

Table of contents

3-	1	LDMEMT -- EMT to mount a logical disk
4-	1	LDDEMT -- EMT to dismount a logical disk
5-	1	LDIEMT -- EMT to get information about a logical disk
6-	1	LDAEMT -- Dismount all logical disks
7-	1	LDDMT -- Dismount a logical disk
8-	1	LDCLEN -- Perform SET LD CLEAN operation
9-	1	LDMNT -- Set up information about LD being mounted
10-	1	ASNSRC -- Search assign table for logical name
11-	1	LOGASN -- Perform full logical device assignment
12-	1	DEADEV -- Deassign physical device
13-	1	LDDEVN -- See if device name is a LD unit
14-	1	ADLDAC -- Add LD entry to access control table
15-	1	DLLDAC -- Delete LD entry from access control table
16-	1	CKLDAC -- Check if LD is in access control table
17-	1	LDOFCK -- See if any files are open on a logical disk

```

1          .TITLE  TSEM5  TSX-Plus EMT Overlay
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
5
6          ; Copyright (C) 1986.
7
8          ; S&H Computer Systems, Inc.
9          ; Nashville, Tennessee
10
11         ; This software is furnished under a license for use only
12         ; on a single computer system and may be copied only with
13         ; the inclusion of the above copyright notice.  This
14         ; software, or any other copies thereof, may not be provided
15         ; or otherwise made available to any other person except
16         ; for use on such system and to one who agrees to these
17         ; license terms.  Title to and ownership of the software
18         ; shall at all times remain with S&H Computer Systems, Inc.
19
20 000000          .CSECT  TSEM5
21 000000 020553 TSEM2: .RAD50  /EM5/          ;Overlay id
22          177776 PS      =      177776      ;Processor Status Word
23
24         ; Macro calls
25
26         .MCALL  .FPROT, .LOOKUP, .PURGE
27
28         ; Global definitions
29
30         .GLOBL  LDMENT, LDDENT, LDIEMT, LDAEMT
31
32         ; Global references
33
34         .GLOBL  CXTRMN, R$CHN, R$XCHN, CHNSIZ, NLCHN
35         .GLOBL  LDSIZE, C. DEVQ, C. SBLK, CS$RON, LDBASE, LDDEVX
36         .GLOBL  OVRHC, CHNADR, CS$OPN, C. CSW, SETERR, EMTBLK, MAXLD
37         .GLOBL  VLDSYS, VALADW, CSIFIL, CSIARE, LDNAME, LDFLAG
38         .GLOBL  LD$RON, SERFLG, EMTXIT, LDPDEV, MAXASN, ASNTBL, AT$LOG
39         .GLOBL  AT$$SZ, AT$DEV, AT$FIL, AT$EXT, AT$SIZ, SYNAME
40         .GLOBL  RESDEV, OKFILE, OKFAND, OKFNND, OF$FIL, OF$$SZ, OF$DEV
41         .GLOBL  OF$UNT, OF$FLG, OT$RON, WLDNAM
42
43         ; Data areas
44
45 000002 045676 R5OLD0: .RAD50  /LD0/
46 000004 045705 R5OLD7: .RAD50  /LD7/
47 000006 045640 R5OLD:  .RAD50  /LD /
48 000010 015270 R5ODK:  .RAD50  /DK /
49 000012 075250 R5OSY:  .RAD50  /SY /
50
51         ; EMT arg block to enable caching for a device
52
53 000014      000      134 MNTARG: .BYTE  0,134
54 000016 000000G      WORD  CSIARE
55
56         ; EMT arg block to dismount a device
57

```

58	000020	000	135	DMTARG:	. BYTE	0,135
59	000022	000000G			. WORD	CSIARE

```
1 ; -----
2 ; Macros to enable and disable interrupts.
3 ;
4 .MACRO DISABL ;DISABLE INTERRUPTS
5 BIS #340, @#PS
6 .ENDM DISABL
7 ;
8 .MACRO ENABL ;ENABLE INTERRUPTS
9 BIC INTPRI, @#PS
10 .ENDM ENABL
11 ; -----
12 ; Macro to print an error message when a system crash occurs.
13 ;
14 ; Arguments:
15 ; MSG = Name of error message to print.
16 ; ARG = (Optional) argument value to display with error message.
17 ;
18 .GLOBL DIEMSG, DIEARG, SYSHLT
19 .MACRO DIE MSG, ARG
20 MOV MSG, @#DIEMSG
21 .IF NB, ARG
22 MOV ARG, @#DIEARG
23 .ENDC
24 CALL @#SYSHLT
25 .ENDM DIE
26 ; -----
27 ; Macro definition for calling global routines residing in mapped
28 ; system regions.
29 ;
30 ;
31 .MACRO OCALL ENTADD
32 .IF B, ENTADD
33 .ERROR ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
34 .MEXIT
35 .ENDC
36 CALL OVRHC ;CALL THE OVERLAY HANDLER
37 .WORD ENTADD ;SPECIFY THE ENTRY POINT
38 .ENDM
```

```

1          .SBTTL  LDMENT -- EMT to mount a logical disk
2          ;-----
3          ; EMT to mount a logical disk:
4          ; Argument block format:
5          ;
6          ; .BYTE   chan,163
7          ; .BYTE   ld_number,flags
8          ; .WORD   file_spec_pointer
9          ;
10         LDMENT:
11         ;
12         ; Make sure the specified channel is closed
13         ;
14         000024 013703 0000000  MOV     CHNADR,R3      ;Get address of channel block
15         000030 032763 0000000 0000000  BIT     #CS#OPN,C.CSW(R3);Is the channel open or closed?
16         000036 001403          BEQ     1$           ;Br if closed
17         000040 005000          CLR     R0           ;Error 0 -- Channel is open
18         000042 000137 0000000  JMP     SETERR
19         ;
20         ; Make sure the logical disk number is valid
21         ;
22         000046 113702 0000020 1$:  MOVB    EMTBLK+2,R2      ;Get specified LD number
23         000052 020227 0000000  CMP     R2,#MAXLD      ;Must be in range 0 -- MAXLD
24         000056 103404          BLD     2$           ;Br if ok
25         000060 012700 0000001  MOV     #1,R0          ;Error 1 -- Invalid LD number
26         000064 000137 0000000  JMP     SETERR
27         ;
28         ; Make sure LD support included in system
29         ;
30         000070 105737 0000000 2$:  TSTB    VLDSYS          ;Is LD support included in system?
31         000074 001004          BNE     3$           ;Br if yes
32         000076 012700 0000002  MOV     #2,R0          ;Error 2 -- LD support not included in system
33         000102 000137 0000000  JMP     SETERR
34         ;
35         ; Error if this LD is already mounted
36         ;
37         000106 006302 3$:  ASL     R2              ;Get LD index
38         000110 010200  MOV     R2,R0              ;Get LD index
39         000112 072027 0000002  ASH     #2,R0          ;Convert to index into LDNAME table
40         000116 005760 0000000  TST     LDNAME(R0)      ;Is this LD unit already mounted?
41         000122 001404          BEQ     4$           ;Br if not
42         000124 012700 0000003  MOV     #3,R0          ;Error 3 -- LD unit already mounted
43         000130 000137 0000000  JMP     SETERR
44         ;
45         ; Move file spec to internal buffer
46         ;
47         000134 013700 0000040 4$:  MOV     EMTBLK+4,R0      ;Get address of user's file spec
48         000140 001423          BEQ     6$           ;Br if no file spec specified
49         000142 004737 0000000  CALL    VALADW          ;Make sure it is valid
50         000146 012704 0000000  MOV     #CSIFIL,R4      ;Point to area where we will store file spec
51         000152 012703 0000004  MOV     #4,R3           ;Get # words to move
52         000156 106520 5$:  MFPD    (R0)+          ;Get word from user's area
53         000160 012624  MOV     (SP)+,(R4)+          ;Move to internal area
54         000162 077303  SOB     R3,5$              ;Loop till all of file spec moved
55         ;
56         ; Translate logical file device name to physical device
57         ; and make sure we have a non-null file name.

```

```

58
59 000164 012705 0000000 MOV    #CSIFIL,R5      ;Point to file spec to be translated
60 000170 004737 001466' CALL    LOGASN        ;Do the assign
61 000174 005737 0000000 TST     CSIFIL         ;Is device name null?
62 000200 001403 BEQ     6$           ;Br if yes
63 000202 005737 0000020 TST     CSIFIL+2       ;Is file name null?
64 000206 001004 BNE     7$           ;Br if not
65 000210 012700 000004 6$: MOV    #4,R0      ;Error 4 -- Invalid file spec
66 000214 000137 0000000 JMP     SETERR
67
68 ; Make sure file is not on same LD as is being mounted
69
70 000220 013700 0000000 7$: MOV    CSIFIL,R0      ;Get name of device file is on
71 000224 004737 001640' CALL    LDDEVN        ;See if file is on a LD device
72 000230 103406 BCS     8$           ;Br if not
73 000232 020002 CMP     R0,R2      ;Is it on a lower # LD?
74 000234 103404 BLO     8$           ;Br if yes
75 000236 012700 000005 MOV    #5,R0      ;Error 5 -- File is on an invalid LD
76 000242 000137 0000000 JMP     SETERR
77
78 ; Move file name into logical disk name table
79
80 000246 010205 8$: MOV    R2,R5      ;Get LD index number
81 000250 072527 000002 ASH     #2,R5        ;Cvt to index into name table
82 000254 062705 0000000 ADD     #LDNAME,R5    ;Point to name entry for this LD
83 000260 012704 0000000 MOV    #CSIFIL,R4     ;Point to cell with file spec
84 000264 012700 000004 MOV    #4,R0      ;Get # words to move
85 000270 012425 9$: MOV    (R4)+,(R5)+ ;Move file spec into LD name table
86 000272 077002 SOB     R0,9$      ;Loop till all of spec moved
87
88 ; Set up flags for logical disk
89
90 000274 005062 0000000 CLR     LDFLAG(R2)    ;Clear all flags
91 000300 105737 0000030 TSTB    EMTBLK+3      ;Was NOWRITE flag specified?
92 000304 001403 BEQ     10$        ;Br if not
93 000306 052762 0000000 0000000 BIS     #LD$RON,LDFLAG(R2);Set NOWRITE flag to the LD
94
95 ; Lookup file and set up information about position of logical
96 ; disk on physical disk.
97
98 000314 113703 0000000 10$: MOVB    EMTBLK,R3      ;Get channel # for lookup
99 000320 004737 001174' CALL    LDMNT        ;Set up info about file for logical disk
100 000324 103004 BCC     11$        ;Br if ok
101 000326 012700 000006 MOV    #6,R0      ;Error 6 -- Cannot lookup file
102 000332 000137 0000000 JMP     SETERR        ;Error on lookup
103
104 ; Protect the file
105
106 000336 113746 0000000 11$: MOVB    SERFLG,-(SP) ;Save hard-error flag
107 000342 112737 000001 0000000 MOVB    #1,SERFLG    ;Don't abort on hard errors
108 000350 113703 0000000 MOVB    EMTBLK,R3      ;Get channel number
109 000354 FPROT    #CSIARE,R3,#CSIFIL,#1 ;Protect the file
110 000404 112637 0000000 MOVB    (SP)+,SERFLG ;Restore error control
111
112 ; Initiate directory caching
113
114 000410 113700 0000020 MOVB    EMTBLK+2,R0    ;Get LD #

```

LDMEMT -- EMT to mount a logical disk

```
115 000414 063700 000002'      ADD    R5OLD0,R0      ;Add "LDO" to form rad50 device name
116 000420 010037 000000G      MOV    R0,CSIARE      ;Store in cell pointed to by arg block
117 000424 012700 000014'      MOV    #MNTARG,R0      ;Point to EMT argument block
118 000430 104375                EMT    375          ;Enable caching for the device
119                               ;
120                               ; Now do a SET LD CLEAN operation to make sure all logical disks
121                               ; are set up correctly.
122                               ;
123 000432 113703 000000G      MOVB   EMTBLK,R3      ;Get channel number for lookups
124 000436 004737 001050'      CALL   LDCLN          ;Do SET LD CLEAN
125                               ;
126                               ; Finished mounting the EMT
127                               ;
128 000442 000137 000000G      JMP     EMTXIT
```

```

1          .SBTTL  LDDEMT -- EMT to dismount a logical disk
2          -----
3          ; The form of the EMT used to dismount a logical disk is:
4          ;
5          ; .BYTE 3,135
6          ; .BYTE ld_unit,0
7          ;
8 000446 LDDEMT:
9          ;
10         ; Make sure the LD unit number is valid
11         ;
12 000446 113702 0000020 MOVBLK+2,R2 ;Get LD unit #
13 000452 020227 0000000 CMP R2,#MAXLD ;See if valid
14 000456 103404 BLO 1$ ;Br if ok
15 000460 012700 0000001 MOV #1,R0 ;Error 1 -- Invalid LD #
16 000464 000137 0000000 JMP SETERR
17         ;
18         ; See if the LD is active
19         ;
20 000470 006302 1$: ASL R2 ;Convert to unit index
21 000472 010200 MOV R2,R0 ;Get unit index
22 000474 072027 0000002 ASH #2,R0 ;Convert to index into LDNAME table
23 000500 005760 0000000 TST LDNAME(R0) ;Is this LD mounted?
24 000504 001003 BNE 2$ ;Br if yes
25 000506 005000 CLR R0 ;Error 0 -- Specified LD is not mounted
26 000510 000137 0000000 JMP SETERR
27         ;
28         ; Try to do the dismount
29         ;
30 000514 004737 000760' 2$: CALL LDDMT ;Try to dismount the LD
31 000520 103004 BCC 3$ ;Br if ok
32 000522 012700 0000003 MOV #3,R0 ;Error 3 -- Open file on LD being dismounted
33 000526 000137 0000000 JMP SETERR
34         ;
35         ; Finished
36         ;
37 000532 000137 0000000 3$: JMP EMTXIT ;Finished with dismount EMT

```

LDIEMT -- EMT to get information about a logical disk

```

1          .SBTTL  LDIEMT -- EMT to get information about a logical disk
2          ;-----
3          ; The LDIEMT is used to obtain information about a logical disk structure.
4          ;
5          ; The form of the argument block is:
6          ;
7          ;     .BYTE    4,135
8          ;     .BYTE    ld_number,0
9          ;     .WORD    buffer_pointer
10         ;
11 000536 LDIEMT:
12         ;
13         ; Make sure the LD unit number is valid
14         ;
15 000536 113702 000002G      MOVB    EMTBLK+2,R2      ;Get LD unit #
16 000542 020227 000000G      CMP     R2,#MAXLD      ;See if valid
17 000546 103404             BLO     7$              ;Br if ok
18 000550 012700 000001      MOV     #1,R0           ;Error 1 -- Invalid LD #
19 000554 000137 000000G      JMP     SETERR
20         ;
21         ; Make sure the buffer address is valid
22         ;
23 000560 013700 000004G 7$:  MOV     EMTBLK+4,R0      ;Get address of buffer
24 000564 004737 000000G      CALL    VALADW          ;Make sure it is valid
25         ;
26         ; Transfer the LD file name to the buffer
27         ;
28 000570 006302             ASL     R2              ;Get LD unit index
29 000572 010203             MOV     R2,R3           ;Get LD unit index
30 000574 072327 000002      ASH     #2,R3          ;Cvt to index into LDNAME table
31 000600 062703 000000G      ADD     #LDNAME,R3     ;Point to LDNAME entry
32 000604 005713             TST     (R3)           ;Is there a specified file name?
33 000606 001423             BEQ     2$              ;Br if not
34 000610 012704 000004      MOV     #4,R4          ;Move 4 words
35 000614 012346 1$:  MOV     (R3)+,-(SP)          ;Move a word of the file name
36 000616 106620             MTPD    (R0)+          ;to user's buffer
37 000620 077403             SOB     R4,1$
38         ;
39         ; Pass flag word
40         ;
41 000622 005046             CLR     -(SP)           ;Start with all flags off
42 000624 032762 000000G 000000G  BIT     #LD$RON,LDFLAG(R2); Is read-only flag set?
43 000632 001402             BEQ     5$              ;Br if not
44 000634 052716 000001      BIS     #1,(SP)        ;Set read-only flag for return
45 000640 005762 000000G 5$:  TST     LDPDEV(R2)     ;Is LD currently accessible?
46 000644 001002             BNE     6$              ;Br if yes
47 000646 052716 000002      BIS     #2,(SP)        ;Set not-accessible flag
48 000652 106620 6$:  MTPD    (R0)+          ;Move flag word to user's buffer
49 000654 000405             BR      9$
50         ;
51         ; This LD is not active.
52         ; Return all zeroes to user's buffer
53         ;
54 000656 012704 000005 2$:  MOV     #5,R4          ;We will move 5 words of zeroes
55 000662 005046 4$:  CLR     -(SP)          ;Move 5 words of zeroes
56 000664 106620             MTPD    (R0)+
57 000666 077403             SOB     R4,4$

```

TSEM5 TSX-Plus EMT Overlay MACRO V05.05 Wednesday 18-Jan-89 12:11 Page 5-1
LDIEMT -- EMT to get information about a logical disk

```
58 ;  
59 ; Finished  
60 ;  
61 000670 000137 0000000 9$: JMP EMTXIT ;Finished with EMT
```

1				.SBTTL LDAEMT -- Dismount all logical disks
2				-----
3				
4				; The form of the argument block is:
5				; .BYTE 5,135
6				; .WORD 0
7				
8	000674			LDAEMT:
9				
10				; Set up for loop to dismount all logical disks
11				
12	000674	005001		CLR R1 ;Set up error flag
13	000676	005002		CLR R2 ;First LD to be dismounted
14				
15				; Loop to dismount all logical disks
16				
17	000700			1\$:
18				
19				; See if the LD is active
20				
21	000700	010200		MOV R2,R0 ;Get unit index
22	000702	072027	000002	ASH #2,R0 ;Convert to index into LDNAME table
23	000706	005760	0000000	TST LDNAME(R0) ;Is this LD mounted?
24	000712	001405		BEG 2\$;Br if no
25				
26				; Dismount logical disks
27				
28	000714	004737	000760'	CALL LDDMT ;Attempt to dismount LD
29	000720	103002		BCC 2\$;Br if OK
30	000722	012701	000001	MOV #1,R1 ;Set error flag
31	000726			2\$:
32	000726	062702	000002	ADD #2,R2 ;Increment logical disk
33	000732	020227	000020	CMP R2,#16. ;All done?
34	000736	002760		BLT 1\$;Loop if not
35	000740	005701		TST R1 ;Was an error encountered?
36	000742	001404		BEG 3\$;Br if not
37	000744	012700	000003	MOV #3,R0 ;Error #3 ==> LD has open files
38	000750	000137	0000000	JMP SETERR
39				
40				; Finished
41				
42	000754	000137	0000000	3\$: JMP EMTXIT
43				

```

1          .SBTTL  LDDMT  -- Dismount a logical disk
2          ;-----
3          ; Dismount a logical disk
4          ;
5          ; Inputs:
6          ;   R2 = LD device index
7          ;
8          ; Outputs:
9          ;   C-flag cleared ==> No error on dismount
10         ;   C-flag set ==> Error on dismount.  Open file on LD being dismounted.
11         ;
12         ;
13 000760   LDDMT:
14         ;
15         ; Error if this job has any open files on LD being dismounted
16         ;
17 000760   004737   002206'   CALL    LDOFCK           ;Any open files on this LD?
18 000764   103430           BCS      9$              ;Br if open file on LD being dismounted
19         ;
20         ; Stop doing directory caching for the device
21         ;
22 000766   010200           MOV      R2,R0           ;Get the LD unit index
23 000770   006200           ASR      R0              ;Convert to unit number
24 000772   063700   000002'   ADD      R5OLD0,R0     ;Add "LD0" to form device name
25 000776   010037   000000G   MOV      R0,CSIARE     ;Store in work cell
26 001002   113746   000000G   MOVB    SERFLG,-(SP)   ;Save error control flag
27 001006   112737   000001'   000000G   MOVB    #1,SERFLG ;Suppress aborts
28 001014   012700   000020'   MOV      #DMTARG,R0   ;Stop caching on device
29 001020   104375           EMT      375
30 001022   112637   000000G   MOVB    (SP)+,SERFLG ;Reset error control
31         ;
32         ; Clear logical disk data tables
33         ;
34 001026   005062   000000G   CLR      LDPDEV(R2)    ;No physical device assignment
35 001032   010200           MOV      R2,R0           ;Get unit index
36 001034   072027   000002   ASH      #2,R0         ;Convert to index into LDNAME table
37 001040   005060   000000G   CLR      LDNAME(R0)   ;No file associated with LD
38         ;
39         ; Deassign any logical names assigned to this LD
40         ;
41         ;   MOV      CSIARE,R0           ;Get LD device name
42         ;   CALL     DEADEV             ;Deassign any logical names
43         ;
44         ; Successfully finished
45         ;
46 001044   000241           CLC                      ;Signal success on return
47         ;
48         ; Finished
49         ;
50 001046   000207           9$:    RETURN

```

```

1          .SBTTL  LDCLN -- Perform SET LD CLEAN operation
2          ;-----
3          ; LDCLN is called to perform the SET LD CLEAN function.
4          ; It also causes the file directory cache to be cleaned out.
5          ;
6          ; Inputs:
7          ;   R3 = Number of free channel to use for lookup
8          ;   CHNADR = Pointer to channel block of free channel
9          ;
10         LDCLN: MOV     R2, -(SP)
11              MOV     R5, -(SP)
12         ;
13         ; Perform LD CLEAN operation (reset logical disk information)
14         ;
15         001054 010246 000002'      MOV     R5, LDO, R5      ;Get name of first logical disk (LDO)
16         001060 005002              CLR     R2              ;Get index to 1st logical disk
17         ;
18         ; See if this logical disk is in use
19         ;
20         001062 010200 1$:          MOV     R2, R0              ;Get LD index number
21         001064 072027 000002        ASH     #2, R0          ;Convert to index into LDNAME table
22         001070 005760 000000G        TST     LDNAME(R0)      ;Is this logical disk in use?
23         001074 001426              BEQ     2$              ;Br if not
24         ;
25         ; Dismount the logical disk
26         ;
27         001076 010537 000000G        MOV     R5, CSIARE      ;Set device name to be dismounted
28         001102 113746 000000G        MOV     SERFLG, -(SP)    ;Save error control
29         001106 112737 000001 000000G  MOV     #1, SERFLG     ;Do .SERR
30         001114 012700 000020'        MOV     #DMTARG, R0      ;Point to EMT argument block
31         001120 104375              EMT     375              ;Dismount the device
32         001122 112637 000000G        MOV     (SP)+, SERFLG    ;Restore error control
33         ;
34         ; Now reinitialize information about the logical disk
35         ;
36         001126 004737 001174'        CALL    LDMNT           ;Reset information about the LD
37         ;
38         ; Now remount the logical disk
39         ;
40         001132 005762 000000G        TST     LDPDEV(R2)      ;Is the logical disk there?
41         001136 001405              BEQ     2$              ;Br if not
42         001140 010537 000000G        MOV     R5, CSIARE      ;Set device name for the mount
43         001144 012700 000014'        MOV     #MNTARG, R0      ;Mount the device
44         001150 104375              EMT     375
45         ;
46         ; Check the next logical disk
47         ;
48         001152 005205 2$:          INC     R5              ;Advance LDn number
49         001154 062702 000002        ADD     #2, R2          ;Advance logical disk index number
50         001160 020227 000000C        CMP     R2, #<2*MAXLD>   ;More LD's to check?
51         001164 103736              BLO     1$              ;Br if yes
52         ;
53         ; Finished
54         ;
55         001166 012605              MOV     (SP)+, R5
56         001170 012602              MOV     (SP)+, R2
57         001172 000207              RETURN

```

```

1          .SBTTL LDMNT -- Set up information about LD being mounted
2          ;-----
3          ; LDMNT is called to set up information about a logical disk.
4          ;
5          ; Inputs:
6          ; R2 = LD unit index
7          ; R3 = Channel number to use for lookup
8          ; CHNADR = Pointer to channel block for channel used for lookup
9          ; EMTBLK (byte 0) = Free channel to use for looking up file.
10         ; LDNAME(unit) = Name of file associated with logical disk.
11         ;
12         ; Outputs:
13         ; LDPDEV(unit) = Physical device index # and unit #
14         ; LDSIZE(unit) = Size of file
15         ; LDBASE(unit) = Base block on physical disk of start of logical disk.
16         ; Carry-flag is set on return if file cannot be found.
17         ;
18 001174 010446 LDMNT: MOV R4,-(SP)
19 001176 010546      MOV R5,-(SP)
20         ;
21         ; Remove any entry for this logical disk from access control table
22         ;
23 001200 004737 002060' CALL DLLDAC ; Remove LD entry from access control table
24         ;
25         ; Do lookup on file
26         ;
27 001204 010205      MOV R2,R5 ; Get LD unit index
28 001206 072527 000002  ASH #2,R5 ; 4 words per entry
29 001212 062705 000000G ADD #LDNAME,R5 ; Point to file spec in LDNAME table
30 001216 005715      TST (R5) ; Is there a file spec for this LD?
31 001220 001474      BEQ 9$ ; Br if not
32 001222 113746 000000G  MOVB SERFL0,-(SP) ; Save error control byte
33 001226 112737 000001 000000G  MOVB #1,SERFL0 ; Suppress error aborts
34 001234      . LOOKUP #CSIARE,R3,R5 ; Lookup the file
35 001254 112637 000000G  MOVB (SP)+,SERFL0 ; Restore error control byte
36 001260 103454      BCS 9$ ; Br if could not find the file
37 001262 010062 000000G  MOV R0,LDSIZE(R2) ; Save the size of the file
38         ;
39         ; Set up information about the file
40         ;
41 001266 013705 000000G  MOV CHNADR,R5 ; Get pointer to channel block
42 001272 016500 000000G  MOV C.CSW(R5),R0 ; Get channel status word
43 001276 042700 177701  BIC #^C<76>,R0 ; Extract device index number
44 001302 110062 000000G  MOVB R0,LDPDEV(R2) ; Save physical device index number
45 001306 116504 000000G  MOVB C.DEVQ(R5),R4 ; Get physical unit number
46 001312 042704 177770  BIC #^C<7>,R4 ; Extract unit number
47 001316 110462 000001G  MOVB R4,LDPDEV+1(R2) ; Save unit #
48 001322 016562 000000G 000000G  MOV C.SBLK(R5),LDBASE(R2) ; Get base block # of file on disk
49         ;
50         ; If we have read-only access to the file associated with the
51         ; logical disk, set no-write flag for this LD entry.
52         ;
53 001330 032765 000000G 000000G  BIT #CS#RON,C.CSW(R5); Do we have read-only access to the file?
54 001336 001403      BEQ 2$ ; Br if not
55 001340 052762 000000G 000000G  BIS #LD#RON,LDFLAG(R2); Set no-write flag for this LD
56         ;
57         ; See if the logical disk file is itself within a logical disk

```

```

58          ; (i.e., this is a nested logical disk)
59          ;
60 001346 020037 000000G 2$:      CMP      R0,LDDEVX      ;Is file on a logical disk?
61 001352 001007          BNE      1$      ;Br if not
62 001354 006304          ASL      R4      ;Cvt unit # to logical disk table index
63 001356 016462 000000G 000000G      MOV      LDPDEV(R4),LDPDEV(R2) ;Get real physical device & unit #
64 001364 066462 000000G 000000G      ADD      LDBASE(R4),LDBASE(R2) ;Bias base block # by log disk base
65          ;
66          ; Purge the channel we used to open the file
67          ;
68 001372          1$:      .PURGE  R3      ;Purge channel we opened to the file
69          ;
70          ; Make entry for the LD in the access control table
71          ;
72 001402 004737 001704'          CALL      ADLDAC      ;Make entry in access control table
73          ;
74          ; Success
75          ;
76 001406 000241          CLC          ;Signal success on return
77 001410 000403          BR      10$
78          ;
79          ; Error -- Could not find the file
80          ;
81 001412 005062 000000G 9$:      CLR      LDPDEV(R2)      ;Say logical disk is not active
82 001416 000261          SEC          ;Signal error on return
83 001420 012605 10$:      MOV      (SP)+,R5
84 001422 012604          MOV      (SP)+,R4
85 001424 000207          RETURN

```

ASNSRC -- Search assign table for logical name

```

1          .SBTTL  ASNSRC -- Search assign table for logical name
2          ;-----
3          ; ASNSRC is called to search the assign table for an entry
4          ; with a specified logical name.
5          ;
6          ; Inputs:
7          ; R0 = Logical name to search for.
8          ;
9          ; Outputs:
10         ; C-flag set on return if no assign block found with matching name.
11         ; R2 = Address of assign block if one found.
12         ;
13 001426 010146 ASNSRC: MOV     R1,-(SP)
14 001430 012701      MOV     #MAXASN,R1      ;Get # assign entries
15 001434 012702      MOV     #ASNTBL,R2      ;Point to assign table
16 001440 020062      1$:    CMP     R0,AT$LOG(R2) ;Compare logical names
17 001444 001405      BEQ     2$              ;Br if we found entry we are looking for
18 001446 062702      ADD     #AT$$SZ,R2      ;Point to next assign block
19 001452 077106      SOB     R1,1$           ;Loop if more blocks to check
20 001454 000261      SEC                      ;Signal failure
21 001456 000401      BR      3$
22 001460 000241      2$:    CLC                      ;Signal success
23 001462 012601      3$:    MOV     (SP)+,R1
24 001464 000207      RETURN

```

```

1          .SBTTL  LOGASN -- Perform full logical device assignment
2          ;-----
3          ; LOGASN is called to perform a full logical device name assignment.
4          ; The logical name associated with a file specification is translated
5          ; into the corresponding physical device. The file name, extension,
6          ; and size may also be translated if a file spec was specified with
7          ; the assignment of the logical name.
8          ;
9          ; Inputs:
10         ; R5 = Pointer to 5 word block containing file spec (dev,file,file,ext,size)
11         ;
12         ; Outputs:
13         ; File spec is updated to have physical device name and possibly
14         ; altered file name.
15         ;
16 001466 010246 LOGASN: MOV     R2,-(SP)
17 001470 010446      MOV     R4,-(SP)
18         ;
19         ; See if device name is in our assign table
20         ;
21 001472 011500      MOV     (R5),R0      ;Get logical device name
22 001474 004737 001426' CALL    ASNSRC      ;See if name is in assign table
23 001500 103421      BCS     1$          ;Br if name is not in assign table
24         ;
25         ; Found logical device name in the assign table.
26         ; Translate to physical device.
27         ;
28 001502 010504      MOV     R5,R4          ;Get pointer to file spec buffer
29 001504 016224 000000G MOV     AT$DEV(R2),(R4)+;Put in physical device name
30 001510 016200 000000G MOV     AT$FIL(R2),R0      ;Was file name assigned?
31 001514 001413      BEQ     1$          ;Br if not
32 001516 010024      MOV     R0,(R4)+      ;Translate file name
33 001520 016224 000002G MOV     AT$FIL+2(R2),(R4)+
34 001524 016200 000000G MOV     AT$EXT(R2),R0      ;Was file extension specified?
35 001530 001405      BEQ     1$          ;Br if not
36 001532 010024      MOV     R0,(R4)+      ;Translate file extension
37 001534 016200 000000G MOV     AT$SIZ(R2),R0      ;Was file size specified?
38 001540 001401      BEQ     1$          ;Br if not
39 001542 010014      MOV     R0,(R4)        ;Translate file size
40         ;
41         ; Translate "DK" and "SY" to physical device names
42         ;
43 001544 021537 000012' 1$:      CMP     (R5),R5OSY      ;Is device name "SY"?
44 001550 001403      BEQ     2$          ;Br if yes
45 001552 021537 000010'      CMP     (R5),R5ODK      ;Is device name "DK"?
46 001556 001002      BNE     3$          ;Br if not
47 001560 013715 000000G 2$:      MOV     SYNAME,(R5)      ;Translate to physical device
48         ;
49         ; Finished
50         ;
51 001564 012604 3$:      MOV     (SP)+,R4
52 001566 012602      MOV     (SP)+,R2
53 001570 000207      RETURN

```

DEADEV -- Deassign physical device

```

1
2      .SBTTL  DEADEV -- Deassign physical device
3      -----
4      ; DEADEV is called to remove from the assign table all entries
5      ; for logical device names that are assigned to a specified
6      ; physical device.
7      ;
8      ; Inputs:
9      ;   R0 = Name of physical device.
10     DEADEV: MOV     R2, -(SP)
11             MOV     R3, -(SP)
12             MOV     #ASNTBL, R2      ; Point to assign table
13             MOV     #MAXASN, R3      ; Get # assign table entries
14 1$:  CMP     R0, AT$DEV(R2)          ; Is this entry for specified phys device?
15             BNE     2$              ; Br if not
16             CLR     AT$LOG(R2)       ; Clear logical device name
17             CLR     AT$DEV(R2)       ; Clear physical device name
18 2$:  ADD     #AT$$SZ, R2              ; Point to next assign entry
19             SOB     R3, 1$           ; Loop if more to check
20             MOV     (SP)+, R3
21             MOV     (SP)+, R2
22             RETURN

```

```

1          .SBTTL  LDDEVN -- See if device name is a LD unit
2          ;-----
3          ; LDDEVN is called to determine if a device name is of the form LDn and
4          ; if so, to determine the unit number.
5          ;
6          ; Inputs:
7          ;   RO = Device name (rad50)
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Device is LDn
11         ;   C-flag set    ==> Device is not LDn
12         ;   RO = LD device index number (2*n)
13         ;
14 001640 020037 000006' LDDEVN: CMP      RO,R5OLD      ; Is name "LD"?
15 001644 001002          BNE      1$              ; Br if not
16 001646 013700 000002'          MOV      R5OLD0,RO    ; Replace "LD" by "LDO"
17 001652 020037 000002' 1$:      CMP      RO,R5OLD0    ; Is name in the range LDO to LD7?
18 001656 103410          BLO      2$              ; Br if not
19 001660 020037 000004'          CMP      RO,R5OLD7
20 001664 101005          BHI      2$
21 001666 163700 000002'          SUB      R5OLD0,RO    ; Get unit number only
22 001672 006300          ASL      RO                ; Convert to device index number
23 001674 000241          CLC                      ; Signal success on return
24 001676 000401          BR       3$
25 001700 000261          2$:      SEC                      ; Signal failure on return
26 001702 000207          3$:      RETURN

```

```

1          .SBTTL  ADLDAC -- Add LD entry to access control table
2          -----
3          ; ADLDAC is called to add a logical disk entry to the device/file
4          ; access control table.  If there are no protected devices or files
5          ; no entry is made.
6          ;
7          ; Inputs:
8          ;   R2 = Logical disk index number (2 * unit number)
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> all ok
12         ;   C-flag set    ==> Access table overflow
13         ;
14 001704 010246 ADLDAC: MOV      R2, -(SP)
15 001706 010546      MOV      R5, -(SP)
16         ;
17         ; See if there are any entries in the access table
18         ;
19 001710 005737 000000G      TST      RESDEV      ;Any entries in access table?
20 001714 001446      BEQ      4$                ;Br if not
21         ;
22         ; There are entries in the access control table
23         ;
24 001716 006202      ASR      R2                ;Convert ld index # to unit #
25         ;
26         ; Find a free entry in the access table
27         ;
28 001720 012705 000000G      MOV      #OKFILE, R5      ;Point to start of access table
29 001724 020537 000000G 1$:  CMP      R5, OKFNND      ;Reached end of table?
30 001730 103036      BHIS     3$                ;Br if yes -- table overflow
31 001732 005765 000000G      TST      OF$FIL(R5)      ;Is this entry free?
32 001736 001403      BEQ      2$                ;Br if free
33 001740 062705 000000G      ADD      #OF$$SZ, R5      ;Point to next entry
34 001744 000767      BR       1$                ;Go check it
35         ;
36         ; We found a free entry.  Add entry for LD.
37         ;
38 001746 113765 000000G 000000G 2$:  MOVB     LDDEVX, OF$DEV(R5); Set logical disk device index number
39 001754 110265 000000G      MOVB     R2, OF$UNT(R5)  ;Set logical disk unit #
40 001760 012700 000000G      MOV      #WLDNAM, R0     ;Set file name to wildcards
41 001764 010065 000000G      MOV      R0, OF$FIL(R5)
42 001770 010065 000002G      MOV      R0, OF$FIL+2(R5)
43 001774 010065 000004G      MOV      R0, OF$FIL+4(R5)
44 002000 105065 000000G      CLRB     OF$FLG(R5)      ;Initially clear all control flags
45 002004 006302      ASL      R2                ;Cvt unit # to LD index #
46 002006 032762 000000G 000000G      BIT      #LD$RON, LDFLAG(R2); Is logical disk write protected?
47 002014 001406      BEQ      4$                ;Br if not
48 002016 152765 000000G 000000G      BISB     #OT$RON, OF$FLG(R5); Set read-only flag in access table
49 002024 000402      BR       4$
50         ;
51         ; Error -- Access table overflow
52         ;
53 002026 000261 3$:  SEC                      ;Signal error on return
54 002030 000410      BR       10$
55         ;
56         ; Success! If this entry extends the ACCESS table, bump up the
57         ; table end pointer to include it.

```

ADLDAC -- Add LD entry to access control table

```
58
59 002032 062705 000000G      4$:    ADD    #OF$$SZ,R5      ;Point past this LD ACCESS entry
60 002036 020537 000000G      ;    CMP    R5,OKFAND      ;Do we need to increase last ACCESS ptr?
61 002042 101402                ;    BLOS   41$          ;Br if not
62 002044 010537 000000G      ;    MOV    R5,OKFAND      ;Extend ACCESS table to contain entry
63 002050 000241      41$:    CLC          ;Signal success on return
64
65      ; Finished
66
67 002052 012605      10$:    MOV    (SP)+,R5
68 002054 012602      ;    MOV    (SP)+,R2
69 002056 000207      ;    RETURN
```

DLLDAC -- Delete LD entry from access control table

```

1          .SBTTL  DLLDAC -- Delete LD entry from access control table
2          ;-----
3          ; DLLDAC is called to delete any entry in the access control table
4          ; for a specified logical disk.
5          ;
6          ; Inputs:
7          ; R2 = Logical disk index number (2 * unit number)
8          ;
9 002060 004737 002124' DLLDAC: CALL    CKLDAC          ;Is there an entry for this LD in table?
10 002064 103416          BCS      1$          ;Br if not
11          ;
12          ; We have found an entry in the access table for this LD.
13          ; R0 = Pointer to the entry.
14          ;
15 002066 105060 000000G      CLRB    OF$DEV(R0)      ;Mark entry as free
16 002072 105060 000000G      CLRB    OF$UNT(R0)
17 002076 005060 000000G      CLR     OF$FIL(R0)
18          ;
19          ; If the ACCESS table was extended to include this entry,
20          ; reduce the table end ptr below it.
21          ;
22 002102 062700 000000G      ADD     #OF$$SZ,R0      ;Point past this entry
23 002106 020037 000000G      CMP     R0,OKFAND      ;Was it the last entry?
24 002112 103403          BLO      1$          ;Br if not
25 002114 162737 000000G 000000G      SUB     #OF$$SZ,OKFAND ;It was, lower the end below it
26          ;
27          ; Finished
28          ;
29 002122 000207          1$:      RETURN

```

```

1          .SBTTL  CKLDAC -- Check if LD is in access control table
2          -----
3          ; CKLDAC is called to determine if a certain logical disk is in the
4          ; device/file access control table.
5          ;
6          ; Inputs:
7          ;   R2 = Logical disk index number (2 * unit #)
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Found logical disk entry in access control table.
11         ;   C-flag set    ==> Logical disk entry is not in access table.
12         ;   R0 = Pointer to access control entry.
13         ;
14 002124 010246 CKLDAC: MOV     R2,-(SP)      ;save original LD index number
15 002126 006202      ASR     R2              ;Convert index # to unit #
16 002130 005737 000000G      TST     RESDEV    ;Are there any entries in access table?
17 002134 001417      BEQ     4$              ;Br if not
18
19         ; There are entries in the access control table.
20         ; Search for entry matching our logical disk.
21         ;
22 002136 012700 000000G      MOV     #OKFILE,R0    ;Point to start of access table
23 002142 020037 000000G 1$:  CMP     R0,OKFAND    ;Checked all entries?
24 002146 103012      BHIS    4$              ;Br if so
25 002150 126037 000000G 000000G  CMPB   OF$DEV(R0),LDDEVX ;Is this entry for a logical disk?
26 002156 001003      BNE     2$              ;Br if not
27 002160 120260 000000G  CMPB   R2,OF$UNT(R0)    ;Is entry for specified unit?
28 002164 001405      BEQ     3$              ;Br if yes -- found entry
29 002166 062700 000000G 2$:  ADD     #OF$$SZ,R0    ;Point to next access entry
30 002172 000763      BR      1$              ;Check all entries
31
32         ; LD entry is not in access table
33         ;
34 002174 000261 4$:  SEC          ;Signal failure on return
35 002176 000401      BR      9$
36
37         ; Found LD entry in table
38         ;
39 002200 000241 3$:  CLC          ;Signal success on return
40 002202 012602 9$:  MOV     (SP)+,R2    ;Recover LD index number
41 002204 000207      RETURN

```

LDOFCK -- See if any files are open on a logical disk

```

1          .SBTTL  LDOFCK -- See if any files are open on a logical disk
2          ;-----
3          ; LDOFCK is called to determine if the current job has any files
4          ; open on a specified logical disk.
5          ;
6          ; Inputs:
7          ;   R2 = LD unit index
8          ;
9          ; Outputs:
10         ;   C-flag set if any files are open on the LD.
11         ;
12 002206 010146 LDOFCK: MOV     R1,-(SP)
13 002210 010346      MOV     R3,-(SP)
14 002212 010446      MOV     R4,-(SP)
15 002214 010546      MOV     R5,-(SP)
16         ;
17         ; Begin loop to check each channel
18         ;
19 002216 016204 000000G      MOV     LDBASE(R2),R4 ;Get base block # of LD
20 002222 066204 000000G      ADD     LDSIZE(R2),R4 ;Get block # above top of LD
21 002226 005003              CLR     R3           ;Start with channel 0
22         ;
23         ; Compute address of channel block for this channel
24         ;
25 002230 010305 1$:      MOV     R3,R5           ;Get channel number
26 002232 013700          MOV     CXTRMN,R0       ;Get address of simulated rmon
27 002236 062700          ADD     #R$CHN,R0       ;Point to 1st channel area
28 002242 020527 000021    CMP     R5,#17.        ;Is this channel in extended channel area?
29 002246 103404          BLO     2$             ;Br if not
30 002250 162705 000021    SUB     #17.,R5        ;Get channel # in extended channel area
31 002254 062700          ADD     #R$XCHN-R$CHN,R0 ;Point to extended channel area
32 002260 070527 000000G  2$:      MUL     #CHNSIZ,R5 ;Multiply by # bytes per channel
33 002264 060005          ADD     R0,R5           ;Get absolute address of channel block
34         ;
35         ; See if this channel is open
36         ;
37 002266 016500 000000G      MOV     C.CSW(R5),R0 ;Get channel status word
38 002272 032700 000000G      BIT     #CS$OPN,R0   ;Is this channel open?
39 002276 001427          BEQ     3$             ;Br if not
40         ;
41         ; See if file is within LD
42         ;
43 002300 042700 177701      BIC     #^C<76>,R0    ;Extract device index number
44 002304 020037 000000G    CMP     R0,LDDEVX      ;Is file on a logical disk?
45 002310 001022          BNE     3$             ;Br if not
46 002312 116501 000000G    MOVB    C.DEVQ(R5),R1 ;Get unit number
47 002316 042701 177770      BIC     #^C<7>,R1    ;Extract unit #
48 002322 006301          ASL     R1              ;Convert unit # to index
49 002324 026162 000000G 000000G    CMP     LDPDEV(R1),LDPDEV(R2);Are LD's on same physical drive?
50 002332 001011          BNE     3$             ;Br if not
51 002334 016500 000000G      MOV     C.SBLK(R5),R0 ;Get base block of file
52 002340 066100 000000G      ADD     LDBASE(R1),R0 ;Add base block of LD file is on
53 002344 020062 000000G    CMP     R0,LDBASE(R2) ;Is it below base of LD?
54 002350 103402          BLO     3$             ;Br if yes
55 002352 020004          CMP     R0,R4          ;Is it above top of LD?
56 002354 103406          BLO     4$             ;Br if it is within LD
57         ;

```

```

58          ; Check next channel
59          ;
60 002356 005203      3$:      INC      R3          ; Advance channel #
61 002360 020327 000000G      CMP      R3, #NLCHN      ; Checked all channels?
62 002364 103721      BLO      1$          ; Br if more to check
63          ;
64          ; No file is open on the LD
65          ;
66 002366 000241      CLC          ; Signal OK on return
67 002370 000401      BR      9$
68          ;
69          ; There is a file open on the LD
70          ;
71 002372 000261      4$:      SEC          ; Signal error on return
72          ;
73          ; Finished
74          ;
75 002374 012605      9$:      MOV      (SP)+, R5
76 002376 012604      MOV      (SP)+, R4
77 002400 012603      MOV      (SP)+, R3
78 002402 012601      MOV      (SP)+, R1
79 002404 000207      RETURN
80
81          . END

```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 9621 Words (38 Pages)
 Size of core pool: 18176 Words (71 Pages)
 Operating system: RT-11

Elapsed time: 00:00:23.01
 ,LP: TSEM5=DK: TSEM5/C/N: SYM

... V1	3-109	9-34	9-34	9-34	9-68						
... V2	3-109	3-109	3-109#	3-109#	9-34	9-34	9-34#	9-34#	9-68	9-68#	
ADLDAC	9-72	14-14#									
ASNSRC	10-13#	11-22									
ASNTBL	1-38	10-15	12-12								
AT\$SZ	1-39	10-18	12-18								
AT\$DEV	1-39	11-29	12-14	12-17*							
AT\$EXT	1-39	11-34									
AT\$FIL	1-39	11-30	11-33								
AT\$LOG	1-38	10-16	12-16*								
AT\$SIZ	1-39	11-37									
C. CSW	1-36	3-15	9-42	9-53	17-37						
C. DEVQ	1-35	9-45	17-46								
C. SBLK	1-35	9-48	17-51								
CHNADR	1-36	3-14	9-41								
CHNSIZ	1-34	17-32									
CKLDAC	15-9	16-14#									
CS\$OPN	1-36	3-15	17-38								
CS\$RON	1-35	9-53									
CSIARE	1-37	1-54	1-59	3-109	3-116*	7-25*	8-27*	8-42*	9-34		
CSIFIL	1-37	3-50	3-59	3-61	3-63	3-70	3-83	3-109			
CXTRMN	1-34	17-26									
DEADEV	12-10#										
DIEARG	2-18										
DIEMSG	2-18										
DLLDAC	9-23	15-9#									
DMTARG	1-58#	7-28	8-30								
EMTBLK	1-36	3-22	3-47	3-91	3-98	3-108	3-114	3-123	4-12	5-15	5-23
EMTXIT	1-38	3-128	4-37	5-61	6-42						
LD\$RON	1-38	3-93	5-42	9-55	14-46						
LDAEMT	1-30	6-8#									
LDBASE	1-35	9-48*	9-64	9-64*	17-19	17-52	17-53				
LDCLN	3-124	8-10#									
LDDEMT	1-30	4-8#									
LDDEVN	3-71	13-14#									
LDDEVX	1-35	9-60	14-38	16-25	17-44						
LDDMT	4-30	6-28	7-13#								
LDFLAG	1-37	3-90*	3-93*	5-42	9-55*	14-46					
LDIEMT	1-30	5-11#									
LDMEMT	1-30	3-10#									
LDMNT	3-99	8-36	9-18#								
LDNAME	1-37	3-40	3-82	4-23	5-31	6-23	7-37*	8-22	9-29		
LDOFCK	7-17	17-12#									
LDPDEV	1-38	5-45	7-34*	8-40	9-44*	9-47*	9-63	9-63*	9-81*	17-49	17-49
LDSIZE	1-35	9-37*	17-20								
LOGASN	3-60	11-16#									
MAXASN	1-38	10-14	12-13								
MAXLD	1-36	3-23	4-13	5-16	8-50						
MNTARG	1-53#	3-117	8-43								
NLCHN	1-34	17-61									
OF\$SZ	1-40	14-33	14-59	15-22	15-25	16-29					
OF\$DEV	1-40	14-38*	15-15*	16-25							
OF\$FIL	1-40	14-31	14-41*	14-42*	14-43*	15-17*					
OF\$FLG	1-41	14-44*	14-48*								
OF\$UNT	1-41	14-39*	15-16*	16-27							
OKFAND	1-40	14-60	14-62*	15-23	15-25*	16-23					

[illegible]

...CM1	3-109	9-34		
...CM2	3-109	3-109	9-34	9-34
...CM3	9-68			
...CM5	3-109	9-34		
.FPROT	1-26#	3-109		
.LOOKU	1-26#	9-34		
.PURGE	1-26#	9-68		
DIE	2-19#			
DISABL	2-4#			
ENABL	2-8#			
OCALL	2-31#			