TSKM2B -- Keyboard DEFINE Comma MACRO V05.05 Thursday 19-Jan-89 14:53
Table of contents

```
                                        .TITLE  TSKM2B -- Keyboard DEFINE Command routines
                                        .ENABL  LC
                                        .DSABL  GBL
000000                                  .CSECT  TSKM2B
000000                          TSKM2B:
                                ;
                                ;  TSKM2B is the portion of TSKMON that contains the code
                                ;  to implement the DEFINE command.
                                ;
                                ;  Copyright 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .ELRG,.ELAW,.CRRG,.CRAW,.TTYOUT,.PRINT
                                        .MCALL  .LOOKUP,.READW,.CLOSE,.TTYIN
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  CMDDEF,SHOKEY,INIUKD,GETSYP,CMDRCL,CMDACC,CMDNAC
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  OKFAND,OKFNND,TBLOVF,R50BUF,RESDEV,BADACC,PRTWRN
                                        .GLOBL  SPACE,CORUSR,PO$SYS,GTRD50,ASNSRC,AT$DEV,CHKDLM
                                        .GLOBL  R50SY,R50DK,SYINDX,SYUNIT,NUMDEV,PNAME,PRIVCO
                                        .GLOBL  OF$DEV,OF$UNT,OF$FIL,OF$FLG,OF$$SZ,OT$RON
                                        .GLOBL  $DOOFF,LSW,EM$SLO,LSW7,$SLON,SLSPTR,SLLPTR,SLRPTR
                                        .GLOBL  EM$CSE,PRTFIX,SPACE2,SLLEND,SLLBUF,INVOPT
                                        .GLOBL  P2$VIR,FIXPRV,$SCCA,LSW5,PRIVA2,PRIVC2,$PRGLK
                                        .GLOBL  LSW2,CVTUC,VONTM,ACRTXT,RCLREV,$KINIT
                                        .GLOBL  SYPSWD,SYPSPR,$ECHO,$NOIN,LSW3,$SUCF,LSW9
                                        .GLOBL  TM$KND,TM$GLD,CRLF,R.GSIZ,W.NSIZ,W.NLEN,BLKO
                                        .GLOBL  RSFBLK,ERRLOC,EM$KNS,ACRDEC,EM$NSF
                                        .GLOBL  VSLEDT,EM$NSL,RS.NEW,RS.EGR,KEYRCB,PRTSPC,KD$TYP
                                        .GLOBL  INVOPT,ILLCMD,EM$KCR,EM$KWC,WS.CRW,EM$KNT
                                        .GLOBL  BLKO,ACRSTR,KEYMXT,EM$STL,W.NRID,W.NSTS,WS.MAP
                                        .GLOBL  R.GID,R.GSTS,RS.CRR,RS.GBL,RS.CGR,RS.PVT,XAREA,VPAR1
                                        .GLOBL  KF$ECO,KF$TRM,KT$GLD,CVTTAB,SKPSPC,SEARCH,AMBOPT,FKILL
                                        .GLOBL  KT$NRM,OPTLST,KT$LET,KD$COD,KD$FLG,KD$TXT,RDCMD,KEYPAR
                                        .GLOBL  VKEYMX,KD$$SZ,EM$KTF,KT$GLT,EM$KNU,KEYRCB,RC$BAS
                                        .GLOBL  KC$KP0,KC$KP1,KC$KP2,KC$KP3,KC$KP4,KC$KP5,KC$KP6
                                        .GLOBL  KC$KP7,KC$KP8,KC$KP9,KC$DOT,KC$COM,KC$MIN,KC$ENT
                                        .GLOBL  KC$UP,KC$DWN,KC$LFT,KC$RIT,KC$E1,KC$E2,KC$E3,KC$E4
                                        .GLOBL  KC$E5,KC$E6,KC$F6,KC$F7,KC$F8,KC$F9,KC$F10,KC$F11
                                        .GLOBL  KC$F12,KC$F13,KC$F14,KC$F15,KC$F16,KC$F17,KC$F18
                                        .GLOBL  KC$F19,KC$F20,KC$PF1,KC$PF2,KC$PF3,KC$PF4
                                ;
                                ;  Assembly constants
                                ;
000012                          LF      =       12              ;LINE FEED
000015                          CR      =       15              ;CARRIAGE RETURN
000040                          BLANK   =       40              ;ASCII SPACE
000007                          BELL    =       07              ;ASCII BELL
000011                          TAB     =       11              ;HORIZONTAL TAB
000014                          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 #)
```

```
                                ;*******************************************************************************
                                ;  Data areas
                                ;
000000  000000                KFLAG:  .WORD   0               ;Flags set during parsing
000002  000000                KCODE:  .WORD   0               ;Type and code value
        000003'               KTYPE   =       KCODE+1         ;Key type code is in upper byte of KCODE
000004                        PSFNAM: .BLKW   4               ;Local copy of PLAS region swap file name
000014  000000                RCLPTR: .WORD   0               ;Ptr to save area for RECALL command
                                ;
                                ;  Region definition block for key definition region
                                ;
000016  000000                KRDB:   .WORD   0                       ;Will get addr of region control blk
000020  000000                        .WORD   0                       ;# 64-byte blocks needed for region
000022  0000000C                      .WORD   RS.GBL!RS.CGR!RS.PVT    ;Status flags
000024  042641  014716                .RAD50  /KEYDEF/                ;Region name
                                ;
                                ;  Window definition block used to map PAR 1 to key definition region
                                ;
000030     000                KWDB:   .BYTE   0                       ;Will get window ID
000031     001                        .BYTE   1                       ;Select PAR 1
000032  000000                        .WORD   0                       ;Will get base virtual address
000034  000000                        .WORD   0                       ;# 64-byte blocks for window
000036  000000                        .WORD   0                       ;Addr of region control block
000040  000000                        .WORD   0                       ;Offset into region of window base
000042  000000                        .WORD   0                       ;# 64-byte blocks to map
000044  0000006                       .WORD   WS.MAP                  ;Status flags
                                ;
                                ;  Emt arg block to copy key definition data from our parent job
                                ;
000046     000      126        EMTUKC: .BYTE   0,126
000050  000017                        .WORD   17
000052  000000                        .WORD   0               ;Parent job number
000054  000000                        .WORD   0               ;Page number of data
000056  0000000                       .WORD   BLKO            ;Address of destination buffer
                                ;
                                ;  Emt arg block to set a TT read timeout
                                ;
000060     000      117        RDTIME: .BYTE   0,117
000062  000000                        .WORD   0               ;Timeout value (0.5 second units)
000064  000001                        .WORD   1               ;Timeout signal character
                                ;
                                ;  Emt arg block to log off and drop DTR after short time
                                ;
000066     000      126        HNGEMT: .BYTE   0,126
000070  000002                        .WORD   2
000072  000001                        .WORD   1               ;Time before DTR dropped
                                ;
                                ;  Byte data
                                ;
000074     072      040   200  COLSPC: .ASCIZ  /: /<200>
000077     000
000100                        KTEXT:  .BLKB   80.             ;Key definition text string
                                        .EVEN
```

```
                        ;-----------------------------------------------------
                        ;  Macro to cause a fatal error message to be printed.
                        ;
                              .MACRO  FERR    MSG
                              MOV     R5,-(SP)
                              MOV     MSG,R5
                              CALL    FPRINT
                              MOV     (SP)+,R5
                              .ENDM   FERR
                        ;
                        ;-----------------------------------------------------
                        ;  Macro to print a fatal error message, clean up
                        ;  and then jump to RDCMD.
                        ;
                              .MACRO  FABORT  MSG
                              MOV     MSG,R5
                              JMP     FKILL
                              .ENDM   FABORT
                        ;
                        ;-----------------------------------------------------------
                        ;  Macro to print a warning message
                        ;
                              .MACRO  FWARN   MSG
                              MOV     R5,-(SP)
                              MOV     MSG,R5
                              CALL    PRTWRN
                              MOV     (SP)+,R5
                              .ENDM   FWARN
                        ;
                        ;-----------------------------------------------------------
                        ;  Macro to start a standard option table.
                        ;  Name = 1 to 4 character table name.
                        ;  NA = Number of arguments per table entry.
                        ;
                              .MACRO  TBLDEF  NAME,NA
                        NARGS =       NA
                              .CSECT  CMDV2B
                        NAME'HD: .WORD  2*NA
                              .ENDM   TBLDEF
                        ;
                        ;-----------------------------------------------------------
                        ;  Macro to enter an option text name and a set of parameters
                        ;  into the currently open table.
                        ;  STRNG = Ascii name
                        ;  A,B,C = Set of option parameters to store in table with name.
                        ;
                              .MACRO  CMDDEF  STRNG,A,B,C
                              .CSECT  NAME2B
                        L     =       .
                              .ASCIZ  /STRNG/
                              .CSECT  CMDV2B
                              .WORD   L               ;POINTER TO NAME STRING
                              .WORD   A
                        .IIF  GE,<NARGS-2>    .WORD   B
                        .IIF  GE,<NARGS-3>    .WORD   C
                              .ENDM   CMDDEF
                        ;
```

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

```
                                ;------------------------------------------------------------------------
                                ;  Define initial qualifiers for the DEFINE command.
                                ;
000220                                TBLDEF   DEF,1
000002                                CMDDEF   KEY,DEFKEY
000006                                TBLEND
                                ;------------------------------------------------------------------------
                                ;  Define qualifiers for the DEFINE/KEY command.
                                ;
000220                                TBLDEF   KEY,2
000012                                CMDDEF   ECHO,KEYSOP,KF$ECO
000020                                CMDDEF   NOE*CHO,KEYROP,KF$ECO
000026                                CMDDEF   TER*MINATE,KEYSOP,KF$TRM
000034                                CMDDEF   NOT*ERMINATE,KEYROP,KF$TRM
000042                                CMDDEF   GO*LD,KEYTYP,KT$GLD
000050                                CMDDEF   NOG*OLD,KEYTYP,KT$NRM
000056                                CMDDEF   LET*TER,KEYTYP,KT$LET
000064                                TBLEND
```

```
                        ;-----------------------------------------------------------------------
                        ;  Define names of keys and associated code values.
                        ;
000220                          TBLDEF  KNM,1
000070                          CMDDEF  PF1,KC$PF1
000074                          CMDDEF  PF2,KC$PF2
000100                          CMDDEF  PF3,KC$PF3
000104                          CMDDEF  PF4,KC$PF4
000110                          CMDDEF  KP0,KC$KP0
000114                          CMDDEF  KP1,KC$KP1
000120                          CMDDEF  KP2,KC$KP2
000124                          CMDDEF  KP3,KC$KP3
000130                          CMDDEF  KP4,KC$KP4
000134                          CMDDEF  KP5,KC$KP5
000140                          CMDDEF  KP6,KC$KP6
000144                          CMDDEF  KP7,KC$KP7
000150                          CMDDEF  KP8,KC$KP8
000154                          CMDDEF  KP9,KC$KP9
000160                          CMDDEF  0,KC$KP0
000164                          CMDDEF  1,KC$KP1
000170                          CMDDEF  2,KC$KP2
000174                          CMDDEF  3,KC$KP3
000200                          CMDDEF  4,KC$KP4
000204                          CMDDEF  5,KC$KP5
000210                          CMDDEF  6,KC$KP6
000214                          CMDDEF  7,KC$KP7
000220                          CMDDEF  8,KC$KP8
000224                          CMDDEF  9,KC$KP9
000230                          CMDDEF  PER*IOD,KC$DOT
000234                          CMDDEF  COM*MA,KC$COM
000240                          CMDDEF  MIN*US,KC$MIN
000244                          CMDDEF  ENT*ER,KC$ENT
000250                          CMDDEF  LEFT,KC$LFT
000254                          CMDDEF  RIGHT,KC$RIT
000260                          CMDDEF  UP,KC$UP
000264                          CMDDEF  DO,KC$F16
000270                          CMDDEF  DOWN,KC$DWN
000274                          CMDDEF  FIND,KC$E1
000300                          CMDDEF  <INS*ERT-HERE>,KC$E2
000304                          CMDDEF  REM*OVE,KC$E3
000310                          CMDDEF  SEL*ECT,KC$E4
000314                          CMDDEF  <PREV*-SCREEN>,KC$E5
000320                          CMDDEF  <NEXT*-SCREEN>,KC$E6
000324                          CMDDEF  E1,KC$E1
000330                          CMDDEF  E2,KC$E2
000334                          CMDDEF  E3,KC$E3
000340                          CMDDEF  E4,KC$E4
000344                          CMDDEF  E5,KC$E5
000350                          CMDDEF  E6,KC$E6
000354                          CMDDEF  HELP,KC$F15
000360                          CMDDEF  F6,KC$F6
000364                          CMDDEF  F7,KC$F7
000370                          CMDDEF  F8,KC$F8
000374                          CMDDEF  F9,KC$F9
000400                          CMDDEF  F10,KC$F10
000404                          CMDDEF  F11,KC$F11
000410                          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
```

```
                                        .SBTTL  DEFINE command
                                ;-----------------------------------------------------------------------
                                ;  Process the DEFINE command.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;   R3 = Pointer to start of qualifiers for DEFINE command.
                                ;
 000220  004767  0000006        CMDDEF: CALL    CVTTAB          ;Convert tab and FF chars to spaces
                                ;
                                ;  If 1st character is slash, skip it
                                ;
 000224  004767  0000006                CALL    SKPSPC          ;Skip over any spaces
 000230  121327  000057                 CMPB    (R3),#'/        ;Is 1st character slash?
 000234  001001                         BNE     1$              ;Br if not
 000236  005203                         INC     R3              ;Point beyond slash
                                ;
                                ;  Branch off to major processing routines based on first qualifier
                                ;
 000240  012704  000000'        1$:     MOV     #DEFHD,R4       ;Point to table of options
 000244  004767  0000006                CALL    SEARCH          ;Try to find correct processing routine
 000250  103401                         BCS     2$              ;Br if don't recognize option keyword
 000252  000134                         JMP     @(R4)+          ;Enter major processing routine
                                ;
                                ;  Invalid keyword
                                ;
 000254  005704                 2$:     TST     R4              ;Ambiguous or unrecognized option?
 000256  001404                         BEQ     3$              ;Br if unrecognized
 000260                                 FABORT  #AMBOPT         ;Ambigious option
 000270                         3$:     FABORT  #INVOPT         ;Invalid option
```

```
                                        .SBTTL  DEFINE/KEY
                                ;-------------------------------------------------------------------
                                ;  Define a character string which will be substituted for some
                                ;  terminal key.
                                ;
                                ;  Command format:  DEFINE/KEY  key string
                                ;
                                ;  Inputs:
                                ;   R3 = Points past "/KEY" option.
                                ;
000300                          DEFKEY:
                                ;
                                ;  Make sure Single line editor is genned into system
                                ;
000300  105767  0000000                 TSTB    VSLEDT          ;Is SL available?
000304  001004                         BNE     4$              ;Br if yes
000306                                  FABORT  #EM$NSL         ;SL is not available
                                ;
                                ;  See if any user-defined keys are allowed
                                ;
000316  005767  0000000         4$:     TST     VKEYMX          ;Are any user-defined keys allowed?
000322  001004                         BNE     5$              ;Br if yes
000324                                  FABORT  #EM$KNS         ;User-defined keys not supported
                                ;
                                ;  Initialize some values
                                ;
000334  105067  177442          5$:     CLRB    KCODE           ;No key code yet
000340  112767  0000000G 177435         MOVB    #KT$NRM,KTYPE   ;Default to normal key type
000346  112767  0000000C 177424         MOVB    #KF$TRM!KF$ECO,KFLAG ;Initialize flags
                                ;
                                ;  Process any options
                                ;
000354  012704  000010'                 MOV     #KEYHD,R4       ;Point to option list
000360  004767  0000000G                CALL    OPTLST          ;Process the options
                                ;
                                ;  R3 should now be pointing to the key name
                                ;
000364  105713                          TSTB    (R3)            ;Was something specified?
000366  001456                         BEQ     20$             ;Br if not
                                ;
                                ;  If the /LETTER or /GOLDLETTER qualifiers were specified, the key
                                ;  is a single letter.
                                ;
000370  126727  177407  0000000G         CMPB    KTYPE,#KT$LET   ;Key type = letter?
000376  001404                         BEQ     1$              ;Br if yes
000400  126727  177377  0000000G         CMPB    KTYPE,#KT$GLT   ;Key type = goldletter?
000406  001031                         BNE     3$              ;Br if not
000410  004767  0000000G          1$:    CALL    SKPSPC          ;Skip up to start of string
000414  121327  000047                 CMPB    (R3),#47        ;Is letter enclosed in quotes?
000420  001413                         BEQ     6$              ;Br if yes
000422  121327  000042                 CMPB    (R3),#42
000426  001410                         BEQ     6$              ;Br if yes
000430  112367  177346                 MOVB    (R3)+,KCODE     ;Get the letter
000434  111300                         MOVB    (R3),R0         ;Get following letter
000436  001424                         BEQ     2$              ;Br if null
000440  120027  000040                 CMPB    R0,#'           ;Blank?
000444  001421                         BEQ     2$              ;Br if yes
```

```
000446  000432                          BR      21$             ;Not single letter
000450  004767  000000G         6$:     CALL    ACRTXT          ;Accrue the letter
000454  020027  000001                  CMP     R0,#1           ;Should have gotten exactly 1 char
000460  001025                          BNE     21$             ;Br if not
000462  116767  000000G 177312          MOVB    BLK0,KCODE      ;Save the letter as key code
000470  000407                          BR      2$
                                ;
                                ;  Convert key name into key code
                                ;
000472  012704  000066'         3$:     MOV     #KNMHD,R4       ;Point to key name table
000476  004767  000000G                 CALL    SEARCH          ;Try to translate key name
000502  103414                          BCS     21$             ;Br if can't recognize key name
000504  111467  177272                  MOVB    (R4),KCODE      ;Set code value for key
                                ;
                                ;  Accrue the associated text string and store into KTEXT
                                ;
000510  004767  000346          2$:     CALL    KEYTXT          ;Accrue text string
                                ;
                                ;  If the associated text string is null, we are deleting the definition
                                ;
000514  105767  177360                  TSTB    KTEXT           ;Is string null?
000520  001523                          BEQ     KEYDEL          ;Delete this key definition
000522  000443                          BR      KEYADD          ;Add this key definition
                                ;
                                ;  Invalid command syntax
                                ;
000524                          20$:     FABORT  #ILLCMD         ;Invalid command
                                ;
                                ;  Unrecognized key name
                                ;
000534                          21$:     FABORT  #EM$KNU         ;Unrecognized key name
```

```
                                        .SBTTL  KEYSOP -- Set key definition option flags
                                ;-------------------------------------------------------------------------
                                ;  Set some key definition option flag.
                                ;
000544  151467  177230          KEYSOP: BISB    (R4),KFLAG      ;Set option flag
000550  000207                          RETURN

                                        .SBTTL  KEYROP -- Reset key definition option flags
                                ;-------------------------------------------------------------------------
                                ;  Reset (turn off) some key definition option.
                                ;
000552  141467  177222          KEYROP: BICB    (R4),KFLAG      ;Reset option flag
000556  000207                          RETURN

                                        .SBTTL  KEYTYP -- Set key definition type
                                ;-------------------------------------------------------------------------
                                ;  Set type of key being defined (Normal, Gold, etc.)
                                ;
000560  111400                  KEYTYP: MOVB    (R4),R0         ;Get specified type
                                ;
                                ;  Specially handle /GOLD with /LETTER
                                ;
000562  120027  0000000                 CMPB    R0,#KT$GLD      ;Is type /GOLD?
000566  001005                         BNE     1$              ;Br if not
000570  126727  177207  0000000         CMPB    KTYPE,#KT$LET   ;Is current type /LETTER?
000576  001012                         BNE     2$              ;Br if not
000600  000407                         BR      3$
000602  120027  0000000         1$:     CMPB    R0,#KT$LET      ;Is type /LETTER?
000606  001006                         BNE     2$              ;Br if not
000610  126727  177167  0000000         CMPB    KTYPE,#KT$GLD   ;Is current type /GOLD?
000616  001002                         BNE     2$              ;Br if not
000620  112700  0000000         3$:     MOVB    #KT$GLT,R0      ;Set type to gold-letter
                                ;
                                ;  Set new key type
                                ;
000624  110067  177153          2$:     MOVB    R0,KTYPE        ;Set key type
                                ;
                                ;  Finished
                                ;
000630  000207                          RETURN
```

```
                                        .SBTTL  KEYADD -- Add a new key definition
                                ;-----------------------------------------------------------------------
                                ;  Add a new key definition.
                                ;
                                ;  Inputs:
                                ;   KCODE = Key character code (KC$xxx).
                                ;   KTYPE = Key type code (KT$xxx).
                                ;   KTEXT = Asciz string to be defined for key.
                                ;
000632                          KEYADD:
                                ;
                                ;  Don't allow /NOECHO to be specified with /NOTERMINATE
                                ;
000632  032767  000000C 177140          BIT     #KF$ECO!KF$TRM,KFLAG ;Either ECHO or TERMINATE specified?
000640  001004                          BNE     5$                  ;Br if yes
000642                                  FABORT  #EM$KNT             ;Can't have both NOECHO and NOTERMINATE
                                ;
                                ;  Create a PLAS region for key definitions if we don't already have one
                                ;
000652  005767  0000000         5$:     TST     KEYRCB              ;Do we have a key region?
000656  001003                          BNE     1$                  ;Br if yes
000660  004767  000374                  CALL    KEYREG              ;Create PLAS region and map to it
000664  000402                          BR      4$
                                ;
                                ;  Set up PAR 1 to map to the key region
                                ;
000666  004767  000366          1$:     CALL    KEYREG              ;Map a par to the key region
                                ;
                                ;  Try to find an existing definition for this key
                                ;
000672  016700  177104          4$:     MOV     KCODE,R0            ;Get key code
000676  004767  000310                  CALL    KEYSRC              ;Try to find existing definition
000702  103012                          BCC     2$                  ;Br if found existing entry
                                ;
                                ;  Try to find a free entry
                                ;
000704  005000                          CLR     R0                  ;Look for free entry
000706  004767  000300                  CALL    KEYSRC
000712  103006                          BCC     2$                  ;Br if found free entry
                                ;
                                ;  Error -- No free entries
                                ;
000714  004767  000672                  CALL    KEYUMP              ;Unmap par
000720                                  FABORT  #EM$KTF             ;Key table full
                                ;
                                ;  The entry to use is pointed to by R2.
                                ;  Make the entry.
                                ;
000730  016762  177046  0000000 2$:     MOV     KCODE,KD$COD(R2);Save key code
000736  116762  177036  0000000         MOVB    KFLAG,KD$FLG(R2);Save option flags
000744  062702  0000000                 ADD     #KD$TXT,R2          ;Point to area for text string
000750  012703  000100'                 MOV     #KTEXT,R3
000754  112322                  3$:     MOVB    (R3)+,(R2)+         ;Store the string
000756  001376                          BNE     3$
                                ;
                                ;  Finished
                                ;
```

```
     58 000760  004767  000626                   CALL    KEYUMP          ;Unmap par
     59 000764  000167  0000000                  JMP     RDCMD
```

```
                                        .SBTTL  KEYDEL -- Delete a key definition
                                ;-----------------------------------------------------------------------
                                ;  Delete a key definition.
                                ;
000770                          KEYDEL:
                                ;
                                ;  See if there is any key region now.
                                ;
000770  005767  000000G                TST     KEYRCB          ;Any key definitions now?
000774  001430                        BEQ     9$              ;Br if not
                                ;
                                ;  Set up par to map to the key region
                                ;
000776  004767  000256                CALL    KEYREG          ;Map a par to the key region
                                ;
                                ;  Try to find entry for specified key
                                ;
001002  016700  176774                MOV     KCODE,R0        ;Get key code
001006  004767  000200                CALL    KEYSRC          ;Try to find existing key definition
001012  103417                        BCS     8$              ;Br if key not defined
                                ;
                                ;  We found the key definition.  Mark it as free.
                                ;
001014  005062  000000G               CLR     KD$COD(R2)      ;Say this key no longer defined
                                ;
                                ;  See if there are any remaining key definitions
                                ;
001020  012702  000000G               MOV     #VPAR1,R2       ;Point to 1st key definition entry
001024  016703  000000G               MOV     VKEYMX,R3       ;Get max # key entries
001030  005762  000000G        1$:    TST     KD$COD(R2)      ;Is this key entry used?
001034  001006                        BNE     8$              ;Br if yes
001036  062702  000000G               ADD     #KD$$SZ,R2      ;Point to next key entry
001042  077306                        SOB     R3,1$           ;Loop if more to check
                                ;
                                ;  There are no remaining key definitions.
                                ;  Delete the key region.
                                ;
001044  004767  000612                CALL    KEYELR          ;Eliminate the key definition region
001050  000402                        BR      9$
                                ;
                                ;  Unmap the par
                                ;
001052  004767  000534          8$:   CALL    KEYUMP          ;Unmap the par
                                ;
                                ;  Finished
                                ;
001056  000167  000000G         9$:   JMP     RDCMD
```

```
                                        .SBTTL  KEYTXT -- Accrue key definition string
                                ;-----------------------------------------------------------------------
                                ;  Accrue the text string which comprises a key definition.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to start of text string
                                ;
                                ;  Outputs:
                                ;   KTEXT = String in asciz form.
                                ;
001062  010246                  KEYTXT: MOV     R2,-(SP)
001064  105067  177010                  CLRB    KTEXT           ;Initially say no string accrued
                                ;
                                ;  Skip up to start of string
                                ;
001070  004767  000000G                 CALL    SKPSPC          ;Skip over any spaces
001074  121327  000075                  CMPB    (R3),#'=        ;Was equal sign specified before string?
001100  001003                          BNE     1$              ;Br if not
001102  005203                          INC     R3              ;Skip past equal sign
001104  004767  000000G                 CALL    SKPSPC          ;Skip up to string start
                                ;
                                ;  See if the string is enclosed in quote marks
                                ;
001110  111300                  1$:     MOVB    (R3),R0         ;Get 1st char of string
001112  001431                          BEQ     9$              ;Br if no string specified
001114  120027  000047                  CMPB    R0,#47          ;Single quote?
001120  001403                          BEQ     2$              ;Br if yes
001122  120027  000042                  CMPB    R0,#42          ;Double quote?
001126  001014                          BNE     3$              ;Br if not
                                ;
                                ;  Accrue a string that is enclosed in quotes
                                ;
001130  004767  000000G         2$:     CALL    ACRSTR          ;Accrue the string
001134  020027  177777G                 CMP     R0,#KEYMXT-1    ;Is string too long?
001140  101020                          BHI     5$              ;Br if yes
001142  012700  000000G                 MOV     #BLK0,R0        ;Point to buffer with accrued string
001146  012702  000100'                 MOV     #KTEXT,R2       ;Point to buffer where we want result
001152  112022                  4$:     MOVB    (R0)+,(R2)+     ;Move the string
001154  001376                          BNE     4$              ;Loop till null moved
001156  000407                          BR      9$
                                ;
                                ;  Accrue a string that is not enclosed in quotes
                                ;
001160  012702  000100'         3$:     MOV     #KTEXT,R2       ;Point to result area
001164  020227  000000C         6$:     CMP     R2,#KTEXT+KEYMXT;Is string too long?
001170  101004                          BHI     5$              ;Br if yes
001172  112322                          MOVB    (R3)+,(R2)+     ;Move a character to buffer
001174  001373                          BNE     6$              ;Loop if more to move
                                ;
                                ;  Finished
                                ;
001176  012602                  9$:     MOV     (SP)+,R2
001200  000207                          RETURN
                                ;
                                ;  String is too long
                                ;
001202                          5$:     FABORT  #EM$STL         ;String too long
```

```
                                        .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.
                                ;
001212  010346                  KEYSRC: MOV     R3,-(SP)
                                ;
                                ;  See if there are any defined keys
                                ;
001214  005767  0000000                 TST     KEYRCB          ;Are there any defined keys?
001220  001412                          BEQ     7$              ;Br if not
                                ;
                                ;  Begin loop to search for specified key
                                ;
001222  012702  0000000                 MOV     #VPAR1,R2       ;Point to 1st key definition block
001226  016703  0000000                 MOV     VKEYMX,R3       ;Get # of key entries
001232  020062  0000000         1$:     CMP     R0,KD$COD(R2)   ;Is this the one we want?
001236  001405                          BEQ     2$              ;Br if yes
001240  062702  0000000                 ADD     #KD$$SZ,R2      ;Point to next key def block
001244  077306                          SOB     R3,1$           ;Loop if more to check
                                ;
                                ;  Key is not currently defined
                                ;
001246  000261                  7$:     SEC                     ;Signal failure on return
001250  000401                          BR      9$
                                ;
                                ;  We found the key definition
                                ;
001252  000241                  2$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
001254  012603                  9$:     MOV     (SP)+,R3
001256  000207                          RETURN
```

```
                                        .SBTTL  KEYREG -- Create PLAS region for key definitions
                                ;-----------------------------------------------------------------------
                                ;  Try to associate with an existing key definition region.
                                ;  Create a new region if one does not already exist.
                                ;  Mapping is set up to access the key region.
                                ;
001260  010246                  KEYREG: MOV     R2,-(SP)
001262  010346                          MOV     R3,-(SP)
                                ;
                                ;  Set up region definition block for region creation
                                ;
001264  012767  000000C 000001C        MOV     #<RS.GBL!RS.CGR!RS.PVT>,KRDB+R.GSTS ;Init status flags
001272  016703  0000000G               MOV     VKEYMX,R3       ;Get max # key definitions
001276  070327  0000000G               MUL     #KD$$SZ,R3      ;Time size of each entry
001302  062703  000077                 ADD     #63.,R3         ;Round up to next block size
001306  072327  177772                 ASH     #-6.,R3         ;Convert to # 64-byte blocks
001312  010367  000001C                MOV     R3,KRDB+R.GSIZ  ;Set size of region
                                ;
                                ;  Try to create the region
                                ;
001316                                 .CRRG   #XAREA,#KRDB    ;Try to create the region
001336  103435                         BCS     10$             ;Br if cannot create region
001340  032767  0000000G 000001C      BIT     #RS.CRR,KRDB+R.GSTS ;Was region successfully created?
001346  001431                         BEQ     10$             ;Br if not
                                ;
                                ;  Save information from the region definition block
                                ;
001350  016700  000001C                MOV     KRDB+R.GID,R0   ;Get address of region control block
001354  010067  0000000G               MOV     R0,KEYRCB       ;Save address of region control block
001360  016067  0000000G 0000000G      MOV     RC$BAS(R0),KEYPAR;Save mapping value to access region
                                ;
                                ;  Set up mapping to access the region
                                ;
001366  004767  000104                 CALL    KEYMAP          ;Map PAR 1 to the region
                                ;
                                ;  If we just created a new region, initialize it.
                                ;
001372  032767  0000000G 000001C      BIT     #RS.NEW,KRDB+R.GSTS ;Did we just create the region?
001400  001411                         BEQ     9$              ;Br if not
                                ;
                                ;  Say no key definitions exist in region
                                ;
001402  012702  0000000G               MOV     #VPAR1,R2       ;Get pointer to 1st entry
001406  016700  0000000G               MOV     VKEYMX,R0       ;Get # entries
001412  005062  0000000G       1$:     CLR     KD$COD(R2)      ;Say this entry is free
001416  062702  0000000G               ADD     #KD$$SZ,R2      ;Point to next entry
001422  077005                         SOB     R0,1$           ;Mark all entries as free
                                ;
                                ;  Finished
                                ;
001424  012603                 9$:     MOV     (SP)+,R3
001426  012602                         MOV     (SP)+,R2
001430  000207                         RETURN
                                ;
                                ;  Error: Unable to create region
                                ;  If we are trying to init a virtual line ($KINIT not set yet),
                                ;  just print a warning.
```

```
      58                                        ;
      59 001432  116702  0000006               10$:    MOVB    CORUSR,R2          ;Get our job index
      60 001436  032762  0000006 0000006               BIT     #$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
```

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

```
                                        .SBTTL  KEYUMP -- Unmap the key definition region
                                ;-----------------------------------------------------------------------
                                ;  Eliminate the window used to access the key definition region.
                                ;
001612                          KEYUMP:
                                ;
                                ;  Eliminate the window
                                ;
001612                              .ELAW   #XAREA,#KWDB    ;Eliminate the window
                                ;
                                ;  Disassociate from the region but don't delete it
                                ;
001632  012767  000000C 000010      MOV     #RS.GBL!RS.PVT,KRDB+R.GSTS ;Set status flags
001640                              .ELRG   #XAREA,#KRDB    ;Disassociate the region
                                ;
                                ;  Finished
                                ;
001660  000207                      RETURN
```

```
                                        .SBTTL  KEYELR -- Eliminate the key definition region
                                ;-----------------------------------------------------------------------
                                ;  Eliminate the PLAS region that is used to store key definitions.
                                ;
001662                          KEYELR:
                                ;
                                ;  Eliminate the window
                                ;
001662                              .ELAW   #XAREA,#KWDB    ;Eliminate the window
                                ;
                                ;  Eliminate the region
                                ;
001702  012767  000000C 000001C     MOV     #RS.GBL!RS.PVT!RS.EGR,KRDB+R.GSTS ;Set status flags
001710                              .ELRG   #XAREA,#KRDB    ;Eliminate the region
                                ;
                                ;  Say region is gone
                                ;
001730  005067  0000000             CLR     KEYRCB          ;Region is gone
001734  005067  0000000             CLR     KEYPAR
                                ;
                                ;  Finished
                                ;
001740  000207                      RETURN
                                ;
```

```
                                .SBTTL  SHOW KEYS
                        ;-------------------------------------------------------------------
                        ;  Process the SHOW KEYS command.
                        ;
001742                  SHOKEY:
                        ;
                        ;  See if there are any defined keys
                        ;
001742  005767  000000G         TST     KEYPAR          ;Are there any defined keys?
001746  001005                 BNE     1$              ;Br if yes
001750                          .PRINT  #TM$KND         ;No defined keys
001756  000167  000000G         JMP     RDCMD
                        ;
                        ;  There are some defined keys.
                        ;  Associated the key region.
                        ;
001762  004767  177272  1$:     CALL    KEYREG          ;Associate the key region
                        ;
                        ;  Begin loop to print information for each defined key
                        ;
001766  012702  000000G         MOV     #VPAR1,R2       ;Get address of first key entry
001772  016703  000000G         MOV     VKEYMX,R3       ;Get # of key entries
                        ;
                        ;  See if this key entry is defined
                        ;
001776  005762  000000G 3$:     TST     KD$COD(R2)      ;Is this key entry defined?
002002  001402                 BEQ     2$              ;Br if not
                        ;
                        ;  Print information for this key entry
                        ;
002004  004767  000016          CALL    KEYPRT          ;Print info for the key definition
                        ;
                        ;  See if there are more key entries
                        ;
002010  062702  000000G 2$:     ADD     #KD$$SZ,R2      ;Point to next key entry
002014  077310                 SOB     R3,3$           ;Br if more to check
                        ;
                        ;  Disassociate the key region
                        ;
002016  004767  177570          CALL    KEYUMP          ;Disassociate key region
                        ;
                        ;  Finished
                        ;
002022  000167  000000G         JMP     RDCMD
```

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

```
     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
```

```
                                        .SBTTL  KEYCOD -- Print key name based on code
                                ;-----------------------------------------------------------------------
                                ;  Print the name of a key based on the key code in the current key
                                ;  descriptor block.  The key name is printed in a 12 character field.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to key descriptor block
                                ;
002300  010346                  KEYCOD: MOV     R3,-(SP)
002302  010546                          MOV     R5,-(SP)
                                ;
                                ;  Enter loop to find the entry for this key code
                                ;
002304  012705  000070'                 MOV     #KNMHD+2,R5     ;Point to first entry in table
                                ;
                                ;  See if this is the entry we want
                                ;
002310  126265  000000G 000002  1$:     CMPB    KD$COD(R2),2(R5);Is this the entry we want?
002316  001405                          BEQ     2$              ;Br if yes
002320  062705  000004                  ADD     #4.,R5          ;Point to next entry
002324  005715                          TST     (R5)            ;Is there another entry
002326  001370                          BNE     1$              ;Br if yes
002330  000416                          BR      9$              ;Should never happen
                                ;
                                ;  We found the entry.  Print the key name.
                                ;
002332  011505                  2$:     MOV     (R5),R5         ;Get pointer to key name string
002334  012703  000014                  MOV     #12.,R3         ;Get field size
002340  112500                  3$:     MOVB    (R5)+,R0        ;Get next character from name
002342  001407                          BEQ     4$              ;Br if hit end of name
002344  120027  000052                  CMPB    R0,#'*          ;Is it an asterisk?
002350  001773                          BEQ     3$              ;Skip them
002352                                  .TTYOUT                 ;Print a character
002356  005303                          DEC     R3              ;Say one less fill character needed
002360  000767                          BR      3$              ;Keep printing
                                ;
                                ;  Reached end of name.  Blank fill to end of field.
                                ;
002362  004767  0000000G        4$:     CALL    PRTSPC          ;Blank fill rest of field
                                ;
                                ;  Finished
                                ;
002366  012605                  9$:     MOV     (SP)+,R5
002370  012603                          MOV     (SP)+,R3
002372  000207                          RETURN
```

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

```
     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
```

```
                                        .SBTTL  KEYMOV -- Move key definition data from parent job
                                ;-------------------------------------------------------------------
                                ;  Move up to one block of key definition data from our parent job
                                ;  to our key definition region.
                                ;
                                ;  Inputs:
                                ;   R2 = Number of page to be moved.
                                ;   R3 = Number of words to move (256 max)
                                ;   R4 = 0==>Channel not opened to PLAS swap file; 1==>Channel open.
                                ;
                                ;  Outputs:
                                ;   R4 = 1==> channel opened to plas swap file
                                ;
002550  010246                  KEYMOV: MOV     R2,-(SP)
002552  010346                          MOV     R3,-(SP)
002554  010546                          MOV     R5,-(SP)
                                ;
                                ;  Try to obtain the data by accessing in-memory image of parent job
                                ;
002556  010267  175272                  MOV     R2,EMTUKC+6     ;Set # of block we want
002562  012700  000046'                 MOV     #EMTUKC,R0      ;Point to EMT arg block
002566  104375                          EMT     375             ;Try to copy data from in-memory image
002570  103034                          BCC     1$              ;Br if we got the data
002572  010005                          MOV     R0,R5           ;Save starting block of region in swap file
002574  060205                          ADD     R2,R5           ;Add block within region
                                ;
                                ;  The parent job is not in memory.
                                ;  Open a channel to the PLAS swap file unless it is already open.
                                ;
002576  005704                          TST     R4              ;Have we opened a channel to plas swap file?
002600  001012                          BNE     2$              ;Br if yes
002602                                  .LOOKUP #XAREA,#1,#PSFNAM ;Open channel to PLAS swap file
002622  103427                          BCS     9$              ;Br if error on lookup
002624  005204                          INC     R4              ;Set flag saying channel open
                                ;
                                ;  Read data from plas swap file
                                ;
002626                          2$:     .READW  #XAREA,#1,#BLK0,R3,R5 ;Read data from plas swap file
                                ;
                                ;  Move data from BLK0 to key def region
                                ;
002662  072227  000011          1$:     ASH     #9.,R2          ;Convert block # to byte offset
002666  062702  000000G                 ADD     #VPAR1,R2       ;Get virtual address in region
002672  012705  000000G                 MOV     #BLK0,R5        ;Point to current buffer
002676  012522                  3$:     MOV     (R5)+,(R2)+     ;Move the data
002700  077302                          SOB     R3,3$           ;Loop if more to move
                                ;
                                ;  Finished
                                ;
002702  012605                  9$:     MOV     (SP)+,R5
002704  012603                          MOV     (SP)+,R3
002706  012602                          MOV     (SP)+,R2
002710  000207                          RETURN
```

```
                                        .SBTTL  RECALL command
                                ;-----------------------------------------------------------------------
                                ;  Process the RECALL command that is used with the single line editor.
                                ;
                                ;  Inputs:
                                ;   R1 = User index number
                                ;   R2 = Address of end of command string
                                ;   R3 = Address of start of command argument field.
                                ;
002712                          CMDRCL:
                                ;
                                ;  Error if SL is not turned on
                                ;
002712  032761  000000G 000000G        BIT     #$SLON,LSW7(R1) ;Is Single line editor turned on?
002720  001004                         BNE     1$              ;Br if yes
002722                                  FABORT  #EM$SLO         ;SL not turned on
                                ;
                                ;  Advance over the RECALL command
                                ;
002732  016704  000000G        1$:     MOV     SLSPTR,R4       ;Get ptr to last command (RECALL)
002736  010467  175052                 MOV     R4,RCLPTR       ;Save ptr to RECALL command
002742  004767  000534                 CALL    SLMVUP          ;Advance to next command
002746  010467  000000G                MOV     R4,SLSPTR       ;Set new pointers
002752  010467  000000G                MOV     R4,SLLPTR
                                ;
                                ;  Convert command arguments to upper case
                                ;
002756  004767  000000G                CALL    CVTTAB          ;Convert tabs and FF to spaces
002762  004767  000000G                CALL    CVTUC           ;Convert lower case letters to upper case
                                ;
                                ;  If command has no arguments, treat it like "RECALL 1".
                                ;
002766  105713                         TSTB    (R3)            ;Are there any arguments?
002770  001004                         BNE     2$              ;Br if yes
002772  012701  000001                 MOV     #1,R1           ;Get command number to recall
002776  000167  000100                 JMP     RCLVAL          ;Go do the recall
                                ;
                                ;  See if "/ALL" was specified
                                ;
003002  121327  000057         2$:     CMPB    (R3),#'/        ;Is there a switch present?
003006  001016                         BNE     3$              ;Br if not
003010  126327  000001 000101        CMPB    1(R3),#'A       ;/ALL?
003016  001502                         BEQ     RCLALL          ;Br if yes
003020  126327  000001 000103        CMPB    1(R3),#'C       ;/CLEAR?
003026  001404                         BEQ     4$              ;Br if yes
003030                                  FABORT  #INVOPT         ;Invalid option
003040  000167  000000G        4$:     JMP     RDCMD           ;Ignore /CLEAR
                                ;
                                ;  If the first character is a digit, then we are recalling a specific line
                                ;
003044  121327  000060         3$:     CMPB    (R3),#'0        ;Is this a digit?
003050  103432                         BLO     RCLSTR          ;Br if not
003052  121327  000071                 CMPB    (R3),#'9
003056  101027                         BHI     RCLSTR
003060  000400                         BR      RCLNUM          ;Numeric parameter
```

```
                        ;+-----------------------------------------------------------------------------
                        ;  Recall a command whose index number is specified.
                        ;
003062                  RCLNUM:
                        ;
                        ;  Accrue the number
                        ;
003062  004767  000000G        CALL    ACRDEC          ;Accrue the decimal value
003066  105700                TSTB    R0              ;Null should be the delimiter
003070  001404                BEQ     RCLVAL          ;Br if ok
003072                        FABORT  #EM$CSE         ;Syntax error
                        ;
                        ;  The index number of the command to recall is in R1.
                        ;  Recall that command.
                        ;
003102  016704  000000G RCLVAL: MOV     SLSPTR,R4       ;Get ptr to last saved line
003106  005301          1$:     DEC     R1              ;Need to go back to an earlier line?
003110  003406                  BLE     2$              ;Br if not
003112  004767  000364          CALL    SLMVUP          ;Move back to earlier line
003116  020467  000000G        CMP     R4,SLSPTR       ;Wrapped around to beginning?
003122  001371                  BNE     1$              ;Br if not
003124  000402                  BR      9$              ;Nothing to recall
                        ;
                        ;  Set pointer to line to recall when next input is done
                        ;
003126  010467  000000G 2$:     MOV     R4,SLRPTR       ;Recall this line on next input
003132  000167  000000G 9$:     JMP     RDCMD
```

```
                                ;-----------------------------------------------------------------------
                                ;  Recall a command by searching for a specific character string.
                                ;
003136  016704  0000006G        RCLSTR: MOV     SLSPTR,R4       ;Point to most recently saved command
003142  005001                          CLR     R1              ;Init command recall index
                                ;
                                ;  Compare target string with 1st characters of command
                                ;
003144  005201                  5$:     INC     R1              ;Increment command recall index
003146  010405                          MOV     R4,R5           ;Get ptr to saved command
003150  010302                          MOV     R3,R2           ;Get ptr to target string
003152  105712                  1$:     TSTB    (R2)            ;Are we at the end of the target string?
003154  001752                          BEQ     RCLVAL          ;Br if hit end of target
003156  112500                          MOVB    (R5)+,R0        ;Get next char from saved cmd
003160  001412                          BEQ     3$              ;Br if hit end of saved cmd
003162  020027  000141                  CMP     R0,#'a          ;Cvt lower-case to upper case
003166  103405                          BLO     4$
003170  020027  000172                  CMP     R0,#'z
003174  101002                          BHI     4$
003176  162700  000040                  SUB     #40,R0
003202  122200                  4$:     CMPB    (R2)+,R0        ;Does target match saved cmd?
003204  001762                          BEQ     1$              ;Loop if yes
                                ;
                                ;  This command does not match target.
                                ;  Advance to the next saved command.
                                ;
003206  004767  000270          3$:     CALL    SLMVUP          ;Advance to next saved command
003212  020467  0000006G                CMP     R4,SLSPTR       ;Wrapped around to beginning?
003216  001352                          BNE     5$              ;Br if not
003220  000167  0000006G                JMP     RDCMD           ;Nothing to recall
```

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

```
      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 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
```

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

```
                                ;-----------------------------------------------------------------------
                                ;  This routine returns a pointer to the previous saved command moving
                                ;  in an up-arrow direction.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to current command.
                                ;
                                ;  Outputs:
                                ;   R4 = Pointer to previous command.
                                ;
003502  010546                  SLMVUP: MOV     R5,-(SP)
                                ;
                                ;  Skip up to the null at the end of the current command
                                ;
003504  010405                          MOV     R4,R5
003506  020527  0000000         1$:     CMP     R5,#SLLEND      ;Hit end of buffer?
003512  103402                          BLO     2$              ;Br if not
003514  012705  0000000                 MOV     #SLLBUF,R5      ;Wrap around to the top
003520  105725                  2$:     TSTB    (R5)+           ;Reached null at end of saved command?
003522  001371                          BNE     1$              ;Br if not
                                ;
                                ;  Now skip over nulls at the end of the command
                                ;
003524  020527  0000000         3$:     CMP     R5,#SLLEND      ;Hit end of buffer?
003530  103402                          BLO     4$              ;Br if not
003532  012705  0000000                 MOV     #SLLBUF,R5      ;Wrap around to the top
003536  105725                  4$:     TSTB    (R5)+           ;More nulls to skip?
003540  001003                          BNE     5$              ;Br if not
003542  020504                          CMP     R5,R4           ;Wrapped around to beginning?
003544  001367                          BNE     3$              ;Br if not
003546  000407                          BR      9$              ;Nothing to recall
                                ;
                                ;  Point to 1st character of command
                                ;
003550  020527  0000000         5$:     CMP     R5,#SLLBUF      ;At top of buffer now?
003554  101002                          BHI     6$              ;Br if not
003556  012705  0000000                 MOV     #SLLEND,R5      ;Wrap around to bottom
003562  005305                  6$:     DEC     R5              ;Point to 1st char of next command
                                ;
                                ;  Finished
                                ;
003564  010504                          MOV     R5,R4           ;Return pointer in R5
003566  012605                  9$:     MOV     (SP)+,R5
003570  000207                          RETURN
```

```
                                        .SBTTL  ACCESS Command
                                ;-----------------------------------------------------------------------
                                ;  Process the ACCESS command
                                ;
003572  004767  000410          CMDACC: CALL    OKFCOK          ;Cmd only valid if priv'd or in startup cf
003576  103004                          BCC     1$              ;Br if OK
003600                                  FABORT   #EM$NSF        ;Else abort command
                                ;
003610  004767  000000G         1$:     CALL    CVTTAB          ;Clean up white spaces in command line
003614  016704  000000G                 MOV     OKFAND,R4       ;Point to end of ACCESS entries
                                ;
                                ;  Top of outer loop for each ACCESS specification
                                ;
003620  105713                  3$:     TSTB    (R3)            ;End of command line?
003622  001002                          BNE     4$              ;Br if not
003624  000167  000000G                 JMP     RDCMD           ;If so, finished, get next command
                                ;
003630  020467  000000G         4$:     CMP     R4,OKFNND       ;Are we about to run over NOACCESS entries?
003634  103404                          BLO     5$              ;Br if OK to add here
003636                                  FABORT  #TBLOVF         ;Else report table overflow and abort
                                ;
003646  005764  000000G         5$:     TST     OF$FIL(R4)      ;Is this entry free?
003652  001403                          BEQ     6$              ;Br if so
003654  062704  000000G                 ADD     #OF$$SZ,R4      ;If not, point to next slot
003660  000763                          BR      4$              ;And keep checking
                                ;
                                ;  R4 now points to free ACCESS table entry, add entry from command line
                                ;  Errors: If there is an error on the dev:fil.ext specification for an
                                ;  ACCESS command, then the only problem is forbidding access to a requested
                                ;  device -- this error only needs to be a warning.
                                ;  However, if there is an error on the switch, then we might end
                                ;  up granting full access when some restriction is desired -- abort this.
                                ;  In general warn if they get less than they want, but abort if there is
                                ;  a chance they could get too much.
                                ;
003662  004767  000350          6$:     CALL    OKFINI          ;Init this ACCESS file entry (to all wild)
003666  004767  000406                  CALL    OKFADD          ;Add cmd line entry to ACCESS table
003672  103417                          BCS     ACBDFS          ;Report any specification errors
003674  012700  000122                  MOV     #'R,R0          ;"Readonly" is only valid switch
003700  004767  000562                  CALL    OKFSWI          ;Go handle switch if specified
003704  103447                          BCS     ACBDSW          ;Report any switch errors
                                ;
                                ;  Finished getting file specification, step up to next entry and repeat
                                ;
003706  005267  000000G         15$:    INC     RESDEV          ;Count another entry added
003712  062704  000000G                 ADD     #OF$$SZ,R4      ;Point to next ACCESS table entry
003716  010467  000000G                 MOV     R4,OKFAND       ;Remember end of ACCESS entries
003722  004767  000242                  CALL    SKPD2           ;Skip over terminating delimiters
003726  000167  177666                  JMP     3$              ;Repeat to end of ACCESS command line
```

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

```
                                        .SBTTL  NOACCESS Command
                                ;-----------------------------------------------------------------------
                                ;  Process the NOACCESS command
                                ;
004034  004767  000146          CMDNAC: CALL    OKFCOK          ;Cmd only valid if priv'd or in startup cf
004040  103004                          BCC     1$              ;Br if OK
004042                                  FABORT  #EM$NSF         ;Else abort command
                                ;
004052  004767  000000G         1$:     CALL    CVTTAB          ;Compress command line white space
004056  016704  000000G                 MOV     OKFNND,R4       ;Point to start of NOACCESS entries
                                ;
                                ;  Top of outer loop for each NOACCESS specification
                                ;
004062  105713                  3$:     TSTB    (R3)            ;Have we hit end of command line?
004064  001002                          BNE     4$              ;Br if not
004066  000167  000000G                 JMP     RDCMD           ;If so, finished, get next command
                                ;
004072  162704  000000G         4$:     SUB     #OF$$SZ,R4      ;Try to make room for new entry
004076  020467  000000G                 CMP     R4,OKFAND       ;Have we stepped into the ACCESS entries?
004102  103004                          BHIS    5$              ;Br if OK to add here
004104                                  FABORT  #TBLOVF         ;Else report table overflow and abort
                                ;
004114  005764  000000G         5$:     TST     OF$FIL(R4)      ;Is this entry free?
004120  001364                          BNE     4$              ;If not, point to next slot
                                ;
                                ;  R4 now points to free NOACCESS table entry, add entry from command line
                                ;  Since any error on a NOACCESS would be less restrictive than desired,
                                ;  abort, rather than warn, on any errors.
                                ;
004122  004767  000110          6$:     CALL    OKFINI          ;Init this NOACCESS file entry (to all wild)
004126  004767  000146                  CALL    OKFADD          ;Add cmd line entry to NOACCESS table
004132  103730                          BCS     ACBDFX          ;Abort on any specification errors
004134  012700  000127                  MOV     #'W,R0          ;"Write" is only valid switch
004140  004767  000322                  CALL    OKFSWI          ;Go handle switch if specified
004144  103727                          BCS     ACBDSW          ;Report any switch errors
                                ;
                                ;  Finished getting file specification, step up to next entry and repeat
                                ;
004146  005267  000000G         15$:    INC     RESDEV          ;Count another entry added
004152  010467  000000G                 MOV     R4,OKFNND       ;Remember new start of NOACCESS entries
004156  004767  000006                  CALL    SKPD2           ;Skip over terminating delimiters
004162  000167  177674                  JMP     3$              ;Repeat to end of NOACCESS command line
```

```
                                        .SBTTL          NO[ACCESS] Command subroutines
                                ;-------------------------------------------------------------------
                                ;  SUBROUTINE TO SKIP OVER COMMAS AND SPACES
                                ;
004166  005203                  SKPD1:  INC     R3              ;POINT TO NEXT CHAR
004170  121327  000040          SKPD2:  CMPB    (R3),#'         ;IS IT A SPACE
004174  001774                          BEQ     SKPD1           ;KEEP SKIPPING IF YES
004176  121327  000054                  CMPB    (R3),#',        ;IS IT A COMMA
004202  001771                          BEQ     SKPD1           ;KEEP SKIPPING IF YES
004204  000207                          RETURN
                                ;
                                        .SBTTL          Check for privilege to issue [NO]ACCESS
                                ;-------------------------------------------------------------------
                                ;  ACCESS and NOACCESS commands are only allowed in start-up command files
                                ;  or if the user is privileged. Return with carry clear if either condition
                                ;  is met; return with carry set if not.
                                ;
                                ;  Inputs:
                                ;       R1 contains the job index
                                ;
004206  032761  000000G 000000G OKFCOK: BIT     #$SUCF,LSW9(R1) ;Are we in start-up command file?
004214  001006                          BNE     9$              ;Return OK if so
004216  032767  000000G 000000G        BIT     #PO$SYS,PRIVCO  ;Is job privileged?
004224  001002                          BNE     9$              ;Return OK if so
004226  000261                          SEC                     ;If neither, return with error
004230  000401                          BR      10$
004232  000241                  9$:     CLC
004234  000207                  10$:    RETURN
                                ;
                                        .SBTTL          Init a [NO]ACCESS table entry
                                ;-------------------------------------------------------------------
                                ;  Initialize a [NO]ACCESS table entry to all wildcards.
                                ;
                                ;  Inputs:
                                ;       R4 points to [NO]ACCESS table entry.
                                ;
004236  012702  132500          OKFINI: MOV     #WLDNAM,R2      ;Get RAD50 wildcard value
004242  010264  000000G                 MOV     R2,OF$FIL(R4)   ;Set FIL to wildcard
004246  010264  000002G                 MOV     R2,OF$FIL+2(R4) ;Set NAM to wildcard
004252  010264  000004G                 MOV     R2,OF$FIL+4(R4) ;Set EXT to wildcard
004256  112764  177777  000000G        MOVB    #-1,OF$DEV(R4)  ;Set device index to wild
004264  112764  177777  000000G        MOVB    #-1,OF$UNT(R4)  ;Set device unit to wild
004272  105064  000000G                CLRB    OF$FLG(R4)      ;No flags yet
004276  000207                          RETURN
                                ;
                                        .SBTTL          Add entry to [NO]ACCESS tables
                                ;-------------------------------------------------------------------
                                ;  Accrue device or file name and add entry to [NO]ACCESS table.
                                ;  [NO]ACCESS table entry is assumed to be initialized to all wild.
                                ;
                                ;  Inputs:
                                ;       R3 points to current input command line location
                                ;       R4 points to empty [NO]ACCESS table entry
                                ;
004300  004767  000000G         OKFADD: CALL    GTRD50          ;Scan first name
004304  121327  000072                  CMPB    (R3),#':        ;Was this a device name?
004310  001036                          BNE     10$             ;Nope, leave device as wild, go treat as file
```

```
                                        ;
                                        ;  We have a device specification
                                        ;  Translate logical assignments if necessary and insert the entry
                                        ;
004312  016700  000000G                       MOV     R50BUF,R0       ;Get specified device name
004316  020027  132500                        CMP     R0,#WLDNAM      ;Was it *: ?
004322  001417                              BEQ     8$              ;If *, skip to file (device is default wild)
004324  004767  000000G                       CALL    ASNSRC          ;Not wild, convert logical to physical
004330  103403                              BCS     7$              ;Br if entry not in assign table
004332  016267  000000G 000000G              MOV     AT$DEV(R2),R50BUF ;Replace logical name with physical name
                                        ;
                                        ;  Translate physical device specification to device table index
                                        ;
004340  016701  000000G                 7$:   MOV     R50BUF,R1       ;Copy device name
004344  004767  000170                        CALL    CVTDEV          ;Convert into dev index (R1) and unit # (R2)
004350  103444                              BCS     13$             ;Br if bad device name
004352  110164  000000G                       MOVB    R1,OF$DEV(R4)   ;Set device index into access table
004356  110264  000000G                       MOVB    R2,OF$UNT(R4)   ;Set device unit into access table
                                        ;
                                        ;  We need to accrue the file specification
                                        ;
004362  005203                          8$:   INC     R3              ;Skip over :
004364  111300                              MOVB    (R3),R0         ;Get first char of file spec
004366  120027  000052                        CMPB    R0,#'*          ;Wild card? (CHKDLM would find it)
004372  001403                              BEQ     9$              ;If yes, go accrue it
004374  004767  000000G                       CALL    CHKDLM          ;1st char of filnam shouldn't be delim
004400  103426                              BCS     12$             ;If it is, must be dev:, or dev:/switch
004402  004767  000000G                 9$:   CALL    GTRD50          ;Accrue file name
                                        ;
                                        ;  We have a file specification
                                        ;
004406  016700  000000G                 10$:  MOV     R50BUF,R0       ;Get RAD50 file name (1-3)
004412  001423                              BEQ     13$             ;Br on invalid file spec
004414  020027  132500                        CMP     R0,#WLDNAM      ;File name wild?
004420  001405                              BEQ     11$             ;Br if so (already init'd to wild)
004422  010064  000000G                       MOV     R0,OF$FIL(R4)   ;Set file name into access table
004426  016764  000002G 000002G              MOV     R50BUF+2,OF$FIL+2(R4) ;Set 2nd half (4-6) of name
                                        ;
                                        ;  Check for file extension
                                        ;
004434  121327  000056                  11$:  CMPB    (R3),#'.        ;Was file extension specified?
004440  001006                              BNE     12$             ;Br if not (already init'd to wild)
004442  005203                              INC     R3              ;Step over .
004444  004767  000000G                       CALL    GTRD50          ;Accrue the extension
004450  016764  000000G 000004G              MOV     R50BUF,OF$FIL+4(R4) ;Set extension into access table
                                        ;
                                        ;  Normal return with no error
                                        ;
004456  000241                          12$:  CLC
004460  000207                              RETURN
                                        ;
                                        ;  Return thru here on invalid device or file specification
                                        ;
004462  000261                          13$:  SEC
004464  000207                              RETURN
                                        ;
                                                .SBTTL          Handle [NO]ACCESS switches
```

```
                                ;-----------------------------------------------------------------------
                                ;  This routine checks to see if a switch was specified and set the
                                ;  read-only flag for the entry if so.  The only valid switches are
                                ;  /R or /r for ACCESS and /W or /w for NOACCESS.  Examples:
                                ;
                                ;       ACCESS   dev:[filnam[.ext]]/Readonly
                                ;       NOACCESS dev:[filnam[.ext]]/Write
                                ;
                                ;  Inputs:
                                ;       R0 contains 1st char (uppercase) of valid switch
                                ;       R3 points to 1st char after dev or file specification
                                ;       R4 points to [NO]ACCESS table entry
                                ;
004466  121327  000057        OKFSWI: CMPB    (R3),#'/        ;Switch specified?
004472  001020                        BNE     9$              ;Just return if not
004474  116346  000001                MOVB    1(R3),-(SP)     ;Get 1st char of switch
004500  042716  000040                BIC     #40,(SP)        ;Force it to uppercase ('a-'A)
004504  122600                        CMPB    (SP)+,R0        ;Does it say read-only?
004506  001402                        BEQ     13$             ;Br if so
004510  000261                        SEC                     ;Else this is an invalid switch
004512  000411                        BR      10$             ;And return
004514  152764  000000G 000000G 13$:     BISB    #OT$RON,OF$FLG(R4) ;Set read-only flag on file spec
                                ;
                                ;  Now step the cmd line ptr up to the end of the switch
                                ;
004522  005203                14$:     INC     R3              ;Step up to next delimiter
004524  111300                        MOVB    (R3),R0         ;Get the next char
004526  004767  0000000               CALL    CHKDLM          ;Is it a delimiter?
004532  103373                        BCC     14$             ;Keep searching for end of switch
                                ;
004534  000241                9$:     CLC                     ;Normal return
004536  000207                10$:    RETURN
```

```
                                        .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.
                                ;
004540                          CVTDEV:
                                ;
                                ;  Get device name and split off unit number.
                                ;
004540  012702  177777                  MOV     #-1,R2          ;ASSUME NO UNIT NUMBER SPECIFIED
004544  005000                          CLR     R0              ;SET FOR DIVIDE
004546  071027  000050                  DIV     #50,R0          ;SPLIT OFF LOW-ORDER RAD50 CHARACTER
004552  005701                          TST     R1              ;WAS A UNIT NUMBER SPECIFIED?
004554  001412                          BEQ     6$              ;BR IF NOT (LEAVE -1, INDICATES UNSPECIFIED)
004556  010102                          MOV     R1,R2           ;Not unspecified, assume * for the moment
004560  020127  000035                  CMP     R1,#R50AST      ;Is unit number a wild card?
004564  001406                          BEQ     6$              ;Preserve it if so, else
004566  162701  000036                  SUB     #36,R1          ;CONVERT RAD50 DIGIT TO BINARY VALUE
004572  010102                          MOV     R1,R2           ;PUT UNIT NUMBER IN R2
004574  020227  000011                  CMP     R2,#9.          ;MAKE SURE UNIT NUMBER IN RANGE 0-9.
004600  101027                          BHI     8$              ;BR IF INVALID UNIT NUMBER
004602  070027  000050          6$:     MUL     #50,R0          ;GET DEVICE NAME WITHOUT UNIT #
                                ;
                                ;  The RAD50 device name less unit number is now in R1.
                                ;  The unit number is in R2.
                                ;
                                ;  Translate SY and DK.
                                ;
004606  020167  000000G                 CMP     R1,R50SY        ;IS DEVICE NAME SY?
004612  001403                          BEQ     2$              ;BR IF YES
004614  020167  000000G                 CMP     R1,R50DK        ;IS DEVICE NAME DK?
004620  001007                          BNE     3$              ;BR IF NOT
004622  016701  000000G         2$:     MOV     SYINDX,R1       ;GET SY DEVICE INDEX NUMBER
004626  005702                          TST     R2              ;Was unit number specified?
004630  002016                          BGE     4$              ;Yes, must have been 0-9 or *
004632  116702  000001G                 MOVB    SYUNIT+1,R2     ;No, just SY:, use real boot unit
004636  000413                          BR      4$
                                ;
                                ;  Look up device name in permanent device name table.
                                ;
004640  016700  000000G         3$:     MOV     NUMDEV,R0       ;GET INDEX OF LAST PERM NAME
004644  020160  000000G         5$:     CMP     R1,PNAME(R0)    ;LOOK UP DEVICE NAME
004650  001405                          BEQ     7$              ;BR IF FOUND
004652  162700  000002                  SUB     #2,R0           ;CHECK NEXT ENTRY
004656  002372                          BGE     5$
                                ;
                                ;  Invalid device name
                                ;
004660  000261                  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
```

```
                        .SBTTL  GETSYP -- Accept system logon password
                ;-----------------------------------------------------------------------
                ;  The system logon password is an optional password which the system
                ;  may require for some lines before entering the normal logon sequence.
                ;
                ;  Inputs:
                ;   R1 = Job index number
                ;
004712  010246  GETSYP: MOV     R2,-(SP)
004714  010346         MOV     R3,-(SP)
004716  010546         MOV     R5,-(SP)
                ;
                ;  Don't ask for a password if there is none defined
                ;
004720  105767  000000G        TSTB    SYPSWD          ;Is there a defined system password?
004724  001535               BEQ     9$              ;Br if not
                ;
                ;  Pretend to lock KMON to line to prevent ^C aborts
                ;
004726  052761  000000G 000000G        BIS     #$PRGLK,LSW5(R1)
                ;
                ;  Log off job immediately if it has been aborted
                ;
004734  032761  000000G 000000G        BIT     #$DOOFF,LSW(R1) ;Should the job be logged off?
004742  001132               BNE     20$             ;Br if yes
                ;
                ;  Disable control-C abort and control-W subprocess switch
                ;
004744  042767  000000G 000000G        BIC     #P2$VIR,PRIVC2  ;Disable subprocess use
004752  004767  000000G        CALL    FIXPRV          ;Transfer privilege flag to LSW tables
004756  052761  000000G 000000G        BIS     #$SCCA,LSW5(R1) ;Suppress control-C aborts
                ;
                ;  Initialize count of number of times password has been entered
                ;
004764  012705  000002         MOV     #2,R5           ;Init password attempt count
                ;
                ;  Print password prompt
                ;
004770         1$:     .PRINT  #CRLF           ;Print CR-LF
004776                 .PRINT  #SYPSPR         ;Print password prompt
                ;
                ;  Set read time-out time
                ;
005004  012700  000060'        MOV     #RDTIME,R0      ;Point to EMT arg block
005010  016760  000000G 000002 MOV     VONTM,2(R0)     ;Set timeout value
005016  104375         EMT     375             ;Set read time-out
                ;
                ;  Accept a line of input
                ;
005020  012702  000000G        MOV     #BLK0,R2        ;Point to buffer for input string
005024  042761  000000G 000000G MOV     #$ECHO,LSW2(R1) ;Turn off character echoing
005032         2$:     .TTYIN                  ;Get next input character
005036  120027  000001         CMPB    R0,#1           ;Did read time-out?
005042  001472               BEQ     20$             ;Br if yes
005044  120027  000015         CMPB    R0,#CR          ;Ignore carriage return
005050  001770               BEQ     2$
005052  120027  000012         CMPB    R0,#LF          ;End of input line?
```

```
     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                          MOVB    R0,(R2)+        ;Store received character
     62 005070  000760                          BR      2$              ;Continue receiving input
     63 005072  105012                  3$:     CLRB    (R2)            ;Store null at end of string
     64                                 ;
     65                                 ;  Convert received and expected passwords to upper case
     66                                 ;
     67 005074  012703  000000G                 MOV     #BLK0,R3        ;Point to received password
     68 005100  004767  000000G                 CALL    CVTUC           ;Convert to upper case
     69 005104  012703  000000G                 MOV     #SYPSWD,R3      ;Point to expected password
     70 005110  004767  000000G                 CALL    CVTUC           ;Convert to upper case
     71                                 ;
     72                                 ;  See if received password matches expected password
     73                                 ;
     74 005114  012702  000000G                 MOV     #BLK0,R2        ;Point to received string
     75 005120  012703  000000G                 MOV     #SYPSWD,R3      ;Point to expected password
     76 005124  122213                  4$:     CMPB    (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         000001                          .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
```

Cross reference table (CREF V05.05)

```
$DOOFF    1-29      34-24
$ECHO     1-33      34-51     34-90
$KINIT    1-32      13-60
$NOIN     1-33      34-91
$PRGLK    1-31      34-20     34-93
$SCCA     1-31      34-31     34-94
$SLON     1-29      22-14
$SUCF     1-33      32-21     34-92
...V1    13-21      14-20     15-9      15-14     16-9      16-14     17-11     18-17     18-28     18-32     18-34     18-37
          18-39     18-55     18-57     18-62     19-33     20-60     20-60     21-32     21-32     21-32     21-38     21-38
          25-17     25-21     25-26     34-39     34-40     34-97
...V2    20-60      20-60#    21-32     21-32     21-32#    21-32#    21-38     21-38     21-38#    21-38#
ACBDFS    29-38     30-4#
ACBDFX    30-28#    31-32
ACBDSW    29-41     30-29#    31-35
ACRDEC     1-35     23-8
ACRSTR     1-38     11-33
ACRTXT     1-32      7-59
AMBOPT     1-40      6-29
ASNSRC     1-26     32-65
AT$DEV     1-26     32-67
BADACC     1-25     30-4      30-28
BELL      1-55#
BLANK     1-54#
BLKO       1-34      1-38      2-34      7-62      11-36     21-38     21-44     34-50     34-59     34-67     34-74
BLKWDS    1-59#
CHKDLM     1-26     32-83     32-142
CMDACC     1-21     29-5#     30-24
CMDDEF     1-21      6-9#
CMDNAC     1-21     30-23     31-5#
CMDRCL     1-21     22-10#
COLSPC    2-50#     25-17
COMMA     1-58#     30-14
CORUSR     1-26     13-59     30-21
CR        1-53#     34-55
CRLF       1-34     18-62     25-26     34-39     34-97
CVTDEV    32-72     33-14#
CVTTAB     1-40      6-9      22-28     29-9      31-9
CVTUC      1-32     22-29     34-68     34-70
DEFHD      4-4#      6-20
DEFKEY     4-5       7-11#
EM$CSE     1-30     23-11
EM$KCR     1-37     13-62     13-69
EM$KNS     1-35      7-23
EM$KNT     1-37      9-16
EM$KNU     1-42      7-88
EM$KTF     1-42      9-44
EM$KWC     1-37     14-32
EM$NSF     1-35     29-7      31-7
EM$NSL     1-36      7-17
EM$SLO     1-29     22-16
EM$STL     1-38     11-57
EMTUKC     2-30#    20-16*    20-17*    20-18     21-20*    21-21
ERRLOC     1-35     20-21
FF        1-57#
FIND20    25-9      26-12#
```

Cross reference table (CREF V05.05)

```
FIXPRV    1-31      34-30     34-96
FKILL     1-40       6-29      6-30      7-17      7-23      7-84      7-88      9-16      9-44     11-57     13-69     14-32
         22-16      22-46     23-11      29-7      29-20     30-28     30-29     31-7      31-21
GETSYP    1-21      34-9#
GTRD50    1-26      32-55     32-85     32-101
HNGEMT    2-44#     34-109
ILLCMD    1-37       7-84
INIUKD    1-21      20-10#
INVOPT    1-30       1-37      6-30      22-46     30-29
KC$COM    1-44       5-30
KC$DOT    1-44       5-29
KC$DWN    1-45       5-37
KC$E1     1-45       5-38      5-44
KC$E2     1-45       5-39      5-45
KC$E3     1-45       5-40      5-46
KC$E4     1-45       5-41      5-47
KC$E5     1-46       5-42      5-48
KC$E6     1-46       5-43      5-49
KC$ENT    1-44       5-32
KC$F10    1-46       5-55
KC$F11    1-46       5-56
KC$F12    1-47       5-57
KC$F13    1-47       5-58
KC$F14    1-47       5-59
KC$F15    1-47       5-50      5-60
KC$F16    1-47       5-36      5-61
KC$F17    1-47       5-62
KC$F18    1-47       5-63
KC$F19    1-48       5-64
KC$F20    1-48       5-65
KC$F6     1-46       5-51
KC$F7     1-46       5-52
KC$F8     1-46       5-53
KC$F9     1-46       5-54
KC$KP0    1-43       5-9       5-19
KC$KP1    1-43       5-10      5-20
KC$KP2    1-43       5-11      5-21
KC$KP3    1-43       5-12      5-22
KC$KP4    1-43       5-13      5-23
KC$KP5    1-43       5-14      5-24
KC$KP6    1-43       5-15      5-25
KC$KP7    1-44       5-16      5-26
KC$KP8    1-44       5-17      5-27
KC$KP9    1-44       5-18      5-28
KC$LFT    1-45       5-33
KC$MIN    1-44       5-31
KC$PF1    1-48       5-5
KC$PF2    1-48       5-6
KC$PF3    1-48       5-7
KC$PF4    1-48       5-8
KC$RIT    1-45       5-34
KC$UP     1-45       5-35
KCODE     2-5#       2-6       7-27*     7-53*     7-62*     7-70*     9-31      9-49     10-18
KD$$SZ    1-42      10-32     12-26     13-14     13-46     14-12     17-35     20-44
KD$COD    1-41       9-49*    10-24*    10-30     12-24     13-45*    17-26     18-29     19-18
KD$FLG    1-41       9-50*
```

Cross reference table (CREF V05.05)

```
KD$TXT    1-41      9-51      18-50
KD$TYP    1-36     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   20-50     21-14#
KEYMXT    1-38     11-34      11-45
KEYPAR    1-41     13-30*     16-19*    17-9
KEYPRT   17-31     18-8#
KEYRCB    1-36      1-42       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     20-57
KF$ECO    1-40      4-12       4-13      7-29      9-14
KF$TRM    1-40      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#    13-12*     13-17*    13-21     13-23     13-28     13-38     15-13*    15-14     16-13*    16-14
KT$GLD    1-40      4-16       8-23      8-30     18-13
KT$GLT    1-42      7-46       8-32     18-15     18-26
KT$LET    1-41      4-18       7-44      8-25      8-28     18-24
KT$NRM    1-41      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        1-52#    34-57
LSW       1-29     13-60      34-24
LSW2      1-32     34-51*     34-90*
LSW3      1-33     34-91*
LSW5      1-31     34-20*     34-31*    34-93*    34-94*
LSW7      1-29     22-14
LSW9      1-33     32-21      34-92*
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
```

Cross reference table (CREF V05.05)

```
                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
NUMDEV          1-27    33-49
OF$$SZ          1-28    29-24   29-46   31-18
OF$DEV          1-28    32-41*  32-74*
OF$FIL          1-28    29-22   30-5*   31-23   32-38*  32-39*  32-40*  32-93*  32-94*  32-102*
OF$FLG          1-28    32-43*  32-136*
OF$UNT          1-28    32-42*  32-75*
OKFADD         29-37    31-31   32-55#
OKFAND          1-25    29-10   29-47*  31-19
OKFCOK         29-5     31-5    32-21#
OKFINI         29-36    31-30   32-37#
OKFNND          1-25    29-18   30-22   31-10   31-40*
OKFSWI         29-40    31-34   32-128#
OPTLST          1-41     7-34
OT$RON          1-28    32-136
PO$SYS          1-26    32-23
P2$VIR          1-31    34-29
PNAME           1-27    33-50
PRIVA2          1-31    34-95
PRIVCO          1-27    32-23
PRIVC2          1-31    34-29*  34-95*
PRTFIX          1-30    25-13
PRTSPC          1-36    18-20   18-40   19-39
PRTWRN          1-25    13-62   30-4
PSFNAM          2-7#    20-29   21-32
R.GID           1-39    13-28
R.GSIZ          1-34    13-17*
R.GSTS          1-39    13-12*  13-23   13-38   15-13*  16-13*
R50AST          1-61#   33-24   33-63
R50BUF          1-25    32-62   32-67*  32-71   32-89   32-94   32-102
R50DK           1-27    33-39
R50SY           1-27    33-37
RC$BAS          1-42    13-30
RCLALL         22-43    25-4#
RCLNUM         22-55    23-4#
RCLPTR          2-8#    22-21*  25-41   26-18
RCLREV          1-32    25-7    25-30
RCLSTR         22-52    22-54   24-4#
RCLVAL         22-36    23-10   23-16#  24-13
RDCMD           1-41     9-59   10-47   17-12   17-44   22-47   23-27   24-30   25-43   29-16   30-11   31-16
RDTIME          2-38#   34-44
RESDEV          1-25    29-45*  31-39*
RS.CGR          1-39     2-14   13-12
RS.CRR          1-39    13-23
RS.EGR          1-36    16-13
RS.GBL          1-39     2-14   13-12   15-13   16-13
RS.NEW          1-36    13-38
RS.PVT          1-39     2-14   13-12   15-13   16-13
RSFBLK          1-35    20-28
SEARCH          1-40     6-21    7-68
```

Cross reference table (CREF V05.05)

```
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
```

Cross reference table (CREF V05.05)

```
. . . 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   20-60
. . . 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    34-39    34-40
            34-97
. . . CM6   13-21    14-20    15-9     15-14    16-9     16-14
. . . CM7   21-38
. CLOSE      1-17#   20-60
. 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    34-39    34-40    34-97
. READW      1-17#   21-38
. TTYIN      1-17#   34-52
. 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    13-69    14-32
            22-16    22-46    23-11    29-7     29-20    30-28    30-29    31-7     31-21
FERR         3-4#
FWARN        3-23#   13-62    30-4
TBLDEF       3-35#    4-4      4-11     5-4
TBLEND       3-61#    4-6      4-19     5-66
TSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX                                  -PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-Pl
usTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusT  S&H Computer Systems, Inc.  SX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-
PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-Plu                              sTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTSX-PlusTS
```

```
TTTTT  SSS  K   K M   M  222   CCCC
  T   S   S K  K  MM MM 2   2 C
  T   S     K K   M M M    2  C
  T    SSS  KK    M   M   2   C
  T       S K K   M   M  2    C
  T   S   S K  K  M   M 2     C
  T    SSS  K   K M   M 22222  CCCC
```

```
TTTTTTTTTT    SSSSSS    KK      KK  MM      MM    222222      CCCCCCCC
TTTTTTTTTT    SSSSSS    KK      KK  MM      MM    222222      CCCCCCCC
    TT       SS      SS  KK    KK    MMMM  MMMM  22      22  CC
    TT       SS      SS  KK    KK    MMMM  MMMM  22      22  CC
    TT       SS          KK  KK      MM  MM  MM        22    CC
    TT       SS          KK  KK      MM  MM  MM        22    CC
    TT         SSSSSS    KKKK        MM      MM      22      CC
    TT         SSSSSS    KKKK        MM      MM      22      CC
    TT               SS  KK  KK      MM      MM    22        CC
    TT               SS  KK  KK      MM      MM    22        CC
    TT       SS      SS  KK    KK    MM      MM  22          CC
    TT       SS      SS  KK    KK    MM      MM  22          CC
```