DATA file is executed. We will now ask for a number.
.ASKN NUM Enter number
.DATA You entered the number 'NUM'.
.CLOSE
```

This will produce the following output:

```
This text will appear on your terminal when this command
file is executed. We will now ask for a number.
>*Enter number [0]: 23
You entered the number 23.
```

Notice that using .DATA does away with the system prompt and the semicolon in front of all the text that you get when you use comments for output.

If you have a whole bunch of text to output at once, you can open the terminal with .OPEN TI: and then, any any point, use .ENABLE DATA to specify that all following lines are data until a .DISABLE DATA is encountered. For example:

```
.OPEN TI:
.ENABLE DATA
This text will appear at the terminal. If it is
preceded by spaces or tabs those will also be
        output to the terminal.
This method is good for outputting tables and
such in columns, because what you see is what
you get. When you want to stop outputting, you
must end with
.DISABLE DATA
```

The .DISABLE DATA must appear in column one, at the very left of your screen. If it is preceded by blanks or tabs it (and all of your command file that follows) will be taken as part of the data to be output to the terminal!

### 3.6.3 Quiet Mode -

If a command file contains the directive .ENABLE QUIET, any MCR commands and comments that follow will not be output to the terminal, although the MCR commands will be executed. Data using the terminal as a file, though, will still be output. Using .OPEN TI: is a good way to output to the terminal when QUIET mode is in effect. Enabling QUIET is a way of either hiding from the user exactly what the command file is doing, or of avoiding needless and meaningless output. Command files used by unsophisticated users should run in QUIET mode to avoid confusing the poor buggers with

complex system commands.

For example, if we took our MACRO command file above and added a few lines, we could eliminate all but the necessary output, like this:

```
Command File                Output
--------------------        ----------------------------
.ENABLE SUBSTITUTION
.ENABLE QUIET
.ASKS FILE File name                >*File name [S]: prog1
.OPEN TI:
.DATA 'FILE'.MAC assembly,          PROG1.MAC assembly,
.data task build and run.           task build and run.
.ASK YN Continue            >*Continue? [Y/N]: Y
.IFF YN .EXIT
MAC 'FILE'='FILE'
TKB 'FILE'='FILE'
RUN 'FILE'
.data 'FILE' run complete.  PROG1 run complete.
                            >@ <EOF>
```

There would be no blank lines in the output; I have merely attempted to show the output next to the line that causes it. I did not CLOSE the terminal because that happens automatically when the command file terminates, although it is good practice always to explicitly close files you open. The MAC, TKB and RUN commands are still issued to MCR, but they are issued silently, hidden from the terminal user.

## 3.7 Comments

A comment is a line or part of a line in a command file that is neither an indirect command nor an MCR command; it serves the purpose of annotating your command file. There are three basic types of comment: totally silent, quiet comments, and partial comments.

A totally silent comment begins with dot-semicolon (.;), e.g.:

```
.;  This line will never be output to the terminal
```

A quiet comment begins with a semicolon (;), and is affected by .ENABLE QUIET; it will be listed at the terminal if QUIET is disabled, and not listed if QUIET is enabled:

```
;  If QUIET is disabled this will type out.
```

In both totally silent and quiet comments the leading characters (either ".;" or ";") may be preceded by any number of blanks or tabs to allow indentation. In general in command files leading spaces are ignored.

A third type of comment can be a partial line added to a command line without affecting the command; the partial comment is set off from the command by an exclamation mark (!):

```
.SETN FLAG 0 !  Initialize FLAG
```

## 3.8 Conditional Execution

There are eleven conditional statements you can use in ICF's:

```
.IF comparison statement
    (Execute statement if comparison is true)
.IFT logical-symbol statement
    (Execute statement if symbol is true)
.IFF logical-symbol statement
    (Execute statement if symbol is false)
.IFDF symbol statement
    (Execute statement if symbol is defined)
.IFNDF symbol statement
    (Execute statement if symbol is not defined)
.IFINS taskname statement
    (Execute statement if task is installed)
.IFNINS taskname statement
    (Execute statement if task is not installed)
.IFLOA driver statement
    (Execute statement if driver is loaded)
.IFNLOA driver statement
    (Execute statement if driver is not loaded)
.IFACT taskname statement
    (Execute statement if task is active)
.IFNACT taskname statement
    (Execute statement if task is not active)
```

The basic use of conditionals is testing some condition and executing a statement if the condition is met. We've seen one example already:

```
.ASK YN Continue
.IFF YN .EXIT
```

Another example is:

```
.ASKS FILE Enter file name
.ASK YN List 'FILE'
.IFT YN PIP TI:='FILE'
```

Here's an example of testing whether a task is active and if so, waiting until it is done:

```
_; Assume another user is running a task called "PAYRL"
.SETF YN      ! Default to a NO answer.
```

```
.IFACT PAYRL .ASK YN PAYRL program in use; do you want to wait?
.IFT YN .WAIT PAYRL
.IFF YN .EXIT
```

The simple ".IF" takes a comparison test using one of the following operators to compare two symbols or a symbol with an expression:

```
GT or >    --   Greater than
LT or <    --   Less than
EQ or =    --   Equals
NE or <>   --   Not equal
GE or >=   --   Greater or equal
LE or <=   --   Less or equal
```

For example:

```
.IF X > Y .SETN Z X
.IF Z = X .EXIT
```

Conditionals may be strung together (all conditions listed must be true before the statement will execute):

```
.IF X = Y .IF Y = Z ;  X, Y and Z are all equal
```

They can also be separated by ".OR" to test of one or the other condition is true:

```
.IF X > Y .OR .IF Z > Y ;  either X or Z is GT Y
```

Two other directives are often involved in testing conditions: .TEST and .TESTFILE.

### 3.8.1 The .TEST Directive -

.TEST is used to determine certain information about variables; when executed, it sets several special symbols that can then be examined with an IF statement. The format is:

```
.TEST symbol
```

Testing a string symbol sets the following special symbols:

<SYMTYP> is set to 4
<STRLEN> is set to the length of the string
<ALPHAN> is set true if the string contains all alphanumeric characters
<NUMBER> is set true if the string contains all numeric characters
<OCTAL> is set true if the string contains all numeric characters in the range 0-7
<RAD50> is set true if the string contains all RAD50 characters

All of these except <SYMTYP> are also set by any .ASKS directive. The .SETS directive does NOT set these symbols.

Testing a numeric symbol sets the following symbols:

<SYMTYP> is set to 2
<OCTAL> is set true if the numeric symbol is octal

Testing a logical symbol sets the symbol <SYMTYP> to 0. No other symbols are set.

Here's an example of how it could be used:

```
.TEST STRING
.IFT <NUMBER> .SETN NUM 'STRING'
```

We test whether STRING contains all digits. If it does, we know it's safe to substitute the string into a .SETN statement. Otherwise .SETN would cause a syntax error.

Two idiosyncrasies I've found: a TEST of a string containing all alphabetic characters and blanks will set <ALPHAN> false because of the blanks (I guess that makes some sense); a TEST of a string containing "10." will set <OCTAL> true despite the presence of the decimal point. To me the latter is a bug.

A special form of .TEST can be used with string symbols or literals:

```
.TEST STRNG1 STRNG2
```

It locates the occurance in STRNG1 of STRNG2 and sets <STRLEN> to the character position at which STRNG2 starts, or to 0 if STRNG2 is not contained in STRNG1.

```
.TEST "ABCDE" "CD"
```

This sets <STRLEN> to 3.

Another example:

```
.ASKS DEV Enter device name
.SETN LEN <STRLEN>  ! Save length of entry
.TEST DEV ":"
.IF <STRLEN> NE LEN ; You forgot the colon
```

### 3.8.2 The .TESTFILE Directive -

The .TESTFILE statement sets a numeric symbol, <FILERR>, to the FCS error return code, and a string symbol, <FILSPC>, to the expanded file specification. A <FILERR> of 1 means that the file was found. Thus:

```
.TESTFILE LOGIN.CMD
.IF <FILERR> EQ 1 .GOTO GOTFIL
; LOGIN.CMD File was not found
.EXIT
.GOTFIL: ; Filespec is '<FILSPC>'
```

## 3.9 Labels, Branches And Loops

### 3.9.1 Labels -

Indirect statements may be labelled with any one to six character (alphanumeric plus dollar sign) identifier. A label begins with a dot and ends with a colon:

```
.EXAMP:  PIP /LI !  A labelled statement
```

Labels may be the same as other symbols or even Indirect commands:

```
        .SETS EXIT "For fun"
.EXIT: .EXIT ! Exiting 'EXIT'
```

A frequently referenced label should be put on a line by itself. Indirect then remembers its location for rapid action when it is referenced. Regular labels have to be found by scanning the entire file.

### 3.9.2 The GOTO Statement -

The .GOTO statement transfers control to a labelled statement:

```
        .ASK YN Are you done
        .IFT YN .GOTO DONE
```

```
        .;(Intervening statements)
.DONE:  .; Termination code follows
```

### 3.9.3 The GOSUB And RETURN Statements -

You can write subroutines in Indirect. This is discussed in more detail in the "Nifty Things to Do with Indirect Command Files" handout.

".GOSUB label" transfers control to the statement bearing the given label. When a ".RETURN" statement is encountered, control transfers back to the statement following the ".GOSUB".

```
        .GOSUB INIT
        .; (Intervening code)
        .EXIT

.INIT:  .; (Initialization code, for example)
        .RETURN
```

### 3.9.4 The EXIT And STOP Statements -

".STOP" does just that; it stops execution of Indirect whenever it is encountered. A slash (/) on a line by itself is shorthand for .STOP.

".EXIT" in a single command file does the same thing as .STOP. However, command files can be nested to four levels. That is, within a command file you can execute a second command file; from that a third, and from that a fourth. This is similar to calling external subroutines in a high-level language.

When you execute .EXIT from a second, third, or fourth level command file, it returns control to the previous level file at the statement following the call to the command file.

```
FILEA.CMD                   FILEB.CMD
-------------------------------------
;Statements                 ;Statements
@FILEB                        !
;Resumes here after          !
;FILEB executes .EXIT       .EXIT
```

### 3.9.5 Loops Using .INC Or .DEC -

.INC and .DEC increment or decrement numeric symbols, e.g.:

```
.SETN N 1
.INC N
; N now equals 2
```

Both .INC and .DEC can be used to write repetative loops as follows:

```
        .SETN N 1
.LOOP:  .IF N > 10. .GOTO END
        .; Intervening statements
        .INC N
        .GOTO LOOP
.END:   ; More stuff
```

The statements between LOOP and END will be executed ten times. See Appendix A for examples of structured programming constructs simulating DO/WHILE, FOR, and CASE using Indirect statements such as .INC and .DEC.

## 3.10 Opening Data Files

Text files may be opened for reading, writing, or appending using .OPENR, .OPEN, and .OPENA. (We have already seen .OPEN TI: which opened the terminal as an output data file.)

The .OPEN statement will open a new file for output. If a file of the same name already exists, a new version will be created. Form: ".OPEN filename". The statements .DATA and .ENABLE DATA may then be used to output to the file.

The .OPENR statement will open a file for reading. Successive .READ statements will read a record (delimited by a carriage-return/line-feed sequence) into a string variable, e.g. ".READ LINE". The symbol <EOF> is set true when a .READ fails to find a new record (end of file).

The .OPENA file will open an existing file and position at the end for appending additional records. If the file does not exist this behaves just like a .OPEN, that is, a new file is created.

Here is an example of reading through a file line by line, taking an action on each line, and testing for end of file. NOTMINE.CMD writes a directory to a temporary file and reports on any file in the directory whose owner UIC is not the same as the current UIC in which the file is found.

NOTMINE.CMD

```
.; This command file finds any files in the current UFD whose file
.; ownership differs from the current UIC.
.;
.; The version of SRD you have may produce output in different
.; format from ours, in which case the subscripts in the .SETS
.; FILNAM and .SETS OWNER commands may need to be altered. The same
.; is true if you use PIP /FU instead of SRD /FU. The values for
.; PIP are given as comments below. You will find that the SRD
.; version (if you have SRD) runs faster.

.ENABLE SUBSTITUTION,DECIMAL,QUIET
.OPEN _#1 TI:
.DATA _#1 LIST OF FILES IN '<UIC>' NOT OWNED BY '<UIC>'
.; Include next line to use PIP, delete SRD line
.;PIP TEMP.DIR=*.*;*/FU
SRD TEMP.DIR=*.*/FU
.SETN FOUND 0
.OPENR TEMP.DIR
.READ LINE ! Discard directory header
.; Include next two lines if using PIP
.;.READ LINE        ! Discard directory header for PIP
.;.READ LINE        ! Discard directory header for PIP
.LOOP:      .IFT <EOF> .GOTO DONE
   .READ LINE
   .SETS FILNAM LINE[1:19]
   .SETS OWNER LINE[43:51]
.; Delete 3 preceding lines and include following 4 lines for PIP
.; .READ LINE
.; .SETS FILNAM LINE[3:19]
.; .READ LINE
.; .SETS OWNER LINE[50:58]
   .SETS OWNER "'OWNER%C'" ! Strip trailing blanks
   .IF OWNER EQ <UIC> .GOTO LOOP
.; Include next line if using PIP
.; .IF OWNER EQ "" .GOTO LOOP ! Ignore total line
   .INC FOUND
   .DATA _#1 'FILNAM' owned by 'OWNER'
   .GOTO LOOP
.DONE:  .IF FOUND NE 0 .GOTO OUT
   .DATA _#1
   .DATA _#1 ALL FILES IN THIS UIC OWNED BY '<UIC>'
.OUT:       .DATA _#1 'FOUND' files found.
   .CLOSE
   .CLOSE _#1
   PIP TEMP.DIR;*/DE/NM
```

The example above uses two files. Indirect can handle up to four files. The default file is File #0; any other file references must be explicit, as in the ".OPEN #1 TI:" and ".DATA #1" statements above. If additional files were needed you would refer to them as "#2" and "#3".

## 3.11 Testing Command Files

Anybody starting out to write command files must learn to use two switches on the MCR command line: /TR (trace) and /-MCR (suppress MCR). These switches are invaluable for testing command files. They can prevent disasters from occurring during testing.

The /-MCR switch suppresses any actual execution of MCR (or DCL) commands contained in the file, but lists them as comments at your terminal so you can see if you have formatted and selected them correctly.

The /TR switch generates a trace at your terminal of all Indirect commands as they are executed -- you can see if your GOTO's and loops are executing as you expected them to. You will also see what substitutions Indirect has made into your command lines.

The switches are simply appended to the file spec when you invoke Indirect, e.g. "@MYFILE/TR/-MCR". The results look like this:

```
>@MOD/TR/-MC
>1!.GOSUB MOD 10 3 REM
>1!.enable substitution
>1!.PARSE COMMAN " " A B C
>1!.SETN REM 10-((10/3)*3)
>1!.RETURN
>; REM=2
>1!.exit
>@ <EOF>
```

The above is a trace run of the MOD command file presented in the next section.

The Trace and MCR modes can also be enabled and disabled within your command file with the ".ENABLE" and ".DISABLE" directives, e.g. ".ENABLE TRACE" or ".DISABLE MCR". This allows you to debug selected portions of your command file without tracing or disabling MCR for the entire file.

## 3.12 Expressions

So far we have presented data mainly as symbols and literals. But Indirect can also interpret and evaluate parenthesized expressions. Let's look at them by symbol type.

### 3.12.1 Logical Expressions -

The .SETL command can set a logical symbol to the value of a logical expression. Logical operators available are:

_! ... logical OR (inclusive)

_& ... logical AND
_# ... logical NOT

To set a symbol true if another symbol is false, we can say:

```
.SETL SINGLE #MARIED
```

To set a symbol true if one of several others is true we can say:

```
.SETL MALE MAN!BOY
```

To set a symbol true only if two or more other symbols are all true:

```
.SETL HAPPY HEALTH&WEALTH&WISDOM
```

The operators can be mixed. Also (see below) they can be used with numeric symbols to do bit manipulation.

### 3.12.2 Numeric Expressions -

All four basic arithmatic operators are available for numeric expressions:

```
+ ... addition
- ... subtraction
* ... multiplication
/ ... division
```

Parentheses can be used to control precedence; unless parenthesized, expressions are evaluated strictly left-to-right regardless of the operations being performed. There can be no spaces in expressions. Numeric values are limited to integers in the range of 0 to 65535 decimal or -32767 to +32768 in signed representation (octal 0 to 177777). Division trucates to an integer result, that is, 10./3 results in 3.

```
.SETN A 2.
.SETN B 3.
.SETN C A+B*4.
```

The above sets C to 20 decimal.

```
.SETN A 2.
.SETN B 3.
.SETN C A+(B*4.)
```

sets C to 14 decimal.

Here is a subroutine to calculate the remainder of a division (modulo), based on the fact that division truncates to an integer:

```
.MOD:  .; Subroutine to calculate C=MOD(A,B)
       .; C = symbol name to receive result
       .; A is the dividend, symbol or literal
       .; B is the divisor, symbol or literal
       .; Calling sequence: .GOSUB MOD A B C,
       .; for example: .GOSUB MOD 10. 3 REM
       .enable substitution
       .PARSE COMMAN " " A B C
       .SETN 'C' 'A'-(('A'/'B')*'B')
       .RETURN
```

For more information on subroutines and parameters, see the "Nifty Things" handout.

Numeric expressions are permitted as second operands in .IF and .SETN directives. They are also permitted as range and default arguments in .ASKN and .ASKS directives and (not mentioned in the MCR manual) as substring range specifiers in a string expression (see next section).

The restriction of expressions to the second operand in .IF statements means that ".IF A+B <= 25." is illegal; you must say:

```
.SETN TEST 25.
.IF TEST > A+B
```

The logical operators mentioned under "Logical Expressions" can also be used with numbers.

```
.SETN A 000011
.SETN B 000007
.SETN C A_&B   ! Results in 000001
.SETN C A!B    ! Results in 000017
.SETN C _#A    ! Results in 177766
```

### 3.12.3 String Expressions -

String expressions are limited to two operations in .SETS and .IF directives: concatenation and substring extraction. These two operations, however, when combined with numeric expressions and the .TEST directive to set <STRLEN> (string length), provide all you need for very sophisticated string manipulation.

Concatenation (represented by the + operator) appends one string to another:

```
.SETS S1 "A"
.SETS S2 "CDEF"
.SETS S3 S1+"B"+S2     ! S3 will equal "ABCDEF"
```

As with numeric expressions, string expressions are limited to the second operand in .IF statements:

```
.SETS SYSDEV "DB4"
.ENABLE SUBSTITUTION
.ASKS [2:2] TYPE Enter 2 character device type
.ASKN NUM Enter unit number
.IF SYSDEV = TYPE+"'NUM'" .GOTO NONO
```

Here we have used substitution to convert NUM, a numeric symbol, to a string. The resulting string is then enclosed in double quotes to make it a string literal that can be used in the string expression.

Substring extraction is accomplished by adding subscript-like references to a string symbol. Thus, if S1="ABCDEF", then S1[2:4] = "BCD". The number used as subscripts may be numeric symbols, literals or numeric expressions.

Assume we have asked the user to enter a disk device name of four characters in the form "DBn:" and we wish to extract the unit number "n". We know it is the third character. Normally it isn't wise to assume the user will enter data correctly and we should protect against errors, but for the sake of the example we'll assume he enters it right:

```
.ASKS DEV [4:4] Enter 4 character device name
.SETS UNIT DEV[3:3]
```

The range specifier "[4:4]" in the .ASKS limits the response to four characters, no more no less. If we want to allow two-digit unit numbers we have to get more sophisticated, using the <STRLEN> symbol which is set automatically by the .ASKS.

```
.ASKS DEV [4:5] Enter 4 or 5 character device name
.IF <STRLEN> = 4 .SETS UNIT DEV[3:3]
.IF <STRLEN> = 5 .SETS UNIT DEV[3:4]
```

See the "Nifty Things" handout for more on string handling and the .TEST directive.

## 4.0 CONCLUSION

This presentation has only been able to scratch the surface of the potential and power in RSX Indirect Command Files. I hope it has given you, if nothing else, an incentive to study -- not just read -- the chapter on Indirect in the MCR manual.

I have been writing command files for over three years, and even in preparing for this session I have discovered new things I could do. The benefits to your installation, to say nothing of your own personal convenience, will more than repay continual study and experimentation.

If you'd like some examples of advanced command files, there will be many submitted to the RSX SIG tape from the BERGEN RECORD and other sites. Ours will probably be found in group 333 on the tape. They are far too lengthly to include in this handout. You can also learn a great deal by studying the SYSGEN command files that come on your distribution kit.

## APPENDIX A
## SIMULATION OF STRUCTURED CONTROL FLOW WITH IND

The Indirect Processor does not have a full set of flow control commands such as is found in modern structured languages like Pascal, Ada, and C. It is limited to IF, GOTO, and GOSUB. I have found it useful, however, to program complex command files in a pseudo-language using the structured control flow commands found in C and then translate into indirect command language. The translation templates given here should aid anyone else who wishes to do this.

1. The WHILE statement

   The WHILE statement follows this basic structure in C:

```
WHILE (expression)
  statement (or statement block);
```

   It can be simulated in an indirect command file as follows:

```
.LABEL: .IF (expression is false) .GOTO END
        statement(s)
        .GOTO LABEL
.END:   ...
```

   The nature of the test expression must be reversed; that is, if the WHILE would be written "WHILE( A > B)", the .IF must be written, ".IF A <= B".

2. DO/WHILE statement

In C, the DO/WHILE (similar to Pascal's REPEAT UNTIL) follows this form:

```
DO
  statement or block
WHILE (expression);
```

In an indirect command file this can be simulated by:

```
.LABEL: statement(s)
        .IF (expression) .GOTO LABEL
```

In this case, the nature of the test expression is the same in both forms; that is, if testing for "A > B" in the DO/WHILE, you perform the same test in indirect commands.

3. THE FOR LOOP

In C a FOR loop takes this form:

```
FOR( expression1; expression2; expression3)
  statement or statement block;
```

Expression1 is the initialization expression, expression2 is the relational comparison test, and expression3 is (usually) the increment of a control variable. For example: "for(i=0; i < 20; i=i+1)".

In an indirect command file this is simulated as follows:

```
        expression1
.LABEL: .IF (expression2 is false) .GOTO END
        statement(s)
        expression3
        .GOTO LABEL
.END:   ...
```

Translating the simple loop controlled by "I" to perform a set of statements 20 times, we have:

```
        .SETN I 0.
.LABEL:
        .IF ( I >= 20.) .GOTO END
        statement(s)
        .INC I
        .GOTO LABEL
.END:   ...
```

In any command file with a frequently referenced label, such as "LABEL" above, I highly recommend you place the label on a line by itself. This GREATLY speeds execution of the file; try

your own timings and you'll see what I mean.

4.

THE CASE STATEMENT

A "case" statement in a structured language handles the situation where you have a multi-level decision with only one executable path. A typical example is a menu-selection, where you have the user respond with one of several answers and execute a block of code depending on his answer.

Unfortunately, the Indirect Command Processor does not support IF-ELSE, which is the simplest way most programming languages offer to simulate "case". There is no "ELSE" directive in IND. Indirect does have a BEGIN/END block command which would seem useful; however, all that BEGIN/END does is control the local definition of symbols. It cannot be used to enclose a block of statements for conditional execution. The manual specifically states that ".IF expression .BEGIN" is illegal.

I have found a simple format using indirect commands that serves the "case" situation. It involves simply substituting the input command into a ".GOTO" statement. This allows labelling the block of code with the actual command mnemonic, and is (I think) rather elegant.

What follows is a segment of code from the file DEBUG.CMD on the last SIG tape.

```
.enable substitution
.C1: .OPEN TI:

.MENU:.enable data
DEBUGGER OPTION LIST
DUMP  -- Dump entire symbol table
EXIT  -- Exit from DEBUG back to caller
GOTO  -- Resume execution at a certain label
LIST  -- List current file (or other)
MCR   -- Toggle ENABLE/DISABLE MCR
MENU  -- Display this menu
QUIE  -- Toggle ENABLE/DISABLE QUIET
RETN  -- Return from a GOSUB call to DEBUG.
SETF  -- Set logical symbol false
SETN  -- Set value of numeric symbol
SETS  -- Set value of string symbol
SETT  -- Set logical symbol true
SPEC  -- Dump special symbols
STOP  -- Stop execution, return to MCR (or usual CLI)
SYMB  -- Display a local symbol value
TRAC  -- Toggle ENABLE/DISABLE TRACE

.disable data

.GETOPT: .asks OPT Enter option
```

```
        .ONERR NOSUCH
        .GOTO 'OPT'
.NOSUCH: .data No such command as 'OPT'
        .GOTO MENU
.; Blocks of code occur here for each valid command
.;
```

This example makes use of the ability to "trap" errors such as invalid GOTO's. The remainder of the file has blocks of code with labels corresponding to the options listed in the menu: "DUMP", "EXIT", "GOTO", and so on. When the user types in an option name, the ONERR statement tells the ICP where to go if an error (a GOTO to a non-existent label) occurs -- in this case, to the label NOSUCH, which issues a "No such command" message and goes back to output the menu of commands again.

If the option typed in is valid, we simply GOTO the label of the same name. There, we execute whatever the option calls for, and at the end, issue a ".GOTO MENU" to return to the option list display. It is good practice to start each block of code with a null ".ONERR" statement (one with no operand), so that any other errors that occur will be treated normally by Indirect.

While this is not elementary indirect command file coding, it is fairly simple and easy to understand.

## APPENDIX B
### Index to Online Help for Indirect

```
Help file qualifiers for: MCR INDIRECT
rootfile is LB:[1,2]MCR.HLP
1 INDIRECT
    2 OPERATORS
    2 SUMMARY
    2 .ASK(see ASK)
    2 ASK
    2 .ASKN(see ASKN)
    2 ASKN
    2 .ASKS(see ASKS)
    2 ASKS
    2 .BEGIN(see BEGIN)
    2 BEGIN
    2 .CHAIN(see CHAIN)
    2 CHAIN
    2 .CLOSE(see CLOSE)
    2 CLOSE
    2 .DATA(see DATA)
    2 DATA
    2 .END(see END)
    2 END
    2 .EXIT(see EXIT)
    2 .IF(see IF)
    2 IF
    2 .INC(see INC)
    2 INC
    2 .LABEL:
    2 LABEL:
    2 .ONERR(see ONERR)
    2 ONERR
    2 .OPEN(see OPEN)
    2 OPEN
    2 .PARSE(see PARSE)
    2 PARSE
      3 EXAMPLE
    2 .PAUSE(see PAUSE)
    2 PAUSE
    2 .READ(see READ)
    2 READ
      3 DETAILS
      3 EXAMPLE
```

```
2 EXIT
2 .FORM(see FORM)
2 FORM
  3 COMMANDS
  3 SUMMARY
2 .DISABLE(see ENABLE)
2 DISABLE(see ENABLE)
2 .ENABLE(see ENABLE)
2 ENABLE
2 .DEC(see DEC)
2 DEC
2 .DELAY(see DELAY)
2 DELAY
2 .ERASE(see ERASE)
2 ERASE
2 .GOTO(see GOTO)
2 GOTO
2 .GOSUB(see GOSUB)
2 GOSUB
2 XQT
2 ACCOUN
2 ALPHAN
2 ALTMOD
2 BASLIN
2 CLI
2 CONFIG
2 DATE
2 DEFAUL
2 EOF
2 ERRCTL
2 ERRNUM
2 SEVERE(see ERROR )
2 SUCCES(see ERROR)
2 WARNIN(see ERROR)
2 ERROR
2 ERRSEV
2 ERSEEN
2 ESCAPE
2 EXSTAT
2 EXSTRI
2 FALSE
2 FILERR
2 FILER2
2 FILATR
2 FILSPC
2 FORATT
2 FMASK
2 IAS(see RSX11D)
2 LIBUIC
2 LOCAL
2 LOGDEV
2 LOGUIC
2 MAPPED
2 MEMSIZ
2 NETUIC
2 .RETURN(see RETURN)
2 RETURN
2 .STOP(see STOP)
2 STOP
2 .SET(see SET)
2 SET
2 .TEST(see TEST)
2 TEST
2 .TESTDEVICE(see TESTD
2 TESTDEVICE
2 .TESTPARTITION(see TE
2 TESTPARTITION
2 .TESTFILE(see TESTFIL
2 TESTFILE
  3 EXAMPLES
2 FILE
2 .WAIT(see WAIT)
2 WAIT
2 .XQT(see XQT)
2 NUMBER
2 NXTSYM
2 OCTAL
2 PRIVIL
2 RAD50
2 RSX11D
2 SPACE
2 STRLEN
2 SYDISK
2 SYMTYP
2 SYSDEV
2 SYSID
2 SYSTEM
2 SYSUIC
2 SYUNIT
2 TIME
2 TICLPP
2 TICWID
2 TIMOUT
2 TITYPE
2 TISPED
2 TRUE
2 UIC
2 P0(see PARAMETER)
2 P1(see PARAMETER)
2 P2(see PARAMETER)
2 P3(see PARAMETER)
2 P4(see PARAMETER)
2 P5(see PARAMETER)
2 P6(see PARAMETER)
2 P7(see PARAMETER)
2 P8(see PARAMETER)
2 P9(see PARAMETER)
2 COMMAN(see PARAMETER)
2 PARAMETER
2 ADVANCED
```

# SRD V6.4

SRD Working Group
Bob Turkelson
NASA/Goddard Space Flight Center
Mail Code 933
Greenbelt, MD 20771
(301) 344-5003

The SRD Working Group submitted a revised version of SRD to the Fall 1983 RSX SIG Tape. SRD V6.4 appears in [352,4]. Many of the important items on our wish-list were completed for this release. We are continuing to modify the program, documentation, and help files for the next SIG tape. If you would like to share any modification you have made, if you have any suggestions, or if you would like to help in any area, please contact us.

The README.1ST file describes the modifications made since SRD V6.3 (Spring 1983). They are:

o Fixed a problem which caused SRD to abort when listing large directories whose contents could not all fit into memory. (I also tried several other versions of SRD from recent SIG tapes and found that they all had the same problem.) When built on RSX-11M V4.0 with dynamic checkpoint allocation, SRD is able to read in a dense directory containing 3360 entriesbefore having to list the directory in sections (2976 entries if built with FCSRES).

o Fixed a problem which did not allow SRD to do any listings or selections based upon information contained in the file header if the entire directory did not all into memory (that is, when SRD needs to separate the directory contents into pieces).

o Fixed a problem where instead of cancelling directory write- back for directories which could not all fit into memory, SRD wrote back only the last section of it, wiping out most of the directory! This problemprobably only showed up when SRD was built without the dynamic checkpoint allocation option, for otherwise it would have

aborted anyway due to the error described above in the first modification. Now SRD issues a fatal message informing you that write-back will not be done (and SRD means it). This message is issued before any directory sorting is attempted, so you don't have to wait for sorting the large segment of the directory which did fit into memory.

o Fixed a memory protection violation problem than occurred only occasionally with some configurations when listing a dense directory which exactly fills the last block of the directory.

o Improved the organization and contents of the documentation to reflect the current state of SRD, while converting to a true RUNOFF format. This work was done by Gerard Stackhouse, who is maintaining the documentation. Any suggestions for improvements may be sent to him at United Telecom Computer Group, United Information Services, Inc., 300 Second Avenue, Waltham, MA 02154 (617-890-6820).

o Modified SRD so that it displays the original command line the user entered, before replacement of certain special symbols which may appear in the /SE, /SD, and date switches.

o Added new capabilities in specifying the sorting to be done. Sorting by date may now be requested. SRD allows the major and minor sort keys (file name, type, version, date) to be specified in any order in the /SR:x:x:x switch, and also allows the sort to be either ascending or descending sort for any key independently. (To see an unusual looking directory listing, try using Version as the primary key.Or try your own combination.) The sorting by date feature was merged from Henry Tumblin's version of SRD (which is not on the SIG tapes). Allowing sorting to be done ascending or descending was a feature adapted from changes by Dave Sides (Sachs/Freeman Assoc., Inc., c/o JHU/Applied Physics Laboratory), and from the SRD in the U. S. Forest Service collection of programs submitted to the Spring 1982 SIG tape. Ideas for being able to specify the major and minor keys also came from the latter version, although the V6.4 implementation is quite a bit different. Coding changes were required in the sections of the selection/listing routine dealing with the highest and obsolete version selection switches (/SV, /NV, /OV, /PU) since files may be sorted with ascending version numbers. The former code compared each entry in the sorted directory with both the previous entry and the following entry. The current code looks only forward and remembers where it finds a different file name.

o Added the /CM:xxx:fffff switch to enable the generation of command files. This feature was merged from the version submitted in the U. S. Forest Service program collection cited above (eliminating a problem that version had in handling null file

names). The format and meaning of the switch values are taken from that version. For details see the documentation or help files; the information given there is also from the the earlier program.

o Added the /PA switch to pack the files names (removing any blanks between the name and type and between the type and version number).

o Added the /TB switch to include in the summary the number of blocks used/allocated even if it would not otherwise appear. For example, /OV/-LI shows only the number of obsolete files, not the total number of blocks. /OV/-LI/TB does show the block total.

o Added the /GT:n switch to select files with allocated sizes of at least n. blocks. This is useful for finding large files on user disk packs, as well as easily determining large directory sizes. It may also be useful in obtaining some contiguous disk space. Use /GT and /BE (and perhaps /-SR) to search for large files which existed at the time of the last full disk copy. Rename one such file, then copy it to the original name using PIP /CD (assuming enough free space remains). Then delete the renamed file, which will usually free up contiguous blocks.

o Added the /FI:n switch to select the file with File ID n (the sequence number should not be specified). This helps to find a file when a utility such as BRU issues a diagnostic containing the File ID without a file name, or reports the file name from the header, but the file has been renamed. Up to four File IDs may be specified: /FI:n1:n2:n3:n4.

o Added the /FO:[g,m] switch to select files with file owner [g,m]. Either the group or member may be specified as a wildcard (* or zero). The UIC of the directory being listed will be the default switch value if /FO is specified without group and member. An alternate form of the switch, /FO:g:m, is supported (but a wildcard must be a zero, not *). The switch may be negated (/-FO:[g,m]) to select files not owned by [g,m]. SRD [*,*]/-FO would list all files not owned by the directory in which they reside.

o Added the /ER switch to display files which produce errors when attempting to read their headers.

o Added the /NV (new version) switch with the same meaning as /SV. Some other versions of SRD use /NV, and this seems to be a more appropriate name.

o Changed the listing of a file's used and allocated blocks to reference the double word values stored in the header instead of single word values. In V6.3 the number of allocated blocks was calculated from the retrieval pointers if the file header showed a value of zero. Now this is also done if the file has multiple headers, which requires SRD to read each header. NOTE: There is a problem in RSX-11M V4.0 and RSX-11M-PLUS V2.0 which does not allow SRD to read an extension header unless your "login UIC" is a privileged UIC or the file owner's UIC (actually the file owner stored in the extension header). To get around this temporarily, SRD ignores such extension header read errors and treats the file as if it were not a multi-header one. DMP shows the same problem. The undocumented SRD switch /H2 inhibits this work around and causes the error message to be displayed - this is for testing purposes. We have learned that this does not show up in RSX-11M V3.2 or V4.1. For our next version we probably will obtain the number of allocated blocks from the statistics block. Until recently we felt there were reasons we could not use the statistics block.

o Corrected handling the case when the date stored in the file header is corrupt, by issuing a non-fatal diagnostic message. Previously the same fatal message was issued for this situation as was given for when an invalid date was specified in a date switch in the command line.

o Modified the program to use the three letters of the task name in the program prompt and in the error messages.

o Modified SRD to make sure /SR (sort switch) is set if /SV, /NV, /OV, or /PU is specified, even if /-SR is specified. The directory needs to be sorted for these functions.

o Changed the diagnostic processing so that the UIC and file name are included in the appropriate messages.

o Changed the listing routine so that a file with a header read error will never be "selected" (as long as a switch which requires reading headers is specified, of course). In addition, if file selection by date is requested and a corrupt date is detected in the file header, that file will not be "selected."

o Modified SRD to assure that the UIC heading line for the directory appears in the listing whenever issuing a diagnostic for a header read error or for a corrupt date in the header.

o Correctly terminated each switch value table so that if too many values are specified for a particular switch, a syntax error message is given. With previous versions, extra values on certain switches would clobber fields for other switches. For example, SRD /SV/MI:80:3 would behave as if SRD /SV:3/MI:80 had be specified.

o Made a few changes to enable this version of SRD to run on IAS systems. Recently I traced the failure to the method employed by SRD to reuse some of the memory taken by the largest overlay segment (SRDINI) once the routines in that segment have completed. IAS V3.0 is running on the system I was able to use. The current code seems to work OK(testing just a few switches) if SRD is built /-MU and the special dummy PSECTs $$$XX1, $$$XX2, and $$$XX3 are given the RO attribute for IAS systems, so these modifications are in this distribution. Of course this means that the resulting task is not multi-user. By the next SIG tape we plan to get it working as a multi-user task under IAS; there is no time left to make and test such changes before this submission. There were several modifications in the command file for IAS: eliminating certain SYSLIB routine references from the generated ODL file; and, jumping over any the references to the <DATE> symbol (which is not defined in the version of Indirect on the IAS system I used).

o Eliminated the extra 0 byte appearing at the end of the UIC line.

o Eliminated the extra 0 byte appearing at the end of a diagnostic message.

o Made a few other internal coding changes, documented in the source files.

o Kept the help file up to date, added a HELP SRD NEW section to list new switches in V6.4, and changed the HELP SRD SWITCH DEFAULTS section so that it references a file created by the SRD generation command file.

o Modified the command file to allow the user to select certain groups of default switches, to eliminate the need to ask as many questions.

```
>; An example of building SRD V6.4 follows:
>
>@srd
>;
>;       Question/Answer phase
>;
>;       To get help hit ESCAPE from any prompt.
>;
>*       Are you using RSX11M+? [Y/N]:
```

```
>*      Are you using RSX-11M, MAPPED? [Y/N]: y
>*      Are you running RSX-11M Version 4.0 or later? [Y/N]: y
>*      Do you have DCL or User-written CLI support? [Y/N]: y
>*      Do you have the extend task directive? [Y/N]: y
>*      Are you using dynamic checkpoint allocation? [Y/N]: y
>*      Are you using FCSRES? [Y/N]:
>*      Does your machine have EIS? [Y/N]: y
>;
>;      Creating Switch Defaults
>;
>; The next few questions are for users who are familiar with the
>;   available SRD switches.  Some of the switches are presented
>;   in groups, allowing you to choose the entire group as a
>;   default.  Simply answer N if you do not want the entire
>;   group - questions for the individual switches will then be
>;   presented later.
>;
>; Do you want these switch defaults:
>*      /LI  /SR  /-WD  /-OO? [Y/N]: y
>*      /AT  /M2  /WI       ? [Y/N]: y
>*      /BK  /HD  /SM       ? [Y/N]: y
>*      /-NA /-RD           ? [Y/N]: y
>;
>; In the following section, you will be asked which switches you
>;   would like to have as the default.  Help is also available
>;   at these prompts by typing the ESCAPE key.  Answer Y for any
>;   switch default you would like.
>;
>;   As a default, would you like switch ...
>*      /SY - System Accounts in [*,*] searches? [Y/N]: y
>;
>;      Creating SRDDEF.HLP
>;
>;
>*      Q/A complete, do you want to continue? [Y/N]: y
>;      Creating SRD.BLD
>;      Creating SRD.ODL
>;      Creating SRDSYS.MAC
>*      Command files generated, Continue? [Y/N]: y
>;
>;      Assemble the source modules
>;
>* Do you want to assemble all the modules? [Y/N]: y
>MAC SRDATA=SRDSYS,SRDPRE,SRDATA
>MAC SRDINI=SRDSYS/PA:1,SRDPRE/PA:1,SRDINI
>MAC SRDLST=SRDSYS/PA:1,SRDPRE/PA:1,SRDLST
>MAC SRDNUD=SRDSYS/PA:1,SRDPRE/PA:1,SRDNUD
>MAC SRDOPR=SRDSYS/PA:1,SRDPRE/PA:1,SRDOPR
>MAC SRDROT=SRDSYS/PA:1,SRDPRE/PA:1,SRDROT
>MAC SRDSRT=SRDSYS/PA:1,SRDPRE/PA:1,SRDSRT
>MAC SRDSUB=SRDSYS/PA:1,SRDPRE/PA:1,SRDSUB
>MAC SRDTST=SRDSYS/PA:1,SRDPRE/PA:1,SRDTST
>MAC SRDTRP=SRDSYS/PA:1,SRDPRE/PA:1,SRDTRP
>MAC SRDREP=SRDSYS/PA:1,SRDPRE/PA:1,SRDREP
>MAC SRDDBF=SRDSYS/PA:1,SRDPRE/PA:1,SRDDBF
```

```
>MAC SRDXX1=SRDSYS/PA:1,SRDPRE/PA:1,SRDXX1
>MAC SRDXX2=SRDSYS/PA:1,SRDPRE/PA:1,SRDXX2
>MAC SRDXX3=SRDSYS/PA:1,SRDPRE/PA:1,SRDXX3
>;
>;      Now taskbuild SRD
>;
>TKB @SRD.BLD
>;
>;      SRD Build complete
>;
>@ <EOF>
>
>; Make sure that your master help file(s) point to the SRD help
>;      file:
>uic 1 2
>edt mcr.hlp
    1           @MCRGENRL
*in 1.1
                1 SRD
                @[1,222]SRD
                ^Z
    2           1 LOCAL
*EX
DB0:[1,2]MCR.HLP;15 595 lines
>
>; We place our local help files in [1,222].  [1,2] or any other
>;   UIC would
>;   be fine.
>uic 1 222
>pip /cd/nv=[352,4]srd.hlp,srddef.hlp
>
>; Rename (or copy) the task into the system directory:
>uic 1 54
>pip [1,54]/nv/re=[352,4]srd.tsk
>
>; Install the new version:
>rem srd
>ins $srd/pri=55.
>
>srd /id
SRD -- Version WG-6.4, November 1983

>; Display parts of the help file:
>help srd
 SRD is a do-all directory utility.  There is so much to know
 about it that you SHOULD read the SRD V6.4 User's Manual
 before using it. Type HELP SRD NEW for switches which are new
 in V6.4.  Available switches:

 /00    Wild UFD [0,0]
 /AE:xx Select on/after date(/time)
 /AF:xx Select after date(/time)
 /AT    Attach output dev if TTY
 /BE:xx Select on/before date(/time)
 /BF:xx Select before date(/time)
```

```
 /BK    File names preceded by 2 blanks
 /CM:xx:xx Generate command file format
 /BK    File names preceded by 2 blanks
 /CO    Select Contiguous Files
 /DA:xx Select on date(/time)
 /DE    Delete Files
 /ER    Files with header read errors
 /FI:n  Select file with File ID n
 /FO:[g,m] Select files by file owner
 /FU:n  Full Directory
 /GT:n  Select files of size >= n
 /HD    Cmd line shown in Header
 /HE    Give brief help
 /HV:n  Select by Version Number
 /ID    Print current Version of SRD
 /LI:n  Brief directory
 /LO    Select Locked Files
 /M2    Type 2 middle directory
 /MI:n  Middle Directory (2 types)
 /MU    Select Multi-Header Files
 /NA    Sort by name first
 /NE    Select Files not Specified
 /NV:n  Select most recent versions
 /OV:n  Select Obsolete Versions
 /PA    Pack file names (remove blanks)
 /PU:n  Purge the directory
 /PA    Pack file names (remove blanks)
 /RD    Use Revision Date
 /SD:xx Selectively Delete Files
 /SE:xx Use special wildcards
 /SI    Print available Buffer Size
 /SM    Include Summary lines
 /SP    Spool the Output File
 /SR:x:x:x Sort the Directory
 /SV:n  Select Most Recent Versions
 /SY    Wild system UFD's
 /TB    Include block total in summary
 /WB    Write Back the Directory
 /WD    Implicit wild delete
 /WI    Implicit wildcard
 /ZE    Select zero-blocks-used files

 For help on a given switch type HELP SRD <switch>
 For help on switch defaults type HELP SRD SWITCH DEFAULTS
     or HELP SRD DEFAULTS
>
>help srd switch defaults

 The following SRD switch defaults apply on this system:

 /-NA - sort by file type first, then by file name
  /LI - list the selected file names
 /-RD - use creation date in date selection and in /MI/M2 listing
  /SR - sort the file names before listing
  /SY - include system UFD's (group 1-10) in [*,*] and [*,n]
```

```
        searches
 /-OO - do not include [0,0] in [*,*] searches
  /AT - attach output device if a terminal
  /WI - implicit wildcarding
 /-WD - no implicit wildcarding during delete operations
  /M2 - use type 2 middle directory listing when /MI specified
  /HD - display command line in the heading
  /SM - include summary lines following the file listing
  /BK - precede each file name with two blank characters

>help srd new

 New SRD switches for V6.4 are:

 /CM:xx:xx Generate command file format
 /ER       Files with header read errors
 /FI:n     Select file with File ID n
 /FO:[g,m] Select files by file owner
 /GT:n     Select files of size >= n
 /NV:n     Select most recent versions
 /PA       Pack file names (remove blanks)
 /SR:x:x:x Sort the Directory
 /TB       Include block total in summary

 Type HELP SRD <switch> for details on any switch.

>
>; Copy the documentation file to the appropriate account:
>
>pip db3:[373,101]/nv/cd=[352,4]srd.doc,.rno
>
>;Example of SRD.CMD with answers to each switch default question:
>
>uic 352 4
>
>@srd
>;
>;       Question/Answer phase
>;
>;       To get help hit ESCAPE from any prompt.
>;
>*       Are you using RSX11M+? [Y/N]:
>*       Are you using RSX-11M, MAPPED? [Y/N]: y
>*       Are you running RSX-11M Version 4.0 or later? [Y/N]: y
>*       Do you have DCL or User-written CLI support? [Y/N]: y
>*       Do you have the extend task directive? [Y/N]: y
>*       Are you using dynamic checkpoint allocation? [Y/N]: y
>*       Are you using FCSRES? [Y/N]:
>*       Does your machine have EIS? [Y/N]: y
>;
>;       Creating Switch Defaults
>;
>; The next few questions are for users who are familiar with
>;   the available SRD switches.  Some of the switches are
>;   presented in groups, allowing you to choose the entire
```

```
>;   group as a default.  Simply answer N if you do not want
>;   the entire group - questions for the individual switches
>;   will then be presented later.
>;
>; Do you want these switch defaults:
>*      /LI  /SR  /-WD  /-00? [Y/N]:
>*      /AT  /M2  /WI       ? [Y/N]:
>*      /BK  /HD  /SM       ? [Y/N]:
>*      /-NA /-RD           ? [Y/N]:
>;
>; In the following section, you will be asked which switches
>;   you would like to have as the default.  Help is also
>;   available at these prompts by typing the ESCAPE key.
>;   Answer Y for any switch default you would like.
>;
>;   As a default, would you like switch ...
>*      /NA - the sort to be in name order? [Y/N]:
>*      /LI - the directory to be listed by default? [Y/N]: y
>*      /RD - to use the Revision Date in Date Matches? [Y/N]:
>*      /SR - the listings sorted by default? [Y/N]: y
>*      /SY - System Accounts in [*,*] searches? [Y/N]: y
>*      /00 - to include [0,0] in [*,*] searches? [Y/N]:
>*      /AT - output device attached if terminal? [Y/N]: y
>*      /WI - implicit wildcarding? [Y/N]: y
>*      /WD - Do you REALLY want implicit wildcarding during
>             delete? [Y/N]:
>*      /M2 - the Type 2 Middle Size Listing? [Y/N]: y
>*      /HD - the command line to appear in the header? [Y/N]: y
>*      /SM - the summary listing by default? [Y/N]: y
>*      /BK - two blanks preceding file names listed? [Y/N]: y
>;
>;      Creating SRDDEF.HLP
>;
>;
>*      Q/A complete, do you want to continue? [Y/N]: y
>;      Creating SRD.BLD
>;      Creating SRD.ODL
>;      Creating SRDSYS.MAC
>*      Command files generated, Continue? [Y/N]: y
>;
>;      Assemble the source modules
>;
>* Do you want to assemble all the modules? [Y/N]: y
>MAC SRDATA=SRDSYS,SRDPRE,SRDATA
>MAC SRDINI=SRDSYS/PA:1,SRDPRE/PA:1,SRDINI
>MAC SRDLST=SRDSYS/PA:1,SRDPRE/PA:1,SRDLST
>MAC SRDNUD=SRDSYS/PA:1,SRDPRE/PA:1,SRDNUD
>MAC SRDOPR=SRDSYS/PA:1,SRDPRE/PA:1,SRDOPR
>MAC SRDROT=SRDSYS/PA:1,SRDPRE/PA:1,SRDROT
>MAC SRDSRT=SRDSYS/PA:1,SRDPRE/PA:1,SRDSRT
>MAC SRDSUB=SRDSYS/PA:1,SRDPRE/PA:1,SRDSUB
>MAC SRDTST=SRDSYS/PA:1,SRDPRE/PA:1,SRDTST
>MAC SRDTRP=SRDSYS/PA:1,SRDPRE/PA:1,SRDTRP
>MAC SRDREP=SRDSYS/PA:1,SRDPRE/PA:1,SRDREP
>MAC SRDDBF=SRDSYS/PA:1,SRDPRE/PA:1,SRDDBF
```

```
>MAC SRDXX1=SRDSYS/PA:1,SRDPRE/PA:1,SRDXX1
>MAC SRDXX2=SRDSYS/PA:1,SRDPRE/PA:1,SRDXX2
>MAC SRDXX3=SRDSYS/PA:1,SRDPRE/PA:1,SRDXX3
>;
>;       Now taskbuild SRD
>;
>TKB @SRD.BLD
>;
>;       SRD Build complete
>;
>@ <EOF>
```

Editor, The Multi-Tasker
c/o DECUS
1 Iron Way
MR2-3/E55
Marlboro, MA 01752

## XONing a XOFFed terminal

We have an 11/44 doing Office Automation for about 50 users. There are 10 Letter Quality printers attached via terminal lines (TT:). The problem we had was that our printers have no keyboards and the 11/44 would see a XOFF (transmitter off) and stop sending output to the printer. We now had to tell the 11/44 to ignore the fact that the printer told it to stop, if we wanted any further output from the printer. RSX-11M provides no easy solution to this problem. We could shut the system down and reboot (the users would just love this). We could pull out system maps and go poking arround with OPEn (a real safe and secure method). Or, we could try to write a program to reset the state of the terminal. After a little digging, we found that the bit S1.CTS in the terminal UCB was the culprit. Also digging into the RSX SIG tapes we found a program that does this for RSTS. We modified the RSTS version to work under RSX-11M V4.0 and to reset the propper bit. We now awaited our chance to test this new piece of code. When our chance came, we used it and the printer just sat idle. Using OPE, we found that the bit was set propperly but there was no output. Telephone support in Colorado Springs proved very helpfull in our new dilema. We were told that our method was correct but lacking one final item -- we must recycle the Terminal Driver. The suggested method was to do a null break-through write (IO.WBT) to the device after reseting the bit. We added this little extra code and everything worked fine.

We have been using this program about once every week for the last 18 months with no problems. I am enclosing the Macro source and Indirect command files for anyone who wishes, wants or needs a program like this.

Gerry Van Trieste
RCA Corporate Staff
Mini Computer Systems Support
Cherry Hill, NJ
(609) 338-6025

---

```
.; XON.CMD
.;
.; XON  -- Assemble and build XON task
.;
   .ENABLE SUBSTITUTION
   .SETT INSMAC
   .SETT INSTKB
   .IFNINS MAC      .SETF INSMAC
   .IFF INSMAC      INS $MAC
   .IFNINS TKB      .SETF INSTKB
   .IFF INSTKB      INS $TKB
.;
   MAC XON,XON/-SP=LB:[1,1]EXEMC/ML,LB:[11,10]RSXMC/PA:1,SY:'<UIC>'XON
.;
   .IF <EXSTAT> NE 1        .GOTO ERRASM
   TKB @XONBLD
   .IF <EXSTAT> NE 1        .GOTO ERRTKB
   .SETO EXITST 1
   .GOTO EXIT
.ERRASM:
;
;  XON - XON could not be assembled.
;        Check to see if all files are present.
;
   .SETO EXITST 1
   .GOTO EXIT
.ERRTKB:
;
```

```
;  XON - XON could not be task built.
;
   .SETO EXITST 1
   .GOTO EXIT
.EXIT:
   .IFF INSMAC      REM ...MAC
   .IFF INSTKB      REM ...TKB
   .EXIT 'EXITST'

   ------------------------------------------------------

;  XONBLD.CMD -- Task build command file for XON task
;
   XON/PR,XON/-SP=XON,LB:[1,54]RSX11M.STB/SS
   /
   ASG=TI:1:2
   UNITS=2
   ACTFIL=0
   UIC=[1,54]
   TASK=...XON
   PRI=51
   //

   ------------------------------------------------------

   XON.MAC

   .TITLE  XON
   .IDENT  -V0.1.1-
   .SBTTL  TITLE PAGE
;
; Written: 14-FEB-79, -0.0.0-, Bruce C. Wright
; Modified:
; Verified:
;
; Modified:         10-AUG-82, -0.1.0-, Gerald P. Van Trieste, Jr.
;          RCA Corp. Cherry Hill, NJ 08358
;          RSX-11M V4.0 support for lower case messages
;          and change in system data structures
;
; Modified:         01-NOV-82, -0.1.1-, Gerald P. Van Trieste, Jr.
;          RCA Corp, Cherry Hill, NJ 08358
;          Reset XON (S1.CTS) in terminal UCB and do a
;          IO.WBT to recycle driver.
;
   .SBTTL  MACRO CALLS
   .MCALL  ALUN$S,GMCR$,DIR$,QIOW$,EXIT$S
   .MCALL  DCBDF$,UCBDF$
;
   .MACRO  ERROR   STRING
   .NCHR   N,<STRING>
   JSR     R0,ERROR
   .WORD   N
   .ASCII  "STRING"
   .EVEN
```

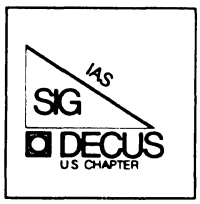
```
   .ENDM
;
   .SBTTL  OFFSET DEFINITIONS
;
   DCBDF$
   UCBDF$
;
;
   .SBTTL  MAIN LINE CODE
;
XON::
   DIR$    #GMCR             ; GET MCR COMMAND LINE
   BCC     1$                ; SKIP IF OK
   ERROR   <XON -- Cannot read MCR command line.>
1$:        MOV     @#$DSW,R0         ; GET LENGTH OF MCR LINE
   ADD     #GMCR+G.MCRB,R0 ; GET THE TERMINATING CHAR
10$:       MOV     #1,R1             ; SET .TPARS OPTION WORD
;                          ; THIS MEANS THAT KEYWORDS CANNOT
;                          ; BE ABBREVIATED AND BLANKS ARE
;                          ; SIGNIFICANT.
   MOV     #KEYTBL,R2        ; GET ADDRESS OF KEYWORD TABLE
   MOV     @#$DSW,R3         ; SET CHARACTER COUNT
   MOV     #GMCR+G.MCRB,R4 ; GET ADDRESS OF DATA STRING
   MOV     #START,R5         ; GET ADDRESS OF FIRST STATE
   CALL    .TPARS            ; CALL .TPARS TO PARSE STRING
   BCC     20$               ; CONTINUE IF NO ERROR
15$:       ERROR   <XON -- illegal device name>
20:        CMP     DEV,#"TT          ; DEVICE A TT
   BNE     15$               ; NO!
   CMP     UNIT,#77          ; UNIT TOO BIG?
   BLOS    21$               ; NO
   ERROR   <XON -- illegal device unit number>
21$:       ALUN$S  #2,#"TT,UNIT      ; ASSIGN DEVICE TO LUN 2
   BCC     22$               ; CONTINUE IF NO ERROR
   ERROR   <XON -- Cannot assign LUN>
22$:       MOV     UNIT,R2           ;
   CALL    CLRCTS            ;
   DIR$    #N1WBT            ;
   BCC     24$               ; CONTINUE IF IT WORKED
   ERROR   <XON -- IO.WBT directive error>
24:        EXIT$S                    ;
;
   .SBTTL  COMMAND LINE PARSE TABLES
   .MCALL  ISTAT$,STATE$,TRAN$
;
;  INITIALIZE STATE TABLES
;
   ISTAT$  STAT1,KEYTBL
;
;  SKIP OVER COMMAND NAME
;
   STATE$  START
   TRAN$   "XON"
;
   STATE$
```

```
   TRAN$   $BLANK
;
;  READ DEVICE AND UNIT NUMBER
;
   STATE$
   TRAN$   $ANY,,SETDV1
;
   STATE$
   TRAN$   $ANY,,SETDV2
;
   STATE$
   TRAN$   $NUMBR,DEV1,SETUNT
   TRAN$   $LAMDA
;
   STATE$  DEV1
   TRAN$   ':
;
;  SKIP TO NEXT NON-BLANK
;
   STATE$
   TRAN$   $BLANK,EOS1
   TRAN$   $LAMDA
;
   STATE$  EOS1
   TRAN$   $EOS,$EXIT
;
   STATE$
;
;
   .SBTTL  COMMAND LINE PARSE ACTION ROUTINES
;
;  ACTION ROUTINES
;
;  GET DEVICE NAME CHARACTER 1
;
SETDV1:    MOVB    .PCHAR,DEV
   RETURN
;
;  GET DEVICE NAME CHARACTER 2
SETDV2:    MOVB    .PCHAR,DEV+1
   RETURN
;
;  GET UNIT NUMBER
;
SETUNT:    MOV     .PNUMB,UNIT
   RETURN
;
;
   .SBTTL  SUBROUTINE TO PRINT ERROR MESSAGES
;
;  THIS SUBROUTINE IS INVOKED BY THE "ERROR" MACRO. IT
;  PRINTS THE ARGUMENT GENERATED BY THE "ERROR" MACRO ON
;  THE USER'S TERMINAL.
;
;  CALLING SEQUENCE;
```

```
;
;  JSR     R0,ERROR
;  .WORD   STRINGLENGTH
;  .ASCII  "STRING"
;
;  THE SUBROUTINE WILL NEVER RETURN TO THE CALLER.
;
ERROR:     MOV     (R0)+,QIO+Q.IOPL+2 ; MOVE IN QIO LENGTH
   MOV     R0,QIO+Q.IOPL   ; SET THE QIO ADDRESS
   DIR$    #QIO            ;
   EXIT$S                  ; AND EXIT
;
;
   .SBTTL  SUBROUTINE TO CLEAR XON IN UCB
;
;  ROUTINE TO CLEAR S1.CTS IN UCB OF GIVEN TERMINAL
;
CLRCTS:    MOV     $DEVHD,R0       ; GET FIRST DCB
10$:       CMP     #"TT,D.NAM(R0)  ; IS IT A TERMINAL?
   BNE     50$             ; NO
   CMPB    R2,D.UNIT+1(R0) ; IN THIS DCB?
   BGT     50$             ; NO - NEXT DCB
   MOVB    D.UNIT(R0),R3   ; LOAD R3 WITH LOWEST UNIT NO.
   MOV     D.UCB(R0),R1    ; Y - GET FIRST UCB
20$:       CMPB    R2,R3           ; RIGHT UCB ?
   BEQ     30$             ; YES
   INC     R3              ; UPDATE OUR UNIT NUMBER
   BR      40$             ; AND GET ANOTHER UCB
30$:       BIC     #S1.CTS,U.TSTA(R1) ; CLEAR XON
   BR      60$             ; AND EXIT
40$:       ADD     D.UCBL(R0),R1   ; GET NEXT UCB ADDRESS
   CMP     R0,U.DCB(R1)    ; SAME DCB ?
   BEQ     20$             ; YES
50$:       MOV     D.LNK(R0),R0    ; NO, GET NEXT DCB
   BNE     10$             ; AND TRY AGAIN
   ERROR   <XON -- No such TT>
60$:       RETURN                  ;
;
;
   .SBTTL  DATA AREA
;
;  RESULTS FROM PARSE OF COMMAND STRING
;
DEV:       .ASCII  /  /            ; ASCII DEVICE NAME
UNIT:      .WORD   0               ; UNIT NUMBER
;
;  MISCELLANEOUS DATA AREAS
;
IOST:      .WORD   0,0             ; I/O STATUS BLOCK
;
NULL:      .WORD   0               ; NULL STRING FOR IO.WBT
NULLEN = .-NULL
;
;  DIRECTIVE PARAMETER BLOCKS
;
```

```
QIO:        QIOW$    IO.WVB,1,1,,IOST,,<,,40>
N1WBT:      QIOW$    IO.WBT,2,2,,IOST,,<NULL,NULLEN,0>
GMCR:       GMCR$
;
   .END     XON
```


# The DeVIAS Letter




## Section Two

## In This Issue



## From the Editor

Does anyone have a technical article about IAS, DEC or DECUS that is just begging to be published, perhaps an opinion on any of the aforementioned or even a question that could spark a dialogue?

Has anyone migrated for IAS to VMS? Are you willing to share your story? Would you be willing to publish the source code for your favourite short subroutine or program? It may bring you fame and glory!

## Contributions

The DeVIAS Letter needs contributions in order to continue as an effective medium for exchange of information regarding IAS.

Contributions may be submitted in any form you wish. Originals on 8½ x 11 paper are preferred. However, even photocopies of relevant match-book covers would be appreciated.

Send all contributions to:

Ontario Hydro
700 University Avenue
Toronto, Ontario
CANADA, M5G 1X6

Attn: John W. Drummond
Mail Stop - M2E10

TYPE NL18.MEM

Department of Radiation Therapy
University of Pennsylvania
Room 410 - 133 South 36th Street
Philadelphia, Pennsylvania 19104

Dear IAS SIG Member,

The Las Vegas Symposium is over. Another overwhelming, exhausting, valuable experience. To answer the most frequently asked question, "No, I didn't!"

One highlight was the Wednesday evening dinner to celebrate the 10th anniversary of the RSX SIG. I took considerable interest in it, because I feel that while they have inherited the name "RSX" we are the real inheritors of "RSX-11D" which is what started the SIG. At any rate, they treated us well and even singled me out to honor and acknowledge IAS and the new IAS SIG and our contribution, over the years, to the RSX/IAS SIG. It was all very nicely done. Ralph Stammerjohn, the principal architect of the dinner/celebration, deserves the applause that he received and more. His contribution to the RSX/IAS SIG has been incalculably large. Now he has been elected Chairman of the RSX SIG to succeed Legare Coleman who did not run because of chainging job committments. I wish to thank Legare Coleman for his help and support in the formation of the IAS SIG and Ralph Stammerjohn for his support over the years of the IAS minority within the RSX/IAS SIG. There were others who helped, of course, but these two men stand out because of their position and should receive the plaudits for all.

Las Vegas was the first Symposium of the IAS SIG. We were well treated. We got good rooms for our meetings, not too big and not crowded. The times assigned to our sessions did not conflict with things that I wanted to attend. And, everything that we thought that we could do well was scheduled. Jim "J.R." Hopp, the Symposia Coordinator of the RSX SIG, was the principal architect of the Las Vegas scheduling for us. John Jenkinson set it up and did the ground work and Jim Hopp executed it. Thanks, Jim and John.

The next Symposium is at Cincinnati, Ohio in June (June 4-8, 1984). Ron Fussell (Air Force Intelligence Service) has accepted the job of Symposia Coordinator for the IAS SIG. Thank you very much, Ron. This task involves all the SIG's responsibilities for the Symposia. Specifically, Ron is responsible for all the IAS sessions at Cincinnati. He has a list of 15 sessions and two pre-symposium seminars that we will sponsor. Further, he will try to get us good rooms, perfect times, and see that none of our good sessions are edged out by another SIG. It is a hard job that, when done well, is invisible.

Tim Leisman, the IAS Product Manager, was there with four members of the "IAS Development Team." One question that never gets a definative answer is, "How many people do you have on the IAS Development Team?" Tim manages to sashay left every time it comes up. Remember when there were two people working on IAS during the last few years of the Reading, England, Development Team? With that as a reference I was reasonably pleased during the last two Symposia when Mike Reilly and Mike Garcia were presented as the "Team". There were

a few veiled references to the people back home, but nothing concrete. Now we have seen four. And, during one of the sessions, someone asked Mike Reilly about 'autobaud' for IAS; to which he relpied, "They are working on that back in Maynard this week." There MUST be more than four.

I like programmer jokes. Especially the kind that I can use on the blackboard of a classroom. For example: "Why is that programmers can't tell the difference between Haloween and Christmas?"

31 OCT = 25 DEC

Now that is not too terrific, but it is a welcome relief from my lecture whenever I use it in class! I would like contributions to my small collection of 'blackboard jokes' if you could, and I'll share the clean ones here. A chemistry type joke - "What compound is this:"

HIOAg"

Answer: Hi Ho Silver. Bad? How about: "What compound is this:"

HIJKLMNO

Answer: Water - H to O. You can see why I need help and contributions of new material.

Do you have a presentation that you would like to give in Cincinnati? A half-hour talk about what you're doing with IAS or to IAS? I would be glad to discuss it with you, if you wish. Or, just send in the pink "Call for Participation" form that you received before Las Vegas. If you have already disposed of it, call DECUS (617-467-4875) to request another. The same goes for a pre-symposium seminar. The SIG could use the profits from a successful seminar.

There were some explosive issues about the newsletter. All newsletters. I expect that the problems reflect growing pains and start-up troubles that will be ironed out. This newsletter will continue while we have material to fill it according to the best information that I have.

Happy Thanksgiving,
Robert F. Curley

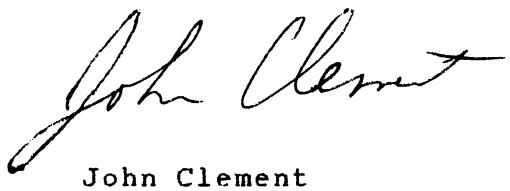
Bonner Lab
Rice University
PO Box 1892
Houston, Tx 77251
November 3, 1983

Ontario Hydro
700 University Ave.
Toronto, Ontario
CANADA, M5G 1x6

Dear Mr. Drummond:

I am submitting 2 possible articles for the DeVIAS/MULTI-TASKER. Since the initial release of Bonner Lab Runoff I have learned that a number of sites are using it. The few responses I have gotten are very positive, so I suspect there are more DEC users who would use it if they knew about it. The list of bugs is only of interest to current users, so publication of it is entirely up to you as to whether it is of any use. The list of bugs is included in the readme files for each new version.

Sincerely,



John Clement

# Bonner Lab Runoff

Author:John Clement
Bonner Lab
Rice University
PO Box 1892
Houston, Tx 77251
Tel (713) 527-4018

The following article is designed to acquaint the DEC user community with a one of the better versions version of Runoff. This version is available under [332,12] on the Fall symposium tape. A few features such as the odd/even paging will be available in Spring 84. The design goals of this version have been the following:

1. DSR(Digital Standard Runoff) should be emulated very closely.
2. The output format must be acceptable for publication in scientific journals, and also acceptable as a thesis under the rules of style at Rice University.
3. Scientific text processing must be supported including subscripts,superscripts, and equations.
4. All printer control codes should be supported no matter what type of printer is available.

The current version meets all of these goals. The first goal, DSR compatibility, allows use of the DSR manual as a primary source for input syntax. Only the commands imbedded in the input text are compatible. The command line switches are of course different. A few minor features are left out, but a large number of extra features are available.

The main additions to the command line switches are /-CR,/CH,/AP,/EV, and /OD. /-CR produces output text without imbedded carriage control characters. The output text may then be edited or copied by FLX in DOS format without problems. /CH and /AP allow selecting output text by chapter or appendix number. /EV and /OD are used to select output of either even or odd page numbers (Spring 84).

The output can be controlled by a variety of commands:

1. LAYOUT - This command controls the position of the title/subtitle and page numbers. This is a slight extension of the command in DSR.
2. HEADER SPACING - Controls the page header spacing.
3. CHAPTER LAYOUT - This changes the page layout only for the first page of a chapter or appendix. This is effectively a LAYOUT command which applies to only the first page of each chapter. With this command it is possible to have the page numbers in the upper right hand corner of each page, except for the first page of a chapter where it will be centered at the bottom.
4. STYLE HEADERS - This command modifies the header levels the same way as in DSR.

5. STYLE CHAPTER - This command modifies the position and spacing of the chapter header.
6. ENABLE ODD - forces the first page of each chapter onto odd pages, and PAGE ODD or PAGE EVEN force text onto odd or even page numbers.
7. DISPLAY - These commands define formats for numbers. Any number may be printed as either decimal, roman numeral, or letters. All numbers including the page, subpage, chapter, appendix, header level, or list entry may be modified. This is the same as in DSR. Unlike DSR a prefix, and postfix may bracket the number. For example the word PAGE is the prefix for the page number, and the word CHAPTER is the prefix for the chapter number. As an example chapter 4 may appear as
   CHAP [IV]

Permanent margins have been implemented. These are controlled by the PAGE SIZE command. The LEFT MARGIN, RIGHT MARGIN, and TOP MARGIN commands are used to control only temporary margins. The permanent margins control page headers and act as defaults for the other margin commands.

Two classes of commands not available in other versions of RUNOFF or in DSR have been added. These are the TEXT commands and IMMEDIATE commands. The IMMEDIATE commands mirror other commands such as TEST PAGE, but do not produce a break in the text line. The TEXT commands treat a whole block of text as a unit. They are:

1. CENTER TEXT - Centers text till an END CENTER command.
2. TEXT - The text after this command until the next TEXT or END TEXT command is treated as a block and will all be on the same page. If the block of text can not fit on the current page the rest of the page is left blank and the text begins on the next page.
3. TEXT DEFERRED - This command is similar to the previous one, except that the current page is not left blank to accomodate the block of text. The block is deferred and printed on a later page if necessary, and the current page is filled with text after the END TEXT command. This command will be available in Spring 84.
4. TEST TEXT n - checks if enough room is left on the page for n lines of text independent of spacing. This is similar to TEST PAGE, except that the amount of room reserved depends on the SPACING command.

TAB RIGHT and TAB LEFT commands allow either left or right justification of text in tabular form. In addition a break flag or AUTOBREAK flag may be used to line up text at arbitrary characters such as the decimal points. An ELLIPSES command may be used to fill between columns with ellipses. Additional commands allow or prevent automatic hyphenation, titles, or subtitles. UNDERLINE commands control which characters may be underlined.

Hyphenation has been improved. It now follows the standard rules of style. In addition the AUTOHYPHENATE command has parameters which can produce a more "pleasing" effect.

The most important additions to RUNOFF are the DEFINE and RESET commands. These allow definition of control codes and text abbreviations. The simplest of these commands is the DEFINE SUBSTITUTE. This allows definition of an abbreviated label for a whole line of text. While this is seldom useful for ordinary text, it is invaluable for scientific text. Complex equations or expressions may be pre-defined and reproduced wherever needed by inserting a substitute label. In addition some DSR compatible substitutions are automatically defined to produce the current date and time. DEFINE COMMAND does exactly what it implies. It makes up new commands from already existing ones. This may be used to abbreviate often used command sequences, or to make this version of RNO more compatible with other versions. RESET as the name implies removes definitions.

DEFINE ESCAPE is used to declare flag characters as triggers for escape sequences. This is the only way control codes or escape sequences can be used since all control characters imbedded in the input are discarded. In addition to defining the sequence you give it attributes. The attributes allow RUNOFF to handle them intelligently. Changes in pitch may be declared so RUNOFF can center double width characters. Extra width characters may not be justified except where imbedded in normal width text. The spacing of printable sequences may be defined. Vertical motion caused by control codes is also declared so that RUNOFF can properly take care of the vertical position. Ribbon, font, or attribute changes are declared in turn on/off pairs so RUNOFF can turn them off at the end of a page and back on after printing the header on the next page. By combining the escape sequence definition with FLAGS SPECIAL, you may simulate DEC standard syntax for flags. For example you may define * as the bold flag for printers that support bold face. Then ^* turns on bolding, * turns off bolding and * will bold a single character. This can simulate DSR bolding syntax on appropriate printers. You can also support underlining for printers via hardware.

By using the various define commands support for any printer may be generated in a device independent manner. In other words the flags used in the text will do the same thing on different model printers. If a printer is not capable of supporting a given feature you may define a dummy control code. It is also possible to use the control codes to trigger actions by post processors.

Three features are available in a semi-independent manner. These are the fractional line spacing and equation formatting and variable spacing. Naturally this article was printed with variable spacing. RUNOFF can produce half line spacing on printers supporting sub/superscripting. This includes a wide variety of word

processing printers. The control codes to perform this may be defined via the .DEFINE SUBSCRIPT and .DEFINE SUPERSCRIPT commands. These definitions are not necessary if the printer supports Diablo style control codes. Some printers which do this are Diablo, Spinwriter, Florida Data, DEC LA-50, and Qume. The other feature is equation formatting. Fractions may be generated automatically on word processing printers. Fractions may be nested up to a minimum of 6 levels deep. Spacing between the fractions may be automatically controlled, and sub/superscripts may be intermixed freely. In other words RUNOFF can automatically skip lines to accomodate expressions that take up several lines. The variable spacing is supported on printers capable of fractional spacing. This mode pads the blank spaces between words with fractions of a space to give the output a more uniform look. The default for variable spacing is DIABLO. This may be changed by the DEFINE VARSP command.

Finally error messages are informative, and full traceback is generated. The message pinpoints the input file names, line numbers and the location within the line where the error occurred. If the error is inside a substitution, then the label and substitute text are also printed. All intermediate substitutions which called the erroneous one are printed, as are the names of all open input files.

I am committed to supporting this program in the immediate future. It will run under VAX/VMS, RSX-11M, IAS and presumably RSTS. A student here at RICE is working on an RT version which may be ready sometime in the next year. Various versions of this program have been made available on the symposium tapes for Fall-82 through Fall-83 on uic [332,12]. A new version will be on the next Spring-84 tape. Once the RT version is finished the current version will be submitted to the DECUS library. The distribution now includes all of the sources, a complete manual and help files for RSX or VMS. In addition a set of .TST files contain tests and illustrations of the available syntax.

I would very much like to hear from the user community about bugs, or suggestions for enhancements. Unfortunately by the time most users have received the latest symposium tape the next version of Bonner Lab RUNOFF has been shipped to the next symposium. We need faster symposium distribution!!

John Clement
Bonner Nuclear Labs
Rice University
P.O. Box 1892
Houston, Tx, 77251
April 29, 1982

The following is a complete list of known bugs in BONNER LAB RUNOFF. Each version fixed the bugs in the previous one and the Fall 83 bugs will be fixed in the next release.

## BUGS in the Fall 82 version

1. Footnotes would not correctly justify and fill. This could have been worked around by using .NO FILL inside footnotes.
2. Footnotes would sometimes cause the first line of the next page to have incorrect fill and justification.
3. STYLE HEADERS wouldn't work in conjuction with NO HEADER, and TOP MARGIN 0.
4. .ELLIPSES would sometimes cause mistabulation.
5. .SUBPAGE command at the beginning of the first page causes RUNOFF to bomb. This is a nonsensical command sequence so nobody should have had problems with this one.
6. SUBSTITUTION failed unpredictably if a large number of substitutes were defined.
7. /RI switch did not work at all properly. This could be worked around with a permanent left margin.
8. RNO failed to underline the last word of a section terminated by \& if the last word occurred on the first word of an output line.
9. In .HEADER LEVEL command if capitalized first character is desired and the first char. was already in caps, the second char would be capitalized. The workaround was to not capitalize any section headers or use .STYLE HEADERS to disable automatic capitalization.
10. .LITERAL or .END LITERAL commands may bomb RNO with odd address trap. Also .LITERAL didn't turn off the flags properly if it worked. The use of .FLAGS ALL and .NO FLAGS ALL provided a partial workaround.
11. If overstriking preceeds underlining on the same line, underlining by /UL:S or /UL:L will fail to underline correctly. /UL:B worked however.
12. If both .NUMBER PAGE and .NUMBER SUBPAGE are used before a .SUBPAGE command the page number will be wrong.
13. Runoff can mess up the number of spaces in a line if the following conditions were met. .FILL .PERIOD are enabled and the end of the input line is terminated by a punctuation followed by a space. Since there is no need to have a space after a punctuation at the end of the line the workaround was to remove the

final space.

## BUGS in Spring 83 version

1. Equations imbedded inside text were not handled properly if the line had to be split. They are now properly moved to the next line.
2. Lines did not properly justify if they included equations. .NJ was the only solution in previous versions.
3. If .FILL is enabled and a printable escape sequence is braketed by a pair of spaces the final space was removed. This no longer happens if the HSP attribute is defined for the escape sequence.
4. If an escape sequence changed the pitch of the printer, improper underlining may result. If the pitch change can be specified by the PSP and HSP attributes then the underlining will be correct, otherwise the user must use /UL:B instead of UL:L.
5. Module RNOIF was in the wrong overlay. It must be in the same overlay with OPEN,CLOSE or an overlay that calls OPEN,CLOSE. This has been fixed. RNO would bomb when a .IF statement was not folowed by the appropriate .ENDIF inside a "required file".
6. If hyphenation is enabled and escape sequences are used occasionally RNO will hyphenate the output in strange places. The workaround is to .DHY during escape sequences.
7. Strange hyphenation may occurr after tabs have been used. In general hyphenation only worked properly for normal text. Now it should work properly for all text. Authohyphenation has been restricted to normal text, but user specified hyphenation may be used in any text except equations.
8. Roman numerals were incorrectly converted for 9,90-99,900-999. . .
9. .NUMBER LIST n started numbering at n+1 rather than n. .NUMBER LEVEL also worked incorrectly. The defaults have been modified for the .NUMBER commands to correspond exactly to DSR.
10. .LITERAL or .NO FLAGS ALL would turn off SPECIAL FLAGS permanently never to turn on again.
11. The LCK attribute for escape sequences did not work correctly unless .FLAGS ESCAPE command was used.
12. The /WA switch worked incorrectly when combined with /PA. RNO would wait at the top of pages which are not being printed.

## BUGS in the Fall 83 version now FIXED

1. The subindex flag could not be used as a regular character after a .INDEX command even when preceeded

by _.

2. Subindex terms with trailing or leading blanks are improperly indexed.
3. The index flag does not terminate an autoindex term.
4. .NUMBER LIST causes a fatal error in subsequent .LIST or .LIST ELEMENT commands.
5. If .NUMBER CHAPTER and a .LAYOUT other than 0 is used the last page of each chapter is improperly numbered.
6. .SUBPAGE, and .END SUBPAGE did not work correctly. The subpage letter was correct, but the page number was incorrect.
7. .DEFINE COMMAND will not work properly if the defined command line includes literal parameters. Unpredictable results occurr when the command is used.
8. .IMMEDIATE TEST TEXT and .IMMEDIATE TEST PAGE incorrectly broke the input lines.
9. /-ff:n did not work correctly. n had to be in half lines and if n=0 an infinite loop resulted.
10. /PS:n also had to be in half lines. It is now in lines.
11. Occasionally when .JUSTIFY and .VARIABLE SPACING are in effect the first line after the page header is improperly justified.

## BUGS NOT YET FIXED

There are currently a few bugs which have not yet been fixed. Any help from other users in fixing them is gratefully appreciated.

1. Indirect command files do not work as input to RNO under IAS. They work for RSX-11M and VMS under compatibility mode! Since the documentation for RSX and IAS I/O routines and parsing is identical the problem is very maddening. This may actually be a bug in IAS!

DeVIAS Questions - Answers

Name: William H. Yost

Mailing Address: Naval Avionics Center D/822
6000 E. 21st Street
Indpls, IN 46218

Phone Number: ( 317 ) 353 - 3225

Instructions

1) Complete relevant part of form
2) Mail to Editor, The DeVIAS Letter
3) Question and/or Answer will be published in next newsletter.

Date Submitted: NOV / 2 / 1983

Question: What happened to the RSX SIG Multi-Tasker newsletter? Ever since we started paying for it, we have been getting a combined IAS/RSX newsletter but the 1st one I got had no articles on RSX11M and the 2nd one I got had only one, a VIC directory of the RSX SIG Tape. Has the multi-tasker editor resigned or what happened?

Answer:

1) THE PEOPLE RESPONSIBLE FOR DECUS PUBLICATIONS DECIDED TO COMBINE THE DeVIAS LETTER AND THE MULTI-TASKER INTO ONE PUBLICATION.

2) THE MULTI-TASKER EDITOR, CHARLES GOODPASTURE, DID NOT SUBMIT MATERIAL FOR PUBLICATION.

3) THE MULTI-TASKER IS PUBLISHING AGAIN, WITH A NEW EDITOR.

DeVIAS Questions - Answers

Name: JONES C. CHEN

Mailing Address: SUN TECH INC.
P.O. BOX 309
BISHOP HOLLOW ROAD, NEWTOWN SQUARE, PA. 19073

Phone Number: ( 215 ) 356 - 1800 - EXT 81

Instructions

1) Complete relevant part of form
2) Mail to Editor, The DeVIAS Letter
3) Question and/or Answer will be published in next newsletter.

Date Submitted: NOV / 3 / 1983

Question:

① WHERE AND HOW CAN I GET A COPY OF THE SPRING & FALL, 1983 DECUS TAPE IN OUR LOCAL LUG?

② MAY BE YOU CAN LIST OUT ALL THE RSX SIG LUGS IN THE NEXT MULTITASKER NEWSLETTER

Answer:

# RATFIV VERSION 2

RATFIV IS A STRUCTURED FORTRAN PREPROCESSOR PROVIDING SWITCH, IF - ELSE, WHILE, FOR, DO, REPEAT - UNTIL, STRING, AND BREAK AND NEXT CONSTRUCTS. ALSO SUPPORTED ARE INCLUDE FILES, DEFINE FOR SYMBOLIC CONSTANTS AND MACROS WITH ARGUMENTS, CONDITIONAL COMPILATION, FORMATS IN READ, WRITE, ENCODE, AND DECODE STATEMENTS, USE OF >, <, ETC. INSTEAD OF .GT., .LT., ETC, AND THE RETURN VALUE CONSTRUCT IN FUNCTIONS.

RATFIV WAS DEVELOPED FROM THE RATFOR COMPILER DISTRIBUTED BY LAWRENCE BERKLEY LABS; VERSIONS ARE AVAILABLE FOR VAX VMS AND PSX/IAS SYSTEMS. THE MAJOR ENHANCEMENTS IN RATFIV ARE:

1. ABILITY TO SPECIFY A FORMAT STATEMENT INSIDE READ, WRITE, ENCODE, AND DECODE STATEMENTS;

2. CONSISTENT LINE CONTINUATION USING THE UNDERLINE CHARACTER;

3. PRODUCTION OF PROPERLY INDENTED UPPER CASE FORTRAN CODE WITH COMMENTS PASSED THROUGH;

4. OPTIONAL OUTPUT OF FORTRAN 77 CODE WITH THE /F77 SWITCH;

5. ADDITION OF A /SYMBOLS SWITCH TO THE COMMAND LINE TO OPTIONALLY READ THE SYMBOLS FILE;

6. OUTPUT OF QUOTED STRINGS OR OPTIONALLY HOLLERITH STRINGS (QUOTED STRING OUTPUT ALLOWS RATFIV TO BE USED WITH DEC FORTRAN OPEN STATEMENTS, THE FORTRAN 77 CHARACTER DATA TYPE, ETC.);

7. EVALUATED AND UNEVALUATED ARGUMENTS IN MACROS;

8. CORRECT LINE NUMBER REPORTING;

9. EXIT WITH ERROR STATUS IF AN ERROR OCCURS DURING COMPILATION;

10. USE OF CHARACTER CONSTANTS IN CASE LABELS;

11. NUMEROUS BUG FIXES;

12. COMPREHENSIVE DOCUMENTATION;

13. RATFIV KEYWORDS NEED NOT APPEAR AT THE BEGINNING OF A SOURCE LINE IN ORDER TO BE RECOGNIZED.

THIS VERSION OF RATFIV MAY BE USED IF YOU WANT TO COMPILE THE RATFIV SOURCE TO THE OTHER PROGRAMS WRITTEN IN RATFIV ON THIS TAPE.

VERSION 2 OF RATFIV HAS A FEW ENHANCEMENTS, BUG FIXES, AND CHANGES SINCE VERSION 1. THE ENHANCEMENTS ARE:

1. THE DOCUMENTATION HAS BEEN REVISED, PARTICULARLY THE SECTION ON DEFINING MACROS;

2. HOLLERITH STRINGS MAY BE OPTIONALLY OUTPUTTED INSTEAD OF QUOTED STRINGS;

3. THE INITIALIZATION AND INCREMENT PARTS OF FOR LOOPS MAY HAVE MORE THAN ONE STATEMENT IN THEM;

4. MACRO DEFINITIONS MAY HAVE COMMAS IN THEM;

5. ARGUMENTS TO THE _LEN AND _INCLUDE MACROS MAY HAVE COMMAS IN THEM;

6. THE SPECIAL MACRO ARGUMENTS $& AND %& ARE REPLACED BY ALL THE ARGUMENTS PASSED TO THE MACRO;

7. OCTAL CONSTANTS MAY BE INSERTED IN STRING VARIABLES;

8. THE NUMBER OF SPECIAL ESCAPED CHARACTERS IN THE STRING STATEMENT HAS BEEN INCREASED AND THESE CHARACTERS MAY ALSO BE USED WITH THE CASE STATEMENT;

9. QUOTED STRINGS ARE CONSISTENT IN THAT DOUBLE AND SINGLE QUOTED STRINGS MAY BOTH HAVE DOUBLE OR SINGLE QUOTES IN THEM, EVEN IN THE STRING AND CASE STATEMENTS;

10. MACROS MAY BE DEFINED WHICH EMULATE THE _IFDEF, _IFNDEF, _ENDDEF, AND _ELSEDEF MACROS (SEE THE FILE RATFOR.SYM FOR AN EXAMPLE OF THIS);

11. SOME UNUSED CONTINUE STATEMENTS IN THE FORTRAN OUTPUT HAVE BEEN ELIMINATED.

A FEW CHANGES WERE MADE THAT COULD CAUSE INCOMPATIBILITY WITH VERSION 1. I AM SORRY FOR THESE CHANGES, BUT THEY WERE NECESSARY FOR MY PEACE OF MIND. I EXPECT THIS VERSION TO BE PRETTY STABLE. THE CHANGES ARE:

1. THE DOT (.) CHARACTER IS NOT ALLOWED IN MACRO NAMES ANYMORE. THIS CHANGE WAS MADE BECAUSE IT COULD

CAUSE PROBLEMS WHEN A MACRO NAME APPEARED NEXT TO A VALID DOT CHARACTER, AS IN "IF (STATUS.EQ.EOF)".

2. THE BREAK STATEMENT MAY NOT BE USED TO BREAK FROM A SWITCH STATEMENT ANYMORE. THIS CHANGE WAS MADE FOR COMPATIBILITY WITH RATFOR PREPROCESSORS AND TO MAKE BREAK CONSISTENT WITH THE NEXT STATEMENT. IT WAS A BAD IDEA IN THE FIRST PLACE TO HAVE BREAK BREAK FROM A SWITCH STATEMENT.

3. BRACKETS ([]) HAVE NO SPECIAL MEANING WITHIN MACRO DEFINITIONS ANYMORE. THE USE OF BRACKETS IN MACROS WAS TOO OBSCURE TO JUSTIFY THEM. IN ANY CASE THERE WAS LITTLE NEED FOR THEM.

4. THE NEWLINE CHARACTER WAS CHANGED FROM CARRIAGE-RETURN TO LINE FEED FOR COMPATIBILITY WITH RATFOR PREPROCESSORS.

SOME OF THE BUG FIXES ARE (THERE WEREN'T TOO MANY BUGS):

1. INCLUDE STATEMENTS INSERT THEIR INPUT INTO THE INPUT STREAM CORRECTLY WHEN THEY APPEAR IN A MACRO;

2. ERROR LINE NUMBER REPORTING IS REALLY CORRECT THIS TIME;

3. LITERAL FORTRAN CODE IS ALLOWED IN MACROS.

THE FILE "RATFOR.SYM" CONTAINS SOME MACRO DEFINITIONS WHICH, IF PLACED AT THE FRONT OF THE SOFTWARE TOOLS "SYMBOLS." FILE, MAKE RATFIV LARGELY COMPATIBLE WITH THE LATEST (SPRING 1981 VAX SIG TAPE) VERSION OF SOFTWARE TOOLS RATFOR AVAILABLE FROM JOE SVENTEK. I HAVE COMPILED ABOUT 15 THOUSAND LINES OF HIS CODE WITH RATFIV; THE ONLY PROBLEM WAS THAT SOME VARIABLES NAMED "STRING" ("STRING" IS A RATFIV KEYWORD) HAD TO BE RENAMED. SOFTWARE TOOLS RATFOR CODE SHOULD BE COMPILED WITH RATFIV'S "/SYMBOLS" SWITCH, AND WITH RATFIV'S "/HOLLERITH" SWITCH FOR COMPATIBILITY ON THE VAX. THIS IS THE ONLY PLACE WHERE THE "/HOLLERITH" SWITCH HAS BEEN NEEDED SO FAR. NOTE THAT SHORTENING OF LONG NAMES IS NOT AVAILABLE IN RATFIV; I HAVE NOT CHECKED WHETHER LONG NAMES ARE USED IN THE RSX/IAS VERSION OF THE SOFTWARE TOOLS. ALSO, THE "$(" AND "$)" BRACKETS FOR USE WITH MACROS ARE NOT AVAILABLE IN RATFIV; THERE SEEMS TO BE LITTLE USE FOR THEM IN RATFIV.

I EXPECT THAT RATFIV WILL BE FAIRLY STABLE FROM NOW ON, ALTHOUGH IT'S POSSIBLE I WILL ADD ENHANCEMENTS. I AM INTERESTED IN HEARING ABOUT AND FIXING BUGS, HOWEVER.

TO BUILD RATFIV, DO

@BUILD

DOCUMENTATION IS IN THE FILE RATFIV.DOC.

THE "@BUILD" FILE BUILDS RATFIV AND ALSO MACRO, WHICH IS JUST THE MACRO PREPROCESSOR PORTION OF RATFIV. INPUT PASSES THROUGH THE MACRO PROCESSOR UNCHANGED EXCEPT THAT MACROS ARE EXPANDED. THERE IS NO DOCUMENTATION FOR MACRO, HOWEVER THE FORM OF THE COMMAND LINE IS LIKE RATFIV'S AND THE MACROS ARE THE SAME, EXCEPT THAT "DEFINE" ISN'T AVAILABLE; USE "=MACRO" INSTEAD.

THERE IS A SYSTEM-WIDE FILE CALLED "SYMBOLS." WHICH CAN BE READ BY SPECIFYING "/SYMBOLS" ON THE RATFIV COMMAND LINE (SEE THE SECTION, "USING RATFIV" IN "RATFIV.DOC"). THE DEFAULT LOCATION OF THIS FILE MAY BE CHANGED BY CHANGING THE DEFINE FOR "USERSBIN" IN THE "SYMBOLS." FILE AND REBUILDING THE COMPILER FROM SCRATCH. MOST PEOPLE WON'T NEED THIS FILE, HOWEVER, SO YOU CAN JUST LEAVE IT ON THE DIRECTORY WHERE RATFIV IS LOCATED, SINCE RATFIV FIRST CHECKS THE DEFAULT DIRECTORY FOR THE "SYMBOLS." FILE BEFORE LOOKING FOR THE SYSTEM-WIDE FILE.

IF YOU ARE USING THE F4P COMPILER INSTEAD OF F4P ON A PDP11, BE SURE TO COMPILE THE FORTRAN SOURCES WITH THE -SN SWITCH. IGNORE THE ERRORS IN TO.FTN; THEY ARE IN MODULES WHICH AREN'T USED BY RATFIV. YOU WILL ALSO HAVE TO CHANGE THE RATFIV.TKB FILE AS FOLLOWS:

1. DELETE THE MAXBUF=512 OPTION.

2. CHANGE ACTFIL=8 TO ACTFIL=7

3. CHANGE UNITS=10 TO UNITS=9

THESE CHANGES REDUCE THE NUMBER OF FILE INCLUSION LEVELS ALLOWED BY RATFIV BY ONE, BUT THE TASK WOULD BE TOO LARGE OTHERWISE. IF THE TASK IS STILL TOO LARGE, YOU CAN REDUCE ACTFIL TO 6 AND UNITS TO 8, BUT DON'T REDUCE THEM ANY MORE AS THEN YOU WILL NOT BE ABLE TO INCLUDE ANY FILES.

AFTER REDUCING THE NUMBER OF FILE INCLUSION LEVELS, THEN AFTER SUCCESSFULLY BUILDING RATFIV YOU SHOULD CHANGE THE DEFINE FOR NFILES IN THE FILE "MACSYM." TO REFLECT THE CHANGES (I.E SUBTRACT 1 OR 2 FROM THE DEFINED VALUE OF NFILES), AND THEN REBUILD THE COMPILER FROM SCRATCH.

SEND COMMENTS, PROBLEMS, ETC. TO:

WILLIAM P. WOOD, JR.
INSTITUTE FOR CANCER RESEARCH
7701 BURHOLME AVE.
PHILADELPHIA, PA. 19111
(215) 728 2760

# IAS QUESTIONS AND ANSWERS

MICHAEL REILLY
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MA

MICHAEL GARCIA
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MA

TIM LEISMAN
DIGITAL EQUIPMENT CORPORATION
STOW, MA

ROBERT CORLEY, SESSION CHAIRPERSON
UNIVERSITY OF PENNSYLVANIA
FLOURTOWN, PA

REPORTED BY KEVIN CARLIN, DECUS SCRIBE SERVICE

Q. I HAD A COMMAND FILE WHICH SOUGHT TO CONCATENATE MORE THAN 30 OCTAL FILES IN A SINGLE, MULTILINE COPY COMMAND. WHEN IT REACHED THE 30TH FILE IT GAVE A PDL ERROR AND THEN LOGGED ME OFF.

A. PDS RAN OUT OF BUFFER SPACE. SINCE VERSION 2 THIS HAS BEEN A MEMORY MANAGEMENT PROBLEM, WHICH MEANS THAT THE SYSTEM WILL AUTOMATICALLY PERFORM THE LOGOUT.

---

Q. WE HAVE A SITE WHERE WE USE A LOT OF COMMONS. CAN WE GET I/D SPACE ON INSTALL?

A. THE NEW RSX-11M TASK BUILDER SHOULD SOLVE ALL OF YOUR PROBLEMS. (AUDIENCE): RESIDENT OVERLAYS CAN BE USED TO IMPLEMENT SHAREABLE SEGMENTS. SAY THE FIRST TASK USES 30K RESIDENT OVERLAY AND 8K BESIDES THAT OVERLAY, ANY SUCCESSIVE, CONCURRENT LIKE TASKS WILL ONLY REQUIRE 8K MORE SPACE.

---

Q. I WOULD LIKE TO EXPAND THE SPACE AVAILABLE TO MY USERS, HOW WOULD I GO ABOUT IT?

A. MOST OF THE MAPPING IS DECIDED BY INSTALL, WHICH SETS THE APPROPRIATE REGISTERS.

---

Q. HOW DO I FIND AN ADDRESS IN THE MODULE HEADER.

A. WE DON'T KNOW.

---

Q. DISPATCHES HAVE BEEN USED IN THE PAST TO INFORM USERS OF CURRENT SOFTWARE PROBLEMS. SINCE DECNET'S RELEASE, SPR SOLUTION TURNAROUND HAS BEEN ON THE ORDER OF SIX MONTHS SLOW. WHY CAN'T WE GET THE SOLUTIONS SOONER?

A. WE DO NOT SUPPORT DECNET IAS, BECAUSE THAT GROUP FEELS THERE IS ENOUGH VARIATION IN SYSTEMS OUT THERE WITHOUT SENDING OUT "NON-CRITICAL" PATCHES. THIS SITUATION WILL BE CHANGED WHEN WE TAKE OVER SUPPORT.

---

Q. IN VERSION 3.1, I HAVE SEVERAL TASK IMAGES WITH NO EXECUTIVE SOURCES. WHEN I TRY TO COMBINE THEM WITH MY SOURCED MATERIAL, I GET AN UNMATCHED DATES ERROR. I FOUND AND MODIFIED ONE DATE IN ONE OF THE IMAGES, BUT THIS DIDN'T STOP THE ERROR.

A. NORMALLY WE REBUILD OUR TASKS IN THAT MANNER, SO IT IS POSSIBLE TO DO THAT.

---

Q. WE'RE WRITING "WRITE-PASS-ALL" TO A GRAPHIC TERMINAL DEVICE. THIS WORKS FINE UNTIL ANOTHER TASK DOES A SIMULTANEOUS READ FROM LINE. IS THERE ANY WAY WE CAN KEEP THESE READS FROM INTERFERING WITH OUR DISPLAY?

A. SUPPRESS THE CONTROL-R OUTPUT IN TTY HANDLER OR SET I/O TO FULL DUPLEX AND NO ECHO.

---

Q. DBMS-11 INSISTS ON SENDING ME MESSAGES WHETHER I WANT THEM OR NOT. HOW CAN I MAKE IT STOP.

A. SEE THE LATEST EDITION OF DATABASE.

---

Q. IS THERE ANY PLAN TO SUPPORT TIME OF YEAR/UNIBUS CLOCKS.

A. WE HAVEN'T REALLY THOUGHT ABOUT IT, WE'LL HAVE TO SEE WHAT'S OUT THERE NOW AND MAKE A DECISION. (AUDIENCE) WE USE NESTED INDIRECT COMMAND FILES TO BOOT OUR SYSTEM. IT AUTOMATICALLY SETS UP THE DATE AND TIME.

---

Q. WE HAVE A PROBLEM WITH SAVE'D FILES .... WE HAVE A DISK DRIVE PROBLEM.

A. IF YOU HAVE A DISK DRIVE PROBLEM, SAVE IS THE BEST WAY TO

KEEP TASKS FROM RUNNING.

---

Q. HOW CAN I GET A LIST OF ALL THE FILES WHICH WILL BE PURGED WITHOUT ACTUALLY PURGING THEM?

A. USE THE OBSOLETE VERSION OPTION ON SRD.

---

Q. IS THERE ANY WAY TO GET BRU TO STORE OR READ FILES ACROSS USER NUMBERS?

A. YES, IF YOU CAN FIND A WAY TO LINK BRU WITH PIP.

---

Q. I HAVE A DATA SECURITY PROBLEM. A USER CREATED A DIRECT ACCESS DATA FILE USING FORTRAN 77. WHEN THE USER PRINTED THAT FILE OUT HE ENDED UP WITH SEVERAL PAGES OF SENSITIVE BUSINESS INFORMATION BELONGING TO ANOTHER USER. THE INFORMATION BEING PICKED UP SEEMS TO BE STUFF MARKED FOR DELETE.

A. A PROGRAM HAS BEEN WRITTEN TO ZERO A FILE BEFORE MARKING IT FOR DELETE. THERE IS ALSO A PROGRAM CALLED BROOM WHICH WILL SET ALL INFORMATION MARKED FOR DELETE ON A GIVEN DISK TO BINARY ZEROES. A TAPE HAS BEEN COMPILED FROM SEVERAL PDP-11 LIBRARY SOURCES AND THE MATERIAL ADAPTED TO THE IAS SYSTEM. THESE TAPES ARE AVAILABLE THROUGH ROBERT CURLEY AT THE UNIVERSITY OF PENNSYLVANIA. THEY ARE STORED AS <14,*> ACCOUNTS AND MAKE UP THREE 320 FOOT REELS OF DOS MAGTAPES. THEY INCLUDE A PROGRAM TO ZERO A FILE AND MARK IT FOR DELETE AND ANOTHER PROGRAM, CALLED RECREATE, WHICH WILL ATTEMPT TO RECONSTRUCT A FILE ALREADY MARKED FOR DELETE WHEN GIVEN CERTAIN VITAL INFORMATION ABOUT THE FILE.

---

Q. IS THERE ANY WAY TO GET AN INDIRECT COMMAND FILE TO ACCEPT DATA FROM A COMMAND FILE RATHER THAN "TTY:".

A. NOT CURRENTLY. A NEW VERSION WILL BE ABLE TO WRITE A FILE AND READ IT LATER.

---

Q. WHEN ARE WE GOING TO SEE THE MANUAL WE'VE BEEN ASKING FOR?

A. WE SET THE PRICE PER COPY AT $1000 AND HAVEN'T RECEIVED ANY TAKERS AS YET. IF YOU WANT TO OUTLINE A SERIES OF TALKS ALONG THE LINES OF A TUTORIAL WE WILL BE GLAD TO DELIVER PAPERS ACCORDING TO YOUR OUTLINE AT FUTURE DECUS SYMPOSIA.

---

Q. ARE YOU GOING TO SUPPORT A TOOLKIT FOR THE 350S?

A. WE HAVEN'T LOOKED AT THE IDEA YET. WE'LL LOOK AT THE RSX TOOLKIT AND DECIDE WHAT TO DO WITH THAT. IF IT'S NOT A MAJOR CHANGE AND IF THERE'S A DEMAND, THEN THAT KIT MIGHT BE USED TO START A 350 TOOLKIT.

---

Q. COULD YOU DESCRIBE HOW THE TERMINAL HANDLER HANDLES XON AND XOFF. DOES TYPE AHEAD USE XON AND XOFF WHEN THE BUFFER OVERFLOWS?

A. WHEN THE TYPE AHEAD BUFFER OVERFLOWS (IT CAN HOLD 80 CHARACTERS) IT CALLS A ROUTINE WHICH RINGS THE TERMINAL BELL AND RETURNS.

---

Q. WILL BRU BE MADE COMPATIBLE WITH VMS?

A. WE'RE COMMON WITH RSX, AND THEY DON'T INTEND TO TO DO SO.

---

Q. CAN WE HAVE A STAND-ALONE BRU?

A. IT'S BEEN VERY DIFFICULT TO WEED A PARTICULAR IMAGE OUT OF THE SYSTEM IN THE PAST. WE CAN'T CURRENTLY GIVE JUST A STAND-ALONE RSX UTILITY.

---

Q. ANY CHANCE PDS WILL RECOGNIZE NODE NAMES LIKE VMS?

A. IT WAS IMPLEMENTED IN A DEVELOPMENT VERSION OF VERSION 2, USING A DIFFERENT STANDARD THAN VMS, BUT WAS REMOVED BEFORE VERSION 2 WAS RELEASED FOR UNKNOWN REASONS.

## DeVIAS Questions - Answers

Name: _____

Mailing
Address: _____
_____
_____

Phone
Number: (  ) _____-_____

Instructions

1) Complete relevant part of form
2) Mail to Editor, The DeVIAS Letter
3) Question and/or Answer will be published in next newsletter.

Date Submitted: _____/_____/_____

Question:

Answer:

DECUS

DECUS SUBSCRIPTION SERVICE
DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MRO2-1/C11
MARLBORO, MASSACHUSETTS 01752

Bulk Rate
U.S. Postage
PAID
Fitchburg, MA
Permit No.
21

**MOVING OR REPLACING A DELEGATE?**

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

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: _____

Name: _____

Company: _____

Address: _____

_____

State/Country: _____

Zip/Postal Code: _____

Mail to: DECUS - ATT: Subscription Service
One Iron Way, MRO2-1/C11
Marlboro, Massachusetts 01752 USA

ing label
bel is not
print old
re.
me of
n, com-
ersity,