THE -DEVIAS- LETTER

Issue No. 27

JUNE 1985

I N T H I S I S S U E

DeVIAS Letter - Issue 27
July, 1985

The DeVIAS Letter needs contributions in order to continue as an effective medium for exchange of information regarding IAS. All contributions should be either Runoff source or camera ready copy, e.g. sharp black type on 8.5 x 11 inch paper with 1 inch margins.

Please send all contributions to:

John Roman
McDonnell Douglas Corporation - Dept N436
600 McDonnell Blvd.
Hazelwood, Missouri 63042

Letter from the Editor

I have just returned from the spring DECUS Symposium in New Orleans and the news is good. Many of the IAS users present had received Release 3.2. Although it is still early, we have two reports in this issue on experiences in bringing up 3.2. Perhaps the experiences of others will help you as you upgrade to 3.2. As you bring up 3.2, write up your adventures and send them in.

Update B was delivered to SDC on May 24. It should be reaching you about the time that this issue arrives. An article in this issues indicates the bug fixes and enhancements in Update B. Give Update B a try and enter the "Find the Hidden Feature" Contest. Details follow.

Please fill out the WHIMS (Wishes, Hopes, Ideas, Muses, and Suggestions) form and send it in. It is located at the back of this issue. This is the way we can indicate to Digital what directions we want IAS to take - both hardware support and software enhancements. Make copies and send in one for each WHIM. Your input can make a difference.

Multiprocessors is an idea which will not die. Please let DEC know if you are interested in such a processor. Again, if enough people indicate their support, perhaps DEC will listen. So send in those cards and letters.

This is the last issue of the separate newsletters. Starting in August all of the SIG newsletters will be combined into a monthly publication. You will receive all of the newsletters for one low price! Hopefully the Subscription Service will send out details. The success of this combined newsletter depends on you. The DeVI-AS Letter depends on articles, letters, and other communications from you, our community. Please send in your material. You will be glad you did.

I just realized that this letter contains several pleas for your help and your action. This SIG depends on your support, so please, get involved.

John Roman

UPDATE B FOR IAS V3.2

The following problems have been corrected on Update B.

Component	Problem
MTAACP	Unable to read or write ANSI standard magnetic tapes
TKB	Tasks built using the SYMPAT option may fail when run
EXECUTIVE	Random system crashes on 18-bit processors
HNDLIB	With shadow recording enabled certain QIOs were being passed to both disks when not necessary
DEV	Missing RESAPR option in task build command file caused the task to abort when displaying device information
IAS (STR)	Any errors encountered during timesharing startup caused a binary dump to the terminal
DSC	Unable to find bad block information on RA80 device
SYSLIB	Tasks opening multiple files or using multi-buffering could hang when trying to close the open files
SYSLIB	Various problems with multi-buffering

Due to space limitation the above corrections have not been included in the library SYSRES. If you encounter problems when accessing files from a task linked to SYSRES, replace the link to SYSRES with the updated modules found in SYSLIB.

PIP	Rebuilt to not use SYSRES resident library (due to above problems with SYSRES)

HNDLIB has been rebuilt to include necessary corrections. ALL tasks which reference HNDLIB must be rebuilt.

The following modifications to IAS system software has been included on Update B.

IAS now supports the RC25 as a valid device. The DU device handler is used to interface to the RC25 and it uses the same standard address and vector as other DU type devices. The 11/84 processor is also supported EXCEPT when using RH11 type controllers (DB, DR, and DS devices) as the system disk.

Component	Modification
MOU	Mount may be patched at task built time to rewind magtapes being mounted foreign
MOU	The /FOR , /DCF, and /ATCH qualifiers may be used in place of the /CHA=[...] qualifier
F11ACP	Accepts the /DCF and /ATCH qualifiers passed by the MOU task
DMO	Dismount may be patched at task build time to rewind foreign mounted magtapes on dismount
MTAACP	The ACP may be patched at task build time to not unload magtapes on dismount
HNDLIB	All entry points in the resident library have been vectored. Thus all tasks built to use the new resident library will not have to be rebuilt when a new version of the library is provided on future IAS updates
TS11	The TS11 handler now supports multiple TS11/TU80 devices on a single system
DSC	Supports RC25 device type

BRU	Supports RC25 device type
SGN1	Supports RC25 device type and 11/84 processor type
All handlers	Re-built to use the new HNDLIB resident library

News from the SIG Librarian
Mike Robitaille

At long last I've finally received my IAS V3.2 kit on 9 May. I received the 800BPI distribution, along with DATATREIVE and the Autopatch A. I'm still missing my copy of FORTRAN-77, but I wasn't too upset right off. After all, I figured that the five tapes I received would keep me busy. I plowed right in and ran into my first stumbling block. BOOT A TAPE? The last release (IAS V3.1) was sent to me on RK05's, and it took a few phone calls and stumbling around some hardware manuals. I finally figured out what to do (it depends on your boot ROMS, what kind of ROMS, the address they're at...) and in short order had the system tape up and creating a single-user IAS system on an RP06.

The IASSYS tape is an interesting way to go. You answer about five questions and ignore the machine for a few hours. I spent the time getting the 3.2 documentation in order. In fact, I finished the documentation updates before the tape finished.

The OBJECTS BRU tape does not work well. First off, the release notes are wrong in that the backup set is 'V32OBJECTS' instead of 'OBJECTS'. I strongly recommend that after the tape has been mounted that you geet a directory of the backup sets on ANY tape you get from DEC in BRU format.

```
BRU /REW/DIRECTORY MM0:
```

works just fine.

After I had generated an IAS system from the system tape, and done initial work on the TT handler and the SYSGEN files and sysgened a multi-user system on an RP06 using DB0: as SY:, I executed the following:

```
MOU MM0:/CHA=[FOR,ATCH]
MOU DB3:/CHA=[FOR,ATCH]
```

```
BRU /INI/DIS/BACK:V32OBJECTS MM0: DB3:
```

After the BRU seemingly went through the entire tape and properly displayed all files found on the tape, the BRU went haywire and generated an awful lot of errors of the form:

```
BRU -- *WARNING* -- Data was lost due to IO errors
or
BRU -- *WARNING* -- Data was lost due to IO errors [001,001]filespec
File ID nnnnnn,nnnnnn LBN nnnnn.
```

There were too many of these to bother counting. DEMO indicated that there were no hardware hits. I would like to note that there is no sample command line in the documentation for how to BRU the OBJECTS tape. There is a real question whether to BRU the OBJECTS directly onto the new IASSYS pack just created, and if so, whether to use /SUPERSEDE or /NEWVERSION. Since I could not decide, I chose the prudent path and BRUed the tape as shown above to an empty pack and then tried to make up my mind at that point what to do about putting everything onto the IASSYS pack.

Incidently, MACIAS appears to have a problem if one compiles a source that is incomplete due to a BRU failure. I tried to compile the TT handler from the fully commented source that is on the OBJECTS tape, and I failed to spot the fact that the TSKDAT.MAC source was one of the files that failed to copy completely. MACIAS just plain hangs. It hung for FIVE HOURS until I aborted it. So much for the objects tape.

I had difficulty FLXing the SORT/MERGE tape, but after a couple of retries (reboots), I finally got past all the FLX lines in the installation procedure, and then the following happened:

```
;
PIP INSM11.TSK=INSM11.TSK/CO/NV
;
;       Run INSM11.TSK to get the sort/merge parameters
;       and to create SRTUTIL.CMD and MGEUTIL.CMD with
;       appropriate parameters.
;
RUN INSM11.TSK
;
;       This next command file will abort the
INS -- ILLEGAL POOL LIMIT FILE INSM11.TSK;3
;       job if something went wrong in SM11IN
;

AT. -- FILE NOT FOUND
@SMINRS.CMD
MCR>
```

I wonder if I received the wrong SORT/MERGE distribution. The tape and the documentation that went with it indicates that this is the SORT/MERGE for RSX and not for IAS. Specifically, the documentation indicated that this product was for RSX-11M V4.1 or RSX-11M PLUS V2.1. No indication for IAS at all.

DATATRIEVE did not install. The first time I ran the installation procedure, I said yes to the ' Do you want to build the distributed DTR-11 components?', and the installation procedure got very upset when it found out that I had no DECNET. The second time, I answered no, and it chugged along well until :

```
INSTALL LB:[11,1]DTR
DTR @DTR.TST
WRONG VERSION OF RMS (V1.8 OR LATER REQUIRED)
INITIALIZATION OF RM MODULE FAILED
```

I understood that RMS was pre-installed in IAS V3.2, no RMS installation manual exists for V3.2, and it seems that RMS appears to be in the system. Do not understand what is wrong.

I have not yet received F77 V5.0, but I did install F77 V4.1 with no apparent problems. I note that I did not insert the OTS modules into SYSLIB, but generated a F77FCSOTS.OLB and a F77RMSOTS.OLB in LB:[1,1] instead.

The Autopatch A tape BRUed like a charm. The notes that went with the tape said to set your UIC to the work UIC for the particular layered product that was to be BRUed. This resulted in privilege violations for every file. When the UIC was set to [1,1], there was no problem.

I found out that if you were to get the distribution tapes in 1600BPI format, you are getting the tapes later than the 800BPI people like me. Since the tapes I received were in such sorry shape, I convinced the local DEC people to get me copies of the 1600BPI tapes. When I told this to Norm Booth, he indicated that the 1600BPI tapes will have the Autopatch B stuff as well.

The DEC hardware people came in to check the tape drive, and they did find a problem. The serviceman said that the problem would manifest itself as select errors on the tape drive. That was exactly the problem during my attempts to FLX the SORT tape. Took a day to put the parts in and now all I can say is that the problem has changed. Now NO tapes work. Consistent parity errors in the drive, and ERRLOG catches them. Happens on any read at all.

Decided to do some setups on various and sundry things, build RUNOFF, SRD, etc. It seems that they build ok. Appears to be a bug in SRD in that an empty directory can contain the same files as the directory directly ahead of it in the listing.

Ran into a good one. Rebuilt SPR2 for just a leading banner and no trailing banner. If I install that copy of SPR2, it will hang in IR4 state just when it is not printing the trailing banner. Haven't DARED to try to remove both banners yet.

Speaking of banners, the BAT and BPR programs crank out the job listing as 'RSX-11D on IAS V3.1". I guess that one slipped through. Errors in batch jobs are still sent to the TI: and not in the spooled job report.

Got a few of the Library programs built as well. They are:

BROOM	(disk file residue cleaner)
COOKIE	(fortune cookie proverb generator)
DSO	(Directory SOrt)
GTC	(Get Terminal Characteristics)
INFORM	(Whatcha wanna know about what?)
PDL	("Poor Man's PDL")
SRD	(yet another one...)
TCR,TSE,TCS	(TCU-150 programmable clock programs)
RUNNL	(run programs on privileged TI of NL0:)
SCHED	(scheduler running under RUNNL)
GAMES	(control games via RUNNL to off-hours)

There are a whole pile of other stuff to get up and running, but it really would help to have a working tape drive. Well, should have more news for next issue. But now, back to the distribution.

The documentation set that comes with the V3.2 distribution set is really a change set for the V3.1 documentation. Those of us who have dog-eared V3.1 manuals apparently are going to have to live with them. Those of us who have the blue binders will have an amusing time trying to insert the wider orange binder spine covers. I have asked my DEC account people if there was a documentation upgrade for V3.1 that I missed that may have included the new orange binders. Plainly DEC shouldn't be sending me things that I can't use.

I feel that the instructions for generating the additional node pool were needlessly vague and in one instance just plain wrong. The example given in the SYSGEN and Startup Manual for a PAR directive gave an example for how to specify SENPAR as

```
PAR=SENPAR,*,4K,U
```

This gets SGN1 upset. If you change the line to

```
PAR=SENPAR,,4K,U
```

SGN1 accepts the directive. But I still do not know if the partition is being used. It does not show up on DEMO. I don't even know if DEMO shows partitions. The manual specifies that SENPAR must be the first partition defined, but does not indicate that it should be defined before or after the SCOM directive. There are no documentation changes for the portion of the manual containing console printouts of sample SYSGEN runs that indicate how to accomplish the node pool extension. It would have been fairly simple to provide a commented line in SYSGEN.CMD that had the SENPAR stuff. Even better, the prompt in QASGN.CMD asking you if you want the node pool extension could easily SLP this into SYSGEN.CMD. A follow-on question could ask you how big to make the SENPAR partition.

Also, there is no place in the SYSGEN documentation where it is indicated that RESFCP is not built, and must be built prior to being used in a SYSGEN. SYSGEN phase 1 really trashes things up when one specifies an ACP that does not yet exist. That took a while to recover from.

There was a foulup on the IAS taskbuilder manual. There is a missing page number (4-19) that would have contained the second of three pages for table 4-3 ("MCR TKB command - switches"). The switches that are not mentioned because of this missing page are those after /LI and before /WI.

If you would like to share your trials and tribulations installing V3.2, or you've run across something you haven't seen described in DEVIAS, please let me know either by calling me or writing:

Mike Robitaille
Grumman-CTEC, Inc.
6862 Elm St
McLean, VA 22101
(703) 556-7400

Also, if you've got a program that you feel would be useful to us all, let me know.

Klaus Centmayer
Techn. University Muenchen
Inst. f. Datenverarbeitung
Franz-Josef-Str. 38
D 8000 Muenchen 40

Robert E. Curley
Department of Radiation Therapy
Hospital of the University of Pensylvania
3400 Spruce Street
Philadelphia, Pensylvania 19104
USA

16-APR-85

Dear Robert,

Enclosed yoú find a letter concerning my first impressions about the new IAS V3.2, which I sent to the european IAS-users. It may bee of your interrest as well.
I am the IAS coordinator within the european RSX-SIG, because we have no separate IAS-SIG.
I am DeVIAS Member as well, and I hope I am still on your mailing list.

Sincerely,

[illegible]

Copy to: John Roman , DeVIAS Letter Editor

Klaus Centmayer
Techn. University Muenchen
Inst. f. Datenverarbeitung
Franz-Josef-Str. 38
D 8000 Muenchen 40

Munic, 15-Apr-85

Dear IAS Users !

As you know there is a new IAS update Version 3.2, which will arrive in this days, as far as you have a valid support.
I got the new version on a direct way, some weeks earlier.
This are the first impressions and errors from a short time usage, which may help you starting with the new version.

Generation is equivalent to earlier versions, no problem.
Because of a new SYSRES library (at least the date) all programs, using SYSRES must be linked new. A temporary solution is to install the old SYSRES as ALTRES and patch all tasks, where relinking is a problem, via ZAP: task.tsk/AB

```
Adr. 32 is 75273  (RAD50 SYS) new: 4064  (RAD50 ALT)
     34    70533  (RAD50 RES)
```

or adr. 52/54 if other resident libraries are linked.
This works without problems (at least with non priviledged progr.).
The system and TT-handler is somewhat larger, that arises some problems on my 11/45.

Changements:

INI Volume must be mounted FOReign, as in RSX11M+.

BRU For a disk to MT backup the disk now must be mounted FOReign, (not DCF), there is a second BRUFR, which works on normal mounted volumes, using the new TKB switch PR:0. There is a optional patch for BRU in the .cmd file to override file protection.

DEM has some new displays, but starts with a default memory size display and not the actual size.

Errors:

PIP does not work with MT, tape writing is not possible: Bad Parameters reading a old tape produces only a nonsense listing. (SPR).
New features: wild characters (slow), and /TD switch and perhaps other things are not documented.
Solution: use for MT transfer the old PIP.

EDT is new version V3 as in RSX11M, but is larger and slower than V2, and does not work correct in screen mode, there are sporadic CR LF characters included. (SPR). There is no documentation about the new features.
Solution: use the old EDT , i.e. no 8bit support, error message at exit.

SPR2 Output spooler, when patched according the cmd file without the banner pages hangs in IR4 state. (SPR).
Solution: ~~use with banner pages~~. TKB-error, see SW Disp. Apr 85 5.3.1.1

PRE MT-MT copy with /VE does not rewind the input tape, this is the same error as it was in V3.1, but the published patch (5.1.17. , Apr.82) produces with ZAP a checksum error. (SPR).
No solution.

TT-handler has been changed, there should be a character AST, I could not check this up to now.
But there' is still the same error, the ctrl S , ctrl Q does not work allways correct (see my still open SPR , JUN-84).
(new SPR).

SWAP seems to be slow, may be some changed tuning is necessary.

Kermit which was told would run under V3.2, arises still some problems at compilation, there are some macros, one system directive missing.
Kermit was my main reason to use the V3.2, but I think it needs still some work to get up Kermit.

Documentation:
There are a few new manuals, and some updates in the kit. Some documentation is missing.

Caution: In my update kit for 800BPI MT was one tape (objects) written in 1600 BPI.

This are my first experiences with the new version.
I would be glad to hear better news from you,
Enclosed you find my list of the european IAS-users, please let me know, if somebody is missing.

Klaus Centmayer

A N N O U N C I N G

FIND THE HIDDEN FEATURE CONTEST

The IAS Development Team has included certain undocumented features in Update B. These are currently for internal Digital use and may be supported in future updates. Three features have been identified by the IAS Development Team. This is your chance to show what you know about the system. Find the hidden features and win a prize.

Rules

1. To redeem your prize you must indicate:

    o what the feature is,
    o what it does,
    o where it is,
    o and how to use it.

2. Judges are the Digital IAS Developers and their decision is final.
3. You must describe one of the three known features.
4. No Digital employees or their families may enter.
5. The entries must be received by the DeVIAS Letter Editor by December 1, 1985.
6. Winners will be announced and awards presented at the Fall, 1985 DECUS Symposium at Anaheim.
7. The awards have not been determined at press time. However, they will undoubtedly be truly outstanding.
8. Void where prohibited, regulated, or taxed.

So hurry now! Send all entries to:

John Roman
McDonnell Douglas Corporation
Department N436
600 McDonnell Blvd.
Hazelwood, Missouri 63042

Borger's browsings

WHIMs list entry H.1

MT: modify the TU10 handlers so that it does not automatically allocate 8 UMR's upon loading and never release them.

When we updated to our 11/44 we ran into this problem in spades. We have an old TU10 (called MT:) and a newer Brand-X board with a streamer tape (called MM:) that also emulates a TU10. Had we used the standard DEC handler, we would have lost half our UMR's to those boys. Since we also had 3 separate DMA disk interfaces plus a DMA electrostatic printer plotter, we came up woefully short in the UMR department. Since it was forced upon us, I modified the TU10 handler to do dynamic UMR allocation.

Our version of the TU10 handler also corrects a problem with the TU10 hardware failing to recover correctly from a BUS GRANT late error. (See what happens when you put too many devices on a Unibus, bus grant late for something as slow as a TU10!!!) The TU10 would not backspace properly when retrying a write, (probably because the record was not properly terminated due to the error,) and the resultant tape would have an abnormally truncated record followed by the same record correctly written. Strangely enough, the Brand-X controller was buffered enough that it did not exhibit the problem.

The following SLP command file will create our modified handler from the standard DEC source in [311,14]. Note that there are a couple of things you can do:

1. If you have just a TU10 and want to include the bus grant late patch, the Slp file will work fine, unless your TU10 has the non-standard name of "MM"

2. If you want the bus grant late patch to only work on real TU10 hardware and not on emulators, install your other device with the name "MM" or edit the conditional in the source to match the name of a second tape controller.

3. If your TU10 works correctly, (which it should do unless you have our case with 3 disk controllers on the system,) and don't want the bus grant late patch, just skip the changes starting at line 2423.

In case anyone else has modified things, I also am including a standard CMP file comparing stock DEC and our modifications

Note that this handlers has been in service for 2.5 years with no apparent problems, but obviously has not been extensively pounded as it would have been in a field test.

```
-39
;          +++15.1 F BORGER         10-JUN-81         DO DOUBLE BACKSPACE ON RETRY
;                                                     AFTER BUS GRANT LATE
;          +++15.2 F BORGER         4-OCT-82          BUT ONLY FOR TU10, NOT FOR
;                                                     DILOG STREAMER
;          +++016  F.BORGER         DEC-82            DO DYNAMIC UMR ALLOCATION
-837,837
;          CLR     UMRSAV           ;ERS001 INITIALIZE SAVE AREA              +++016
-890,906
;          TST     UMRSAV           ;ERS001 ALREADY HAVE THEM                 +++016
;          BNE     10$              ;ERS001 YES ALL SET                       +++016
 MOV        1,R3          ;ERS001 YES GET 1 ONLY PRE-ALLOCATE 1     +++016
 MOV       R3,MRB+M.DF    ;SAVE IN MRB                              +++016
20$:       CALL    ..URAL           ;ERS001
 BCC       5$             ;ERS001 GOT THEM
;          DEC     R3               ;ERS001 NO TRY FOR ONE LESS THEN          +++016
;          BNE     20$              ;ERS001 GET ANY                           +++016
 BR        15$            ;ERS001 NO EXIT
;5$:       MOV     R3,UMRSAV        ;ERS001 SAVE SLOT FOR LATER               +++016
5$:        MOV     R3,MRB+M.PW      ;ERS001 SAVE SLOT FOR LATER               +++016
;          CALL    ..URAD           ;ERS001 GET 18 BIT ADDRESS                +++016
;          MOV     R4,UMRADD        ;ERS001 SAVE IT FOR TRANSFERS             +++016
;          MOV     R5,UMRADD+2                                                +++016
;          MOVB    UMRSAV+1,R2      ;ERS001 CALCULATE MAX TRANSFER LENGTH     +++016
;          ASH      13.,R2 ;ERS001 SHIFT FOR 8KBYTES PER UMR                  +++016
;          BCC     30$              ;ERS001 32K OVERFLOW                      +++016
;          MOV      177777,R2       ;ERS001 YES 32K-1                         +++016
;30$:      MOV     R2,UMRSZ         ;ERS001 SAVE THAT TOO                     +++016
-987,989
;UMRSAV:   .WORD   0                ;ERS001 UMR SLOT NUMBER         +++016
;UMRADD:   .BLKW   2                ;ERS001 UMR 18 BIT ADDRESS      +++016
;UMRSZ:    .BLKW   1                ;ERS001 MAX TRANSFER LENGTH               +++016
MRB:       .WORD   0,0,0,0,0,0      ;+++016 UMR MAP REGISTER BLOCK
-1257,1259
;EXIT1:    MOV     UMRSAV,R3        ;ERS001 SET UP TO DEALLOCATE UMRS         +++016
;          BEQ     1$               ;ERS001 NONE THERE                        +++016
;          CALL    ..URDA           ;ERS001 DO IT                             +++016
EXIT1:     MOV     MRB+M.PW,R3      ;GET PREALLOCATED UMRS                    +++016
 BEQ       1$             ;IF EQ WE HAVE NONE                       +++016
 CALL      @ ..URDA       ;DEALLOCATE THE PREALLOCATED UMR'S        +++016
-1351
 MOV        MRB,R2        ;POINT TO MAP REGISTER BLOCK              +++016
 CALL      @ ..DEMR       ;DE-ALLOCATE ANY UMRS DYNAMICALLY USED    +++016
-1668,1672
;          TST     UMRSAV           ;ERS001 NEED TO CHECK SIZE                +++016
;          BEQ     10$              ;ERS001 NO                                +++016
;          CMP     UMRSZ,R.PB+2(R1);ERS001 YES                                +++016
;          BLO     BADADR           ;ERS001 TOO BIG                           +++016
 PUSH      <R2,R5,R3>     ;SAVE R5,R2
-1678,1690
;1$:       TST     UMRSAV           ;ERS001 NEED UMRS                         +++016
```

```
1$:      BITB     ON.UM,,UMR22+1 ;NEED UMR'S ?                              +++016
 BEQ     5$                ;ERS001 NO
 CALL    ..VXUR            ;ERS001 VERIFY UMR TRANSFER
 BCS     IOCOM2            ;ERS001 ERROR
 MOV     R4,R.PB(R1)       ;FILL PAR BUFFER FOR ALMR               +++016
 MOV     R5,R.PB+2(R1)     ; LOW 16 BITS OF PHYSICAL ADDRESS       +++016
 MOV     R3,R.PB+4(R1)     ; BYTE COUNT                            +++016
 MOV     R2,-(SP)          ;SAVE PUD POINTER                       +++016
 MOV      MRB,R2           ;GET MAP REGISTER BLOCK ADDRESS         +++016
 CALL    @ ..ALMR          ;ALLOCATE UMR'S FOR THIS TRANSFER       +++016
 MOV     M.UL(R2),R5       ;LOW 16 BITS OF ADDRESS => R5           +++016
 MOV     M.UH(R2),R4       ;HIGH 2 BITS OF ADDRESS => R4           +++016
 MOV     (SP)+,R2          ;RESTORE PUD POINTER                    +++016
 BR      IOCOM2            ;AND RE-JOIN SEQUENCE                   +++016
;        BIT      RF.IT,R.WA(R1) ;ERS001 INTERMEDIATE BUFFERED              +++016
;        BNE      IOCOM2          ;ERS001 YES DONT FILL UMRS                +++016
;        MOV      UMRSAV,R3       ;ERS001 SET UP FOR FILL                   +++016
;        ASH       -4,R4          ;ERS001 SHIFT R4 BITS DOWN                +++016
;        CALL     ..URF2          ;ERS001 FILL UMRS FOR TRANSFER            +++016
;        MOV      UMRADD,R4       ;ERS001 SET UP ADDRESS                    +++016
;        MOV      UMRADD+2,R5                                               +++016
;        CLC                      ;ERS001 CLEAR C BIT FOR RETURN            +++016
;        BR       IOCOM2          ;ERS001 GO OFF                            +++016
-2423,2423
;                                 START OF BUS GRANT LATE PATCH FOR WRITE ;15.1
 BIT      BGL,SMTS         ;WAS ERROR A BUS GRANT LATE?            ;15.1
 BEQ     77$               ;NO                                     ;15.1
 MOV     XMTC,-(SP)        ;GET COMMAND AGAIN                      ;15.1
 BIC      177761,(SP)      ;ISOLATE FUNCTION                       ;15.1
 CMP     (SP), 2           ;IS IT A READ?                          ;15.1
 BEQ     76$               ;DON'T DO DOUBLE BACKSPACE ON READ      ;15.1
 CMP     U.DN(R0), "MM     ;IS IT THE DILOG INTERFACE ?            ;15.2
 BEQ     76$               ;IF SO, DON'T DO DOUBLE BACKSPACE       ;15.2
 MOV      -2,2(R5)         ;BGL DOESN'T PUT PARITY CHAR'S SO DO    ;15.1
                           ;TWO BACKSPACES, THE HARDWARE DOES ONLY ;15.1
                           ;ONE EFFECTIVELY                        ;15.1
76$:     TST      (SP)+           ;RESTORE STACK POINTER                    ;15.1
77$:     INC      XRFLG           ;INDICATE RETRY IN PROGRESS
/
```

```
**************************************************
 1)  TU10.MAC;2
  40    .PAGE
***************
 2)  TU10MRH.MAC;22
  40    ;               +++15.1 F BORGER        10-JUN-81       DO DOUBLE BACKSPACE ON RETRY
  41    ;                                                       AFTER BUS GRANT LATE
  42    ;               +++15.2 F BORGER        4-OCT-82        BUT ONLY FOR TU10, NOT FOR
  43    ;                                                       DILOG STREAMER
  44    ;               +++016  F.BORGER        DEC-82          DO DYNAMIC UMR ALLOCATION
  45    .PAGE
**************************************************
 1)  TU10.MAC;2
 837    CLR     UMRSAV          ;ERS001 INITIALIZE SAVE AREA
 838    CALL    UMRALL          ;ERS001 GET UMRS IF NEEDED
***************
 2)  TU10MRH.MAC;22
 842    ;       CLR     UMRSAV          ;ERS001 INITIALIZE SAVE AREA            +++016
 843    CALL    UMRALL          ;ERS001 GET UMRS IF NEEDED
**************************************************
 1)  TU10.MAC;2
 890    TST     UMRSAV          ;ERS001 ALREADY HAVE THEM
 891    BNE     10$             ;ERS001 YES ALL SET
 892    MOV     10,R3           ;ERS001 YES GET 8
 893    20$:    CALL    ..URAL          ;ERS001
 894    BCC     5$              ;ERS001 GOT THEM
 895    DEC     R3              ;ERS001 NO TRY FOR ONE LESS THEN
 896    BNE     20$             ;ERS001 GET ANY
 897    BR      15$             ;ERS001 NO EXIT
 898    5$:     MOV     R3,UMRSAV       ;ERS001 SAVE SLOT FOR LATER
 899    CALL    ..URAD          ;ERS001 GET 18 BIT ADDRESS
 900    MOV     R4,UMRADD       ;ERS001 SAVE IT FOR TRANSFERS
 901    MOV     R5,UMRADD+2
 902    MOVB    UMRSAV+1,R2     ;ERS001 CALCULATE MAX TRANSFER LENGTH
 903    ASH     13.,R2          ;ERS001 SHIFT FOR 8KBYTES PER UMR
 904    BCC     30$             ;ERS001 32K OVERFLOW
 905    MOV     177777,R2       ;ERS001 YES 32K-1
 906    30$:    MOV     R2,UMRSZ        ;ERS001 SAVE THAT TOO
 907    10$:    CLC                     ;ERS001 CLEAR CARRY
***************
 2)  TU10MRH.MAC;22
 895    ;       TST     UMRSAV          ;ERS001 ALREADY HAVE THEM               +++016
 896    ;       BNE     10$             ;ERS001 YES ALL SET                     +++016
 897    MOV     1,R3            ;ERS001 YES GET 1 ONLY PRE-ALLOCATE 1   +++016
 898    MOV     R3,MRB+M.DF     ;SAVE IN MRB                            ;++016
 899    20$:    CALL    ..URAL          ;ERS001
 900    BCC     5$              ;ERS001 GOT THEM
 901    ;       DEC     R3              ;ERS001 NO TRY FOR ONE LESS THEN        +++016
 902    ;       BNE     20$             ;ERS001 GET ANY                         +++016
 903    BR      15$             ;ERS001 NO EXIT
 904    ;5$:    MOV     R3,UMRSAV       ;ERS001 SAVE SLOT FOR LATER             +++016
```

```
905   5$:       MOV     R3,MRB+M,PW       ;ERS001 SAVE SLOT FOR LATER              +++016
906   ;         CALL    ..URAD            ;ERS001 GET 18 BIT ADDRESS               +++016
907   ;         MOV     R4,UMRADD         ;ERS001 SAVE IT FOR TRANSFERS            +++016
908   ;         MOV     R5,UMRADD+2                                                +++016
909   ;         MOVB    UMRSAV+1,R2       ;ERS001 CALCULATE MAX TRANSFER LENGTH    +++016
910   ;         ASH      13.,R2 ;ERS001 SHIFT FOR 8KBYTES PER UMR                  +++016
911   ;         BCC     30$               ;ERS001 32K OVERFLOW                     +++016
912   ;         MOV      177777,R2        ;ERS001 YES 32K-1                        +++016
913   ;30$:     MOV     R2,UMRSZ          ;ERS001 SAVE THAT TOO                    +++016
914   10$:      CLC                       ;ERS001 CLEAR CARRY
**************************************************
 1)  TU10.MAC;2
987   UMRSAV:   .WORD   0                 ;ERS001 UMR SLOT NUMBER
988   UMRADD:   .BLKW   2                 ;ERS001 UMR 18 BIT ADDRESS
989   UMRSZ:    .BLKW   1                 ;ERS001 MAX TRANSFER LENGTH
990   ;
***************
 2)  TU10MRH.MAC;22
994   ;UMRSAV:  .WORD   0                 ;ERS001 UMR SLOT NUMBER        +++016
995   ;UMRADD:  .BLKW   2                 ;ERS001 UMR 18 BIT ADDRESS     +++016
996   ;UMRSZ:   .BLKW   1                 ;ERS001 MAX TRANSFER LENGTH              +++016
997   MRB:      .WORD   0,0,0,0,0,0,0     ;+++016 UMR MAP REGISTER BLOCK
998   ;
**************************************************
 1)  TU10.MAC;2
1257  EXIT1:    MOV     UMRSAV,R3         ;ERS001 SET UP TO DEALLOCATE UMRS
1258   BEQ      1$              ;ERS001 NONE THERE
1259   CALL     ..URDA          ;ERS001 DO IT
1260  1$:       MOV      UIT,R0           ;GET UIT ADDRESS
***************
 2)  TU10MRH.MAC;22
1265  ;EXIT1:   MOV     UMRSAV,R3         ;ERS001 SET UP TO DEALLOCATE UMRS        +++016
1266  ;         BEQ     1$                ;ERS001 NONE THERE                       +++016
1267  ;         CALL    ..URDA            ;ERS001 DO IT                            +++016
1268  EXIT1:    MOV     MRB+M,PW,R3       ;GET PREALLOCATED UMRS                   +++016
1269   BEQ      1$              ;IF EQ WE HAVE NONE                      +++016
1270   CALL     @ ..URDA        ;DEALLOCATE THE PREALLOCATED UMR'S       +++016
1271  1$:       MOV      UIT,R0           ;GET UIT ADDRESS
**************************************************
 1)  TU10.MAC;2
1352   CLR      R2              ;ADJUSTMENT TO UNITY IS ZERO
***************
 2)  TU10MRH.MAC;22
1363   MOV       MRB,R2         ;POINT TO MAP REGISTER BLOCK             +++016
1364   CALL     @ ..DEMR        ;DE-ALLOCATE ANY UMRS DYNAMICALLY USED   +++016
1365   CLR      R2              ;ADJUSTMENT TO UNITY IS ZERO
**************************************************
 1)  TU10.MAC;2
1668   TST      UMRSAV          ;ERS001 NEED TO CHECK SIZE
1669   BEQ      10$             ;ERS001 NO
1670   CMP      UMRSZ,R,PB+2(R1);ERS001 YES
```

```
 1671    BLO      BADADR           ;ERS001 TOO BIG
 1672   10$:      PUSH    <R2,R5,R3>       ;SAVE R5,R2
 1673    MOV      R.PB(R1),R2      ;GET VIRTUAL ADDRESS
***************
 2)  TU10MRH.MAC;22
 1681    ;        TST      UMRSAV            ;ERS001 NEED TO CHECK SIZE            +++016
 1682    ;        BEQ      10$               ;ERS001 NO                            +++016
 1683    ;        CMP      UMRSZ,R.PB+2(R1);ERS001 YES                             +++016
 1684    ;        BLO      BADADR            ;ERS001 TOO BIG                       +++016
 1685    PUSH     <R2,R5,R3>       ;SAVE R5,R2
 1686    MOV      R.PB(R1),R2      ;GET VIRTUAL ADDRESS
****************************************************
 1)  TU10.MAC;2
 1678   1$:       TST      UMRSAV            ;ERS001 NEED UMRS
 1679    BEQ      5$               ;ERS001 NO
 1680    CALL     ..VXUR           ;ERS001 VERIFY UMR TRANSFER
 1681    BCS      IOCOM2           ;ERS001 ERROR
 1682    BIT       RF.IT,R.WA(R1) ;ERS001 INTERMEDIATE BUFFERED
 1683    BNE      IOCOM2           ;ERS001 YES DONT FILL UMRS
 1684    MOV      UMRSAV,R3        ;ERS001 SET UP FOR FILL
 1685    ASH       -4,R4           ;ERS001 SHIFT R4 BITS DOWN
 1686    CALL     ..URF2           ;ERS001 FILL UMRS FOR TRANSFER
 1687    MOV      UMRADD,R4        ;ERS001 SET UP ADDRESS
 1688    MOV      UMRADD+2,R5
 1689    CLC                       ;ERS001 CLEAR C BIT FOR RETURN
 1690    BR       IOCOM2           ;ERS001 GO OFF
 1691   5$:       CALL     ..VXFR            ;VALIDATE TRANSFER
***************
 2)  TU10MRH.MAC;22
 1691   ;1$:      TST      UMRSAV            ;ERS001 NEED UMRS                     +++016
 1692   1$:       BITB      ON.UM,.UMR22+1 ;NEED UMR'S ?                           +++016
 1693    BEQ      5$               ;ERS001 NO
 1694    CALL     ..VXUR           ;ERS001 VERIFY UMR TRANSFER
 1695    BCS      IOCOM2           ;ERS001 ERROR
 1696    MOV      R4,R.PB(R1)      ;FILL PAR BUFFER FOR ALMR             +++016
 1697    MOV      R5,R.PB+2(R1)    ; LOW 16 BITS OF PHYSICAL ADDRESS     +++016
 1698    MOV      R3,R.PB+4(R1)    ; BYTE COUNT                          +++016
 1699    MOV      R2,-(SP)         ;SAVE PUD POINTER                     +++016
 1700    MOV       MRB,R2          ;GET MAP REGISTER BLOCK ADDRESS       +++016
 1701    CALL     @ ..ALMR         ;ALLOCATE UMR'S FOR THIS TRANSFER     +++016
 1702    MOV      M.UL(R2),R5      ;LOW 16 BITS OF ADDRESS => R5         +++016
 1703    MOV      M.UH(R2),R4      ;HIGH 2 BITS OF ADDRESS => R4         +++016
 1704    MOV      (SP)+,R2         ;RESTORE PUD POINTER                  +++016
 1705    BR       IOCOM2           ;AND RE-JOIN SEQUENCE                 +++016
 1706    ;        BIT       RF.IT,R.WA(R1) ;ERS001 INTERMEDIATE BUFFERED          +++016
 1707    ;        BNE      IOCOM2            ;ERS001 YES DONT FILL UMRS            +++016
 1708    ;        MOV      UMRSAV,R3         ;ERS001 SET UP FOR FILL               +++016
 1709    ;        ASH       -4,R4            ;ERS001 SHIFT R4 BITS DOWN            +++016
 1710    ;        CALL     ..URF2            ;ERS001 FILL UMRS FOR TRANSFER        +++016
 1711    ;        MOV      UMRADD,R4         ;ERS001 SET UP ADDRESS                +++016
 1712    ;        MOV      UMRADD+2,R5                                             +++016
 1713   ;        CLC                    ;ERS001 CLEAR C BIT FOR RETURN         +++016
 1714   ;        BR       IOCOM2        ;ERS001 GO OFF                         +++016
 1715   5$:      CALL     ..VXFR        ;VALIDATE TRANSFER
***************************************************
  1)  TU10.MAC;2
 2423    INC     XRFLG          ;INDICATE RETRY IN PROGRESS
 2424   9$:      CALL     MTYGO         ;LOAD UP COMMAND
**************
  2)  TU10MRH.MAC;22
 2447   ;                               START OF BUS GRANT LATE PATCH FOR WRITE ;15.1
 2448    BIT      BGL,SMTS      ;WAS ERROR A BUS GRANT LATE?           ;15.1
 2449    BEQ     77$            ;NO                                    ;15.1
 2450    MOV     XMTC,-(SP)     ;GET COMMAND AGAIN                     ;15.1
 2451    BIC      177761,(SP)   ;ISOLATE FUNCTION                      ;15.1
 2452    CMP     (SP), 2        ;IS IT A READ?                         ;15.1
 2453    BEQ     76$            ;DON'T DO DOUBLE BACKSPACE ON READ     ;15.1
 2454    CMP     U.DN(R0), "MM  ;IS IT THE DILOG INTERFACE ?           ;15.2
 2455    BEQ     76$            ;IF SO, DON'T DO DOUBLE BACKSPACE      ;15.2
 2456    MOV      -2,2(R5)      ;BGL DOESN'T PUT PARITY CHAR'S SO DO   ;15.1
 2457                           ;TWO BACKSPACES, THE HARDWARE DOES ONLY ;15.1
 2458                           ;ONE EFFECTIVELY                       ;15.1
 2459   76$;     TST      (SP)+         ;RESTORE STACK POINTER                 ;15.1
 2460   77$:     INC      XRFLG         ;INDICATE RETRY IN PROGRESS
 2461   9$:      CALL     MTYGO         ;LOAD UP COMMAND

    9 DIFFERENCES FOUND
TU10.DIF=TU10.MAC.TU10MRH.MAC
```

My Favorite Macro

Dave Brown
McDonnell Douglas Corp.
600 McDonnell Blvd.
Hazelwood, Missouri 63042

For those of you who may have some need to program the VT100 terminal, the following macros might prove useful. They were adopted from macros for VT52 control, supplied along with the program debugging tool BUG, available through the DECUS library.

These macros have proved to be sufficient for minor in-house utility programs, and can be improved. One enhancement would be to provide error checking. The assumption has been that if the QIO fails, there must be a hardware problem.

While not directly related to the VT100, several other macros are included, such as ringing the terminal bell. Also, the popular PUSH and POP macros have been included for reference.

Please refer to the VT100 programmers guide for more complete descriptions of the control sequences used in these macros, and the IAS Device Handlers Reference Manual for QIO parameters.

```
;
;       VT100 macros
;
ESC =   27.                             ; Assignment for ASCII escape

        .MCALL QIOW$S

        .MACRO  PUSH    LIST            ; Push list onto stack
        .IRP    R,<LIST>
        MOV     R,-(SP)
        .ENDR
        .ENDM   PUSH

        .MACRO  POP     LIST            ; Pop list off stack
        .IRP    R,<LIST>
        MOV     (SP)+,R
        .ENDR
        .ENDM   POP

        .MACRO  TAB     ROW,COL         ; Direct cursor addressing
        ROW2 =  48                      ; The following assignments
        COL2 =  48                      ; convert row and column
        RD2  =  ROW/10.                 ; numbers to ASCII
        .IIF    NE,RD2  ROW2 = RD2+48.
```

```
         ROW1 =   <ROW-<RD2*10.>>+48.
         CD2  =   COL/10.
         .IIF     NE,CD2  COL2 = CD2+48.
         COL1 =   <COL-<CD2*10.>>+48.
         MOV      #ROW2,-(SP)
         MOV      #ROW1,-(SP)
         MOV      #COL2,-(SP)
         MOV      #COL1,-(SP)
         CALL     $TAB
         .ENDM    TAB

         .MACRO   CLEAR                    ; Clear the screen
         CALL     $CLEAR
         .ENDM    CLEAR

         .MACRO   ERSEOS                   ; Erase to end of screen
         CALL     $EREOS
         .ENDM    ERSEOS

         .MACRO   ERSEOL                   ; Erase to end of line
         CALL     $EREOL
         .ENDM    ERSEOL

         .MACRO   ERSLIN                   ; Erase entire line
         CALL     $ERLIN
         .ENDM    ERSLIN

         .MACRO   TYPEAT  ROW,COL,TXT,?TAG,?TEXT
         JMP      TAG                      ; Type message with cursor
         ROW2 =   48                       ; positioning
         COL2 =   48
         RD2  =   ROW/10.
         .IIF     NE,RD2  ROW2 = RD2+48.
         ROW1 =   <ROW-<RD2*10.>>+48.
         CD2  =   COL/10.
         .IIF     NE,CD2  COL2 = CD2+48.
         COL1 =   <COL-<CD2*10.>>+48.
TEXT:
         .ASCII   <ESC>/[/
         .BYTE    ROW2,ROW1,59.,COL2,COL1,102.
         .ASCII   \TXT\
         .NCHR    LEN,<TXT>
         LEN =    LEN+8.
         .EVEN
TAG:
         PUSH     <#LEN,#TEXT>
         CALL     $TYPE
         .ENDM    TYPEAT

         .MACRO   INPUT   DATA,LEN         ; Receive input
         PUSH     <#DATA,#LEN>
```

```
        CALL    $INPUT
        .ENDM   INPUT

        .MACRO  TTY     ?MESAG,?TAG     ; Type the character in R1
        MOVB    R1,MESAG                ; on TI:
        QIOW$S  #IO.WVB,#5,#1,,,,<#MESAG,#1,#00>
        BR      TAG
MESAG:  .BYTE   0
        .EVEN
TAG:
        .ENDM   TTY

        .MACRO  CRLF                    ; Carriage return
        CALL    $CRLF                   ; and line feed
        .ENDM   CRLF

        .MACRO  TYPE    MSG,?MESAG,?TAG ; Type message
        .NCHR   LEN,<MSG>
        MOV     #LEN,-(SP)
        MOV     #MESAG,-(SP)
        CALL    $TYPE
        BR      TAG
MESAG:  .ASCII  \MSG\
        .EVEN
TAG:
        .ENDM   TYPE

        .MACRO  TTYI    MSGPTR,LEN      ; Type message pointed
        PUSH    <LEN,MSGPTR>            ; to by MSGPTR with
        CALL    $TYPE                   ; a length of LEN
        .ENDM   TTYI

        .MACRO  TYPECR MSG              ; Type a string with CR/LF
        TYPE    <MSG>
        CRLF
        .ENDM   TYPECR

        .MACRO  BELL                    ; Ring the terminal bell
        CALL    $BELL
        .ENDM   BELL

;
;       Supporting subroutines
;

$TAB:                                   ; Direct cursor addressing
        MOVB    8.(SP),TROW             ; QIO
        MOVB    6(SP),TROW+1
        MOVB    4(SP),TCOL
        MOVB    2(SP),TCOL+1
        QIOW$S  #IO.WAL,#5,#1,,,,<#TABX,#8.,#0>
        MOV     (SP)+,(SP)
        MOV     (SP)+,(SP)
        MOV     (SP)+,(SP)
        MOV     (SP)+,(SP)
        RETURN

$CLEAR:                                 ; Clear screen QIO
        QIOW$S  #IO.WAL,#5,#1,,,,<#CLSTG,#CLSTG$,#0>
        RETURN

$EREOS:                                 ; Erase to end of screen QIO
        QIOW$S  #IO.WAL,#5,#1,,,,<#ESSTG,#ESSTG$,#0>
        RETURN

$EREOL:                                 ; Erase to end of line QIO
        QIOW$S  #IO.WAL,#5,#1,,,,<#ERSTG,#ERSTG$,#0>
        RETURN

$ERLIN:                                 ; Erase entire line QIO
        QIOW$S  #IO.WAL,#5,#1,,,,<#LNSTG,#LNSTG$,#0>
        RETURN

$BELL:                                  ; Ring bell QIO
        QIOW$S  #IO.WAL,#5,#1,,,,<#BLSTG,#BLSTG$,#0>
        RETURN

$TYPE:                                  ; Type message
        PUSH    <R0,R1>                 ; with cursor positioning
        MOV     6(SP),R0
        MOV     8.(SP),R1
        QIOW$S  #IO.WLB!TF.WAL,#5,#1,,,,<R0,R1,#0>
        POP     <R1,R0>
        MOV     (SP)+,(SP)
        MOV     (SP)+,(SP)
        RETURN

$INPUT:                                 ; Input QIO
        PUSH    <R0,R1>                 ; Note that the user must
        MOV     6(SP),R0                ; supply the I/O status
        MOV     8.(SP),R1               ; buffer IOST.
        QIOW$S  #IO.RVB,#5,#MEF,,#IOST,,<R1,R0>
        POP     <R1,R0>                 ; IOST will contain the
        MOV     (SP)+,(SP)              ; number of characters
        MOV     (SP)+,(SP)              ; actually input.
        RETURN

$CRLF:                                  ; Carriage return / Line feed
        TTYI    #100$,#2
        RETURN
100$:   .BYTE   12,15
```

```
;
;       VT100 Control sequences
;

TABX:   .ASCII  <ESC>/[/                        ; Direct cursor addressing
TROW:   .BYTE   0,0                             ; Row
        .ASCII  /;/
TCOL:   .BYTE   0,0                             ; Column
        .BYTE   102.

CLSTG:  .ASCII  <ESC>/[H/                       ; Home and clear
        .ASCII  <ESC>/[2J/
CLSTG$=.-CLSTG

ERSTG:  .ASCII  <ESC>/[2K/                      ; Erase to end of line
ERSTG$=.-ERSTG

ESSTG:  .ASCII  <ESC>/[0J/                      ; Erase to end of screen
ESSTG$=.-ESSTG

BLSTG:  .BYTE   7.                              ; Control/G - Bell
BLSTG$=.-BLSTG

LNSTG:  .ASCII  <ESC>/[2K/                      ; Erase entire line
LNSTG$=.-LNSTG

        .EVEN

;
;       This sample program demonstrates a number of the
;       above macros, and includes one method for setting
;       special attributes.
;

        .MCALL  EXIT$S
        .RADIX  10

START:
        CLEAR                                   ; Clear screen
        TYPECR  <VT100 Sample Program>          ; Type message with CR/LF
        TAB     3,10                            ; Cursor at row 3, col 10
        TTYI    #OUTPUT,#OUTLEN                 ; Type message in OUTPUT
        BELL                                    ; Wake up user
        TAB     3,18                            ; Cursor at row 3, col 18
        INPUT   INPUT,1                         ; One character into INPUT
        ERSLIN                                  ; Erase current line
        TYPEAT  5,10,<Just testing!>            ; Type message at
        EXIT$S                                  ; row 5, col 10
```

```
INPUT:   .ASCII   / /
OUTPUT:  .ASCII   <ESC>/[7m/                 ; Set to reverse video
         .ASCII   /Prompt/
         .ASCII   <ESC>/[0m/                 ; Reset to no attributes
OUTLEN=  .-OUTPUT
         .EVEN
IOST:    .BLKW    2                          ; I/O Status

         .END     START
```

SUPERVISOR-MODE LIBRARIES

Mike Reilly

IAS Development Group

A supervisor-mode library is a resident library that doubles a user task's virtual address space by mapping the instruction space of the processor's supervisor mode. Supervisor-mode libraries are available only on systems running on PDP-11/44s, PDP-11/45s and PDP-11/70s.

INTRODUCTION

A call from within a user task to a subroutine within a supervisor-mode library causes the processor to switch from user to supervisor mode. The user task transfers control to a mode-switching vector that TKB includes within the task. The mode-switching vector performs the mode switch and then transfers control to the called subroutine within the supervisor-mode library. The library routine executes with the processor in supervisor mode. When the library routine finishes executing, it transfers control to a completion routine within the library. The completion routine mode switches the processor back to user mode. The user task continues executing with the processor in user mode at the return address on the stack. This process recurs whenever the user task calls a subroutine in the supervisor-mode library.

MODE-SWITCHING VECTORS

In a task that links to a supervisor-mode library, TKB includes a 4-word, mode-switching vector in the user task's address space for each entry point referenced of a subroutine in the library.

The following shows the contents of a mode-switching vector:

```
MOV #COMPLETION-ROUTINE,-(SP)
CSM #SUPERVISOR-MODE-ROUTINE ADDRESS
```

NOTE

When mode switching from user to supervisor mode, all registers of the referencing task are preserved. All condition codes in the PS saved on the stack are cleared and must be restored by the completion routine.

SUPERVISOR-MODE LIBRARIES

COMPLETION ROUTINES

After the subroutine finishes executing, its RETURN statement transfers control to a completion routine that mode-switches from supervisor to user mode. The completion routine returns program control back to the referencing task at the instruction after the call to the subroutine. There are two completion routines in SYSLIB:

- $CMPCS restores only the carry bit in the user-mode PS.
- $CMPAL restores all the condition code bits in the user-mode PS.

RESTRICTIONS ON THE CONTENTS OF SUPERVISOR-MODE LIBRARIES

The following restrictions are placed on the contents of a supervisor-mode library:

- Only subroutines using the form JSR PC, x should be used within the library.
- The library must not contain subroutines that use the stack to pass parameters.
- If both the library and the referencing task link to a subroutine from SYSLIB, then the entry point name of the subroutine must be excluded from the .STB file for the library.
- The library must not contain data of any kind (even R/O) because the user supervisor D-space APRs map the user task by default. This includes user data, buffers, I/O status blocks, and directive parameter blocks (only the $S directive form can be used, because the DPB for this form is pushed onto the user stack at run time).

SUPERVISOR-MODE LIBRARY MAPPING

Supervisor-mode libraries are mapped with the supervisor I-space APRs. Supervisor D-space APRs map the user task.

Supervisor D-space APRs are copies of user I-space APRs, which map the entire user task. This gives the library access to data within the user task.

SUPERVISOR-MODE LIBRARIES

BUILDING AND LINKING TO SUPERVISOR-MODE LIBRARIES

Building and linking to a supervisor-mode library is essentially the same as building and linking to a conventional resident library. When you build a supervisor-mode library using the TKB command line, you suppress the header by attaching /-HD to the task image file. If you use LINK, you use the /NOHEAD qualifier in the LINK command line. During option input, you suppress the stack area by specifying STACK=0. You optionally specify the base address of the library with the BASE option.

Relevant TKB Options

Use the following options to build and reference supervisor-mode libraries:

CMPRT	Indicates that you are building supervisor-mode library and specifies the name of the completion routine.
SUPLIB (RESSUP)	Indicates that your task references a supervisor-mode library.
GBLXCL	Excludes a global symbol from the .STB file of the supervisor-mode library.

These options are discussed briefly below.

Building The Library

You indicate to the TKB that you are building a supervisor-mode library with the CMPRT option. The argument for this option identifies the entry symbol of the completion routine. When the TKB processes this option, it places the completion routine entry point in the library's STB file. To exclude a global symbol from the library's .STB file, you specify the name of the global symbol as the argument of the GBLXCL option. You must exclude from the .STB file of a supervisor-mode library any symbol defined in the library that represents the following:

- o An entry point to a subroutine that uses the stack to pass parameters
- o An entry point to a subroutine mapped in user mode that the referencing user task calls

SUPERVISOR-MODE LIBRARIES

Building The Referencing Task

When you build a task that references a supervisor-mode library, use the RESSUP option if you are referencing a user-owned, supervisor-mode library and SUPLIB if you are referencing a system-owned, supervisor-mode library. (Like the RESLIB and LIBR options for linking to conventional libraries, RESSUP and SUPLIB are functionally the same.) The arguments for these options are:

- The filespec (RESSUP option) or name (SUPLIB) of the library to be referenced
- A switch that tells TKB whether to use system-supplied vectors to perform mode switching from user to supervisor mode.
- For position-independent libraries, the first available supervisor-mode I-space APR that you want to map the library.

Mode Switching Instruction

Mode switching occurs with a new instruction available on the 11/44 and emulated by the Executive on the 11/70. Throughout the remainder of this handout, supervisor-mode libraries are referred to as CSM (change supervisor mode) libraries.

CSM LIBRARIES

This section discusses how you build and link to CSM libraries. It also shows an extended example of building and linking to a CSM library and explains the context-switching vectors and completion routines for CSM libraries.

Building A CSM Library

You indicate to the Task Builder that you are building a CSM library by specifying the name of the completion routine as the argument for the CMPRT option. This option places the name of the completion routine into the library's .STB file. Link the completion routine, either $CMPAL or $CMPCS, located in LB:[1,2]SYSLIB.OLB, as the first input file. Although the completion routines are located in SYSLIB (which is ordinarily referenced by default), you must explicitly indicate it and link it as the first input file. You must also specify in the BASE option a 0 base for the library. These two steps locate the completion routine at virtual 0 of the library's virtual address space.

You specify the name of any global symbols that you would like to exclude from the library's .STB file as the argument to the GBLXCL option. You must exclude from the .STB file of a supervisor-mode library any symbol defined in the library that represents the following:

- o An entry point to a subroutine that uses the stack to pass parameters
- o An entry point to a subroutine mapped in user mode that the referencing user task calls

A sample TKB command sequence for building a CSM library in UFD [301,55] on SY: follows:

```
TKB>CSM/-HD/LI,CSM/MA,CSM=
TKB>LB:[[1,2]]SYSLIB/LB:CMPAL,SY:[301,55]CSM
TKB>/
Enter Options:
TKB>STACK=0
TKB>UIC=[1,1]
TKB>BASE=0
TKB>CMPRT=$CMPCS
TKB>GBLXCL=$SAVAL
TKB>//
```

Or, you can use the following LINK command sequence to build the same library:

```
LINK/TAS:CSM/NOH/LIB/MAP:CSM/SYM:CSM/OPT -
LB:[1,2]SYSLIB/INCLUDE:CMPAL,SY:[301,55]CSM
Option? STACK=0
Option? UIC=[1,1]
Option? BASE=0
Option? CMPRT=CMPCS
Option? GBLXCL=$SAVAL
Option? <RET>
```

The library is built without a header or stack, like all shared regions. It is built with a starting virtual address of 0. The /LI switch in TKB or the /LIB qualifier in LINK switch eliminate program section name conflicts between the library and the referencing task.

The completion routine module of SYSLIB, CMPAL, is specified first in the input line. These are two aspects of building supervisor-mode libraries specific to CSM libraries: the completion routine must be linked first, and must reside at virtual 0. Why the CSM library must reside at virtual 0 is discussed in in a later section of this handout.

The CMPRT option specifies the global symbol $CMPCS, which is the entry point of the completion routine. Note that the SYSLIB module name is "CMPCS" and its corresponding global symbol is "$CMPCS".

The GBLXCL option excludes $SAVAL from the library's .STB file because the user task must reference a copy of $SAVAL that is mapped with user mode APRs.

Linking To A CSM Library

If your task links to a user-owned CSM library, you use the RESSUP option. If your task links to a system-owned CSM library, you use the SUPLIB option. These options tell TKB that the task will link to a supervisor-mode library. The option takes up to three arguments:

- The filespec (RESSUP option) or name (SUPLIB option) of the library
- A switch that tells the TKB whether to use system-supplied, mode-switching vectors
- For position-independent libraries, an APR that must be APR 0 so that the library's completion routine is mapped at virtual 0.

This information enables the TKB to find the .STB file for the CSM library, include a 4-word, mode-switching vector within the user task for each call to a subroutine within the library, and correctly map the library at virtual 0 in the library image.

The following examples of TKB and LINK command sequences build a task named REF, which references the library SUPER that you built in the previous section:

```
TKB>REF,REF=REF
TKB>/
Enter Options:
TKB>RESSUP=SUPER/SV:0
TKB>//

LINK/TAS/MAP/OPT REF
Option? RESSUP=SUPER/SV:0
Option? <RET>
```

This sequence tells TKB to include in the logical address space of REF a user-owned, supervisor-mode library named SUPER. TKB includes a 4-word mode switching vector within the user task for each call to a subroutine within the library. The CSM library is position independent and is mapped with APR 0.

Example CSM Library And Linking Task

This example shows you the code and maps and the TKB and LINK command sequences for building and linking to a CSM library that contains no data in a system not having user data space. Example 1 shows the code for the library SUPER, and Example 2 shows its accompanying map. Example 3 shows the code for referencing task TSUP, and Example 4 shows its accompanying map.

Example 1 Code for SUPER.MAC

```
        .TITLE  SUPER
        .IDENT  /01/

SORT::
        CALL    $SAVAL          ; SAVE ALL REGISTERS
        TST     (R5)+           ; SKIP OVER NUMBER OF ARGUMENTS
        MOV     (R5)+,R0        ; GET ADDRESS OF LIST
        MOV     (R5)+,R4        ; GET ADDRESS OF LENGTH OF LIST
        MOV     (R4),R4         ; GET LENGTH OF LIST
        BEQ     40$             ; IF NO ARGUMENTS
        MOV     R0,R5           ;
        DEC     R4              ;
10$:
        MOV     R5,R0           ; COPY
        MOV     R4,R3           ; COPY LENGTH OF LIST
20$:
        TST     (R0)+           ; MOVE POINTER TO NEXT ITEM
        CMP     (R5),(R0)       ; COMPARE ITEMS
        BLE     30$             ; IF LE IN CORRECT ORDER
        MOV     (R5),R2         ; SWAP ITEMS
        MOV     (R0),(R5)       ;
        MOV     R2,(R0)         ;
30$:
        DEC     R3              ; DECREMENT LOOP COUNT
        BGE     20$             ; IF NE LOOP
        DEC     R4              ; DECREMENT
        BLE     40$             ; IF EQ SORT COMPLETED
        TST     (R5)+           ; GET POINTER TO NEXT ITEM TO BE COMPARED
        BR      10$
40$:
        RETURN

SEARCH::
        CALL    $SAVAL          ; SAVE ALL THE REGISTERS
        CMP     #4,(R5)+        ; FOUR ARGUMENTS?
        BNE     20$             ; IF NE NO
        MOV     (R5)+,R0        ; GET ADDRESS OF NUMBER TO LOCATE
        MOV     (R5)+,R1        ; ADDRESS OF LIST SEARCHING
        MOV     (R5)+,R2        ; GET ADDRESS OF LENGTH OF LIST
        MOV     (R2),R2         ; GET LENGTH OF LIST
        BEQ     20$             ; IF NO ARGUMENTS
        MOV     (R5),R5         ; ADDRESS OF RETURNED VALUE
        MOV     R2,R3           ; COPY LENGTH
```

```
10$:
          CMP     (R0),(R1)+        ; IS THIS THE NUMBER?
          BEQ     30$               ; IF EQ YES
          BMI     20$               ; IF MI NUMBER NOT THERE
          DEC     R2                ; DECREMENT LOOP COUNT
          BNE     10$               ; IF NE NOT AT END OF LIST
20$:
          MOV     #-1,(R5)          ; END OF LIST PASS BACK ERROR
          RETURN
30$:

          SUB     R2,R3             ; NUMBER FOUND - GET INDEX INTO LIST
          INC     R3                ;
          MOV     R3,(R5)           ; RETURN INDEX
          RETURN
          .END
```

Example 2 Memory Allocation Map for SUPER

```
SUPER.TSK;1   Memory allocation map  TKB D40.0       Page 1
                     8-DEC-84    00:29


Identification : 0204
Task  UIC      : [1,1]
Task attributes: PI,-HD
Total attachment descriptors: 0.
Task  image  size  : 128. WORDS
Task address limits: 000000 000343
R-W disk blk limits: 000002 000001 000000 00000.
R-O disk blk limits: 000002 000001 000000 00000.

*** Root segment: CMPAL


R/W mem  limits: 000000 000343 000344 00228.
Disk blk limits: 000002 000002 000001 00001.


Memory allocation synopsis:

Section                                                    Title  Ident  File
-------                                                    -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)      000000 000342 00226.
                               000000 000140 00096. CMPAL  0204   SYSLIB.OLB;14
                               000140 000140 00096. SUPER  01     SUPER.OBJ;1
                               000300 000042 00034. SAVAL  00     SYSLIB.OLB;14


Global symbols:

SEARCH 000220-R  SORT   000140-R  $CMPAL 000022-R  $CMPCS 000110-R  $SAVAL 000300-R  $SRTI   000002-R


*** Task builder statistics:

    Total work file references: 284.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 15014. words (58. pages)
    Size of work file: 768. words (3. pages)

    Elapsed time:00:00:07
```

Example 3 Code for TSUP.MAC

```
        .TITLE  TSUP
        .IDENT  /01/

        .MCALL  QIOW$,DIR$,QIOW$S
WRITE:  QIOW$   IO.WVB,5,1,,,,<OUT,,40>
READIN: QIOW$   IO.RVB,5,1,,,,<OUT,5>

IARRAY: .BLKW   12.
LEN:    .BLKW   1
IART:   .BLKW   1
INDEX:  .WORD   0
OUT:    .BLKW   100.
ARGBLK:
EDBUF:  .BLKW   10.

FMT1:   .ASCIZ  /%2SARRAY(%D)=/
FMT2:   .ASCIZ  /%N%2SNUMBER TO SEARCH FOR?/
FMT3:   .ASCIZ  /%N%2S%D WAS FOUND IN ARRAY(%D)/
FMT4:   .ASCIZ  /%N%2S%D WAS NOT IN ARRAY/
FMT5:   .ASCIZ  /%2SARRAY(%D)=%D/

        .EVEN
START:
        MOV     #IARRAY,R0          ; GET ADDRESS OF ARRAY
        MOV     #10,R1              ; SET LENGTH OF ARRAY
5$:
        CLR     (R0)+               ; INITIALIZE ARRAY
        DEC     R1                  ; LOOP
        BNE     5$
        MOV     #IARRAY,R0          ;
        MOV     #INDEX,R2
10$:
        MOV     #FMT1,R1            ; FORMAT SPECIFICATION (ADDRESS
                                    ; OF INPUT STRING)
        MOV     (R2),EDBUF          ; GET INDEX
        INC     EDBUF               ;
        CALL    PRINT               ; PRINT MESSAGE
        CALL    READ                ; READ INPUT
        MOV     IART,(R0)+          ; PUT BINARY KEYBOARD INPUT INTO ARRAY
        BEQ     20$                 ; ZERO MARKS END OF INPUT
        INC     (R2)                ;
        CMP     (R2),#10.
        BNE     10$                 ; IF NE YES
```

```
20$:
        MOV     (R2),LEN          ; CALCULATE LENGTH OF ARRAY
        MOV     #ARGBLK,R5        ; GET ADDRESS OF ARGUMENT BLOCK
        MOV     #2,(R5)+          ; NUMBER OF ARGUMENTS
        MOV     #IARRAY,(R5)+     ; PUT ADDRESS OF ARRAY
        MOV     #LEN,(R5)         ;
        MOV     #ARGBLK,R5        ;
        CALL    SORT              ; SORT ARRAY
;+
;Task Builder replaced call to SORT subroutine in SUPLIB with 4-word
;context switching vector. Flow of control switches to SUPLIB via
;the vector and back via the completion routine $CMPCS. TSUP
;continues excuting at the next instruction.
;-
        CLR     R2                ;
        MOV     #IARRAY,R0        ; GET ARRAY ADDRESS
30$:
        INC     R2                ; INCREMENT INDEX
        MOV     R2,EDBUF          ; GET INDEX FOR PRINT
        MOV     (R0)+,EDBUF+2     ; GET CONTENTS OF ARRAY
        MOV     #FMT5,R1          ; GET ADDRESS OF FORMAT SPECIFICATION
        CALL    PRINT             ;
        CMP     R2,LEN            ; MORE TO PRINT?
        BLT     30$               ; IF LE YES
        MOV     #FMT2,R1          ; GET ADDRESS OF FORMAT SPECIFICATION
        CALL    PRINT             ; OUTPUT MESSAGE
        CALL    READ              ; READ RESPONSE
        MOV     #ARGBLK,R5        ;
        MOV     #4,(R5)+          ; SET NUMBER OF ARGUMENTS
        MOV     #IART,(R5)+       ; SET ADDRESS OF NUMBER LOOKING FOR
        MOV     #IARRAY,(R5)+     ; SET ADDRESS OF ARRAY
        MOV     #LEN,(R5)+        ; SET ADDRESS OF LEN OF ARRAY
        MOV     #INDEX,(R5)       ; ADDRESS OF RESULT
        MOV     #ARGBLK,R5        ;
        CALL    SEARCH            ; SEARCH FOR NUMBER IN IART
;
;Call to SUPLIB for SEARCH subroutine.
;
        TST     INDEX             ; WAS NUMBER FOUND?
        BLT     40$               ; IF LT NO
        MOV     IART,EDBUF        ; GET NUMBER LOOKING FOR
        MOV     INDEX,EDBUF+2     ; GET ARRAY NUMBER
        MOV     #FMT3,R1          ; GET FORMAT ADDRESS
        CALL    PRINT             ;
        BR      100$              ; DONE
40$:
        MOV     #FMT4,R1          ; GET FORMAT ADDRESS
        MOV     IART,EDBUF        ; GET NUMBER
        CALL    PRINT
100$:
        CALL    $EXST             ; EXIT WITH STATUS
```

```
PRINT:
        CALL    $SAVAL              ; SAVE ALL REGISTERS
        MOV     #OUT,R0             ; ADDRESS OF OUTPUT BLOCK
        MOV     #EDBUF,R2           ; START ADDRESS OF ARGUMENT BLOCK
        CALL    $EDMSG              ; FORMAT MESSAGE
        MOV     R1,WRITE+Q.IOPL+2 ; PUT LENGTH OF OUTPUT
                                    ; BLOCK INTO PARAMETER BLOCK
        DIR$    #WRITE              ; WRITE OUTPUT BLOCK
        RETURN

READ:
        CALL    $SAVAL              ; SAVE ALL REGISTERS
```

```
        DIR$    #READIN ; READ REQUEST
        MOV     #OUT,R0 ; GET KEYBOARD INPUT
        CALL    $CDTB               ; CONVERT KEYBOARD INPUT TO BINARY
        MOV     R1,IART             ; PUT INPUT INTO BUFFER
        RETURN

        .END    START
```

Example 4 Memory Allocation Map for TSUP

```
TSUP.TSK;1   Memory allocation map  TKB D40.0        Page 1
                       8-DEC-84   00:32


Identification : 01
Stack    limits: 000000 000777 001000 00512.
PRG xfr address: 001616
Total attachment descriptors: 3.
Task  image  size   : 800. WORDS
Task header size    : 192. WORDS
R-O region size: 448. WORDS
Task address limits: 000000 003043
R-W disk blk limits: 000003 000006 000004 00004.
R-O disk blk limits: 000007 000010 000002 00002.

*** Root segment: TSUP


R/W mem  limits: 000000 003043 003044 01572.
R-O mem  limits: 160000 161577 001600 00896.
Disk blk limits: 000003 000006 000004 00004.


Memory allocation synopsis:

Section                                                 Title  Ident  File
-------                                                 -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)     001000 002042 01058.
                              001000 001220 00656. TSUP   01     TSUP.OBJ;1
CMPAL :(RW,I,GBL,REL,CON)     000000 000342 00226.
PUR$D :(RO,D,LCL,REL,CON)     160000 000146 00102.
PUR$I :(RO,I,LCL,REL,CON)     160146 001204 00644.
$$RESL:(RO,I,LCL,REL,CON)     161352 000124 00084.
$$SLVC:(RO,I,LCL,REL,CON)     161476 000020 00016.



*** Task builder statistics:

    Total work file references: 2215.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 15014. words (58. pages)
    Size of work file: 1024. words (4. pages)

    Elapsed time:00:00:14
```

TSUP prompts you to enter numbers at your terminal. It calls a subroutine in SUPER to sort the numbers. Then it displays the numbers you entered as array entries and prompts you to request a number to search for. TSUP calls a subroutine in SUPER to search for the number. Finally, TSUP indicates at your terminal either that the number was not found or the array location in which the number is stored.

Building SUPER - To build SUPER in UFD [301,55] on SY:, use the following TKB or LINK command sequence:

```
TKB>SUPER/-HD/LI/PI,SUPER/MA,SUPER=
TKB>LB:[1,2]SYSLIB/LB:CMPAL,SY:[301,55]SUPER
TKB>/
Enter Options:
TKB>STACK=0
TKB>UIC=[1,1]
TKB>UNITS=0
TKB>CMPRT=$CMPCS
TKB>GBLXCL=$SAVAL
TKB>//

LINK/TAS:SUPER/NOH/LIB/PDSIS/MAP:SUPER/SYMBOL:SUPER/OPT -
LB:[1,2]SYSLIB/INC:CMPAL,SY:[301,55]SUPER
Option? STACK=0
Option? PAR=UIC=[1,1]
Option? UNITS=0
Option? CMPRT=$CMPCS
Option? GBLXCL=$SAVAL
Option? <RET>
```

SUPER is built without a header or stack. It is position independent and has only one program section, named .BLK. The /LI switch or /LIB qualifier eliminates program section name conflicts between the library and the referencing task.

The completion routine module of SYSLIB, CMPAL, is specified first in the input line.

The GBLXCL option excludes $SAVAL from the library's .STB file. You exclude $SAVAL from the .STB file because the referencing task, TSUP, also calls $SAVAL. If TSUP finds $SAVAL in the .STB file of SUPER, it will not link a separate copy of $SAVAL into its task image from SYSLIB. If TSUP cannot link to a copy of $SAVAL that is mapped through user APRs, the TSUP would call $SAVAL as a subroutine residing within the supervisor-mode library, but without the necessary mode-switching vector and completion routine support. This option forces TKB to link $SAVAL from SYSLIB into the task image for TSUP.

SUPERVISOR-MODE LIBRARIES

The memory allocation map in Example 2 shows the following:

- o SUPER begins at virtual 0.
- o The completion routine, $CMPAL, is linked into the library from SYSLIB at virtual 0.
- o The entry point $CMPAL is located at virtual 22, SEARCH is located at 220, and sort is located at 140. All of these entry points are relocatable.

Building TSUP - Use the following TKB or LINK command sequence to build a task, TSUP, that links to SUPER:

```
TKB>TSUP,TSUP=TSUP
TKB>/
Enter Options:
TKB>RESSUP=SUPER/SV:0
TKB>//

LINK/TAS/MAP/OPT SUPER
Option? RESSUP=SUPER/SV:0
Option? <RET>
```

These two command sequences tell TKB to include in the logical address space of TSUP a user-owned, supervisor-mode library named SUPER. TKB includes a 4-word, mode-switching vector within the task image for each call to a subroutine within the library. The library is position independent and is mapped with supervisor I-space APR0. This is a requirement for CSM libraries because the CSM expects to find the entry point of the completion routine at location 10.

he memory allocation map for TSUP (Example 4) shows:

- o $CMPAL is linked from the .STB file of the library and begins at location 0.
- o The mode-switching vectors begin at 161476 and are 16. bytes. That means that TSUP calls subroutines within the library 2 times (4 words per vector).

Running TSUP - After building SUPER and TSUP as indicated in the task-build command sequence discussed previously, you install SUPER and run TSUP. TSUP prompts you for a number:

```
ARRAY (x)
```

x

The position in which to store the number in the array. You enter a number. TSUP stores the number in the array and prompts you again for a number. This continues until you either have entered a 0, an illegal number, or 10 numbers. Then TSUP calls the SORT routine in SUPER.

You enter a number. TSUP calls the SEARCH routine in SUPER. Then TSUP outputs a message indicating whether the number was in the array.

The CSM Library Dispatching Process

When you build the referencing task, if you specify the SV argument to the RESSUP or SUPLIB option, then TKB includes a 4-word context-switching vector for each call to a subroutine in the library. This section discusses the CSM library vector in detail.

CSM mode switching occurs as follows:

1. The vector is entered with the return address on top of the stack (TOS).
2. The vector pushes the completion routine address on the stack.
3. A CSM instruction is executed with the supervisor-mode entry point as the immediate addressing mode parameter. The CSM instruction:
   a. Evaluates the source parameter and stores the entry point address in a temporary register
   b. Copies the user stack pointer to the supervisor stack pointer
   c. Places the current PS and PC on the supervisor stack clearing the condition codes in the PS
   d. Pushes the entry point address on the supervisor stack
   e. Places the contents of location 10 in supervisor I-space into the PC

The stack looks like this when the processor begins to execute at the contents of virtual 10 in supervisor mode:

```
        user sp ---->  return address
                       completion routine address
                                PS
                                PC

        super sp ---->  entry point address
```

The most important aspect of how the CSM library mode-switching vector works is that the processor begins executing at the contents of virtual 10 in supervisor mode. This is why the completion routine must be located at virtual 0, so that virtual location 10 is within the completion routine.

SUPERVISOR-MODE LIBRARIES

USING SUPERVISOR-MODE LIBRARIES AS RESIDENT LIBRARIES

Supervisor-mode libraries can double as conventional resident libraries. For position-independent, supervisor-mode libraries, you rebuild the referencing task using the RESLIB option instead of the RESSUP option. Indicate the first available user-mode APR that you want to map the library. For CSM libraries this will always change, because you cannot map a shared region with APR 0. You do not have to rebuild the library.

MULTIPLE SUPERVISOR-MODE LIBRARIES

A user task can reference multiple supervisor-mode CSM libraries. However, all the CSM libraries must use the completion routine that begins at virtual zero in supervisor-mode instruction space.

LINKING SUPERVISOR-MODE LIBRARIES

You cannot link supervisor-mode libraries together, and you cannot link a supervisor-mode library to a resident user-mode library. Calling a user-mode library is not possible because its code is not mapped through the I-space APRs while in the supervisor-mode library. However, you can link user-mode libraries to a supervisor-mode library.

WRITING YOUR OWN VECTORS AND COMPLETION ROUTINES

You can write your own mode-switching vectors and completion routines. This may be necessary for threaded code. If you use your own vectors, build them into the task and use the -SV switch on the RESSUP or RESLIB option when you build the referencing task. If you create your own completion routines, write your completion routine to resemble the system-supplied completion routines (see Appendix A) as much as possible.

APPENDIX A

Supervisor Mode Library Dispatch Completion Routines

Executive Emulation of the CSM Instruction

Command File to Build a Supervisor Mode FCS Library (FCSSL)

Supervisor Mode Library Dispatch and Completion Routines

```
        .TITLE  CMPAL
        .IDENT  /0204/

;
;               COPYRIGHT (c) 1981 BY
;       DIGITAL EQUIPMENT CORPORATION, MAYNARD
;        MASSACHUSETTS.  ALL RIGHTS RESERVED.
;
; COPYRIGHT (c) 1982 BY DIGTIAL EQUIPMENT CORPORATION
;
; THIS  SOFTWARE  IS  FURNISHED  UNDER  A LICENSE AND MAY BE USED
; AND  COPIED  ONLY IN  ACCORDANCE WITH THE TERMS OF SUCH LICENSE
; AND WITH  THE INCLUSION  OF THE ABOVE  COPYRIGHT  NOTICE.  THIS
; SOFTWARE  OR ANY OTHER  COPIES  THEREOF, MAY NOT BE PROVIDED OR
; OTHERWISE MADE  AVAILABLE TO ANY OTHER PERSON.  NO TITLE TO AND
; OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERED.
;
; THE INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT  BY  DIGITAL
; EQUIPMENT CORPORATION.
;
; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF
; ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
;

        .ENABL  LC

;
; The CSM dispatcher routine and the standard completion routines,
; $CMPAL and $CMPCS are included in this module due to the close
; interaction between them.
;


;
; **-CSM Dispatcher-Dispatch CSM entry
;
; This module must be linked at virtual zero in the supervisor mode
; library.  It is entered via a four word transfer vector of the form:
;
;       MOV     #completion-routine,-(SP)
;       CSM     #routine
;
; The CSM instruction transfers control to the address contained in
; supervisor mode virtual 10.  At this point the stack is the following:
;
;         (SP)  routine address
;        2(SP)  PC (past end of transfer vector)
;        4(SP)  PS with condition codes cleared
;        6(SP)  Completion-routine address
;       10(SP)  Return address
```

Supervisor Mode Library Dispatch and Completion Routines

```
;
; A routine address of 0 is special cased to support return to
; supervisor mode from a user mode debugging aid (ODT).  In this case
; stack is the following:
;
;          (SP)   zero
;         2(SP)   PC from CSM to be discarded
;         4(SP)   PS from CSM to be discarded
;         6(SP)   Super mode PC supplied by debugger
;        10(SP)   Super mode PS supplied by debugger
;

; To allow positioning at virtual zero, this code must be in the blank
; PSECT which is first in the TKBs PSECT ordering.

        .PSECT

        .ENABL  LSB

; Debugger return to super mode entry. Must start at virtual zero

        CMP     (SP)+,(SP)+     ; Clean off PS and PC from CSM

;
; **-$SRTI-Super mode RTI
;
; This entry point performs the necessary stack management to allow
; an RTI from super mode to either super mode or user mode.
; The is as required for an RTI:
;
;          (SP)   Super mode PC
;         2(SP)   Super mode PS
;

$SRTI:: TST     2(SP)           ; Returning to user mode?
        BR      70$             ; Join common code

; CSM transfer address, this word must be at virtual 10 in super mode

        .WORD   CSMSVR          ; CSM dispatcher entry

; Dispatch CSM entry

CSMSVR: MOV     6(SP),2(SP)     ; Set completion routine address for RETURN
        JMP     @(SP)+          ; Transfer to super mode library routine
```

```
;
; **-$CMPAL-Completion routine which sets up NZVC in the PS
;
; Copy all condition codes to stacked PS.  Current stack:
;
;        (SP)   PS with condtion codes cleared
;       2(SP)   Completion routine address (to be discarded)
;       4(SP)   Return address
;

$CMPAL::BPL     40$             ;
        BNE     20$             ;
        BVC     10$             ;
        BIS     #16,(SP)        ; Set NZV
        BR      $CMPCS          ;
10$:    BIS     #14,(SP)        ; Set NZ
        BR      $CMPCS          ;
20$:    BVC     30$             ;
        BIS     #12,(SP)        ; Set NV
        BR      $CMPCS          ;
30$:    BIS     #10,(SP)        ; Set N
        BR      $CMPCS          ;
40$:    BNE     60$             ;
        BVC     50$             ;
        BIS     #6,(SP)         ; Set ZV
        BR      $CMPCS          ;
50$:    BIS     #4,(SP)         ; Set Z
        BR      $CMPCS          ;
60$:    BVC     $CMPCS          ;
        BIS     #2,(SP)         ; Set V

;
; **-$CMPCS-Completion routine which sets up only C in the PS
;
; Copy only carry to stacked PS.  Current stack:
;
;        (SP)   PS with condtion codes cleared
;       2(SP)   Completion routine address (to be discarded)
;       4(SP)   Return address
;

$CMPCS::ADC     (SP)            ; Set up carry
        MOV     4(SP),2(SP)     ; Setup return address for RTT
        MOV     (SP)+,2(SP)     ; And PS. Returning to super mode?
70$:    BPL     80$             ; If PL yes
        MOV     #6,-(SP)        ; Number of bytes for (SP), PS, and PC
        ADD     SP,(SP)         ; Compute clean stack value
        MTPI    SP              ; Set up previous stack pointer
80$:    RTT                     ; Return to previous mode and caller

        .DSABL  LSB

        .END
        Executive emulation of the CSM instruction.

;
; This routine is entered when the CSM instruction is executed
; on a processor that does not support CSM. It will field the illegal
; instruction trap and emulate the CSM instruction.
;
; NOTE: Only one form of the CSM instruction is supported.
;
;       MOV     #<Address of super routine>,-(SP)
;       CSM     (SP)+
;
; Kernel stack on entry.
;
;       0(SP) = Trap PC + 2
;       2(SP) = PS prior to trap
;
; Kernel stack immediately before RTI instruction.
;
;       0(SP) = Contents of virtual address 10 in supervisor mode
;       2(SP) = PS with proper previous and current mode SUPER
;
; Super stack immediately before RTI instruction.
;
;       0(SP) = Address of supervisor mode routine
;       2(SP) = Return address
;       4(SP) = PS Prior to trap
;
;-
RESERV::
        BIT     #PRVMOD,PS.EXP  ;;;Trap occur in kernal mode?
        BEQ     170$            ;;;BR if yes, CSM not legal in kernal mode
        BITB    #SYF.SD,SY$FLG  ;;;Super D-space on?
        BEQ     170$            ;;;If EQ no - illegal instruction trap
        BIT     #ON.CSM,SR3     ;;;Is the instruction legal on this machine?
        BNE     170$            ;;;If NE yes - illegal instruction trap

        MOV     (SP),-(SP)      ;;;Copy trap PC + 2
        SUB     #2,(SP)         ;;;Calculate trap PC
        .GETPI  @(SP)+,BADCSM   ;;;Pick up trap instruction
        CMP     (SP)+,#7027     ;;;Is it CSM #N ?
        BNE     170$            ;;;If NE no - illegal instruction trap
        MOV     R0,-(SP)        ;;;Save R0
        MOV     2(SP),-(SP)     ;;;Get address of second half of instruction
        .GETPI  @(SP)+,BADCSM   ;;;Get immediate operand
157$:   MOV     SP,R0           ;;;Copy kernel stack pointer
        ADD     #4,R0           ;;;Point to trap PC
        MOV     (R0)+,-(SP)     ;;;Push PC of caller
        MOV     (R0),-(SP)      ;;;Push PS of caller
        MOV     #CURSUP!PRVMOD!RSET1,(R0) ;;; Current=super, previous=user
        BIC     #17,(SP)        ;;;Clear stacked condition codes
        BMI     16$             ;;;If mi previous mode was user
        BIC     #^CPRVSUP&PRVMOD,(R0) ;;;Set previous mode to super
16$:    MFPI    SP              ;;;Get previous mode SP
        MOV     (SP),R0         ;;;Copy it
        SUB     #6,(SP)         ;;;Adjust SP for pushed PS,PC,Super routine
        BIC     #^CPRVSUP&PRVMOD,PS.EXP ;;;Set previous mode to super
```

Executive emulation of the CSM instruction.

```
MTPI    SP              ;;;Set super SP
.PUTP   -(R0),BADCSM    ;;;Push PS on user stack
.PUTP   -(R0),BADCSM    ;;;Push PC on super stack
.PUTP   -(R0),BADCSM    ;;;Push address of super routine
MOV     (SP)+,R0        ;;;Restore R0
TST     (SP)+           ;;;Point to trap PS
.GETPI  10,BADCSM       ;;;Set return PC in super mode from super 10
RTI                     ;;;Return to (super virtual 10)
```

Command File to Build a Supervisor Mode FCS Library

```
[1,1]FCSSL/LI/-HD,[111,2]FCSSL/-SP/MA/CR,[1,1]FCSSL=
;
; The completion routine MUST be first in the library image
;
[1,1]SYSLIB/LB:CMPAL
;
; Begin the list of modules to include
;
[1,1]SYSLIB/LB:OPNJMP:OPENR
[1,1]SYSLIB/LB:.ODCVT:ASSLUN:ASCPPN:PPNASC:MKDL
[1,1]SYSLIB/LB:PARSE:PARSDI:PARSDV:PARSFN:DIRECT:DEL
[1,1]SYSLIB/LB:RQLCB:CLOSE
[1,1]SYSLIB/LB:UDIREC:EXTEND:TRNCLS:PRINT:DSPAT:PPNR50
[1,1]SYSLIB/LB:GETDID:MRKDL:DELETE:DELJMP:RENAME:GETDIR
[1,1]SYSLIB/LB:GPTJMP:GET:PUT:FLUSH
[1,1]SYSLIB/LB:READ:WRITE:FCSTYP
[1,1]SYSLIB/LB:ARITH:DARITH:CONTRL:POSIT:POSREC
[1,1]SYSLIB/LB:PNTMRK:RWFSR2:WAITU:XQIOU
[1,1]SYSLIB/LB:CDDMG:CBTA:CATB:EXST
[1,1]SYSLIB/LB:EDMGSL:EDDTSL
[1,1]SYSLIB/LB:ULA:ULAFD:ULAIN
[1,1]SYSLIB/LB:.CSI1:.CSI2:OD2CT
;
/
IDENT=3.201
CMPRT=$CMPCS
STACK=0
UNITS=0
BASE=0
UIC=[1,1]
;
;Exclude all the routines that have parameters passed on the stack
;
GBLXCL=.SAVR1
GBLXCL=$SAVRG
;
;Exclude all the internal fcs routines
;
GBLXCL=..ALC1
GBLXCL=..ALOC
GBLXCL=..ALUN
GBLXCL=..ANSP
GBLXCL=..BDRC
GBLXCL=..BKRG
GBLXCL=..CEFB
GBLXCL=..CREA
GBLXCL=..CTRL
GBLXCL=..DEL1
GBLXCL=..DFDI
GBLXCL=..DID
GBLXCL=..DIDF
GBLXCL=..DID1
GBLXCL=..DIRF
GBLXCL=..EFCK
GBLXCL=..EFC1
```

```
GBLXCL=..ENTR
GBLXCL=..EXTD
GBLXCL=..EXT1
GBLXCL=..FCSX
GBLXCL=..FIND
GBLXCL=..FINI
GBLXCL=..GTDI
GBLXCL=..IBB1
GBLXCL=..IDPB
GBLXCL=..INBB
GBLXCL=..MFID
GBLXCL=..MKDL
GBLXCL=..MLD
GBLXCL=..MVR1
GBLXCL=..PARS
GBLXCL=..PDI
GBLXCL=..PDID
GBLXCL=..PGCR
GBLXCL=..PNT1
GBLXCL=..PSDI
GBLXCL=..PSDV
GBLXCL=..PSFN
GBLXCL=..PSIT
GBLXCL=..PSRC
GBLXCL=..PSRG
GBLXCL=..PSR1
GBLXCL=..QIOW
GBLXCL=..RBLK
GBLXCL=..RDRN
GBLXCL=..RFDB
GBLXCL=..RMOV
GBLXCL=..RSEF
GBLXCL=..RTAD
GBLXCL=..RWAC
GBLXCL=..RWAT
GBLXCL=..RWCK
GBLXCL=..RWLG
GBLXCL=..SCR5
GBLXCL=..SEFB
GBLXCL=..SGR5
GBLXCL=..STFN
GBLXCL=..WAEF
GBLXCL=..WAIT
GBLXCL=..WAND
GBLXCL=..WAST
GBLXCL=..WBLK
GBLXCL=..WTRD
GBLXCL=..WTWA
GBLXCL=..WTWD
GBLXCL=..WTW1
GBLXCL=..XQIO
GBLXCL=..XQI1
GBLXCL=.FATAL
GBLXCL=.OFNBR
GBLXCL=.OPENR
GBLXCL=$DSW
```

```
;
; Exclude OD2CT entry point used only by .CSI2
;
GBLXCL=..D2CT
;
; Exclude any potential multiply defined equates
;
GBLXCL=CR
GBLXCL=FF
;
; Exclude infrequently used routines
;
GBLXCL=$DSPAT
GBLXCL=.CTRL
/
```

IAS WHIMS

WHAT: (Describe your WHIM) (Please print)

WHY: (Describe the reason for the WHIM)

HOW: (Make any suggestions for a possible implementation)

Name: ______________	Please mail to:
Company: ______________	Kathleen M. Anderson
______________	EATON Information Management Systems Division
Address: ______________	2017 Cunningham Drive
______________	Suite 208
______________	Hampton, Virginia 23666
Phone: ______________	Phone: (804) 826-1941

Printed in the U.S.A.