

Table of contents

6-	1	DEFINE command
7-	1	DEFINE/KEY
8-	1	KEYSOP -- Set key definition option flags
8-	8	KEYROP -- Reset key definition option flags
8-	15	KEYTYP -- Set key definition type
9-	1	KEYADD -- Add a new key definition
10-	1	KEYDEL -- Delete a key definition
11-	1	KEYTXT -- Accrue key definition string
12-	1	KEYSRC -- Search for key definition
13-	1	KEYREG -- Create PLAS region for key definitions
14-	1	KEYMAP -- Map PAR 1 to key definition region
15-	1	KEYUMP -- Unmap the key definition region
16-	1	KEYELR -- Eliminate the key definition region
17-	1	SHOW KEYS
18-	1	KEYPRT -- Print key definition
19-	1	KEYCOD -- Print key name based on code
20-	1	INIUKD -- Init user key definitions from parent job
21-	1	KEYMOV -- Move key definition data from parent job
22-	1	RECALL command
29-	1	ACCESS Command
30-	1	ACBDxx -- [NO]ACCESS error handlers
31-	1	NOACCESS Command
32-	1	NO[ACCESS] Command subroutines
32-	12	Check for privilege to issue [NO]ACCESS
32-	30	Init a [NO]ACCESS table entry
32-	46	Add entry to [NO]ACCESS tables
32-	114	Handle [NO]ACCESS switches
33-	1	Convert RAD50 dev name to index and unit
34-	1	GETSYP -- Accept system logon password

```

1           .TITLE  TSKM2B -- Keyboard DEFINE Command routines
2           .ENABL  LC
3           .DSABL  GBL
4 000000    .CSECT  TSKM2B
5 000000
6           TSKM2B:
7           ;
8           ; TSKM2B is the portion of TSKMON that contains the code
9           ; to implement the DEFINE command.
10          ;
11          ; Copyright 1985.
12          ; S&H Computer Systems, Inc.
13          ; Nashville, Tennessee
14          ;
15          ; Macro calls
16          ;
17          .MCALL .ELRG, .ELAW, .CRRG, .CRAW, .TTYOUT, .PRINT
18          .MCALL .LOOKUP, .READW, .CLOSE, .TTYIN
19          ;
20          ; Global definitions
21          ;
22          .GLOBL  CMDDEF, SHOKEY, INIUKD, GETSYP, CMDRCL, CMDACC, CMDNAC
23          ;
24          ; Global references
25          ;
26          .GLOBL  OKFAND, OKFNND, TBLOVF, R50BUF, RESDEV, BADACC, PRTWRN
27          .GLOBL  SPACE, CORUSR, P0$SYS, GTRD50, ASNSRC, AT$DEV, CHKDLM
28          .GLOBL  R50SY, R50DK, SYINDX, SYUNIT, NUMDEV, PNAME, PRIVCO
29          .GLOBL  OF$DEV, OF$UNT, OF$FIL, OF$FLG, OF$$SZ, OT$RON
30          .GLOBL  $DOOFF, LSW, EM$SLO, LSW7, $SLO, SLSPTR, SLLPTR, SLRPTR
31          .GLOBL  EM$CSE, PRTFIX, SPACE2, SLEND, SLLBUF, INVOPT
32          .GLOBL  P2$VIR, FIXPRV, $SCCA, LSW5, PRIVA2, PRIVC2, $PRGLK
33          .GLOBL  LSW2, CVTUC, VONTM, ACRTXT, RCLREV, $KINIT
34          .GLOBL  SYPSWD, SYSPSR, $ECHO, $NOIN, LSW3, $SUCF, LSW9
35          .GLOBL  TM$KND, TM$GLD, CRLF, R. GSIZ, W. NSIZ, W. NLEN, BLKO
36          .GLOBL  RSFBLK, ERRLOC, EM$KNS, ACRDEC, EM$NSF
37          .GLOBL  VSLEDT, EM$NSL, RS. NEW, RS. EGR, KEYRCB, PRTSPC, KD$TYP
38          .GLOBL  INVOPT, ILLCMD, EM$KCR, EM$KWC, WS. CRW, EM$KNT
39          .GLOBL  BLKO, ACRSTR, KEYMXT, EM$STL, W. NRID, W. NSTS, WS. MAP
40          .GLOBL  R. GID, R. GSTS, RS. CRR, RS. GBL, RS. CGR, RS. PVT, XAREA, VPAR1
41          .GLOBL  KF$ECO, KF$TRM, KT$GLD, CVTTAB, SKPSPC, SEARCH, AMBOPT, FKILL
42          .GLOBL  KT$NRM, OPTLST, KT$LET, KD$COD, KD$FLG, KD$TXT, RDCMD, KEYPAR
43          .GLOBL  VKEYMX, KD$$SZ, EM$KTF, KT$GLT, EM$KNU, KEYRCB, RC$BAS
44          .GLOBL  KC$KP0, KC$KP1, KC$KP2, KC$KP3, KC$KP4, KC$KP5, KC$KP6
45          .GLOBL  KC$KP7, KC$KP8, KC$KP9, KC$DOT, KC$COM, KC$MIN, KC$ENT
46          .GLOBL  KC$UP, KC$DWN, KC$LFT, KC$RIT, KC$E1, KC$E2, KC$E3, KC$E4
47          .GLOBL  KC$E5, KC$E6, KC$F6, KC$F7, KC$F8, KC$F9, KC$F10, KC$F11
48          .GLOBL  KC$F12, KC$F13, KC$F14, KC$F15, KC$F16, KC$F17, KC$F18
49          .GLOBL  KC$F19, KC$F20, KC$PF1, KC$PF2, KC$PF3, KC$PF4
50          ;
51          ; Assembly constants
52          ;
53          LF      =      12      ; LINE FEED
54          CR      =      15      ; CARRIAGE RETURN
55          BLANK   =      40      ; ASCII SPACE
56          BELL    =      07      ; ASCII BELL
57          TAB     =      11      ; HORIZONTAL TAB
          FF      =      14      ; FORM FEED

```

58	000054	COMMA	=	54	; COMMA
59	000400	BLKWDS	=	256.	; # OF WORDS IN DISK BLOCK
60	132500	WLDNAM	=	132500	; RAD50 /*/ (WILDCARD)
61	000035	R50AST	=	35	; RAD50 / /*/ (WILDCARD UNIT #)

```

1          ; -----
2          ; Data areas
3          ;
4 000000 000000 KFLAG: .WORD 0 ;Flags set during parsing
5 000002 000000 KCODE: .WORD 0 ;Type and code value
6          000003' KTYPE = KCODE+1 ;Key type code is in upper byte of KCODE
7 000004 PSFNAM: .BLKW 4 ;Local copy of PLAS region swap file name
8 000014 000000 RCLPTR: .WORD 0 ;Ptr to save area for RECALL command
9          ;
10         ; Region definition block for key definition region
11         ;
12 000016 000000 KRDB: .WORD 0 ;Will get addr of region control blk
13 000020 000000 .WORD 0 ;# 64-byte blocks needed for region
14 000022 000000C .WORD RS.GBL!RS.CGR!RS.PVT ;Status flags
15 000024 042641 014716 .RAD50 /KEYDEF/ ;Region name
16         ;
17         ; Window definition block used to map PAR 1 to key definition region
18         ;
19 000030 000 KWDB: .BYTE 0 ;Will get window ID
20 000031 001 .BYTE 1 ;Select PAR 1
21 000032 000000 .WORD 0 ;Will get base virtual address
22 000034 000000 .WORD 0 ;# 64-byte blocks for window
23 000036 000000 .WORD 0 ;Addr of region control block
24 000040 000000 .WORD 0 ;Offset into region of window base
25 000042 000000 .WORD 0 ;# 64-byte blocks to map
26 000044 000000G .WORD WS.MAP ;Status flags
27         ;
28         ; Emt arg block to copy key definition data from our parent job
29         ;
30 000046 000 126 EMTUKC: .BYTE 0,126
31 000050 000017 .WORD 17
32 000052 000000 .WORD 0 ;Parent job number
33 000054 000000 .WORD 0 ;Page number of data
34 000056 000000G .WORD BLK0 ;Address of destination buffer
35         ;
36         ; Emt arg block to set a TT read timeout
37         ;
38 000060 000 117 RDTIME: .BYTE 0,117
39 000062 000000 .WORD 0 ;Timeout value (0.5 second units)
40 000064 000001 .WORD 1 ;Timeout signal character
41         ;
42         ; Emt arg block to log off and drop DTR after short time
43         ;
44 000066 000 126 HNGEMT: .BYTE 0,126
45 000070 000002 .WORD 2
46 000072 000001 .WORD 1 ;Time before DTR dropped
47         ;
48         ; Byte data
49         ;
50 000074 072 040 200 COLSPC: .ASCIZ /: /<200>
51 000077 000
52 000100 KTEXT: .BLKB 80. ;Key definition text string
          .EVEN

```

```

1      ; -----
2      ; Macro to cause a fatal error message to be printed.
3      ;
4      .MACRO FERR MSG
5      MOV R5, -(SP)
6      MOV MSG, R5
7      CALL FPRINT
8      MOV (SP)+, R5
9      .ENDM FERR
10     ;
11     ; -----
12     ; Macro to print a fatal error message, clean up
13     ; and then jump to RDCMD.
14     ;
15     .MACRO FABORT MSG
16     MOV MSG, R5
17     JMP FKILL
18     .ENDM FABORT
19     ;
20     ; -----
21     ; Macro to print a warning message
22     ;
23     .MACRO FWARN MSG
24     MOV R5, -(SP)
25     MOV MSG, R5
26     CALL PRTWRN
27     MOV (SP)+, R5
28     .ENDM FWARN
29     ;
30     ; -----
31     ; Macro to start a standard option table.
32     ; Name = 1 to 4 character table name.
33     ; NA = Number of arguments per table entry.
34     ;
35     .MACRO TBLDEF NAME, NA
36     NARGS = NA
37     .CSECT CMDV2B
38     NAME 'HD: .WORD 2*NA
39     .ENDM TBLDEF
40     ;
41     ; -----
42     ; Macro to enter an option text name and a set of parameters
43     ; into the currently open table.
44     ; STRNG = Ascii name
45     ; A,B,C = Set of option parameters to store in table with name.
46     ;
47     .MACRO CMDDEF STRNG, A, B, C
48     .CSECT NAME2B
49     L =
50     .ASCIZ /STRNG/
51     .CSECT CMDV2B
52     .WORD L ; POINTER TO NAME STRING
53     .WORD A
54     .IIF GE, <NARGS-2> .WORD B
55     .IIF GE, <NARGS-3> .WORD C
56     .ENDM CMDDEF
57     ;

```

58
59
60
61
62
63
64
65

```
-----  
; Macro to end a set of table entries.  
;  
; .MACRO TBLEND  
; .CSECT CMDV2B  
; .WORD 0  
; .CSECT TSKM2B  
; .ENDM TBLEND
```

```
1 ;-----  
2 ; Define initial qualifiers for the DEFINE command.  
3 ;  
4 000220 TBLDEF DEF,1  
5 000002 CMDDEF KEY,DEFKEY  
6 000006 TBLEND  
7 ;-----  
8 ;  
9 ; Define qualifiers for the DEFINE/KEY command.  
10 ;  
11 000220 TBLDEF KEY,2  
12 000012 CMDDEF ECHO,KEYSOP,KF$ECO  
13 000020 CMDDEF NOE*CHO,KEYROP,KF$ECO  
14 000026 CMDDEF TER*MINATE,KEYSOP,KF$TRM  
15 000034 CMDDEF NOT*ERMINATE,KEYROP,KF$TRM  
16 000042 CMDDEF GO*LD,KEYTYP,KT$GLD  
17 000050 CMDDEF NOG*OLD,KEYTYP,KT$NRM  
18 000056 CMDDEF LET*TER,KEYTYP,KT$LET  
19 000064 TBLEND
```

```

1
2 ; -----
3 ; Define names of keys and associated code values.
4 TBLDEF KNM, 1
5 CMDDEF PF1, KC$PF1
6 CMDDEF PF2, KC$PF2
7 CMDDEF PF3, KC$PF3
8 CMDDEF PF4, KC$PF4
9 CMDDEF KP0, KC$KP0
10 CMDDEF KP1, KC$KP1
11 CMDDEF KP2, KC$KP2
12 CMDDEF KP3, KC$KP3
13 CMDDEF KP4, KC$KP4
14 CMDDEF KP5, KC$KP5
15 CMDDEF KP6, KC$KP6
16 CMDDEF KP7, KC$KP7
17 CMDDEF KP8, KC$KP8
18 CMDDEF KP9, KC$KP9
19 CMDDEF 0, KC$KP0
20 CMDDEF 1, KC$KP1
21 CMDDEF 2, KC$KP2
22 CMDDEF 3, KC$KP3
23 CMDDEF 4, KC$KP4
24 CMDDEF 5, KC$KP5
25 CMDDEF 6, KC$KP6
26 CMDDEF 7, KC$KP7
27 CMDDEF 8, KC$KP8
28 CMDDEF 9, KC$KP9
29 CMDDEF PER*IOD, KC$DOT
30 CMDDEF COM*MA, KC$COM
31 CMDDEF MIN*US, KC$MIN
32 CMDDEF ENT*ER, KC$ENT
33 CMDDEF LEFT, KC$LFT
34 CMDDEF RIGHT, KC$RIT
35 CMDDEF UP, KC$UP
36 CMDDEF DO, KC$F16
37 CMDDEF DOWN, KC$DWN
38 CMDDEF FIND, KC$E1
39 CMDDEF <INS*ERT-HERE>, KC$E2
40 CMDDEF REM*OVE, KC$E3
41 CMDDEF SEL*ECT, KC$E4
42 CMDDEF <PREV*-SCREEN>, KC$E5
43 CMDDEF <NEXT*-SCREEN>, KC$E6
44 CMDDEF E1, KC$E1
45 CMDDEF E2, KC$E2
46 CMDDEF E3, KC$E3
47 CMDDEF E4, KC$E4
48 CMDDEF E5, KC$E5
49 CMDDEF E6, KC$E6
50 CMDDEF HELP, KC$F15
51 CMDDEF F6, KC$F6
52 CMDDEF F7, KC$F7
53 CMDDEF F8, KC$F8
54 CMDDEF F9, KC$F9
55 CMDDEF F10, KC$F10
56 CMDDEF F11, KC$F11
57 CMDDEF F12, KC$F12

```


58 000414	CMDDEF	F13, KC#F13
59 000420	CMDDEF	F14, KC#F14
60 000424	CMDDEF	F15, KC#F15
61 000430	CMDDEF	F16, KC#F16
62 000434	CMDDEF	F17, KC#F17
63 000440	CMDDEF	F18, KC#F18
64 000444	CMDDEF	F19, KC#F19
65 000450	CMDDEF	F20, KC#F20
66 000454	TBLEND	

DEFINE command

```

1
2
3
4
5
6
7
8
9 000220 004767 000000G
10
11
12
13 000224 004767 000000G
14 000230 121327 000057
15 000234 001001
16 000236 005203
17
18
19
20 000240 012704 000000'
21 000244 004767 000000G
22 000250 103401
23 000252 000134
24
25
26
27 000254 005704
28 000256 001404
29 000260
30 000270

```

```

.SBTTL DEFINE command
-----
; Process the DEFINE command.
;
; Inputs:
; R1 = Job index number
; R3 = Pointer to start of qualifiers for DEFINE command.
;
CMDDEF: CALL CVTTAB ;Convert tab and FF chars to spaces
;
; If 1st character is slash, skip it
;
CALL SKSPC ;Skip over any spaces
CMPB (R3),# '/' ;Is 1st character slash?
BNE 1$ ;Br if not
INC R3 ;Point beyond slash
;
; Branch off to major processing routines based on first qualifier
;
1$: MOV #DEFHD,R4 ;Point to table of options
CALL SEARCH ;Try to find correct processing routine
BCS 2$ ;Br if don't recognize option keyword
JMP @(R4)+ ;Enter major processing routine
;
; Invalid keyword
;
2$: TST R4 ;Ambiguous or unrecognized option?
BEQ 3$ ;Br if unrecognized
FABORT #AMBOPT ;Ambiguous option
3$: FABORT #INVOPT ;Invalid option

```

```

1                                     .SBTTL  DEFINE/KEY
2                                     -----
3                                     ; Define a character string which will be substituted for some
4                                     ; terminal key.
5                                     ;
6                                     ; Command format:  DEFINE/KEY  key string
7                                     ;
8                                     ; Inputs:
9                                     ;   R3 = Points past "/KEY" option.
10                                    ;
11 000300 DEFKEY:
12                                    ;
13                                    ; Make sure Single line editor is genned into system
14                                    ;
15 000300 105767 0000000          TSTB    VSLEDT          ; Is SL available?
16 000304 001004                BNE     4$              ; Br if yes
17 000306                        FABORT  #EM$NSL          ; SL is not available
18                                    ;
19                                    ; See if any user-defined keys are allowed
20                                    ;
21 000316 005767 0000000 4$: TST     VKEYMX          ; Are any user-defined keys allowed?
22 000322 001004                BNE     5$              ; Br if yes
23 000324                        FABORT  #EM$KNS          ; User-defined keys not supported
24                                    ;
25                                    ; Initialize some values
26                                    ;
27 000334 105067 177442          5$: CLRB    KCODE          ; No key code yet
28 000340 112767 0000000 177435  MOVB   #KT$NRM,KTYPE ; Default to normal key type
29 000346 112767 0000000 177424  MOVB   #KF$TRM!KF$ECO,KFLAG ; Initialize flags
30                                    ;
31                                    ; Process any options
32                                    ;
33 000354 012704 000010'        MOV     #KEYHD,R4      ; Point to option list
34 000360 004767 0000000        CALL   OPTLST         ; Process the options
35                                    ;
36                                    ; R3 should now be pointing to the key name
37                                    ;
38 000364 105713                TSTB    (R3)           ; Was something specified?
39 000366 001456                BEQ     20$           ; Br if not
40                                    ;
41                                    ; If the /LETTER or /GOLDLETTER qualifiers were specified, the key
42                                    ; is a single letter.
43                                    ;
44 000370 126727 177407 0000000  CMPB   KTYPE,#KT$LET  ; Key type = letter?
45 000376 001404                BEQ     1$              ; Br if yes
46 000400 126727 177377 0000000  CMPB   KTYPE,#KT$GLT  ; Key type = goldletter?
47 000406 001031                BNE     3$              ; Br if not
48 000410 004767 0000000 1$: CALL   SKSPC          ; Skip up to start of string
49 000414 121327 000047        CMPB   (R3),#47        ; Is letter enclosed in quotes?
50 000420 001413                BEQ     6$              ; Br if yes
51 000422 121327 000042        CMPB   (R3),#42        ;
52 000426 001410                BEQ     6$              ; Br if yes
53 000430 112367 177346        MOVB   (R3)+,KCODE    ; Get the letter
54 000434 111300                MOVB   (R3),R0        ; Get following letter
55 000436 001424                BEQ     2$              ; Br if null
56 000440 120027 000040        CMPB   R0,#'         ; Blank?
57 000444 001421                BEQ     2$              ; Br if yes

```

DEFINE/KEY

```

58 000446 000432          BR      21$          ;Not single letter
59 000450 004767 000000G 6$:    CALL    ACRTXT        ;Accrue the letter
60 000454 020027 000001          CMP     RO,#1          ;Should have gotten exactly 1 char
61 000460 001025          BNE     21$          ;Br if not
62 000462 116767 000000G 177312  MOVB   BLKO,KCODE     ;Save the letter as key code
63 000470 000407          BR      2$
64
65          ; Convert key name into key code
66
67 000472 012704 000066' 3$:    MOV     #KNMHD,R4    ;Point to key name table
68 000476 004767 000000G          CALL    SEARCH        ;Try to translate key name
69 000502 103414          BCS     21$          ;Br if can't recognize key name
70 000504 111467 177272          MOVB   (R4),KCODE     ;Set code value for key
71
72          ; Accrue the associated text string and store into KTEXT
73
74 000510 004767 000346 2$:    CALL    KEYTXT        ;Accrue text string
75
76          ; If the associated text string is null, we are deleting the definition
77
78 000514 105767 177360          TSTB   KTEXT         ;Is string null?
79 000520 001523          BEQ     KEYDEL        ;Delete this key definition
80 000522 000443          BR     KEYADD        ;Add this key definition
81
82          ; Invalid command syntax
83
84 000524          20$:    FABORT #ILLCMD    ;Invalid command
85
86          ; Unrecognized key name
87
88 000534          21$:    FABORT #EM$KNU    ;Unrecognized key name

```

```

1          .SBTTL KEYSOP -- Set key definition option flags
2          ;-----
3          ; Set some key definition option flag.
4          ;
5 000544 151467 177230 KEYSOP: BISB (R4),KFLAG ;Set option flag
6 000550 000207          RETURN
7          ;
8          .SBTTL KEYROP -- Reset key definition option flags
9          ;-----
10         ; Reset (turn off) some key definition option.
11         ;
12 000552 141467 177222 KEYROP: BICB (R4),KFLAG ;Reset option flag
13 000556 000207          RETURN
14         ;
15         .SBTTL KEYTYP -- Set key definition type
16         ;-----
17         ; Set type of key being defined (Normal, Gold, etc.)
18         ;
19 000560 111400 KEYTYP: MOVB (R4),RO ;Get specified type
20         ;
21         ; Specially handle /GOLD with /LETTER
22         ;
23 000562 120027 000000G          CMPB RO,#KT$GLD ;Is type /GOLD?
24 000566 001005          BNE 1$ ;Br if not
25 000570 126727 177207 000000G  CMPB KTYPE,#KT$LET ;Is current type /LETTER?
26 000576 001012          BNE 2$ ;Br if not
27 000600 000407          BR 3$
28 000602 120027 000000G 1$: CMPB RO,#KT$LET ;Is type /LETTER?
29 000606 001006          BNE 2$ ;Br if not
30 000610 126727 177167 000000G  CMPB KTYPE,#KT$GLD ;Is current type /GOLD?
31 000616 001002          BNE 2$ ;Br if not
32 000620 112700 000000G 3$: MOVB #KT$GLT,RO ;Set type to gold-letter
33         ;
34         ; Set new key type
35         ;
36 000624 110067 177153 2$: MOVB RO,KTYPE ;Set key type
37         ;
38         ; Finished
39         ;
40 000630 000207          RETURN

```

KEYADD -- Add a new key definition

```

1          .SBTTL  KEYADD -- Add a new key definition
2          ;-----
3          ; Add a new key definition.
4          ;
5          ; Inputs:
6          ;   KCODE = Key character code (KC$xxx).
7          ;   KTYPE = Key type code (KT$xxx).
8          ;   KTEXT = Asciz string to be defined for key.
9          ;
10         000632 KEYADD:
11         ;
12         ; Don't allow /NOECHO to be specified with /NOTERMINATE
13         ;
14         000632 032767 000000C 177140          BIT    #KF$ECC!KF$TRM,KFLAG ;Either ECHO or TERMINATE specified?
15         000640 001004                          BNE    5$          ;Br if yes
16         000642                          FABORT #EM$KNT          ;Can't have both NOECHO and NOTERMINATE
17         ;
18         ; Create a PLAS region for key definitions if we don't already have one
19         ;
20         000652 005767 000000G          5$:    TST    KEYRCB          ;Do we have a key region?
21         000656 001003                          BNE    1$          ;Br if yes
22         000660 004767 000374          CALL   KEYREG          ;Create PLAS region and map to it
23         000664 000402                          BR     4$
24         ;
25         ; Set up PAR 1 to map to the key region
26         ;
27         000666 004767 000366          1$:    CALL   KEYREG          ;Map a par to the key region
28         ;
29         ; Try to find an existing definition for this key
30         ;
31         000672 016700 177104          4$:    MOV    KCODE,R0          ;Get key code
32         000676 004767 000310          CALL   KEYSRC          ;Try to find existing definition
33         000702 103012                          BCC   2$          ;Br if found existing entry
34         ;
35         ; Try to find a free entry
36         ;
37         000704 005000                          CLR    R0          ;Look for free entry
38         000706 004767 000300          CALL   KEYSRC
39         000712 103006                          BCC   2$          ;Br if found free entry
40         ;
41         ; Error -- No free entries
42         ;
43         000714 004767 000672          CALL   KEYUMP          ;Unmap par
44         000720                          FABORT #EM$KTF          ;Key table full
45         ;
46         ; The entry to use is pointed to by R2.
47         ; Make the entry.
48         ;
49         000730 016762 177046 000000G 2$:    MOV    KCODE,KD$COD(R2);Save key code
50         000736 116762 177036 000000G      MOVB   KFLAG,KD$FLG(R2);Save option flags
51         000744 062702 000000G          ADD    #KD$TXT,R2          ;Point to area for text string
52         000750 012703 000100'          MOV    #KTEXT,R3
53         000754 112322          3$:    MOVB   (R3)+,(R2)+          ;Store the string
54         000756 001376                          BNE   3$
55         ;
56         ; Finished
57         ;

```

KEYADD -- Add a new key definition

58 000760 004767 000626
59 000764 000167 000000G

CALL KEYUMP
JMP RDCMD

;Unmap par

KEYDEL -- Delete a key definition

```

1          .SBTTL  KEYDEL -- Delete a key definition
2          ;-----
3          ; Delete a key definition.
4          ;
5 000770   KEYDEL:
6          ;
7          ; See if there is any key region now.
8          ;
9 000770   005767   000000G   TST    KEYRCB      ;Any key definitions now?
10 000774   001430   BEQ     9$      ;Br if not
11          ;
12          ; Set up par to map to the key region
13          ;
14 000776   004767   000256   CALL   KEYREG      ;Map a par to the key region
15          ;
16          ; Try to find entry for specified key
17          ;
18 001002   016700   176774   MOV    KCODE,R0    ;Get key code
19 001006   004767   000200   CALL  KEYSRC      ;Try to find existing key definition
20 001012   103417   BCS    8$      ;Br if key not defined
21          ;
22          ; We found the key definition. Mark it as free.
23          ;
24 001014   005062   000000G   CLR    KD$COD(R2) ;Say this key no longer defined
25          ;
26          ; See if there are any remaining key definitions
27          ;
28 001020   012702   000000G   MOV    #VPAR1,R2  ;Point to 1st key definition entry
29 001024   016703   000000G   MOV    VKEYMX,R3  ;Get max # key entries
30 001030   005762   000000G   1$:   TST    KD$COD(R2) ;Is this key entry used?
31 001034   001006   BNE    8$      ;Br if yes
32 001036   062702   000000G   ADD    #KD$SZ,R2  ;Point to next key entry
33 001042   077306   SOB    R3,1$    ;Loop if more to check
34          ;
35          ; There are no remaining key definitions.
36          ; Delete the key region.
37          ;
38 001044   004767   000612   CALL  KEYELR      ;Eliminate the key definition region
39 001050   000402   BR     9$
40          ;
41          ; Unmap the par
42          ;
43 001052   004767   000534   8$:   CALL  KEYUMP      ;Unmap the par
44          ;
45          ; Finished
46          ;
47 001056   000167   000000G   9$:   JMP    RDCMD

```


KEYTXT -- Accrue key definition string

```

1          .SBTTL  KEYTXT -- Accrue key definition string
2          ;-----
3          ; Accrue the text string which comprises a key definition.
4          ;
5          ; Inputs:
6          ;   R3 = Pointer to start of text string
7          ;
8          ; Outputs:
9          ;   KTEXT = String in asciz form.
10         ;
11 001062 010246 KEYTXT: MOV     R2, -(SP)
12 001064 105067 177010      CLRB   KTEXT          ; Initially say no string accrued
13         ;
14         ; Skip up to start of string
15         ;
16 001070 004767 000000G    CALL   SKPSPC          ; Skip over any spaces
17 001074 121327 000075    CMPB   (R3), #'='      ; Was equal sign specified before string?
18 001100 001003          BNE    1$              ; Br if not
19 001102 005203          INC    R3                ; Skip past equal sign
20 001104 004767 000000G    CALL   SKPSPC          ; Skip up to string start
21         ;
22         ; See if the string is enclosed in quote marks
23         ;
24 001110 111300          1$:   MOVB   (R3), R0          ; Get 1st char of string
25 001112 001431          BEQ    9$              ; Br if no string specified
26 001114 120027 000047    CMPB   R0, #47         ; Single quote?
27 001120 001403          BEQ    2$              ; Br if yes
28 001122 120027 000042    CMPB   R0, #42         ; Double quote?
29 001126 001014          BNE    3$              ; Br if not
30         ;
31         ; Accrue a string that is enclosed in quotes
32         ;
33 001130 004767 000000G    2$:   CALL   ACRSTR          ; Accrue the string
34 001134 020027 177777G    CMP    R0, #KEYMXT-1 ; Is string too long?
35 001140 101020          BHI    5$              ; Br if yes
36 001142 012700 000000G    MOV    #BLKO, R0      ; Point to buffer with accrued string
37 001146 012702 000100'   MOV    #KTEXT, R2     ; Point to buffer where we want result
38 001152 112022          4$:   MOVB   (R0)+, (R2)+   ; Move the string
39 001154 001376          BNE    4$              ; Loop till null moved
40 001156 000407          BR     9$
41         ;
42         ; Accrue a string that is not enclosed in quotes
43         ;
44 001160 012702 000100'   3$:   MOV    #KTEXT, R2   ; Point to result area
45 001164 020227 000000C   6$:   CMP    R2, #KTEXT+KEYMXT ; Is string too long?
46 001170 101004          BHI    5$              ; Br if yes
47 001172 112322          MOVB   (R3)+, (R2)+   ; Move a character to buffer
48 001174 001373          BNE    6$              ; Loop if more to move
49         ;
50         ; Finished
51         ;
52 001176 012602          9$:   MOV    (SP)+, R2
53 001200 000207          RETURN
54         ;
55         ; String is too long
56         ;
57 001202          5$:   FABORT #EM$STL      ; String too long

```

KEYSRC -- Search for key definition

```

1
2
3
4
5
6
7
8
9
10
11
12
13 001212 010346
14
15
16
17 001214 005767 000000G
18 001220 001412
19
20
21
22 001222 012702 000000G
23 001226 016703 000000G
24 001232 020062 000000G
25 001236 001405
26 001240 062702 000000G
27 001244 077306
28
29
30
31 001246 000261
32 001250 000401
33
34
35
36 001252 000241
37
38
39
40 001254 012603
41 001256 000207

```

```

.SBTTL KEYSRC -- Search for key definition
-----
; Search the key definitions for a specified key code and type.
;
; Inputs:
;   R0 = Key type and code
;
; Outputs:
;   C-flag cleared ==> Found key definition.
;   C-flag set     ==> Key is not defined.
;   R2 = Pointer to key definition if it is found.
;
KEYSRC: MOV     R3, -(SP)
;
; See if there are any defined keys
;
;           TST     KEYRCB           ;Are there any defined keys?
;           BEQ     7$              ;Br if not
;
; Begin loop to search for specified key
;
;           MOV     #VPAR1, R2       ;Point to 1st key definition block
;           MOV     VKEYMX, R3       ;Get # of key entries
1$:        CMP     R0, KD$COD(R2)    ;Is this the one we want?
;           BEQ     2$              ;Br if yes
;           ADD     #KD$$SZ, R2      ;Point to next key def block
;           SOB     R3, 1$          ;Loop if more to check
;
; Key is not currently defined
;
7$:        SEC                      ;Signal failure on return
;           BR      9$
;
; We found the key definition
;
2$:        CLC                      ;Signal success on return
;
; Finished
;
9$:        MOV     (SP)+, R3
;           RETURN

```

KEYREG -- Create PLAS region for key definitions

```

1          .SBTTL  KEYREG -- Create PLAS region for key definitions
2          ;-----
3          ; Try to associate with an existing key definition region.
4          ; Create a new region if one does not already exist.
5          ; Mapping is set up to access the key region.
6          ;
7 001260   010246   KEYREG: MOV     R2,-(SP)
8 001262   010346           MOV     R3,-(SP)
9          ;
10         ; Set up region definition block for region creation
11         ;
12 001264   012767   000000C 000001C           MOV     #<RS.GBL!RS.CGR!RS.PVT>,KRDB+R.GSTS ;Init status flags
13 001272   016703   000000G           MOV     VKEYMX,R3           ;Get max # key definitions
14 001276   070327   000000G           MUL     #KD##SZ,R3           ;Time size of each entry
15 001302   062703   000077           ADD     #63.,R3             ;Round up to next block size
16 001306   072327   177772           ASH     #-6.,R3             ;Convert to # 64-byte blocks
17 001312   010367   000001C           MOV     R3,KRDB+R.GSIZ     ;Set size of region
18         ;
19         ; Try to create the region
20         ;
21 001316           .CRRG   #XAREA,#KRDB           ;Try to create the region
22 001336   103435           BCS     10$                ;Br if cannot create region
23 001340   032767   000000G 000001C           BIT     #RS.CRR,KRDB+R.GSTS ;Was region successfully created?
24 001346   001431           BEQ     10$                ;Br if not
25         ;
26         ; Save information from the region definition block
27         ;
28 001350   016700   000001C           MOV     KRDB+R.GID,RO      ;Get address of region control block
29 001354   010067   000000G           MOV     RO,KEYRCB         ;Save address of region control block
30 001360   016067   000000G 000000G           MOV     RC#BAS(RO),KEYPAR ;Save mapping value to access region
31         ;
32         ; Set up mapping to access the region
33         ;
34 001366   004767   000104           CALL    KEYMAP            ;Map PAR 1 to the region
35         ;
36         ; If we just created a new region, initialize it.
37         ;
38 001372   032767   000000G 000001C           BIT     #RS.NEW,KRDB+R.GSTS ;Did we just create the region?
39 001400   001411           BEQ     9$                ;Br if not
40         ;
41         ; Say no key definitions exist in region
42         ;
43 001402   012702   000000G           MOV     #VPAR1,R2         ;Get pointer to 1st entry
44 001406   016700   000000G           MOV     VKEYMX,RO         ;Get # entries
45 001412   005062   000000G           1$: CLR     KD#COD(R2)      ;Say this entry is free
46 001416   062702   000000G           ADD     #KD##SZ,R2         ;Point to next entry
47 001422   077005           SOB     RO,1$             ;Mark all entries as free
48         ;
49         ; Finished
50         ;
51 001424   012603   9$: MOV     (SP)+,R3
52 001426   012602           MOV     (SP)+,R2
53 001430   000207           RETURN
54         ;
55         ; Error: Unable to create region
56         ; If we are trying to init a virtual line (#KINIT not set yet),
57         ; just print a warning.

```

KEYREG -- Create PLAS region for key definitions

```

58
59 001432 116702 000000G      10$:   MOVB   CORUSR,R2      ;Get our job index
60 001436 032762 000000G 000000G   BIT    $$KINIT,LSW(R2) ;Are we trying to init virtual line?
61 001444 001010                BNE    11$           ;Abort if not.  If initing virt, just warn
62 001446                FWARN  #EM$KCR      ;Warn that keys won't work
63 001462 000261                SEC                    ;Flag error
64 001464 000757                BR     9$           ;And return to INIUKD
65
66                ; If not trying to copy key definitions to a virtual line,
67                ; then abort.
68
69 001466                11$:   FABORT #EM$KCR      ;Cannot create region

```

```

1                                     .SBTTL  KEYMAP -- Map PAR 1 to key definition region
2                                     ;-----
3                                     ; Map PAR 1 (address range 20000 to 37777) to the key definition region.
4                                     ;
5 001476 010346 KEYMAP: MOV      R3, -(SP)
6                                     ;
7                                     ; Initialize window definition block
8                                     ;
9 001500 012767 000000 000001C      MOV      #WS. MAP, KWDB+W. NSTS ;Set status flags
10 001506 016767 000000 000001C     MOV      KEYRCB, KWDB+W. NRID ;Set address of region control block
11 001514 016703 000000 000000G     MOV      VKEYMX, R3           ;Get max # key definitions
12 001520 070327 000000 000000G     MUL      #KD#$SZ, R3         ;Time size of each entry
13 001524 062703 000077 000000G     ADD      #63, R3            ;Round up to next block size
14 001530 072327 177772 000000G     ASH      #-6, R3            ;Convert to # 64-byte blocks
15 001534 010367 000001C 000001C     MOV      R3, KWDB+W. NSIZ   ;Set size of window
16 001540 010367 000001C 000001C     MOV      R3, KWDB+W. NLEN
17                                     ;
18                                     ; Try to create and map the window
19                                     ;
20 001544                                     .CRAW  #XAREA, #KWDB      ;Try to create the window
21 001564 103406 BCS      10#           ;Br if cannot create the window
22 001566 032767 000000 000001C     BIT      #WS. CRW, KWDB+W. NSTS ;Was window successfully created?
23 001574 001402 BEQ      10#           ;Br if not
24                                     ;
25                                     ; Finished
26                                     ;
27 001576 012603 MOV      (SP)+, R3
28 001600 000207 RETURN
29                                     ;
30                                     ; Error: Unable to create the window
31                                     ;
32 001602 10#:  FABORT  #EM$KWC      ;Unable to create window

```

KEYUMP -- Unmap the key definition region

```

1           .SBTTL  KEYUMP -- Unmap the key definition region
2           ; -----
3           ; Eliminate the window used to access the key definition region.
4           ;
5 001612    KEYUMP:
6           ;
7           ; Eliminate the window
8           ;
9 001612    .ELAW  #XAREA,#KWDB  ;Eliminate the window
10          ;
11          ; Disassociate from the region but don't delete it
12          ;
13 001632   012767 000000C 000001C    MOV    #RS.GBL!RS.PVT,KRDB+R.GSTS ;Set status flags
14 001640    .ELRG  #XAREA,#KRDB  ;Disassociate the region
15          ;
16          ; Finished
17          ;
18 001660   000207    RETURN

```

KEYELR -- Eliminate the key definition region

```

1          .SBTTL  KEYELR -- Eliminate the key definition region
2          ;-----
3          ; Eliminate the PLAS region that is used to store key definitions.
4          ;
5 001662   KEYELR:
6          ;
7          ; Eliminate the window
8          ;
9 001662   .ELAW  #XAREA,#KWDB  ;Eliminate the window
10         ;
11        ; Eliminate the region
12        ;
13 001702  012767  000000C 000001C  MOV  #RS. GBL!RS. PVT!RS. EGR, KRDB+R. GSTS ;Set status flags
14 001710  .ELRG  #XAREA,#KRDB  ;Eliminate the region
15        ;
16        ; Say region is gone
17        ;
18 001730  005067  000000G  CLR  KEYRCB  ;Region is gone
19 001734  005067  000000G  CLR  KEYPAR
20        ;
21        ; Finished
22        ;
23 001740  000207  RETURN
24        ;

```

SHOW KEYS

1				. SBTTL SHOW KEYS
2				-----
3				; Process the SHOW KEYS command.
4				;
5	001742			SHOKEY:
6				;
7				; See if there are any defined keys
8				;
9	001742	005767	000000G	TST KEYPAR ;Are there any defined keys?
10	001746	001005		BNE 1\$;Br if yes
11	001750			.PRINT #TM\$KND ;No defined keys
12	001756	000167	000000G	JMP RDCMD
13				;
14				; There are some defined keys.
15				; Associated the key region.
16				;
17	001762	004767	177272	1\$: CALL KEYREG ;Associate the key region
18				;
19				; Begin loop to print information for each defined key
20				;
21	001766	012702	000000G	MOV #VPAR1,R2 ;Get address of first key entry
22	001772	016703	000000G	MOV VKEYMX,R3 ;Get # of key entries
23				;
24				; See if this key entry is defined
25				;
26	001776	005762	000000G	3\$: TST KD\$COD(R2) ;Is this key entry defined?
27	002002	001402		BEQ 2\$;Br if not
28				;
29				; Print information for this key entry
30				;
31	002004	004767	000016	CALL KEYPRT ;Print info for the key definition
32				;
33				; See if there are more key entries
34				;
35	002010	062702	000000G	2\$: ADD #KD\$\$SZ,R2 ;Point to next key entry
36	002014	077310		SOB R3,3\$;Br if more to check
37				;
38				; Disassociate the key region
39				;
40	002016	004767	177570	CALL KEYUMP ;Disassociate key region
41				;
42				; Finished
43				;
44	002022	000167	000000G	JMP RDCMD


```

1                                     .SBTTL  KEYPRT -- Print key definition
2                                     ;-----
3                                     ; Print information about a specific key definition.
4                                     ;
5                                     ; Inputs:
6                                     ; R2 = Pointer to key definition entry
7                                     ;
8 002026 010346 KEYPRT: MOV      R3,-(SP)
9 002030 010546          MOV      R5,-(SP)
10                                    ;
11                                    ; If key type is Gold or Gold-letter, print "Gold"
12                                    ;
13 002032 126227 000000G 000000G          CMPB   KD$TYP(R2),#KT$GLD;Key type gold?
14 002040 001404          BEQ     1$      ;Br if yes
15 002042 126227 000000G 000000G          CMPB   KD$TYP(R2),#KT$GLT;Key type gold letter?
16 002050 001004          BNE     2$      ;Br if not
17 002052          1$:      .PRINT  #TM$GLD      ;Print "Gold "
18 002060 000404          BR      3$
19 002062 012703 000005          2$:      MOV      #5.,R3      ;Print 5 spaces
20 002066 004767 000000G          CALL   PRTSPC
21                                    ;
22                                    ; If key type is letter or gold-letter, key is a single letter
23                                    ;
24 002072 126227 000000G 000000G 3$:      CMPB   KD$TYP(R2),#KT$LET;Letter?
25 002100 001404          BEQ     4$      ;Br if yes
26 002102 126227 000000G 000000G          CMPB   KD$TYP(R2),#KT$GLT;Gold letter?
27 002110 001041          BNE     5$      ;Br if not
28 002112          4$:      .TTYOUT #' "      ;Print opening quote
29 002122 116203 000000G          MOVB   KD$COD(R2),R3      ;Get the letter
30 002126 120327 000040          CMPB   R3,#40      ;Is it a control character?
31 002132 103014          BHIS   10$     ;Br if not
32 002134          .TTYOUT #' ^      ;Print up-arrow
33 002144 062703 000100          ADD    #100,R3      ;Convert char to printing value
34 002150          .TTYOUT R3      ;Print the character
35 002156 012703 000010          MOV    #8.,R3      ;Print 8 spaces
36 002162 000405          BR     11$
37 002164          10$:     .TTYOUT R3      ;Print the character
38 002172 012703 000011          MOV    #9.,R3      ;Print 9 spaces
39 002176          11$:     .TTYOUT #' "      ;Print closing quote
40 002206 004767 000000G          CALL   PRTSPC      ;Print spaces
41 002212 000402          BR     6$
42                                    ;
43                                    ; We must convert key code into key name
44                                    ;
45 002214 004767 000060          5$:      CALL   KEYCOD      ;Convert key code into key name
46                                    ;
47                                    ; Now print the key definition string
48                                    ;
49 002220 010203          6$:      MOV    R2,R3      ;Point to key definition entry
50 002222 062703 000000G          ADD    #KD$TXT,R3      ;Point to start of asciz string
51 002226 112305          8$:      MOVB   (R3)+,R5      ;Get next char from string
52 002230 001415          BEQ    9$      ;Br if hit end of string
53 002232 120527 000040          CMPB   R5,#40      ;Is this a control character?
54 002236 103006          BHIS   7$      ;Br if not
55 002240          .TTYOUT #136      ;Print carret
56 002250 062705 000100          ADD    #100,R5      ;Convert to printing character
57 002254          7$:      .TTYOUT R5      ;Print the character

```

KEYPRT -- Print key definition

```
58 002262 000761          BR      8$          ;Print rest of string
59                          ;
60                          ; Terminate the print line
61                          ;
62 002264          9$:      .PRINT  #CRLF      ;Terminate the print line
63                          ;
64                          ; Finished
65                          ;
66 002272 012605          MOV      (SP)+,R5
67 002274 012603          MOV      (SP)+,R3
68 002276 000207          RETURN
```

```

1                                     .SBTTL KEYCOD -- Print key name based on code
2                                     ;-----
3                                     ; Print the name of a key based on the key code in the current key
4                                     ; descriptor block. The key name is printed in a 12 character field.
5                                     ;
6                                     ; Inputs:
7                                     ; R2 = Pointer to key descriptor block
8                                     ;
9 002300 010346 KEYCOD: MOV R3, -(SP)
10 002302 010546      MOV R5, -(SP)
11                                     ;
12                                     ; Enter loop to find the entry for this key code
13                                     ;
14 002304 012705 000070'      MOV #KNMHD+2, R5 ;Point to first entry in table
15                                     ;
16                                     ; See if this is the entry we want
17                                     ;
18 002310 126265 000000G 000002 1$: CMPB KD$COD(R2), 2(R5); Is this the entry we want?
19 002316 001405      BEQ 2$ ;Br if yes
20 002320 062705 000004      ADD #4, R5 ;Point to next entry
21 002324 005715      TST (R5) ;Is there another entry
22 002326 001370      BNE 1$ ;Br if yes
23 002330 000416      BR 9$ ;Should never happen
24                                     ;
25                                     ; We found the entry. Print the key name.
26                                     ;
27 002332 011505      2$: MOV (R5), R5 ;Get pointer to key name string
28 002334 012703 000014      MOV #12, R3 ;Get field size
29 002340 112500      3$: MOVB (R5)+, R0 ;Get next character from name
30 002342 001407      BEQ 4$ ;Br if hit end of name
31 002344 120027 000052      CMPB R0, #'* ;Is it an asterisk?
32 002350 001773      BEQ 3$ ;Skip them
33 002352      . TTYOUT ;Print a character
34 002356 005303      DEC R3 ;Say one less fill character needed
35 002360 000767      BR 3$ ;Keep printing
36                                     ;
37                                     ; Reached end of name. Blank fill to end of field.
38                                     ;
39 002362 004767 000000G      4$: CALL PRTSPC ;Blank fill rest of field
40                                     ;
41                                     ; Finished
42                                     ;
43 002366 012605      9$: MOV (SP)+, R5
44 002370 012603      MOV (SP)+, R3
45 002372 000207      RETURN

```

INIUKD -- Init user key definitions from parent job

```

1          .SBTTL  INIUKD -- Init user key definitions from parent job
2          ;-----
3          ; INIUKD is called during the start-up processing for a virtual job
4          ; to copy any user key definitions from the parent job.
5          ;
6          ; Inputs:
7          ;   R1 = Current job index number.
8          ;   R2 = Index number of parent job.
9          ;
10         002374  010346  INIUKD: MOV      R3, -(SP)
11         002376  010446          MOV      R4, -(SP)
12         002400  010546          MOV      R5, -(SP)
13         ;
14         ; See if our parent job has any key definitions
15         ;
16         002402  010267  175444          MOV      R2, EMTUKC+4      ;Set our parent job number
17         002406  005067  175442          CLR      EMTUKC+6        ;Clear the block number
18         002412  012700  000046'        MOV      #EMTUKC, R0     ;Point to EMT argument block
19         002416  104375          EMT      375            ;Determine if parent has key defs
20         002420  103003          BCC     1$              ;Br if it has key defs
21         002422  105737  000000G        TSTB   @#ERRLOC        ;Error code 0 ==> No key defs
22         002426  001444          BEQ    9$              ;Br if parent has no key defs
23         ;
24         ; Parent job has user key definitions.
25         ; Copy name of PLAS region swap file to local memory before we
26         ; set up mapping to key region.
27         ;
28         002430  012702  000000G        1$:    MOV      #RSFBLK, R2      ;Point to name cell in TSGEN
29         002434  012703  000004'        MOV      #PSFNAM, R3      ;Point to local name cell
30         002440  012700  000004          MOV      #4, R0           ;Move 4 words
31         002444  012223        4$:    MOV      (R2)+, (R3)+      ;Move file spec to local cell
32         002446  077002          SOB     R0, 4$
33         ;
34         ; Initialize local named region to hold our key definitions.
35         ;
36         002450  004767  176604          CALL   KEYREG            ;Initialize a key def region
37         002454  103431          BCS    9$              ;Skip rest of init (error already reported)
38         ;
39         ; Begin loop to copy key definition info from parent job
40         ;
41         002456  005002          CLR    R2              ;Start with page 0
42         002460  005004          CLR    R4              ;Have not opened chan to PLAS swap file
43         002462  016705  000000G        MOV    VKEYMX, R5      ;Get max # key definitions
44         002466  070527  000000G        MUL   #KD#$SZ, R5     ;Get total # bytes for all key defs
45         002472  006205          ASR    R5              ;Get total # words
46         002474  010503        2$:    MOV    R5, R3        ;Get # words remaining to be moved
47         002476  020327  000400          CMP   R3, #256        ;More than 256 left?
48         002502  101402          BLOS  3$              ;Br if not
49         002504  012703  000400          MOV   #256, R3        ;Move 256 words this time
50         002510  004767  000034        3$:    CALL  KEYMOV          ;Move the key data
51         002514  005202          INC   R2              ;Point to next block
52         002516  160305          SUB   R3, R5          ;Get # words left to be moved
53         002520  003365          BGT   2$              ;Loop if need to move more words
54         ;
55         ; Finished copying data
56         ;
57         002522  004767  177064          CALL  KEYUMP          ;Release region mapping

```

INIUKD -- Init user key definitions from parent job

```
58 002526 005704          TST      R4          ;Did we open chan to PLAS swap file?
59 002530 001403          BEQ      9#          ;Br if not
60 002532                .CLOSE  #1          ;Close plas swap file channel
61                          ;
62                          ; Finished
63                          ;
64 002540 012605          9#:    MOV      (SP)+,R5
65 002542 012604          MOV      (SP)+,R4
66 002544 012603          MOV      (SP)+,R3
67 002546 000207          RETURN
```

KEYMOV -- Move key definition data from parent job

```

1          .SBTTTL  KEYMOV -- Move key definition data from parent job
2          ;-----
3          ; Move up to one block of key definition data from our parent job
4          ; to our key definition region.
5          ;
6          ; Inputs:
7          ;   R2 = Number of page to be moved.
8          ;   R3 = Number of words to move (256 max)
9          ;   R4 = 0==>Channel not opened to PLAS swap file; 1==>Channel open.
10         ;
11         ; Outputs:
12         ;   R4 = 1==> channel opened to plas swap file
13         ;
14 002550 010246 KEYMOV: MOV      R2, -(SP)
15 002552 010346      MOV      R3, -(SP)
16 002554 010546      MOV      R5, -(SP)
17         ;
18         ; Try to obtain the data by accessing in-memory image of parent job
19         ;
20 002556 010267 175272      MOV      R2, EMTUKC+6      ;Set # of block we want
21 002562 012700 000046'    MOV      #EMTUKC, R0      ;Point to EMT arg block
22 002566 104375            EMT      375          ;Try to copy data from in-memory image
23 002570 103034            BCC     1$          ;Br if we got the data
24 002572 010005            MOV     R0, R5        ;Save starting block of region in swap file
25 002574 060205            ADD     R2, R5        ;Add block within region
26         ;
27         ; The parent job is not in memory.
28         ; Open a channel to the PLAS swap file unless it is already open.
29         ;
30 002576 005704            TST     R4          ;Have we opened a channel to plas swap file?
31 002600 001012            BNE     2$          ;Br if yes
32 002602                    .LOOKUP #XAREA, #1, #PSFNAM ;Open channel to PLAS swap file
33 002622 103427            BCS     9$          ;Br if error on lookup
34 002624 005204            INC     R4          ;Set flag saying channel open
35         ;
36         ; Read data from plas swap file
37         ;
38 002626                2$: .READW #XAREA, #1, #BLKO, R3, R5 ;Read data from plas swap file
39         ;
40         ; Move data from BLKO to key def region
41         ;
42 002662 072227 000011    1$:  ASH     #9, R2          ;Convert block # to byte offset
43 002666 062702 000000G   ADD     #VPAR1, R2      ;Get virtual address in region
44 002672 012705 000000G   MOV     #BLKO, R5      ;Point to current buffer
45 002676 012522            3$:  MOV     (R5)+, (R2)+    ;Move the data
46 002700 077302            SOB     R3, 3$        ;Loop if more to move
47         ;
48         ; Finished
49         ;
50 002702 012605            9$:  MOV     (SP)+, R5
51 002704 012603            MOV     (SP)+, R3
52 002706 012602            MOV     (SP)+, R2
53 002710 000207            RETURN

```

RECALL command

```

1          .SBTTL  RECALL command
2          ;-----
3          ; Process the RECALL command that is used with the single line editor.
4          ;
5          ; Inputs:
6          ;   R1 = User index number
7          ;   R2 = Address of end of command string
8          ;   R3 = Address of start of command argument field.
9          ;
10         CMDRCL:
11         ;
12         ; Error if SL is not turned on
13         ;
14         002712 032761 000000G 000000G      BIT    $$SLON,LSW7(R1) ;Is Single line editor turned on?
15         002720 001004                      BNE    1$                ;Br if yes
16         002722                                FABORT #EM#SLO          ;SL not turned on
17         ;
18         ; Advance over the RECALL command
19         ;
20         002732 016704 000000G      1$:    MOV    SLSPTR,R4          ;Get ptr to last command (RECALL)
21         002736 010467 175052        MOV    R4,RCLPTR          ;Save ptr to RECALL command
22         002742 004767 000534        CALL   SLMVUP           ;Advance to next command
23         002746 010467 000000G      MOV    R4,SLSPTR          ;Set new pointers
24         002752 010467 000000G      MOV    R4,SLLPTR
25         ;
26         ; Convert command arguments to upper case
27         ;
28         002756 004767 000000G      CALL   CVTTAB          ;Convert tabs and FF to spaces
29         002762 004767 000000G      CALL   CVTUC           ;Convert lower case letters to upper case
30         ;
31         ; If command has no arguments, treat it like "RECALL 1".
32         ;
33         002766 105713                TSTB   (R3)            ;Are there any arguments?
34         002770 001004                BNE    2$                ;Br if yes
35         002772 012701 000001        MOV    #1,R1           ;Get command number to recall
36         002776 000167 000100        JMP    RCLVAL          ;Go do the recall
37         ;
38         ; See if "/ALL" was specified
39         ;
40         003002 121327 000057      2$:    CMPB   (R3),# '/'    ;Is there a switch present?
41         003006 001016                BNE    3$                ;Br if not
42         003010 126327 000001 000101  CMPB   1(R3),#'A        ;/ALL?
43         003016 001502                BEQ    RCLALL           ;Br if yes
44         003020 126327 000001 000103  CMPB   1(R3),#'C        ;/CLEAR?
45         003026 001404                BEQ    4$                ;Br if yes
46         003030                        FABORT #INVOPT          ;Invalid option
47         003040 000167 000000G      4$:    JMP    RDCMD          ;Ignore /CLEAR
48         ;
49         ; If the first character is a digit, then we are recalling a specific line
50         ;
51         003044 121327 000060      3$:    CMPB   (R3),#'0        ;Is this a digit?
52         003050 103432                BLD    RCLSTR           ;Br if not
53         003052 121327 000071        CMPB   (R3),#'9
54         003056 101027                BHI    RCLSTR
55         003060 000400                BR     RCLNUM           ;Numeric parameter

```

RECALL command

```

1          ; -----
2          ; Recall a command whose index number is specified.
3          ;
4 003062   RCLNUM:
5          ;
6          ; Accrue the number
7          ;
8 003062   004767   000000G   CALL   ACRDEC   ;Accrue the decimal value
9 003066   105700          TSTB   RO       ;Null should be the delimiter
10 003070   001404          BEQ    RCLVAL  ;Br if ok
11 003072          FABORT  #EM#CSE   ;Syntax error
12          ;
13          ; The index number of the command to recall is in R1.
14          ; Recall that command.
15          ;
16 003102   016704   000000G   RCLVAL: MOV    SLSPTR,R4   ;Get ptr to last saved line
17 003106   005301   1$:      DEC    R1       ;Need to go back to an earlier line?
18 003110   003406          BLE    2$       ;Br if not
19 003112   004767   000364          CALL   SLMVUP   ;Move back to earlier line
20 003116   020467   000000G   CMP    R4,SLSPTR ;Wrapped around to beginning?
21 003122   001371          BNE    1$       ;Br if not
22 003124   000402          BR     9$       ;Nothing to recall
23          ;
24          ; Set pointer to line to recall when next input is done
25          ;
26 003126   010467   000000G   2$:     MOV    R4,SLRPTR   ;Recall this line on next input
27 003132   000167   000000G   9$:     JMP    RDCMD

```


RECALL command

```

1
2 ; -----
3 ; Recall a command by searching for a specific character string.
4 003136 016704 000000G RCLSTR: MOV SLSPTR,R4 ;Point to most recently saved command
5 003142 005001 CLR R1 ;Init command recall index
6
7 ; Compare target string with 1st characters of command
8
9 003144 005201 5#: INC R1 ;Increment command recall index
10 003146 010405 MOV R4,R5 ;Get ptr to saved command
11 003150 010302 MOV R3,R2 ;Get ptr to target string
12 003152 105712 1#: TSTB (R2) ;Are we at the end of the target string?
13 003154 001752 BEQ RCLVAL ;Br if hit end of target
14 003156 112500 MOVB (R5)+,R0 ;Get next char from saved cmd
15 003160 001412 BEQ 3# ;Br if hit end of saved cmd
16 003162 020027 000141 CMP R0,#'a ;Cvt lower-case to upper case
17 003166 103405 BLO 4#
18 003170 020027 000172 CMP R0,#'z
19 003174 101002 BHI 4#
20 003176 162700 000040 SUB #40,R0
21 003202 122200 4#: CMPB (R2)+,R0 ;Does target match saved cmd?
22 003204 001762 BEQ 1# ;Loop if yes
23
24 ; This command does not match target.
25 ; Advance to the next saved command.
26
27 003206 004767 000270 3#: CALL SLMVUP ;Advance to next saved command
28 003212 020467 000000G CMP R4,SLSPTR ;Wrapped around to beginning?
29 003216 001352 BNE 5# ;Br if not
30 003220 000167 000000G JMP RDCMD ;Nothing to recall

```

RECALL command

```

1          ; -----
2          ; Recall all commands -- Display the commands.
3          ;
4 003224   016704   000000G  RCLALL: MOV     SLSPTR,R4      ;Point to most recently saved command
5 003230   012705   000001      MOV     #1,R5                ;Initialize command index #
6 003234   012703   000003      MOV     #3,R3                ;Set # columns to print
7 003240   105767   000000G      TSTB   RCLREV              ;Display commands in reverse order?
8 003244   001402           BEQ     3$                  ;Br if normal order
9 003246   004767   000106      CALL   FIND20              ;If reverse order, find 20th/last command
10         ;
11         ; Print the command index number
12         ;
13 003252   004767   000000G  3$:    CALL   PRTFIX        ;Print command index number
14         ;
15         ; Now print the command text
16         ;
17 003256           .PRINT #COLSPC      ;Print colon, space
18 003264   010402      MOV     R4,R2              ;Get ptr to start of command text
19 003266   112200  1$:    MOVB   (R2)+,R0          ;Get next char to print
20 003270   001410      BEQ     2$                  ;Br if hit end of string
21 003272           .TTYOUT              ;Print it
22 003276   020227   000000G      CMP     R2,#SLLEND        ;Hit end of buffer?
23 003302   103771      BLD     1$                  ;Br if not
24 003304   012702   000000G      MOV     #SLLBUF,R2        ;Wrap around to the top
25 003310   000766      BR      1$
26 003312  2$:    .PRINT #CRLF          ;Print CR-LF
27         ;
28         ; Advance to the next command
29         ;
30 003320   105767   000000G      TSTB   RCLREV              ;Normal or reverse order?
31 003324   001405      BEQ     4$                  ;Br if normal order
32 003326   005305      DEC     R5                  ;Any commands left to display?
33 003330   001411      BEQ     9$                  ;Br if not
34 003332   004767   000064      CALL   SLMVDN              ;Else find previous command
35 003336   000745      BR      3$                  ;And br back to display it
36 003340  4$:
37         ;
38         ; ; CMP     R5,#20.          ;Displayed 20 commands?
39         ; ; BHIS   9$                ;That is enough
39 003340   004767   000136      CALL   SLMVUP              ;Advance ptr to next command
40 003344   005205      INC     R5                  ;Inc command index number
41 003346   020467   174442  8$:    CMP     R4,RCLPTR        ;Is there another saved command?
42 003352   001337      BNE     3$                  ;Br if yes
43 003354   000167   000000G  9$:    JMP     RDCMD

```

RECALL command

```

1
2 ; -----
3 ; This routine finds the 20th or last command available in the SL
4 ; recall buffer. It is used when recalling commands in reverse order.
5 ;
6 ; Inputs:
7 ;   R4 = Pointer to most recent saved command
8 ;
9 ; Outputs:
10 ;   R4 = Pointer to last or 20th command, whichever comes first
11 ;   R5 = Number of last command
12 ;
13 ;
14 ;
15 ;
16 ;
17 ;
18 ;
19 ;
20 ;
21 ;
22 ;
23 ;
24 ;
25 ;
26 ;

```

12	003360	010246		FIND20:	MOV	R2,-(SP)	
13	003362	012705	000001		MOV	#1,R5	; Init command index
14	003366	010402			MOV	R4,R2	; Get copy of current command pointer
15	003370	004767	000106	1\$:	CALL	SLMVUP	; Get pointer to previous command
16	003374	020402			CMP	R4,R2	; Is this where we started?
17	003376	001405			BEQ	8\$; Br if done
18	003400	020467	174410		CMP	R4,RCLPTR	; Is this a saved recall command?
19	003404	001402			BEQ	8\$; Don't count recall commands
20	003406	005205			INC	R5	; Count another command
21				::	CMP	R5,#20.	; Must this be the last?
22				::	BHIS	9\$; Br if so
23	003410	000767			BR	1\$; Look for more if not
24	003412	004767	000004	8\$:	CALL	SLMVDN	; Back up to last command
25	003416	012602		9\$:	MOV	(SP)+,R2	
26	003420	000207			RETURN		

RECALL command

```

1
2 ; -----
3 ; This routine is used to locate the beginning of the subsequent
4 ; command in the SL command buffer. It is used when displaying
5 ; recalled commands in reverse order.
6 ;
7 ; Inputs:
8 ;     R4 = Pointer to beginning of current command
9 ;
10 ; Outputs:
11 ;     R4 = Pointer to next command
12 003422 010246 SLMVDN: MOV     R2,-(SP)
13 003424 010402      MOV     R4,R2      ;Get copy of command pointer
14 003426 005304      DEC     R4        ;Step out of it
15 ;
16 ; Find the end of the next command
17 ;
18 003430 020427 000000G 1$:    CMP     R4,#SLLBUF    ;Past the beginning of the buffer?
19 003434 103002      BHIS    2$           ;Br if not
20 003436 012704 177777G      MOV     #<SLEND-1>,R4 ;Wrap around if needed
21 003442 020402      2$:    CMP     R4,R2      ;Back where we started?
22 003444 001414      BEQ     9$           ;Return if so
23 003446 105714      TSTB   @R4          ;Any char here?
24 003450 001002      BNE    3$           ;Br if found a command
25 003452 005304      DEC     R4          ;Else back up some more
26 003454 000765      BR     1$          ;And keep looking
27 ;
28 ; Found the end of a command, find its beginning
29 ;
30 003456 020427 000000G 3$:    CMP     R4,#SLLBUF    ;Past the beginning of the buffer?
31 003462 101002      BHI    4$           ;Br if not
32 003464 012704 000000G      MOV     #SLEND,R4     ;Wrap around if needed
33 003470 105744      4$:    TSTB   -(R4)      ;A char here?
34 003472 001371      BNE    3$           ;Yes, keep looking for beginning
35 003474 005204      INC     R4          ;Aha! Past the begin, point back to it
36 003476 012602      9$:    MOV     (SP)+,R2
37 003500 000207      RETURN

```

RECALL command

```

1
2 ; -----
3 ; This routine returns a pointer to the previous saved command moving
4 ; in an up-arrow direction.
5 ;
6 ; Inputs:
7 ; R4 = Pointer to current command.
8 ;
9 ; Outputs:
10 ; R4 = Pointer to previous command.
11 003502 010546 SLMVUP: MOV R5, -(SP)
12 ;
13 ; Skip up to the null at the end of the current command
14 ;
15 003504 010405 MOV R4, R5
16 003506 020527 000000G 1$: CMP R5, #SLEND ;Hit end of buffer?
17 003512 103402 BLO 2$ ;Br if not
18 003514 012705 000000G MOV #SLLBUF, R5 ;Wrap around to the top
19 003520 105725 2$: TSTB (R5)+ ;Reached null at end of saved command?
20 003522 001371 BNE 1$ ;Br if not
21 ;
22 ; Now skip over nulls at the end of the command
23 ;
24 003524 020527 000000G 3$: CMP R5, #SLEND ;Hit end of buffer?
25 003530 103402 BLO 4$ ;Br if not
26 003532 012705 000000G MOV #SLLBUF, R5 ;Wrap around to the top
27 003536 105725 4$: TSTB (R5)+ ;More nulls to skip?
28 003540 001003 BNE 5$ ;Br if not
29 003542 020504 CMP R5, R4 ;Wrapped around to beginning?
30 003544 001367 BNE 3$ ;Br if not
31 003546 000407 BR 9$ ;Nothing to recall
32 ;
33 ; Point to 1st character of command
34 ;
35 003550 020527 000000G 5$: CMP R5, #SLLBUF ;At top of buffer now?
36 003554 101002 BHI 6$ ;Br if not
37 003556 012705 000000G MOV #SLEND, R5 ;Wrap around to bottom
38 003562 005305 6$: DEC R5 ;Point to 1st char of next command
39 ;
40 ; Finished
41 ;
42 003564 010504 MOV R5, R4 ;Return pointer in R5
43 003566 012605 9$: MOV (SP)+, R5
44 003570 000207 RETURN

```

ACCESS Command

```

1          .SBTTL  ACCESS Command
2          ;-----
3          ; Process the ACCESS command
4          ;
5 003572  004767  000410  CMDACC: CALL  OKFCOK      ;Cmd only valid if priv'd or in startup cf
6 003576  103004          BCC   1$           ;Br if OK
7 003600          FABORT  #EM$NSF    ;Else abort command
8
9 003610  004767  000000G  1$:   CALL  CVTTAB      ;Clean up white spaces in command line
10 003614  016704  000000G  MOV   OKFAND,R4      ;Point to end of ACCESS entries
11
12          ; Top of outer loop for each ACCESS specification
13          ;
14 003620  105713          3$:   TSTB   (R3)       ;End of command line?
15 003622  001002          BNE   4$           ;Br if not
16 003624  000167  000000G  JMP   RDCMD        ;If so, finished, get next command
17
18 003630  020467  000000G  4$:   CMP   R4,OKFNND   ;Are we about to run over NOACCESS entries?
19 003634  103404          BLD   5$           ;Br if OK to add here
20 003636          FABORT  #TBLOVF    ;Else report table overflow and abort
21
22 003646  005764  000000G  5$:   TST   OF$FIL(R4)   ;Is this entry free?
23 003652  001403          BEQ   6$           ;Br if so
24 003654  062704  000000G  ADD   #OF$$SZ,R4    ;If not, point to next slot
25 003660  000763          BR    4$           ;And keep checking
26
27          ; R4 now points to free ACCESS table entry, add entry from command line
28          ; Errors: If there is an error on the dev:fil.ext specification for an
29          ; ACCESS command, then the only problem is forbidding access to a requested
30          ; device -- this error only needs to be a warning.
31          ; However, if there is an error on the switch, then we might end
32          ; up granting full access when some restriction is desired -- abort this.
33          ; In general warn if they get less than they want, but abort if there is
34          ; a chance they could get too much.
35          ;
36 003662  004767  000350  6$:   CALL  OKFINI      ;Init this ACCESS file entry (to all wild)
37 003666  004767  000406  CALL  OKFADD        ;Add cmd line entry to ACCESS table
38 003672  103417          BCS   ACBDFS        ;Report any specification errors
39 003674  012700  000122  MOV   #'R,RO        ;"Readonly" is only valid switch
40 003700  004767  000562  CALL  OKFSWI        ;Go handle switch if specified
41 003704  103447          BCS   ACBDSW        ;Report any switch errors
42
43          ; Finished getting file specification, step up to next entry and repeat
44          ;
45 003706  005267  000000G  15$:  INC   RESDEV        ;Count another entry added
46 003712  062704  000000G  ADD   #OF$$SZ,R4    ;Point to next ACCESS table entry
47 003716  010467  000000G  MOV   R4,OKFAND     ;Remember end of ACCESS entries
48 003722  004767  000242  CALL  SKPD2         ;Skip over terminating delimiters
49 003726  000167  177666  JMP   3$           ;Repeat to end of ACCESS command line

```

ACBDxx -- [NO]ACCESS error handlers

```

1          .SBTTL  ACBDxx -- [NO]ACCESS error handlers
2          ;-----
3          ;
4 003732   ACBDFS:  FWARN  #BADACC      ;Warn of invalid ACCESS specification
5 003746   005064   000000G          CLR      OF$FIL(R4)      ;Make sure slot is released
6          ;
7          ; Now skip up to end of this dev:fil/sw specification
8          ;
9 003752   112300   1$:      MOVVB    (R3)+,R0      ;Get next char
10 003754   001002          BNE      2$      ;Br if not end of command line
11 003756   000167   000000G          JMP      RDCMD      ;At end of line, go get next command
12 003762   120027   000000G          2$:      CMPB    R0,#SPACE      ;Hit space separator?
13 003766   001403          BEQ      3$      ;Br if done skipping
14 003770   120027   000054          CMPB    R0,#COMMA      ;Hit comma separator?
15 003774   001366          BNE      1$      ;Continue skipping if neither
16          ;
17          ; Have skipped over invalid dev:fil/sw specification, but have
18          ; not hit end of line, go back and restart processing of the
19          ; appropriate [NO]ACCESS command and finish the input line
20          ;
21 003776   116701   000000G          3$:      MOVVB    CORUSR,R1      ;Ensure that R1 is correct
22 004002   020467   000000G          CMP      R4,OKFNND      ;Were we ACCESSing or NOACCESSING?
23 004006   103012          BHIS    CMDNAC      ;Must have been NOACCESS, take short branch
24 004010   000167   177556          JMP      CMDACC      ;Must have been ACCESS, take long jump
25          ;
26          ;-----
27          ;
28 004014   ACBDFX:  FABORT  #BADACC      ;Abort command file!
29 004024   ACBDSW:  FABORT  #INVOPT      ;Abort command file!
30          ;

```

NOACCESS Command

```

1          .SBTTL  NOACCESS Command
2          ;-----
3          ; Process the NOACCESS command
4          ;
5 004034 004767 000146  CMDNAC: CALL  OKFCOK      ;Cmd only valid if priv'd or in startup cf
6 004040 103004          BCC   1$          ;Br if OK
7 004042          FABORT #EM$NSF      ;Else abort command
8          ;
9 004052 004767 000000G 1$:   CALL  CVTTAB      ;Compress command line white space
10 004056 016704 000000G  MOV   OKFNND,R4      ;Point to start of NOACCESS entries
11          ;
12          ; Top of outer loop for each NOACCESS specification
13          ;
14 004062 105713          3$:   TSTB   (R3)          ;Have we hit end of command line?
15 004064 001002          BNE   4$          ;Br if not
16 004066 000167 000000G  JMP   RDCMD          ;If so, finished, get next command
17          ;
18 004072 162704 000000G  4$:   SUB   #OF$$SZ,R4      ;Try to make room for new entry
19 004076 020467 000000G  CMP   R4,OKFAND      ;Have we stepped into the ACCESS entries?
20 004102 103004          BHIS  5$          ;Br if OK to add here
21 004104          FABORT #TBLOVF      ;Else report table overflow and abort
22          ;
23 004114 005764 000000G  5$:   TST   OF$FIL(R4)      ;Is this entry free?
24 004120 001364          BNE   4$          ;If not, point to next slot
25          ;
26          ; R4 now points to free NOACCESS table entry, add entry from command line
27          ; Since any error on a NOACCESS would be less restrictive than desired,
28          ; abort, rather than warn, on any errors.
29          ;
30 004122 004767 000110  6$:   CALL  OKFINI      ;Init this NOACCESS file entry (to all wild)
31 004126 004767 000146  CALL  OKFADD      ;Add cmd line entry to NOACCESS table
32 004132 103730          BCS   ACBDFX      ;Abort on any specification errors
33 004134 012700 000127  MOV   #'W,RO      ;"Write" is only valid switch
34 004140 004767 000322  CALL  OKFSWI      ;Go handle switch if specified
35 004144 103727          BCS   ACBDSW      ;Report any switch errors
36          ;
37          ; Finished getting file specification, step up to next entry and repeat
38          ;
39 004146 005267 000000G 15$:  INC   RESDEV      ;Count another entry added
40 004152 010467 000000G  MOV   R4,OKFNND      ;Remember new start of NOACCESS entries
41 004156 004767 000006  CALL  SKPD2      ;Skip over terminating delimiters
42 004162 000167 177674  JMP   3$          ;Repeat to end of NOACCESS command line

```


NO[ACCESS] Command subroutines

```

1          .SBTTL          NO[ACCESS] Command subroutines
2          ;-----
3          ; SUBROUTINE TO SKIP OVER COMMAS AND SPACES
4          ;
5 004166   005203          SKPD1:  INC      R3          ;POINT TO NEXT CHAR
6 004170   121327   000040  SKPD2:  CMPB    (R3),#'          ;IS IT A SPACE
7 004174   001774          BEQ     SKPD1          ;KEEP SKIPPING IF YES
8 004176   121327   000054  CMPB    (R3),#','          ;IS IT A COMMA
9 004202   001771          BEQ     SKPD1          ;KEEP SKIPPING IF YES
10 004204   000207          RETURN
11          ;
12          .SBTTL          Check for privilege to issue [NO]ACCESS
13          ;-----
14          ; ACCESS and NOACCESS commands are only allowed in start-up command files
15          ; or if the user is privileged. Return with carry clear if either condition
16          ; is met; return with carry set if not.
17          ;
18          ; Inputs:
19          ; R1 contains the job index
20          ;
21 004206   032761   000000G 000000G  OKFCOK: BIT     $$SUCF,LSW9(R1) ;Are we in start-up command file?
22 004214   001006          BNE     9$          ;Return OK if so
23 004216   032767   000000G 000000G  BIT     #P0$SYS,PRIVCO    ;Is job privileged?
24 004224   001002          BNE     9$          ;Return OK if so
25 004226   000261          SEC          ;If neither, return with error
26 004230   000401          BR     10$
27 004232   000241          9$:    CLC
28 004234   000207          10$:   RETURN
29          ;
30          .SBTTL          Init a [NO]ACCESS table entry
31          ;-----
32          ; Initialize a [NO]ACCESS table entry to all wildcards.
33          ;
34          ; Inputs:
35          ; R4 points to [NO]ACCESS table entry.
36          ;
37 004236   012702   132500  OKFINI: MOV     #WLDNAM,R2    ;Get RAD50 wildcard value
38 004242   010264   000000G  MOV     R2,OF$FIL(R4)      ;Set FIL to wildcard
39 004246   010264   000002G  MOV     R2,OF$FIL+2(R4)   ;Set NAM to wildcard
40 004252   010264   000004G  MOV     R2,OF$FIL+4(R4)   ;Set EXT to wildcard
41 004256   112764   177777   000000G  MOVB   #-1,OF$DEV(R4)     ;Set device index to wild
42 004264   112764   177777   000000G  MOVB   #-1,OF$UNT(R4)     ;Set device unit to wild
43 004272   105064   000000G  CLRB   OF$FLG(R4)        ;No flags yet
44 004276   000207          RETURN
45          ;
46          .SBTTL          Add entry to [NO]ACCESS tables
47          ;-----
48          ; Accrue device or file name and add entry to [NO]ACCESS table.
49          ; [NO]ACCESS table entry is assumed to be initialized to all wild.
50          ;
51          ; Inputs:
52          ; R3 points to current input command line location
53          ; R4 points to empty [NO]ACCESS table entry
54          ;
55 004300   004767   000000G  OKFADD: CALL   GTRD50       ;Scan first name
56 004304   121327   000072  CMPB   (R3),#'          ;Was this a device name?
57 004310   001036          BNE    10$          ;Nope, leave device as wild, go treat as file

```

Add entry to [NOJACCESS tables

```

58 ;
59 ; We have a device specification
60 ; Translate logical assignments if necessary and insert the entry
61 ;
62 004312 016700 000000G      MOV      R50BUF,R0      ;Get specified device name
63 004316 020027 132500      CMP      RO,#WLDNAM    ;Was it *: ?
64 004322 001417              BEQ      8$            ;If *, skip to file (device is default wild)
65 004324 004767 000000G      CALL     ASNSRC        ;Not wild, convert logical to physical
66 004330 103403              BCS     7$            ;Br if entry not in assign table
67 004332 016267 000000G 000000G  MOV      AT$DEV(R2),R50BUF ;Replace logical name with physical name
68 ;
69 ; Translate physical device specification to device table index
70 ;
71 004340 016701 000000G 7$:  MOV      R50BUF,R1      ;Copy device name
72 004344 004767 000170      CALL     CVTDEV        ;Convert into dev index (R1) and unit # (R2)
73 004350 103444              BCS     13$           ;Br if bad device name
74 004352 110164 000000G      MOVB    R1,OF$DEV(R4)  ;Set device index into access table
75 004356 110264 000000G      MOVB    R2,OF$UNT(R4) ;Set device unit into access table
76 ;
77 ; We need to accrue the file specification
78 ;
79 004362 005203 8$:  INC      R3            ;Skip over :
80 004364 111300      MOVB    (R3),R0        ;Get first char of file spec
81 004366 120027 000052      CMPB   RO,#'*        ;Wild card? (CHKDLM would find it)
82 004372 001403      BEQ      9$            ;If yes, go accrue it
83 004374 004767 000000G      CALL     CHKDLM        ;1st char of filnam shouldn't be delim
84 004400 103426      BCS     12$           ;If it is, must be dev:, or dev:/switch
85 004402 004767 000000G 9$:  CALL     GTRD50        ;Accrue file name
86 ;
87 ; We have a file specification
88 ;
89 004406 016700 000000G 10$: MOV      R50BUF,R0     ;Get RAD50 file name (1-3)
90 004412 001423      BEQ      13$           ;Br on invalid file spec
91 004414 020027 132500      CMP      RO,#WLDNAM   ;File name wild?
92 004420 001405      BEQ      11$           ;Br if so (already init'd to wild)
93 004422 010064 000000G      MOV      RO,OF$FIL(R4) ;Set file name into access table
94 004426 016764 000002G 000002G  MOV      R50BUF+2,OF$FIL+2(R4) ;Set 2nd half (4-6) of name
95 ;
96 ; Check for file extension
97 ;
98 004434 121327 000056 11$:  CMPB   (R3),#'        ;Was file extension specified?
99 004440 001006      BNE     12$           ;Br if not (already init'd to wild)
100 004442 005203      INC     R3            ;Step over .
101 004444 004767 000000G      CALL     GTRD50        ;Accrue the extension
102 004450 016764 000000G 000004G  MOV      R50BUF,OF$FIL+4(R4) ;Set extension into access table
103 ;
104 ; Normal return with no error
105 ;
106 004456 000241 12$:  CLC
107 004460 000207      RETURN
108 ;
109 ; Return thru here on invalid device or file specification
110 ;
111 004462 000261 13$:  SEC
112 004464 000207      RETURN
113 ;
114 .SBTTL          Handle [NOJACCESS switches

```

Handle [NO]ACCESS switches

```

115 ; -----
116 ; This routine checks to see if a switch was specified and set the
117 ; read-only flag for the entry if so. The only valid switches are
118 ; /R or /r for ACCESS and /W or /w for NOACCESS. Examples:
119 ;
120 ;     ACCESS  dev:[filnam[.ext]]/Readonly
121 ;     NOACCESS dev:[filnam[.ext]]/Write
122 ;
123 ; Inputs:
124 ;     R0 contains 1st char (uppercase) of valid switch
125 ;     R3 points to 1st char after dev or file specification
126 ;     R4 points to [NO]ACCESS table entry
127 ;
128 004466 121327 000057 OKFSWI: CMPB    (R3),#'/      ; Switch specified?
129 004472 001020      BNE     9$          ; Just return if not
130 004474 116346 000001      MOVB   1(R3),-(SP)    ; Get 1st char of switch
131 004500 042716 000040      BIC    #40,(SP)      ; Force it to uppercase ('a-'A)
132 004504 122600      CMPB   (SP)+,R0      ; Does it say read-only?
133 004506 001402      BEQ    13$          ; Br if so
134 004510 000261      SEC     ; Else this is an invalid switch
135 004512 000411      BR     10$          ; And return
136 004514 152764 000000G 000000G 13$:  BISB   #OT$RON,OF$FLG(R4) ; Set read-only flag on file spec
137 ;
138 ; Now step the cmd line ptr up to the end of the switch
139 ;
140 004522 005203      14$:  INC     R3          ; Step up to next delimiter
141 004524 111300      MOVB   (R3),R0      ; Get the next char
142 004526 004767 000000G      CALL   CHKDLM        ; Is it a delimiter?
143 004532 103373      BCC    14$          ; Keep searching for end of switch
144 ;
145 004534 000241      9$:   CLC          ; Normal return
146 004536 000207      10$:  RETURN

```

Convert RAD50 dev name to index and unit

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 004540
15
16
17
18 004540 012702 177777
19 004544 005000
20 004546 071027 000050
21 004552 005701
22 004554 001412
23 004556 010102
24 004560 020127 000035
25 004564 001406
26 004566 162701 000036
27 004572 010102
28 004574 020227 000011
29 004600 101027
30 004602 070027 000050
31
32
33
34
35
36
37 004606 020167 000000G
38 004612 001403
39 004614 020167 000000G
40 004620 001007
41 004622 016701 000000G
42 004626 005702
43 004630 002016
44 004632 116702 000001G
45 004636 000413
46
47
48
49 004640 016700 000000G
50 004644 020160 000000G
51 004650 001405
52 004652 162700 000002
53 004656 002372
54
55
56
57 004660 000261

```

```

.SBTTL          Convert RAD50 dev name to index and unit
-----
; CVTDEV is called to convert a RAD50 device name to the corresponding
; device table index number and unit number.
;
; Inputs:
;   R1 = RAD50 device name.
;
; Outputs:
;   R1 = Device table index number for device.
;   R2 = Unit number.
;   C-flag set on return if invalid device name.
;
CVTDEV:
;   Get device name and split off unit number.
;
;       MOV     #-1,R2          ;ASSUME NO UNIT NUMBER SPECIFIED
;       CLR     R0              ;SET FOR DIVIDE
;       DIV     #50,R0         ;SPLIT OFF LOW-ORDER RAD50 CHARACTER
;       TST     R1              ;WAS A UNIT NUMBER SPECIFIED?
;       BEQ     6$             ;BR IF NOT (LEAVE -1, INDICATES UNSPECIFIED)
;       MOV     R1,R2          ;Not unspecified, assume * for the moment
;       CMP     R1,#R50AST     ;Is unit number a wild card?
;       BEQ     6$             ;Preserve it if so, else
;       SUB     #36,R1         ;CONVERT RAD50 DIGIT TO BINARY VALUE
;       MOV     R1,R2          ;PUT UNIT NUMBER IN R2
;       CMP     R2,#9         ;MAKE SURE UNIT NUMBER IN RANGE 0-9.
;       BHI     8$             ;BR IF INVALID UNIT NUMBER
6$:      MUL     #50,R0         ;GET DEVICE NAME WITHOUT UNIT #
;
;   The RAD50 device name less unit number is now in R1.
;   The unit number is in R2.
;
;   Translate SY and DK.
;
;       CMP     R1,R50SY      ;IS DEVICE NAME SY?
;       BEQ     2$           ;BR IF YES
;       CMP     R1,R50DK     ;IS DEVICE NAME DK?
;       BNE     3$           ;BR IF NOT
2$:      MOV     SYINDX,R1    ;GET SY DEVICE INDEX NUMBER
;       TST     R2           ;Was unit number specified?
;       BGE     4$           ;Yes, must have been 0-9 or *
;       MOVB   SYUNIT+1,R2  ;No, just SY:, use real boot unit
;       BR     4$
;
;   Look up device name in permanent device name table.
;
3$:      MOV     NUMDEV,R0    ;GET INDEX OF LAST PERM NAME
5$:      CMP     R1,PNAME(R0) ;LOOK UP DEVICE NAME
;       BEQ     7$           ;BR IF FOUND
;       SUB     #2,R0        ;CHECK NEXT ENTRY
;       BGE     5$
;
;   Invalid device name
;
8$:      SEC                ;SIGNAL ERROR ON RETURN

```

```
58 004662 000412          BR      9$
59                          ;
60                          ; Found device name.
61                          ;
62 004664 010001          7$:     MOV      R0,R1          ;GET DEVICE NAME TABLE INDEX
63 004666 020227 000035  4$:     CMP      R2,#R50AST        ;Unit number or * or unspecified?
64 004672 103405          BLO      10$          ;0-9, leave as is
65 004674 101003          BHI      11$          ;Unspecified, replace with 0
66 004676 012702 177777  MOV      #-1,R2        ;Replace * with wild (-1)
67 004702 000401          BR      10$          ;Done
68 004704 005002          11$:    CLR      R2          ;Unspecified unit becomes 0
69 004706 000241          10$:    CLC          ;SIGNAL NO ERROR ON RETURN
70                          ;
71                          ; Return
72                          ;
73 004710 000207          9$:     RETURN
```

GETSYSP -- Accept system logon password

```

1          .SBTTL  GETSYSP -- Accept system logon password
2          ;-----
3          ; The system logon password is an optional password which the system
4          ; may require for some lines before entering the normal logon sequence.
5          ;
6          ; Inputs:
7          ; R1 = Job index number
8          ;
9 004712 010246 GETSYSP: MOV     R2, -(SP)
10 004714 010346    MOV     R3, -(SP)
11 004716 010546    MOV     R5, -(SP)
12          ;
13          ; Don't ask for a password if there is none defined
14          ;
15 004720 105767 000000G    TSTB    SYPSWD      ;Is there a defined system password?
16 004724 001535    BEQ     9$          ;Br if not
17          ;
18          ; Pretend to lock KMON to line to prevent ^C aborts
19          ;
20 004726 052761 000000G 000000G    BIS     #$PRQLK, LSW5(R1)
21          ;
22          ; Log off job immediately if it has been aborted
23          ;
24 004734 032761 000000G 000000G    BIT     #$DOOFF, LSW(R1) ;Should the job be logged off?
25 004742 001132    BNE     20$          ;Br if yes
26          ;
27          ; Disable control-C abort and control-W subprocess switch
28          ;
29 004744 042767 000000G 000000G    BIC     #P2$VIR, PRIVC2 ;Disable subprocess use
30 004752 004767 000000G          CALL    FIXPRV      ;Transfer privilege flag to LSW tables
31 004756 052761 000000G 000000G    BIS     #$SCCA, LSW5(R1) ;Suppress control-C aborts
32          ;
33          ; Initialize count of number of times password has been entered
34          ;
35 004764 012705 000002    MOV     #2, R5      ;Init password attempt count
36          ;
37          ; Print password prompt
38          ;
39 004770 1$: .PRINT  #CRLF      ;Print CR-LF
40 004776    .PRINT  #SYSPSR   ;Print password prompt
41          ;
42          ; Set read time-out time
43          ;
44 005004 012700 000060'    MOV     #RDTIME, R0 ;Point to EMT arg block
45 005010 016760 000000G 000002    MOV     VONTM, 2(R0) ;Set timeout value
46 005016 104375          EMT     375          ;Set read time-out
47          ;
48          ; Accept a line of input
49          ;
50 005020 012702 000000G    MOV     #BLKO, R2   ;Point to buffer for input string
51 005024 042761 000000G 000000G    BIC     #$ECHO, LSW2(R1) ;Turn off character echoing
52 005032 2$: .TTYIN          ;Get next input character
53 005036 120027 000001    CMPB   R0, #1      ;Did read time-out?
54 005042 001472    BEQ     20$          ;Br if yes
55 005044 120027 000015    CMPB   R0, #CR     ;Ignore carriage return
56 005050 001770    BEQ     2$          ;
57 005052 120027 000012    CMPB   R0, #LF     ;End of input line?

```

GETSYP -- Accept system logon password

```

58 005056 001405          BEQ      3$          ;Br if yes
59 005060 020227 000062G  CMP      R2,#BLK0+50. ;Input line too long?
60 005064 101362          BHI      2$          ;Br if yes
61 005066 110022          MOVVB   R0,(R2)+      ;Store received character
62 005070 000760          BR       2$          ;Continue receiving input
63 005072 105012          3$:     CLRBB   (R2)    ;Store null at end of string
64                                     ;
65                                     ; Convert received and expected passwords to upper case
66                                     ;
67 005074 012703 000000G  MOV      #BLK0,R3     ;Point to received password
68 005100 004767 000000G  CALL    CVTUC        ;Convert to upper case
69 005104 012703 000000G  MOV      #SYPSWD,R3   ;Point to expected password
70 005110 004767 000000G  CALL    CVTUC        ;Convert to upper case
71                                     ;
72                                     ; See if received password matches expected password
73                                     ;
74 005114 012702 000000G  MOV      #BLK0,R2     ;Point to received string
75 005120 012703 000000G  MOV      #SYPSWD,R3   ;Point to expected password
76 005124 122213          4$:     CMPBB   (R2)+,(R3) ;Does password match?
77 005126 001003          BNE      5$          ;Br if not
78 005130 105723          TSTB    (R3)+        ;Are we at the end of the password?
79 005132 001374          BNE      4$          ;Br if not
80 005134 000402          BR       6$          ;Got correct password
81                                     ;
82                                     ; Invalid password.
83                                     ; Give one extra try then give up.
84                                     ;
85 005136 077564          5$:     SOB      R5,1$   ;Loop if user gets another try
86 005140 000433          BR       20$         ;Give up
87                                     ;
88                                     ; Password successfully entered
89                                     ;
90 005142 052761 000000G 000000G 6$:     BIS      #$ECHO,LSW2(R1) ;Turn echoing back on
91 005150 052761 000000G 000000G  BIS      #$NOIN,LSW3(R1) ;Reset flags for start-up command file
92 005156 052761 000000G 000000G  BIS      #$SUCF,LSW9(R1)
93 005164 042761 000000G 000000G  BIC      #$PRGLK,LSW5(R1) ;Unlock KMON
94 005172 042761 000000G 000000G  BIC      #$SCCA,LSW5(R1) ;Remove control-C suppression
95 005200 016767 000000G 000000G  MOV      PRIVA2,PRIVC2 ;Reset privilege flags
96 005206 004767 000000G  CALL    FIXPRV       ;Transfer privilege flag to LSW tables
97 005212          .PRINT #CRLF      ;Print CR-LF
98                                     ;
99                                     ; Finished
100                                    ;
101 005220 012605          9$:     MOV      (SP)+,R5
102 005222 012603          MOV      (SP)+,R3
103 005224 012602          MOV      (SP)+,R2
104 005226 000207          RETURN
105                                    ;
106                                     ; Password not successfully entered.
107                                     ; Log off the job.
108                                     ;
109 005230 012700 000066' 20$:    MOV      #HNGEMT,R0 ;Point to logoff emt
110 005234 104375          EMT      375          ;Log off the job
111                                     ;
112                                     ; .END

```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
Work file writes: 0
Size of work file: 10168 Words (40 Pages)
Size of core pool: 18176 Words (71 Pages)
Operating system: RT-11

Elapsed time: 00:01:10.35
.LP: TSKM2B=DK: TSKM2B/C/N: SYM

SHOKEY	1-21	17-5#							
SKPD1	32-5#	32-7	32-9						
SKPD2	29-48	31-41	32-6#						
SKPSPC	1-40	6-13	7-48	11-16	11-20				
SLLBUF	1-30	25-24	27-18	27-30	28-18	28-26	28-35		
SLLEND	1-30	25-22	27-20	27-32	28-16	28-24	28-37		
SLLPTR	1-29	22-24*							
SLMVDN	25-34	26-24	27-12#						
SLMVUP	22-22	23-19	24-27	25-39	26-15	28-11#			
SLRPTR	1-29	23-26*							
SLSPTR	1-29	22-20	22-23#	23-16	23-20	24-4	24-28	25-4	
SPACE	1-26	30-12							
SPACE2	1-30								
SYINDX	1-27	33-41							
SYPSPR	1-33	34-40							
SYPSWD	1-33	34-15	34-69	34-75					
SYUNIT	1-27	33-44							
TAB	1-56#								
TBLOVF	1-25	29-20	31-21						
TM#GLD	1-34	18-17							
TM#KND	1-34	17-11							
TSKM2B	1-5#								
VKEYMX	1-42	7-21	10-29	12-23	13-13	13-44	14-11	17-22	20-43
VONTM	1-32	34-45							
VPAR1	1-39	10-28	12-22	13-43	17-21	21-43			
VSLEDT	1-36	7-15							
W. NLEN	1-34	14-16*							
W. NRID	1-38	14-10*							
W. NSIZ	1-34	14-15*							
W. NSTS	1-38	14-9*	14-22						
WLDNAM	1-60#	32-37	32-63	32-91					
WS. CRW	1-37	14-22							
WS. MAP	1-38	2-26	14-9						
XAREA	1-39	13-21	14-20	15-9	15-14	16-9	16-14	21-32	21-38

