Table of contents

TSWIN -- TSX-Plus Display Windo MACRO V05.04 Monday 21-Dec-87 07:45
Table of contents

Table of contents

```
                                        .TITLE  TSWIN -- TSX-Plus Display Windows
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
                                        .CSECT  TSWIN
000000                          TSWIN:  .RAD50  /WIN/           ;Overlay region id
000000  110466
                                ;
                                ;  TSWIN is the TSX-Plus system overlay that provides display window support.
                                ;
                                ;  Copyright (c) 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee USA
                                ;  All rights reserved.
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  WINCHR,EMTWIN,WININI,WINREL,WINDSP
                                        .GLOBL  WINST,WINSF,WINPRT
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  LSW11,$V52EM,LNPRIM
                                        .GLOBL  LSTSL,LPRQ1,LPRQ2,LBRKCQ,NMFREQ,GETRTQ
                                        .GLOBL  IOQSIZ,CQ$R1,QCOMPL,VALADW,VALADB
                                        .GLOBL  P2$CXT,PRIVC2,$8BIT,LSW2,AW$PRT
                                        .GLOBL  RC$EXC,RC$FLG,LOTSIZ,LOTSPC,LOTNXT,LOTPNT
                                        .GLOBL  $CTRLO,LSW3,$RFRSH,LSW4,AW$SPN
                                        .GLOBL  $WDISP,LSW6,S$OTWT,LSTATE,RC$BLK,RC$CNT
                                        .GLOBL  $VNOTT,LSW,PCSPND,FRKPRI,FORCEX
                                        .GLOBL  DW$CSB,DW$CSR,DW$CSP,TCSBSZ,LWINDO
                                        .GLOBL  ESC,CSICHR,SS3CHR,AW$52,DW$AW,BUFCHR,AW$S52
                                        .GLOBL  KPAR6,VPAR6,OVRHC,INTPRI,PSW,PR7
                                        .GLOBL  DW$CCA,DW$COL,DW$CPL,DW$LIN,DW$SRB
                                        .GLOBL  DW$SRT,DW$LPP,DW$TLN,DW$LPT,DW$MAP
                                        .GLOBL  DW$RID,DW$JOB,DW$ID,RC$BAS,R.GID
                                        .GLOBL  R.GSIZ,R.GSTS,R.NAME,RS.CGR,RS.PVT
                                        .GLOBL  RS.GBL,EMTBLK,SETERR,EMTXIT,RS.EGR
                                        .GLOBL  DW$$SZ,CR,LF,AC$BLD,AC$BLK,AC$REV
                                        .GLOBL  CORUSR,AC$ULN,AL$DHT,AL$DHB,AL$DWD
                                        .GLOBL  BUFCHR,VMXWIN,VT52,VT2007,VT2008,LTRMTP
                                        .GLOBL  AW$200,AW$132,AW$INS,AW$ACK,AW$REV,AW$ORS
                                        .GLOBL  AW$AKM,AW$VCR,LSW7,$SLON,$SLKED,LCOL
                                        .GLOBL  AW$PRM,$DETCH,LSW,LSW6,$WDISP,AW$RPT
                                        .GLOBL  AW$SS,DW$GLS,DW$GLM,DW$GRM,AC$SET
                                        .GLOBL  DW$MSL,DW$NSL,AW$DDC
                                        .GLOBL  DW$GOM,DW$G1M,DW$G2M,DW$G3M
                                        .GLOBL  DW$SLN,DW$SCL,DW$SCA,WINTOP
```

```
                        ;-----------------------------------------------------------------------
                        ;  Macro definitions
                        ;
                        ;  Macros to enable and disable interrupts
                        ;
                              .MACRO  DISABL          ;Disable interrupts
                              BIS     #PR7,@#PSW
                              .ENDM   DISABL

                              .MACRO  ENABL           ;Enable interrupts
                              BIC     INTPRI,@#PSW
                              .ENDM   ENABL
                        ;
                        ;  Macro to call a system overlay
                        ;
                              .MACRO  OCALL   ENTADD
                              .IF     B,ENTADD
                              .ERROR  ;OCALL without entry address
                              .ENDC
                              CALL    OVRHC
                              .WORD   ENTADD
                              .ENDM   OCALL
                        ;
                        ;  Macros to map to screen buffer and restore mapping
                        ;
                              .MACRO  BUFMAP
                              DISABL
                              MOV     @#KPAR6,MAPHLD
                              MOV     DW$MAP(R2),@#KPAR6
                              .ENDM   BUFMAP

                              .MACRO  UNMAP
                              MOV     MAPHLD,@#KPAR6
                              ENABL
                              .ENDM   UNMAP
                        ;
                        ;  Macro to send a character to the terminal
                        ;
                              .MACRO  SEND    CHAR
                              MOVB    CHAR,R0
                              CALL    GENCHR
                              .ENDM   SEND

                        ;-----------------------------------------------------------------------
                        ;  Data areas and parameters
                        ;
 000073                 SEMI    =       73              ;Semicolon
                        ;
 000002  015050'        DWBAS:  .WORD   WINTOP          ;Pointer to 1st window control block
 000004  000000         DWEND:  .WORD   0               ;Pointer past last window control block
 000006  000000         MAPHLD: .WORD   0               ;Temp cell used by BUFMAP/UNMAP macros
 000010  110466         R50WIN: .RAD50  /WIN/
 000012  063344         R50PRT: .RAD50  /PRT/
 000014                 CSBUF:  .BLKB   16.             ;Cursor string work buffer
                              .EVEN
```

```
                                        .SBTTL  WINCHR -- Process next output character
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to process each character sent to the terminal
                                ;  by a running program.  It determines if the character is part of an
                                ;  escape sequence or is a regular character.
                                ;
                                ;  Inputs:
                                ;   R0 = Character being sent.
                                ;   R1 = Job index number.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Do not send this char to terminal now.
                                ;
000034  010046                  WINCHR: MOV     R0,-(SP)
000036  010246                          MOV     R2,-(SP)
000040  010546                          MOV     R5,-(SP)
000042  010005                          MOV     R0,R5           ;Carry character in R5
                                ;
                                ;  Get pointer to current window control block for job
                                ;
000044  016102  000000                  MOV     LWINDO(R1),R2   ;Get pointer to window control block
                                ;
                                ;  See if processing is suspended for this window
                                ;
000050  032762  000000 000000          BIT     #AW$SPN,DW$AW(R2);Is window processing suspended?
000056  001045                         BNE     8$              ;Br if yes -- Just pass through the char
                                ;
                                ;  Mask character to 7 bits unless 8 bit support has been enabled
                                ;
000060  032761  000000 000000          BIT     #$8BIT,LSW2(R1) ;Is 8 bit char support wanted?
000066  001002                         BNE     1$              ;Br if yes
000070  042705  177600                 BIC     #^C<177>,R5     ;Mask character to 7 bits
                                ;
                                ;  Ignore nulls
                                ;
000074  105705                  1$:    TSTB    R5              ;Is character null?
000076  001427                         BEQ     7$              ;Br if null
                                ;
                                ;  Determine if this character begins a terminal control sequence
                                ;
000100  004737  000510'                CALL    WCSCHK          ;Check for control sequence start
000104  103024                         BCC     7$              ;Br if start of control sequence
                                ;
                                ;  See if we are currently accepting a terminal control sequence
                                ;
000106  016200  000000                  MOV     DW$CSR(R2),R0   ;Are we processing a control sequence?
000112  001402                         BEQ     3$              ;Br if not
000114  004710                         CALL    (R0)            ;Call routine to process this character
000116  000417                         BR      7$              ;Finished with this character
                                ;
                                ;  If output is being directed to the terminal printer port, then
                                ;  don't update window image with this character.
                                ;
000120  032762  000000 000000  3$:     BIT     #AW$PRT,DW$AW(R2);Is output going to printer?
000126  001013                         BNE     7$              ;Br if yes -- Bypass window update
                                ;
                                ;  Determine if this is a normal or control character
```

```
                                ;
000130  010500                          MOV     R5,R0           ;Get the character
000132  042700  177600                  BIC     #^C<177>,R0     ;Mask to 7 bits
000136  020027  000037                  CMP     R0,#37          ;Is this a control character?
000142  101003                          BHI     4$              ;Br if not
000144  004737  000370'                 CALL    DOCTRL          ;Process a control character
000150  000402                          BR      7$
                                ;
                                ;  Process regular character
                                ;
000152  004737  000204'         4$:     CALL    REGCHR          ;Process a regular character
                                ;
                                ;  See if we should display this character now
                                ;
000156  032762  0000000 0000006 7$:     BIT     #AW$DDC,DW$AW(R2);Should we suppress char display?
000164  001402                          BEQ     8$              ;Br if not
                                ;
                                ;  We don't want to display this character
                                ;
000166  000261                          SEC                     ;Signal to not display the character
000170  000401                          BR      9$
                                ;
                                ;  We want to display this character
                                ;
000172  000241                  8$:     CLC                     ;Signal to display the character
                                ;
                                ;  Finished
                                ;
000174  012605                  9$:     MOV     (SP)+,R5
000176  012602                          MOV     (SP)+,R2
000200  012600                          MOV     (SP)+,R0
000202  000207                          RETURN
```

```
                                        .SBTTL  REGCHR -- Process regular characters
                                ;-----------------------------------------------------------------
                                ;  Process regular characters which are stored into the appropriate
                                ;  cell of the character matrix.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Character
                                ;
000204  010346                 REGCHR: MOV     R3,-(SP)
000206  010546                         MOV     R5,-(SP)
                                ;
                                ;  Get pointer to 1st character on current line
                                ;
000210  016203  0000000                 MOV     DW$LPT(R2),R3   ;Get pointer to 1st char on line
                                ;
                                ;  Get pointer to current character on the line
                                ;
000214  016200  0000000                 MOV     DW$COL(R2),R0   ;Get current column number
000220  005300                          DEC     R0              ;1st column is # 1
000222  006300                          ASL     R0              ;Two bytes used per column
000224  060003                          ADD     R0,R3           ;Get pointer to char byte for current col
                                ;
                                ;  Get number of character set for this character
                                ;
000226  116200  0000000                 MOVB    DW$GRM(R2),R0   ;Assume character is in GR
000232  032705  000200                  BIT     #200,R5         ;Is character in GR (8 bit character)
000236  001013                          BNE     1$              ;Br if yes
000240  116200  0000000                 MOVB    DW$GLM(R2),R0   ;Determine which set GL is mapped to
000244  032762  0000000 0000000        BIT     #AW$SS,DW$AW(R2);Are we single-shifting this character?
000252  001405                          BEQ     1$              ;Br if not
000254  116200  0000000                 MOVB    DW$GLS(R2),R0   ;Get single-shift mapping for char
000260  042762  0000000 0000000        BIC     #AW$SS,DW$AW(R2);Reset single-shift flag
000266  060200                  1$:     ADD     R2,R0           ;Add base address of window control block
000270  116000  0000000                 MOVB    DW$GOM(R0),R0   ;Get char set for this character
000274  042705  177600                  BIC     #^C<177>,R5     ;Mask character to 7 bits
                                ;
                                ;  Get character attribute flags (bold, blinking, etc.)
                                ;
000300  156200  0000000                 BISB    DW$CCA(R2),R0   ;Add character attribute flags
                                ;
                                ;  Store character and attribute into that cell
                                ;
000304                                  BUFMAP                  ;;;Map to screen buffer
000326  110523                          MOVB    R5,(R3)+        ;;;Store character
000330  110023                          MOVB    R0,(R3)+        ;;;Store character attribute flags
000332                                  UNMAP                   ;Restore mapping
                                ;
                                ;  Advance column number unless we are at the end of the line
                                ;
000346  026262  0000000 0000000        CMP     DW$COL(R2),DW$CPL(R2);Are we at the end of the line now?
000354  103002                          BHIS    9$              ;Br if at end of the line now
000356  005262  0000000                 INC     DW$COL(R2)      ;Advance to next column
                                ;
                                ;  Finished
                                ;
000362  012605                  9$:     MOV     (SP)+,R5
```

```
     58 000364  012603                          MOV     (SP)+,R3
     59 000366  000207                          RETURN
```

```
                                        .SBTTL  DOCTRL -- Process control characters
                                ;-----------------------------------------------------------------------
                                ;  This routine is called when the character being output is a
                                ;  control character (000 - 037).
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Character
                                ;
000370                          DOCTRL:
                                ;
                                ;  Call processing routine for this character
                                ;
000370  010500                         MOV     R5,R0           ;Get the character
000372  042700  177600                 BIC     #^C<177>,R0     ;Mask to 7 bits
000376  006300                         ASL     R0              ;Convert to word table index
000400  004770  000406'                CALL    @CCRTN(R0)      ;Call appropriate processing routine
                                ;
                                ;  Finished
                                ;
000404  000207                         RETURN

                                ;-----------------------------------------------------------------------
                                ;  Branch table for control character processing routines
                                ;
000406  000506'                 CCRTN:  .WORD   CCEXIT          ; 000 - NUL
000410  000506'                         .WORD   CCEXIT          ; 001 - SOH
000412  000506'                         .WORD   CCEXIT          ; 002 - STX
000414  000506'                         .WORD   CCEXIT          ; 003 - ETX
000416  000506'                         .WORD   CCEXIT          ; 004 - EOT
000420  000506'                         .WORD   CCEXIT          ; 005 - ENQ
000422  000506'                         .WORD   CCEXIT          ; 006 - ACK
000424  000506'                         .WORD   CCEXIT          ; 007 - BEL
000426  002230'                         .WORD   TCBS            ; 010 - BS
000430  002246'                         .WORD   TCHT            ; 011 - HT
000432  002454'                         .WORD   TCLF            ; 012 - LF
000434  002454'                         .WORD   TCLF            ; 013 - VT
000436  002454'                         .WORD   TCLF            ; 014 - FF
000440  002306'                         .WORD   TCCR            ; 015 - CR
000442  003516'                         .WORD   TCSO            ; 016 - SO
000444  003526'                         .WORD   TCSI            ; 017 - SI
000446  000506'                         .WORD   CCEXIT          ; 020 - DLE
000450  000506'                         .WORD   CCEXIT          ; 021 - DC1
000452  000506'                         .WORD   CCEXIT          ; 022 - DC2
000454  000506'                         .WORD   CCEXIT          ; 023 - DC3
000456  000506'                         .WORD   CCEXIT          ; 024 - DC4
000460  000506'                         .WORD   CCEXIT          ; 025 - NAK
000462  000506'                         .WORD   CCEXIT          ; 026 - SYN
000464  000506'                         .WORD   CCEXIT          ; 027 - ETB
000466  000506'                         .WORD   CCEXIT          ; 030 - CAN
000470  000506'                         .WORD   CCEXIT          ; 031 - EM
000472  000204'                         .WORD   REGCHR          ; 032 - SUB (ctrl-Z)
000474  000506'                         .WORD   CCEXIT          ; 033 - ESC
000476  000506'                         .WORD   CCEXIT          ; 034 - FS
000500  000506'                         .WORD   CCEXIT          ; 035 - GS
000502  000506'                         .WORD   CCEXIT          ; 036 - RS
000504  000506'                         .WORD   CCEXIT          ; 037 - US
```

```
                                ;
                                ;  Immediate return for most control characters
                                ;
000506  000207                 CCEXIT: RETURN
```

```
                                        .SBTTL  WCSCHK -- Check for start of control sequence
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to determine if the current character begins
                                ;  a terminal control sequence.
                                ;
                                ;  Inputs:
                                ;   R2 = Window control block.
                                ;   R5 = Current character
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Start of control sequence
                                ;   C-flag set     ==> Not start of control sequence
                                ;
000510                          WCSCHK:
                                ;
                                ;  See if this is the start of an escape sequence
                                ;
000510  120527  0000000                 CMPB    R5,#ESC         ;Is this character escape?
000514  001003                          BNE     1$              ;Br if not
000516  004737  000546'                 CALL    WCPESC          ;Begin escape sequence
000522  000405                          BR      8$
                                ;
                                ;  See if this character is CSI, beginning of VT200 control sequence
                                ;
000524  120527  0000000         1$:     CMPB    R5,#CSICHR      ;CSI character?
000530  001004                          BNE     7$              ;Br if not
000532  004737  000600'                 CALL    WCPCSI          ;Start VT200 control sequence
                                ;
                                ;  We began a control sequence
                                ;
000536  000241                  8$:     CLC                     ;Signal control sequence began
000540  000401                          BR      9$
                                ;
                                ;  We did not begin a control sequence
                                ;
000542  000261                  7$:     SEC                     ;Signal that not control sequence
                                ;
                                ;  Finished
                                ;
000544  000207                  9$:     RETURN
```

```
                                        .SBTTL  WCPESC -- Escape processing
                                ;-----------------------------------------------------------------------
                                ;  Process an Escape character
                                ;
                                ;  Inputs:
                                ;   R2 = Window control block
                                ;   R5 = Escape character
                                ;
000546                          WCPESC:
                                ;
                                ;  Initialize control sequence buffer pointer
                                ;
000546  005062  0000000                 CLR     DW$CSP(R2)      ;Say no chars accrued yet
                                ;
                                ;  Set address of routine to be called to process 1st character after
                                ;  escape.
                                ;
000552  012700  000624'                 MOV     #EPCH1,R0       ;Routine for 1st character after escape
000556  032762  000000C 0000000         BIT     #<AW$52!AW$S52>,DW$AW(R2);Is terminal in VT52 emulation mode?
000564  001402                          BEQ     1$              ;Br if not
000566  012700  000710'                 MOV     #EP52,R0        ;Set routine for VT52 mode
000572  010062  0000000         1$:     MOV     R0,DW$CSR(R2)   ;Set address of processing routine
000576  000207                          RETURN
                                ;-----------------------------------------------------------------------
                                ;  Received a CSI control character for a VT200 terminal.
                                ;
                                ;  Inputs:
                                ;    R2 = Window control block address
                                ;
000600  012762  000624' 0000000 WCPCSI:  MOV     #EPCH1,DW$CSR(R2) ;Set address of routine for next char
000606  005062  0000000                 CLR     DW$CSP(R2)      ;Say no chars accrued yet
000612  112700  000133                  MOVB    #'[,R0          ;Treat CSI char like ESC [
000616  004737  001136'                 CALL    ACRCS           ;Store "[" as 1st char of sequence
000622  000207                          RETURN
```

```
                                        .SBTTL  EPCH1  -- Accrue a terminal control sequence
                                ;-----------------------------------------------------------------------
                                ;  EPCH1 is called to accrue a VT100 or VT200 control sequence.
                                ;
                                ;  There are two types of "control" sequences:
                                ;
                                ;   Escape sequence format:   ESC I...I F
                                ;       I = Intermediate character in the range 040 to 057
                                ;       F = Final character in the range 060 to 176
                                ;
                                ;   Control sequence format:   ESC P...P I...I F
                                ;       CSI = CSI character or ESC [
                                ;       P   = Parameter values in the range 060 to 077
                                ;       I   = Intermediate characters in the range 040 to 057
                                ;       F   = Final character in the range 100 to 176
                                ;
                                ;  Inputs:
                                ;   R2 = Address of window control block.
                                ;   R5 = Received character
                                ;
000624  010546                  EPCH1:  MOV     R5,-(SP)
000626  042705  177600                  BIC     #^C<177>,R5     ;Mask character to 7 bits
                                ;
                                ;  Add received character to end of control sequence already received
                                ;
000632  010500                          MOV     R5,R0           ;Get received character
000634  004737  001136'                 CALL    ACRCS           ;Store into control sequence buffer
                                ;
                                ;  Determine if this character terminates the control sequence
                                ;
000640  120527  000060                  CMPB    R5,#60          ;Is this the terminating character?
000644  103417                          BLO     9$              ;Br if not
000646  120527  000176                  CMPB    R5,#176         ;Is this the terminating character?
000652  101014                          BHI     9$              ;Br if not
000654  120527  000133                  CMPB    R5,#'[          ;Is this part of string header?
000660  001411                          BEQ     9$              ;Br if string header
000662  120527  000100                  CMPB    R5,#100         ;100 to 176 always terminates
000666  103004                          BHIS    1$
000670  126227  0000000 000133          CMPB    DW$CSB(R2),#'[  ;Is this an escape or control sequence?
000676  001402                          BEQ     9$              ;Br if control sequence. Only 100-176 term
                                ;
                                ;  This is the final character of the control sequence
                                ;
000700  004737  001174'          1$:     CALL    CSFIN           ;End of control sequence
                                ;
                                ;  Finished
                                ;
000704  012605                  9$:     MOV     (SP)+,R5
000706  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Accrue VT52 control sequences.
                                ;  All VT52 control sequences consist of ESC followed by a single character
                                ;  except for the cursor addressing sequence ESC Y line col.
                                ;  The VT52 sequences are converted to VT52 compatible sequences.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of window control block
                                ;   R5 = Received character
                                ;
000710  010546                 EP52:   MOV     R5,-(SP)
000712  042705  177600                 BIC     #^C<177>,R5     ;Mask character to 7 bits
                                ;
                                ;  If this character is a letter, insert "[" in front of it.
                                ;
000716  120527  000101                 CMPB    R5,#'A          ;Letter
000722  103416                 BLO     2$              ;Br if not
000724  120527  000132                 CMPB    R5,#'Z          ;Letter
000730  101013                 BHI     2$              ;Br if not
000732  012700  000133                 MOV     #'[,R0          ;Store "["
000736  004737  001136'                CALL    ACRCS
                                ;
                                ;  Determine if this is the cursor addressing function
                                ;
000742  120527  000131                 CMPB    R5,#'Y          ;Cursor addressing function?
000746  001004                 BNE     2$              ;Br if not
000750  012762  000776' 000000G        MOV     #EP52L,DW$CSR(R2);Set routine to process next char
000756  000405                 BR      9$
                                ;
                                ;  This character terminates the control sequence.
                                ;
000760  010500                 2$:     MOV     R5,R0           ;Get received character
000762  004737  001136'                CALL    ACRCS           ;Add to string we are accruing
000766  004737  001174'                CALL    CSFIN           ;This terminates the control sequence
                                ;
                                ;  Finished
                                ;
000772  012605                 9$:     MOV     (SP)+,R5
000774  000207                 RETURN

                                ;-----------------------------------------------------------------------
                                ;  Accept the character which represents the line number in a VT52
                                ;  cursor addressing sequence.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of window control block
                                ;   R5 = Received character (whose binary value is line number + 037)
                                ;
000776  042705  177600          EP52L:  BIC     #^C<177>,R5     ;Mask character to 7 bits
001002  120527  000070                 CMPB    R5,#70          ;Octal 70 ==> Remain on current line
001006  103404                 BLO     1$              ;Br if new line specified
001010  016205  000000G                MOV     DW$LIN(R2),R5   ;Get current line number
001014  062705  000037                 ADD     #37,R5          ;Convert to ascii character
001020  004737  001066'         1$:     CALL    CV52            ;Convert char to digit string
001024  112700  000073                 MOVB    #SEMI,R0        ;Get parameter separator character
001030  004737  001136'                CALL    ACRCS           ;Store after line number value
001034  012762  001044' 000000G        MOV     #EP52C,DW$CSR(R2);Set routine to process column number char
```

```
                        RETURN

                ;-------------------------------------------------------------------
                ;  Accept the character which represents the column number in a VT52
                ;  cursor addressing sequence.
                ;
                ;  Inputs:
                ;   R2 = Address of window control block
                ;   R5 = Received character (whose binary value is column number + 037)
                ;
EP52C:  CALL    CV52            ;Convert char to digit string
        MOVB    #'H,R0          ;Store terminating character
        CALL    ACRCS
        CALL    CSFIN           ;This terminates the control sequence
        RETURN

                ;-------------------------------------------------------------------
                ;  Convert a received character whose binary value represents a line
                ;  number or a column number in a VT52 cursor addressing sequence
                ;  into an ascii digit string and store into the accrued control
                ;  sequence.
                ;
                ;  Inputs:
                ;   R2 = Address of window control block
                ;   R5 = Received character whose binary value - 37 is the stored result
                ;
CV52:   MOV     R4,-(SP)
        MOV     R5,-(SP)
                ;
                ;  Subtract bias of 37
                ;
        SUB     #37,R5          ;Remove bias from character value
                ;
                ;  Convert to digit string
                ;
        CLR     R4              ;Clear high-order for divide
        DIV     #10.,R4         ;Split into two decimal digits
        ADD     #'0,R4          ;Form high-order ascii digit
        MOV     R4,R0           ;Get the character
        CALL    ACRCS           ;Store into control string
        ADD     #'0,R5          ;Form low-order ascii digit
        MOV     R5,R0           ;Get the character
        CALL    ACRCS           ;Store into control string
                ;
                ;  Finished
                ;
        MOV     (SP)+,R5
        MOV     (SP)+,R4
        RETURN
```

```
                                        .SBTTL  ACRCS  -- Store a character into control seq buffer
                                ;-----------------------------------------------------------------
                                ;  This routine is called to add another character to the end of the
                                ;  accrued control sequence in the DW$CSB buffer area.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be added to end of string
                                ;   R2 = Address of window descriptor block
                                ;
001136  010346                  ACRCS:  MOV     R3,-(SP)
001140  042700  177600                  BIC     #^C<177>,R0     ;Mask character to 7 bits
                                ;
                                ;  Make sure we don't overflow the control sequence buffer
                                ;
001144  016203  0000000                 MOV     DW$CSP(R2),R3   ;Get # chars in buffer now
001150  020327  1777770                 CMP     R3,#TCSBSZ-1    ;Is buffer full already?
001154  103005                          BHIS    9$              ;Br if yes -- Ignore this character
                                ;
                                ;  Store character into buffer
                                ;
001156  060203                          ADD     R2,R3           ;Point to next char pos in buffer
001160  110063  0000000                 MOVB    R0,DW$CSB(R3)   ;Store char into buffer
001164  005262  0000000                 INC     DW$CSP(R2)      ;Increment character count
                                ;
                                ;  Finished
                                ;
001170  012603                  9$:     MOV     (SP)+,R3
001172  000207                          RETURN
```

```
                                        .SBTTL  CSFIN  -- Process received control sequence
                                ;-----------------------------------------------------------------
                                ;  We have finished accruing a control sequence, process it.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of window control block.
                                ;
001174  010346                 CSFIN:  MOV     R3,-(SP)
001176  010446                         MOV     R4,-(SP)
001200  010546                         MOV     R5,-(SP)
                                ;
                                ;  Store null at end of control sequence string
                                ;
001202  010203                          MOV     R2,R3           ;Get address of window control block
001204  066203  0000000                 ADD     DW$CSP(R2),R3   ;Point to next char pos in buffer
001210  105063  0000000                 CLRB    DW$CSB(R3)      ;Store null at end of string
                                ;
                                ;  Move control sequence to work buffer and remove any parameter characters
                                ;
001214  012705  000014'                 MOV     #CSBUF,R5       ;Point to work buffer
001220  010203                          MOV     R2,R3           ;Get address of buffer with full string
001222  062703  0000000                 ADD     #DW$CSB,R3
001226  121327  000133                  CMPB    (R3),#'[        ;Do we need to exclude parameter values?
001232  001014                          BNE     3$              ;Br if not
                                ;
                                ;  This is a control sequence.  Ignore parameter value characters in the
                                ;  range 060 to 076.
                                ;
001234  112300                 1$:     MOVB    (R3)+,R0        ;Get next char from input string
001236  001410                         BEQ     4$              ;Br if hit end of string
001240  120027  000060                  CMPB    R0,#060         ;Is this a parameter character?
001244  103403                          BLO     2$              ;Br if not
001246  120027  000076                  CMPB    R0,#076         ;Parameter character?
001252  101770                         BLOS    1$              ;Br if parameter character
001254  110025                 2$:     MOVB    R0,(R5)+        ;Store char into work buffer
001256  000766                         BR      1$              ;Keep moving
001260  105025                 4$:     CLRB    (R5)+           ;Store null at end of string
001262  000402                         BR      5$
                                ;
                                ;  This is an escape sequence.  We don't have to remove parameter values.
                                ;
001264  112325                 3$:     MOVB    (R3)+,(R5)+     ;Move char to work buffer
001266  001376                         BNE     3$              ;Loop till null moved
                                ;
                                ;  The control sequence is now in the work buffer (CSBUF) with any parameter
                                ;  value strings removed.
                                ;  Begin loop to try to identify the control sequence.
                                ;
001270  012704  001432'         5$:     MOV     #CSTBLL,R4      ;Point to 1st set of control sequences
001274  126527  177776  000140           CMPB    -2(R5),#140     ;Is final char of sequence less than 140?
001302  103002                          BHIS    10$             ;Br if not
001304  012704  001560'                 MOV     #CSTBLU,R4      ;Point to control seq table for final < 140
                                ;
                                ;  Compare accrued control sequence with sequence stored in table
                                ;
001310  012703  000014'         10$:    MOV     #CSBUF,R3       ;Point to accrued control sequence
001314  122324                  12$:    CMPB    (R3)+,(R4)+     ;Compare the strings
```

```
                                        BNE     13$             ;Br if mismatch
001320  105764  177777                  TSTB    -1(R4)          ;Was that the end of both strings?
001324  001373                          BNE     12$             ;Br if not
001326  000412                          BR      14$             ;We have a match
                                ;
                                ;  Strings do not match -- Move on to next table entry
                                ;
001330  005304                  13$:    DEC     R4              ;Point to last char compared in table
001332  105724                  15$:    TSTB    (R4)+           ;Search for null at end of asciz string
001334  001376                          BNE     15$             ;Loop till null found
001336  062704  000003                  ADD     #3,R4           ;Bound up and skip over following word
001342  042704  000001                  BIC     #1,R4           ;Get to word boundary
001346  005714                          TST     (R4)            ;Reached end of table?
001350  001357                          BNE     10$             ;Loop if not
001352  000417                          BR      20$             ;Control sequence is not in our table
                                ;
                                ;  We found the control sequence in our table.
                                ;  Call processing routine.
                                ;  Set R5 to point to start of any possible parameter string in control seq.
                                ;
001354  010205                  14$:    MOV     R2,R5           ;Point to window control block
001356  062705  0000010                 ADD     #DW$CSB+1,R5    ;Point to start of possible parameter string
001362  005204                          INC     R4              ;Bound up to next word
001364  042704  000001                  BIC     #1,R4           ;Get to word boundary
001370  011404                          MOV     (R4),R4         ;Get address of routine to call
001372  032762  0000000 0000000         BIT     #AW$PRT,DW$AW(R2);Are we directing output to printer port?
001400  001403                          BEQ     16$             ;Br if not
001402  020427  004666'                 CMP     R4,#TCPRT       ;Is this printer control operation?
001406  001001                          BNE     20$             ;Br if not -- Ignore till printer turned off
001410  004714                  16$:    CALL    (R4)            ;Call processing routine
                                ;
                                ;  Finished with control sequence
                                ;
001412  005062  0000000         20$:    CLR     DW$CSP(R2)      ;No accrued chars in control sequence
001416  005062  0000000                 CLR     DW$CSR(R2)      ;Not accruing a control sequence now
001422  012605                          MOV     (SP)+,R5
001424  012604                          MOV     (SP)+,R4
001426  012603                          MOV     (SP)+,R3
001430  000207                          RETURN
```

```
                        ;------------------------------------------------------------------------
                        ;  Generate table of terminal control sequences and associated processing
                        ;  routines.
                        ;
                                .MACRO  ESCSEQ  STRING,RTN
                                .ASCIZ  /STRING/
                                .EVEN
                                .WORD   RTN
                                .ENDM   ESCSEQ
                        ;
                        ;  Table for control sequences whose final character is in the
                        ;  range 140 to 177
                        ;
001432                  CSTBLL:
001432                          ESCSEQ  <[f>,TCCADR     ;Cursor addressing
001440                          ESCSEQ  <["p>,TC100     ;Select VT100/VT200 mode
001446                          ESCSEQ  <[h>,TCSAA      ;Set ANSI terminal attribute
001454                          ESCSEQ  <[l>,TCRAA      ;Reset ANSI terminal attribute
001462                          ESCSEQ  <[?h>,TCSDA     ;Set DEC terminal attribute
001470                          ESCSEQ  <[?l>,TCRDA     ;Reset DEC terminal attribute
001476                          ESCSEQ  <[i>,TCPRT      ;Printer control code
001504                          ESCSEQ  <[m>,TCSCA      ;Set character attribute
001512                          ESCSEQ  <[r>,TCSSR      ;Set scrolling region
001520                          ESCSEQ  <[!p>,TCREST    ;Soft reset
001526                          ESCSEQ  <c>,TCREST      ;Hard reset
001532                          ESCSEQ  <~>,TCLS1R      ;Lock shift G1 as GR character set
001536                          ESCSEQ  <n>,TCLS2       ;Lock shift G2 as GL character set
001542                          ESCSEQ  <o>,TCLS3       ;Lock shift G3 as GL character set
001546                          ESCSEQ  <}>,TCLS2R      ;Lock shift G2 as GR character set
001552                          ESCSEQ  <|>,TCLS3R      ;Lock shift G3 as GR character set
001556  000000                  .WORD   0               ;End of table
                        ;
                        ;  Table for control sequences whose final character is in the
                        ;  range 60 to 137
                        ;
001560                  CSTBLU:
001560                          ESCSEQ  <[A>,TCUP       ;Move cursor up
001566                          ESCSEQ  <[B>,TCDOWN     ;Move cursor down
001574                          ESCSEQ  <[C>,TCRIT      ;Move cursor right
001602                          ESCSEQ  <[D>,TCLEFT     ;Move cursor left
001610                          ESCSEQ  <[F>,TCGODS     ;Select graphic char set for VT52
001616                          ESCSEQ  <[G>,TCGOAS     ;Select normal char set for VT52
001624                          ESCSEQ  <[H>,TCCADR     ;Cursor addressing
001632                          ESCSEQ  <[I>,TCIXUP     ;Reverse index (VT52)
001640                          ESCSEQ  <[J>,TCPERS     ;Erase within a page
001646                          ESCSEQ  <[K>,TCLERS     ;Erase within a line
001654                          ESCSEQ  <[L>,TCINSL     ;Insert lines
001662                          ESCSEQ  <[M>,TCDELL     ;Delete lines
001670                          ESCSEQ  <D>,TCLF        ;Index (Move cursor down one line)
001674                          ESCSEQ  <E>,TCESCE      ;Move to 1st column of next line
001700                          ESCSEQ  <M>,TCIXUP      ;Reverse index (move up 1 line)
001704                          ESCSEQ  <N>,TCSS2       ;Single shift next char to G2
001710                          ESCSEQ  <O>,TCSS3       ;Single shift next char to G3
001714                          ESCSEQ  <(A>,TCGOUK     ;Designate G0 as UK
001722                          ESCSEQ  <(B>,TCGOAS     ;Designate G0 as ascii
001730                          ESCSEQ  <(0>,TCGOGR     ;Designate G0 as graphics
001736                          ESCSEQ  <)A>,TCG1AS     ;Designate G1 as UK
```

```
                                                ESCSEQ  <(B>,TCG1AS    ;Designate G1 as ascii
                                                ESCSEQ  <(0>,TCG1GR    ;Designate G1 as graphics
                                                ESCSEQ  <*B>,TCG2AS    ;Designate G2 as ascii
                                                ESCSEQ  <*0>,TCG2GR    ;Designate G2 as graphics
                                                ESCSEQ  <+B>,TCG3AS    ;Designate G3 as ascii
                                                ESCSEQ  <+0>,TCG3GR    ;Designate G3 as graphics
                                                ESCSEQ  <#3>,TCN3      ;Line is top half of double-high line
                                                ESCSEQ  <#4>,TCN4      ;Line is bottom half of double high
                                                ESCSEQ  <#5>,TCN5      ;Line is single width
                                                ESCSEQ  <#6>,TCN6      ;Line is double width
                                                ESCSEQ  <7>,TCCPSV     ;Save cursor position
                                                ESCSEQ  <8>,TCCPRS     ;Restore cursor position
                                                ESCSEQ  <=>,TCAKM      ;Select application keypad mode
002054      050     074     000                 .ASCIZ  /(/<74>        ;"(<"
                                                .EVEN
002060  003354'                                 .WORD   TCG0DS         ;Designate G0 as DEC supplemental graphics
002062      051     074     000                 .ASCIZ  /)/<74>        ;")<"
                                                .EVEN
002066  003412'                                 .WORD   TCG1DS         ;Designate G1 as DEC supplemental graphics
002070      052     074     000                 .ASCIZ  /*/<74>        ;"*<"
                                                .EVEN
002074  003450'                                 .WORD   TCG2DS         ;Designate G2 as DEC supplemental graphics
002076      053     074     000                 .ASCIZ  /+/<74>        ;"+<"
                                                .EVEN
002102  003476'                                 .WORD   TCG3DS         ;Designate G3 as DEC supplemental graphics
002104      074     000                         .ASCIZ  <74>           ;Select ANSI mode with "<" character
002106  004466'                                 .WORD   TC100          ;Select ANSI mode
002110      076     000                         .ASCIZ  <76>           ;Generate entry with ">" character
002112  004232'                                 .WORD   TCNKM          ;Select numeric keypad mode
002114  000000                                  .WORD   0              ;End of table
```

```
                                        .SBTTL  CSPARM -- Extract parameter value from control string
                                ;-----------------------------------------------------------------------
                                ;  CSPARM obtains the next parameter value from the control string.
                                ;  If there is no parameter value, 0 is returned as the value.
                                ;  Non-parameter characters are skipped over.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer into control string where parameter may start.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> No more parameter values.
                                ;   R0 = Accrued parameter value.
                                ;   R5 = Points past end of parameter value.
                                ;
002116  010146                  CSPARM: MOV     R1,-(SP)
002120  005001                          CLR     R1              ;Accrue the value in R1
                                ;
                                ;  Scan up to the start of a parameter value
                                ;
002122  112500                  1$:     MOVB    (R5)+,R0        ;Get next char from control seq
002124  001430                          BEQ     5$              ;Br if hit end of the string
002126  120027  000073                  CMPB    R0,#SEMI        ;Semicolon is value separator
002132  001432                          BEQ     7$              ;Br if no value specified for this param
002134  120027  000071                  CMPB    R0,#'9          ;Is this a parameter character?
002140  101370                          BHI     1$              ;Br if not
002142  120027  000060                  CMPB    R0,#'0
002146  103765                          BLO     1$              ;Br if not digit
                                ;
                                ;  We found the start of a parameter value.  Accrue it.
                                ;
002150  000412                          BR      4$              ;Enter loop in middle
002152  112500                  3$:     MOVB    (R5)+,R0        ;Get next char of parameter
002154  001420                          BEQ     2$              ;Br if hit null at end of string
002156  120027  000071                  CMPB    R0,#'9          ;Is this char part of parameter?
002162  101016                          BHI     7$              ;Br if not
002164  120027  000060                  CMPB    R0,#'0
002170  103413                          BLO     7$              ;Br if not
002172  070127  000012                  MUL     #10.,R1         ;Multiply previous value by 10.
002176  162700  000060          4$:     SUB     #'0,R0          ;Convert ascii digit to binary value
002202  060001                          ADD     R0,R1           ;Add into previous value
002204  000762                          BR      3$              ;Keep accruing
                                ;
                                ;  There are no more parameter values
                                ;
002206  005305                  5$:     DEC     R5              ;Point to string terminator
002210  005000                          CLR     R0              ;Return 0 as "value"
002212  000261                          SEC                     ;Signal that there are no more parameters
002214  000403                          BR      9$
                                ;
                                ;  Finished getting the parameter value
                                ;
002216  005305                  2$:     DEC     R5              ;Point to parameter terminator
002220  010100                  7$:     MOV     R1,R0           ;Get accrued value
002222  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
```

```
     58 002224  012601                    7$:     MOV     (SP)+,R1
     59 002226  000207                            RETURN
```

```
      1                                       .SBTTL  TCBS   -- Process backspace character
      2                                 ;-----------------------------------------------------------------------
      3                                 ;  Process a backspace character.
      4                                 ;
      5                                 ;  Inputs:
      6                                 ;   R2 = Pointer to window control block
      7                                 ;
      8 002230                          TCBS:
      9                                 ;
     10                                 ;  Set column number back one unless we are already at column one
     11                                 ;
     12 002230  026227  0000000 000001          CMP     DW$COL(R2),#1   ;Are we already at column 1?
     13 002236  001402                          BEQ     9$              ;Br if yes
     14 002240  005362  0000000                 DEC     DW$COL(R2)      ;Move column number back one
     15                                 ;
     16                                 ;  Finished
     17                                 ;
     18 002244  000207                  9$:     RETURN
```

```
                                        .SBTTL  TCHT   -- Process horizontal tab character
                                ;-----------------------------------------------------------------------
                                ;  Process horizontal tab character.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
002246                          TCHT:
                                ;
                                ;  Get current column number and advance to next multiple of 8
                                ;
002246  016200  0000000                 MOV     DW$COL(R2),R0   ;Get current column number
002252  005300                          DEC     R0              ;Make 1st column be # 0
002254  062700  000010                  ADD     #8.,R0          ;Advance to next tab stop
002260  042700  000007                  BIC     #7,R0           ;Bound to next multiple of 8
002264  005200                          INC     R0              ;Make 1st column be # 1
                                ;
                                ;  Make sure we don't go beyond end of line
                                ;
002266  020062  0000000                 CMP     R0,DW$CPL(R2)   ;Compare with # columns per line
002272  101402                          BLOS    1$              ;Br if within line
002274  016200  0000000                 MOV     DW$CPL(R2),R0   ;Force to end of line
002300  010062  0000000         1$:     MOV     R0,DW$COL(R2)   ;Set new column number
                                ;
                                ;  Finished
                                ;
002304  000207                          RETURN
```

```
      1                                             .SBTTL  TCCR   -- Process carriage return
      2                                       ;-----------------------------------------------------------------
      3                                       ;  Process carriage return, move to 1st char of current line.
      4                                       ;
      5                                       ;  Inputs:
      6                                       ;   R2 = Pointer to window control block.
      7                                       ;
      8 002306  012762  000001  000000G TCCR:   MOV     #1,DW$COL(R2)   ;Say current column = 1
      9                                       ;
     10                                       ;  Finished
     11                                       ;
     12 002314  000207                        RETURN
```

```
                                        .SBTTL  TCUP   -- Move cursor up specified number of lines
                                ;-----------------------------------------------------------------------
                                ;  Move the cursor up the page the specified number of lines.
                                ;  Do not scroll if we reach the bottom of the page.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter value in control sequence
                                ;
002316                          TCUP:
                                ;
                                ;  Get count of number of lines to move up
                                ;
002316  004737  002116'                 CALL    CSPARM          ;Accrue parameter value
002322  005700                          TST     R0              ;Was specified value 0?
002324  001001                          BNE     1$              ;Br if not
002326  005200                          INC     R0              ;Treat 0 like 1
                                ;
                                ;  Move up specified number of lines
                                ;
002330  005400                  1$:     NEG     R0              ;Get negative number of lines to move
002332  066200  0000000                 ADD     DW$LIN(R2),R0   ;Compute new line
002336  003002                          BGT     2$              ;Br if didn't go past top of page
002340  012700  000001                  MOV     #1,R0           ;Constrain to top of page
002344  026262  0000000 0000000 2$:     CMP     DW$LIN(R2),DW$SRT(R2);Were we already above top of scroll rgn?
002352  103405                          BLO     3$              ;Br if yes -- Don't constrain
002354  020062  0000000                 CMP     R0,DW$SRT(R2)   ;Would this put us above top of scroll regn?
002360  103002                          BHIS    3$              ;Br if not
002362  016200  0000000                 MOV     DW$SRT(R2),R0   ;Constrain to top of scroll region
002366  004737  006702'          3$:     CALL    SETLIN          ;Set new line for cursor
                                ;
                                ;  Finished
                                ;
002372  000207                          RETURN
```

```
                                        .SBTTL  TCDOWN -- Move cursor down specified number of lines
                                ;-----------------------------------------------------------------------
                                ;  Move the cursor down the page the specified number of lines.
                                ;  Do not scroll if we reach the bottom of the page.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter value in control sequence
                                ;
002374                          TCDOWN:
                                ;
                                ;  Get count of number of lines to move down
                                ;
002374  004737  002116'                 CALL    CSPARM          ;Accrue parameter value
002400  005700                         TST     R0              ;Was specified value 0?
002402  001001                         BNE     1$              ;Br if not
002404  005200                         INC     R0              ;Treat 0 like 1
                                ;
                                ;  Move down specified number of lines
                                ;
002406  066200  000000G        1$:     ADD     DW$LIN(R2),R0   ;Compute new line
002412  020062  000000G                CMP     R0,DW$LPP(R2)   ;Would this go past end of page?
002416  101402                         BLOS    3$              ;Br if not
002420  016200  000000G                MOV     DW$LPP(R2),R0   ;Constrain to bottom of page
                                ;
                                ;  If we are currently with the scroll region, don't move out
                                ;
002424  026262  000000G 000000G 3$:     CMP     DW$LIN(R2),DW$SRB(R2);Were we already past end of scroll?
002432  101005                         BHI     2$              ;Br if past end -- Don't constrain
002434  020062  000000G                CMP     R0,DW$SRB(R2)   ;Did we go past end of scroll region?
002440  101402                         BLOS    2$              ;Br if not
002442  016200  000000G                MOV     DW$SRB(R2),R0   ;Stop at bottom of scroll region
002446  004737  006702'        2$:     CALL    SETLIN          ;Set new line for cursor
                                ;
                                ;  Finished
                                ;
002452  000207                         RETURN
```

```
                                        .SBTTL  TCLF    -- Process Line Feed character
                                ;-----------------------------------------------------------------
                                ;  Process a line feed character.  Move down to the next line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
002454                          TCLF:
                                ;
                                ;  If we are not at the bottom line of a scrolling region, simply
                                ;  advance the line number.
                                ;
002454  016200  0000000                 MOV     DW$LIN(R2),R0   ;Get the current line number
002460  020062  0000000                 CMP     R0,DW$SRB(R2)   ;Are we at bottom of scrolling region?
002464  001411                          BEQ     1$              ;Br if yes
002466  005200                          INC     R0              ;Increment the line number
002470  020062  0000000                 CMP     R0,DW$LPP(R2)   ;Are we past the end of the page?
002474  101402                          BLOS    2$              ;Br if not
002476  016200  0000000                 MOV     DW$LPP(R2),R0   ;Constrain to bottom of page
002502  004737  006702'         2$:     CALL    SETLIN          ;Set this as current line
002506  000402                          BR       9$
                                ;
                                ;  We are at the bottom line of a scroll region.
                                ;  We must scroll up one line.
                                ;
002510  004737  005146'         1$:     CALL    SCRLUP          ;Scroll up one line
                                ;
                                ;  Finished
                                ;
002514  000207                  9$:     RETURN
```

```
      1                                       .SBTTL  TCESCE -- Move to 1st column of next line
      2                                ;-----------------------------------------------------------------------
      3                                ;  Move to first column of next line.  Equivalent to carriage-return,
      4                                ;  line-feed.
      5                                ;
      6                                ;  Inputs:
      7                                ;   R2 = Pointer to window control block.
      8                                ;
      9 002516  004737  002306'        TCESCE: CALL    TCCR            ;Simulate carriage return
     10 002522  004737  002454'                CALL    TCLF            ;Simulate line feed
     11 002526  000207                          RETURN
```

```
                                        .SBTTL  TCIXUP -- Index up one line
                                ;-----------------------------------------------------------------------
                                ;  Move the cursor up one line.  Scroll the screen if we are on the top line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter value start in control sequence
                                ;
002530                          TCIXUP:
                                ;
                                ;  If we are not at the top line of a scrolling region,
                                ;  simply decriment the line number.
                                ;
002530  016200  0000000                 MOV     DW$LIN(R2),R0   ;Get current line number
002534  020062  0000000                 CMP     R0,DW$SRT(R2)   ;Are we at top of scrolling region?
002540  001406                          BEQ     1$              ;Br if yes
002542  005300                          DEC     R0              ;Decrement line number
002544  003001                          BGT     2$              ;Br if didn't go past line 1
002546  005200                          INC     R0              ;Constrain to line 1
002550  004737  006702'         2$:     CALL    SETLIN          ;Set this as current line
002554  000407                          BR      9$
                                ;
                                ;  We are at the top line of a scroll region.
                                ;  We must scroll down one line.
                                ;
002556  004737  005266'         1$:     CALL    SCRLDN          ;Scroll down one line
                                ;
                                ;  Finished
                                ;
002562  000207                  9$:     RETURN
```

```
                                        .SBTTL  TCRIT  -- Move cursor right specified number of chars
                                ;-----------------------------------------------------------------------
                                ;  Move cursor right a specified number of characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter value in control sequence
                                ;
002564                          TCRIT:
                                ;
                                ;  Get parameter value of number of chars to move
                                ;
002564  004737  002116'                 CALL    CSPARM          ;Get parameter value
                                ;
                                ;  Translate parameter value of 0 to 1
                                ;
002570  005700                          TST     R0              ;Parameter value zero?
002572  001001                          BNE     1$              ;Br if not
002574  005200                          INC     R0              ;Translate to 1
                                ;
                                ;  Move the cursor the specified number of columns but don't go past
                                ;  right margin.
                                ;
002576  066200  0000000         1$:     ADD     DW$COL(R2),R0   ;Get updated cursor column
002602  020062  0000000                 CMP     R0,DW$CPL(R2)   ;Did we go past right column?
002606  101402                          BLOS    2$              ;Br if not
002610  016200  0000000                 MOV     DW$CPL(R2),R0   ;Constrain to right column
002614  010062  0000000         2$:     MOV     R0,DW$COL(R2)   ;Set new column number
                                ;
                                ;  Finished
                                ;
002620  000207                          RETURN
```

```
                                .SBTTL  TCLEFT -- Move cursor left specified number of chars
                        ;-----------------------------------------------------------------------
                        ;  Move cursor left a specified number of characters.
                        ;
                        ;  Inputs:
                        ;   R2 = Pointer to window control block
                        ;   R5 = Pointer to parameter value in control sequence
                        ;
002622                  TCLEFT:
                        ;
                        ;  Get parameter value of number of chars to move
                        ;
002622  004737  002116'         CALL    CSPARM          ;Get parameter value
                        ;
                        ;  Translate parameter value of 0 to 1
                        ;
002626  005700                  TST     R0              ;Parameter value zero?
002630  001001                  BNE     1$              ;Br if not
002632  005200                  INC     R0              ;Translate to 1
                        ;
                        ;  Move the cursor the specified number of columns but don't go past
                        ;  left margin.
                        ;
002634  005400          1$:     NEG     R0              ;Get negative movement count
002636  066200  000000G         ADD     DW$COL(R2),R0   ;Get updated cursor column
002642  003002                  BGT     2$              ;Br if didn't go past column 1
002644  012700  000001          MOV     #1,R0           ;Constrain to column 1
002650  010062  000000G 2$:     MOV     R0,DW$COL(R2)   ;Set new column number
                        ;
                        ;  Finished
                        ;
002654  000207                  RETURN
```

```
                                        .SBTTL  TCCADR -- Do direct cursor addressing
                                ;-----------------------------------------------------------------------
                                ;  Move cursor to a specified line and column.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;   R5 = Pointer to parameter string within control sequence
                                ;
002656                          TCCADR:
                                ;
                                ;  Get line number parameter
                                ;
002656  004737  002116'                 CALL    CSPARM          ;Accrue line number parameter
002662  005700                          TST     R0              ;Translate 0 to 1
002664  001001                          BNE     1$
002666  005200                          INC     R0              ;Force to 1
002670  032762  000000G 000000G 1$:      BIT     #AW$ORS,DW$AW(R2);Is origin relative to scrolling region?
002676  001410                          BEQ     5$              ;Br if not
002700  066200  000000G                 ADD     DW$SRT(R2),R0   ;Bias line number by top of scrolling region
002704  005300                          DEC     R0              ;Top line number is 1
002706  020062  000000G                 CMP     R0,DW$SRB(R2)   ;Don't allow to go past end of scroll region
002712  101402                          BLOS    5$              ;Br if ok
002714  016200  000000G                 MOV     DW$SRB(R2),R0   ;Constrain to bottom of scroll region
002720  020062  000000G         5$:     CMP     R0,DW$LPP(R2)   ;Don't go past bottom of page
002724  101402                          BLOS    2$              ;Br if ok
002726  016200  000000G                 MOV     DW$LPP(R2),R0   ;Constrain to bottom of page
002732  004737  006702'         2$:     CALL    SETLIN          ;Set new line as current line
                                ;
                                ;  Get column number parameter
                                ;
002736  004737  002116'                 CALL    CSPARM          ;Accrue column number parameter
002742  005700                          TST     R0              ;Don't allow column 0
002744  001001                          BNE     3$              ;Br if non zero
002746  005200                          INC     R0              ;Translate 0 to 1
002750  020062  000000G         3$:     CMP     R0,DW$CPL(R2)   ;Don't allow to go past right margin
002754  101402                          BLOS    4$              ;Br if ok
002756  016200  000000G                 MOV     DW$CPL(R2),R0   ;Constrain to right margin
002762  010062  000000G         4$:     MOV     R0,DW$COL(R2)   ;Set new column number
                                ;
                                ;  Finished
                                ;
002766  000207                          RETURN
```

```
                                        .SBTTL  TCCPSV -- Save current cursor position
                                ;-----------------------------------------------------------------
                                ;  Save the current cursor position and character attributes.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
002770                          TCCPSV:
                                ;
                                ;  Save cursor position
                                ;
002770  116262  000000G 000000G         MOVB    DW$LIN(R2),DW$SLN(R2);Save current line number
002776  116262  000000G 000000G         MOVB    DW$COL(R2),DW$SCL(R2);Save current column number
                                ;
                                ;  Save character attributes
                                ;
003004  116262  000000G 000000G         MOVB    DW$CCA(R2),DW$SCA(R2);Save character attribute flags
                                ;
                                ;  Finished
                                ;
003012  000207                          RETURN

                                        .SBTTL  TCCPRS -- Restore cursor position
                                ;-----------------------------------------------------------------
                                ;  Restore cursor position and character attributes from saved data.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003014                          TCCPRS:
                                ;
                                ;  Restore the column number
                                ;
003014  116262  000000G 000000G         MOVB    DW$SCL(R2),DW$COL(R2);Restore column number
                                ;
                                ;  Restore the line number
                                ;
003022  116200  000000G                 MOVB    DW$SLN(R2),R0   ;Get saved line number
003026  004737  006702'                 CALL    SETLIN          ;Set as current line number
                                ;
                                ;  Restore character attributes
                                ;
003032  116262  000000G 000000G         MOVB    DW$SCA(R2),DW$CCA(R2);Restore character attribute flags
                                ;
                                ;  Finished
                                ;
003040  000207                          RETURN
```

```
                                        .SBTTL  TCINSL -- Insert lines
                                ;-----------------------------------------------------------------
                                ;  Insert a specified number of lines in front of the line in which
                                ;  the cursor is currently resting.  The cursor is left positioned
                                ;  at column 1 of the first inserted line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to command string argument
                                ;
003042  010146                  TCINSL: MOV     R1,-(SP)
003044  010346                          MOV     R3,-(SP)
003046  010446                          MOV     R4,-(SP)
003050  010546                          MOV     R5,-(SP)
                                ;
                                ;  Ignore this command if we are not within the scroll region
                                ;
003052  016203  0000000                 MOV     DW$LIN(R2),R3   ;Get current line number
003056  020362  0000000                 CMP     R3,DW$SRT(R2)   ;Are we above top to scroll region?
003062  103444                          BLO     9$              ;Br if yes
003064  020362  0000000                 CMP     R3,DW$SRB(R2)   ;Are we below bottom of scroll region?
003070  101041                          BHI     9$              ;Br if yes
                                ;
                                ;  Determine how many lines will be inserted
                                ;
003072  004737  002116'                 CALL    CSPARM          ;Get command parameter
003076  005700                          TST     R0              ;Was 0 specified?
003100  003002                          BGT     1$              ;Br if not
003102  012700  000001                  MOV     #1,R0           ;Set it to 1
003106  016201  0000000         1$:     MOV     DW$SRB(R2),R1   ;Get bottom line # of scroll region
003112  160301                          SUB     R3,R1           ;Get # lines from current to end of region
003114  005201                          INC     R1              ;This is max # inserted lines
003116  020100                          CMP     R1,R0           ;Can we scroll as many as requested?
003120  101413                          BLOS    2$              ;Br if all lines are scrolled off
                                ;
                                ;  Scroll down lines to make room for inserted lines
                                ;
003122  010103                          MOV     R1,R3           ;Get number of lines to scroll
003124  160003                          SUB     R0,R3
003126  010001                          MOV     R0,R1           ;Get # lines being inserted
003130  016205  0000000                 MOV     DW$SRB(R2),R5   ;Get # of line at bottom of scroll reg
003134  010504                  3$:     MOV     R5,R4
003136  160104                          SUB     R1,R4           ;Get # of line to copy to bottom
003140  004737  006346'                 CALL    CPYLIN          ;Copy a line down
003144  005305                          DEC     R5              ;Get next destination line number
003146  077306                          SOB     R3,3$           ;Loop if more lines to scroll down
                                ;
                                ;  Blank fill inserted lines
                                ;
003150  016203  0000000         2$:     MOV     DW$LIN(R2),R3   ;Get top line to blank
003154  010300                  4$:     MOV     R3,R0           ;Get # of line to clear
003156  004737  006232'                 CALL    CLRLIN          ;Clear that line
003162  005203                          INC     R3              ;Advance line number
003164  077105                          SOB     R1,4$           ;Loop if more lines to clear
                                ;
                                ;  Move cursor to column 1
                                ;
```

```
      58 003166  012762  000001  000000          MOV     #1,DW$COL(R2)   ;Say cursor is at column 1
      59                                ;
      60                                ;  Finished
      61                                ;
      62 003174  012605                 9$:     MOV     (SP)+,R5
      63 003176  012604                         MOV     (SP)+,R4
      64 003200  012603                         MOV     (SP)+,R3
      65 003202  012601                         MOV     (SP)+,R1
      66 003204  000207                         RETURN
```

```
                                        .SBTTL  TCDELL -- Delete specified number of lines
                                ;-----------------------------------------------------------------
                                ;  Delete the specified number of lines starting with the one in which
                                ;  the cursor is positioned.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to command string argument
                                ;
003206  010146                  TCDELL: MOV     R1,-(SP)
003210  010346                          MOV     R3,-(SP)
003212  010446                          MOV     R4,-(SP)
003214  010546                          MOV     R5,-(SP)
                                ;
                                ;  Ignore this command if we are not within the scroll region
                                ;
003216  016203  000000G                 MOV     DW$LIN(R2),R3   ;Get current line number
003222  020362  000000G                 CMP     R3,DW$SRT(R2)   ;Are we above top to scroll region?
003226  103442                          BLO     9$              ;Br if yes
003230  020362  000000G                 CMP     R3,DW$SRB(R2)   ;Are we below bottom of scroll region?
003234  101037                          BHI     9$              ;Br if yes
                                ;
                                ;  Determine how many lines will be deleted
                                ;
003236  004737  002116'                 CALL    CSPARM          ;Get command parameter
003242  005700                          TST     R0              ;Was 0 specified?
003244  003002                          BGT     1$              ;Br if not
003246  012700  000001                  MOV     #1,R0           ;Set it to 1
003252  016201  000000G         1$:     MOV     DW$SRB(R2),R1   ;Get bottom line # of scroll region
003256  160301                          SUB     R3,R1           ;Get # lines from current to end of region
003260  005201                          INC     R1              ;This is max # deleted lines
003262  020100                          CMP     R1,R0           ;Can we delete as many as requested?
003264  101412                          BLOS    2$              ;Br if all lines are deleted
                                ;
                                ;  Scroll up the remaining lines.
                                ;
003266  010305                          MOV     R3,R5           ;Get current line number
003270  010103                          MOV     R1,R3           ;Get number of lines to scroll
003272  160003                          SUB     R0,R3
003274  010001                          MOV     R0,R1           ;Get # lines being inserted
003276  010504                  3$:     MOV     R5,R4
003300  060104                          ADD     R1,R4           ;Get line to copy from
003302  004737  006346'                 CALL    CPYLIN          ;Copy a line up
003306  005205                          INC     R5              ;Advance line number
003310  077306                          SOB     R3,3$           ;Br if more lines to copy
                                ;
                                ;  Blank fill lines at end of scrolling region
                                ;
003312  016203  000000G         2$:     MOV     DW$SRB(R2),R3   ;Get # of line at bottom of region
003316  010300                  4$:     MOV     R3,R0           ;Get # of line to blank fill
003320  004737  006232'                 CALL    CLRLIN          ;Blank the line
003324  077104                          SOB     R1,4$           ;Loop if more lines to blank
                                ;
                                ;  Say cursor is at column 1
                                ;
003326  012762  000001  000000G         MOV     #1,DW$COL(R2)   ;Say cursor is at column 1
                                ;
```

```
                                        ;  Finished
                                        ;
003334  012605                  9$:       MOV     (SP)+,R5
003336  012604                            MOV     (SP)+,R4
003340  012603                            MOV     (SP)+,R3
003342  012601                            MOV     (SP)+,R1
003344  000207                            RETURN
```

```
                                        .SBTTL  TCGOAS -- Designate G0 as ascii character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G0 character set as ascii characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
003346  105062  000000          TCGOAS: CLRB    DW$GOM(R2)      ;Set character set number
003352  000207                          RETURN

                                        .SBTTL  TCGODS -- Designate G0 as DEC supplemental character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G0 character set as DEC supplemental character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003354  112762  000001  000000G TCGODS: MOVB    #1,DW$GOM(R2)   ;Set character set number
003362  000207                          RETURN

                                        .SBTTL  TCGOUK -- Designate G0 as UK national character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G0 character set as UK national character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003364  112762  000002  000000G TCGOUK: MOVB    #2,DW$GOM(R2)   ;Set character set number
003372  000207                          RETURN

                                        .SBTTL  TCGOGR -- Designate G0 as graphics character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G0 character set as graphics characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003374  112762  000003  000000G TCGOGR: MOVB    #3,DW$GOM(R2)   ;Set character set number
003402  000207                          RETURN
```

```
                                        .SBTTL  TCG1AS -- Designate G1 as ascii character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G1 character set as ascii characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
003404  105062  000000  TCG1AS: CLRB    DW$G1M(R2)      ;Set character set number
003410  000207                  RETURN

                                        .SBTTL  TCG1DS -- Designate G1 as DEC supplemental character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G1 character set as DEC supplemental character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003412  112762  000001  000000G TCG1DS: MOVB    #1,DW$G1M(R2)   ;Set character set number
003420  000207                  RETURN

                                        .SBTTL  TCG1UK -- Designate G1 as UK national character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G1 character set as UK national character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003422  112762  000002  000000G TCG1UK: MOVB    #2,DW$G1M(R2)   ;Set character set number
003430  000207                  RETURN

                                        .SBTTL  TCG1GR -- Designate G1 as graphics character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G1 character set as graphics characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003432  112762  000003  000000G TCG1GR: MOVB    #3,DW$G1M(R2)   ;Set character set number
003440  000207                  RETURN
```

```
                                        .SBTTL  TCG2AS -- Designate G2 as ascii character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G2 character set as ascii characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
003442  105062  000000  TCG2AS:  CLRB    DW$G2M(R2)      ;Set character set number
003446  000207                  RETURN

                                        .SBTTL  TCG2DS -- Designate G2 as DEC supplemental character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G2 character set as DEC supplemental character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003450  112762  000001  000000  TCG2DS:  MOVB    #1,DW$G2M(R2)   ;Set character set number
003456  000207                  RETURN

                                        .SBTTL  TCG2GR -- Designate G2 as graphics character set
                                ;-----------------------------------------------------------------------
                                ;  Designate the G2 character set as graphics characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003460  112762  000003  000000  TCG2GR:  MOVB    #3,DW$G2M(R2)   ;Set character set number
003466  000207                  RETURN
```

```
                                        .SBTTL  TCG3AS -- Designate G3 as ascii character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G3 character set as ascii characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
003470  105062  000000          TCG3AS: CLRB    DW$G3M(R2)      ;Set character set number
003474  000207                          RETURN

                                        .SBTTL  TCG3DS -- Designate G3 as DEC supplemental character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G3 character set as DEC supplemental character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003476  112762  000001  000000G TCG3DS: MOVB    #1,DW$G3M(R2)   ;Set character set number
003504  000207                          RETURN

                                        .SBTTL  TCG3GR -- Designate G3 as graphics character set
                                ;-----------------------------------------------------------------
                                ;  Designate the G3 character set as graphics characters.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003506  112762  000003  000000G TCG3GR: MOVB    #3,DW$G3M(R2)   ;Set character set number
003514  000207                          RETURN
```

```
                                        .SBTTL  TCSO    -- Shift-out character - Lock shift G1 to GL
                                ;-----------------------------------------------------------------------
                                ;  The shift-out character is used to lock character set G1 to GL.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
003516  112762  000001  000000G TCSO:   MOVB    #1,DW$GLM(R2)   ;Map GL to G1
003524  000207                          RETURN

                                        .SBTTL  TCSI    -- Shift-in character - Lock shift G0 to GL
                                ;-----------------------------------------------------------------------
                                ;  The shift-in character is used to lock character set G0 to GL.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003526  105062  000000G         TCSI:   CLRB    DW$GLM(R2)      ;Map GL to G0
003532  000207                          RETURN

                                        .SBTTL  TCLS2  -- Lock shift G2 to GL
                                ;-----------------------------------------------------------------------
                                ;  Set G2 as current GL character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003534  112762  000002  000000G TCLS2:  MOVB    #2,DW$GLM(R2)   ;Map GL to G2
003542  000207                          RETURN

                                        .SBTTL  TCLS3  -- Lock shift G3 to GL
                                ;-----------------------------------------------------------------------
                                ;  Set G3 as current GL character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003544  112762  000003  000000G TCLS3:  MOVB    #3,DW$GLM(R2)   ;Map GL to G3
003552  000207                          RETURN

                                        .SBTTL  TCLS1R -- Lock shift G1 into GR
                                ;-----------------------------------------------------------------------
                                ;  Set G1 as current GR character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
003554  112762  000001  000000G TCLS1R: MOVB    #1,DW$GRM(R2)   ;Map GR to G1
003562  000207                          RETURN

                                        .SBTTL  TCLS2R -- Lock shift G2 into GR
                                ;-----------------------------------------------------------------------
                                ;  Set G2 as the GR character set.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
```

```
      58 003564  112762  000002  000000G TCLS2R: MOVB    #2,DW$GRM(R2)   ;Map GR to G2
      59 003572  000207                       RETURN
      60
      61                                      .SBTTL  TCLS3R -- Lock shift G3 into GR
      62                                ;-----------------------------------------------------------------------
      63                                ;   Set G3 as the GR character set.
      64                                ;
      65                                ;  Inputs:
      66                                ;   R2 = Pointer to window control block.
      67                                ;
      68 003574  112762  000003  000000G TCLS3R: MOVB    #3,DW$GRM(R2)   ;Map GR to G3
      69 003602  000207                       RETURN
      70
      71                                      .SBTTL  TCSS2  -- Single shift G2 to GL
      72                                ;-----------------------------------------------------------------------
      73                                ;  Set G2 as GL for next character only.
      74                                ;
      75                                ;  Inputs:
      76                                ;   R2 = Pointer to window control block.
      77                                ;
      78 003604  112762  000002  000000G TCSS2:  MOVB    #2,DW$GLS(R2)   ;Map GL to G2 for next character only
      79 003612  052762  000000G 000000G        BIS     #AW$SS,DW$AW(R2);Remember single shift is pending
      80 003620  000207                       RETURN
      81
      82                                      .SBTTL  TCSS3  -- Single shift G3 to GL
      83                                ;-----------------------------------------------------------------------
      84                                ;  Set G3 as GL for next character only.
      85                                ;
      86                                ;  Inputs:
      87                                ;   R2 = Pointer to window control block.
      88                                ;
      89 003622  112762  000003  000000G TCSS3:  MOVB    #3,DW$GLS(R2)   ;Map GL to G3 for next character only
      90 003630  052762  000000G 000000G        BIS     #AW$SS,DW$AW(R2);Remember single shift is pending
      91 003636  000207                       RETURN
```

```
                                        .SBTTL  TCSAA  -- Set ANSI attribute
                                ;-------------------------------------------------------------------
                                ;  Set an ANSI terminal attribute.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter values
                                ;
003640  010346                  TCSAA:  MOV     R3,-(SP)
003642  012703  004112'                 MOV     #ASATAT,R3      ;Point to ANSI conversion table
003646  004737  003730'                 CALL    TCSTA           ;Go set the attribute
003652  012603                          MOV     (SP)+,R3
003654  000207                          RETURN

                                        .SBTTL  TCRAA  -- Reset ANSI attribute
                                ;-------------------------------------------------------------------
                                ;  Reset an ANSI terminal attribute.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter values
                                ;
003656  010346                  TCRAA:  MOV     R3,-(SP)
003660  012703  004112'                 MOV     #ASATAT,R3      ;Point to ANSI conversion table
003664  004737  003766'                 CALL    TCRTA           ;Reset the attribute
003670  012603                          MOV     (SP)+,R3
003672  000207                          RETURN

                                        .SBTTL  TCSDA  -- Set DEC attribute
                                ;-------------------------------------------------------------------
                                ;  Set a DEC terminal attribute
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter values
                                ;
003674  010346                  TCSDA:  MOV     R3,-(SP)
003676  012703  004120'                 MOV     #DECTAT,R3      ;Point to DEC conversion table
003702  004737  003730'                 CALL    TCSTA           ;Set the attribute
003706  012603                          MOV     (SP)+,R3
003710  000207                          RETURN

                                        .SBTTL  TCRDA  -- Reset DEC attribute
                                ;-------------------------------------------------------------------
                                ;  Reset a DEC terminal attribute
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter values
                                ;
003712  010346                  TCRDA:  MOV     R3,-(SP)
003714  012703  004120'                 MOV     #DECTAT,R3      ;Point to DEC conversion table
003720  004737  003766'                 CALL    TCRTA           ;Reset the attribute
003724  012603                          MOV     (SP)+,R3
003726  000207                          RETURN
```

```
                                        .SBTTL  TCSTA  -- Set terminal attribute
                                ;-----------------------------------------------------------------
                                ;  Set a terminal attribute
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R3 = Pointer to table to convert attribute value to flag bit
                                ;   R5 = Pointer to parameter values
                                ;
003730  010446                  TCSTA:  MOV     R4,-(SP)
                                ;
                                ;  Begin loop to accrue attribute parameters until end of string hit
                                ;
003732  004737  002116'         1$:     CALL    CSPARM          ;Accrue next parameter value
003736  103411                          BCS     9$              ;Br if no more parameter values
                                ;
                                ;  Convert parameter value into flag mask
                                ;
003740  010004                          MOV     R0,R4           ;Save parameter value
003742  004737  004060'                 CALL    CVTTAV          ;Convert parameter value into flag mask
003746  103771                          BCS     1$              ;Br if parameter not recognized
                                ;
                                ;  Set appropriate flag bit
                                ;
003750  050062  0000000                 BIS     R0,DW$AW(R2)    ;Set appropriate attribute flag
                                ;
                                ;  See if this change requires doing more than setting a flag bit
                                ;
003754  004737  004152'                 CALL    CKTAC           ;Check for additional changes
                                ;
                                ;  See if there are more attributes to set
                                ;
003760  000764                          BR      1$
                                ;
                                ;  Finished
                                ;
003762  012604                  9$:     MOV     (SP)+,R4
003764  000207                          RETURN
```

```
                                        .SBTTL  TCRTA  -- Reset terminal attribute
                                ;-----------------------------------------------------------------------
                                ;  Reset a terminal attribute
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R3 = Pointer to table to convert attribute value to flag bit
                                ;   R5 = Pointer to parameter values
                                ;
003766  010446                  TCRTA:  MOV     R4,-(SP)
                                ;
                                ;  Begin loop to accrue attribute parameters until end of string hit
                                ;
003770  004737  002116'         1$:     CALL    CSPARM          ;Accrue next parameter value
003774  103427                          BCS     9$              ;Br if no more parameters
                                ;
                                ;  Check for special parameter value 2 which means select VT52 mode
                                ;
003776  020027  000002                  CMP     R0,#2           ;Select VT52 mode?
004002  001013                          BNE     2$              ;Br if not
004004  052762  000000@ 000000@         BIS     #AW$S52,DW$AW(R2);Enable VT52 simulation mode for window
004012  116204  000000@                 MOVB    DW$JOB(R2),R4   ;Get out job index number
004016  116404  000000@                 MOVB    LNPRIM(R4),R4   ;Get our primary job #
004022  052764  000000@ 000000@         BIS     #$V52EM,LSW11(R4);Say we are emulating a VT52
004030  000757                          BR      1$
                                ;
                                ;  Convert parameter value into flag mask
                                ;
004032  010004                  2$:     MOV     R0,R4           ;Save parameter value
004034  004737  004060'                 CALL    CVTTAV          ;Convert parameter value into flag mask
004040  103753                          BCS     1$              ;Br if parameter value not recognized
                                ;
                                ;  Reset appropriate flag bit
                                ;
004042  040062  000000@                 BIC     R0,DW$AW(R2)    ;Reset appropriate attribute flag
                                ;
                                ;  See if this change requires doing more than clearing a bit
                                ;
004046  004737  004152'                 CALL    CKTAC           ;Check for special mode changes
                                ;
                                ;  See if there are more attributes to reset
                                ;
004052  000746                          BR      1$
                                ;
                                ;  Finished
                                ;
004054  012603                  9$:     MOV     (SP)+,R3
004056  000207                          RETURN
```

```
                                        .SBTTL  CVTTAV -- Convert terminal attribute value to flag
                                ;-----------------------------------------------------------------------
                                ;  Convert a terminal attribute parameter value into a flag mask.
                                ;
                                ;  Inputs:
                                ;   R0 = Attribute value
                                ;   R3 = Pointer to table to convert attribute value to flag bit
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Attribute value not recognized
                                ;   R0 = Flag mask if attribute value recognized
                                ;
004060  010346                  CVTTAV: MOV     R3,-(SP)
                                ;
                                ;  See if we can find attribute value in table
                                ;
004062  020023                  1$:     CMP     R0,(R3)+        ;Is next entry for our value?
004064  001406                          BEQ     2$              ;Br if yes
004066  005723                          TST     (R3)+           ;Skip over flag mask
004070  005713                          TST     (R3)            ;Is there another entry in table?
004072  001373                          BNE     1$              ;Br if yes
                                ;
                                ;  Attribute value is not in table
                                ;
004074  005000                          CLR     R0              ;Return no flag mask
004076  000261                          SEC                     ;Signal failure on return
004100  000402                          BR      9$
                                ;
                                ;  We found the attribute in the table
                                ;
004102  011300                  2$:     MOV     (R3),R0         ;Get the flag mask
004104  000241                          CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
004106  012603                  9$:     MOV     (SP)+,R3
004110  000207                          RETURN
                                ;-----------------------------------------------------------------------
                                ;
                                ;  Table to relate ANSI attribute values with flag masks
                                ;
004112  000004  0000000         ASATAT: .WORD   4.,AW$INS       ;Insert mode
004116  000000                          .WORD   0
                                ;
                                ;  Table to relate DEC attribue values with flag masks
                                ;
004120  000001  0000000         DECTAT: .WORD   1.,AW$ACK       ;Application mode for cursor keys
004124  000003  0000000                 .WORD   3.,AW$132       ;132 column mode
004130  000005  0000000                 .WORD   5.,AW$REV       ;Reverse video
004134  000006  0000000                 .WORD   6.,AW$ORS       ;Origin relative to scroll region
004140  000010  0000000                 .WORD   8.,AW$RPT       ;Automatic keypad repeat
004144  000031  0000000                 .WORD   25.,AW$VCR      ;Make cursor visible
004150  000000                          .WORD   0               ;End of table
```

```
                                        .SBTTL  CKTAC  -- Check for complex mode changes
                                ;-----------------------------------------------------------------
                                ;  This routine is called each time a terminal attribute changes to see
                                ;  if we must do things other than setting or clearing an attribute flag.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R4 = Parameter value specified with attribute change command.
                                ;
004152                          CKTAC:
                                ;
                                ;  If we just changed the origin mode, home the cursor
                                ;
004152  020427  000006                  CMP     R4,#6           ;Change origin mode?
004156  001003                  BNE     1$              ;Br if not
004160  004737  005110'                 CALL    SETHOM          ;Say cursor has moved to home
004164  000415                  BR      9$
                                ;
                                ;  If we just changed line width (80/132 mode), clear the screen,
                                ;  set scrolling region to full screen, and home the cursor.
                                ;
004166  020427  000003          1$:     CMP     R4,#3           ;Change line width?
004172  001012                  BNE     9$              ;Br if not
004174  012762  000001  0000000         MOV     #1,DW$SRT(R2)   ;Set top scroll line to 1
004202  016262  0000000 0000000         MOV     DW$LPP(R2),DW$SRB(R2) ;Set bottom scroll line to end of page
004210  004737  005110'                 CALL    SETHOM          ;Set cursor to home position
004214  004737  006210'                 CALL    ERSPAG          ;Erase the entire page
                                ;
                                ;  Finished
                                ;
004220  000207                  9$:     RETURN
```

```
                                                .SBTTL  TCAKM  -- Select application keypad mode
                                ;-----------------------------------------------------------------------
                                ;  Select application keypad mode
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
004222  052762  0000006 0000006 TCAKM:  BIS     #AW$AKM,DW$AW(R2);Say we are in application keypad mode
004230  000207                          RETURN

                                                .SBTTL  TCNKM  -- Select numeric keypad mode
                                ;-----------------------------------------------------------------------
                                ;  Select numeric keypad mode
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
004232  042762  0000006 0000006 TCNKM:  BIC     #AW$AKM,DW$AW(R2);Reset application keypad mode
004240  000207                          RETURN
```

```
                                        .SBTTL  TCN3    -- Set line attributes
                                ;-----------------------------------------------------------------------
                                ;  Set attributes for the current line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
                                ;  Line is top half of double-high line
                                ;
004242  004737  006612'         TCN3:   CALL    GETLAB          ;Get current line attribute flags
004246  042700  0000000         BIC     #AL$DHB,R0      ;Clear double-high-bottom flag
004252  052700  0000000         BIS     #AL$DHT,R0      ;Set double-high-top flag
004256  004737  006522'         CALL    SETLAB          ;Store new attribute flags for line
004262  000207                  RETURN
                                ;
                                ;  Line is bottom half of double-high line
                                ;
004264  004737  006612'         TCN4:   CALL    GETLAB          ;Get current line attribute flags
004270  042700  0000000         BIC     #AL$DHT,R0      ;Clear double-high-top flag
004274  052700  0000000         BIS     #AL$DHB,R0      ;Set double-high-bottom flag
004300  004737  006522'         CALL    SETLAB          ;Store new attribute flags for line
004304  000207                  RETURN
                                ;
                                ;  Line is single width
                                ;
004306  004737  006612'         TCN5:   CALL    GETLAB          ;Get current line attribute flags
004312  042700  0000000         BIC     #AL$DWD,R0      ;Clear double-wide attribute
004316  004737  006522'         CALL    SETLAB          ;Store new attribute flags for line
004322  000207                  RETURN
                                ;
                                ;  Line is double width
                                ;
004324  004737  006612'         TCN6:   CALL    GETLAB          ;Get current line attribute flags
004330  052700  0000000         BIS     #AL$DWD,R0      ;Set double-wide flag
004334  004737  006522'         CALL    SETLAB          ;Store new attribute flags for line
004340  000207                  RETURN
```

```
                                        .SBTTL  TCSCA  -- Set character attributes
                                ;-----------------------------------------------------------------------
                                ;  Specify attributes for characters that follow
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to start of parameter values
                                ;
004342  010346                  TCSCA:  MOV     R3,-(SP)
                                ;
                                ;  If no parameters were specified, reset all character attributes
                                ;
004344  121527  000155                  CMPB    (R5),#'m        ;Any attributes specified?
004350  001004                  1$:     BNE     1$              ;Br if yes
004352  142762  000000C 000000G         BICB    #<AC$BLD!AC$BLK!AC$REV!AC$ULN>,DW$CCA(R2);Clear all attributes
004360  000422                          BR      9$              ;Finished
                                ;
                                ;  Begin loop to accrue and process each parameter
                                ;
004362  004737  002116'         1$:     CALL    CSPARM          ;Accrue next parameter value
004366  103417                          BCS     9$              ;Br if no more parameter values
                                ;
                                ;  Look up parameter value to determine which attribute flags
                                ;  are to be set and reset.
                                ;
004370  012703  004432'                  MOV     #CATABL,R3      ;Point to attribute table
004374  120023                  2$:     CMPB    R0,(R3)+        ;Search for parameter value in table
004376  001406                          BEQ     3$              ;Br if found it
004400  062703  000002                  ADD     #2,R3           ;Skip over flag bytes
004404  020327  004465'                 CMP     R3,#CATEND      ;Reached end of table?
004410  103771                          BLO     2$              ;Keep looking if not
004412  000763                          BR      1$              ;Cannot find this parameter value
                                ;
                                ;  Found parameter value in table.
                                ;  Set and reset some flags
                                ;
004414  152362  0000000G        3$:     BISB    (R3)+,DW$CCA(R2);Set some attributes
004420  142362  0000000G                BICB    (R3)+,DW$CCA(R2);Reset some attributes
004424  000756                          BR      1$              ;Process any other attributes
                                ;
                                ;  Finished
                                ;
004426  012603                  9$:     MOV     (SP)+,R3
004430  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Table to convert attribute parameter values into flags to be
                                ;  set and reset.
                                ;  Each entry has three values: (1) numeric parameter value that
                                ;  occurs in control sequence; (2) flags to be set; (3) flags to
                                ;  be reset.
                                ;
004432      000      000      000C CATABL:  .BYTE    0.,0,AC$BLD!AC$BLK!AC$REV!AC$ULN
004435      001      000@     000          .BYTE    1.,AC$BLD,0
004440      004      000@     000          .BYTE    4.,AC$ULN,0
004443      005      000@     000          .BYTE    5.,AC$BLK,0
004446      007      000@     000          .BYTE    7.,AC$REV,0
004451      026      000      000@         .BYTE    22.,0,AC$BLD
004454      030      000      000@         .BYTE    24.,0,AC$ULN
004457      031      000      000@         .BYTE    25.,0,AC$BLK
004462      033      000      000@         .BYTE    27.,0,AC$REV
004465                                CATEND:
                                             .EVEN
```

```
                                        .SBTTL  TC100  -- Select VT100/VT200 mode
                                ;-----------------------------------------------------------------------
                                ;  Set terminal to VT100 or VT200 mode of operation
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter values
                                ;
004466                          TC100:
                                ;
                                ;  Say terminal is not in VT52 mode and not in VT200 mode
                                ;  (i.e., set to VT100 mode)
                                ;
004466  042762  000000C 000000G        BIC     #<AW$52!AW$552!AW$200>,DW$AW(R2) ;Terminal is in VT100 mode
                                ;
                                ;  Reset VT52 emulation-mode flag in entry for primary line so that
                                ;  other windows connected to this terminal will know.
                                ;
004474  116200  0000000                MOVB    DW$JOB(R2),R0   ;Get our line index number
004500  116000  0000000                MOVB    LNPRIM(R0),R0   ;Get number of primary line
004504  042760  0000000 000000G       BIC     #$V52EM,LSW11(R0) ;Say terminal not emulating a VT52
                                ;
                                ;  If job's terminal type is VT200, set that mode
                                ;
004512  116200  0000000                MOVB    DW$JOB(R2),R0   ;Get job index number
004516  032760  000000C 000000G       BIT     #<VT2007!VT2008>,LTRMTP(R0);Is terminal a VT200?
004524  001005                        BNE     1$              ;Br if yes
                                ;
                                ;  Get 1st parameter value and see if we should go to VT200 mode
                                ;
004526  004737  002116'                CALL    CSPARM          ;Accrue 1st parameter value
004532  020027  000076                 CMP     R0,#62.         ;VT200 mode?
004536  001003                        BNE     9$              ;Br if not
                                ;
                                ;  Set terminal type to VT200
                                ;
004540  052762  0000000 0000000 1$:    BIS     #AW$200,DW$AW(R2);Set VT200 mode
                                ;
                                ;  Finished
                                ;
004546  000207                 9$:    RETURN
```

```
                                        .SBTTL  TCREST -- Terminal reset
                                ;-----------------------------------------------------------------------
                                ;  Perform a terminal reset operation
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
004550  010146                  TCREST: MOV     R1,-(SP)
                                ;
                                ;  Turn off VT52 emulation mode
                                ;
004552  042762  000000G 000000G         BIC     #AW$S52,DW$AW(R2);Turn off attribute for window
004560  116201  000000G                 MOVB    DW$JOB(R2),R1   ;Get our job index number
004564  116100  000000G                 MOVB    LNPRIM(R1),R0   ;Get primary line number
004570  042760  000000G 000000G         BIC     #$V52EM,LSW11(R0) ;Say terminal is not in VT52 mode
                                ;
                                ;  Reset scrolling region
                                ;
004576  012762  000001  000000G         MOV     #1,DW$SRT(R2)   ;Set top scroll line to 1
004604  016262  000000G 000000G         MOV     DW$LPP(R2),DW$SRB(R2) ;Set bottom scroll line to end of page
                                ;
                                ;  Reset misc window status flags
                                ;
004612  042762  000000C 000000G         BIC     #<AW$132!AW$INS!AW$ACK!AW$ORS!AW$AKM!AW$SS!AW$PRT>,DW$AW(R2)
                                ;
                                ;  Say cursor is at home
                                ;
004620  004737  005110'                 CALL    SETHOM          ;Set cursor to home position
                                ;
                                ;  Erase the entire page
                                ;
004624  004737  006210'                 CALL    ERSPAG          ;Erase the entire page
                                ;
                                ;  Reset character set mapping for a VT200
                                ;
004630  032761  000000C 000000G         BIT     #<VT2007!VT2008>,LTRMTP(R1) ;Is this a VT200?
004636  001411                          BEQ     9$              ;Br if not
004640  112762  000001  000000G         MOVB    #1,DW$G2M(R2)   ;Map G2 to DEC supplemental chars
004646  112762  000001  000000G         MOVB    #1,DW$G3M(R2)   ;Map G3 to DEC supplemental chars
004654  112762  000002  000000G         MOVB    #2,DW$GRM(R2)   ;Map GR to G2
                                ;
                                ;  Finished
                                ;
004662  012601                  9$:     MOV     (SP)+,R1
004664  000207                          RETURN
```

```
                                        .SBTTL  TCPRT  -- Printer control operation
                                ;-----------------------------------------------------------------------
                                ;  Direct output to printer or back to terminal.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to start of parameter value
                                ;
004666                          TCPRT:
                                ;
                                ;  Get first parameter value and see if we should direct output to printer
                                ;  or to terminal.
                                ;
004666  004737  002116'                 CALL    CSPARM          ;Get first parameter value
004672  020027  000005                  CMP     R0,#5           ;Direct output to printer?
004676  001004                          BNE     1$              ;Br if not
004700  052762  0000000 0000000         BIS     #AW$PRT,DW$AW(R2);Direct output to printer
004706  000406                          BR      9$
004710  020027  000004          1$:     CMP     R0,#4           ;Direct output to terminal?
004714  001003                          BNE     9$              ;Br if not
004716  042762  0000000 0000000         BIC     #AW$PRT,DW$AW(R2);Direct output to terminal
                                ;
                                ;  Finished
                                ;
004724  000207                  9$:     RETURN
```

```
                                        .SBTTL  TCLERS -- Erase within a line
                                ;-----------------------------------------------------------------------
                                ;  Erase within a line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to start of parameter value
                                ;
004726                          TCLERS:
                                ;
                                ;  Accrue parameter value to determine what type of erase this is
                                ;
004726  004737  002116'                 CALL    CSPARM          ;Accrue parameter value
004732  005700                          TST     R0              ;Parameter = 0?
004734  001003                          BNE     1$              ;Br if not
004736  004737  005632'                 CALL    ERSCTL          ;Erase from cursor to end of line
004742  000413                          BR      9$
004744  020027  000001         1$:      CMP     R0,#1           ;Parameter = 1?
004750  001003                          BNE     2$              ;Br if not
004752  004737  005744'                 CALL    ERSLTC          ;Erase from beginning of line to cursor
004756  000405                          BR      9$
004760  020027  000002         2$:      CMP     R0,#2           ;Parameter = 2?
004764  001002                          BNE     9$              ;Br if not
004766  004737  005546'                 CALL    ERSLIN          ;Erase entire line
                                ;
                                ;  Finished
                                ;
004772  000207                  9$:     RETURN
```

```
                                        .SBTTL  TCPERS -- Erase within a page
                                ;-----------------------------------------------------------------------
                                ;  Erase characters within the current page.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to start of parameter value
                                ;
004774                          TCPERS:
                                ;
                                ;  Accrue parameter value and determine how much to erase
                                ;
004774  004737  002116'                 CALL    CSPARM          ;Accrue parameter
005000  005700                          TST     R0              ;Parameter = 0?
005002  001003                          BNE     1$              ;Br if not
005004  004737  006030'                 CALL    ERSCTP          ;Erase from cursor to end of page
005010  000413                          BR      9$
005012  020027  000001          1$:     CMP     R0,#1           ;Parameter = 1?
005016  001003                          BNE     2$              ;Br if not
005020  004737  006112'                 CALL    ERSPTC          ;Erase from beginning of page to cursor
005024  000405                          BR      9$
005026  020027  000002          2$:     CMP     R0,#2           ;Parameter = 2?
005032  001002                          BNE     9$              ;Br if not
005034  004737  006210'                 CALL    ERSPAG          ;Erase entire page
                                ;
                                ;  Finished
                                ;
005040  000207                  9$:     RETURN
```

```
                                        .SBTTL  TCSSR  -- Set scrolling region
                                ;-----------------------------------------------------------------------
                                ;  Set the scrolling region.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R5 = Pointer to parameter values
                                ;
005042                          TCSSR:
                                ;
                                ;  Accrue scrolling region top value
                                ;
005042  004737  002116'                 CALL    CSPARM          ;Accrue scrolling region top line
005046  005700                          TST     R0              ;Was top line number specified?
005050  001002                          BNE     1$              ;Br if yes
005052  012700  000001                  MOV     #1,R0           ;Set 1 as top line
005056  010062  000000G         1$:     MOV     R0,DW$SRT(R2)   ;Set top line of scrolling region
                                ;
                                ;  Accrue scrolling region bottom value
                                ;
005062  004737  002116'                 CALL    CSPARM          ;Accrue scrolling region bottom line
005066  005700                          TST     R0              ;Was bottom line specified?
005070  001002                          BNE     2$              ;Br if yes
005072  016200  000000G                 MOV     DW$LPP(R2),R0   ;Set last line as bottom of region
005076  010062  000000G         2$:     MOV     R0,DW$SRB(R2)   ;Set bottom line of scrolling region
                                ;
                                ;  Set the cursor to the home position
                                ;
005102  004737  005110'                 CALL    SETHOM          ;Set cursor to home position
                                ;
                                ;  Finished
                                ;
005106  000207                          RETURN
```

```
                                        .SBTTL  SETHOM -- Set cursor to the home position
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to set the current cursor position to
                                ;  home.  The home line depends on whether absolute or relative
                                ;  origin has been selected.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
005110                          SETHOM:
                                ;
                                ;  Set the cursor line number.
                                ;  See if absolute or relative origin has been selected.
                                ;
005110  032762  0000000 0000000         BIT     #AW$ORS,DW$AW(R2);Is origin relative to scroll region?
005116  001403                          BEQ     1$                      ;Br if not
005120  016200  0000000                 MOV     DW$SRT(R2),R0   ;Say cursor is at top of scroll reg
005124  000402                          BR      2$
005126  012700  000001          1$:     MOV     #1,R0                   ;Say cursor is at line 1
005132  004737  006702'         2$:     CALL    SETLIN                  ;Set the line number
                                ;
                                ;  Set column to 1
                                ;
005136  012762  000001  0000000         MOV     #1,DW$COL(R2)   ;Say cursor is at column 1
                                ;
                                ;  Finished
                                ;
005144  000207                          RETURN
```

```
                                        .SBTTL  SCRLUP -- Scroll screen up one line
                                ;-----------------------------------------------------------------------
                                ;  Scroll the screen up one line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
005146  010346                  SCRLUP: MOV     R3,-(SP)
005150  010446                          MOV     R4,-(SP)
005152  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if we are allowed to scroll any more lines
                                ;
005154  012700  000001                  MOV     #1,R0           ;We are scrolling up one line
005160  004737  005414'                 CALL    SCRCHK          ;See if scrolling allowed
                                ;
                                ;  If the scrolling region is the entire screen, we can do the scrolling
                                ;  by simply changing the number of the top line.
                                ;
005164  026227  000000@ 000001          CMP     DW$SRT(R2),#1   ;Top line of scrolling region = 1?
005172  001013                          BNE     2$              ;Br if not
005174  026262  000000@ 000000@         CMP     DW$SRB(R2),DW$LPP(R2);Scrolling region go to bottom of page?
005202  001007                          BNE     2$              ;Br if not
                                ;
                                ;  Scrolling region covers the entire page
                                ;
005204  005362  000000@                 DEC     DW$TLN(R2)      ;Change top line number
005210  003016                          BGT     1$              ;Br if did not just scroll line 1
005212  016262  000000@ 000000@         MOV     DW$LPP(R2),DW$TLN(R2);Cycle to last line on page
005220  000412                          BR      1$
                                ;
                                ;  The scrolling region does not cover the entire page.
                                ;  Move characters from line to line to do the scrolling.
                                ;
005222  016203  000000@         2$:     MOV     DW$SRT(R2),R3   ;Get top line # of scrolling region
005226  010305                  3$:     MOV     R3,R5           ;Set number of line to copy to
005230  005203                          INC     R3
005232  010304                          MOV     R3,R4           ;Set number of line to copy from
005234  004737  006346'                 CALL    CPYLIN          ;Copy characters from one line to another
005240  020362  000000@                 CMP     R3,DW$SRB(R2)   ;Have we copied all that is needed?
005244  103770                          BLO     3$              ;Loop if not
                                ;
                                ;  Clear the line at the bottom of the scrolling region
                                ;
005246  016200  000000@         1$:     MOV     DW$SRB(R2),R0   ;Get # of line at bottom of scrolling regn
005252  004737  006232'                 CALL    CLRLIN          ;Clear that line
                                ;
                                ;  Finished
                                ;
005256  012605                          MOV     (SP)+,R5
005260  012604                          MOV     (SP)+,R4
005262  012603                          MOV     (SP)+,R3
005264  000207                          RETURN
```

```
                                        .SBTTL  SCRLDN -- Scroll screen down one line
                                ;-----------------------------------------------------------------------
                                ;  Scroll the screen down one line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
005266  010346                  SCRLDN: MOV     R3,-(SP)
005270  010446                          MOV     R4,-(SP)
005272  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if we are allowed to scroll any more lines
                                ;
005274  012700  177777                  MOV     #-1,R0          ;Say we are scrolling down one line
005300  004737  005414'                 CALL    SCRCHK          ;See if we are allowed to scroll more
                                ;
                                ;  If the scrolling region is the entire screen, we can do the scrolling
                                ;  by simply changing the number of the top line.
                                ;
005304  026227  0000000 000001          CMP     DW$SRT(R2),#1   ;Top line of scrolling region = 1?
005312  001016                          BNE     2$              ;Br if not
005314  026262  0000000 0000000         CMP     DW$SRB(R2),DW$LPP(R2);Scrolling region go to bottom of page?
005322  001012                          BNE     2$              ;Br if not
                                ;
                                ;  Scrolling region covers the entire page
                                ;
005324  005262  0000000                 INC     DW$TLN(R2)      ;Change top line number
005330  026262  0000000 0000000         CMP     DW$TLN(R2),DW$LPP(R2);Did we just scroll last line?
005336  101416                          BLOS    1$              ;Br if not
005340  012762  000001  0000000         MOV     #1,DW$TLN(R2)   ;Cycle back to line 1
005346  000412                          BR      1$
                                ;
                                ;  The scrolling region does not cover the entire page.
                                ;  Move characters from line to line to do the scrolling.
                                ;
005350  016203  0000000         2$:     MOV     DW$SRB(R2),R3   ;Get bottom line # of scrolling region
005354  010305                  3$:     MOV     R3,R5           ;Set number of line to copy to
005356  005303                          DEC     R3
005360  010304                          MOV     R3,R4           ;Set number of line to copy from
005362  004737  006346'                 CALL    CPYLIN          ;Copy characters from one line to another
005366  020362  0000000                 CMP     R3,DW$SRT(R2)   ;Have we copied all that is needed?
005372  101370                          BHI     3$              ;Loop if not
                                ;
                                ;  Clear the line at the top of the scrolling region
                                ;
005374  016200  0000000         1$:     MOV     DW$SRT(R2),R0   ;Get # of line at top of scrolling regn
005400  004737  006232'                 CALL    CLRLIN          ;Clear that line
                                ;
                                ;  Finished
                                ;
005404  012605                          MOV     (SP)+,R5
005406  012604                          MOV     (SP)+,R4
005410  012603                          MOV     (SP)+,R3
005412  000207                          RETURN
```

```
                                        .SBTTL  SCRCHK -- Check for scrolling limits
                                ;-----------------------------------------------------------------------
                                ;  Determine if we are allowed to scroll the screen any further while the
                                ;  job is detached from the terminal.  If we have reached the limit of
                                ;  our scrolling, the job is suspended until the terminal is reconnected.
                                ;
                                ;  Inputs:
                                ;   R0 = +1 if scrolling up, -1 if scrolling down.
                                ;   R2 = Pointer to window control block.
                                ;
005414  010146                  SCRCHK: MOV     R1,-(SP)
                                ;
                                ;  See if job is currently attached to terminal
                                ;
005416  032762  0000000 0000000         BIT     #AW$DDC,DW$AW(R2);Are we suppressing char display?
005424  001426                          BEQ     9$              ;Br if not
005426  116201  0000000                 MOVB    DW$JOB(R2),R1   ;Get number of job
005432  032761  0000000 0000000         BIT     #$VNOTT,LSW(R1) ;Is job currently connected to terminal?
005440  001420                          BEQ     9$              ;Br if yes
                                ;
                                ;  Job is not currently attached to the terminal.
                                ;  See if we are allowed to scroll an unlimited number of lines.
                                ;
005442  116201  0000000                 MOVB    DW$MSL(R2),R1   ;Are we allowed to scroll unlimited # lines?
005446  002415                          BLT     9$              ;Br if yes
005450  001412                          BEQ     2$              ;Br if no scrolling is allowed
                                ;
                                ;  See if this scroll would exceed limit.
                                ;
005452  116201  0000000                 MOVB    DW$NSL(R2),R1   ;Get # lines scrolled so far
005456  060001                          ADD     R0,R1           ;Add number scrolled this time
005460  110162  0000000                 MOVB    R1,DW$NSL(R2)   ;Save new total
005464  002001                          BGE     1$              ;Br if total is positive
005466  005401                          NEG     R1              ;Get positive line count
005470  120162  0000000         1$:     CMPB    R1,DW$MSL(R2)   ;Would this exceed max allowed scroll?
005474  101402                          BLOS    9$              ;Br if not
                                ;
                                ;  Suspend job until job reattaches to the terminal
                                ;
005476  004737  005506'         2$:     CALL    WINSPN          ;Suspend job
                                ;
                                ;  Finished
                                ;
005502  012601                  9$:     MOV     (SP)+,R1
005504  000207                          RETURN
```

```
                                        .SBTTL  WINSPN -- Suspend job until it reattaches to terminal
                                ;-----------------------------------------------------------------------
                                ;  Suspend the execution of a job until it reattaches to a terminal.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
005506  010146                  WINSPN: MOV     R1,-(SP)
                                ;
                                ;  Never suspend if we are running at fork level (doing character echoing)
                                ;
005510  105737  000000G                 TSTB    FRKPRI          ;Are we running at fork level?
005514  001012                          BNE     9$              ;Br if yes
                                ;
                                ;  See if job has reattached to the terminal
                                ;
005516  116201  000000G         1$:     MOVB    DW$JOB(R2),R1   ;Get job index number
005522  032761  000000G 000000G        BIT     #$VNOTT,LSW(R1) ;Is job attached to terminal?
005530  001404                          BEQ     9$              ;Br if yes
                                ;
                                ;  Suspend the job
                                ;
005532                                  QCALL   PCSPND          ;Suspend execution of the job
005540  000766                          BR      1$              ;Wait till job reconnects to terminal
                                ;
                                ;  Finished
                                ;
005542  012601                  9$:     MOV     (SP)+,R1
005544  000207                          RETURN
```

```
                                        .SBTTL  ERSLIN -- Erase the current line
                                ;-----------------------------------------------------------------------
                                ;  Erase the current line.  Do not change the line attributes and leave
                                ;  the cursor where it is.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
005546  010346                  ERSLIN: MOV     R3,-(SP)
                                ;
                                ;  Store blanks with all attributes cleared into all line columns
                                ;
005550  016203  0000006                 MOV     DW$LPT(R2),R3   ;Get pointer to 1st char in line
005554  016200  0000004                 MOV     DW$CPL(R2),R0   ;Get number of columns per line
005560                          1$:     BUFMAP                  ;;;Map to display buffer
005602  112723  000040                  MOVB    #40,(R3)+       ;;;Store blank as the character
005606  105023                          CLRB    (R3)+           ;;;Clear all attribute flags
005610                                  UNMAP                   ;Restore mapping
005624  077023                          SOB     R0,1$           ;Clear entire line
                                ;
                                ;  Finished
                                ;
005626  012603                          MOV     (SP)+,R3
005630  000207                          RETURN
```

```
                        .SBTTL  ERSCTL -- Erase from cursor to the end of the line
                ;-----------------------------------------------------------------------
                ;  Erase all characters from the cursor position to the end of the line.
                ;  The cursor does not move.  Character attributes are reset for all
                ;  erased characters.
                ;
                ;  Inputs:
                ;   R2 = Pointer to window control block
                ;
005632  010346  ERSCTL: MOV     R3,-(SP)
005634  010446          MOV     R4,-(SP)
                ;
                ;  Get pointer to current cursor position and to end of line
                ;
005636  016203  0000000         MOV     DW$COL(R2),R3   ;Get current column number
005642  005303                  DEC     R3              ;Convert to column offset
005644  006303                  ASL     R3              ;Two bytes are used by each column
005646  066203  0000000         ADD     DW$LPT(R2),R3   ;Point to byte for current column
005652  016204  0000000         MOV     DW$CPL(R2),R4   ;Get number of columns per line
005656  005304                  DEC     R4              ;Convert to offset
005660  006304                  ASL     R4              ;Two bytes per column
005662  066204  0000000         ADD     DW$LPT(R2),R4   ;Get pointer to byte for last column
                ;
                ;  Clear the line to blanks
                ;
005666  1$:     BUFMAP                          ;;;Map to display buffer
005710  112723  000040          MOVB    #40,(R3)+       ;;;Store blank as character
005714  105023                  CLRB    (R3)+           ;;;Clear attribute flags
005716                          UNMAP                   ;Restore mapping
005732  020304                  CMP     R3,R4           ;Finished line?
005734  101754                  BLOS    1$              ;Loop if not
                ;
                ;  Finished
                ;
005736  012604                  MOV     (SP)+,R4
005740  012603                  MOV     (SP)+,R3
005742  000207                  RETURN
```

```
                                        .SBTTL  ERSLTC -- Erase from beginning of line to cursor
                                ;-----------------------------------------------------------------------
                                ;  Erase from the beginning of the line up to and including the character
                                ;  pointed to by the cursor.  The cursor does not move.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;
005744  010346                  ERSLTC: MOV     R3,-(SP)
                                ;
                                ;  Set up pointer to start of line and to cursor position
                                ;
005746  016203  0000000                 MOV     DW$LPT(R2),R3   ;Get pointer to 1st byte for the line
005752  016200  0000000                 MOV     DW$COL(R2),R0   ;Get current cursor column
                                ;
                                ;  Blank the characters
                                ;
005756                          1$:     BUFMAP                  ;;;Map to display buffer
006000  112723  000040                  MOVB    #40,(R3)+       ;;;Store blank as the character
006004  105023                          CLRB    (R3)+           ;;;Clear attribute flags
006006                                  UNMAP                   ;Restore mapping
006022  077023                          SOB     R0,1$           ;Loop if more characters to clear
                                ;
                                ;  Finished
                                ;
006024  012603                          MOV     (SP)+,R3
006026  000207                          RETURN
```

```
                                        .SBTTL  ERSCTP -- Erase from cursor to the end of the page
                                ;-----------------------------------------------------------------------
                                ;  Erase from the current cursor position to the end of the page.
                                ;  The cursor does not move.  Line attributes for completely erased
                                ;  lines are reset.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
006030  010546                  ERSCTP: MOV     R5,-(SP)
006032  016246  0000000                 MOV     DW$LIN(R2),-(SP);Save the current line number
                                ;
                                ;  Initialize the line number to the current line
                                ;
006036  016205  0000000                 MOV     DW$LIN(R2),R5   ;Get current line number
                                ;
                                ;  Determine if we are clearing all of the current line
                                ;
006042  026227  0000000 000001          CMP     DW$COL(R2),#1   ;Is cursor at column 1 of current line?
006050  001403                          BEQ     1$              ;Br if yes -- Clear all of current line
                                ;
                                ;  Clear from cursor to end of current line
                                ;
006052  004737  005632'                 CALL    ERSCTL          ;Erase to end of current line
006056  005205                          INC     R5              ;Advance line number
                                ;
                                ;  Now clear all lines below current line
                                ;
006060  020562  0000000         1$:     CMP     R5,DW$LPP(R2)   ;Have we erased all lines?
006064  101005                          BHI     9$              ;Br if yes
006066  010500                          MOV     R5,R0           ;Get number of line to be cleared
006070  004737  006232'                 CALL    CLRLIN          ;Clear the line
006074  005205                          INC     R5              ;Increment line number
006076  000770                          BR      1$              ;See if there are more to clear
                                ;
                                ;  Finished
                                ;
006100  012600                  9$:     MOV     (SP)+,R0        ;Recover original line number
006102  004737  006702'                 CALL    SETLIN          ;Restore line number
006106  012605                          MOV     (SP)+,R5
006110  000207                          RETURN
```

```
                                        .SBTTL  ERSPTC -- Erase from beginning of page to cursor
                                ;-----------------------------------------------------------------
                                ;  Erase all of the characters from the beginning of the page up to and
                                ;  including the character under the cursor.  Line attributes are cleared
                                ;  for lines that are completely erased.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
006112  010446                  ERSPTC: MOV     R4,-(SP)
006114  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize line numbers
                                ;
006116  016205  0000000                 MOV     DW$LIN(R2),R5   ;Get number of current line
006122  012704  000001                  MOV     #1,R4           ;Get number of first line
                                ;
                                ;  Erase all lines up to, but not including, current line
                                ;
006126  020405                  1$:     CMP     R4,R5           ;Have we reached current line?
006130  103005                          BHIS    2$              ;Br if yes
006132  010400                          MOV     R4,R0           ;Get number of line to erase
006134  004737  006232'                 CALL    CLRLIN          ;Erase the line
006140  005204                          INC     R4              ;Advance the line number
006142  000771                          BR      1$              ;See if more lines to erase
                                ;
                                ;  Erase the line the cursor is on
                                ;
006144  010400                  2$:     MOV     R4,R0           ;Get number of current line
006146  026262  0000000 0000000         CMP     DW$COL(R2),DW$CPL(R2);Is cursor at the end of the line?
006154  001003                          BNE     3$              ;Br if not
006156  004737  006232'                 CALL    CLRLIN          ;Clear it
006162  000404                          BR      9$
006164  004737  006702'          3$:     CALL    SETLIN          ;Select current line
006170  004737  005744'                 CALL    ERSLTC          ;Erase up to cursor
                                ;
                                ;  Finished
                                ;
006174  010500                  9$:     MOV     R5,R0           ;Get back current line number
006176  004737  006702'                 CALL    SETLIN          ;Select as current line
006202  012605                          MOV     (SP)+,R5
006204  012604                          MOV     (SP)+,R4
006206  000207                          RETURN
```

```
                                        .SBTTL  ERSPAG -- Clear entire page to spaces
                                ;-----------------------------------------------------------------------
                                ;  Clear the entire page to spaces
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
006210  010546                  ERSPAG: MOV     R5,-(SP)
                                ;
                                ;  Begin loop to clear all lines
                                ;
006212  016205  0000000                 MOV     DW$LPP(R2),R5   ;Get last line number
006216  010500                  1$:     MOV     R5,R0           ;Get number of line to clear
006220  004737  006232'                 CALL    CLRLIN          ;Clear line to spaces
006224  077504                          SOB     R5,1$           ;Loop if more lines to clear
                                ;
                                ;  Finished
                                ;
006226  012605                          MOV     (SP)+,R5
006230  000207                          RETURN
```

```
                                        .SBTTL  CLRLIN -- Clear a line to spaces
                                ;-----------------------------------------------------------------------
                                ;  Clear a line to spaces and set the line attributes to single high,
                                ;  single wide.
                                ;
                                ;  Inputs:
                                ;   R0 = Number of line to be cleared
                                ;   R2 = Pointer to window control block
                                ;
006232  010346                  CLRLIN: MOV     R3,-(SP)
006234  010446                          MOV     R4,-(SP)
006236  016246  0000000                 MOV     DW$LIN(R2),-(SP);Save current line number
                                ;
                                ;  Select the line to be cleared
                                ;
006242  004737  006702'                 CALL    SETLIN          ;Select line to be cleared
                                ;
                                ;  Blank the line
                                ;
006246  016203  0000000                 MOV     DW$LPT(R2),R3   ;Get pointer to start of line
006252  016204  0000000                 MOV     DW$CPL(R2),R4   ;Get # columns in the line
006256                           1$:    BUFMAP                  ;;;Map to screen buffer
006300  112723  000040                  MOVB    #40,(R3)+       ;;;Store blank character
006304  105023                          CLRB    (R3)+           ;;;Clear character attributes
006306                                  UNMAP                   ;Restore mapping
006322  077423                          SOB     R4,1$           ;Loop if need to store more blanks
                                ;
                                ;  Reset line attributes
                                ;
006324  005000                          CLR     R0              ;Set attribute to 0
006326  004737  006522'                 CALL    SETLAB          ;Reset all line attributes
                                ;
                                ;  Finished
                                ;
006332  012600                          MOV     (SP)+,R0        ;Get back original line #
006334  004737  006702'                 CALL    SETLIN          ;Reselect original line
006340  012604                          MOV     (SP)+,R4
006342  012603                          MOV     (SP)+,R3
006344  000207                          RETURN
```

```
                                        .SBTTL  CPYLIN -- Copy characters from one line to another
                                ;-----------------------------------------------------------------------
                                ;  Copy the characters and attributes from one line to another.
                                ;  The line attribute is also copied.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R4 = Number of line to copy from
                                ;   R5 = Number of line to copy to
                                ;
006346  010346                  CPYLIN: MOV     R3,-(SP)
006350  010446                          MOV     R4,-(SP)
006352  010546                          MOV     R5,-(SP)
006354  016246  0000000                 MOV     DW$LIN(R2),-(SP);Save current line number
                                ;
                                ;  Copy the line attribute byte
                                ;
006360  010400                          MOV     R4,R0           ;Get source line #
006362  004737  006702'                 CALL    SETLIN          ;Set up info about that line
006366  004737  006612'                 CALL    GETLAB          ;Get line attribute information (to R0)
006372  010003                          MOV     R0,R3           ;Save line attribute
006374  010500                          MOV     R5,R0           ;Get dest line #
006376  004737  006702'                 CALL    SETLIN          ;Set up info about that line
006402  010300                          MOV     R3,R0           ;Get line attribute byte
006404  004737  006522'                 CALL    SETLAB          ;Set line attribute byte
                                ;
                                ;  Get pointers to characters in from and to lines
                                ;
006410  010400                          MOV     R4,R0           ;Get # of line we are copying from
006412  004737  006702'                 CALL    SETLIN          ;Set up information about the line
006416  016204  0000000                 MOV     DW$LPT(R2),R4   ;Get pointer to 1st char in from line
006422  010500                          MOV     R5,R0           ;Get # of line we are copying to
006424  004737  006702'                 CALL    SETLIN          ;Set up information about the line
006430  016205  0000000                 MOV     DW$LPT(R2),R5   ;Get pointer to 1st char in to line
                                ;
                                ;  Begin loop to copy the characters
                                ;
006434  016200  0000000                 MOV     DW$CPL(R2),R0   ;Get # columns to copy
006440                          1$:     BUFMAP                  ;;;Map to screen buffer
006462  112425                          MOVB    (R4)+,(R5)+     ;;;Copy character
006464  112425                          MOVB    (R4)+,(R5)+     ;;;Copy attribute
006466                                  UNMAP                   ;Restore mapping
006502  077022                          SOB     R0,1$           ;Loop to copy all chars in the line
                                ;
                                ;  Finished
                                ;
006504  012600                          MOV     (SP)+,R0        ;Get back original line #
006506  004737  006702'                 CALL    SETLIN          ;Reselect that line
006512  012605                          MOV     (SP)+,R5
006514  012604                          MOV     (SP)+,R4
006516  012603                          MOV     (SP)+,R3
006520  000207                          RETURN
```

```
                                        .SBTTL  SETLAB -- Set attributes for current line
                                ;-----------------------------------------------------------------------
                                ;  SETLAB is called to set the line attribute byte for the current line.
                                ;
                                ;  Inputs:
                                ;   R0 = Line attribute byte value
                                ;   R2 = Pointer to window control block
                                ;
006522  010346                  SETLAB: MOV     R3,-(SP)
                                ;
                                ;  Compute virtual address of byte with attribute for current line
                                ;
006524  016203  0000000                 MOV     DW$LIN(R2),R3   ;Get current line number
006530  166203  0000000                 SUB     DW$TLN(R2),R3   ;Get offset from line at top of buffer
006534  002002                          BGE     1$              ;Br if line greater than 1st line
006536  066203  0000000                 ADD     DW$LPP(R2),R3   ;Get offset to line we want
006542  062703  0000000         1$:     ADD     #VPAR6,R3       ;Add virtual base address
                                ;
                                ;  Store the new attribute byte
                                ;
006546                                  BUFMAP                  ;;;Map to screen buffer
006570  110013                          MOVB    R0,(R3)         ;;;Store new attribute for line
006572                                  UNMAP                   ;Restore mapping
                                ;
                                ;  Finished
                                ;
006606  012603                          MOV     (SP)+,R3
006610  000207                          RETURN
```

```
                                        .SBTTL  GETLAB -- Get attributes for current line
                                ;-----------------------------------------------------------------
                                ;  GETLAB is called to get the line attribute byte for the current line.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
                                ;  Outputs:
                                ;   R0 = Line attribute byte for current line.
                                ;
006612  010346                  GETLAB: MOV     R3,-(SP)
                                ;
                                ;  Compute virtual address of byte with attribute for current line
                                ;
006614  016203  0000000                 MOV     DW$LIN(R2),R3   ;Get current line number
006620  166203  0000000                 SUB     DW$TLN(R2),R3   ;Get offset from line at top of buffer
006624  002002                          BGE     1$              ;Br if line greater than 1st line
006626  066203  0000000                 ADD     DW$LPP(R2),R3   ;Get offset to line we want
006632  062703  0000000         1$:     ADD     #VPAR6,R3       ;Add virtual base address
                                ;
                                ;  Get the attribute byte
                                ;
006636                                  BUFMAP                  ;;;Map to screen buffer
006660  111300                          MOVB    (R3),R0         ;;;Get the line attribute byte
006662                                  UNMAP                   ;Restore mapping
                                ;
                                ;  Finished
                                ;
006676  012603                  MOV     (SP)+,R3
006700  000207                  RETURN
```

```
                                        .SBTTL  SETLIN -- Select a line as current line
                                ;-----------------------------------------------------------------------
                                ;  Select a specified line as the current line and set up pointer information
                                ;  for it.
                                ;
                                ;  Inputs:
                                ;   R0 = Line number being selected (1-24)
                                ;   R2 = Pointer to window control block
                                ;
                                ;  Outputs:
                                ;   DW$LIN(R2) = Set to current line number
                                ;   DW$LPT(R2) = Virtual address of 1st character on current line
                                ;
006702  010146                  SETLIN: MOV     R1,-(SP)
                                ;
                                ;  Set the current line number in WCB
                                ;
006704  010062  0000000                 MOV     R0,DW$LIN(R2)   ;Set current line number
                                ;
                                ;  Compute virtual address of 1st character in this line
                                ;
006710  166200  0000000                 SUB     DW$TLN(R2),R0   ;Get # lines from top to line wanted
006714  002002                          BGE     1$              ;Br if top line is LE one wanted
006716  066200  0000000                 ADD     DW$LPP(R2),R0   ;Add total number of lines per page
006722  010001                  1$:     MOV     R0,R1           ;Get # lines offset to line we want
006724  070162  0000000                 MUL     DW$CPL(R2),R1   ;Multiply by number of columns per line
006730  006301                          ASL     R1              ;Times two because of attribute bytes
006732  066201  0000000                 ADD     DW$LPP(R2),R1   ;Skip over bytes with line attributes
006736  062701  0000000                 ADD     #VPAR6,R1       ;Add base virtual address
006742  010162  0000000                 MOV     R1,DW$LPT(R2)   ;This is virtual addr of start of line
                                ;
                                ;  Finished
                                ;
006746  012601                          MOV     (SP)+,R1
006750  000207                          RETURN
```

```
                                        .SBTTL  WINSF  -- Switch from a job with a display window
                                ;-----------------------------------------------------------------------
                                ;  WINSF is called from TSTTY when we are switching from a job which
                                ;  has an active display window.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to display window control block for job being switched from
                                ;
006752                          WINSF:
                                ;
                                ;  Set flag which prevents characters from being passed through to the
                                ;  screen.
                                ;
006752  052762  0000000 0000000         BIS     #AW$DDC,DW$AW(R2);Don't pass through any more chars
                                ;
                                ;  Reset number of lines scrolled since we disconnected from window
                                ;
006760  105062  0000000                 CLRB    DW$NSL(R2)      ;No lines scrolled yet
                                ;
                                ;  Finished
                                ;
006764  000207                          RETURN
```

```
                                        .SBTTL  WINST  -- Switch to job with a window
                                ;-----------------------------------------------------------------
                                ;  WINST is called from TSTTY when we are switching to a job that has
                                ;  a display window active.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to current window control block for job being switched to
                                ;
006766  010146                  WINST:  MOV     R1,-(SP)
                                ;
                                ;  Get job index number
                                ;
006770  116201  000000G                 MOVB    DW$JOB(R2),R1   ;Get job index number
                                ;
                                ;  Set flag which will cause the window to be redisplayed when the
                                ;  job resumes execution.
                                ;
006774  052761  000000G 000000G         BIS     #$WDISP,LSW6(R1);Set flag saying to redisplay window
                                ;
                                ;  If job is in terminal-output-wait state, restart its execution
                                ;
007002  026127  000000G 000000G         CMP     LSTATE(R1),#S$OTWT;TT-output-wait state?
007010  001002                          BNE     9$              ;Br if not
007012  004737  000000G                 CALL    FORCEX          ;Restart execution of job
                                ;
                                ;  Finished
                                ;
007016  012601                  9$:     MOV     (SP)+,R1
007020  000207                          RETURN
```

```
                                        .SBTTL  WINDSP -- Redisplay current window for job
                                ;-----------------------------------------------------------------------
                                ;  WINDSP is called from the scheduler when it is about ready to
                                ;  reenter a job whose window needs to be redisplayed.
                                ;
                                ;  Inputs:
                                ;   R1 = Current job index number
                                ;
007022  010246                  WINDSP: MOV     R2,-(SP)
                                ;
                                ;  Reset flag that says window redisplay needed
                                ;  and set flag that says a window refresh is being done.
                                ;
007024  042761  000000G 000000G         BIC     #$WDISP,LSW6(R1);Window redisplay no longer needed
007032  052761  000000G 000000G         BIS     #$RFRSH,LSW4(R1);Window refresh currently being done
                                ;
                                ;  Redisplay current window for job
                                ;
007040  016102  000000G                 MOV     LWINDO(R1),R2   ;Get addr of current window control block
007044  001402                          BEQ     9$              ;Br if job is not doing windowing
007046  004737  007064'                 CALL    REFRSH          ;Redisplay window
                                ;
                                ;  Finished
                                ;
007052  042761  000000G 000000G 9$:      BIC     #$RFRSH,LSW4(R1);Say window refresh is finished
007060  012602                          MOV     (SP)+,R2
007062  000207                          RETURN
```

```
                                        .SBTTL  REFRSH -- Refresh screen from window contents
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to clear the screen and refresh its contents
                                ;  from the screen buffer.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
007064  010146                  REFRSH: MOV     R1,-(SP)
007066  010546                          MOV     R5,-(SP)
                                ;
                                ;  Stop characters from being passed from program to screen while we are
                                ;  doing the refresh
                                ;
007070  052762  0000000 0000000         BIS     #AW$DDC,DW$AW(R2);Suppress character pass-through
                                ;
                                ;  Clear contents of line's output buffer and reset control-O flag
                                ;
007076  116201  0000000                 MOVB    DW$JOB(R2),R1   ;Get job index number
007102                                  DISABL                  ;;;Disable interrupts
007110  016161  0000000 0000000         MOV     LOTSIZ(R1),LOTSPC(R1);;;Clear contents of line's output buffer
007116  016161  0000000 0000000         MOV     LOTNXT(R1),LOTPNT(R1);;;
007124                                  ENABL                   ;Enable interrupts
007132  042761  0000000 0000000         BIC     #$CTRLO,LSW3(R1);Reset control-O flag
                                ;
                                ;  See if we need to turn VT52 emulation mode on or off
                                ;
007140  004737  011110'                 CALL    GENTEM          ;Check VT52 emulation mode status
                                ;
                                ;  Clear the screen
                                ;
007144  004737  007612'                 CALL    GENCLR          ;Clear the screen
                                ;
                                ;  Set overall window attributes
                                ;
007150  004737  011240'                 CALL    GENWAF          ;Set overall window attributes
                                ;
                                ;  Initialize character attribute flags
                                ;
007154  005005                          CLR     R5              ;No character attribute flags
                                ;
                                ;  Begin loop to send each line to the screen
                                ;
007156  016246  0000000                 MOV     DW$LIN(R2),-(SP);Save current line number
007162  012701  000001                  MOV     #1,R1           ;Get 1st line number
007166  010100                  1$:     MOV     R1,R0           ;Get line number
007170  004737  006702'                 CALL    SETLIN          ;Select as current line
007174  004737  007302'                 CALL    SNDLIN          ;Send line to screen
007200  005201                          INC     R1              ;Increment line number
007202  020162  0000000                 CMP     R1,DW$LPP(R2)   ;Have we done all lines?
007206  101767                          BLOS    1$              ;Loop if not
                                ;
                                ;  Reset line number
                                ;
007210  012600                          MOV     (SP)+,R0        ;Get back line where we should be
007212  004737  006702'                 CALL    SETLIN          ;Set this as the current line
                                ;
```

```
                                        ;  Set up scrolling region status
                                        ;
007216  004737  010746'                       CALL    GENSSS          ;Set up split screen scrolling
                                        ;
                                        ;  Place cursor where it should be
                                        ;
007222  004737  012104'                       CALL    GENCSR          ;Gen cursor positioning commands
007226  016200  000000G                       MOV     DW$COL(R2),R0   ;Get current column number
007232  005300                              DEC     R0              ;Make 1st column be 0
007234  116201  000000G                       MOVB    DW$JOB(R2),R1   ;Get our job number
007240  110061  000000G                       MOVB    R0,LCOL(R1)     ;Tell TSTTY where cursor is
                                        ;
                                        ;  Make sure character attribute is set correctly
                                        ;
007244  120562  000000G                       CMPB    R5,DW$CCA(R2)   ;Is character attribute set correctly?
007250  001404                              BEQ     2$              ;Br if yes
007252  116200  000000G                       MOVB    DW$CCA(R2),R0   ;Get needed character attribute
007256  004737  010122'                       CALL    GENCAF          ;Set character attribute
                                        ;
                                        ;  Set up correct character set mapping
                                        ;
007262  004737  010534'                 2$:   CALL    GENMAP          ;Set up correct character set mapping
                                        ;
                                        ;  Now allow characters to be passed to the screen
                                        ;
007266  042762  000000G 000000G               BIC     #AW$DDC,DW$AW(R2);Enable character pass-through
                                        ;
                                        ;  Finished
                                        ;
007274  012605                          9$:   MOV     (SP)+,R5
007276  012601                              MOV     (SP)+,R1
007300  000207                              RETURN
```

```
                                        .SBTTL  SNDLIN -- Send line of characters to terminal
                                ;-----------------------------------------------------------------------
                                ;  Send a line of characters from the screen buffer to the terminal.
                                ;  The currently selected line is the one sent.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block.
                                ;   R5 = Current character attribute flags
                                ;
                                ;  Outputs:
                                ;   R5 = Character attribute flags for last character on line
                                ;
007302  010146                  SNDLIN: MOV     R1,-(SP)
007304  010346                          MOV     R3,-(SP)
007306  010446                          MOV     R4,-(SP)
007310  016246  0000000                 MOV     DW$COL(R2),-(SP);Save current column number
                                ;
                                ;  Set line attribute flag and determine number of columns to display
                                ;
007314  004737  006612'                 CALL    GETLAB          ;Get line attribute flags for cur line to R0
007320  012701  000120                  MOV     #80.,R1         ;Assume 80 columns to display
007324  032762  0000000@ 0000000@       BIT     #AW$132,DW$AW(R2);Are we in 132 column mode?
007332  001402                          BEQ     1$              ;Br if not in 132 col mode
007334  012701  000204                  MOV     #132.,R1        ;Say we need to display 132 columns
007340  132700  000000C         1$:     BITB    #<AL$DHT!AL$DHB!AL$DWD>,R0 ;Double wide line?
007344  001401                          BEQ     3$              ;Br if not double wide
007346  006201                          ASR     R1              ;Only need to display half as many chars
007350  004737  011622'           3$:     CALL    GENLAF          ;Generate line attribute flag info
                                ;
                                ;  Store number of columns to display on top of stack
                                ;
007354  020162  0000000                 CMP     R1,DW$CPL(R2)   ;Are we storing as many cols as needed?
007360  101402                          BLOS    5$              ;Br if yes
007362  016201  0000000                 MOV     DW$CPL(R2),R1   ;Get # columns we are storing
007366  010146                  5$:     MOV     R1,-(SP)        ;Store # columns to display on top of stack
                                ;
                                ;  Begin loop to output the characters on the line
                                ;
007370  012701  000001                  MOV     #1,R1           ;Init column number
007374  010162  0000000                 MOV     R1,DW$COL(R2)   ;Init col where next character goes
007400  016203  0000000                 MOV     DW$LPT(R2),R3   ;Get pointer to 1st char on this line
                                ;
                                ;  Scan over spaces that have no attributes set
                                ;
007404                          4$:     BUFMAP                  ;;;Map to screen buffer
007426  112304                          MOVB    (R3)+,R4        ;;;Get character
007430  112300                          MOVB    (R3)+,R0        ;;;Get attribute flags for character
007432                                  UNMAP                   ;Restore mapping
007446  120427  000040                  CMPB    R4,#40          ;Is character a space?
007452  001005                          BNE     2$              ;Br if not
007454  120005                          CMPB    R0,R5           ;Are attributes changing?
007456  001003                          BNE     2$              ;Br if yes
007460  032700  000000C                 BIT     #<AC$BLD!AC$BLK!AC$REV!AC$ULN>,R0 ;Bold, blink, reverse, under
007464  001420                          BEQ     8$              ;Br if not
                                ;
                                ;  We found a character that we need to send.
                                ;  (R4 = character,  R0 = attribute flags)
```

```
                                        ;  Position to correct column for the character.
                                        ;
007466  020162  0000000         2$:     CMP     R1,DW$COL(R2)   ;Are we at correct column now?
007472  001404                          BEQ     6$              ;Br if yes
007474  010046                          MOV     R0,-(SP)        ;Save character attribute flags
007476  004737  011736'                GENSPC                  ;Generate spaces to move to correct col
007502  012600                          MOV     (SP)+,R0        ;Restore character attribute flags
                                        ;
                                        ;  See if we need to change attributes for this character
                                        ;
007504  120005                  6$:     CMPB    R0,R5           ;Do we need to change attributes?
007506  001402                          BEQ     7$              ;Br if not
007510  004737  010122'                GENCAF                  ;Set character attribute flags
                                        ;
                                        ;  Send the character to the terminal
                                        ;
007514  010400                  7$:     MOV     R4,R0           ;Get character to send
007516  004737  012314'                GENCHR                  ;Send the character to the terminal
007522  005262  0000000                 INC     DW$COL(R2)      ;Increment current column number
                                        ;
                                        ;  Advance to next character
                                        ;
007526  005201                  8$:     INC     R1              ;Increment column number
007530  020116                          CMP     R1,(SP)         ;Done all columns?
007532  101724                          BLOS    4$              ;Loop if not
                                        ;
                                        ;  Output carriage-return line-feed at the end of the line
                                        ;
007534  026262  0000000 0000000         CMP     DW$LIN(R2),DW$LPP(R2);Is this the last line on the page?
007542  001414                          BEQ     20$             ;Br if yes -- Don't send CR-LF
007544  026227  0000000 000001          CMP     DW$COL(R2),#1   ;Are we still at col 1 (entire line blank)?
007552  001404                          BEQ     10$             ;Br if yes
007554                                  SEND    #CR             ;Send carriage return
007564                          10$:    SEND    #LF             ;Send line feed
                                        ;
                                        ;  Finished
                                        ;
007574  005726                  20$:    TST     (SP)+           ;Pop number of columns per line
007576  012662  0000000                 MOV     (SP)+,DW$COL(R2);Restore cursor column number
007602  012604                          MOV     (SP)+,R4
007604  012603                          MOV     (SP)+,R3
007606  012601                          MOV     (SP)+,R1
007610  000207                          RETURN
```

```
                                        .SBTTL  GENCLR -- Gen control sequence to clear screen
                                ;-----------------------------------------------------------------------
                                ;  Generate the appropriate terminal control sequence to clear the screen.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
007612                          GENCLR:
                                ;
                                ;  See if we need to generate a VT100 or VT52 sequence
                                ;
007612  032762  000000C 000000G         BIT     #<AW$52!AW$G52>,DW$AW(R2);VT52?
007620  001115                          BNE     1$                      ;Br if yes
                                ;
                                ;  Generate VT100 sequence
                                ;  Reset scrolling region
                                ;
007622  004737  012272'                 CALL    GENCSI                  ;Generate CSI header
007626                                  SEND    #'r                     ;Reset scrolling region
                                ;
                                ;  Set absolute origin
                                ;
007636  004737  012272'                 CALL    GENCSI                  ;Generate CSI header
007642                                  SEND    #'?                     ;Send ?
007652                                  SEND    #'6                     ;Send 6
007662                                  SEND    #'1                     ;Send 1
                                ;
                                ;  Home cursor and clear to end of screen
                                ;
007672  004737  012272'                 CALL    GENCSI                  ;Generate CSI header
007676                                  SEND    #'H                     ;Send H -- Home cursor
007706  004737  012272'                 CALL    GENCSI                  ;Generate CSI header
007712                                  SEND    #'J                     ;Send J -- Clear to end of screen
                                ;
                                ;  Reset character attributes
                                ;
007722  004737  012272'                 CALL    GENCSI                  ;Send CSI 0 m  to reset char attributes
007726                                  SEND    #'0
007736                                  SEND    #'m
                                ;
                                ;  Map GL to G0 and designate G0 as ascii
                                ;
007746                                  SEND    #17                     ;Send shift-in -- Lock shift G0 to GL
007756  004737  012260'                 CALL    GENESC                  ;Send escape
007762                                  SEND    #'(                     ;Select G0
007772                                  SEND    #'B                     ;Map G0 to ascii
                                ;
                                ;  Map GR to G2 and designate G2 as ascii
                                ;
010002  032762  0000000 0000000         BIT     #AW$200,DW$AW(R2);Is this a VT200 terminal?
010010  001443                          BEQ     9$                      ;Br if not -- Don't have to mess with GR
010012  004737  012260'                 CALL    GENESC                  ;Send ESC
010016                                  SEND    #'}                     ;Lock shift GR to G2
010026  004737  012260'                 CALL    GENESC                  ;Send escape
010032                                  SEND    #'*                     ;Select G2
010042                                  SEND    #'B                     ;Designate ascii
010052  000422                          BR      9$
```

```
                                        ;
                                        ;  Generate VT52 sequence
                                        ;
010054  004737  012260'         1$:     CALL    GENESC          ;Generate ESC character
010060                                  SEND    #'H             ;Send H -- Home cursor
010070  004737  012260'                 CALL    GENESC          ;Generate ESC character
010074                                  SEND    #'J             ;Send J -- Clear to end of screen
010104  004737  012260'                 CALL    GENESC          ;Generate ESC character
010110                                  SEND    #'G             ;Select ascii character set
                                        ;
                                        ;  Finished
                                        ;
010120  000207                  9$:     RETURN
```

```
                                        .SBTTL  GENCAF -- Generate sequence to set character attributes
                                ;-----------------------------------------------------------------------
                                ;  Generate the terminal sequence to set character attributes.
                                ;
                                ;  Inputs:
                                ;   R0 = Character attribute wanted
                                ;   R2 = Pointer to window control block
                                ;   R5 = Current character attribute
                                ;
                                ;  Outputs:
                                ;   R5 = New character attributes
                                ;
010122  010146                  GENCAF: MOV     R1,-(SP)
010124  010346                          MOV     R3,-(SP)
010126  010446                          MOV     R4,-(SP)
010130  010001                          MOV     R0,R1           ;Carry new attributes in R1
                                ;
                                ;  See if we need to change character sets
                                ;
010132  004737  010354'                 CALL    GENCSC          ;See if we need to do character set change
                                ;
                                ;  Don't have to worry about attributes if this is a VT52
                                ;
010136  032762  000000C 000000G         BIT     #<AW$52!AW$552>,DW$AW(R2);Is this a VT52?
010144  001076                          BNE     9$              ;Br if yes
                                ;
                                ;  See if character attributes have changed
                                ;
010146  010103                          MOV     R1,R3           ;Get new attributes
010150  010504                          MOV     R5,R4           ;Get current attributes
010152  042703  000000C                 BIC     #^C<AC$BLD!AC$BLK!AC$REV!AC$ULN>,R3
010156  042704  000000C                 BIC     #^C<AC$BLD!AC$BLK!AC$REV!AC$ULN>,R4
010162  020304                          CMP     R3,R4           ;Have attributes changed?
010164  001466                          BEQ     9$              ;Br if not
                                ;
                                ;  Initialize control sequence and turn off all attributes
                                ;
010166  004737  012272'                 CALL    GENCSI          ;Send CSI
010172                                  SEND    #'0             ;Send 0
                                ;
                                ;  See if bold is wanted
                                ;
010202  032701  0000000                 BIT     #AC$BLD,R1      ;Is bold wanted?
010206  001410                          BEQ     1$              ;Br if not
010210                                  SEND    #SEMI           ;Send ';1'
010220                                  SEND    #'1
                                ;
                                ;  See if underscore is wanted
                                ;
010230  032701  0000000         1$:     BIT     #AC$ULN,R1      ;Underline wanted?
010234  001410                          BEQ     2$              ;Br if not
010236                                  SEND    #SEMI           ;Send ";4"
010246                                  SEND    #'4
                                ;
                                ;  See if blinking is wanted
                                ;
010256  032701  0000000         2$:     BIT     #AC$BLK,R1      ;Blinking wanted?
```

```
     58 010262  001410                          BEQ     3$                ;Br if not
     59 010264                                  SEND    #SEMI             ;Send ";5"
     60 010274                                  SEND    #'5
     61                                 ;
     62                                 ;  See if reverse video is wanted
     63                                 ;
     64 010304  032701  0000000         3$:     BIT     #AC$REV,R1        ;Reverse video wanted?
     65 010310  001410                          BEQ     7$                ;Br if not
     66 010312                                  SEND    #SEMI             ;Send ";7"
     67 010322                                  SEND    #'7
     68                                 ;
     69                                 ;  Terminate the string by sending "m"
     70                                 ;
     71 010332                          7$:     SEND    #'m               ;Send "m"
     72                                 ;
     73                                 ;  Finished
     74                                 ;
     75 010342  010105                  9$:     MOV     R1,R5             ;Change current attributes to new attributes
     76 010344  012604                          MOV     (SP)+,R4
     77 010346  012603                          MOV     (SP)+,R3
     78 010350  012601                          MOV     (SP)+,R1
     79 010352  000207                          RETURN
```

```
                                        .SBTTL  GENCSC -- Generate terminal sequence to select char set
                                ;-----------------------------------------------------------------------
                                ;  Generate terminal control sequence to select the correct character
                                ;  set for the next character.
                                ;
                                ;  Inputs:
                                ;   R0 = Character attribute wanted
                                ;   R2 = Pointer to window control block
                                ;   R5 = Current character attribute
                                ;
010354  010046                  GENCSC: MOV     R0,-(SP)
010356  010146                          MOV     R1,-(SP)
010360  010346                          MOV     R3,-(SP)
010362  010446                          MOV     R4,-(SP)
010364  010001                          MOV     R0,R1           ;Carry new attributes in R1
                                ;
                                ;  Determine if character set has changed
                                ;
010366  010103                          MOV     R1,R3           ;Get new attributes
010370  042703  000000C                 BIC     #^C<AC$SET>,R3  ;Select character set number only
010374  010504                          MOV     R5,R4           ;Get current attributes
010376  042704  000000C                 BIC     #^C<AC$SET>,R4  ;Select character set number only
010402  020304                          CMP     R3,R4           ;Has character set changed?
010404  001443                          BEQ     9$              ;Br if char set has not changed
                                ;
                                ;  Character set has changed
                                ;
010406  032762  000000C 000000G         BIT     #<AW$52!AW$552>,DW$AW(R2);Is this a VT52 terminal?
010414  001031                          BNE     1$              ;Br if VT52
                                ;
                                ;  VT100/VT200 terminal.
                                ;  Generate sequence to map G0 and G2 to correct character set
                                ;
010416  004737  012260'                 CALL    GENESC          ;Send ESC character
010422                                  SEND    #'(             ;Select G0
010432                                  SEND    SET200(R3)      ;Send char to map G0 to correct char set
010442  032762  0000000 0000000         BIT     #AW$200,DW$AW(R2);Is this a VT200 terminal?
010450  001421                          BEQ     9$              ;Br if not -- Don't need to change G2
010452  004737  012260'                 CALL    GENESC          ;Send ESC character
010456                                  SEND    #'*             ;Select G2
010466                                  SEND    SET200(R3)      ;Send char to map G2 to correct char set
010476  000406                          BR      9$
                                ;
                                ;  VT52 terminal.
                                ;  Select ascii or graphics character set.
                                ;
010500  004737  012260'         1$:     CALL    GENESC          ;Send ESC character
010504                                  SEND    SET52(R3)       ;Select ascii or graphics characters
                                ;
                                ;  Finished
                                ;
010514  012604                  9$:     MOV     (SP)+,R4
010516  012603                          MOV     (SP)+,R3
010520  012601                          MOV     (SP)+,R1
010522  012600                          MOV     (SP)+,R0
010524  000207                          RETURN
                                ;
```

```
                                        ;  Characters to be sent to change G0 character set designation
                                        ;
010526      102     074     101     SET200:  .ASCII  /B<AO/
010531      060
                                        ;
                                        ;  Characters to select ascii or graphics mode on a VT52
                                        ;
010532      107     106             SET52:   .ASCII  /GF/
                                             .EVEN
```

```
                                        .SBTTL  GENMAP -- Generate sequence to set up char set mapping
                                ;-----------------------------------------------------------------------
                                ;  Generate control sequence to set up the correct character set
                                ;  mapping for GL, GR, G0, G1, G2, and G3.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
010534  010346                  GENMAP: MOV     R3,-(SP)
010536  032762  000000C 0000000         BIT     #<AW$52!AW$S52>,DW$AW(R2);Is this a VT52 terminal?
010544  001060                          BNE     5$                      ;Br if VT52
                                ;
                                ;  This is a VT100/VT200 terminal.
                                ;  Set up mapping for GL.
                                ;
010546  116203  0000000               MOVB    DW$GLM(R2),R3           ;Get mapping for GL
010552  120327  000001                CMPB    R3,#1                   ;Mapping to G0 or G1?
010556  101402                      BLOS    1$                      ;Br if yes
010560  004737  012260'              CALL    GENESC                  ;Send ESC
010564                           1$:    SEND    LSLCHR(R3)              ;Send correct lock shift character
                                ;
                                ;  Set up mapping for GR
                                ;
010574  032762  0000000 0000000       BIT     #AW$200,DW$AW(R2);Is this a VT200?
010602  001410                      BEQ     2$                      ;Br if not -- Don't need to mess with GR
010604  116203  0000000             MOVB    DW$GRM(R2),R3           ;Get mapping for GR
010610  004737  012260'              CALL    GENESC                  ;Send ESC
010614                               SEND    LSRCHR(R3)              ;Send correct lock shift character
                                ;
                                ;  Set up mapping for G0, G1, G2, and G3
                                ;
010624  012703  000001           2$:    MOV     #1,R3                   ;Assume we only need to map G0 and G1
010630  032762  0000000 0000000       BIT     #AW$200,DW$AW(R2);Is this a VT200?
010636  001402                      BEQ     3$                      ;Br if not
010640  012703  000003               MOV     #3,R3                   ;Map G2 and G3 too
010644  004737  012260'           3$:    CALL    GENESC                  ;Send ESC
010650                               SEND    GNMCHR(R3)              ;Select G#
010660  010300                       MOV     R3,R0                   ;Get G number
010662  060200                       ADD     R2,R0                   ;Point into window control block
010664  116000  0000000             MOVB    DW$GOM(R0),R0           ;Get mapping wanted for G#
010670                               SEND    SET200(R0)              ;Set mapping for the G#
010700  005303                       DEC     R3                      ;Have more G# to do?
010702  002360                       BGE     3$                      ;Loop if yes
010704  000410                       BR      9$
                                ;
                                ;  Set up mapping for a VT52 terminal
                                ;
010706  004737  012260'           5$:    CALL    GENESC                  ;Generate ESC character
010712  116203  0000000             MOVB    DW$GOM(R2),R3           ;Get ascii/graphics mapping code
010716                               SEND    SET52(R3)               ;Select correct mapping
                                ;
                                ;  Finished
                                ;
010726  012603                   9$:    MOV     (SP)+,R3
010730  000207                       RETURN
                                ;
                                ;  Characters used to lock shift a mapping into GL
```

```
                                        ;
010732  017     016     156     LSLCHR: .ASCII  <17><16>/no/
010735  157
                                        ;
                                        ;  Characters used to lock shift a mapping into GR
                                        ;
010736  077     176     175     LSRCHR: .ASCII  /?~}|/
010741  174
                                        ;
                                        ;  Characters used to select G0, G1, G2, or G3 for mapping
                                        ;
010742  050     051     052     GNMCHR: .ASCII  /()*+/
010745  053
                                .EVEN
```

```
                                        .SBTTL  GENSSS -- Generate sequence for split screen scrolling
                                ;-----------------------------------------------------------------------
                                ;  Generate the terminal control sequence needed to set the terminal to
                                ;  split screen scrolling mode if that is needed.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
 010746                         GENSSS:
                                ;
                                ;  We never set split screen scrolling for VT52 terminals
                                ;
 010746  032762  000000C 000000G        BIT     #<AW$52!AW$S52>,DW$AW(R2);Is this a VT52?
 010754  001054                         BNE     9$                ;Br if yes
                                ;
                                ;  See if see need to set split screen scrolling
                                ;
 010756  026227  0000000 000001         CMP     DW$SRT(R2),#1     ;Is top line of scrolling region 1?
 010764  001004                         BNE     2$                ;Br if not
 010766  026262  0000000 0000000        CMP     DW$SRB(R2),DW$LPP(R2);Is bottom scroll line bottom of page?
 010774  001444                         BEQ     9$                ;Br if yes
                                ;
                                ;  Declare scrolling region
                                ;
 010776  004737  012272'        2$:     CALL    GENCSI            ;Generate CSI character
 011002  016200  0000000                MOV     DW$SRT(R2),R0     ;Get top line # of scrolling region
 011006  004737  012334'                CALL    GENVAL            ;Cvt to ascii digits and send them
 011012                                 SEND    #';               ;Send semicolon
 011022  016200  0000000                MOV     DW$SRB(R2),R0     ;Get bottom line of scrolling region
 011026  004737  012334'                CALL    GENVAL            ;Cvt to ascii digits and send them
 011032                                 SEND    #'r               ;Terminate control sequence
                                ;
                                ;  See if we need to set relative origin mode
                                ;
 011042  032762  0000000 0000000        BIT     #AW$ORS,DW$AW(R2);Is origin to be relative to scroll region?
 011050  001416                         BEQ     9$                ;Br if not
 011052  004737  012272'                CALL    GENCSI            ;Generate CSI character
 011056                                 SEND    #'?               ;Send ?
 011066                                 SEND    #'6               ;Send 6
 011076                                 SEND    #'h               ;Send h
                                ;
                                ;  Finished
                                ;
 011106  000207                 9$:     RETURN
```

```
                                                .SBTTL  GENTEM -- Turn VT52 emulation mode on or off
                                ;-----------------------------------------------------------------------
                                ;  Generate the terminal control sequence to turn VT52 emulation mode
                                ;  on or off as needed by this window.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
                                ;
                                ;  See if we need to go into VT52 emulation mode
                                ;
011110  116200  0000000         GENTEM: MOVB    DW$JOB(R2),R0   ;Get our job index number
011114  116000  0000000                 MOVB    LNPRIM(R0),R0   ;Get our primary line index number
011120  032762  0000000 0000000         BIT     #AW$S52,DW$AW(R2) ;Does window want VT52 emulation mode?
011126  001426                          BEQ     2$              ;Br if not
011130  032760  0000000 0000000         BIT     #$V52EM,LSW11(R0);Is VT52 emulation mode on now?
011136  001037                          BNE     9$              ;Br if yes
011140  052760  0000000 0000000         BIS     #$V52EM,LSW11(R0);Say we are turning on VT52 emulation
011146  004737  012272'                 CALL    GENCSI          ;Send CSI character
011152                                  SEND    #'?             ;Send "?21" to reset ANSI mode
011162                                  SEND    #'2
011172                                  SEND    #'1
011202  000415                          BR      9$
                                ;
                                ;  See if we need to turn off VT52 emulation mode
                                ;
011204  032760  0000000 0000000 2$:     BIT     #$V52EM,LSW11(R0);Is VT52 emulation mode on now?
011212  001411                          BEQ     9$              ;Br if not
011214  042760  0000000 0000000         BIC     #$V52EM,LSW11(R0);Say we are turning off VT52 emulation
011222  004737  012260'                 CALL    GENESC          ;Send ESC
011226                                  SEND    #'<             ;Tell VT52 to enter ANSI mode
                                ;
                                ;  Finished
                                ;
011236  000207                  9$:     RETURN
```

```
                                        .SBTTL  GENWAF -- Generate terminal sequence to set window attrib
                                ;-----------------------------------------------------------------------
                                ;  Generate the terminal control sequence needed to set attributes for
                                ;  the entire window.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
011240  010346                  GENWAF: MOV     R3,-(SP)
011242  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get control flags for the window attributes
                                ;
011244  016203  000000G                 MOV     DW$AW(R2),R3    ;Get window attribute flags
                                ;
                                ;  Generate control sequence to reset some flags
                                ;
011250  005103                          COM     R3              ;Set flags that should be reset
011252  012704  000154                  MOV     #'l,R4          ;Terminate string with lower-case L
011256  004737  011350'                 CALL    GENAAS          ;Generate for ANSI attributes
011262  004737  011412'                 CALL    GENDAS          ;Generate for DEC attributes
                                ;
                                ;  Generate control sequence to set some flags
                                ;
011266  005103                          COM     R3              ;Set flags that should be set
011270  012704  000150                  MOV     #'h,R4          ;Terminate string with lower-case H
011274  004737  011350'                 CALL    GENAAS          ;Generate for ANSI attributes
011300  004737  011412'                 CALL    GENDAS          ;Generate for DEC attributes
                                ;
                                ;  See if we should set or reset application keypad mode
                                ;
011304  004737  012260'                 CALL    GENESC          ;Send escape
011310  032762  000000G 000000G        BIT     #AW$AKM,DW$AW(R2);Is application keypad mode wanted?
011316  001405                          BEQ     1$              ;Br if not
011320                                  SEND    #'=             ;Send equal sign
011330  000404                          BR      9$
011332                          1$:     SEND    #'>             ;Send greater-than sign
                                ;
                                ;  Finished
                                ;
011342  012604                  9$:     MOV     (SP)+,R4
011344  012603                          MOV     (SP)+,R3
011346  000207                          RETURN
```

```
                                        .SBTTL  GENAAS -- Generate control sequence for ANSI attributes
                                ;-----------------------------------------------------------------------
                                ;  Generate the control sequence to set or reset ANSI attribute flags.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R3 = Word with flags
                                ;   R4 = Character to terminate string with
                                ;
011350                          GENAAS:
                                ;
                                ;  Don't send anything if this is a VT52
                                ;
011350  032762  000000C 0000000         BIT     #<AW$52!AW$S52>,DW$AW(R2);Is this a VT52?
011356  001014                          BNE     9$                      ;Br if yes
                                ;
                                ;  See if insert mode is wanted
                                ;
011360  032703  0000008                 BIT     #AW$INS,R3              ;Insert mode wanted?
011364  001411                          BEQ     9$                      ;Br if not
011366  004737  012272'                 CALL    GENCSI                  ;Send CSI character
011372                                  SEND    #'4                     ;Send 4
011402                                  SEND    R4                      ;Terminate string
                                ;
                                ;  Finished
                                ;
011410  000207                  9$:     RETURN
```

```
                                        .SBTTL  GENDAS -- Generate control sequence for DEC attributes
                                ;-----------------------------------------------------------------------
                                ;  Generate a terminal control sequence to set or reset DEC attributes.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R3 = AW$xxx attribute flags
                                ;   R4 = Character to terminate string with
                                ;
011412  010146                  GENDAS: MOV     R1,-(SP)
011414  010346                          MOV     R3,-(SP)
011416  010546                          MOV     R5,-(SP)
                                ;
                                ;  We don't have to set any DEC private attributes for VT52 terminals.
                                ;
011420  032762  000000C 000000G         BIT     #<AW$52!AW$S52>,DW$AW(R2);Is this a VT52?
011426  001056                          BNE     9$                      ;Br if VT52
                                ;
                                ;  VT100/VT200
                                ;  If this is a VT100, clear VT200-only attribute flags.
                                ;
011430  032762  0000000 0000000         BIT     #AW$200,DW$AW(R2);  Is this a VT200?
011436  001002                          BNE     5$                      ;Br if VT200
011440  042703  0000000                 BIC     #AW$VCR,R3              ;Clear VT200-only flags
                                ;
                                ;  Init flag that says string hasn't been initiated yet
                                ;
011444  005005                  5$:     CLR     R5                      ;Nothing sent yet
                                ;
                                ;  Begin loop to decide what values need to be sent
                                ;
011446  012701  011574'                 MOV     #DECAT,R1               ;Point to attribute table
                                ;
                                ;  See if next attribute needs to be selected
                                ;
011452  036103  000002          1$:     BIT     2(R1),R3                ;Is this attribute needed?
011456  001431                          BEQ     4$                      ;Br if not
                                ;
                                ;  We need to select this attribute, see if we need to send
                                ;  CSI to start the string or semicolon to separate parameter values
                                ;
011460  005705                          TST     R5                      ;Have we initiated the string?
011462  001010                          BNE     2$                      ;Br if yes
011464  004737  012272'                 CALL    GENCSI                  ;Send CSI character
011470                                  SEND    #'?                     ;Send question mark
011500  005205                          INC     R5                      ;Remember string started
011502  000404                          BR      3$
011504                          2$:     SEND    #SEMI                   ;Send semicolon
                                ;
                                ;  Now send the parameter value to selete the attribute
                                ;
011514  011100                  3$:     MOV     (R1),R0                 ;Get parameter value for attribute
011516  004737  012334'                 CALL    GENVAL                  ;Convert value to digit string and send
                                ;
                                ;  If this is a VT100, terminate the string for each attribute.
                                ;
011522  032762  0000000 0000000         BIT     #AW$200,DW$AW(R2);VT200?
```

```
     58 011530  001004                        BNE     4$              ;Br if yes
     59 011532                                SEND    R4              ;Send terminating character
     60 011540  005005                        CLR     R5              ;Say string is not in progress
     61                                ;
     62                                ;  See if more attributes need to be selected
     63                                ;
     64 011542  062701  000004        4$:     ADD     #4,R1           ;Point to next attribute entry
     65 011546  005711                        TST     (R1)            ;Reached end of table?
     66 011550  001340                        BNE     1$              ;Loop if not
     67                                ;
     68                                ;  If we started the string, terminate it
     69                                ;
     70 011552  005705                        TST     R5              ;Did we start the string?
     71 011554  001403                        BEQ     9$              ;Br if not
     72 011556                                SEND    R4              ;Send terminating character
     73                                ;
     74                                ;  Finished
     75                                ;
     76 011564  012605                9$:     MOV     (SP)+,R5
     77 011566  012603                        MOV     (SP)+,R3
     78 011570  012601                        MOV     (SP)+,R1
     79 011572  000207                        RETURN
     80                                ;
     81                                ;-------------------------------------------------------------------------
     82                                ;  Table of attribute values and DEC attribute flags
     83                                ;
     84 011574  000001  0000000       DECAT:  .WORD   1.,AW$ACK       ;Application mode for cursor keys
     85 011600  000003  0000000               .WORD   3.,AW$132       ;132 column mode
     86 011604  000005  0000000               .WORD   5.,AW$REV       ;Reverse video mode
     87 011610  000010  0000000               .WORD   8.,AW$RPT       ;Automatic keypad repeat
     88 011614  000031  0000000               .WORD   25.,AW$VCR      ;Make cursor visible
     89 011620  000000                        .WORD   0               ;End of table
```

```
                                        .SBTTL  GENLAF -- Generate terminal sequence to set line attrib
                                ;-----------------------------------------------------------------------
                                ;  Generate the terminal control sequence needed to set the line attribute.
                                ;
                                ;  Inputs:
                                ;   R0 = Line attribute flags (AL$xxx)
                                ;
011622  010546                  GENLAF: MOV     R5,-(SP)
011624  010005                          MOV     R0,R5           ;Get attribute flags
                                ;
                                ;  See if double height wanted
                                ;
011626  032705  000000C                 BIT     #<AL$DHT!AL$DHB>,R5;Double high wanted?
011632  001422                          BEQ     1$              ;Br if not
011634  004737  012260'                 CALL    GENESC          ;Send ESC
011640                                  SEND    #'#             ;Send "#"
011650  032705  0000000                 BIT     #AL$DHT,R5      ;Top or bottom half of char?
011654  001405                          BEQ     2$              ;Br if bottom
011656                                  SEND    #'3             ;Send 3
011666  000404                          BR      1$
011670                          2$:     SEND    #'4             ;Send 4
                                ;
                                ;  See if double width is wanted
                                ;
011700  032705  0000000         1$:     BIT     #AL$DWD,R5      ;Is double width wanted?
011704  001412                          BEQ     9$              ;Br if not
011706  004737  012260'                 CALL    GENESC          ;Send ESC
011712                                  SEND    #'#             ;Send "#"
011722                                  SEND    #'6             ;Send 6
                                ;
                                ;  Finished
                                ;
011732  012605                  9$:     MOV     (SP)+,R5
011734  000207                          RETURN
```

```
                                        .SBTTL  GENSPC -- Generate sequence to move cursor over
                                ;-----------------------------------------------------------------
                                ;  Generate spaces or a control sequence to move the cursor right
                                ;  from the current cursor position (specified by DW$COL(R2)) to
                                ;  a specified column.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R1 = Column number we are to move to
                                ;
011736  010346                  GENSPC: MOV     R3,-(SP)
                                ;
                                ;  Determine how many columns we want to move over
                                ;
011740  010103                          MOV     R1,R3           ;Get destination column
011742  166203  0000000                 SUB     DW$COL(R2),R3   ;Subtract current column number
011746  003452                          BLE     9$              ;Br if don't need to move at all
                                ;
                                ;  If we need to move no more than 5 columns, output spaces
                                ;
011750  020327  000005                  CMP     R3,#5           ;Need to move more than 5 columns?
011754  003006                          BGT     2$              ;Br if yes
011756                          1$:     SEND    #40             ;Send a space
011766  077305                          SOB     R3,1$           ;Loop to send more
011770  000441                          BR      9$              ;Finished
                                ;
                                ;  We need to move more than 5 columns; generate control sequence.
                                ;  Determine terminal type.
                                ;
011772  032762  000000C 0000000 2$:     BIT     #<AW$52!AW$S52>,DW$AW(R2);VT52 terminal?
012000  001012                          BNE     5$              ;Br if VT52
                                ;
                                ;  Generate control sequence for VT100
                                ;
012002  004737  012272'                 CALL    GENCSI          ;Generate CSI header
012006  010300                          MOV     R3,R0           ;Get # columns we need to move
012010  004737  012334'                 CALL    GENVAL          ;Convert and output that value
012014                                  SEND    #'C             ;Terminate control sequence
012024  000423                          BR      9$
                                ;
                                ;  Generate control sequence for VT52
                                ;
012026  004737  012260'         5$:     CALL    GENESC          ;Generate escape
012032                                  SEND    #'Y             ;Start of cursor addressing sequence
012042  016200  0000000                 MOV     DW$LIN(R2),R0   ;Get current line number
012046  062700  000037                  ADD     #37,R0          ;Form ascii char for line number
012052                                  SEND    R0              ;Send it
012060  010100                          MOV     R1,R0           ;Get wanted column number
012062  062700  000037                  ADD     #37,R0          ;Form ascii char for column number
012066                                  SEND    R0              ;Send it
                                ;
                                ;  Change current column number
                                ;
012074  010162  0000000         9$:     MOV     R1,DW$COL(R2)   ;Say we have moved up to this column
                                ;
                                ;  Finished
                                ;
```

```
     58 012100  012603                      MOV     (SP)+,R3
     59 012102  000207                      RETURN
```

```
                                        .SBTTL  GENCSR -- Generate cursor addressing sequence
                                ;-----------------------------------------------------------------
                                ;  Generate the appropriate terminal control sequence to move the
                                ;  cursor to the current line and column.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   DW$LIN(R2) = Current line number
                                ;   DW$COL(R2) = Current column number
                                ;
012104                          GENCSR:
                                ;
                                ;  Determine what type of terminal this is
                                ;
012104  032762  000000C 000000G        BIT     #<AW$52!AW$S52>,DW$AW(R2);Is this a VT52?
012112  001035                         BNE     5$                      ;Br if VT52
                                ;
                                ;  Generate sequence for a VT100
                                ;
012114  004737  012272'                CALL    GENCSI                  ;Generate CSI character
012120  016200  000000G                MOV     DW$LIN(R2),R0           ;Get line number
012124  032762  000000G 000000G        BIT     #AW$ORS,DW$AW(R2);Is origin relative to scroll region?
012132  001406                         BEQ     1$                      ;Br if not
012134  166200  000000G                SUB     DW$SRT(R2),R0           ;Subtract top line of scroll region
012140  005200                         INC     R0                      ;1st line is 1
012142  003002                         BGT     1$                      ;Br if we are within scroll region
012144  012700  000001                MOV     #1,R0                   ;Force to 1st line of scroll region
012150  004737  012334'        1$:     CALL    GENVAL                  ;Convert to ascii digit string
012154                                 SEND    #SEMI                   ;Send semicolon
012164  016200  000000G                MOV     DW$COL(R2),R0           ;Get column number
012170  004737  012334'                CALL    GENVAL                  ;Convert to ascii digit string
012174                                 SEND    #'H                     ;Terminate sequence
012204  000424                         BR      9$
                                ;
                                ;  Generate control sequence for VT52
                                ;
012206  004737  012260'        5$:     CALL    GENESC                  ;Generate escape
012212                                 SEND    #'Y                     ;Start of cursor addressing sequence
012222  016200  000000G                MOV     DW$LIN(R2),R0           ;Get current line number
012226  062700  000037                 ADD     #37,R0                  ;Form ascii char for line number
012232                                 SEND    R0                      ;Send it
012240  016200  000000G                MOV     DW$COL(R2),R0           ;Get current column number
012244  062700  000037                 ADD     #37,R0                  ;Form ascii char for column number
012250                                 SEND    R0                      ;Send it
                                ;
                                ;  Finished
                                ;
012256  000207                 9$:     RETURN
```

```
                                        .SBTTL  GENESC -- Generate escape character
                                ;-----------------------------------------------------------------------
                                ;  Send an escape character to the terminal
                                ;
012260                          GENESC: SEND    #33             ;Send escape to the terminal
012270  000207                          RETURN
                                
                                        .SBTTL  GENCSI -- Generate CSI character sequence
                                ;-----------------------------------------------------------------------
                                ;  Send a CSI (ESC [) sequence to the terminal.
                                ;
012272                          GENCSI:
                                ;
                                ;  Send 8-bit CSI character to VT200 terminals
                                ;
                                ;       BIT     #AW$200,DW$AW(R2);Is this a VT200 terminal?
                                ;       BEQ     1$              ;Br if not
                                ;       SEND    #CSICHR         ;Send 8-bit CSI character to VT200
                                ;       BR      9$
                                ;
                                ;  Send ESC [ to VT100 terminals
                                ;
012272                          1$:     SEND    #33             ;Send escape
012302                                  SEND    #'[             ;Send "["
                                ;
                                ;  Finished
                                ;
012312  000207                  9$:     RETURN
                                
                                        .SBTTL  GENCHR -- Send a character to the terminal
                                ;-----------------------------------------------------------------------
                                ;  Send a single character to the terminal
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be sent
                                ;   R2 = Pointer to window control block
                                ;
012314  010146                  GENCHR: MOV     R1,-(SP)
012316  116201  0000000                 MOVB    DW$JOB(R2),R1   ;Get # of job we are sending to
012322                                  OCALL   BUFCHR          ;Send the character
012330  012601                          MOV     (SP)+,R1
012332  000207                          RETURN
```

```
                                        .SBTTL  @ENVAL -- Convert value to digits and send
                                ;-----------------------------------------------------------------
                                ;  Convert a binary value to an ascii digit string and send to the
                                ;  terminal.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be converted and sent
                                ;
012334  010446                  @ENVAL: MOV     R4,-(SP)
012336  010546                          MOV     R5,-(SP)
012340  010005                          MOV     R0,R5           ;Get value to be converted
                                ;
                                ;  Convert to digits and stack the digits
                                ;
012342  005046                          CLR     -(SP)           ;Store null on stack to indicate end
012344  005004                  1$:     CLR     R4              ;Clear high-order for divide
012346  071427  000012                  DIV     #10.,R4         ;Divide R4-R5 by 10.
012352  062705  000060                  ADD     #'0,R5          ;Convert remainder to ascii digit
012356  010546                          MOV     R5,-(SP)        ;Stack the character
012360  010405                          MOV     R4,R5           ;Get quotient
012362  001370                          BNE     1$              ;Loop if more digits to convert
                                ;
                                ;  Now pop the digits and send them
                                ;
012364  012600                  2$:     MOV     (SP)+,R0        ;Get next char to send
012366  001404                          BEQ     9$              ;Br if finished
012370                                  SEND    R0              ;Send a digit
012376  000772                          BR      2$              ;Send rest of number
                                ;
                                ;  Finished
                                ;
012400  012605                  9$:     MOV     (SP)+,R5
012402  012604                          MOV     (SP)+,R4
012404  000207                          RETURN
```

```
                                        .SBTTL  EMTWIN -- Dispatch window control EMT's
                                ;-----------------------------------------------------------------
                                ;  EMTWIN is entered from EMT processing when a window control EMT
                                ;  is executed.  It dispatches control to the correct processing
                                ;  routine based on the subfunction code.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;   EMTBLK = EMT argument block
                                ;
012406                          EMTWIN:
                                ;
                                ;  Ignore window control EMT's if this is a detached job
                                ;
012406  113700  0000000                 MOVB    EMTBLK,R0       ;Get EMT subfunction code
012412  020027  000006                  CMP     R0,#6           ;Read-window-data function?
012416  001407                          BEQ     2$              ;Br if yes -- OK for detached jobs
012420  032761  0000000 0000000         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
012426  001403                          BEQ     2$              ;Br if not
012430  005000                          CLR     R0              ;Return error 0 for detached job
012432  000137  0000000                 JMP     SETERR
                                ;
                                ;  Get EMT subfunction code and make sure it is valid
                                ;
012436  006300                  2$:     ASL     R0              ;Convert subfun code to word table index
012440  020027  000020                  CMP     R0,#MAXSFC      ;Is the subfunction code valid?
012444  103403                          BLO     1$              ;Br if yes
012446  005000                          CLR     R0              ;Return error code 0 if not
012450  000137  0000000                 JMP     SETERR
                                ;
                                ;  Enter processing routine
                                ;
012454  000170  012460'         1$:     JMP     @WINVEC(R0)     ;Enter processing routine
                                ;
                                ;  Branch vector for subfunction codes
                                ;
012460  012500'                 WINVEC: .WORD   WFNEW            ;0 - Create a new window
012462  013170'                         .WORD   WFMAP            ;1 - Select a window
012464  013236'                         .WORD   WFDEL            ;2 - Delete a window
012466  013350'                         .WORD   WFSPND           ;3 - Suspend window processing
012470  013370'                         .WORD   WFRSUM           ;4 - Resume window processing
012472  013410'                         .WORD   WFPRNT           ;5 - Print the specified window
012474  013456'                         .WORD   WFREAD           ;6 - Read window data into program buffer
012476  013740'                         .WORD   WFSTT            ;7 - Set terminal type
        000020                  MAXSFC  =       .-WINVEC         ;Highest branch vector index
```

```
                                        .SBTTL  WFNEW  -- EMT to create a new window
                                ;-----------------------------------------------------------------
                                ;  WINNEW is the EMT used to define a new window.
                                ;  The form of the EMT argument block is:
                                ;
                                ;       .BYTE   0,161
                                ;       .BYTE   window_number,perm_flag
                                ;       .BYTE   window_width,max_scroll
                                ;       .WORD   copy_window,copy_job
                                ;       .WORD   0
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;   EMTBLK = EMT argument block
                                ;
012500                          WFNEW:
                                ;
                                ;  If a window with this number already exists for this job, delete it.
                                ;
012500  113702  0000020                 MOVB    EMTBLK+2,R2     ;Get specified window number
012504  004737  014140'                 CALL    WINSRC          ;Try to locate window control block
012510  103402                          BCS     1$              ;Br if that window not currently defined
012512  004737  014104'                 CALL    WINDEL          ;Delete the window
                                ;
                                ;  Try to find a free window control block
                                ;
012516  013702  000002'         1$:     MOV     DWBAS,R2        ;Point to 1st window control block
012522  105762  0000000         2$:     TSTB    DW$JOB(R2)      ;Is this block free?
012526  001411                          BEQ     5$              ;Br if yes
012530  062702  0000000                 ADD     #DW$$SZ,R2      ;Point to next window block
012534  020237  000004'                 CMP     R2,DWEND        ;Checked all blocks?
012540  103770                          BLO     2$              ;Br if yes
012542  012700  000001                  MOV     #1,R0           ;Return error 0 if no free blocks
012546  000137  0000000                 JMP     SETERR
                                ;
                                ;  Initialize the window control block
                                ;
012552  012700  0000000         5$:     MOV     #DW$$SZ/2,R0    ;Get # words in window block
012556  010203                          MOV     R2,R3           ;Get pointer to control block
012560  005023                  3$:     CLR     (R3)+           ;Zero the entire block
012562  077002                          SOB     R0,3$
012564  110162  0000000                 MOVB    R1,DW$JOB(R2)   ;Set our job number
012570  113762  0000020 0000000         MOVB    EMTBLK+2,DW$ID(R2);Set window ID number
012576  012762  000000C 0000000         MOV     #<AW$VCR!AW$RPT>,DW$AW(R2);Initialize window attribute flags
012604  113700  0000040                 MOVB    EMTBLK+4,R0     ;Get # columns per line
012610  042700  177400                  BIC     #^C<377>,R0     ;Kill sign extension
012614  001002                          BNE     11$             ;Br if width not zero
012616  012700  000120                  MOV     #80.,R0         ;Set width to 80
012622  020027  000204          11$:    CMP     R0,#132.        ;Compare with max width allowed
012626  101402                          BLOS    12$             ;Br if ok
012630  012700  000204                  MOV     #132.,R0        ;Constrain to 132
012634  010062  0000000         12$:    MOV     R0,DW$CPL(R2)   ;Set number of columns per line
012640  012762  000030  0000000         MOV     #24.,DW$LPP(R2) ;Set number of lines per page
012646  113762  0000050 0000000         MOVB    EMTBLK+5,DW$MSL(R2);Set max-scrolled-lines parameter
012654  012762  000001  0000000         MOV     #1,DW$TLN(R2)   ;Say line 1 is at top of buffer
012662  012762  000001  0000000         MOV     #1,DW$SRT(R2)   ;Say line 1 is top of scrolling region
012670  016262  0000000 0000000         MOV     DW$LPP(R2),DW$SRB(R2);scrolling region goes to end of page
```

```
                                        ;
                                        ;  Initialize terminal type
                                        ;
012676  004737  013774'                 CALL    WINSTT          ;Set correct terminal type
                                        ;
                                        ;  Initialize character set mapping for a VT200
                                        ;
012702  032761  000000C 000000@         BIT     #<VT2007!VT2008>,LTRMTP(R1);Is terminal a VT200?
012710  001411                          BEQ     7$              ;Br if not
012712  112762  000001  000000@         MOVB    #1,DW$G2M(R2)   ;Map G2 to DEC supplemental chars
012720  112762  000001  000000@         MOVB    #1,DW$G3M(R2)   ;Map G3 to DEC supplemental chars
012726  112762  000002  000000@         MOVB    #2,DW$GRM(R2)   ;Map GR to G2
                                        ;
                                        ;  Set application keypad mode if Single line editor is in KED mode
                                        ;
012734  032761  000000@ 000000@ 7$:     BIT     #$SLON,LSW7(R1) ;Is single-line editor turned on?
012742  001407                          BEQ     10$             ;Br if not
012744  032761  000000@ 000000@         BIT     #$SLKED,LSW7(R1);Is SL in KED mode?
012752  001403                          BEQ     10$             ;Br if not
012754  052762  000000@ 000000@         BIS     #AW$AKM,DW$AW(R2);Set application keypad mode
                                        ;
                                        ;  See if this window should be permanent
                                        ;
012762  105737  000003@         10$:    TSTB    EMTBLK+3        ;Is permanent-window flag set?
012766  001403                          BEQ     8$              ;Br if not
012770  052762  000000@ 000000@         BIS     #AW$PRM,DW$AW(R2);Remember window is permanent
                                        ;
                                        ;  See if we should copy any window attributes from an existing window
                                        ;
012776  005737  000006@         8$:     TST     EMTBLK+6        ;Should we copy attributes from another windo?
013002  001443                          BEQ     13$             ;Br if not
013004  010203                          MOV     R2,R3           ;Save addr of new window control block
013006  013702  000006@                 MOV     EMTBLK+6,R2     ;Get window_id and job number
013012  010201                          MOV     R2,R1
013014  042702  177400                  BIC     #^C<377>,R2     ;Clear all but window ID
013020  000301                          SWAB    R1              ;Get job # to low-order byte
013022  042701  177400                  BIC     #^C<377>,R1     ;Clear all but job number
013026  006301                          ASL     R1              ;Get job index number
013030  001002                          BNE     14$             ;Br if job number specified
013032  113701  000000@                 MOVB    CORUSR,R1       ;Assume current job
013036  004737  014140'         14$:    CALL    WINSRC          ;Try to find existing window control block
013042  103422                          BCS     15$             ;Br if could not find one
013044  016263  000000@ 000000@         MOV     DW$LPP(R2),DW$LPP(R3) ;Copy # lines per page
013052  016263  000000@ 000000@         MOV     DW$CPL(R2),DW$CPL(R3) ;Copy # columns per line
013060  116263  000000@ 000000@         MOVB    DW$MSL(R2),DW$MSL(R3) ;Copy max # scroll lines
013066  016200  000000@                 MOV     DW$AW(R2),R0    ;Get window attribute flags
013072  042700  000000C                 BIC     #^C<AW$132!AW$REV>,R0 ;Leave only ones to be copied
013076  042763  000000C 000000@         BIC     #<AW$132!AW$REV>,DW$AW(R3) ;Clear in new control block
013104  050063  000000@                 BIS     R0,DW$AW(R3)    ;Transfer the flags
013110  010302                  15$:    MOV     R3,R2           ;Get back address of new window control block
                                        ;
                                        ;  Create a named local region for the screen buffer for the window
                                        ;
013112  004737  014204'         13$:    CALL    MAKWSB          ;Make a window screen buffer
013116  103006                          BCC     4$              ;Br if we made the region
013120  004737  014104'                 CALL    WINDEL          ;Delete the window
013124  012700  000002                  MOV     #2,R0           ;Return error code 2
```

```
    115 013130  000137  000000G                 JMP     SETERR
    116                                 ;
    117                                 ;  Clear entire screen contents to blanks
    118                                 ;
    119 013134  004737  006210'         4$:     CALL    ERSPAG          ;Clear all positions to spaces
    120                                 ;
    121                                 ;  Set line 1, column 1 as cursor position
    122                                 ;
    123 013140  012700  000001                  MOV     #1,R0           ;Get value 1
    124 013144  110062  000000G                 MOVB    R0,DW$SLN(R2)   ;Init saved line # to 1
    125 013150  110062  000000G                 MOVB    R0,DW$SCL(R2)   ;Init saved column # to 1
    126 013154  010062  000000G                 MOV     R0,DW$COL(R2)   ;Set column = 1
    127 013160  004737  006702'                 CALL    SETLIN          ;Set line =1
    128                                 ;
    129                                 ;  Finished
    130                                 ;
    131 013164  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  WFMAP  -- EMT to select a window as the current window
                                ;-----------------------------------------------------------------------
                                ;  Select a window as the current window for the job and refresh
                                ;  the display screen from the window contents.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;   EMTBLK = EMT argument block
                                ;
013170                          WFMAP:
                                ;
                                ;  If current window number is zero, switch back to no window
                                ;
013170  013702  0000020                 MOV     EMTBLK+2,R2     ;Get specified window number
013174  001003                          BNE     2$              ;Br if some window specified
013176  005061  0000000                 CLR     LWINDO(R1)      ;Say we are not doing windoing
013202  000413                          BR      9$
                                ;
                                ;  Locate the window control block for the specified window
                                ;
013204  004737  014140'         2$:     CALL    WINSRC          ;Find window control block
013210  103004                          BCC     1$              ;Br if found it
013212  012700  000003                  MOV     #3,R0           ;Error 3 if not
013216  000137  0000000                 JMP     SETERR
                                ;
                                ;  Set this window as the current window for the job
                                ;
013222  010261  0000000         1$:     MOV     R2,LWINDO(R1)   ;Say this is current window for job
                                ;
                                ;  Refresh the display from selected window
                                ;
013226  004737  007064'                 CALL    REFRSH          ;Refresh display from window contents
                                ;
                                ;  Finished
                                ;
013232  000137  0000000         9$:     JMP     EMTXIT
```

```
                                        .SBTTL  WFDEL  -- EMT to delete a window
                                ;-------------------------------------------------------------
                                ;  EMT to delete a window.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
013236                          WFDEL:
                                ;
                                ;  Try to locate control block for the window
                                ;
013236  113702  0000020                 MOVB    EMTBLK+2,R2     ;Get specified window number
013242  001406                          BEQ     3$              ;Br if window number is zero
013244  120227  177777                  CMPB    R2,#-1          ;Is window number -1?
013250  001024                          BNE     2$              ;Br if not
                                ;
                                ;  Delete all windows for job (temp and permanent)
                                ;
013252  004737  014052'                 CALL    WINREL          ;Release all windows for the job
013256  000432                          BR      9$
                                ;
                                ;  Window number zero means delete all temporary windows
                                ;
013260  013702  000002'         3$:     MOV     DWBAS,R2        ;Point to 1st window control block
013264  120162  0000000         5$:     CMPB    R1,DW$JOB(R2)   ;Does this window belong to our job?
013270  001006                          BNE     4$              ;Br if not
013272  032762  0000000 0000000         BIT     #AW$PRM,DW$AW(R2);Is this a perm or temp window?
013300  001002                          BNE     4$              ;Br if permanent window
013302  004737  014104'                 CALL    WINDEL          ;Delete this window
013306  062702  0000000         4$:     ADD     #DW$$SZ,R2      ;Point to next window control block
013312  020237  000004'                 CMP     R2,DWEND        ;Checked all?
013316  103762                          BLO     5$              ;Br if not
013320  000411                          BR      9$
                                ;
                                ;  We are deleting a specific window
                                ;
013322  004737  014140'         2$:     CALL    WINSRC          ;Try to locate window control block
013326  103004                          BCC     1$              ;Br if located the window control block
013330  012700  000003                  MOV     #3,R0           ;Error code 3 if cannot locate window
013334  000137  0000000                 JMP     SETERR
                                ;
                                ;  Delete the window
                                ;
013340  004737  014104'         1$:     CALL    WINDEL          ;Delete the window
                                ;
                                ;  Finished
                                ;
013344  000137  0000000         9$:     JMP     EMTXIT
```

```
                                        .SBTTL  WFSPND -- Suspend window processing
                                ;-------------------------------------------------------------------
                                ;  Suspend window processing.  After executing this emt, window
                                ;  processing is suspended. Characters are passed through to the
                                ;  screen without being processed by the window routine.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
013350  016102  000000G         WFSPND: MOV     LWINDO(R1),R2   ;Get pointer to current window control block
013354  001403                          BEQ     9$              ;Br if job has no window active now
013356  052762  000000G 000000G        BIS     #AW$SPN,DW$AW(R2);Set flag suspending window processing
013364  000137  000000G         9$:     JMP     EMTXIT          ;Finished

                                        .SBTTL  WFRSUM -- Resume window processing
                                ;-------------------------------------------------------------------
                                ;  Resume window processing.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
013370  016102  000000G         WFRSUM: MOV     LWINDO(R1),R2   ;Get pointer to current window control block
013374  001403                          BEQ     9$              ;Br if job has no window active now
013376  042762  000000G 000000G        BIC     #AW$SPN,DW$AW(R2);Say window no longer suspended
013404  000137  000000G         9$:     JMP     EMTXIT          ;Finished
```

```
                                        .SBTTL  WFPRNT -- Cause contents of a window to be printed
                                ;-------------------------------------------------------------------
                                ;  Cause the contents of a window for the current job to be printed.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
                                ;  EMT argument block:
                                ;
                                ;       .BYTE   5,161
                                ;       .BYTE   window_id,0
                                ;
013410                          WFPRNT:
                                ;
                                ;  Locate the window control block for the specified window
                                ;
013410  113702  0000020                 MOVB    EMTBLK+2,R2     ;Get specified window ID
013414  001403                          BEQ     3$              ;Zero is invalid
013416  004737  014140'                 CALL    WINSRC          ;Find window control block
013422  103004                          BCC     1$              ;Br if found it
013424  012700  000003          3$:     MOV     #3,R0           ;Error 3 if invalid window ID
013430  000137  0000000                 JMP     SETERR
                                ;
                                ;  Print this window
                                ;
013434  004737  014710'         1$:     CALL    WINPRT          ;Try to print the window
013440  103004                          BCC     9$              ;Br if successful
013442  012700  000004                  MOV     #4,R0           ;Error 4 if can't print
013446  000137  0000000                 JMP     SETERR
                                ;
                                ;  Finished
                                ;
013452  000137  0000000         9$:     JMP     EMTXIT
```

```
                                        .SBTTL  WFREAD -- Copy window information to program buffer
                                ;-------------------------------------------------------------------
                                ;  Copy window data into buffer in program area.
                                ;
                                ;  EMT argument block:
                                ;
                                ;       .BYTE   6,161
                                ;       .BYTE   window_id,job_number
                                ;       .WORD   buffer_address
                                ;       .WORD   buffer_size
                                ;
013456                          WFREAD:
                                ;
                                ;  Make sure specified buffer address and size is ok
                                ;
013456  013700  0000040                 MOV     EMTBLK+4,R0     ;Get base address of buffer
013462  004737  0000000                 CALL    VALADW          ;Make sure it's ok
013466  063700  0000060                 ADD     EMTBLK+6,R0     ;Add specified size
013472  004737  0000000                 CALL    VALADB          ;Make sure end is ok too
                                ;
                                ;  Get the number of the job whose window we want to read
                                ;
013476  113701  0000030                 MOVB    EMTBLK+3,R1     ;Get specified job number
013502  006301                          ASL     R1              ;Convert to job index number
013504  001002                          BNE     1$              ;Br if job number specified
013506  113701  0000000                 MOVB    CORUSR,R1       ;Get current job index number
013512  120137  0000000        1$:     CMPB    R1,CORUSR       ;Are we reading window for our job?
013516  001410                          BEQ     5$              ;Br if yes
013520  032737  0000000 0000000         BIT     #P2$CXT,PRIVC2  ;Do we have GETCXT privilege?
013526  001004                          BNE     5$              ;Br if yes
013530  012700  000001                  MOV     #1,R0           ;Error 1 if not
013534  000137  0000000                 JMP     SETERR
                                ;
                                ;  Find the specified window control block
                                ;
013540  113702  0000020        5$:     MOVB    EMTBLK+2,R2     ;Get window ID
013544  004737  014140'                 CALL    WINSRC          ;Search for specified window
013550  103004                          BCC     2$              ;Br if found the window control blk
013552  012700  000003                  MOV     #3,R0           ;Invalid window ID
013556  000137  0000000                 JMP     SETERR
                                ;
                                ;  Determine how much data must be copied
                                ;
013562  016203  0000000        2$:     MOV     DW$LPP(R2),R3   ;Get # lines on page
013566  070362  0000000                 MUL     DW$CPL(R2),R3   ;Times # columns per line
013572  006303                          ASL     R3              ;Need two bytes for each char cell
013574  066203  0000000                 ADD     DW$LPP(R2),R3   ;Add space used by line attrib vector
013600  005203                          INC     R3              ;Round up to next word
013602  042703  000001                  BIC     #1,R3
013606  010300                          MOV     R3,R0           ;Get # bytes of window buffer data
013610  062700  000006                  ADD     #3*2,R0         ;Add # bytes for parameter words
013614  020037  0000060                 CMP     R0,EMTBLK+6     ;Is buffer large enough?
013620  101404                          BLOS    3$              ;Br if yes
013622  012700  000004                  MOV     #4,R0           ;Error 4 if buffer too small
013626  000137  0000000                 JMP     SETERR
                                ;
                                ;  Copy some window parameter information before actual window buffer
```

WFREAD -- Copy window information to program buffer

```
      58                                ;
      59 013632  013705  0000040       3$:     MOV     EMTBLK+4,R5     ;Get address of user's buffer
      60 013636  016246  0000000               MOV     DW$TLN(R2),-(SP);Store top line number
      61 013642  106625                         MTPD    (R5)+
      62 013644  016246  0000000               MOV     DW$LPP(R2),-(SP);Store # lines per page
      63 013650  106625                         MTPD    (R5)+
      64 013652  016246  0000000               MOV     DW$CPL(R2),-(SP);Store # columns per line
      65 013656  106625                         MTPD    (R5)+
      66                                ;
      67                                ;  Copy data from window region to user's buffer
      68                                ;
      69 013660  006203                         ASR     R3              ;Get # words to copy
      70 013662  012704  0000000               MOV     #VPAR6,R4       ;Get virt address of window region
      71 013666                         4$:     BUFMAP                  ;;;Map to window region
      72 013710  012400                         MOV     (R4)+,R0        ;;;Get a word from screen region
      73 013712                                 UNMAP                   ;Restore mapping
      74 013726  010046                         MOV     R0,-(SP)        ;Push word we are transferring
      75 013730  106625                         MTPD    (R5)+           ;Store data into user's buffer
      76 013732  077323                         SOB     R3,4$           ;Loop if more data to copy
      77                                ;
      78                                ;  Finished
      79                                ;
      80 013734  000137  0000000               JMP     EMTXIT
```

```
                                        .SBTTL  WFSTT  -- EMT to set terminal type for windowing
                                ;-------------------------------------------------------------------
                                ;  Reset the terminal type for all windows being used by this job.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number.
                                ;
013740                          WFSTT:
                                ;
                                ;  Begin loop to process all windows belonging to this job
                                ;
013740  013702  000002'                 MOV     DWBAS,R2        ;Point to 1st window control block
013744  120162  0000000         1$:     CMPB    R1,DW$JOB(R2)   ;Does this one belong to this job?
013750  001002                          BNE     2$              ;Br if not
013752  004737  013774'                 CALL    WINSTT          ;Reset terminal type for this window
013756  062702  0000000         2$:     ADD     #DW$$SZ,R2      ;Point to next window control block
013762  020237  000004'                 CMP     R2,DWEND        ;Checked all?
013766  103766                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
013770  000137  0000000                 JMP     EMTXIT
```

```
                                        .SBTTL  WINSTT -- Set terminal type for a window
                                ;-----------------------------------------------------------------------
                                ;  WINSTT sets up the terminal type in the window control block
                                ;  attribute word based on the terminal type specified for the job.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
013774  010146                  WINSTT: MOV     R1,-(SP)
013776  116201  0000000                 MOVB    DW$JOB(R2),R1   ;Get job index number
                                ;
                                ;  Determine terminal type based on line type
                                ;
014002  042762  000000C 0000000         BIC     #<AW$52!AW$200>,DW$AW(R2) ;Say not VT52 and not VT200
                                ;
                                ;  See if terminal is a VT52
                                ;
014010  032761  0000000 0000000         BIT     #VT52,LTRMTP(R1);Is this a VT52?
014016  001404                          BEQ     1$              ;Br if not
014020  052762  0000000 0000000         BIS     #AW$52,DW$AW(R2);Set VT52 as terminal type
014026  000407                          BR      9$
                                ;
                                ;  See if terminal is a VT200
                                ;
014030  032761  000000C 0000000 1$:     BIT     #<VT2007!VT2008>,LTRMTP(R1) ;VT200 terminal?
014036  001403                          BEQ     9$              ;Br if not
014040  052762  0000000 0000000         BIS     #AW$200,DW$AW(R2);Set VT200 as terminal type
                                ;
                                ;  Finished
                                ;
014046  012601                  9$:     MOV     (SP)+,R1
014050  000207                          RETURN
```

```
                                        .SBTTL  WINREL -- Release all display windows for a job
                                ;-----------------------------------------------------------------------
                                ;  Release all display windows for the current job.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number for job whose windows are to be released.
                                ;
014052                          WINREL:
                                ;
                                ;  Begin loop to search for window control blocks belonging to this
                                ;  job.
                                ;
014052  013702  000002'                 MOV     DWBAS,R2        ;Point to first window control block
014056  120162  0000000         1$:     CMPB    R1,DW$JOB(R2)   ;Is this one belong to our job?
014062  001002                          BNE     2$              ;Br if not
014064  004737  014104'                 CALL    WINDEL          ;Delete the windoe
014070  062702  0000000         2$:     ADD     #DW$$SZ,R2      ;Point to next window control block
014074  020237  000004'                 CMP     R2,DWEND        ;Checked all?
014100  103766                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
014102  000207                          RETURN
```

```
                                        .SBTTL  WINDEL -- Delete a window
                                ;-----------------------------------------------------------------------
                                ;  Delete a window and release its screen buffer region.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
014104                          WINDEL:
                                ;
                                ;  If this is the current window, reset window info for job
                                ;
014104  116200  0000000                 MOVB    DW$JOB(R2),R0   ;Get # of job that owns this window
014110  020260  0000000                 CMP     R2,LWINDO(R0)   ;Is this current window for the job?
014114  001002                          BNE     1$              ;Br if not
014116  005060  0000000                 CLR     LWINDO(R0)      ;Say job is not doing windowing
                                ;
                                ;  Free the region used for the screen buffer
                                ;
014122  004737  014514'         1$:     CALL    FREWSB          ;Free the window screen buffer
                                ;
                                ;  Free the window control block
                                ;
014126  105062  0000000                 CLRB    DW$ID(R2)       ;Say block is free
014132  105062  0000000                 CLRB    DW$JOB(R2)      ;Say no job using block
                                ;
                                ;  Finished
                                ;
014136  000207                          RETURN
```

```
                                        .SBTTL  WINSRC -- Locate control block for a window
                                ;-----------------------------------------------------------------------
                                ;  Locate a window control block for a specified window for the current
                                ;  job.
                                ;
                                ;  Inputs:
                                ;   R1 = Index number of job that owns the window
                                ;   R2 = Window number
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found window control block
                                ;   C-flag set     ==> Cannot find window control block
                                ;   R2 = Pointer to window control block if found
                                ;
014140                          WINSRC:
                                ;
                                ;  Search for specified window control block
                                ;
014140  010200                          MOV     R2,R0           ;Get window ID
014142  013702  000002'                 MOV     DWBAS,R2        ;Point to first window control block
014146  120062  0000000         1$:     CMPB    R0,DW$ID(R2)    ;Does this block have correct ID?
014152  001003                          BNE     2$              ;Br if not
014154  120162  0000000                 CMPB    R1,DW$JOB(R2)   ;Is this control block for right job?
014160  001407                          BEQ     3$              ;Br if yes
014162  062702  0000000         2$:     ADD     #DW$$SZ,R2      ;Point to next control block
014166  020237  000004'                 CMP     R2,DWEND        ;Checked all?
014172  103765                          BLO     1$              ;Loop if not
                                ;
                                ;  Cannot find specified window control block
                                ;
014174  000261                          SEC                     ;Signal error on return
014176  000401                          BR      9$
                                ;
                                ;  Found the window control block
                                ;
014200  000241                  3$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
014202  000207                  9$:     RETURN
```

```
                                        .SBTTL  MAKWSB -- Create named region for window screen buffer
                                ;-----------------------------------------------------------------------
                                ;  Create a named local region large enough to hold the window
                                ;  screen buffer.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   EMTBLK+2 = Arg block for EMT to create the window
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Successfully created region
                                ;   C-flag set     ==> Unable to create region
                                ;
014204  010146                  MAKWSB: MOV     R1,-(SP)
014206  010346                          MOV     R3,-(SP)
014210  010446                          MOV     R4,-(SP)
014212  010546                          MOV     R5,-(SP)
014214  010605                          MOV     SP,R5           ;Save original stack pointer in R5
                                ;
                                ;  Calculate number of 64-byte blocks needed for region
                                ;
014216  016203  000000                  MOV     DW$LPP(R2),R3   ;Get # lines on page
014222  070362  000000                  MUL     DW$CPL(R2),R3   ;Times number of columns
014226  006303                          ASL     R3              ;Need two bytes per character
014230  066203  000000                  ADD     DW$LPP(R2),R3   ;Need byte vector for line attributes
014234  062703  000077                  ADD     #63.,R3         ;Bound up to 64-byte boundary
014240  072327  177772                  ASH     #-6.,R3         ;Get # 64-byte blocks needed
                                ;
                                ;  Build a region definition block on the stack
                                ;
014244  162706  000012                  SUB     #10.,SP         ;Allocate space for region def block
014250  010604                          MOV     SP,R4           ;Get pointer to start of RDB
014252  004737  014404'                 CALL    MAKRDB          ;Make a region definition block on the stack
014256  010364  000000                  MOV     R3,R.GSIZ(R4)   ;Set # 64-byte units needed
014262  012764  000000C 000000          MOV     #<RS.GBL!RS.CGR>,R.GSTS(R4) ;Set status flags
                                ;
                                ;  Build EMT argument block on stack
                                ;
014270  162706  000004                  SUB     #4.,SP          ;Allocate space for EMT arg block
014274  010601                          MOV     SP,R1           ;Save pointer to EMT arg block
014276  010600                          MOV     SP,R0           ;Get pointer to EMT arg block area
014300  012710  017000                  MOV     #<36*400>,(R0)  ;Set EMT function code
014304  010460  000002                  MOV     R4,2(R0)        ;Set address of region def block
                                ;
                                ;  Try to create the region
                                ;
014310  104375                          EMT     375             ;Try to create the region
014312  103426                          BCS     9$              ;Br if error on region creation
                                ;
                                ;  We successfully created the region.
                                ;  Set up information in the window control block.
                                ;
014314  016403  000000                  MOV     R.GID(R4),R3    ;Get region control block address
014320  016362  000000 000000          MOV     RC$BLK(R3),DW$RID(R2);Save addr of global RCB
014326  016362  000000 000000          MOV     RC$BAS(R3),DW$MAP(R2);Save mapping base for region
                                ;
                                ;  Now detach from the global region
```

```
      58                                ;
      59 014334  005064  0000000                CLR     R.GSTS(R4)          ;Clear all status flags
      60 014340  010100                          MOV     R1,R0               ;Get pointer to EMT arg block
      61 014342  012710  017001                  MOV     #<<36*400>+1>,(R0);Set .ELRG function code
      62 014346  104375                          EMT     375                 ;Detach from the region
      63                                ;
      64                                ;  Now go back and set the exclusive-access flag in the global region
      65                                ;  control block and increment its attachment count by one.
      66                                ;
      67 014350  016203  0000000                 MOV     DW$RID(R2),R3       ;Get pointer to global RCB
      68 014354  052763  0000000 0000000         BIS     #RC$EXC,RC$FLG(R3);Set exclusive-use flag
      69 014362  105263  0000000                 INCB    RC$CNT(R3)          ;Increment attachment count
      70                                ;
      71                                ;  Successful completion
      72                                ;
      73 014366  000241                          CLC                         ;Signal success on return
      74                                ;
      75                                ;  Finished -- Carry flag indicates results
      76                                ;
      77 014370  010506                  9$:     MOV     R5,SP               ;Reset stack pointer
      78 014372  012605                          MOV     (SP)+,R5
      79 014374  012604                          MOV     (SP)+,R4
      80 014376  012603                          MOV     (SP)+,R3
      81 014400  012601                          MOV     (SP)+,R1
      82 014402  000207                          RETURN
```

```
                                        .SBTTL  MAKRDB -- Make a region definition block
                                ;-----------------------------------------------------------------------
                                ;  Initialize a region definition block to access a global region
                                ;  used as a window buffer.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;   R4 = Pointer to area where region definition block is to be built
                                ;
014404  010146                  MAKRDB: MOV     R1,-(SP)
014406  010346                          MOV     R3,-(SP)
                                ;
                                ;  Initially zero some cells in the RDB
                                ;
014410  005064  000000                  CLR     R.GID(R4)       ;Zero addr of region control block
014414  005064  000000                  CLR     R.GSTS(R4)      ;No status flags
014420  005064  000000                  CLR     R.GSIZ(R4)      ;No size specified yet
                                ;
                                ;  Construct name for global region.
                                ;  The first three characters are "WIN".
                                ;  The next two characters are the job number.
                                ;  The last character indicates the window ID.
                                ;
014424  013764  000010' 000000          MOV     R50WIN,R.NAME(R4);Set 1st 3 chars of name to "WIN"
014432  116201  000000                  MOVB    DW$JOB(R2),R1   ;Get job index number
014436  006201                          ASR     R1              ;Convert to job number
014440  005000                          CLR     R0              ;Clear high-order for divide
014442  071027  000012                  DIV     #10.,R0         ;Split job number into two digits
014446  062701  000036                  ADD     #36,R1          ;Convert binary value to rad50 digit
014452  070127  000050                  MUL     #50,R1          ;Shift remainder left 1 rad50 digit position
014456  010103                          MOV     R1,R3           ;Save shifted low-order digit of job #
014460  010001                          MOV     R0,R1           ;Get high-order digit of job #
014462  062701  000036                  ADD     #36,R1          ;Convert binary value to rad50 digit
014466  070127  003100                  MUL     #3100,R1        ;Shift over 2 rad50 digits
014472  060103                          ADD     R1,R3           ;Job # is two high-order digits
014474  116201  000000                  MOVB    DW$ID(R2),R1    ;Get window number
014500  060103                          ADD     R1,R3           ;Put window number as 3rd digit of word
014502  010364  000020                  MOV     R3,R.NAME+2(R4) ;Set 2nd word of name
                                ;
                                ;  Finished
                                ;
014506  012603                          MOV     (SP)+,R3
014510  012601                          MOV     (SP)+,R1
014512  000207                          RETURN
```

```
                                        .SBTTL  FREWSB -- Free memory used by window screen buffer
                                ;-----------------------------------------------------------------------
                                ;  Free the memory region used for a window screen buffer.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to window control block
                                ;
014514  010146                  FREWSB: MOV     R1,-(SP)
014516  010446                          MOV     R4,-(SP)
014520  010546                          MOV     R5,-(SP)
014522  010605                          MOV     SP,R5           ;Save initial stack pointer
                                ;
                                ;  See if this window has an associated region
                                ;
014524  016204  0000000                 MOV     DW$RID(R2),R4   ;Does window have an assoc region?
014530  001436                          BEQ     9$              ;Br if not
                                ;
                                ;  Decrement attachment-count in global region RCB and
                                ;  reset exclusive-use flag.
                                ;
014532  105364  0000000                 DECB    RC$CNT(R4)      ;Say one less job using region
014536  042764  0000000 0000000         BIC     #RC$EXC,RC$FLG(R4);Clear exclusive-use flag
                                ;
                                ;  Build region definition block on the stack
                                ;
014544  162706  000012                  SUB     #10.,SP         ;Reserve room for region def block
014550  010604                          MOV     SP,R4           ;Get addr of base of region def block
014552  004737  014404'                 CALL    MAKRDB          ;Build the region definition block
014556  012764  0000000 0000000         MOV     #RS.GBL,R.GSTS(R4);Set global status flag in RDB
                                ;
                                ;  Build EMT argument block on stack to attach to region
                                ;
014564  162706  000004                  SUB     #4.,SP          ;Reserve room for EMT arg block
014570  010601                          MOV     SP,R1           ;Save pointer to EMT arg block
014572  010600                          MOV     SP,R0           ;Get pointer to arg block area
014574  012710  017000                  MOV     #<<36*400>+0>,(R0);Set function code
014600  010460  000002                  MOV     R4,2(R0)        ;Set address of region def block
014604  104375                          EMT     375             ;Try to attach to the region
014606  103407                          BCS     9$              ;Br if cannot attach to region
                                ;
                                ;  Eliminate the region
                                ;
014610  012764  000000C 0000000         MOV     #<RS.GBL!RS.EGR>,R.GSTS(R4);Set status flags
014616  010100                          MOV     R1,R0           ;Point to EMT argument block
014620  012710  017001                  MOV     #<<36*400>+1>,(R0);Set EMT function code
014624  104375                          EMT     375             ;Elimininate the region
                                ;
                                ;  Finished
                                ;
014626  010506                  9$:     MOV     R5,SP           ;Restore stack pointer
014630  012605                          MOV     (SP)+,R5
014632  012604                          MOV     (SP)+,R4
014634  012601                          MOV     (SP)+,R1
014636  000207                          RETURN
```

```
                                        .SBTTL  WININI -- TSWIN initialization
                                ;-------------------------------------------------------------------
                                ;  WININI is called during system initialization to initialize the
                                ;  display management system.
                                ;
014640  010546                  WININI: MOV     R5,-(SP)
                                ;
                                ;  Allocate space for the window control blocks
                                ;
014642  013705  0000000                 MOV     VMXWIN,R5       ;Get # control blocks wanted
014646  070527  0000000                 MUL     #DW$$SZ,R5      ;Times size of each block
014652  062705  015050'                 ADD     #WINTOP,R5      ;Add base of area
014656  020527  000000C                 CMP     R5,#140000-DW$$SZ ;Would this overflow available area?
014662  101402                        BLOS    1$              ;Br if not
014664  012705  000000C                 MOV     #140000-DW$$SZ,R5;Constrain to available area
014670  010537  000004'          1$:    MOV     R5,DWEND        ;Set pointer to end of control blocks
                                ;
                                ;  Zero the entire display control block area
                                ;
014674  005045                  2$:     CLR     -(R5)           ;Zero entire area
014676  020537  000002'                 CMP     R5,DWBAS        ;Reached start of area?
014702  101374                        BHI     2$              ;Loop if not
                                ;
                                ;  Finished
                                ;
014704  012605                         MOV     (SP)+,R5
014706  000207                         RETURN
```

```
                                        .SBTTL  WINPRT -- Window print-screen function
                                ;-----------------------------------------------------------------------
                                ;  WINPRT is called when a control character is received that means
                                ;  that we should perform a print-screen function for the current
                                ;  window for the job.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to current window control block for job
                                ;
                                ;  Outputs:
                                ;   C-flag set on return if unable to print the window.
                                ;
014710  010146                  WINPRT: MOV     R1,-(SP)
014712  010446                          MOV     R4,-(SP)
014714  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if "WINPRT" detached job is running now
                                ;
014716  012701  0000000                 MOV     #LSTSL,R1       ;Get index to last job
014722  026137  0000000 000010' 1$:      CMP     LPRG1(R1),R50WIN;Is this the WINPRT program?
014730  001004                          BNE     2$              ;Br if not
014732  026137  0000000 000012'          CMP     LPRG2(R1),R50PRT
014740  001404                          BEQ     3$              ;Br if yes
014742  162701  000002          2$:     SUB     #2,R1           ;More jobs to check?
014746  003365                          BGT     1$              ;Loop if yes
014750  000432                          BR      8$              ;WINPRT program is not running
                                ;
                                ;  The WINPRT program is running.
                                ;  Schedule a completion routine to tell it to print a window.
                                ;
014752  016104  0000000         3$:     MOV     LBRKCQ(R1),R4   ;Get address of its completion Q element
014756  001427                          BEQ     8$              ;Br if no completion routine pending
014760  005737  0000000                 TST     NMFREQ          ;Can we get a free queue element?
014764  003424                          BLE     8$              ;Br if not
014766  004737  0000000                 CALL    GETRTQ          ;Get a free completion Q element (R1=addr)
014772  012700  000000C                 MOV     #IOQSIZ/2,R0    ;Get # words in queue element
014776  010105                          MOV     R1,R5           ;Get pointer to new queue element
015000  012425                  4$:     MOV     (R4)+,(R5)+     ;Copy original completion Q element
015002  077002                          SOB     R0,4$
015004  116200  0000000                 MOVB    DW$JOB(R2),R0   ;Get # of job whose window to print
015010  006200                          ASR     R0              ;Convert to job number
015012  110061  0000010                 MOVB    R0,CQ$R1+1(R1)  ;Set job # in high byte of R1
015016  116261  0000000 0000000         MOVB    DW$ID(R2),CQ$R1(R1);Set window ID in low byte of R1
015024  010104                          MOV     R1,R4           ;Get addr of compl Q element for QCOMPL
015026  004737  0000000                 CALL    QCOMPL          ;Queue the completion request for WINPRT
015032  000241                          CLC                     ;Signal success on return
015034  000401                          BR      9$
                                ;
                                ;  Unable to print the window
                                ;
015036  000261                  8$:     SEC                     ;Signal failure on return
                                ;
                                ;  Finished
                                ;
015040  012605                  9$:     MOV     (SP)+,R5
015042  012604                          MOV     (SP)+,R4
015044  012601                          MOV     (SP)+,R1
```

```
     58 015046  000207                          RETURN
     59                                ;
     60                                ;-------------------------------------------------------------------------
     61                                ;  Top of TSWIN code (Window control blocks are allocated from this
     62                                ;  point to the end of the overlay).
     63                                ;
     64 015050                         WINTOP:
     65                                ;
     66          000001                        .END
Errors detected:   0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  8354 Words  ( 33 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:45.32
DK:TSWIN,LP:TSWIN=DK:TSWIN.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

```
$8BIT     1-25     3-30
$CTRLO    1-27    67-24
$DETCH    1-43    83-18
$RFRSH    1-27    66-15    66-25
$SLKED    1-42    84-75
$SLON     1-42    84-73
$V52EM    1-22    35-24    42-21    43-15    74-16    74-18    74-27    74-29
$VNOTT    1-29    51-18    52-18
$WDISP    1-28     1-43    65-18    66-14
AC$BLD    1-38    40-15    41-8     41-9     41-13    68-53    70-31    70-32    70-43
AC$BLK    1-38    40-15    41-8     41-11    41-15    68-53    70-31    70-32    70-57
AC$REV    1-38    40-15    41-8     41-12    41-16    68-53    70-31    70-32    70-64
AC$SET    1-44    71-20    71-22
AC$ULN    1-39    40-15    41-8     41-10    41-14    68-53    70-31    70-32    70-50
ACRCS     7-34     8-27     9-21     9-33     9-56     9-70     9-97     9-100    10-10#
AL$DHB    1-39    39-11    39-20    68-25    78-13
AL$DHT    1-39    39-12    39-19    68-25    78-13    78-17
AL$DWD    1-39    39-27    39-34    68-25    78-25
ASATAT   33-10    33-24    36-42#
AW$132    1-41    36-48    43-24    68-22    77-85    84-104   84-105
AW$200    1-41    42-14    42-37    69-50    71-37    72-24    72-33    77-22    77-57    91-14    91-27
AW$52     1-31     7-19    42-14    69-12    70-24    71-28    72-10    73-13    76-14    77-16    79-30    80-15
          91-14    91-20
AW$ACK    1-41    36-47    43-24    77-84
AW$AKM    1-42    38-8     38-18    43-24    75-33    84-77
AW$DDC    1-45     3-72    51-15    64-14    67-15    67-83
AW$INS    1-41    36-42    43-24    76-19
AW$ORS    1-41    24-17    36-50    43-24    48-15    73-35    80-22
AW$PRM    1-43    84-83    86-27
AW$PRT    1-25     3-54    11-83    43-24    44-17    44-21
AW$REV    1-41    36-49    77-86    84-104   84-105
AW$RPT    1-43    36-51    77-87    84-44
AW$S52    1-31     7-19    35-21    42-14    43-12    69-12    70-24    71-28    72-10    73-13    74-14    76-14
          77-16    79-30    80-15
AW$SPN    1-27     3-25    87-12    87-24
AW$SS     1-44     4-30     4-33    32-79    32-90    43-24
AW$VCR    1-42    36-52    77-24    77-88    84-44
BUFCHR    1-31     1-40    81-40
CATABL   40-26    41-8#
CATEND   40-30    41-17#
CCEXIT    5-26     5-27     5-28     5-29     5-30     5-31     5-32     5-33     5-42     5-43     5-44     5-45
          5-46     5-47     5-48     5-49     5-50     5-51     5-53     5-54     5-55     5-56     5-57     5-61#
CCRTN     5-17     5-26#
CKTAC    34-29    35-37    37-10#
CLRLIN   26-52    27-51    49-46    50-47    56-32    57-23    57-32    58-14    59-10#
CORUSR    1-39    84-97    89-26    89-27
CPYLIN   26-44    27-43    49-39    50-40    60-11#
CQ$R1     1-24    99-42*   99-43*
CR        1-38    68-90
CSBUF     2-54#   11-20    11-56
CSFIN     8-44     9-34     9-71    11-8#
CSICHR    1-31     6-25
CSPARM   13-15#   17-14    18-14    22-13    23-13    24-13    24-31    26-26    27-25    34-14    35-14    40-20
          42-31    44-14    45-13    46-13    47-13    47-21
CSTBLL    11-49   12-14#
CSTBLU    11-52   12-18#
```

Cross reference table (CREF V05.04)

```
CV52      9-54     9-68     9-84#
CVTTAV   34-20    35-30    36-13#
DECAT    77-32    77-34#
DECTAT   33-38    33-57    36-47#
DOCTRL    3-63     5-10#
DW$$SZ    1-38    84-30    84-38    86-30    90-16    92-17    94-25    98-11    98-13    98-15
DW$AW     1-31     3-20     3-54     3-72     4-30     4-33*    7-19    11-83    24-17    32-79*   32-90*   34-25*
         35-21*   35-30*   38-8*    38-18*   42-14*   42-37*   43-12*   43-24*   44-17*   44-21*   48-15    51-15
         64-14*   67-15*   67-83*   68-22    69-12    69-50    70-24    71-28    71-37    72-10    72-24    72-33
         73-13    73-35    74-14    75-14    75-33    76-14    77-16    77-22    77-57    79-30    80-15    80-22
         84-44*   84-77*   84-83*   84-103   84-105*  84-106*  86-27    87-12*   87-24*   91-14*   91-20*   91-27*
DW$CCA    1-33     4-40    25-17    25-43*   40-15*   40-37*   40-38*   67-72    67-74
DW$COL    1-33     4-19     4-51     4-53*   14-12    14-14*   15-12    15-23*   16-8*    22-24    22-28*   23-25
         23-28*   24-38*   25-13    25-34*   26-58*   27-56*   48-24*   54-15    55-14    56-19    57-30    67-65
         68-16    68-40*   68-60    68-76*   68-88    68-96*   79-16    79-54*   80-30    80-42    84-126*
DW$CPL    1-33     4-51    15-20    15-22    22-25    22-27    24-35    24-37    53-14    54-19    57-30    59-21
         60-38    63-26    68-32    68-34    84-52*   84-101   84-101*  89-45    89-64    95-23
DW$CSB    1-30     8-39    10-22*   11-16*   11-22    11-79
DW$CSP    1-30     7-13*    7-32*   10-15    10-23*   11-15    11-91*
DW$CSR    1-30     3-46     7-22*    7-31*    9-27*    9-57*   11-92*
DW$GOM    1-46     4-35    28-8*    28-18*   28-28*   28-38*   72-40    72-49
DW$G1M    1-46    29-8*    29-18*   29-28*   29-38*
DW$G2M    1-46    30-8*    30-18*   30-28*   43-38*   84-67*
DW$G3M    1-46    31-8*    31-18*   31-28*   43-39*   84-68*
DW$GLM    1-44     4-29    32-8*    32-18*   32-28*   32-38*   72-16
DW$GLS    1-44     4-32    32-78*   32-89*
DW$GRM    1-44     4-26    32-48*   32-58*   32-68*   43-40*   72-26    84-69*
DW$ID     1-35    84-43*   93-23*   94-21    96-36    99-43
DW$JOB    1-35    35-20    42-19    42-25    43-13    51-17    52-17    65-13    67-19    67-67    74-12    81-39
         84-28    84-42*   86-25    90-13    91-10    92-14    93-12    93-24*   94-23    96-25    99-40
DW$LIN    1-33     9-52    17-22    17-25    18-21    18-28    19-13    21-14    25-12    26-18    26-50    27-17
         56-11    56-15    57-15    59-12    60-14    61-13    62-15    63-18*   67-44    68-86    79-45    80-21
         80-39
DW$LPP    1-34    18-22    18-24    19-17    19-19    24-24    24-26    37-25    43-20    47-24    49-22    49-29
         50-22    50-28    56-29    58-12    61-16    62-18    63-24    63-28    67-50    68-86    73-20    84-53*
         84-57    84-100   84-100*  89-44    89-47    89-62    95-22    95-25
DW$LPT    1-34     4-15    53-13    54-18    54-22    55-13    59-20    60-31    60-34    63-30*   68-41
DW$MAP    1-34     4-44    53-15    54-26    55-18    59-22    60-39    61-21    62-23    68-45    89-71    95-55*
DW$MSL    1-45    51-24    51-35    84-54*   84-102   84-102*
DW$NSL    1-45    51-30    51-32*   64-18*
DW$RID    1-35    95-54*   95-67    97-15
DW$SCA    1-47    25-17*   25-43
DW$SCL    1-47    25-13*   25-34    84-125*
DW$SLN    1-47    25-12*   25-38    84-124*
DW$SRB    1-33    18-28    18-30    18-32    19-14    24-21    24-23    26-21    26-30    26-41    27-20    27-29
         27-49    37-25*   43-20*   47-25*   49-22    49-40    49-45    50-22    50-36    73-20    73-29    84-57*
DW$SRT    1-34    17-25    17-27    17-29    21-15    24-19    26-19    27-18    37-24*   43-19*   47-17*   48-17
         49-20    49-35    50-20    50-41    50-46    73-18    73-26    80-24    84-56*
DW$TLN    1-34    49-27*   49-29*   50-27*   50-28    50-30*   61-14    62-16    63-22    84-55*   89-60
DWBAS     2-49#   84-27    86-24    90-12    92-13    94-20    98-21
DWEND     2-50#   84-31    86-31    90-17    92-18    94-26    98-16*
EMTBLK    1-37    63-15    84-20    84-43    84-45    84-54    84-81    84-87    84-90    85-14    86-12    88-17
         89-16    89-18    89-23    89-36    89-52    89-59
EMTWIN    1-17    83-11#
EMTXIT    1-37    84-131   85-36    86-48    87-13    87-25    88-33    89-80    90-22
EP52      7-21     9-11#
```

Cross reference table (CREF V05.04)

```
EP52C     9-57      9-68#
EP52L     9-27      9-49#
EPCH1     7-18      7-31      8-21#
ERSCTL   45-16     54-10#    56-24
ERSCTP   46-16     56-10#
ERSLIN   45-24     53-9#
ERSLTC   45-20     55-9#     57-35
ERSPAG   37-27     43-32     46-24     58-8#     84-119
ERSPTC   46-20     57-10#
ESC       1-31      6-18
FORCEX    1-29     65-24
FREWSB   93-19     97-8#
FRKPRI    1-29     52-12
GENAAS   75-20     75-27     76-10#
GENCAF   67-75     68-70     70-13#
GENCHR   68-75     68-90     68-91     69-19     69-24     69-25     69-26     69-31     69-33     69-38     69-39     69-43
         69-45     69-46     69-53     69-55     69-56     69-62     69-64     69-66     70-39     70-45     70-46     70-52
         70-53     70-59     70-60     70-66     70-67     70-71     71-35     71-36     71-40     71-41     71-48     72-20
         72-28     72-37     72-41     72-50     73-28     73-31     73-38     73-39     73-40     74-20     74-21     74-22
         74-31     75-35     75-37     76-22     76-23     77-45     77-48     77-59     77-72     78-16     78-19     78-21
         78-28     78-29     79-23     79-38     79-44     79-47     79-50     80-29     80-32     80-38     80-41     80-44
         81-5      81-23     81-24     81-38#    82-27
GENCLR   67-32     69-8#
GENCSC   70-20     71-11#
GENCSI   69-18     69-23     69-30     69-32     69-37     70-38     73-25     73-37     74-19     76-21     77-44     79-35
         80-20     81-12#
GENCSR   67-64     80-11#
GENDAS   75-21     75-28     77-10#
GENESC   69-44     69-52     69-54     69-61     69-63     69-65     71-34     71-39     71-47     72-19     72-27     72-36
         72-48     74-30     75-32     78-15     78-27     79-43     80-37     81-5#
GENLAF   68-28     78-8#
GENMAP   67-79     72-9#
GENSPC   68-63     79-11#
GENSSS   67-60     73-9#
GENTEM   67-28     74-12#
GENVAL   73-27     73-30     77-53     79-37     80-28     80-31     82-9#
GENWAF   67-36     75-9#
GETLAB   39-10     39-18     39-26     39-33     60-20     62-11#    68-20
GETRTQ    1-23     99-35
GNMCHR   72-37     72-67#
INTPRI    1-32      4-47     53-18     54-29     55-21     59-25     60-42     61-23     62-25     67-23     68-48     89-73
IOQSIZ    1-24     99-36
KPAR6     1-32      4-44      4-44*     4-47*     53-15     53-15*    53-18*    54-26     54-26*    54-29*    55-18     55-18*
         55-21*    59-22     59-22*    59-25*    60-39     60-39*    60-42*    61-21     61-21*    61-23*    62-23     62-23*
         62-25*    68-45     68-45*    68-48*    89-71     89-71*    89-73*
LBRKCQ    1-23     99-31
LCOL      1-42     67-68*
LF        1-38     68-91
LNPRIM    1-22     35-23     42-20     43-14     74-13
LOTNXT    1-26     67-22
LOTPNT    1-26     67-22*
LOTSIZ    1-26     67-21
LOTSPC    1-26     67-21*
LPRG1     1-23     99-20
LPRG2     1-23     99-22
LSLCHR   72-20     72-59#
```

Cross reference table (CREF V05.04)

```
LSRCHR   72-28    72-63#
LSTATE    1-28    65-22
LSTSL     1-23    99-19
LSW       1-29     1-43    51-18    52-18    83-18
LSW11     1-22    35-24*   42-21*   43-15*   74-16    74-18*   74-27    74-29*
LSW2      1-25     3-30
LSW3      1-27    67-24*
LSW4      1-27    66-15*   66-25*
LSW6      1-28     1-43    65-18*   66-14*
LSW7      1-42    84-73    84-75
LTRMTP    1-40    42-26    43-36    84-65    91-18    91-25
LWINDO    1-30     3-21    66-19    85-16*   85-28*   87-10    87-22    93-13    93-15*
MAKRDB   95-33    96-10#   97-28
MAKWSB   84-111   95-14#
MAPHLD    2-51#    4-44*    4-47    53-15*   53-18    54-26*   54-29    55-18*   55-21    59-22*   59-25    60-39*
         60-42    61-21*   61-23    62-23*   62-25    68-45*   68-48    89-71*   89-73
MAXSFC   83-26    83-45#
NMFREQ    1-23    99-33
OVRHC     1-32    52-23    81-40
P2$CXT    1-25    89-29
PCSPND    1-29    52-23
PR7       1-32     4-44    53-15    54-26    55-18    59-22    60-39    61-21    62-23    67-20    68-45    89-71
PRIVC2    1-25    89-29
PSW       1-32     4-44*    4-47*   53-15*   53-18*   54-26*   54-29*   55-18*   55-21*   59-22*   59-25*   60-39*
         60-42*   61-21*   61-23*   62-23*   62-25*   67-20*   67-23*   68-45*   68-48*   89-71*   89-73*
QCOMPL    1-24    99-45
R.GID     1-35    95-53    96-15*
R.GSIZ    1-36    95-34*   96-17*
R.GSTS    1-36    95-35*   95-59*   96-16*   97-29*   97-43*
R.NAME    1-36    96-24*   96-38*
R5OPRT    2-53#   99-22
R5OWIN    2-52#   96-24    99-20
RC$BAS    1-35    95-55
RC$BLK    1-28    95-54
RC$CNT    1-28    95-69*   97-21*
RC$EXC    1-26    95-60    97-22
RC$FLG    1-26    95-60*   97-22*
REFRSH   66-21    67-9#    85-32
REGCHR    3-68     4-10#    5-52
RS.CGR    1-36    95-36
RS.EGR    1-37    97-43
RS.GBL    1-37    95-35    97-29    97-43
RS.PVT    1-36
S$OTWT    1-28    65-22
SCRCHK   49-15    50-15    51-11#
SCRLDN   21-26    50-8#
SCRLUP   19-26    49-8#
SEMI      2-47#    9-50    13-22    70-45    70-52    70-59    70-66    73-28    77-48    80-29
SET200   71-36    71-41    71-60#   72-41
SET52    71-48    71-64#   72-50
SETERR    1-37    83-21    83-29    84-34    84-115   85-24    86-40    88-22    88-29    89-32    89-40    89-55
SETHOM   37-16    37-26    43-28    47-29    48-10#
SETLAB   39-13    39-21    39-28    39-35    59-31    60-25    61-9#
SETLIN   17-30    18-33    19-20    21-20    24-27    25-39    48-20    56-39    57-34    57-40    59-16    59-36
         60-19    60-23    60-30    60-33    60-48    63-14#   67-47    67-56    84-127
SNDLIN   67-48    68-13#
```

```
SS3CHR    1-31
TC100    12-16     12-84     42-9#
TCAKM    12-70     38-8#
TCBS      5-34     14-8#
TCCADR   12-15     12-43     24-9#
TCCPRS   12-69     25-30#
TCCPSV   12-68     25-8#
TCCR      5-39     16-8#     20-9
TCDELL   12-48     27-10#
TCDOWN   12-38     18-10#
TCESCE   12-50     20-9#
TCGOAS   12-42     12-65     28-8#
TCGODS   12-41     12-73     28-18#
TCGOGR   12-56     28-38#
TCGOUK   12-54     28-28#
TCG1AS   12-57     12-68     29-8#
TCG1DS   12-76     29-18#
TCG1GR   12-59     29-38#
TCG1UK   29-28#
TCG2AS   12-60     30-8#
TCG2DS   12-79     30-18#
TCG2GR   12-61     30-28#
TCG3AS   12-62     31-8#
TCG3DS   12-82     31-18#
TCG3GR   12-63     31-28#
TCHT      5-35     15-8#
TCINSL   12-47     26-11#
TCIXUP   12-44     12-51     21-9#
TCLEFT   12-40     23-9#
TCLERS   12-46     45-9#
TCLF      5-36      5-37      5-38     12-49     19-8#     20-10
TCLS1R   12-26     32-48#
TCLS2    12-27     32-28#
TCLS2R   12-29     32-58#
TCLS3    12-28     32-38#
TCLS3R   12-30     32-68#
TCN3     12-64     39-10#
TCN4     12-65     39-18#
TCN5     12-66     39-26#
TCN6     12-67     39-34#
TCNKM    12-86     38-18#
TCPERS   12-45     46-9#
TCPRT    11-85     12-21     44-9#
TCRAA    12-18     33-23#
TCRDA    12-20     33-51#
TCREST   12-24     12-25     43-8#
TCRIT    12-39     22-9#
TCRTA    33-25     33-53     35-10#
TCSAA    12-17     33-9#
TCSBSZ    1-30     10-16
TCSCA    12-22     40-9#
TCSDA    12-19     33-37#
TCSI      5-41     32-18#
TCSO      5-40     32-8#
TCSS2    12-52     32-78#
TCSS3    12-53     32-89#
```

Cross reference table (CREF V05.04)

```
TCSSR    12-23     47-9#
TCSTA    33-11     33-39     34-10#
TCUP     12-37     17-10#
TSWIN     1-6#
VALADB    1-24     89-19
VALADW    1-24     89-17
VMXWIN    1-40     98-10
VPAR6     1-32     61-17     62-19     63-29     89-70
VT2007    1-40     42-26     43-36     84-65     91-25
VT2008    1-40     42-26     43-36     84-65     91-25
VT52      1-40     91-18
WCPCSI    6-27      7-31#
WCPESC    6-20      7-9#
WCSCHK    3-41      6-14#
WFDEL    83-39     86-8#
WFMAP    83-38     85-10#
WFNEW    83-37     84-16#
WFPRNT   83-42     88-13#
WFREAD   83-43     89-12#
WFRSUM   83-41     87-22#
WFSPND   83-40     87-10#
WFSTT    83-44     90-8#
WINCHR    1-17      3-14#
WINDEL   84-23     84-113    86-29     86-44     92-16     93-8#
WINDSP    1-17     66-9#
WININI    1-17     98-6#
WINPRT    1-18     88-26     99-13#
WINREL    1-17     86-19     92-8#
WINSF     1-18     64-9#
WINSPN   51-40     52-8#
WINSRC   84-21     84-98     85-21     86-37     88-19     89-37     94-15#
WINST     1-18     65-9#
WINSTT   84-61     90-15     91-9#
WINTOP    1-47      2-49     98-12     99-64#
WINVEC   83-33     83-37#    83-45
```

Cross reference table (CREF V05.04)