

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	GETSYP -- Accept system logon password

```

1          .TITLE  TSKM2B --- Keyboard DEFINE Command routines
2          .ENABL  LC
3          .DSABL  QBL
4 000000    .CSECT  TSKM2B
5 000000    TSKM2B:
6          ;
7          ; TSKM2B is the portion of TSKMON that contains the code
8          ; to implement the DEFINE command.
9          ;
10         ; Copyright 1985.
11         ; S&H Computer Systems, Inc.
12         ; Nashville, Tennessee
13         ;
14         ; Macro calls
15         ;
16         .MCALL  .ELRG, .ELAW, .CRRG, .CRAW, .TTYOUT, .PRINT
17         .MCALL  .LOOKUP, .READW, .CLOSE, .TTYIN
18         ;
19         ; Global definitions
20         ;
21         .GLOBL  CMDDEF, SHOKEY, INIUKD, GETSYP, CMDRCL
22         ;
23         ; Global references
24         ;
25         .GLOBL  $DOOFF, LSW, EM$SLO, LSW7, $SLON, SLSPTR, SLLPTR, SLRPTR
26         .GLOBL  EM$CSE, PRTFIX, SPACE2, SLLEND, SLLBUF, INVOPT
27         .GLOBL  P2$VIR, FIXPRV, $SCCA, LSW5, PRIVA2, PRIVC2, $PRGLK
28         .GLOBL  LSW2, CVTUC, VOFFTM, ACRTXT, RCLREV
29         .GLOBL  SYPSWD, SYSPSR, $ECHO, $NOIN, LSW3, $SUCF, LSW9
30         .GLOBL  TM$KND, TM$GLD, CRLF, R. GSI, W. NSIZ, W. NLEN, BLKO
31         .GLOBL  RSFBLK, ERRLOC, EM$KNS, ACRDEC
32         .GLOBL  VSLEDT, EM$NSL, RS. NEW, RS. EGR, KEYRCB, PRTSPC, KD$TYP
33         .GLOBL  INVOPT, ILLCMD, EM$KCR, EM$KWC, WS. CRW, EM$KNT
34         .GLOBL  BLKO, ACRSTR, KEYMXT, EM$STL, W. NRID, W. NSTS, WS. MAP
35         .GLOBL  R. GID, R. GSTS, RS. CRR, RS. GBL, RS. CGR, RS. PVT, XAREA, VPAR1
36         .GLOBL  KF$ECO, KF$TRM, KT$GLD, CVTTAB, SKPSPC, SEARCH, AMBOPT, FKILL
37         .GLOBL  KT$NRM, OPTLST, KT$LET, KD$COD, KD$FLG, KD$TXT, RDCMD, KEYPAR
38         .GLOBL  VKEYMX, KD$SZ, EM$KTF, KT$GLT, EM$KNU, KEYRCB, RC$BAS
39         .GLOBL  KC$KP0, KC$KP1, KC$KP2, KC$KP3, KC$KP4, KC$KP5, KC$KP6
40         .GLOBL  KC$KP7, KC$KP8, KC$KP9, KC$DOT, KC$COM, KC$MIN, KC$ENT
41         .GLOBL  KC$UP, KC$DWN, KC$LFT, KC$RIT, KC$E1, KC$E2, KC$E3, KC$E4
42         .GLOBL  KC$E5, KC$E6, KC$F6, KC$F7, KC$F8, KC$F9, KC$F10, KC$F11
43         .GLOBL  KC$F12, KC$F13, KC$F14, KC$F15, KC$F16, KC$F17, KC$F18
44         .GLOBL  KC$F19, KC$F20, KC$PF1, KC$PF2, KC$PF3, KC$PF4
45         ;
46         ; Assembly constants
47         ;
48         000012    LF      =      12          ; LINE FEED
49         000015    CR      =      15          ; CARRIAGE RETURN
50         000040    BLANK   =      40          ; ASCII SPACE
51         000007    BELL    =      07          ; ASCII BELL
52         000011    TAB     =      11          ; HORIZONTAL TAB
53         000014    FF      =      14          ; FORM FEED
54         000054    COMMA   =      54          ; COMMA
55         000400    BLKWDS  =      256         ; # OF WORDS IN DISK BLOCK
56         132500    WLDNAM  =      132500      ; RAD50 /*/ (WILDCARD)

```

```

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 0000000 .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 0000000 .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 0000000 .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     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 ; Macro to end a set of table entries.  
60 ;  
61 .MACRO TBLEND  
62 .CSECT CMDV2B  
63 .WORD 0  
64 .CSECT TSKM2B  
65 .ENDM TBLEND
```

```

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

```

```

1
2 ; -----
3 ; Define names of keys and associated code values.
4
5 TBLDEF KNM, 1
6 CMDDEF PF1, KC$PF1
7 CMDDEF PF2, KC$PF2
8 CMDDEF PF3, KC$PF3
9 CMDDEF PF4, KC$PF4
10 CMDDEF KP0, KC$KP0
11 CMDDEF KP1, KC$KP1
12 CMDDEF KP2, KC$KP2
13 CMDDEF KP3, KC$KP3
14 CMDDEF KP4, KC$KP4
15 CMDDEF KP5, KC$KP5
16 CMDDEF KP6, KC$KP6
17 CMDDEF KP7, KC$KP7
18 CMDDEF KP8, KC$KP8
19 CMDDEF KP9, KC$KP9
20 CMDDEF O, KC$KP0
21 CMDDEF 1, KC$KP1
22 CMDDEF 2, KC$KP2
23 CMDDEF 3, KC$KP3
24 CMDDEF 4, KC$KP4
25 CMDDEF 5, KC$KP5
26 CMDDEF 6, KC$KP6
27 CMDDEF 7, KC$KP7
28 CMDDEF 8, KC$KP8
29 CMDDEF 9, KC$KP9
30 CMDDEF PER*IOD, KC$DOT
31 CMDDEF COM*MA, KC$COM
32 CMDDEF MIN*US, KC$MIN
33 CMDDEF ENT*ER, KC$ENT
34 CMDDEF LEFT, KC$LEFT
35 CMDDEF RIGHT, KC$RIGHT
36 CMDDEF UP, KC$UP
37 CMDDEF DO, KC$F16
38 CMDDEF DOWN, KC$DWN
39 CMDDEF FIND, KC$E1
40 CMDDEF <INS*ERT-HERE>, KC$E2
41 CMDDEF REM*OVE, KC$E3
42 CMDDEF SEL*ECT, KC$E4
43 CMDDEF <PREV*--SCREEN>, KC$E5
44 CMDDEF <NEXT*--SCREEN>, KC$E6
45 CMDDEF E1, KC$E1
46 CMDDEF E2, KC$E2
47 CMDDEF E3, KC$E3
48 CMDDEF E4, KC$E4
49 CMDDEF E5, KC$E5
50 CMDDEF E6, KC$E6
51 CMDDEF HELP, KC$F15
52 CMDDEF F6, KC$F6
53 CMDDEF F7, KC$F7
54 CMDDEF F8, KC$F8
55 CMDDEF F9, KC$F9
56 CMDDEF F10, KC$F10
57 CMDDEF F11, KC$F11
58 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          .SBTTL  DEFINE command
2          ;-----
3          ; Process the DEFINE command.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number
7          ;   R3 = Pointer to start of qualifiers for DEFINE command.
8          ;
9 000220 004767 0000000  CMDDEF: CALL    CVTTAB          ;Convert tab and FF chars to spaces
10         ;
11         ; If 1st character is slash, skip it
12         ;
13 000224 004767 0000000          CALL    SKPSPC          ;Skip over any spaces
14 000230 121327 000057          CMPB    (R3),# '/'        ;Is 1st character slash?
15 000234 001001                BNE      1$                ;Br if not
16 000236 005203                INC      R3                ;Point beyond slash
17         ;
18         ; Branch off to major processing routines based on first qualifier
19         ;
20 000240 012704 000000' 1$:      MOV      #DEFHD,R4          ;Point to table of options
21 000244 004767 0000000          CALL    SEARCH          ;Try to find correct processing routine
22 000250 103401                BCS      2$                ;Br if don't recognize option keyword
23 000252 000134                JMP      @(R4)+            ;Enter major processing routine
24         ;
25         ; Invalid keyword
26         ;
27 000254 005704                2$:      TST      R4          ;Ambiguous or unrecognized option?
28 000256 001404                BEQ      3$                ;Br if unrecognized
29 000260                                FABORT #AMBOPT      ;Ambiguous option
30 000270                                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  MOVB      #KT$NRM,KTYPE ;Default to normal key type
29 000346 112767 0000000  MOVB      #KF$TRM!KF$ECD,KFLAG ;Initialize flags
30         ;
31         ; Process any options
32         ;
33 000354 012704 000010'  MOV      #KEYHD,R4      ;Point to option list
34 000360 004767 0000000  CALL      OPTLST      ;Process the options
35         ;
36         ; R3 should now be pointing to the key name
37         ;
38 000364 105713         TSTB      (R3)          ;Was something specified?
39 000366 001456         BEQ      20$          ;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$:  CALL      SKPSPC      ;Skip up to start of string
49 000414 121327 000047  CMPB      (R3),#47      ;Is letter enclosed in quotes?
50 000420 001413         BEQ      6$          ;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 0000000 6$:    CALL    ACRTXT      ;Accrue the letter
60 000454 020027 000001    CMP      R0,#1        ;Should have gotten exactly 1 char
61 000460 001025          BNE      21$          ;Br if not
62 000462 116767 0000000 177312 MOVW    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$:    MOV      #KNMHD,R4      ;Point to key name table
68 000476 004767 0000000    CALL    SEARCH      ;Try to translate key name
69 000502 103414          BCS      21$          ;Br if can't recognize key name
70 000504 111467 177272    MOVW    (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

```

KEYSDP -- Set key definition option flags

```

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    0000000    CMPB      RO,#KT$GLD      ;Is type /GOLD?
24 000566    001005                      BNE      1$          ;Br if not
25 000570    126727    177207    0000000    CMPB      KTYPE,#KT$LET  ;Is current type /LETTER?
26 000576    001012                      BNE      2$          ;Br if not
27 000600    000407                      BR      3$
28 000602    120027    0000000    1$:  CMPB      RO,#KT$LET  ;Is type /LETTER?
29 000606    001006                      BNE      2$          ;Br if not
30 000610    126727    177167    0000000    CMPB      KTYPE,#KT$GLD  ;Is current type /GOLD?
31 000616    001002                      BNE      2$          ;Br if not
32 000620    112700    0000000    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$ECHO!KF$TRM,KFLAG ;Either ECHO or TERMINATE specified?
15         000640 001004                      BNE     5$                ;Br if yes
16         000642                      FABORT  #EM$KNT                ;Can't have both NOECHO and NOTERMINATE
17         ;
18         ; Create a PLAS region for key definitions if we don't already have one
19         ;
20         000652 005767 0000000 5$:          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 000636                      CALL  KEYUMP          ;Unmap par
44         000720                      FABORT  #EM$KTF                ;Key table full
45         ;
46         ; The entry to use is pointed to by R2.
47         ; Make the entry.
48         ;
49         000730 016762 177046 0000000 2$:    MOV     KCODE,KD$COD(R2); Save key code
50         000736 116762 177036 0000000      MOV     KFLAG,KD$FLG(R2); Save option flags
51         000744 062702 0000000      ADD     #KD$TXT,R2          ;Point to area for text string
52         000750 012703 000100'      MOV     #KTEXT,R3
53         000754 112322 3$:      MOV     (R3)+,(R2)+          ;Store the string
54         000756 001376                      BNE     3$
55         ;
56         ; Finished
57         ;

```

KEYADD -- Add a new key definition

58 000760 004767 000572
59 000764 000167 0000000

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 0000000  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 0000000  CLR   KD$COD(R2) ;Say this key no longer defined
25          ;
26          ; See if there are any remaining key definitions
27          ;
28 001020 012702 0000000  MOV   #VPAR1,R2  ;Point to 1st key definition entry
29 001024 016703 0000000  MOV   VKEYMX,R3  ;Get max # key entries
30 001030 005762 0000000 1$: TST   KD$COD(R2) ;Is this key entry used?
31 001034 001006          BNE   8$          ;Br if yes
32 001036 062702 0000000  ADD   #KD$$SZ,R2  ;Point to next key entry
33 001042 077306          SUB   R3,1$       ;Loop if more to check
34          ;
35          ; There are no remaining key definitions.
36          ; Delete the key region.
37          ;
38 001044 004767 000556  CALL  KEYELR     ;Eliminate the key definition region
39 001050 000402          BR    9$
40          ;
41          ; Unmap the par
42          ;
43 001052 004767 000500 8$: CALL  KEYUMP     ;Unmap the par
44          ;
45          ; Finished
46          ;
47 001056 000167 0000000 9$: 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 0000000     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 0000000     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 0000000 2$:    CALL    ACRSTR        ; Accrue the string
34 001134 020027 1777770     CMP     R0, #KEYMXT-1    ; Is string too long?
35 001140 101020           BHI     5$                ; Br if yes
36 001142 012700 0000000     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$:    MOV     #KTEXT, R2     ; Point to result area
45 001164 020227 0000000 6$:    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          .SBTTL  KEYSRC -- Search for key definition
2          ;-----
3          ; Search the key definitions for a specified key code and type.
4          ;
5          ; Inputs:
6          ;   R0 = Key type and code
7          ;
8          ; Outputs:
9          ;   C-flag cleared ==> Found key definition.
10         ;   C-flag set    ==> Key is not defined.
11         ;   R2 = Pointer to key definition if it is found.
12         ;
13 001212  010346  KEYSRC: MOV      R3,-(SP)
14         ;
15         ; See if there are any defined keys
16         ;
17 001214  005767  0000000  TST      KEYRCB      ;Are there any defined keys?
18 001220  001412          BEQ      7$           ;Br if not
19         ;
20         ; Begin loop to search for specified key
21         ;
22 001222  012702  0000000  MOV      #VPAR1,R2      ;Point to 1st key definition block
23 001226  016703  0000000  MOV      VKEYMX,R3      ;Get # of key entries
24 001232  020062  0000000  1$:  CMP      R0,KD$COD(R2) ;Is this the one we want?
25 001236  001405          BEQ      2$           ;Br if yes
26 001240  062702  0000000  ADD      #KD$$SZ,R2      ;Point to next key def block
27 001244  077306          SOB      R3,1$         ;Loop if more to check
28         ;
29         ; Key is not currently defined
30         ;
31 001246  000261  7$:  SEC                      ;Signal failure on return
32 001250  000401          BR      9$
33         ;
34         ; We found the key definition
35         ;
36 001252  000241  2$:  CLC                      ;Signal success on return
37         ;
38         ; Finished
39         ;
40 001254  012603  9$:  MOV      (SP)+,R3
41 001256  000207          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 000000C    MOV      #<CRS.GBL!RS.CGR!RS.PVT>,KRDB+R.GSTS ;Init status flags
13 001272    016703    0000000    MOV      VKEYMX,R3          ;Get max # key definitions
14 001276    070327    0000000    MUL      #KD#$SZ,R3          ;Time size of each entry
15 001302    062703    000077    ADD      #63,R3              ;Round up to next block size
16 001306    072327    177772    ASH      #-6,R3              ;Convert to # 64-byte blocks
17 001312    010367    000000C    MOV      R3,KRDB+R.GSIZ      ;Set size of region
18         ;
19         ; Try to create the region
20         ;
21 001316    .CRRG     #XAREA,#KRDB      ;Try to create the region
22 001336    103435    BCS      10$       ;Br if cannot create region
23 001340    032767    0000000 000000C    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    000000C    MOV      KRDB+R.GID,R0       ;Get address of region control block
29 001354    010067    0000000    MOV      R0,KEYRCB          ;Save address of region control block
30 001360    016067    0000000 000000C    MOV      RC$BAS(R0),KEYPAR;Save mapping value to access region
31         ;
32         ; Set up mapping to access the region
33         ;
34 001366    004767    000050    CALL     KEYMAP              ;Map PAR 1 to the region
35         ;
36         ; If we just created a new region, initialize it.
37         ;
38 001372    032767    0000000 000000C    BIT      #RS.NEW,KRDB+R.GSTS ;Did we just create the region?
39 001400    001411    BEQ      9$        ;Br if not
40         ;
41         ; Say no key definitions exist in region
42         ;
43 001402    012702    0000000    MOV      #VPAR1,R2          ;Get pointer to 1st entry
44 001406    016700    0000000    MOV      VKEYMX,R0          ;Get # entries
45 001412    005062    0000000    1$: CLR      KD$COD(R2)       ;Say this entry is free
46 001416    062702    0000000    ADD      #KD#$SZ,R2          ;Point to next entry
47 001422    077005    SOB      R0,1$      ;Mark all entries as free
48         ;
49         ; Finished
50         ;
51 001424    012603    9$: MOV      (SP)+,R3
52 001426    012602    MOV      (SP)+,R2
53 001430    000207    RETURN
54         ;
55         ; Error: Unable to create region
56         ;
57 001432    10$: FABORT  #EM$KCR          ;Cannot create region

```

KEYMAP -- Map PAR 1 to key definition 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 001442 010346 KEYMAP: MOV      R3, -(SP)
6          ;
7          ; Initialize window definition block
8          ;
9 001444 012767 0000000 0000000C      MOV      #WS.MAP, KWDB+W.NSTS ;Set status flags
10 001452 016767 0000000 0000000C      MOV      KEYRCB, KWDB+W.NRID ;Set address of region control block
11 001460 016703 0000000          MOV      VKEYMX, R3          ;Get max # key definitions
12 001464 070327 0000000          MUL      #KD#$SZ, R3          ;Time size of each entry
13 001470 062703 000077          ADD      #63, R3          ;Round up to next block size
14 001474 072327 177772          ASH      #-6, R3          ;Convert to # 64-byte blocks
15 001500 010367 0000000          MOV      R3, KWDB+W.NSIZ ;Set size of window
16 001504 010367 0000000          MOV      R3, KWDB+W.NLEN
17          ;
18          ; Try to create and map the window
19          ;
20 001510          .CRAW  #XAREA, #KWDB      ;Try to create the window
21 001530 103406          BCS      10$      ;Br if cannot create the window
22 001532 032767 0000000 0000000C      BIT      #WS.CRW, KWDB+W.NSTS ;Was window successfully created?
23 001540 001402          BEQ      10$      ;Br if not
24          ;
25          ; Finished
26          ;
27 001542 012603          MOV      (SP)+, R3
28 001544 000207          RETURN
29          ;
30          ; Error: Unable to create the window
31          ;
32 001546          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 001556    KEYUMP:
6          ;
7          ; Eliminate the window
8          ;
9 001556    .ELAW  #XAREA,#KWDB      ;Eliminate the window
10         ;
11         ; Disassociate from the region but don't delete it
12         ;
13 001576 012767 000000C 000000C    MOV    #RS.GBL!RS.PVT,KRDB+R.GSTS ; Set status flags
14 001604    .ELRG  #XAREA,#KRDB      ;Disassociate the region
15         ;
16         ; Finished
17         ;
18 001624 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 001626    KEYELR:
6          ;
7          ; Eliminate the window
8          ;
9 001626    .ELAW  #XAREA,#KWDB      ;Eliminate the window
10         ;
11         ; Eliminate the region
12         ;
13 001646    012767  000000C 000000C      MOV    #RS.GBL!RS.PVT!RS.EGR,KRDB+R.GSTS ;Set status flags
14 001654    .ELRG  #XAREA,#KRDB      ;Eliminate the region
15         ;
16         ; Say region is gone
17         ;
18 001674    005067  0000000          CLR    KEYRCB      ;Region is gone
19 001700    005067  0000000          CLR    KEYPAR
20         ;
21         ; Finished
22         ;
23 001704    000207          RETURN
24         ;
```

SHOW KEYS

```

1          .SBTTL  SHOW KEYS
2          ;-----
3          ; Process the SHOW KEYS command.
4          ;
5 001706   SHOKEY:
6          ;
7          ; See if there are any defined keys
8          ;
9 001706   005767   0000000   TST      KEYPAR      ;Are there any defined keys?
10 001712   001005   BNE      1$      ;Br if yes
11 001714   .PRINT  #TM#KND      ;No defined keys
12 001722   000167   0000000   JMP      RDCMD
13          ;
14          ; There are some defined keys.
15          ; Associated the key region.
16          ;
17 001726   004767   177326   1$:      CALL      KEYREG      ;Associate the key region
18          ;
19          ; Begin loop to print information for each defined key
20          ;
21 001732   012702   0000000   MOV      #VPAR1,R2      ;Get address of first key entry
22 001736   016703   0000000   MOV      VKEYMX,R3      ;Get # of key entries
23          ;
24          ; See if this key entry is defined
25          ;
26 001742   005762   0000000   3$:      TST      KD$COD(R2)      ;Is this key entry defined?
27 001746   001402   BEQ      2$      ;Br if not
28          ;
29          ; Print information for this key entry
30          ;
31 001750   004767   000016   CALL      KEYPRT      ;Print info for the key definition
32          ;
33          ; See if there are more key entries
34          ;
35 001754   062702   0000000   2$:      ADD      #KD#$SZ,R2      ;Point to next key entry
36 001760   077310   SOB      R3,3$      ;Br if more to check
37          ;
38          ; Disassociate the key region
39          ;
40 001762   004767   177570   CALL      KEYUMP      ;Disassociate key region
41          ;
42          ; Finished
43          ;
44 001766   000167   0000000   JMP      RDCMD

```

KEYPRT -- Print key definition

```

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 001772 010346  KEYPRT: MOV     R3, -(SP)
9 001774 010546      MOV     R5, -(SP)
10         ;
11         ; If key type is Gold or Gold-letter, print "Gold"
12         ;
13 001776 126227 000000G 000000G      CMPB    KD$TYP(R2), #KT$GLD; Key type gold?
14 002004 001404      BEQ     1$          ; Br if yes
15 002006 126227 000000G 000000G      CMPB    KD$TYP(R2), #KT$GLT; Key type gold letter?
16 002014 001004      BNE     2$          ; Br if not
17 002016      1$:      .PRINT    #TM$GLD)      ; Print "Gold "
18 002024 000404      BR      3$
19 002026 012703 000005      2$:      MOV     #5., R3          ; Print 5 spaces
20 002032 004767 000000G      CALL    PRTSPC
21         ;
22         ; If key type is letter or gold-letter, key is a single letter
23         ;
24 002036 126227 000000G 000000G 3$:      CMPB    KD$TYP(R2), #KT$LET; Letter?
25 002044 001404      BEQ     4$          ; Br if yes
26 002046 126227 000000G 000000G      CMPB    KD$TYP(R2), #KT$GLT; Gold letter?
27 002054 001041      BNE     5$          ; Br if not
28 002056      4$:      .TTYOUT    #' '          ; Print opening quote
29 002066 116203 000000G      MOVB    KD$COD(R2), R3      ; Get the letter
30 002072 120327 000040      CMPB    R3, #40          ; Is it a control character?
31 002076 103014      BHIS     10$         ; Br if not
32 002100      .TTYOUT    #'^          ; Print up-arrow
33 002110 062703 000100      ADD     #100, R3          ; Convert char to printing value
34 002114      .TTYOUT    R3          ; Print the character
35 002122 012703 000010      MOV     #8., R3          ; Print 8 spaces
36 002126 000405      BR      11$
37 002130      10$:      .TTYOUT    R3          ; Print the character
38 002136 012703 000011      MOV     #9., R3          ; Print 9 spaces
39 002142      11$:      .TTYOUT    #' '          ; Print closing quote
40 002152 004767 000000G      CALL    PRTSPC          ; Print spaces
41 002156 000402      BR      6$
42         ;
43         ; We must convert key code into key name
44         ;
45 002160 004767 000060      5$:      CALL    KEYCOD          ; Convert key code into key name
46         ;
47         ; Now print the key definition string
48         ;
49 002164 010203      6$:      MOV     R2, R3          ; Point to key definition entry
50 002166 062703 000000G      ADD     #KD$TXT, R3      ; Point to start of asciz string
51 002172 112305      8$:      MOVB    (R3)+, R5      ; Get next char from string
52 002174 001415      BEQ     9$          ; Br if hit end of string
53 002176 120527 000040      CMPB    R5, #40          ; Is this a control character?
54 002202 103006      BHIS     7$          ; Br if not
55 002204      .TTYOUT    #136          ; Print carret
56 002214 062705 000100      ADD     #100, R5          ; Convert to printing character
57 002220      7$:      .TTYOUT    R5          ; Print the character

```


KEYPRT -- Print key definition

```

58 002226 000761          BR      B#          ;Print rest of string
59                          ;
60                          ; Terminate the print line
61                          ;
62 002230          9#:      .PRINT  #CRLF      ;Terminate the print line
63                          ;
64                          ; Finished
65                          ;
66 002236 012605          MOV      (SP)+,R5
67 002240 012603          MOV      (SP)+,R3
68 002242 000207          RETURN

```


KEYCOD -- Print key name based on code

```

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 002244 010346 KEYCOD: MOV     R3, -(SP)
10 002246 010546      MOV     R5, -(SP)
11          ;
12          ; Enter loop to find the entry for this key code
13          ;
14 002250 012705 000070'      MOV     #KNMHD+2, R5      ; Point to first entry in table
15          ;
16          ; See if this is the entry we want
17          ;
18 002254 126265 0000000 000002 1$: CMPB   KD$COD(R2), 2(R5); Is this the entry we want?
19 002262 001405      BEQ     2$      ; Br if yes
20 002264 062705 000004      ADD     #4., R5      ; Point to next entry
21 002270 005715      TST     (R5)      ; Is there another entry
22 002272 001370      BNE     1$      ; Br if yes
23 002274 000416      BR      9$      ; Should never happen
24          ;
25          ; We found the entry. Print the key name.
26          ;
27 002276 011505 2$: MOV     (R5), R5      ; Get pointer to key name string
28 002300 012703 000014      MOV     #12., R3      ; Get field size
29 002304 112500 3$: MOVB    (R5)+, R0      ; Get next character from name
30 002306 001407      BEQ     4$      ; Br if hit end of name
31 002310 120027 000052      CMPB    R0, #'*      ; Is it an asterisk?
32 002314 001773      BEQ     3$      ; Skip them
33 002316      .TTYOUT      ; Print a character
34 002322 005303      DEC     R3      ; Say one less fill character needed
35 002324 000767      BR      3$      ; Keep printing
36          ;
37          ; Reached end of name. Blank fill to end of field.
38          ;
39 002326 004767 0000000 4$: CALL    PRTSPC      ; Blank fill rest of field
40          ;
41          ; Finished
42          ;
43 002332 012605 9$: MOV     (SP)+, R5
44 002334 012603      MOV     (SP)+, R3
45 002336 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         INIUKD: MOV     R3, -(SP)
11         MOV     R4, -(SP)
12         MOV     R5, -(SP)
13         ;
14         ; See if our parent job has any key definitions
15         ;
16         MOV     R2, EMTUKC+4      ; Set our parent job number
17         CLR     EMTUKC+6          ; Clear the block number
18         MOV     #EMTUKC, R0       ; Point to EMT argument block
19         EMT     375               ; Determine if parent has key defs
20         BCC     1$               ; Br if it has key defs
21         TSTB    @#ERRLOC          ; Error code 0 ==> No key defs
22         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         1$: MOV     #RSFBLK, R2    ; Point to name cell in TSGEN
29         MOV     #PSFNAM, R3       ; Point to local name cell
30         MOV     #4, R0            ; Move 4 words
31         4$: MOV     (R2)+, (R3)+   ; Move file spec to local cell
32         SOB     R0, 4$
33         ;
34         ; Initialize local named region to hold our key definitions.
35         ;
36         CALL    KEYREG            ; Initialize a key def region
37         ;
38         ; Begin loop to copy key definition info from parent job
39         ;
40         CLR     R2                ; Start with page 0
41         CLR     R4                ; Have not opened chan to PLAS swap file
42         MOV     VKEYMX, R5        ; Get max # key definitions
43         MUL     #KD#$SZ, R5       ; Get total # bytes for all key defs
44         ASR     R5                ; Get total # words
45         2$: MOV     R5, R3         ; Get # words remaining to be moved
46         CMP     R3, #256          ; More than 256 left?
47         BLOS    3$               ; Br if not
48         MOV     #256, R3          ; Move 256 words this time
49         3$: CALL    KEYMOV        ; Move the key data
50         INC     R2                ; Point to next block
51         SUB     R3, R5            ; Get # words left to be moved
52         BGT     2$               ; Loop if need to move more words
53         ;
54         ; Finished copying data
55         ;
56         CALL    KEYUMP            ; Release region mapping
57         TST     R4                ; Did we open chan to PLAS swap file?

```

INIUKD -- Init user key definitions from parent job

```

58 002472 001403          BEQ      9#          ;Br if not
59 002474          .CLOSE  #1          ;Close plus swap file channel
60
61          ; Finished
62          ;
63 002502 012605      9#:  MOV      (SP)+, R5
64 002504 012604          MOV      (SP)+, R4
65 002506 012603          MOV      (SP)+, R3
66 002510 000207          RETURN

```

KEYMOV -- Move key definition data from parent job

```

1          .SBTTL  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 002512  010246 KEYMOV: MOV      R2,-(SP)
15 002514  010346      MOV      R3,-(SP)
16 002516  010546      MOV      R5,-(SP)
17         ;
18         ; Try to obtain the data by accessing in-memory image of parent job
19         ;
20 002520  010267 175330      MOV      R2,EMTUKC+6      ;Set # of block we want
21 002524  012700 000046'      MOV      #EMTUKC,R0      ;Point to EMT arg block
22 002530  104375      EMT      375      ;Try to copy data from in-memory image
23 002532  103034      BCC      1$      ;Br if we got the data
24 002534  010005      MOV      R0,R5      ;Save starting block of region in swap file
25 002536  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 002540  005704      TST      R4      ;Have we opened a channel to plas swap file?
31 002542  001012      BNE      2$      ;Br if yes
32 002544      .LOOKUP #XAREA,#1,#PSFNAM ;Open channel to PLAS swap file
33 002564  103427      BCS      9$      ;Br if error on lookup
34 002566  005204      INC      R4      ;Set flag saying channel open
35         ;
36         ; Read data from plas swap file
37         ;
38 002570      2$:      .READW  #XAREA,#1,#BLK0,R3,R5 ;Read data from plas swap file
39         ;
40         ; Move data from BLK0 to key def region
41         ;
42 002624  072227 000011      1$:      ASH      #9.,R2      ;Convert block # to byte offset
43 002630  062702 0000006      ADD      #VPAR1,R2      ;Get virtual address in region
44 002634  012705 0000006      MOV      #BLK0,R5      ;Point to current buffer
45 002640  012522      3$:      MOV      (R5)+,(R2)+      ;Move the data
46 002642  077302      SOB      R3,3$      ;Loop if more to move
47         ;
48         ; Finished
49         ;
50 002644  012605      9$:      MOV      (SP)+,R5
51 002646  012603      MOV      (SP)+,R3
52 002650  012602      MOV      (SP)+,R2
53 002652  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         002654 032761 0000000 0000000      BIT    ##SLON,LSW7(R1) ; Is Single line editor turned on?
15         002662 001004                      BNE     1$           ; Br if yes
16         002664                      FABORT   #EM$SLO           ; SL not turned on
17         ;
18         ; Advance over the RECALL command
19         ;
20         002674 016704 0000000      1$:      MOV     SLSPTR,R4      ; Get ptr to last command (RECALL)
21         002700 010467 175110      MOV     R4,RCLPTR      ; Save ptr to RECALL command
22         002704 004767 000534      CALL    SLMVUP      ; Advance to next command
23         002710 010467 0000000      MOV     R4,SLSPTR      ; Set new pointers
24         002714 010467 0000000      MOV     R4,SLLPTR
25         ;
26         ; Convert command arguments to upper case
27         ;
28         002720 004767 0000000      CALL    CVTTAB      ; Convert tabs and FF to spaces
29         002724 004767 0000000      CALL    CVTUC      ; Convert lower case letters to upper case
30         ;
31         ; If command has no arguments, treat it like "RECALL 1".
32         ;
33         002730 105713                      TSTB     (R3)      ; Are there any arguments?
34         002732 001004                      BNE     2$           ; Br if yes
35         002734 012701 000001      MOV     #1,R1      ; Get command number to recall
36         002740 000167 000100      JMP     RCLVAL      ; Go do the recall
37         ;
38         ; See if "/ALL" was specified
39         ;
40         002744 121327 000057      2$:      CMPB     (R3),# '/'      ; Is there a switch present?
41         002750 001016                      BNE     3$           ; Br if not
42         002752 126327 000001 000101      CMPB     1(R3),# 'A      ; /ALL?
43         002760 001502                      BEQ     RCLALL      ; Br if yes
44         002762 126327 000001 000103      CMPB     1(R3),# 'C      ; /CLEAR?
45         002770 001404                      BEQ     4$           ; Br if yes
46         002772                      FABORT   #INVOPT      ; Invalid option
47         003002 000167 0000000      4$:      JMP     RDCMD      ; Ignore /CLEAR
48         ;
49         ; If the first character is a digit, then we are recalling a specific line
50         ;
51         003006 121327 000060      3$:      CMPB     (R3),# '0      ; Is this a digit?
52         003012 103432                      BLO     RCLSTR      ; Br if not
53         003014 121327 000071      CMPB     (R3),# '9
54         003020 101027                      BHI     RCLSTR
55         003022 000400                      BR      RCLNUM      ; Numeric parameter

```

RECALL command

```

1      ; -----
2      ; Recall a command whose index number is specified.
3      ;
4 003024 RCLNUM:
5      ;
6      ; Accrue the number
7      ;
8 003024 004767 0000000 CALL ACRDEC ;Accrue the decimal value
9 003030 105700 TSTB RO ;Null should be the delimiter
10 003032 001404 BEQ RCLVAL ;Br if ok
11 003034 FABORT #EM#CSE ;Syntax error
12      ;
13      ; The index number of the command to recall is in R1.
14      ; Recall that command.
15      ;
16 003044 016704 0000000 RCLVAL: MOV SLSPTR,R4 ;Get ptr to last saved line
17 003050 005301 1$: DEC R1 ;Need to go back to an earlier line?
18 003052 003406 BLE 2$ ;Br if not
19 003054 004767 000364 CALL SLMVUP ;Move back to earlier line
20 003060 020467 0000000 CMP R4,SLSPTR ;Wrapped around to beginning?
21 003064 001371 BNE 1$ ;Br if not
22 003066 000402 BR 9$ ;Nothing to recall
23      ;
24      ; Set pointer to line to recall when next input is done
25      ;
26 003070 010467 0000000 2$: MOV R4,SLRPTR ;Recall this line on next input
27 003074 000167 0000000 9$: JMP RDCMD

```

RECALL command

```

1
2 ; -----
3 ; Recall a command by searching for a specific character string.
4 003100 016704 0000000 RCLSTR: MOV     SLSPTR,R4      ;Point to most recently saved command
5 003104 005001          CLR     R1          ;Init command recall index
6
7 ; Compare target string with 1st characters of command
8
9 003106 005201 5$:      INC     R1          ;Increment command recall index
10 003110 010405        MOV     R4,R5        ;Get ptr to saved command
11 003112 010302        MOV     R3,R2        ;Get ptr to target string
12 003114 105712 1$:     TSTB    (R2)        ;Are we at the end of the target string?
13 003116 001752        BEQ     RCLVAL      ;Br if hit end of target
14 003120 112500        MOVB    (R5)+,R0    ;Get next char from saved cmd
15 003122 001412        BEQ     3$          ;Br if hit end of saved cmd
16 003124 020027 000141 CMP     R0,#'a      ;Cvt lower-case to upper case
17 003130 103405        BLQ     4$
18 003132 020027 000172 CMP     R0,#'z
19 003136 101002        BHI     4$
20 003140 162700 000040 SUB     #40,R0
21 003144 122200 4$:     CMPB    (R2)+,R0    ;Does target match saved cmd?
22 003146 001762        BEQ     1$          ;Loop if yes
23
24 ; This command does not match target.
25 ; Advance to the next saved command.
26
27 003150 004767 000270 3$:   CALL    SLMVUP      ;Advance to next saved command
28 003154 020467 0000000 CMP     R4,SLSPTR    ;Wrapped around to beginning?
29 003160 001352        BNE     5$          ;Br if not
30 003162 000167 0000000 JMP     RDCMD      ;Nothing to recall

```


RECALL command

```

1
2 ; -----
3 ; Recall all commands --- Display the commands.
4 003166 016704 0000000 RCLALL: MOV     SLSPTR,R4      ;Point to most recently saved command
5 003172 012705 0000001      MOV     #1,R5      ;Initialize command index #
6 003176 012703 0000003      MOV     #3,R3      ;Set # columns to print
7 003202 105767 0000000      TSTB    RCLREV     ;Display commands in reverse order?
8 003206 001402      BEQ     3$      ;Br if normal order
9 003210 004767 000106      CALL     FIND20     ;If reverse order, find 20th/last command
10
11 ; Print the command index number
12
13 003214 004767 0000000 3$:      CALL     PRTFIX      ;Print command index number
14
15 ; Now print the command text
16
17 003220      .PRINT   #COLSPC      ;Print colon, space
18 003226 010402      MOV     R4,R2      ;Get ptr to start of command text
19 003230 112200 1$:      MOVB    (R2)+,R0      ;Get next char to print
20 003232 001410      BEQ     2$      ;Br if hit end of string
21 003234      .TTYOUT      ;Print it
22 003240 020227 0000000      CMP     R2,#SLLEND    ;Hit end of buffer?
23 003244 103771      BLD     1$      ;Br if not
24 003246 012702 0000000      MOV     #SLLBUF,R2      ;Wrap around to the top
25 003252 000766      BR      1$
26 003254 2$:      .PRINT   #CRLF      ;Print CR-LF
27
28 ; Advance to the next command
29
30 003262 105767 0000000      TSTB    RCLREV     ;Normal or reverse order?
31 003266 001405      BEQ     4$      ;Br if normal order
32 003270 005305      DEC     R5      ;Any commands left to display?
33 003272 001411      BEQ     9$      ;Br if not
34 003274 004767 0000064      CALL     SLMVDN     ;Else find previous command
35 003300 000745      BR      3$      ;And br back to display it
36 003302 4$:
37 ;
38 ;
39 003302 004767 000136      CMP     R5,#20      ;Displayed 20 commands?
40 003306 005205      BHIS    9$      ;That is enough
41 003310 020467 174500      CALL     SLMVUP     ;Advance ptr to next command
42 003314 001337      INC     R5      ;Inc command index number
43 003316 000167 0000000 8$:      CMP     R4,RCLPTR     ;Is there another saved command?
9$:      BNE     3$      ;Br if yes
      JMP     RDCMD

```


RECALL command

```

1      ;-----
2      ; This routine finds the 20th or last command available in the SL
3      ; recall buffer. It is used when recalling commands in reverse order.
4      ;
5      ; Inputs:
6      ;   R4 = Pointer to most recent saved command
7      ;
8      ; Outputs:
9      ;   R4 = Pointer to last or 20th command, whichever comes first
10     ;   R5 = Number of last command
11     ;
12 003322 010246 FIND20: MOV     R2, -(SP)
13 003324 012705      MOV     #1, R5      ; Init command index
14 003330 010402      MOV     R4, R2      ; Get copy of current command pointer
15 003332 004767 000106 1$: CALL    SLMVUP ; Get pointer to previous command
16 003336 020402      CMP     R4, R2      ; Is this where we started?
17 003340 001405      BEQ     8$          ; Br if done
18 003342 020467 174446      CMP     R4, RCLPTR ; Is this a saved recall command?
19 003346 001402      BEQ     8$          ; Don't count recall commands
20 003350 005205      INC     R5          ; Count another command
21      ;      CMP     R5, #20.         ; Must this be the last?
22      ;      BHS     9$              ; Br if so
23 003352 000767      BR      1$          ; Look for more if not
24 003354 004767 000004 8$: CALL    SLMVDN ; Back up to last command
25 003360 012602      MOV     (SP)+, R2
26 003362 000207      RETURN

```

```

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 003364 010246 SLMVDN: MOV     R2, -(SP)
13 003366 010402      MOV     R4, R2          ;Get copy of command pointer
14 003370 005304      DEC      R4            ;Step out of it
15
16 ; Find the end of the next command
17 ;
18 003372 020427 0000000 1$:    CMP     R4, #SLLBUF      ;Past the beginning of the buffer?
19 003376 103002      BHS     2$                ;Br if not
20 003400 012704 1777777      MOV     #<SLEND-1>, R4 ;Wrap around if needed
21 003404 020402      2$:    CMP     R4, R2          ;Back where we started?
22 003406 001414      BEQ     9$                ;Return if so
23 003410 105714      TSTB    @R4              ;Any char here?
24 003412 001002      BNE     3$                ;Br if found a command
25 003414 005304      DEC      R4              ;Else back up some more
26 003416 000765      BR      1$              ;And keep looking
27
28 ; Found the end of a command, find its beginning
29 ;
30 003420 020427 0000000 3$:    CMP     R4, #SLLBUF      ;Past the beginning of the buffer?
31 003424 101002      BHI     4$                ;Br if not
32 003426 012704 0000000      MOV     #SLEND, R4      ;Wrap around if needed
33 003432 105744      4$:    TSTB    -(R4)          ;A char here?
34 003434 001371      BNE     3$                ;Yes, keep looking for beginning
35 003436 005204      INC      R4              ;Aha! Past the begin, point back to it
36 003440 012602      9$:    MOV     (SP)+, R2
37 003442 000207      RETURN

```

```

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 003444 010546 SLMVUP: MOV     R5, -(SP)
12 ;
13 ; Skip up to the null at the end of the current command
14 ;
15 003446 010405      MOV     R4, R5
16 003450 020527 0000000 1$:     CMP     R5, #SLLEND      ; Hit end of buffer?
17 003454 103402      BLD     2$                      ; Br if not
18 003456 012705 0000000      MOV     #SLLBUF, R5      ; Wrap around to the top
19 003462 105725 2$:     TSTB    (R5)+                ; Reached null at end of saved command?
20 003464 001371      BNE     1$                      ; Br if not
21 ;
22 ; Now skip over nulls at the end of the command
23 ;
24 003466 020527 0000000 3$:     CMP     R5, #SLLEND      ; Hit end of buffer?
25 003472 103402      BLD     4$                      ; Br if not
26 003474 012705 0000000      MOV     #SLLBUF, R5      ; Wrap around to the top
27 003500 105725 4$:     TSTB    (R5)+                ; More nulls to skip?
28 003502 001003      BNE     5$                      ; Br if not
29 003504 020504      CMP     R5, R4                  ; Wrapped around to beginning?
30 003506 001367      BNE     3$                      ; Br if not
31 003510 000407      BR      9$                      ; Nothing to recall
32 ;
33 ; Point to 1st character of command
34 ;
35 003512 020527 0000000 5$:     CMP     R5, #SLLBUF      ; At top of buffer now?
36 003516 101002      BHI     6$                      ; Br if not
37 003520 012705 0000000      MOV     #SLLEND, R5      ; Wrap around to bottom
38 003524 005305 6$:     DEC     R5                  ; Point to 1st char of next command
39 ;
40 ; Finished
41 ;
42 003526 010504      MOV     R5, R4                  ; Return pointer in R5
43 003530 012605 9$:     MOV     (SP)+, R5
44 003532 000207      RETURN

```

GETSYP -- Accept system logon password

```

1          .SBTTL  GETSYP --- 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 003534 010246 GETSYP: MOV     R2,-(SP)
10 003536 010346      MOV     R3,-(SP)
11 003540 010546      MOV     R5,-(SP)
12          ;
13          ; Don't ask for a password if there is none defined
14          ;
15 003542 105767 0000000 TSTB     SYPSWD      ;Is there a defined system password?
16 003546 001535      BEQ      9$             ;Br if not
17          ;
18          ; Pretend to lock KMON to line to prevent ^C aborts
19          ;
20 003550 052761 0000000 0000000 BIS     #$PRGLK,LSW5(R1)
21          ;
22          ; Log off job immediately if it has been aborted
23          ;
24 003556 032761 0000000 0000000 BIT     #$DOOFF,LSW(R1) ;Should the job be logged off?
25 003564 001132      BNE      20$           ;Br if yes
26          ;
27          ; Disable control-C abort and control-W subprocess switch
28          ;
29 003566 042767 0000000 0000000 BIC     #P2$VIR,PRIVC2 ;Disable subprocess use
30 003574 004767 0000000      CALL    FIXPRV   ;Transfer privilege flag to LSW tables
31 003600 052761 0000000 0000000 BIS     #$SCCA,LSW5(R1) ;Suppress control-C aborts
32          ;
33          ; Initialize count of number of times password has been entered
34          ;
35 003606 012705 0000002      MOV     #2,R5      ;Init password attempt count
36          ;
37          ; Print password prompt
38          ;
39 003612 1$: .PRINT  #CRLF      ;Print CR-LF
40 003620      .PRINT  #SYSPSR   ;Print password prompt
41          ;
42          ; Set read time-out time
43          ;
44 003626 012700 000060'      MOV     #RDTIME,R0 ;Point to EMT arg block
45 003632 016760 0000000 0000002 MOV     VOFFTM,2(R0) ;Set timeout value
46 003640 104375      ENT      375             ;Set read time-out
47          ;
48          ; Accept a line of input
49          ;
50 003642 012702 0000000      MOV     #BLK0,R2   ;Point to buffer for input string
51 003646 042761 0000000 0000000 BIC     #$ECHO,LSW2(R1) ;Turn off character echoing
52 003654 2$: .TTYIN              ;Get next input character
53 003660 120027 0000001      CNPB     R0,#1     ;Did read time-out?
54 003664 001472      BEQ      20$             ;Br if yes
55 003666 120027 0000015      CNPB     R0,#CR    ;Ignore carriage return
56 003672 001770      BEQ      2$             ;
57 003674 120027 0000012      CNPB     R0,#LF    ;End of input line?

```

GETSYN -- Accept system logon password

```

58 003700 001405          BEQ      3$          ;Br if yes
59 003702 020227 0000620  CMP      R2,#BLK0+50.    ;Input line too long?
60 003706 101362          BHI      2$          ;Br if yes
61 003710 110022          MOV     R0,(R2)+      ;Store received character
62 003712 000760          BR      2$          ;Continue receiving input
63 003714 105012          3$: CLRB    (R2)      ;Store null at end of string
64
65          ; Convert received and expected passwords to upper case
66
67 003716 012703 0000000  MOV     #BLK0,R3      ;Point to received password
68 003722 004767 0000000  CALL    CVTUC      ;Convert to upper case
69 003726 012703 0000000  MOV     #SYPSWD,R3     ;Point to expected password
70 003732 004767 0000000  CALL    CVTUC      ;Convert to upper case
71
72          ; See if received password matches expected password
73
74 003736 012702 0000000  MOV     #BLK0,R2      ;Point to received string
75 003742 012703 0000000  MOV     #SYPSWD,R3     ;Point to expected password
76 003746 122213          4$: CMPB   (R2)+,(R3)    ;Does password match?
77 003750 001003          BNE     5$          ;Br if not
78 003752 105723          TSTB   (R3)+      ;Are we at the end of the password?
79 003754 001374          BNE     4$          ;Br if not
80 003756 000402          BR      6$          ;Got correct password
81
82          ; Invalid password.
83          ; Give one extra try then give up.
84
85 003760 077564          5$: SOB    R5,1$      ;Loop if user gets another try
86 003762 000433          BR      20$         ;Give up
87
88          ; Password successfully entered
89
90 003764 052761 0000000 0000000 6$: BIS    #$ECHO,LSW2(R1) ;Turn echoing back on
91 003772 052761 0000000 0000000  BIS    #$NOIN,LSW3(R1) ;Reset flags for start-up command file
92 004000 052761 0000000 0000000  BIS    #$SUCF,LSW7(R1)
93 004006 042761 0000000 0000000  BIC    #$PRGLK,LSW5(R1) ;Unlock KMON
94 004014 042761 0000000 0000000  BIC    #$SCCA,LSW5(R1) ;Remove control-C suppression
95 004022 016767 0000000 0000000  MOV     PRIVA2,PRIVC2    ;Reset privilege flags
96 004030 004767 0000000  CALL    FIXPRV      ;Transfer privilege flag to LSW tables
97 004034          PRINT   #CRLF      ;Print CR-LF
98
99          ; Finished
100
101 004042 012605          9$: MOV     (SP)+,R5
102 004044 012603          MOV     (SP)+,R3
103 004046 012602          MOV     (SP)+,R2
104 004050 000207          RETURN
105
106          ; Password not successfully entered.
107          ; Log off the job.
108
109 004052 012700 000066' 20$: MOV     #HNGEMT,R0      ;Point to logoff emt
110 004056 104375          EMT      375      ;Log off the job
111
112          .END

```

Errors detected: 0

TSKM2B -- Keyboard DEFINE Comma MACRO V05.04 Monday 21-Dec-87 12:53 Page 29-2

GETSYP -- Accept system logon password

*** Assembler statistics

Work file reads: 0

Work file writes: 0

Size of work file: 10176 Words (40 Pages)

Size of core pool: 17720 Words (70 Pages)

Operating system: RT-11

Elapsed time: 00:00:47.45

DK: TSKM2B, LP: TSKM2B=DK: TSKM2B, MAC/C/N: SYM

\$DOOFF	1-25	29-24										
\$ECHO	1-29	29-51	29-90									
\$NOIN	1-29	29-91										
\$PRGLK	1-27	29-20	29-93									
\$SCCA	1-27	29-31	29-94									
\$SLON	1-25	22-14										
\$SUCF	1-29	29-72										
... V1	13-21	14-20	15-9	15-14	16-9	16-14	17-11	18-17	18-28	18-32	18-34	18-37
	18-39	18-55	18-57	18-62	19-33	20-59	20-59	21-32	21-32	21-32	21-38	21-38
	25-17	25-21	25-26	29-39	29-40	29-97						
... V2	20-59	20-59#	21-32	21-32	21-32#	21-32#	21-38	21-38	21-38#	21-38#		
ACRDEC	1-31	23-8										
ACRSTR	1-34	11-33										
ACRTXT	1-28	7-59										
AMBOPT	1-36	6-29										
BELL	1-51#											
BLANK	1-50#											
BLKO	1-30	1-34	2-34	7-62	11-36	21-38	21-44	29-50	29-59	29-67	29-74	
BLKWDS	1-55#											
CMDDEF	1-21	6-7#										
CMDRCL	1-21	22-10#										
COLSPC	2-50#	25-17										
COMMA	1-54#											
CR	1-49#	29-55										
CRLF	1-30	18-62	25-26	29-39	29-97							
CVTTAB	1-36	6-7	22-28									
CVTUC	1-28	22-29	29-68	29-70								
DEFHD	4-4#	6-20										
DEFKEY	4-5	7-11#										
EM*CSE	1-26	23-11										
EM*KCR	1-33	13-57										
EM*KNS	1-31	7-23										
EM*KNT	1-33	9-16										
EM*KNU	1-38	7-88										
EM*KTF	1-38	9-44										
EM*KWC	1-33	14-32										
EM*NSL	1-32	7-17										
EM*SLO	1-25	22-16										
EM*STL	1-34	11-57										
EMTUKC	2-30#	20-16*	20-17*	20-18	21-20*	21-21						
ERRLOC	1-31	20-21										
FF	1-53#											
FIND20	25-9	26-12#										
FIXPRV	1-27	29-30	29-96									
FKILL	1-36	6-29	6-30	7-17	7-23	7-84	7-88	9-16	9-44	11-57	13-57	14-32
	22-16	22-46	23-11									
GETSYP	1-21	29-7#										
HNCMT	2-44#	29-109										
ILLCMD	1-33	7-14										
INIUKD	1-21	20-10#										
INVOPT	1-26	1-33	6-30	22-46								
KC*COM	1-40	5-30										
KC*DOT	1-40	5-29										
KC*DWN	1-41	5-37										
KC*E1	1-41	5-38	5-44									
KC*E2	1-41	5-39	5-45									

KC\$E3	1-41	5-40	5-46						
KC\$E4	1-41	5-41	5-47						
KC\$E5	1-42	5-42	5-48						
KC\$E6	1-42	5-43	5-49						
KC\$ENT	1-40	5-32							
KC\$F10	1-42	5-55							
KC\$F11	1-42	5-56							
KC\$F12	1-43	5-57							
KC\$F13	1-43	5-58							
KC\$F14	1-43	5-59							
KC\$F15	1-43	5-50	5-60						
KC\$F16	1-43	5-36	5-61						
KC\$F17	1-43	5-62							
KC\$F18	1-43	5-63							
KC\$F19	1-44	5-64							
KC\$F20	1-44	5-65							
KC\$F6	1-42	5-51							
KC\$F7	1-42	5-52							
KC\$F8	1-42	5-53							
KC\$F9	1-42	5-54							
KC\$KP0	1-39	5-9	5-19						
KC\$KP1	1-39	5-10	5-20						
KC\$KP2	1-39	5-11	5-21						
KC\$KP3	1-39	5-12	5-22						
KC\$KP4	1-39	5-13	5-23						
KC\$KP5	1-39	5-14	5-24						
KC\$KP6	1-39	5-15	5-25						
KC\$KP7	1-40	5-16	5-26						
KC\$KP8	1-40	5-17	5-27						
KC\$KP9	1-40	5-18	5-28						
KC\$LFT	1-41	5-33							
KC\$MIN	1-40	5-31							
KC\$PF1	1-44	5-5							
KC\$PF2	1-44	5-6							
KC\$PF3	1-44	5-7							
KC\$PF4	1-44	5-8							
KC\$RIT	1-41	5-34							
KC\$UP	1-41	5-35							
KCODE	2-5#	2-6	7-27*	7-53*	7-62*	7-70*	9-31	9-49	10-18
KD\$SZ	1-38	10-32	12-26	13-14	13-46	14-12	17-35	20-43	
KD\$COD	1-37	9-49*	10-24*	10-30	12-24	13-45*	17-26	18-29	19-18
KD\$FLG	1-37	9-50*							
KD\$TXT	1-37	9-51	18-50						
KD\$TYP	1-32	18-13	18-15	18-24	18-26				
KEYADD	7-80	9-10#							
KEYCOD	18-45	19-9#							
KEYDEL	7-79	10-5#							
KEYELR	10-38	16-5#							
KEYHD	4-11#	7-33							
KEYMAP	13-34	14-5#							
KEYMOV	20-49	21-14#							
KEYMXT	1-34	11-34	11-45						
KEYPAR	1-37	13-30*	16-19*	17-9					
KEYPRT	17-31	18-8#							
KEYRCB	1-32	1-38	9-20	10-9	12-17	13-29*	14-10	16-18*	
KEYREG	9-22	9-27	10-14	13-7#	17-17	20-36			

KEYROP	4-13	4-15	8-12#									
KEYSOP	4-12	4-14	8-5#									
KEYSRC	9-32	9-38	10-19	12-13#								
KEYTXT	7-74	11-11#										
KEYTYP	4-16	4-17	4-18	8-19#								
KEYUMP	9-43	9-58	10-43	15-5#	17-40	20-56						
KF\$ECO	1-36	4-12	4-13	7-29	9-14							
KF\$TRM	1-36	4-14	4-15	7-29	9-14							
KFLAG	2-4#	7-29*	8-5*	8-12*	9-14	9-50						
KNMHD	5-4#	7-67	19-14									
KRDB	2-12#	13-17*	13-17*	13-21	13-23	13-28	13-38	15-13*	15-14	16-13*	16-14	
KT\$GLD	1-36	4-16	8-23	8-30	18-13							
KT\$GLT	1-38	7-46	8-32	18-15	18-26							
KT\$LET	1-37	4-18	7-44	8-25	8-28	18-24						
KT\$NRM	1-37	4-17	7-28									
KTEXT	2-51#	7-78	9-52	11-12*	11-37	11-44	11-45					
KTYPE	2-6#	7-28*	7-44	7-46	8-25	8-30	8-36*					
KWDB	2-19#	14-9*	14-10*	14-15*	14-16*	14-20	14-22	15-9	16-9			
L	4-5	4-5#	4-12	4-12#	4-13	4-13#	4-14	4-14#	4-15	4-15#	4-16	4-16#
	4-17	4-17#	4-18	4-18#	5-5	5-5#	5-6	5-6#	5-7	5-7#	5-8	5-8#
	5-9	5-9#	5-10	5-10#	5-11	5-11#	5-12	5-12#	5-13	5-13#	5-14	5-14#
	5-15	5-15#	5-16	5-16#	5-17	5-17#	5-18	5-18#	5-19	5-19#	5-20	5-20#
	5-21	5-21#	5-22	5-22#	5-23	5-23#	5-24	5-24#	5-25	5-25#	5-26	5-26#
	5-27	5-27#	5-28	5-28#	5-29	5-29#	5-30	5-30#	5-31	5-31#	5-32	5-32#
	5-33	5-33#	5-34	5-34#	5-35	5-35#	5-36	5-36#	5-37	5-37#	5-38	5-38#
	5-39	5-39#	5-40	5-40#	5-41	5-41#	5-42	5-42#	5-43	5-43#	5-44	5-44#
	5-45	5-45#	5-46	5-46#	5-47	5-47#	5-48	5-48#	5-49	5-49#	5-50	5-50#
	5-51	5-51#	5-52	5-52#	5-53	5-53#	5-54	5-54#	5-55	5-55#	5-56	5-56#
	5-57	5-57#	5-58	5-58#	5-59	5-59#	5-60	5-60#	5-61	5-61#	5-62	5-62#
	5-63	5-63#	5-64	5-64#	5-65	5-65#						
LF	1-48#	29-57										
LSW	1-25	29-24										
LSW2	1-28	29-51*	29-90*									
LSW3	1-29	29-91*										
LSW5	1-27	29-20*	29-31*	29-93*	29-94*							
LSW7	1-25	22-14										
LSW9	1-29	29-92*										
NARGS	4-4#	4-5	4-5	4-11#	4-12	4-12	4-13	4-13	4-14	4-14	4-15	4-15
	4-16	4-16	4-17	4-17	4-18	4-18	5-4#	5-5	5-5	5-6	5-6	5-7
	5-7	5-8	5-8	5-9	5-9	5-10	5-10	5-11	5-11	5-12	5-12	5-13
	5-13	5-14	5-14	5-15	5-15	5-16	5-16	5-17	5-17	5-18	5-18	5-19
	5-19	5-20	5-20	5-21	5-21	5-22	5-22	5-23	5-23	5-24	5-24	5-25

R. GID	1-35	13-28							
R. GSIZ	1-30	13-17*							
R. GSTS	1-35	13-12*	13-23	13-38	15-13*	16-13*			
RC\$BAS	1-38	13-30							
RCLALL	22-43	25-4#							
RCLNUM	22-55	23-4#							
RCLPTR	2-8#	22-21*	25-41	26-18					
RCLREV	1-28	25-7	25-30						
RCLSTR	22-52	22-54	24-4#						
RCLVAL	22-36	23-10	23-16#	24-13					
RDCMD	1-37	9-59	10-47	17-12	17-44	22-47	23-27	24-30	25-43
RDTIME	2-38#	29-44							
RS. CGR	1-35	2-14	13-12						
RS. CRR	1-35	13-23							
RS. EGR	1-32	16-13							
RS. GBL	1-35	2-14	13-12	15-13	16-13				
RS. NEW	1-32	13-38							
RS. PVT	1-35	2-14	13-12	15-13	16-13				
RSFBLK	1-31	20-28							
SEARCH	1-36	6-21	7-68						
SHOKEY	1-21	17-5#							
SKPSPC	1-36	6-13	7-48	11-16	11-20				
SLLBUF	1-26	25-24	27-18	27-30	28-18	28-26	28-35		
SLLEND	1-26	25-22	27-20	27-32	28-16	28-24	28-37		
SLLPTR	1-25	22-24*							
SLMVDN	25-34	26-24	27-12#						
SLMVUP	22-22	23-19	24-27	25-39	26-15	28-11#			
SLRPTR	1-25	23-26*							
SLSPTR	1-25	22-20	22-23*	23-16	23-20	24-4	24-28	25-4	
SPACE2	1-26								
SYPSPR	1-29	29-40							
SYPSPD	1-29	29-15	29-69	29-75					
TAB	1-52#								
TM\$GLD	1-30	18-17							
TM\$KND	1-30	17-11							
TSKM2B	1-5#								
VKEYMX	1-38	7-21	10-29	12-23	13-13	13-44	14-11	17-22	20-42
VOFFTM	1-28	29-45							
VPAR1	1-35	10-28	12-22	13-43	17-21	21-43			
VSLEDT	1-32	7-15							
W. NLEN	1-30	14-16*							
W. NRID	1-34	14-10*							
W. NSIZ	1-30	14-15*							
W. NSTS	1-34	14-9*	14-22						
WLDNAM	1-56#								
WS. CRW	1-33	14-22							
WS. MAP	1-34	2-26	14-9						
XAREA	1-35	13-21	14-20	15-9	15-14	16-9	16-14	21-32	21-38

... CM1	21-32	21-38										
... CM2	13-21	14-20	15-9	15-14	16-9	16-14	21-32	21-32	21-38	21-38	21-38	21-38
... CM3	20-59											
... CM5	13-21	14-20	15-9	15-14	16-9	16-14	17-11	18-17	18-28	18-32	18-34	18-37
	18-39	18-55	18-57	18-62	19-33	21-32	21-38	25-17	25-21	25-26	29-39	29-40
	29-97											
... CM6	13-21	14-20	15-9	15-14	16-9	16-14						
... CM7	21-38											
. CLOSE	1-17#	20-59										
. CRAW	1-16#	14-20										
. CRRG	1-16#	13-21										
. ELAW	1-16#	15-9	16-9									
. ELRG	1-16#	15-14	16-14									
. LOOKU	1-17#	21-32										
. PRINT	1-16#	17-11	18-17	18-62	25-17	25-26	29-39	29-40	29-97			
. READW	1-17#	21-38										
. TTYIN	1-17#	29-57										
. TTYOU	1-16#	18-28	18-32	18-34	18-37	18-39	18-55	18-57	19-33	25-21		
CMDDEF	3-47#	4-5	4-12	4-13	4-14	4-15	4-16	4-17	4-18	5-5	5-6	5-7
	5-8	5-9	5-10	5-11	5-12	5-13	5-14	5-15	5-16	5-17	5-18	5-19
	5-20	5-21	5-22	5-23	5-24	5-25	5-26	5-27	5-28	5-29	5-30	5-31
	5-32	5-33	5-34	5-35	5-36	5-37	5-38	5-39	5-40	5-41	5-42	5-43
	5-44	5-45	5-46	5-47	5-48	5-49	5-50	5-51	5-52	5-53	5-54	5-55
	5-56	5-57	5-58	5-59	5-60	5-61	5-62	5-63	5-64	5-65		
FABORT	3-15#	6-29	6-30	7-17	7-23	7-84	7-88	9-16	9-44	11-57	13-57	14-32
	22-16	22-46	23-11									
FERR	3-4#											
FWARN	3-23#											
TBLDEF	3-35#	4-4	4-11	5-4								
TBLEND	3-61#	4-6	4-19	5-66								