# RSX

# MULTI-TASKER

## JULY 1984 ISSUE

*Printed in the U.S.A.*

# TABLE OF CONTENTS

From the Editors . . . . . 1
Reactions to Cincinatti . . . . . 2
Working Group News . . . . . 4
Local Users Group Newsletter . . . . . 7
Membership Statistics . . . . . 9
Multi-Processors Yes . . . . . 9
18-bit Versions of M-Plus . . . . . 10
Notes on DL Drivers, RMD, Idle Terminal Monitors . . . . . 11
Comparison of Command Line Interpreters . . . . . 23
Problem With DCL Delete in M V4.1C . . . . . 43
Help Yourself . . . . . 44
Responses to Placing Quote in Indirect Symbol . . . . . 46
Organization of Volunteers for RSX . . . . . 51
RSX SIG Volunteer Questionnaire . . . . . 53

# FROM THE EDITOR

Allen Watson
Multi-Tasker Editor

First of all, this is the first time since I began co-editing with Dominic DiNollo that we have been unable to get together to put together an issue, mainly because of schedule conflicts on my side. As a result I have put this issue together myself, and, pressed for time, I am afraid there may be some sloppiness about it. For instance I have not had time to try to group things into "articles" and "columns", etc. Instead I have just strung together all the contributions. So if you feel the quality has gone down in the format of this issue, blame me, not Dom.

I am writing this about ten days after the end of the DECUS Symposium in Cincinatti. It was a good session, as tiring as usual, or more. Any of you reading this who have never attended ... DO IT! The amount of information that exchanges minds in a DECUS Symposium is nothing short of incredible. The cost, even with hotel, is comparable to a one-week course at a DEC Ed Center, and believe me, you will get more useful information at a Symposium than you ever will in a canned course from DEC. One of the main reasons is that you can constantly corner people and ask questions specific to your problems and needs. I managed to get our company to class DECUS under "education" instead of "conventions", so they send me twice a year. I don't feel that I am deceiving them, either.

As Terry comments later in this issue, the RSX sessions were very well attended. Spirits seemed quite high, compared to a year ago when things looked dark and gloomy to many. RSX continues to flourish.

I got conned into re-presenting my sessions on "Introduction to RSX Indirect Command Files" and "Nifty Things to Do With" same, at the next DECUS in Anaheim this coming December. So if any of you didn't hear them, haven't read them where they have been published, and can make it to Disneyland, come on along. There seem to be a lot of neat sessions planned for Anaheim; more about them all when the final list is approved.

The RSX volunteer questionaire at the end of this issue is crucial to the continued health of the SIG. Please take the time to tear out the pages, copy them for as many people as you have in your shop, and mail them all in to Liz Bailey as instructed. It's a chance to let us know what you want to see the SIG doing.

# REACTIONS TO CINCINNATI

Terry Medlin
RSX SIG Chairman

The Cincinnati symposium was any measure a success. If you missed it, you should endeavor to find out what occurred. Purchase of the sessions notes and the audio tapes are highly recommended. Dominic Dinollo coordinated the session notes for Cincinnati and he did a terrific job.

The SIG CHAIRS convened on Saturday night to begin the process. Several high level marketing types and product oriented managers met with the SIG CHAIRS to discuss how SIGS could better work with DEC in the product planning work at DEC. This discussions continued throughout the week with the goal of making the process work. The BOARD is now expected to evaluate the need of this activity and plan for its integration into the new DECUS structure.

Seminars, leadership activities, and steering committee meetings completely filled the Sunday agenda. The RSX SIG sponsored one seminar: Writing I/O Drivers, given by Hans Jung and Steve Katzman. A total of 18 people took the seminar. Leadership activities were focused on presentation skills including better techniques for integration slides and overheads into a talk. Such information is a definite advantage to becoming involved in the SIG. The RSX steering committee met for some three hours to smooth out our plans for the rest of the week. The RSX developers attended part of the meeting so we could get to know each other. Jim Hopp, our symposia coordinator, did and does a super job of making the symposium a happy experience for all of us.

The SIG began its sessions on Monday with the standard "opening" session and the "product panel". Both sessions were well attended and informative. Gary Oden, the RSX product manager, supplemented the product panel with pictures of the development team which was a great idea. The RSX Question and Answer session occurred on Monday. It was so well received that we had to schedule another Q/A session later in the week. A total of some 120 questions were asked. The efforts of Brian McCarthy and Tony Lekas should be noted since they handled the greatest number of questions.

The ever popular System Programming Short Notes was moderated (conducted?) by Jim McGlinchey. Starting at 9:00 P.M., this session is one of the first steps to determine who is truly and undeniably serious about becoming an RSX expert. Also, used to separate "hams" from "?".

Things really got rolling on Tuesday with the advent of sessions such as "RSX Disk Backup Procedures", "Which RSX System is for You", and "VAX-11 RSX". The new VAX AME looks like a real product produced with the quality that RSX users expect. Look forward to article on it in the near future. Allen Bennett and Kerry Wyckoff led a menu session which led to the collection of over 100 items which is superb. Thanks guys. Tuesday night was the full blown DECUS reception which was superb! Scallops, shrimp, and a raft of other truly delightful foods were consummed in great quanity!

The Software Clinic, led by head doctor (psychiatrist?) Ed Cetron started the Wednesday schedule. Thanks to ALL who served as patients, interns, and doctors! Later in the day came two hits: "Everything DEC wanted to Know About RSX" and "A Modest Proposal". Both were smashingly successful. Issues that arose from these sessions include:

- we want a NEW system logic manual
- RSX windowing products should be called WINDEC (as in windex)
- Future disk development should include the BP11 as in "bottemless pit" so as to guarantee adequate disk space
- A future enhancement to RSX might provide for virtually unlimited virtual address space. Are we to call this the VU11 option?
- Better support for "non-terminal" terminals is needed

From 9:30 P.M. to 3:30 A.M., a dedicated group SIG tapes volunteers met to create the Cincinnati SIG tape. I hope that EVERY reader of our newsletter appreciates those who submit things and those who laboriously create this tape. Glenn Everhart spearheads this remarkable effort which is unmatched in volume of software ANYWHERE. Glenn will be writing a future article on the SIG tape.

Several users made presentations on Thursday. Such talks are usually the most creative presentations on the schedule. Thanks to all of you! The extended Q/A was held in the campround using several "virtual chairs" and a "virtual microphone" but it worked well anyway. The steering committee sponsored a "meet-us" mim-reception in the campground. Several new volunteers were identified and we were greatful for that. To close out the Thursday schedule, we had the "lore" session. War stories, weird things, and assorted disasters were related in exacting detail for the amusement of all. We had to terminate at 11:00 due to convention center security policies. This required an adjustment to meet in Ed's hotel so as to continue our comraderie until whenever. I gave up at 2:30.

Friday was spent looking to Anaheim. Of note are the following:

- Seminar on the SIG tapes in detail
- Seminar on RSX internals
- Seminar on industrial automation

- A presentation on M-PLUS 2.2
- A session of the multi-processor features of M-PLUS
- Several new sessions

As per normal, future issues of the Multi-Tasker will contain more information on what happened including the Q/A session, the clinic, and the contents of the menu.

During the entire week, our many working groups met to evaluate current efforts and plan the future. Jeff Hamilton, our working group coordinator, also formed one new group dealing with "cheap networks". Packges such as SRD and RUNOFF have come from our working groups. The chairs of these groups spend their time developing or enhancing packages that many of us use and appreciate.

One of the focal points of the symposium was the goal of integrating volunteers into the SIG. Liz Bailey, our co-chair, and Jeff did a marvelous job at recruiting new people. The fact that such efforts were so successful indicates to me that they did a fine job. It also says that our SIG is very healthy!

I cannot close this report without commenting on the enjoyable relationship that the SIG now has with DEC and with the development team. I am quite sure that that the environment of cooperation and trust that now exists will lead to several significant changes in the near future. When you consider the amount of effort expended by the development team FOR US, I think they deserve a special thanks.

Terry Medlin,
SIG Chair

## Working Group News

Jeff Hamilton
Working Group Coordinator
(214)457-4175

Date of this report: 12JUN84
The working group chairmen are as follows:
RSX-11M Unsupported Versions:
Bill Burton

Texas Research Institute
1300 Moursand
Houston, Texas 77030

System Performance and Accounting
Roy S. Maull
U. S. Air Force
HQ SAC / ADlAE
Offutt AFB, Ne 68113

DECUS Library
Bruce Zielinski
RCA
Marne Highway M/S 138-2
Moorestown, N. J. 08057

SIG Tape Collection
Glen Everhart
RCA Government Systems Division
Route 38
Cherry Hill, New Jersey 08358

SRD
Bob Turkelson
NASA/Goddard Space Flight Center
Mail Code 935
Greenbelt, Maryland 20771

RSX Realtime
Ed Cetron
University of Utah
Center for Biomedical Engineering
3168 Merrill Engineering Building
Salt Lake CIty, Utah 84112

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

Cheap Networks
Evan Kudlajev
Philadelphia Electric Company
P.O. Box 8699
Phildelphia, Pa. 19101

The Unsupported Version working group had their best Symposium ever. Projects of mutual interest to those attending that were discussed included: 1) Exchange of documentation sets among users; also collections of self teaching material. Note: Users getting ready to throw away documentation sets please contact Bill Burton so I can recycle sets to users needing them. 2) Evaluate and collect v3.2 SPRs which were collected by Ralph Stammerjohn. More volunteers for reviewing the material are needed. 3) Exchange patches and problem notes on old versions. There were several problems (bugs) discussed. People presenting solutions will be submitting articles to Multitasker, covering changes to retro-fit Bonner lab Runoff, SRD, etc. on old versions of RSX. 4) Discussion of Anaheim symposium.

The System Performance and Accounting working group is continueing its work in preparing the index of the past RSXSIG tapes in regards to System performance and accounting. There is much interest in continuing the development of performance measuring tools. Further contact with DEC in regards the working groups goals is being pursued.

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

Future submisssions to the tape is being considered.

The SIG Tape working group helped create the Spring 84 Symposium SIg tape. Work started around 10:00 and continued until around 3:30 in the morning. A 40,000 block tape was the result. Due to tape reading problems the new KERMIT submission was not put into the Prerelease tape. If a good copy can be obtained it will be included in the tape that goes out to the tree. A complete report will follow in the multitasker.

The SRD Working Group discussed the current work being done, the current wish list, and plans for sending each member an updated wish list for comments. A user from the IAS community volunteered to help make sure that our version works on IAS. Work will continue on the various wish list items. Items of interest to SRD users will be published in the Multi-tasker.

The Real time working group has reorganized under the leadership of Ed Cetron of the University of Utah. Ed has considerable background in the area of real time and is on the steering committee of the Labs SIG. At the Real time working group meeting discussions were held on the scope and direction of the Real time working group. Discussions of the interest and involvement of DEC in this area were brought out. Site specific applications were brought out and a general discussion session was held later.

The Runoff group has continued its effort to consolidate desirable features of several versions of Runoff into an "official" version. Chuck, due to work efforts, is finding the time to upkeep the working group more and more difficult. If there is anyone willing to help in this position, please get in touch with Chuck.

The Cheap Networks working group got off the ground with the volunteer effort of Evan Kudlajev who volunterred to be the chairman of this group. An evaluation of the current free software for networking applications is the group's first task. Further work in the area of layering a mail application on this software was discussed.

The formation of the computer aided instruction working group still awaits the appointment of a working group chairperson.

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

## INTERMOUNTAIN AREA LOCAL USERS GROUP NEWSLETTER

### FROM THE EDITOR

Thanks to Ed Cetron (who also has two articles in this issue) for submitting this. Ed, all I got from the tape was this Q and A session. What happened to the RSX to VMS article?

### Questions and Answers

1. Where do we get a technical document or manual for the RQDxx devices?

   No technical manual is available, however EK-LEP02-0M-001 is a users guide and is available.

2. How many disk drives per RQDX1 controllers and how many controllers per backplane?

   Maximum of two (2) physical drives per controller. Depending on the drives, this can be 2,3 or 4 logical units (RD5x units have 1 logical unit per physical drive, RX5x units have 2).

   Currently, the RQDX1 controller terminates the DMA lines of the Q-bus. Therefore, only one RQDX1 is supported per backplane and it must be the last DMA device on the bus.

### NOTE

FLASH

It is rumored (in trade magazines) that this limitation will be removed in the future, and multiple controllers per backplane will be supported.

3. Is it possible to upgrade a Rainbow 100 to a Rainbow 100+?

No, the 100+ requires a new, improved mother board. The 100+ 10Mb upgrade kit will upgrade the 100 to a hard disk system with heavy duty power supply, but will not give added memory required to support the hard disk.

4. Why do Micro-PDP/11 winchester disks 'go-to-sleep'?

Bug in operating system under RT-11 which allows the disks to somehow disappear for a while (sleep). Patch for RT-11 version 5 will be available soon.

5. What does Micro-RSX have that RSX-11M+ does not?

Micro-RSX uses less memory and disk space than 11M+. It allows the use of named directories. Micro-RSX can also read P/OS disks. Micro-RSX comes with DTE (data terminal emulator) and allows file transfers to larger 11M and 11M+ hosts. It is NOT as easy to sysgen as 11M/M+ but much easier to install without a sysgen.

6. What is available for the new PRO's?

A new version of P/OS is out (V1.7). From V1.5 on, DCL was bundled with the Tool Kit. This meant that from here on, true multi-tasking was, and is, possible. V1.7 will have DCL standard and will not only support multi-tasking will support multi-user activities (if you know what you are doing and a few tricks). Apparently, Printer port and Communications port are TT2 and TT3.

7. Why Micro-RSX instead of RSX?

More efficient use of resources on small machines. Also for the novice, Micro-RSX is easier to install and learn. Con (online reconfiguration task) makes hardware changes trivial. Micro-RSX is heavily into cluster and reslib environments to fit small memory machines. In addition, 11M/M+ is about 10 times as expensive.

8. What is in Micro-VMS?

Lots of VMS..... 'lean and mean' version of VMS. No exotic stuff (missing weird symbionts like for big printers). No RSX compat mode, and some instructions emulated in firmware instead being in hardware. Otherwise, very similar.

# MEMBERSHIP STATISTICS

Terry Medlin
Survey Sampling, Inc.

Did you know that:

The RSX SIG has 10,124 members. There are 3,571 subsribers to the Multi-Tasker. The SIG is adding 90 members per month.

That is an impressive set of figures which clearly demonstrates the interest in our SIG. Would you like to become more involved in the SIG?

Liz Bailey and Jeff Hamilton both play vital roles in the SIG and both would love to hear from you. There are many tasks to be accomplished to make this SIG operate.

# MULTI-PROCESSORS YES

Terry Medlin

As many of you know, DEC once developed prototypes of a product called the 11/74. This product was a multi-processor based on the 11/70. It was never released although at least two systems are still operational.

When you consider the new 11/73 and the J-11 chip, it is not difficult to dream of a new multi-processor product based on the new technology. Why don't we have such a product? Simple, we have not PROVEN to DEC that such systems will sell.

In order to move in this direction, we ultimately need evidence that such as a system would sell. For those of you interested in the development of such a product, I want you to do the following:

1. Send me a letter indicating your interest. Include your address and phone number. Please do NOT call me - I can't handle all the calls I get.

2. Determine who in your organization authorizes procurements of this type. Meet with this person and get him to send me a letter indicating

- when you need such a product
- what does a multi-processor mean to you
- how would you use it
- how many would you buy over the next five years
- whether you need Q-bus or UNIBUS versions

3. Determine if there are other branches of your company to which item 2 applies and pursue them too.

My address is

Terry Medlin
Survey Sampling Inc.
180 Post Road East
Westport, CT 06880

I have met with several of the technical people who would be involved in the implementation of this "future" product. I have agreement from them that they are receptive to the idea. With the information furnished above, we CAN affect the development of such a product.

DON'T BLOW A GOLDEN OPPORTUNITY!

Terry Medlin

## 18-Bit Versions of M-PLUS

Terry Medlin
Survey Sampling, Inc.

Some of you may have the desire or need to test software or play with RSX-11M-PLUS but you only have an 18-bit processor. This article will describe how you can generate an 18-bit version of M-PLUS. The result will not provide a "many-user" system unless you really play with the partition structure but I used the technique to generate M-PLUS on an 11/23 system which had 124K words of memory and no 22-bit controllers. I used the V2.1 distribution but the same technique will probably work for V2.0.

As you proceed through the sysgen process, you will need to tell the command file what processor you have. I chose the 11/23-PLUS option. When it asked me for the memory size, I selected the minimum answer realizing that I would change this. At the point where sysgen allows you to edit the parameter files, I

took that option and edited [11,10]RSXMC.MAC to make two changes:

1. Change the memory size to 124K words.

2. Insert a ";" in front of M$$EXT to disbale extended addressing. Note that the ";" turns the line into a comment.

I proceeded through the rest of sysgen as one normally does.

After all tasks were built, I have to make some changes to SAV and BOO. As distributed, the code unconditionally enables 22-bit mode. I simply commented the instructions involved in this process, assembled the modules, replaced the modules in the libraries, and rebuilt BOTH SAV and BOO. I then reVMRed the system and booted in the new system using RSX-11M V4.1 under which I was running.

This process as desribed was not difficult for anyone familiar with the sysgen process. The final "product" is not supported by DEC but .... you know that anyway. Right?

I hope this information may be useful to you. If you try it and it provides a useful tool, please write an article back to the Multi-Tasker. If there is enough interest, MAYBE we can get DEC to provide support for 18-bit M-PLUS systems.

## Notes on DL: drivers, RMD and Idle Terminal Monitors

Edward J. Cetron
Center for Biomedical Design
3168 MEB, Univ. of Utah
Salt Lake City, Utah 84112

From the Author

This article was first submitted sometime in Jan. of 1983.... It was lost in the shuffle of newsletter editors and that dry period of Multi-taskers. I finally found it and have resubmitted it. I will apologize in advance for the few instances where the material has become somewhat dated. - EJC

I've been meaning to write this article for quite some time now, but, as usual, I just never got the time to type up and organize some of the interesting modifications I've made to me system. Having just returned from the Fall '82 symposium, with all the comments I received about these modifications, I decided now was the time to finally write the article. (Does anyone know a good chiropractor -- my arm still hurts.)

## DL: drivers and assorted features

In an previous issue, some discussion was made about rumors of dual RL02 controllers not working under 11M V4.0. Not having that problem, I simply ignored it. Unfortunately, we had to modify an 11/23 Q-bus system into an 11/23 Q-22 (yes, we know it voids the warranty -- but it is only four wires, or so we thought). Well, the modifications were made, the baseline system was booted and sysgen completed. The newly generated system would not boot - in fact it would run all over memory, and we could not even generate a crash dump.

We knew that the hardware worked since the baseline system ran perfectly. And the new system would boot and function properly on our 11/44 system. After a great deal of anguish and frustration, we decided, like all good systems people, to read the driver code. It seems the baseline system is, for compatibility reasons, an 18-bit system. Therefore, the extra 4 bits (for Q-22 or 22-bit Unibus) are ignored and our system would work just fine. But what happens when the new system, configured to be a 22-bit system tries to boot?

We learned quite hastily that our RL02's were DMA devices. That is to say the when they wanted to transfer information to memory, they would 'disable' the CPU and, ASSERTING ALL 22-BITS, write directly into memory. But our controllers only used a 16-bit address register -- where were the other bits? It turns out that 2 additional bits are stored in the CSR of the drives. The last four bits are stored by a hardware extension on Unibus machines called the Unibus Map or Unibus Mapping Registers. Since Q-22 systems don't have a Unibus, Digital decided not to give them a Unibus Map (logical, for once).

Instead, they designed a new RL02 controller with 5 device registers instead of the standard 4 and dedicated the fifth register for holding those extra 4 bits. Obviously, we didn't have the 5 register controller. And with the new transportablity over all CPU's of 11M V4.0, 11M expected 5 registers -- sort of.

After reading the code, we realized how, in general terms, the drivers decide which DMA technique to use:

1. 22-bit system?

no --- use old 18-bit DMA (and our baseline worked)
yes -- continue

2. Q-22 or Unibus?
try to access the fifth register; did it work?
yes -- Q-22 so use fifth register for DMA
no --- Unibus so ignore the high 4 bits and
let the Unibus Map handle it.

We now realized that this explained three related problems. First, in our 11/23 system, the driver tried to find the fifth register, failed and then tried to do Unibus DMA. With no Unibus Map, the most significant 4 bits were never set and our DMA transfers went, well... all over the place. Solution: we obtained the RLV12, 5-register controller and all is well.

The second problem that it explains is why our 11/44 system's error logger keeps telling me that my controller has 5 registers -- it thinks its on a Q-22 system. Oh well.... I do understand however, that the patch is on its way. It also explains why 2 controllers on a Unibus system might not work.

When the addresses of multiple controllers are selected, common practice is to put the CSR's of the second controller immediately after those of the first controller. Well, remember how the driver checks to see if it is on a Q-22 system? Yes, it goes out to the first controller, counts out where the fifth register would be, and lo and behold there is something there. It matters not that this is the first register of the SECOND controller, the driver know thinks it is running on a Q-22 system and things really get messed up. Solution: leave a blank address or four between multiple controllers.

## RMD on large memory systems (11M V4.0, V4.1, 11M V2.1)

In the last issue of the Multi-Tasker, it was suggested to use $SYSIZ to modify the size of an RMD display. This is an extremely dangerous procedure and since some tasks and executive routines use this value your system will tend to spontaneously crash quite often (and sometimes quite spectacularly). Our 11/44 has 3.75 Megabytes of memory and the RMD display is quite crowded; if not completely written over itself. At the Center for Biomedical Design, I made a few patches to improve the readability of the RMD display:

1. Treat VT102's, VT105's, and VT125's as 132 columm devices.

2. Move the IN:, OUT: fields up to the SECPOOL area to prevent their being overwritten; this patch will have to be removed for M-plus compatibility. ( I move them past SECPOOL for M-plus )

3. Provide a command to allow dynamic sizing of the memory page.

4. Include the patches provided in the last issue to use the I/O page when using 11M.

The following SLP file will change all the appropriate routines. It is designed to be used with the command file which follows. The general idea is to rename all the appropriate files from .MAC to .OLD (thus preserving the distribution kit for the inevitable DEC patches). The .OLD files are then modified into .MAC form by SLP, assembled and placed in the library (RMD.OLB) out of which RMD.TSK is produced. These files are only templates and some modification of device assignments and UIC's will be necessary depending on your distribution kit.

While I did attempt to comment the new files, more comments mean more typing in the .SLP files that can be mistyped and ruin the checksum checking. Therefore, it is strongly recommended that you read the newly generated source code -- it is well documented already. If you have any problems please let me know. For those of you wondering, the following .SLP file can be entered as one file or as multiple files.

NOTE

With a tape drive on order, I will finally have in-house access to a tape drive. I have removed the .SLP files from this article and will instead submit all the .SLP files and a .TSK image for 11MV4.1, and 11M+ V2.1B on the SIG tape in Cincinnati. For those who can't wait, send me a blank tape and a return mailer, postage if you can afford it would be appreciated.)

## Notes on the Idle Terminal Monitor

In our location, the console terminal is always in use. Even though its use is restricted (only myself), we would still like to have it come under the purview of the Idle Terminal Monitor. Somehow, BIGBRO (short for BIG BROTHER, our name for the Monitor)

must be able to selectively identify certain tasks as what I call 'ghost' tasks. This will enable BIGBRO to skip those tasks that are normally running from TT0: as the console and allow logging off TT0:. This feature is also needed since MCR... remains 'assigned' to the last terminal that used it. This could prevent the logoff of an idle terminal on an idle system.

In addition, I have added some code to set all non-logged in terminals to a site-specific CLI (which may someday appear in these pages) which accepts input from logged out terminals for WHO, RMD, HEL(LO), HELP, and a password controlled ABO(RT). Additional code has been added to reduce to non-privileged any privileged terminal at the issuance of the first warning. I don't have the source for BIGBRO as originally written by Bruce Mitchell so I can't generate a .SLP file. Instead, I've included just my modifications with enough of the existing code so that it can inserted at the correct place (sort of a CMP TI:=SY:USER.MAC, MULTI-TASKER HARDCOPY).

First, the changed sections of USER.MAC, each changed section is delimited by asterisks:

FROM THE EDITOR

I apologize to ED: I had to delete spacing from his comments, which were neatly formatted, to fit things within the less-than-80 column width that we print. In some cases I had to move a comment onto the following line.

```
   .title  USERMN  User Terminal Monitor
   .IDENT  /X01.03/
; USERMN - User Terminal Activity Monitor for RSX-11M/M+
;
;
; Author:
;  Bruce R. Mitchell
;
;  Patterned after the POOL monitor task supplied by Digital
;  Equipment Corporation on RSX-11M-PLUS V1.0 Autopatch E.
;
;
; Source Site:
;  Engineering Systems and Technology Laboratory
;  3M Company, 3M Center, St. Paul, Minnesota  55144
;
; Revision History:
;
;  31-dec-81       First version written
;  3-jan-82   Command file generates Ascii strings for time
;  5-jan-82        In-house version released
;  9-jan-82        Decnet HT: support conditionally added
;  11-jan-82       Decus version released
;  20-jan-82       VT100 support conditionalized
;  27-feb-82  Warning messages cleared for active terminals
;  15-mar-82       Correct conditionlization for HT's
; ************************************************************
;  21-Jun-82       Modified to fit CBD's 11/44 system
;                  First warning sets terminal no-priv
;                  Added software to log off TT0:
;                          by using 'ghost' tasks
;                  Added loop counter
;  25-aug-82       resets non-logged in terminals to the
;                          non-logged in CLI (lgocli)
;

; ************************************************************
   .mcall  wtse$           ; wait for single event flag
   .mcall  scli$           ; set cli
;
; ************************************************************
   .page
   .sbttl Local Variables
;
;
;  Local variables
;
lopcnt:    .word   0                ; loop counter
;
;
datbuf:    .blkw   8.               ; Buffer area for date and time
; ************************************************************
```

```
;  spawn MCR to log out terminal at address MCRTRM, setting
;          event flag SPNEFN on exit or status emmission
mcrspn:    spwn$   MCR...,,,,,,spnefn,,,mcrbye,byelth,,TT
mcrbye:    .ascii  /bye/              ; MCR command to logout terminal
byelth     = . - mcrbye               ; length of command
   .even

setspn:    spwn$   MCR...,,,,,,spnefn,,,mcrset,setlth,,TT
mcrset:    .ascii  \set /nopriv=ti:\
                   ; MCR command to de-privilege terminal
setlth     = . - mcrset               ; length of command
   .even

lgoset:    scli$   LGOCLI,TT,0
sleep:     stse$   efn2
                ; goto sleep and wait until next check time
spnwai:    wtse$   spnefn  ; wait for spawn or timeout event flag
;
;
;
ghosts = 3                    ; number of tasks the are NOT considered active
ghstnm:    .rad50  /COT.../           ; name of the first task to ignore
   .rad50  /BIGBRO/           ; ignore us
   .rad50  /MCR.../           ; and let us forget this too!
;
;
; ***************************************************************
;
warnl:     .ascii  <cr><lf><lf><lf>
   .ascii  /From BIG BROTHER:  First idle /
   .ascii  /terminal warning!/<cr><lf><07>
   .ascii  /                     /
.if df v$tl00                     ; if vtl00 support
   .ascii  <esc><133><65><73><61><155>
                              ; esc[5;1m (bold and blink)
.endc
   .even
   .word   timel              ; remaining time
   .ascii  / minutes/
.if df v$tl00
   .ascii  <esc><133><60><155>     ;esc[0m
.endc
   .ascii  / before forced logout./
   .ascii  <cr><lf>
wrnlln     = . - warnl;
;
;
   .even
warnp:     .ascii  /                     Terminal /
   .ascii  /now non-privileged/<cr><lf><07>
   .ascii  <cr><lf><lf>
wrnpln     = . - warnp
   .even
```

```
; ***************************************************************
;
   .even
banzai:    .ascii  <cr><lf><lf><lf>
.if df v$tl00
;          make a terrible looking display on the screen
   .ascii  <esc><133><61><73><61><110>     ; esc[1;1H
   .ascii  <esc>/#9/                       ; esc#9
   .ascii  <esc>/[>3;11;0;14;132J/
                        ; clear lines 11-14 inclusive
   .ascii  <esc><133><65><155>             ; esc[5m
   .ascii  <esc>/[12;26H/ ; move to row 12, col 26
.endc
   .ascii  /From BIG BROTHER:  BYE, BYE/<cr><lf>
.if df v$tl00
   .ascii  <esc>/[13;26H/
.endc
   .ascii  /      -- Idle terminal logout/<cr><lf>
.if df v$tl00
   .ascii  <esc><133><60><155>             ; esc[0m
.endc
   .ascii  <cr><lf><lf><lf>
banzln     = . - banzai
;
; ***************************************************************
;
;
;  switch to system state and check active task list
;
   call    $SWSTK,compar            ; switch to system state
                                    ;  and return at compar
   mov     $acthd,r0  ;; r0 points to active task listhead
23$:       mov     #ghstnm,r3       ;; get basis of ghost names
   mov     #ghosts,r1               ;; get number of ghosts
   beq     30$                      ;; ignore if no ghosts
25$:       cmp     (r3)+,t.nam(r0)          ;; is this task a ghost?
   beq     26$     ;;  maybe, first three letters match
   inc     r3      ;;  no, skip second word (three letters)
   inc     r3                       ;;  ( by two bytes)
   br      27$     ;;  and then go to the next ghost
26$:       cmp     (r3)+,t.nam+2(r0)
                   ;; do the second three letters match?
   beq     70$                      ;;  yes, so ignore it
27$:       sob     r1,25$                   ;; no, so try again...
30$:       mov     t.ucb(r0),r3     ;; r3 points to UCB for this task
   mov     u.dcb(r3),r1    ;; r1 points to DCB for the UCB
; ***************************************************************

;  set up pointers for next active task, check for list end

70$:       mov     t.actl(r0),r0    ;; point to next active task
   tst     r0                       ;; is this the last task?
   bne     23$                      ;; no, so get more
```

```
; ***************************************************************

.if ndf rs.nsl
45$:    bis     #tm.log,trmdat(r2)      ;; set logged-in flag

.iff
45$:    tst     r4                      ;; TT: or HT:
   bne  50$                     ;; if an HT:, go to it
   bis  #tm.log,trmdat(r2) ;; set flag for logged-in TT:
   br   55$                     ;; go for next unit
50$:    bis     #tm.log,netdat(r2) ;; set flag for logged-in HT:
.endc

55$:    bit     #u2.prv,u.cw2(r1)       ;; is this unit priv'ed
   beq  60$                     ;; no, look at next unit

.if ndf rs.nsl
   bis  #tm.prv,trmdat(r2)      ;; set priv'ed flag
.iff
   tst  r4                      ;; TT: or HT:
   bne  555$                    ;; if an HT:, go to it
   bis  #tm.prv,trmdat(r2) ;; set flag for priv'ed TT:
   br   60$                     ;; go for next unit
555$:   bis     #tm.prv,netdat(r2) ;; set flag for priv'ed HT:
.endc

60$:    add     #4,r2   ;; r2 points at next terminal flag word
   inc  r5
   add  d.ucbl(r0),r1   ;; r1 points at next UCB on DCB
   sob  r3,40$          ;; loop for each UCB on DCB
; ***************************************************************
compar: mov     #<itt*4>,r0  ; r0 counts down terminal data blocks
   mov  #itt,r5       ; r5 counts down terminal number
10$:    bit     #tm.log,trmdat(r0) ; is this terminal logged-in?
   bne  20$             ; if so, skip and process it
   mov  #"TT,lgoset+s.cidv      ; set SCLI$ DPB to TT's
   mov  r5,lgoset+s.ciun        ; set SCLI$ DPB unit
   dir$ #lgoset ; set not logged in terminal to lgocli
   clr  trmdat+2(r0) ; not logged-in, clear idle counter
   br   40$             ; and set up for next data block

;  terminal logged in; check for active task

; ***************************************************************
   mov  #iht,r5
50$:    bit     #tm.log,netdat(r2)      ; is this HT logged-in?
   bne  60$             ; if so, skip and process it
   mov  #"HT,lgoset+s.cidv      ; set SCLI$ DPB to HT's
   mov  r5,lgoset+s.ciun        ; set SCLI$ DPB unit
   dir$ #lgoset ; set not logged in terminal to lgocli
   clr  netdat+2(r0) ; not logged-in, clear idle counter
   br   80$           ; and set up for next data block
```

```
;  HT logged in; check for active task

; ***************************************************************
.if df rs.nsl
   clr     r3              ; assume terminal type is TT:
.endc

   mov     #"TT,asslun+a.luna ; set device to TT for alun$
   mov     #"TT,mcrspn+s.pwdn ; set device to TT for spwn$
   mov     #"TT,lgodev ; set device to TT for console message
10$:       tst     trmdat+2(r2)    ; is this an idle terminal?
   beq     60$             ; yes, set up for next terminal

; ***************************************************************
70$:       clr     r1                      ; r1 is HT: number
   clr     r2              ; r2 is offset in HT: data block
   mov     #1,r3                   ; assume HT: type is HT:
   mov     #"HT,asslun+a.luna ; set device to HT for alun$
   mov     #"HT,mcrspn+s.pwdn ; set device to HT for spwn$
   mov     #"HT,lgodev ; set device to HT for console message
80$:       tst     netdat+2(r2) ; is this an idle HT: terminal?
   beq     130$                    ; yes, set up for next HT:

; ***************************************************************

;  all terminals processed, back to sleep

140$:      dir$    #timdpb                 ; set wake-up call
   inc     lopcnt  ; say we finished another iteration
   mov     lopcnt,r0               ;
   dir$    #sleep                  ; and go to sleep
   jmp     tsks    ; rise and shine and do it all again
; ***************************************************************

   .sbttl  warone issue first warning
;
;  warone - issue first warning and de-privilege the terminal
;
;  inputs: r1 - target terminal number
;          r2 - data block offset
;          r3 - 0 -> TT:,   1 -> HT:
;
;  outputs: terminal flag word is updated
;
;  registers are not save or restored
;
.if df rs.nsl
warone:    tst     r3                      ; is this an HT:?
   beq     5$                      ; a TT:
   bit     #tm.lst,netdat(r2) ; an HT:, has first message been sent?
   beq     10$                     ; no, so send it
   return                          ; yes, so don't bother
5$:        bit     #tm.lst,trmdat(r2)
```

```
                    ; a TT:, has first message been sent?
.iff
warone:     bit     #tm.lst,trmdat(r2)
                    ; has the first message been sent?
.endc

   beq      10$                         ; no, so issue it
   return                               ; yes, don't do it again
10$:        mov     r1,asslun+a.lunu
                    ; load terminal number in alun$ DPB
   dir$     #asslun ; assign lun 2 to target terminal
   bcc      20$     ; if it succeeded, issue message
   return                               ; if failed, punt
20$:        mov     #warnl,brodpb+q.iopl
                    ; load message address into qio DPB
   mov      #wrnlln,brodpb+q.iopl+2
                    ; load message length into qio DPB
   dir$     #brodpb                     ; send out message

.if df rs.nsl
   tst      r3                          ; is this a TT:, or an HT:
   beq      25$                         ; TT:, so go to TT: code
   bis      #tm.lst,netdat(r2)
                    ; HT:, set first message sent flag
   bit      #tm.prv,netdat(r2)          ; was HT: priv'ed
   beq      40$                         ; no, so leave
   br       30$                         ; yes, rejoin common code
.endc

25$:        bis     #tm.lst,trmdat(r2)
                    ; TT:, set first message sent flag
   bit      #tm.prv,trmdat(r2)          ; TT:, was TT: priv'ed
   beq      40$                         ; no, so go home
30$:        mov     r1,setspn+s.pwvt
                    ; load terminal number into spwn$ DPB
   dir$     #setspn                     ; de-privilege terminal
   bcc      35$                         ; if ok, good
   return                               ; if not, punt
35$:        dir$    #spnwai             ; wait for spawn to complete
   mov      #warnp,brodpb+q.iopl
                    ; load message address into qio DPB
   mov      #wrnpln,brodpb+q.iopl+2
                    ; load message length into qio DPB
   dir$     #brodpb                     ; send out message
40$:        return                                ; and go home
; ***************************************************************
```

And now the sections from USERPRE0.MAC (in the same format):

```
;
; USERPRE - User Terminal Activity Monitor prefix file
;
;  Companion source file to USER.MAC
;
; Author:
;  Bruce R. Mitchell
;
;  Patterned after the POOL monitor task supplied by Digital
;  Equipment Corporation on RSX-11M-PLUS V1.0 Autopatch E.
;
;
; Source Site:
;  Engineering Systems and Technology Laboratory
;  3M Company, 3M Center, St. Paul, Minnesota  55144
;
; Revision History:
;
;  1-Jan-82         Source ripped out of SLPRE.MAC
;  2-Jan-82         Warning message bit definitions added
;  21-Jun-82        Modified to fit CBD's 11/44 system
;                   Added software to log off TT0:
; ************************************************************
;  Bit masks for terminal and task characteristics
;
tm.log      =       1               ; terminal logged in
tm.tsk      =       2               ; task active on terminal
tm.mcr      =       4               ; terminal CLI is MCR (not used)
tm.1st      =       10              ; first warning message sent
tm.2nd      =       20 ; second warning message sent and demoted to
                           ;  non-privileged status
tm.3rd      =       40              ; final warning message sent
tm.prv      =       100             ; terminal is privileged
; ************************************************************
```

These corrections should work. For those who don't like to type (like me) these changes will be on the next SIG symposia tape as I mentioned above. If time permits, I will document up our logged-out terminal monitor and include it also. Maybe even a CLI that enables non-privileged users to RUN programs as a privileged user.....

# COMPARISON OF COMMAND LINE INTERPRETERS

# MCR, DCL, TDX and CCL

AUTHOR: Allen A. Watson
BERGEN RECORD
Hackensack, NJ

FROM THE EDITOR

This article is a reprint of the session notes handout from the Spring, 1983 DECUS Symposium in St. Louis. The versions of RSX11M/M+ in the field at that time were M 4.0 and M+ 2.0. The comments on TDX, which was updated in the "point one" versions, should be taken with a grain of salt; the TDX documentation in the latest releases should be consulted for accurate information on TDX. In addition, a later release of CCL has appeared on the RSX SIG tapes (in fact a release is on the Spring 1983 tape). That release incorporated some of the changes I made to CCL at THE RECORD, and also, I believe, included a skeleton CCL such as I describe in this article.

SYNOPSIS: This article will discuss the following topics:

1. How CLI's function
2. CLI's available
3. Advantages and Disadvantages of MCR
4. Advantages and Disadvantages of DCL
5. Advantages and Disadvantages of TDX
6. Advantages and Disadvantages of CCL
7. CLI configurations and combinations
8. Tailoring CCL for specific application packages

## 1.0 HOW CLI'S FUNCTION

### 1.1 What A CLI Does

A Command Line Interpreter (CLI) is a program that accepts unsolicited terminal input and attempts to interpret that input as a command to the operating system. A CLI is the user's interface to the operating system. The way the CLI behaves in response to a user's input determines the user's perception of the system. Is the system friendly? Is it helpful? understanding? obscure? forgiving? annoying? threatening?

The CLI is the most important piece of software in your system as far as determining user reaction to the system.

Primarily, the CLI interprets the user's input and in response causes the execution of appropriate tasks. If parameters are required for a task, the CLI passes these parameters to the task. In some cases, it prompts the user if he forgets parameters.

Interpreting the user's input may mean, for some CLI's, a translation process. For example, DCL converts the input "COPY FILEA FILEB" into an execution of the PIP program, passing it the parameters "FILEB=FILEA".

## 1.2 MCR, First And Basic CLI

MCR, the Monitor Console Routine, is the original CLI for RSX11M systems. Until RSX11M Version 4.0 or M-Plus Version 1.0 it was the only CLI provided by Digital.

MCR is foundational to RSX systems. Other CLI's, in order to execute tasks, must pass an appropriate command to MCR. Functionally other CLI's do little but act as translators between the user and MCR. In the end it is MCR that executes the tasks. (MCR and other CLI's may also execute some simple commands in internal code.)

MCR can initiate any and all functions of which RSX is capable.

## 1.3 How Other CLI's Function And Why

If MCR can "do everything" of which RSX is capable, why are other CLI's needed? (Some oldtimers would reply, "They aren't!")

### 1.3.1 Provide Simpler User Interface -

As RSX became more widely used, non-technical users began to complain about the arcane and frequently inconsistent syntax that MCR requires for its various commands. They asked questions such as, "Why do I have to say "PIP TI:=FILE'? It would be so much easier to say 'TYPE FILE'."

The VAX systems implemented DCL (Digital Command Language), which provided this kind of sensible, obvious easy-to-remember commands. RSX users wanted a similar easy interface. Some RSX users developed CCL -- distributed on the RSX SIG tapes -- to answer that need (among other needs). Meanwhile, Digital provided a limited DCL on RSX11M-PLUS Version 1.0, and recently distributed an improved DCL on the latest

versions of M and M-PLUS.

Alternate CLI's were created mainly to provide a simpler user interface, with commands that more closely represented the function they perform.

### 1.3.2 Add Additional Commands -

Alternate CLI's also allow users to create new commands, using tasks created by themselves as well as DEC utilities.

### 1.3.3 Prompt For Parameters -

Alternate CLI's can be written to prompt the user for omitted parameters and even, as with DCL, to provide for in-context help information. For example, consider the COPY command in DCL:

```
COPY
From?: FILEA
To?: FILEB
```

When "COPY" is typed in with no parameters, DCL prompts for the two parameters required. This helps the novice user remember in which order the "from" and "to" files must occur. If the user typed "?" (question mark) in response to a prompt, he would be given a screen of help information about the COPY command.

### 1.3.4 Translate To MCR Format -

When a DCL user types "DIRECTORY", DCL translates this to "PIP /LI" and passes this command to MCR. A DCL user does not need to learn the peculiar syntax of PIP. Another CLI might take the command "SQUISH MYFILE" and translate it to:

```
MUN SQU,MYFILE.TEC=MYFILE.TES/D:N/L:Y/B:Y/T:N/C:Y/W:N/A:Y/E:N
```

to invoke the TECO "SQU" macro; again, the command would be passed to MCR. This ability to greatly simplify command input is a major benefit of alternate CLI's.

## 1.4 Two Types Of Alternate CLI's

There are two types of CLI's, according to their relationship to MCR: primary (which receives commands before MCR) and catch-all (which receives them after MCR).

### 1.4.1 Primary CLI's -

A primary CLI is one that precedes or replaces MCR. A primary CLI gets "first crack" at interpreting a typed command. It examines and interprets it before MCR even sees it.

An alternate primary CLI can be designed in one of two ways: it can allow any commands it does not recognize to fall through to MCR, or it can simply refuse any unrecognized commands. For example, DCL does not recognize the command "PIP". If DCL is installed as primary CLI with fall-through enabled, when a PIP command is typed, although DCL does not recognize it, the command is passed to MCR anyway to let MCR try to interpret it. If fall-through is not enabled (which is the way Digital distributes it), DCL will respond "Unrecognized command" when you type in "PIP".

To use an alternate CLI as primary CLI it must be installed as a CLI and enabled. Here are the command lines used to install and enable CCL:

```
INS $CCL/CLI=YES/TASK=...CCL/PRI=160.
CLI /INIT=CCL/CPR="<15><12>/CCL>/"/DPR="<15><12/>/"
CLI /ENABLE=CCL
```

The CLI command is described in your MCR operations manual.

This makes CCL available to the system as a CLI, but as yet no users or terminals are assigned to use it.

To assign CCL to a given terminal, you use the MCR command:

```
SET /CLI=TTnn:CCL
```

To set a default CLI for a given user, so that wherever he logs on he gets that CLI, you can run ACNT and modify the "CLI" field for his logon. Note: if a user is assigned in the account file to an alternate CLI and that CLI is not installed and enabled, that user cannot log on.

### 1.4.2 Catch-all CLI -

A second type of CLI is called a catch-all. A catch-all CLI interprets commands after MCR has failed to recognize them. (Obviously a catch-all cannot contain any commands that are identical to MCR's commands; MCR will get a match first and the command will never reach the catch-all.)

When MCR fails to recognize a command it looks to see if a task called "...CA." is installed. If such a task exists, MCR passes the command line on to that task. Any CLI-type task, including TDX, CCL and DCL, can be installed under the name "...CA.". Of course, only one task can be installed with this name at any given time. A catch-all task does not have to be initialized and enabled with the CLI command; simply installing it suffices.

If MCR is the primary CLI and DCL is installed as catch-all, then when a user types a "COPY" command, MCR examines it first and fails to recognize it. MCR then passes the command on to DCL. DCL does recognize it, translates it to the appropriate PIP syntax, and passes that PIP command back to MCR. MCR can now understand and execute the command.

You can see there is a good bit of additional overhead in this technique. One of the decisions you must make in setting up your system is how much overhead you can tolerate for the convenience of your users.

A catch-all CLI can be written so as to look for a second catch-all. CCL, in particular, is written this way; it looks for an installed task called "...CA2" and passes the command on to that third CLI when it (CCL) fails to recognize the command.

## 1.5 You Control CLI Configuration

The main point of this session is to help you answer the questions, "Which CLI should I use as primary? Do I want a catch-all, and if so, which one?" When you bring up RSX you have choices and decisions to make that will affect how all of the users of your system will perceive it. The system as distributed has only MCR. Use of DCL is a SYSGEN option. CCL and TDX are not even mentioned in SYSGEN; you can add them, if you wish, after the system is generated.

Should DCL be primary? Should MCR be directly available, or available at all? What are the relative merits of the available CLI's? Should I bother with CCL? The answers to these questions are up to you, and will depend on the character of your installation.

## 2.0 AVAILABLE CLI'S

There are four CLI's that are readily available to RSX systems. MCR, DCL and TDX are CLI's that are included in your distribution kit. CCL is a CLI that is available on the RSX SIG tapes.

MCR is the basic CLI, and the only one that must be included in every RSX system.

DCL (Digital Command Language) is a CLI designed to be user-friendly, with commands that are more English-like. The RSX DCL is designed to be as nearly like the VAX DCL as is possible.

TDX is a catch-all task only that provides a few DCL-like commands for users who don't want all of DCL, a fairly simple way for users to add their own commands, and an automatic flying install for tasks in the system UIC.

CCL (Console Command Language) is a user-written, general-purpose CLI distributed on the SIG tapes. It had many authors and contributors over the years but was perfected by Jim Downward of KMS Fusion. CCL can function either as a primary CLI or a catch-all.

In the sections that follow we will discuss the relative advantages and disadvantages of each CLI.

## 3.0 MCR: ADVANTAGES AND DISADVANTAGES

### 3.1 MCR Advantages

MCR is the fastest CLI. Because CLI's usually translate commands into MCR commands and pass them to MCR, it is faster to execute MCR commands directly. If your users know MCR and understand its syntax you will save time by using MCR as the primary CLI.

MCR is the most complete CLI. As we said earlier, anything RSX can do can be done via MCR. This is not true with any of the other CLI's without some effort on your part.

MCR is the most familiar CLI to veteran users. Because it is the first and default CLI, most experienced RSX users know it.

### 3.2 MCR Disadvantages

MCR invokes many different tasks that have inconsistent command formats. One command may require commas between parameters while another may insist on spaces.

The commands MCR uses are not always "mnemonic" in nature; that is, the function they perform is not always logically related to the command. For example, "SET /LOG" sounds like it is setting something, while in fact it displays a list of logged-on users.

MCR does not prompt for missing parameters. In fact if parameters are omitted MCR will simply report a syntax error (at best), or try to execute without them.

MCR does not allow any abbreviations. Commands must be entered with the exact correct syntax. That syntax is in many cases difficult to remember.

## 4.0 DCL: ADVANTAGES AND DISADVANTAGES

### 4.1 DCL Advantages

DCL is Digital's standard CLI for RSX and for VAX systems. DEC has attempted in RSX DCL to provide a command language that is identical to VAX DCL, so that RSX users who migrate to VAX will be able to use commands they already know. The attempt is not perfect, but if, as an RSX user, you intend to migrate to VAX or add a VAX system, you should encourage your users to use and learn DCL.

DCL attempts to have consistent command formats, with input file names, output file names, and switch options coming in the same place in each command.

DCL mnemonics are more easily learned and more meaningful than MCR. Compare a few simple examples:

| DCL | MCR |
|---|---|
| COPY A B | PIP B=A |
| DIR | PIP /LI |
| RENAME A B | PIP B=A/RE |
| SET TERM VT100 | SET /VT100=TI: |

DCL will prompt for missing required parameters. If you issue an MCR command "PIP FILEA", you get an error message, "PIP -- Illegal command", and have to try again. If you issue a DCL command "COPY FILEA", you are prompted for the output file name and the command executes successfully.

DCL is also tied in to the system HELP facility, so that is provides in-line help at the parameter level. No other CLI does this.

DCL allows many convenient abbreviations, such as the following:

| | | | |
|---|---|---|---|
| ABORT | A | HELP | H |
| BROADCAST | B | HELP | ? |
| COPY | C | LOGOUT | LO |
| DIRECTORY | D | LINK | L |
| DEALLOCATE | DEA | MACRO | M |
| DEASSIGN | DEAS | PRINT | P |
| EDIT | E | RUN | R |
| FORTRAN | F | SHOW | S |

Almost all commands and switches can be expressed in three letters or less. The flexibility of accepting either long or abbreviated commands allows clarity in command files and ease of learning for the new user, while providing convenience to the more experienced user.

In earlier releases DCL was limited to a basic set of commands. In the current releases the command set has been expanded to include almost all of the Digital utilities and system functions. The average user can do anything he requires using DCL.

DCL is user-modifiable and extendable, and the means for doing so is well documented in the System Management Guide.

## 4.2 DCL Disadvantages

### 4.2.1 Absence Of Some MCR Functions -

Though DCL contains almost all of MCR's functionality it does not contain all of it. There are annoying little omissions (which DEC will probably clear up in future releases). For example, if you want to create a new version of a file when renaming it from another file, you can't do it simply. Assume you have two files, MYJOB.CMD;1 and MYJOB.TST;1, and wish to rename the .TST file to become a new version of MYJOB.CMD. Using MCR you can say:

```
PIP MYJOB.CMD/NV=MYJOB.TST/RE
```

With DCL, the logical command would be:

```
REN MYJOB.TST MYJOB.CMD
```

This produces an error message, however, telling you the "file already exists". And there is no "new version" switch for the RENAME command.

There are many ways around this, of course: specifying explicit version numbers in the RENAME (which requires that you know what they are currently), deleting the old file first, using COPY (which does

create a new version) and then deleting the .TST file -- but the point is, DCL is supposed to be more convenient than MCR and suddenly it is less convenient.

System programmers and other priveleged users especially will find that DCL simply does not have commands for certain functions. The functions can still be accessed (using RUN, perhaps) but once again, this is less convenient.

### 4.2.2 Subtle Differences From MCR -

DCL abounds with "false friends": commands that look like MCR commands but have a different syntax, or even a different meaning. The list of false friends includes RUN, REMOVE, INSTALL, DEALLOCATE, MOUNT, MACRO, and FORTRAN. People who already know MCR will respond badly to DCL because of these conflicts. Existing MCR command files will often have to be rewritten to handle the DCL syntax.

Nothing is more frustrating than typing in a long complex MACRO assembly line only to receive a syntax error because you forgot you were in DCL and used the MCR syntax. My programmers have simply refused to learn the LINK command, preferring the old TKB. If you know the old way you are going to want to use it and not be bothered with learning a new syntax to do the same thing.

### 4.2.3 Too Wordy -

The MACRO and LINK (task build) commands in DCL (for example) seem wordier than the old MCR syntax, especially on complex command lines. Even a simply command like "SET /SLAVE=TT14:" becomes "SET TERM TT15: SLAVE".

### 4.2.4 Inconvenient To Modify -

Just because Digital has clearly documented how to modify DCL does not mean that it is easy. First, you must be a MACRO programmer -- and if you are you probably despise DCL and won't be bothered. Second, you must learn a fairly complex second language called "Meta-macro" that is used in the DCL command tables. The fact that Digital attempts to explain Meta-macro through yet a third notation (BNF) tends to put up too many barriers to the casual programmer.

To create a new DCL command you must do the following:
- o Create a new source module in MACRO.
- o Modify COMMAND.MAC.
- o Modify DISPATCH.MAC.

- Add an overlay to the structure in DCLBLD.ODL and DCDBLD.ODL.
- Assemble your source module and the command and dispatch tables.
- Place the source module in DCL.OLB and replace COMMAND and DISPATCH.
- Rebuild DCL (twice!).
- Remove and reinstall DCL.

## 5.0 TDX: ADVANTAGES AND DISADVANTAGES

### 5.1 TDX Advantages

TDX is simple to install: just install it under the name "...CA." and it will catch any command MCR fails to recognize.

TDX provides basic forms of the most popular DCL commands: TYPE, DELETE, and DIRECTORY. It adds a CHD (change directory) command, equivalent to SET /UIC but with simpler syntax. And if the command passsed to TDX is not one of these, TDX attempts a flying install of a task having the name of the command and passes the rest of the command line to it.

That describes TDX on M Version 4.0 and M-Plus Version 2.0. In the "point one" releases, TDX will be enhanced to include further commands:

| | |
|---|---|
| SYS | SET /SYSUIC |
| DLG | Display logged-on users (SET /LOG) |
| SHQ | Show print and batch queues |
| CVT | Radix conversion |

CVT is a totally new function, not found in MCR. It accepts input in any form and displays it in decimal word, decimal byte, octal word, octal byte, hexadecimal, RAD50, and ASCII formats. It will also evaluate expressions.

In addition you can choose to handle unrecognized commands in one of two ways: by a flying install (as above), or by executing an indirect command file having the same name as the command. In the latter case TDX will look first in the login UIC, then the library UIC, and finally will try the same two directories looking for a file called CATCHALL.CMD. (To me this seems like a lot of overhead to avoid typing an at-sign!)

TDX is fairly easy to modify if you know MACRO. There is no special meta-language and just a simple task build.

In the latest releases, TDX has become a supported product.

## 5.2 TDX: Disadvantages

TDX has a very limited command set. It can help as a second CLI, but that is all.

TDX still requires assembly, task building, and re-installation to modify it or add commands.

In "point zero" versions of RSX, the documentation on TDX is skimpy; this may improve.

TDX functions only as a catch-all and is therefore slower than a primary CLI. For example, if you have only MCR with TDX as catch-all, a TDX command goes through the following steps to execute:

- MCR searches its internal command list.
- MCR searches the installed task list.
- MCR passes the command to TDX.
- TDX locates the command and translates it.
- TDX passes the translated command back to MCR.
- MCR executes it (doing one or both of the above two searches a second time).

# 6.0 CCL: ADVANTAGES AND DISADVANTAGES

## 6.1 CCL Advantages

I am an admirer of CCL, and this list of advantages reflects that. Try it yourself and see if you agree.

### 6.1.1 Easiest To Modify -

CCL is easier to modify than any of the other CLI's. (Modifying MCR takes an expert, by the way.) All you need to do to create a new command is to edit a text file. That's it: one step than anybody can do! You create a new command in seconds, with no assembly, no task building. When you first do this it seems like magic.

### 6.1.2 Simple Internal Syntax -

The syntax used in creating CCL commands ranges from extremely simply to only moderately complex. There is nothing difficult to learn; anybody can do it, even non-programmers. Basic CCL commands are no more difficult to create than writing a simple command file. For example, here is a definition for a "STATUS" command to show several system parameters:

```
$6600STATUS
+SET /POOL
+SET /PAR
*SET /LOG
```

Here is an example of a "BUILD" command that does a FORTRAN compile, task build, and run of a FORTRAN source program, complete with prompt if the filename is omitted.

```
$55llBUILD
?1File
+F4P %1.OBJ;1%=%1%
+TKB %1.TSK;1/CP/FP%=%1%,LB:[1,1]F4POTS/LB
+PIP %1.OBJ;1%/DE
*RUN %1%
```

### 6.1.3 Performs Flying Installs Selectively -

Unlike TDX, which attempts a flying install for every unrecognized command, CCL allows you to define which tasks you want treated that way; other commands are simply reported as unrecognized. You can actually run you system with nothing installed but MCR, basic utilities like PIP, and CCL. If you still have pool problems CCL can be your salvation. It was our salvation under RSXllM V3.2. Now that we are using M-Plus this feature doesn't mean as much because we install almost everything in secondary pool.

### 6.1.4 Prompts For Parameters -

CCL can prompt for missing parameters, and you control what the prompts are and when they are issued (e.g. prompt for all parameters if any are omitted).

### 6.1.5 Multi-line Commands -

A single word command can pass multiple lines to MCR, as in the examples above.

### 6.1.6 Invokes Command Files -

CCL can invoke command files and pass parameters to them. In this way one simple command can set off an entire chain of operations involving multiple tasks, user prompts and responses, and even conditional execution in the command file.

### 6.1.7 Primary CLI Or Catch-all -

CCL can be installed as the primary CLI, or as the catch-all, or both.

As primary CLI, CCL allows use of commands that MCR will not accept -- one or two characters commands, or commands with special characters such as "/FU" for a full directory or "?" for HELP.

As catch-all, CCL can provide DCL-like commands for users who prefer MCR as their primary CLI.

### 6.1.8 Application Level CLI's -

CCL can be installed under several different names with different command sets to provide application-level CLI's. This is so important that a later section is devoted entirely to discussing how and why to do this.

### 6.1.9 User-specific Commands -

CCL is the only CLI that allows a user to define commands that the system recognizes only for that one user. For example, I defined a "MEMO" command for myself that invokes in turn DOC (a RUNOFF preprocessor), EDT to edit the file, and RNO to format the output. Joe Blow could define his "MEMO" command to do something entirely different.

Application packages can thus be set up with multi-level security so that commands are restricted to certain logins, while other commands are available to the whole group. (It would be easy to modify CCL to recognize group command sets as well. Indeed, this was done in an earlier release.)

### 6.1.10 Very Thoroughly Debugged -

CCL has been around for years and is in use by at least hundreds of installations. The last version on the SIG tapes (Spring 1982) was Version 8.0, which lets you know it has been highly refined. Anything that gets through eight versions has had the kinks worked out.

### 6.1.11 Well Documented -

CCL documentation is good, clear, and complete.

I cannot end this section without a word of thanks to Jim Downward of KMS Fusion for his invaluable contribution to the RSX world in the refining and perfecting of CCL.

## 6.2 CCL: Disadvantages

CCL is not supported by DEC, and Jim Downward has since moved on to the VAX, so there is no longer any active support of CCL to my knowledge. Since there are no known bugs this is not a serious problem.

When CCL is installed as catch-all and the commands are defined in text files, it has the slowest response and most overhead of any CLI. However, frequently used commands can be assembled into the program quite easily. This still leaves the general overhead of a catch-all task. All I can say is, the convenience is worth it.

If you make use of the ability of CCL to do flying installs, you pay the price of the overhead of installing and removing tasks every time you run them. If you need the flying install capability, however, it is because you have a pool problem. The alternative is crashing your system. Is there a choice?

# 7.0 CLI CONFIGURATIONS

I haven't calculated how many possible combinations of these four CLI's are possible; it's a lot. In this section I will discuss thirteen of them that have particular advantages or disadvantages. The final choice depends on the needs of your installation.

## 7.1 MCR As Primary CLI

### 7.1.1 MCR Only -

If you are primarily a program development shop with technical types using the system, you may want to use only MCR. There is little reason, however, to not add at least one of the other CLI's as catch-all. Why not provide added capability?

### 7.1.2 MCR Plus DCL -

This gives you all of MCR and adds the convenience of DCL's English-like syntax for those who care to use it. Good for sites where most users already know MCR. You'll encounter conflicts on the "false friends", but the MCR syntax will take precedence. Be sure you build DCL to not fall through to MCR for unrecognized commands (see [1,20]DCLBLD.BLD) or you'll wind up in an infinite loop.

### 7.1.3 MCR Plus TDX -

Gives you the most useful DCL-like commands plus flying install. CCL's selective flying install is better, and DCL's command set is much fuller; TDX alone as catch-all has little to recommend it.

### 7.1.4 MCR Plus CCL -

Again you have all of MCR as primary CLI. CCL as delivered also includes many DCL-like commands built in (more than TDX), and you can add others as needed. You can also add your own specialized commands, including user-specific ones. However, you are missing some of the sophisticated prompting and defaulting of DCL. This is probably the best choice for a systems house that has little need of DCL and prompting, but wants the ability to extend the command set.

### 7.1.5 MCR Plus CCL Plus DCL -

CCL installed as catch-all will look for a second catch-all (...CA2) when it fails to recognize a command. If you install DCL as ...CA2 you can have all three at once: MCR, CCL and DCL. Remove the DCL look-alikes from CCL (a task build option) and those commands will (eventually) fall through to DCL as second catch-all. They will be slower to execute, of course. This is recommended only if usage of DCL commands will be fairly infrequent.

## 7.2 DCL As Primary CLI

### 7.2.1 DCL Only -

Not recommended because of DCL's limitations. You might want to do this if you desire to force your users to learn DCL in preparation for migration to VAX; I can see no other reason. To use DCL only you must build it so as to not fall through to MCR (the default is to not fall through).

DCL as distributed also includes an "MCR" command that forces the command to pass to MCR directly, e.g. "MCR PIP /LI". Thus even with DCL as the only CLI, you can still get to MCR if you must, but with a little extra effort. I recommend you leave this option in (it is a task build option) even if you are trying to push DCL; otherwise there are some things you just can't do.

### 7.2.2 DCL Plus MCR -

Not a bad choice. DCL as primary CLI executes quickly. MCR-type commands fall through to MCR (you must modify [1,20]DCLBLD.BLD to enable this). Users who know MCR will be able to use those commands without much trouble (except the false friends). And, if they really can't stand DCL, they can execute "SET TERMINAL MCR" and make MCR their only CLI. Less sophisticated users have the use of DCL.

Incidentally, you could build two versions of DCL, one with fall-through enabled and the other with fall-through disabled; install the first as primary CLI and the second as catch-all. Then users could choose either MCR or DCL as their primary CLI and still have full access to both!

MCR required to use catch-alls

Because MCR is the only way into a catch-all task, the configurations of "DCL plus TDX" and "DCL plus CCL" are not possible.

### 7.2.3 DCL Plus MCR Plus TDX -

Only if you don't want to use CCL can I see any benefit from this configuration. You can put your own commands into TDX with a little effort. But why not use CCL?

### 7.2.4 DCL Plus MCR Plus CCL -

This is the CLI configuration we have been using at THE RECORD for the last year. It seems to provide the best of all worlds, with all the benefits of all three CLI's, including user-specific CCL commands. The only problems that arise are the false friends when in DCL mode. I still find myself switching to MCR when I want to do mounts and installs, or using the "MCR MOU" form. We have retained the basic DCL commands in CCL so that DIR and TYPE work when users set themselves to MCR.

When you have multiple CLI's like this, you also must be careful when you write command files that may be used under either MCR or DCL. Use the "<CLI>" symbol to check which CLI is in use, and set the terminal appropriately to execute the commands in your file.

### 7.2.5 DCL-MCR-CCL-TDX -

I haven't found this useful (installing TDX as ...CA2) because the current TDX doesn't add a thing to CCL. When the CVT command is added I may throw TDX in as second catch-all.

## 7.3 CCL As Primary CLI

### 7.3.1 CCL Only -

CCL as the only CLI makes sense only when you have many specialized commands defined for CCL. For example, a particular user application package. As distributed, CCL will behave differently when installed as primary CLI or as catch-all; as primary CLI unrecognized commands fall through to MCR, while as catch-all, CCL passes them to ...CA2. (I published a patch to make fall-through function correctly for M-Plus systems, on the Spring 1982 SIG tape.) You must modify CCL to prevent the fall-through to MCR.

Using CCL as the only CLI is discussed more fully in Section 8.0.

### 7.3.2 CCL Plus MCR -

If you install the unmodified CCL as primary CLI this is what you will get. It might make sense, as I said, if your users were mainly executing commands defined in CCL and used MCR commands only occasionally. There is little reason not to add DCL as catch-all, however (the next option).

### 7.3.3 CCL Plus MCR Plus DCL -

Same as above with DCL added. Again, avoid this if you intend heavy use of DCL commands.

### 7.3.4 CCL Plus MCR Plus CCL -

You might want to do this if you had two types of users -- some who would primarily use CCL and therefore want that as primary CLI, and others who would use mainly MCR commands but wanted access to the CCL commands occasionally.

## 7.4 Other Combinations

There are many other combinations possible but none of the rest make much sense to me. You should be aware of these facts: anything installed as a catch-all task will be available to all users; the primary CLI can be different for each user. Thus combinations of combinations are possible; you could have some users with a "DCL-MCR-CCL" path and others with "CCL-MCR-CCL", and (using different task names) you could restrict certain users to either DCL or CCL only.

Whatever you do, you should investigate using CCL somehow.

# 8.0 TAILORING CCL FOR APPLICATIONS

If you wish to define a group of commands that is available to a specific group of users and only to that group, the best way is to use a special command line interpreter. The user's login accounts can then be altered with ACNT to specify this special program as their CLI. New CLI's can be created from scratch as described by DEC in the System Manager's Guide, or, more easily, by making minor changes to CCL and installing it under a different name. You can have up to sixteen CLI's in your system.

## 8.1 A Skeleton CLI

On the RSX SIG tape for Spring 1983 we will provide a stripped down CCL that can be used as an application-specific CLI -- in addition to the four CLI's already discussed. Using this skeleton I see little or no reason to ever write a CLI from scratch.

What we have done is:

1. Removed all built-in, DCL-like commands
2. Removed all historical comments.
3. Changed the symbols on the file definitions to globals so they can be changed by GBLPAT's in the task build file, and edited the task build file accordingly.
4. Made fall-through to MCR a task build option.
5. Added a built-in MCR command like that in DCL.

If you know MACRO it is quite easy to do this yourself, starting from the Downward Version 8.0 on the Spring 1982 tape. If you just want to build one new CLI you can edit in the changes to the file specifications and the MCR option directly instead of the generalized GBLPAT technique.

The result is CLISKEL.MAC. It is described in more detail in CLISKEL.DOC.

## 8.2 How To Do It

To see how easy it is to create multiple CLI's from CCL, you need to understand just how CCL works when installed as primary CLI.

When CCL receives a command it looks in its internal table, which is defined in a module called TABLE.MAC. This table is assembled and task built into the program. In the skeleton CLI provided it contains only the "MCR" command to provide direct passing of commands to MCR.

If the command is not in the internal table, CCL next looks in the current UIC for a file called USERCLI.CLI. If the file is found it is scanned for a matching command definition. If the file is not found, or the command is not in it, CCL procedes to look in LB:[1,5]SYSCLI.CLI.

To define new commands for all users of the CLI, all you have to do is enter them into SYSCLI.CLI. To define commands for individual users, you enter them in their version of USERCLI.CLI.

Now -- get this! To create a new CLI with an entirely different set of commands, all you need to do is:

1) change the names of the two files the program looks at, and
2) install the program as a CLI with a different name.

That's it! Simple enough?

To make this as simple as possible we have made the file definitions into global symbols, so that you can specify new file names in the task build command file using GBLPAT commands. An indirect command file, ASCOCT.CMD, will translate an ASCII file name into the appropriate octal words for use in the GBLPAT's. My suggestion is to change just the three-character file extension, making it the same as the three-character name you give to your new CLI.

Thus, a CLI called "...ABC" would access SYSCLI.ABC and USERCLI.ABC to find its commands.

## 8.3 Speeding It Up

Once commands have been defined and tested in SYSCLI, they can, if desired, be moved into the internal TABLE.MAC file. Downward has explanations of how to do this. You can use his TABLEGEN.CMD to translate the SYSCLI file into a TABLE.MAC file, or you can use the TECO macro TABLEGEN.TEC we have provided on the Spring 1983 tape.

## 8.4 Indirect Command Files

We have added to the skeleton CLI the ability to execute indirect command files. This is not part of standard CCL but is useful when using an alternate CLI as the only CLI. Be aware that since your CLI is primary, Indirect will use it (not MCR) to interpret the commands contained in the indirect file. You could also use an "MCR @filename" command that would pass "@filename" to MCR, but that command file could contain only MCR commands. There is no easy way to create command files that contain both MCR commands and your own application-specific commands unless you build your CLI to allow fall-through to MCR. Then you have to watch out for command name conflicts, that is, your CLI cannot use as a command anything that is already defined as a command in MCR (including names of installed tasks of the form ...xxx).

9.0 CONCLUSION

My contention is that a combination of CLI's is best, and that whatever combination you choose it ought to include CCL. My preference is DCL-MCR-CCL, but the choice you make will have to be based on the needs and abilities of the users of your system.

## Problem with DCL DELETE command in RSX V4.1C

Neil g. Johanning
Mechanical Technology
Boice Division
968 Albany-Shaker Road
Latham, NY 12110

FROM THE EDITOR

Mr. Johanning sent me a copy of his SPR to DEC which was not reproducable, so I have extracted the relevant information here.

The DCL DELETE command does not allow wild card specification on V4.1C but did on V4.1B. PIP still allows wild cards.

```
>DEL [*,*]*.OBJ
   produces error message
   "DELETE -- Illegal UIC: numerical expected."
>PIP [*,*]*.OBJ;*/SD works.
```

I have rebuilt DCL to enable fall through to MCR and flying installs. No other changes have been made.

# HELP YOURSELF

Israel asks for help

A. Goldberg
Environmental and Water Resources Dept.
Techion (Israel Technology Institute)
Haifa, ISRAEL

FROM THE EDITOR

Mr. Goldberg's letter required some translation (due to reasons he notes later in the letter), so this version is my attempt to interpret what he asks. Hope I got it all right.

Dear Mr. Watson,

I would first like to thank all the Multi-Tasker crew. You do a really good job, and it is a pleasure to read Multi-Tasker articles. In fact, everyone in my research group knows it is impossible to talk to me the day I receive the Multi-Tasker.

Here in Haifa I am the System manager of two PDP 11/23's, both of them working with RSX. One is a MNC/DECLAB-23. My job consists of programming real time, graphics, and utilities programs for the users. Since I'm a self-made programmer it is very helpful to be assisted by the Multi-Tasker and to not have to discover things like the indirect command language by myself.

The help I need now is on QIO FORTRAN programming on RSX, and on programming self-made HELP files. Both of them are explained in the RSX literature, but with no or few examples.

EDITOR'S NOTE

OK guys, there are two good topics for articles! How about some help for Mr. Goldberg that would also assist the rest of the RSX world?

I also have a couple of questions for your readers:

1. How can you perform the KED "LEARN" mode in EDT?

2. Does anyone have a cross-assembler for the Z-80?

3. Does anyone have a program for graphics working with RGL and PLOT-10 together?

I am sure that my English writing is a disaster, so please be good enough to correct it. My big problem is that my mother tongue is French, which gives me no problem reading and understanding English, but writing ... !!

Keep on with the good work.

A. Goldberg

## Reply from Allen Watson

I can answer one of your questions but unfortunately can't offer much help. First, let me thank you for the fan letter! I get very few and it's nice to know we are really appreciated.

DEC has been asked repeatedly to add the "LEARN" mode, present in KED, to the EDT editor. They steadfastly refuse, stating that there is not enough room in the task's virtual address space to be able to do so.

For learning QIO programming in FORTRAN I could recommend the Self-Paced Instruction course available from DEC, "Programming RSX-11M FORTRAN". The current catalog from DEC lists it as course number 88 , with each additional student package (order number 89). The course also covers executive directives for memory management, file I/O, intertask communication, static and dynamic regions, and file control services, and has many examples and exercises. The course comes with an 800 BPI mag tape that contains working examples. Order from:

Digital Equipment Corporation
Educational Services Department
12 Crosby Drive, BUO/E55-20
Bedford, MA 01730
USA

I like the idea of an article on writing your own help files; I've done a lot of it but have little time these days to write new RSX articles since most of my work now is on VAX. However, if no one else volunteers, I will try to grind out something in the next

few months. No promises, though.

## FROM THE EDITOR

I had three responses to Roger Jenkins' request for help on placing a double quote character into an indirect symbol. Two offer basically the same solution. The third, however, is the cleanest and to me the neatest, because it uncovers an undocumented feature of Indirect. And what's more, the author found it by making a mistake! That's serendipity.

## Placing a Quote in an Indirect Symbol

Jim Bostwick

Cargill Incorporated
Grain Research Laboratory
3444 Dight Ave. S.
Minneapolis, MN 55406
612/721-8531

## TO EDITOR

This is the hack we talked about on the phone. However, I found that you don't even have to use the "nquote%V" at all. Just .GOSUB DOQUO ", with the " as a literal, and ICP accepts it. Wierd. I have included command files QUOTEJB.CMD and TQUOTEJB.CMD in case you find it easier to format things that way.

This note is my solution to Roger Jenkins' artile in the May '84 Multi-Tasker. Roger's challenge is to come up with a 'clean' way to produce an indirect string symbol containing only the double-quote (") character. While my solution is a bit off the wall, it has the virtue of lending itself to general use.

My first reaction to Roger's article was "I know how to do that! Allen Watson showed me how in "Nifty things to do with Indirect"". So, I cranked up interactive ICP, and typed

```
.setn nquote 42                ;  42 is octal quote

.sets quote 'nquote%v'
```

Alas, the double-quote is apparently the only character that this trick won't work with. Indirect performs the substitution OK, and then processes (internally)

```
.sets quote """
```

which we know won't work.

What is needed is a way to have Indirect define a string without having to enclose it in quotes. Well, one of the quirks of Indirect (that has tripped me up all too often) is that when passing parameters to a subroutine, you DON'T quote the string. Sooo, I tried passing 'nquote%v' as a GOSUB parameter, and looking at COMMAN when I got there. Lo and behold, COMMAN contains a double-quote. All that remains is to assign COMMAN to some convenient string, and return. Works fine. However, as I was writing this up, it occured to me that if Indirect will substitute (using the nquote trick) a double-quote into the GOSUB command, why not just feed the GOSUB a double-quote? Looks wierd, but it too works.

Since our site makes extensive use of system-wide command files, I packaged the thing up as a separate command file (QUOTEJB.CMD), which stuffs the quote character into <EXSTRI>. Now, anyone wanting the quote string calls QUOTEJB, and saves the returned <EXSTRI> in some convenient place.

Files QUOTEJB.CMD and a simple demo TQUOTEJB.CMD are reproduced below. It would be simple to include the guts of QUOTEJB in any command file that needed it, but, with the system-wide facility, why bother?

```
.; QUOTEJB.CMD -- return double-quote character (") in <EXSTRI>
.; Jim Bostwick 24-May-84
.;
.; This command file produces a string containing
.; the double-quote character, and returns that
.; string in <EXSTRI>.
.; Anyone wishing to use such a string should execute
.;    @QUOTEJB
.;    .SETS quote <EXSTRI>
.; from within their command file.
.;

  .enable substitution
  .gosub doquo "
  .sets <EXSTRI> quote    ; quote set up by doquo
  .exit

.doquo:
  .sets quote COMMAN
  .return

.; TQUOTEJB.CMD -- a simple demonstration of QUOTEJB.
.; Jim Bostwick 24-May-84
.;
.; TQUOTEJB simply calls QUOTEJB, and prints out the
.; value received in <EXSTRI>, which is (we hope)
.; a double-quote.
.;
  .enable substitution
  @QUOTEJB
  .sets quote <EXSTRI>
  .test quote
; string "quote" contains  'quote'
;   and it''s length is '<STRLEN>'.
  .exit
```

## Another Reply to the Quote Question

A. Randall Barron
Gas Turbine Laboratory
Massachusetts Institute of Technology
Cambridge, MA 01239

RE: 'Placing a Quote in an Indirect Symbol',
RSX Multi-Tasker, May 1984, p. 27

The general problem of putting quotes (") into a string symbol can be solved using the parameter passing mechanism which is available even to internal subroutine calls. The following code fragment defines an Indirect subroutine, SETSTR, that can be used in the call:

```
.GOSUB SETSTR LINE John said, "My computer just broke."
```

to assign the indicated string value (quotes and all) to LINE. if substitution is enabled, any primes (') appearing in the string will be processed in the usual way befor the jump to SETSTR.

```
.SETSTR:
   .SETF TEMPL
   .IFENABLED SUBSTITUTION .SETT TEMPL
   .IFF TEMPL .ENABLE SUBSTITUTION
   .PARSE COMMAN " " TEMPS TEMPS1
   .SETS 'TEMPS' TEMPS1
   .IFF TEMPL .DISABLE SUBSTITUTION
   .RETURN
```

Of course the block labeled SETSTR must be part of the command file that executes the GOSUB. A minor variation on this method can be used to construct an external Indirect procedure that returns a desired string value to its caller in the symbol <EXSTRI>, where it can be assigned to a local string variable (see below). The external procedure, unlike the internal subroutine, converts lower case letters to upper case. I don't know any way around that.

A. Randall Barron

```
.;+SETSTR -- Procedure to assign a string with embedded quotes
.;+ V1.0 -- Written by A. R. Barron, May 1984
.;
.; Call:
.;         @SETSTR John said, "My computer just broke."
.;         .SETS LINE <EXSTRI>
.;
.;  That leaves LINE containing the indicated string (quotes and
.;  all).  A direct string assignment by the called, e.g.,
.;  something like
.;
.; .SETS LINE "John said, ""My computer just broke."""
.;
.;  would not have the desired effect.  If substitution is enabled
.;  in the caller, any primes (') appearing in the string argument
.;  will be processed in the usual way before the call to SETSTR.

   .PARSE COMMAN " " TEMPS TEMPS1
   .SETS <EXSTRI> TEMPS1
   .EXIT
```

# Reply to Placing a Quote in an Indirect Symbol

Van Miller

Federal Land Bank
P.O. Box 141
Springfield, MA 01101

After reading Roger Jenkins' article in the May 1984 issue, I thought that maybe the substitution format control function might do what he wanted. After trying different ways that seemed to follow the syntax rules, but with no luck, I tried the following:

```
.SETN Q 35.                ! Set Q to the value of quote
.SETS QUOTE 'Q%V'" ! Set QUOTE to a "
```

I wasn't sure why this worked but it set QUOTE to a one character string containing a quote (").

Upon further checking, I found that I had set Q to the wrong value. A quote is 34, not 35. A pound sign (#) is a 35. It seems that the pound sign is an alternate form of quote for use in delimiting strings. A quick check through the sources for INDIRECT verified that the pound sign was a valid alternate string delimiter.

The solution to Roger's problem is now just one line:

```
.SETS QUOTE #"#
```

This solution may not work on older versions of M and M+. I tested it under M V4.0 and verified it with the M+ V2.1 sources. However, it does not work under M+ V1.0.

# Organization of Volunteers for the RSX SIG

by Jeff Hamilton

The RSX SIG is in the process of undertaking to organize a volunteer database. In order to facilitate this task, all SIG members are asked to fill out the questionnaire which follows on the preforated pages and send it to:

Jeff Hamilton
E-Systems
P O Box 1056
Greenville, Texas 75401

by September 21, 1984.

From the results we receive, we will 1) determine what tasks are most wanted by the SIG membership, and 2) organize the SIG's volunteers so that they can be placed in tasks according to their interests and abilities.

RSX SIG Volunteer Questionnaire

Name_____

Company_____

Address_____

City, State, Zip_____

Area Code and Phone Number_____

---

Operating System Used (please check):

_____Current version of RSX-11M

_____Current version of RSX-11M PLUS

_____Unsupported Version of RSX-11M (please indicate which version number)

_____Unsupported Version of RSX-11M Plus (please indicate which version number)

_____Micro RSX

_____P/OS

---

Removable Media Available (please check):

_____800 bpi tape drive

_____1600 bpi tape drive

_____DX's (floppy disks)

_____RSX RX50's

_____Professional 350 RX50's

_____Other (please indicate)_____

---

How often do you attend DECUS symposia:

_____Always

_____Once a year

_____Occasionally

_____Never

---

Are you interested in working on any of the ongoing projects in the SIG? If so, please check the appropriate project:

_____Menu (The menu is an annual process. A compilation is made of changes/features for the RSX system which have been requested by SIG members. The SIG membership votes on these items to determine which are the most important. These items are then submitted to the DEC RSX development team, which responds to them at the next DECUS symposium).

_____SIG Planning (This committee makes long-range plans for the SIG).

_____RSX <---> Other SIG representative
Please indicate other SIG_____
(These representatives act as liaisons between RSX and the other SIG. They keep RSX SIG informed of any developments in the other SIG which are of interest to RSX users).

_____Volunteer Database (This group will organize the database for which this questionnaire is being undertaken).

Working groups (These groups each work on a specific aspect of RSX. If you are interested in working in any of the groups, please check that group. The following groups currently exist):

_____SIG Tape Collection (This group creates, indexes and handles the distribution of the SIG tapes. A SIG tape is generated at each DECUS symposium. Users submit software to be included on the SIG tape. These submissions are organized on a single tape, which is then distributed to SIG members via a LUG tree).

_____DECUS Library (This group evaluates packages in the DECUS library).

_____Data Acquisition, Simulation and Process Control. (This group is interested in real-time applications using RSX).

_____RSX-11M Unsupported Versions (This group tries to make life easier for people who are running unsupported versions of RSX by activities such as making symposium presentations, working on patches, and keeping track of a network of unsupported version users).

_____Runoff (This group works on Runoff, a program which is used for document formatting. Features are added to Runoff, bugs are fixed and documentation is kept up-to-date).

_____System Performance and Accounting (This group works on issues relating to keeping track of and improving system performance of RSX).

_____SRD (This group works on the Sort Directory utility, providing new features, fixing bugs and keeping the documentation up to date).

The following two working groups are proposed and could be formed if sufficient interest existed in the SIG for them:

_____Cheap Networks

_____Computer Aided Instruction

---

If you attend symposia and would be interested in any of the activities which take place there, please check that activity:

_____Making technical presentations

_____Chairing sessions

_____Software clinic (This is a session where users can bring software problems. Knowledgeable RSX doctors are made available at this time to help with such problems).

_____Tape copy (This is usually a very late-night session at which the submitted software is combined onto one tape).

_____Helping to run the SIG campground (The campground is a room where RSXers meet informally between sessions).

_____Pre-symposium seminars (These seminars are offered the Sunday before DECUS starts and cover topics in greater depth than can be achieved during a symposium session).

---

If you are interested in working on any of the SIG's publications, please indicate below:

_____Multi-Tasker (This is the monthly SIG newsletter).

_____Session handouts (These handouts are sold during symposia to facilitate people's ability to take notes during sessions).

What activities of the SIG have been most beneficial to you?

_____

_____

Are there any new activites which you think the SIG should be doing?

_____

_____

_____

Would you be willing to coordinate this new activity?

_____Yes _____No

# NOTES

# NOTES

DECUS SUBSCRIPTION SERVICE
DIGITAL EQUIPMENT COMPUTER SOCIETY
249 NORTHBORO ROAD, (BPO2)
MARLBORO, MA 01752

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

**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: _____
Phone No.: _____

Mail to: **DECUS - Attn: Subscription Service**
**249 Northboro Road, (BPO2)**
**Marlboro, MA 01752 USA**

Affix mailing label