Table of contents

```
                        .TITLE  TSTIOX -- Relocatable terminal service code
000000                  .PSECT  TSTIOX
                        .ENABL  LC
                        .DSABL  GBL
                ;
                ;  TSTIOX contains the portion of the terminal service code that
                ;  is linked as an overlay and relocated over TSINIT when the
                ;  system is started.
                ;
                ;  Copyright (c) 1984, 1985.
                ;  S&H Computer Systems, Inc.
                ;  Nashville, Tennessee  USA
                ;
                ;  Global definitions
                ;
                        .GLOBL  TSTIOX
                ;
                ;  Global references
                ;
                        .GLOBL  OVRHC, CL$LIX, LCXTBL
                        .GLOBL  NEDCDO, NEDCDI, LCLUNT, LINIR, LCDTYP, CTRLS, LOTSIZ, LOTBUF
                        .GLOBL  CTRLQ, CDX$VH, LMSGBF, LOTPNT, CDSXOF, CDSXON, CLTIMR, LXCL
                        .GLOBL  LNMAP, OTRASZ, CDOFLG, LHIRBS, LHIRBE, LHIRBB, INTPRI, LHIRBP
                        .GLOBL  INITFL, LHIRBC, LHIRBG, LHIRBA, $DETCH, CO$8BT, $HARD, $CTRLS
                        .GLOBL  CDX$DH, FRMERR, $INIT, $SXON, LSW, LSW7, S$OTWT
                        .GLOBL  $IITIM, $SXOFF, LSW5, $8BIT, LITIME, S$$RUN, $DHCDO, LSW3
                        .GLOBL  LOTEND, SB$LNK, $SOTFN, NMUMB, $PAGE, DHTIMR, S$OTLO, LSTHL
                        .GLOBL  S$OTFN, $NDICP, INTEN, FP$CDI, MBFFLG, FORK, $RBRK, CDIFLG
                        .GLOBL  LSW2, SNMSHD, CL$OPT, NEDSOT, $ICPFK, $DEAD, NEDCLO, SB$PNT
                        .GLOBL  LNPRIM, LSW10, CDSTRT, $HISTP, LSTATE, FASTIN, CDSTOP
                        .GLOBL  CO$BNO, CO$BNI, LOTSPC, $DNICP, PSW, KPAR6, LTTPAR
                        .GLOBL  LINNUM, RDINT, RSR, RCVDON, RBR, MUXNUM, MXRBUF, MXLNT
```

```
                        ;-----------------------------------------------------------------------
                        ;  Macro definitions:
                        ;
                        ;  Disable interrupts
                        ;
                              .MACRO  DISABL          ;Disable interrupts
                              BIS     #340,@#PSW
                              .ENDM   DISABL
                        ;
                        ;  Enable interrupts
                        ;
                              .MACRO  ENABL           ;Enable interrupts
                              BIC     @#INTPRI,@#PSW
                              .ENDM   ENABL
                        ;
                        ;  Call a routine in a system overlay
                        ;
                              .MACRO  OCALL   ENTADD
                              .IF     B,ENTADD
                                      .ERROR  ;OCALL without entry address
                              .ENDC
                              CALL    @#OVRHC
                              .WORD   ENTADD
                              .ENDM   OCALL
                        ;
                        ;  The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
                        ;  terminal character buffer area.  The previous contents of par6 map
                        ;  register are pushed on the stack and may be restored by using the
                        ;  UNMAP or UNMAPX macros.
                        ;  R1 must contain the line index number of the line whose buffers
                        ;  are being accessed.
                        ;  The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
                        ;  macros is that the X-versions are more efficient but may only be
                        ;  used from within interrupt service routines where we are guaranteed
                        ;  to be running on the system stack.
                        ;  The TTMAP and UNMAP versions of the macros must only be
                        ;  used in sections of code where the interrupts are disabled.
                        ;
                              .MACRO  TTMAPX
                              MOV     LTTPAR(R1),@#KPAR6
                              .ENDM   TTMAPX

                        ;
                        ;
                              .MACRO  UNMAPX
                              .ENDM   UNMAPX
                        ;
                              .MACRO  TTMAP
                              MOV     @#KPAR6,MAPHLD
                              MOV     LTTPAR(R1),@#KPAR6
                              .ENDM   TTMAP
                        ;
                              .MACRO  UNMAP
                              MOV     MAPHLD,@#KPAR6
                              .ENDM   UNMAP
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Module header.  This data must be the first data in this segment
                                ;
                                ;  Overlay segment ID word
                                ;
000000  077167                  TSTIOX: .RAD50  /TIO/
                                ;
                                ;  Table of offsets to entry points in this module
                                ;
000002  000030                          .WORD   ININT-TSTIOX
000004  000172                          .WORD   TTINPT-TSTIOX
000006  000602                          .WORD   SILFET-TSTIOX
000010  000752                          .WORD   TTRSAV-TSTIOX
000012  001270                          .WORD   TRNSTR-TSTIOX
000014  001346                          .WORD   NEDCHR-TSTIOX
000016  001776                          .WORD   CDORTN-TSTIOX
000020  002106                          .WORD   CDIRTN-TSTIOX
000022  001752                          .WORD   SNDFRE-TSTIOX
                                ;
                                ;  Data areas
                                ;
000024  000000                  MAPHLD: .WORD   0               ;Used by TTMAP macro
                                ;
000026      000                 TTIFRK: .BYTE   0               ;Non-zero ==> Request fork level input process
                                        .EVEN
```

```
                        .SBTTL  ININT  -- Terminal input interrupt entry point
                ;-----------------------------------------------------------------------
                ;
                ;  Receiver interrupt routine - enter with interrupts disabled
                ;
000030          ININT:
                ;
                ;  Save registers R0, R1, R4, and R5.
                ;
000030  004567  000716          JSR     R5,TTRSAV       ;Save registers
                ;
                ;  R0, R1, R4 and R5 are now available.
                ;  Determine which line interrupted.
                ;
000034  113704  000000G         MOVB    @#LINNUM,R4     ;Get interrupting line number
000040  001426                  BEQ     1$              ;Branch if mux line interrupted
                ;
                ;  Interrupt came from a DL11 line
                ;
000042  105037  000000G         CLRB    @#LINNUM        ;Reset line number cell
000046  006304                  ASL     R4              ;Get TSX line number index
000050  042774  000000G 000000G BIC     #RDINT,@RSR(R4) ;Drop DL11 interrupt enable
000056  032774  000000G 000000G BIT     #RCVDON,@RSR(R4);Did we receive a character?
000064  001410                  BEQ     3$              ;Branch if not
000066  017405  000000G         MOV     @RBR(R4),R5     ;Get receiver buffer register info
000072  052774  000000G 000000G BIS     #RDINT,@RSR(R4) ;Reenable DL11 interrupt
000100  004767  000066          CALL    TTINPT          ;Process the character
000104  000431                  BR      9$              ;Finished
000106  052774  000000G 000000G 3$:     BIS     #RDINT,@RSR(R4) ;Reenable DL11 interrupts
000114  000425                  BR      9$
                ;
                ;  Interrupt came from a DZ11 or DH11 line
                ;
000116  010346                  1$:     MOV     R3,-(SP)        ;Get another work register for mux
000120  113703  000000G         MOVB    @#MUXNUM,R3     ;Get MUX number
000124  105037  000000G         CLRB    @#MUXNUM        ;Reset cell
000130  006303                  ASL     R3              ;Convert to word table index
000132  017305  000000G         2$:     MOV     @MXRBUF(R3),R5  ;Get MUX receiver buffer register
000136  002013                  BGE     4$              ;Br if mux silo is empty
000140  010504                  MOV     R5,R4           ;Get receiver buffer info
000142  042704  170377          BIC     #^C<7400>,R4    ;Clear all but line #
000146  000304                  SWAB    R4              ;Move to low order byte
000150  066304  000000G         ADD     MXLNT(R3),R4    ;Get pointer into map table
000154  111404                  MOVB    (R4),R4         ;Get TSX line #
000156  001765                  BEQ     2$              ;Br if line not genned into TSX
000160  004767  000006          CALL    TTINPT          ;Process the received character
000164  000762                  BR      2$              ;Go see if more chars in mux silo
000166  012603                  4$:     MOV     (SP)+,R3
                ;
                ;  Finished
                ;
000170  000207                  9$:     RETURN                  ;Return from interrupt
```

```
                        .SBTTL  TTINPT -- Process received character
                ;-----------------------------------------------------------------------
                ;  TTINPT is called each time a character is received from a serial line.
                ;  This routine stores the character into the initial ring buffer for the line
                ;  and then requests a .FORK to trigger the actual input character processing.
                ;
                ;  Inputs:
                ;   R4 = Index number of interrupting line.
                ;   R5 = Received character and status flags in DL11 format.
                ;
000172  010046  TTINPT: MOV     R0,-(SP)
000174  010346         MOV     R3,-(SP)
                ;
                ;  Reset input interrupt timer for the line
                ;
000176  042764  0000000 0000000        BIC     #$IITIM,LSW5(R4);Reset input interrupt timer for line
                ;
                ;  See if a framing error was detected
                ;
000204  032705  0000000        BIT     #FRMERR,R5      ;Was a framing error detected?
000210  001404         BEQ     7$              ;Br if not
000212  052764  0000000 0000000        BIS     #$RBRK,LSW10(R4);Set flag saying break was detected
000220  000456         BR      2$              ;Skip over special character tests
                ;
                ;  See if we should mask character to 7 bits
                ;
000222  042705  177400  7$:     BIC     #^C377,R5       ;Mask character to 8 bits
000226  016403  0000000        MOV     LCLUNT(R4),R3   ;Is this line being used by CL?
000232  002411         BLT     6$              ;Br if not
000234  032763  0000000 0000000        BIT     #CO$BNI,CL$OPT(R3);Binary input mode?
000242  001045         BNE     2$              ;Br if yes -- Accept all characters
000244  032763  0000000 0000000        BIT     #CO$8BT,CL$OPT(R3);Is CL unit set for 8 bit support?
000252  001007         BNE     3$              ;Br if 8 bit mode
000254  000404         BR      4$
000256  032764  0000000 0000000 6$:     BIT     #$8BIT,LSW2(R4) ;Is line set to receive 8-bit chars?
000264  001002         BNE     3$              ;Br if yes
000266  042705  177600  4$:     BIC     #^C177,R5       ;Mask out all but 7 bits of character
                ;
                ;  Ignore nulls unless this is a CL line in binary input mode
                ;
000272  105705  3$:     TSTB    R5              ;Is the character null?
000274  001466         BEQ     9$              ;Ignore nulls unless in binary input mode
                ;
                ;  See if character is ctrl-S (XOFF)
                ;
000276  120527  0000000        CMPB    R5,#CTRLS       ;Is character ctrl-S?
000302  001011         BNE     1$              ;Br if not
000304  005703         TST     R3              ;Is line being used by CL?
000306  002004         BGE     10$             ;Br if yes
000310  032764  0000000 0000000        BIT     #$PAGE,LSW2(R4) ;Is ctrl-S processing wanted for line?
000316  001417         BEQ     2$              ;Br if not -- treat ctrl-S like normal char
000320  004767  000670  10$:     CALL    GOTXOF          ;Process the ctrl-S character
000324  000452         BR      9$              ;Finished with character
                ;
                ;  See if character is ctrl-Q (XON)
                ;
000326  120527  0000000 1$:     CMPB    R5,#CTRLQ       ;Is character ctrl-Q?
```

```
     58 000332  001011                          BNE     2$              ;Br if not
     59 000334  005703                          TST     R3              ;Is line being used by CL?
     60 000336  002004                          BGE     11$             ;Br if yes
     61 000340  032764  000000G 000000G         BIT     #$PAGE,LSW2(R4) ;Is ctrl-Q processing wanted for line?
     62 000346  001403                          BEQ     2$              ;Br if not -- treat ctrl-Q like normal char
     63 000350  004767  000666          11$:    CALL    GOTXON          ;Process the ctrl-Q character
     64 000354  000436                          BR      9$
     65                                 ;
     66                                 ;  Move character into silo buffer for this line
     67                                 ;
     68 000356  004767  000076          2$:     CALL    IRINGP          ;Store character into silo buffer
     69                                 ;
     70                                 ;  Set flag requesting fork-level processing for this line
     71                                 ;
     72 000362  032764  000000G 000000G         BIT     #$NDICP,LSW10(R4);Have we already requested input processing?
     73 000370  001030                          BNE     9$              ;Br if yes
     74 000372  052764  000000G 000000G         BIS     #$NDICP,LSW10(R4);Set flag saying input char processing needed
     75 000400  005237  000000G                 INC     @#NEDCDI        ;Say input character processing needed
     76 000404  005727  000000G                 TST     #FASTIN         ;Are we to allow non-clock driven forks?
     77 000410  001020                          BNE     9$              ;Br if not
     78 000412  032764  000000G 000000G         BIT     #$DNICP,LSW10(R4);Have we already done a fork this clock cycle
     79 000420  001014                          BNE     9$              ;Br if yes -- Only do fork once per clock cycl
     80 000422  032764  000000G 000000G         BIT     #$ICPFK,LSW10(R4);Is fork processing already pending for line?
     81 000430  001010                          BNE     9$              ;Br if yes
     82 000432  105737  000000G                 TSTB    @#INITFL        ;Is system initialization complete?
     83 000436  001005                          BNE     9$              ;Br if not -- Don't fork till init complete
     84 000440  052764  000000G 000000G         BIS     #$DNICP,LSW10(R4);Set flag saying we forked during cycle
     85 000446  105267  177354                  INCB    TTIFRK          ;Request fork processing
     86                                 ;
     87                                 ;  Finished
     88                                 ;
     89 000452  012603                  9$:     MOV     (SP)+,R3
     90 000454  012600                          MOV     (SP)+,R0
     91 000456  000207                          RETURN
```

```
                        .SBTTL  IRINGP -- Store character into silo buffer
                ;-----------------------------------------------------------------------
                ;  IRINGP is called to store a received character into the input
                ;  silo buffer for a line.
                ;
                ;  Inputs:
                ;   R4 = Line index number.
                ;   R5 = Character to be stored.
                ;
000460  010146  IRINGP: MOV     R1,-(SP)
000462  010346         MOV     R3,-(SP)
                ;
                ;  Store character into silo buffer
                ;
000464  005764  0000000         TST     LHIRBS(R4)      ;Any space remaining in ring buffer?
000470  001425                 BEQ     2$              ;Br if not
000472  016403  0000000         MOV     LHIRBP(R4),R3   ;Get pointer to next pos in ring buffer
000476  110523                 MOVB    R5,(R3)+        ;Store char into ring buffer
000500  020364  0000000         CMP     R3,LHIRBE(R4)   ;Have we gone past the end of the buffer?
000504  103402                 BLO     1$              ;Br if not
000506  016403  0000000         MOV     LHIRBB(R4),R3   ;Wrap around to the front of the buffer
000512  010364  0000000 1$:     MOV     R3,LHIRBP(R4)   ;Save pointer for next character
000516  005364  0000000         DEC     LHIRBS(R4)      ;One less free space available
                ;
                ;  If buffer is nearly full, send an XOFF to try to stop transmission
                ;
000522  126464  0000000 0000000 CMPB    LHIRBS(R4),LHIRBC(R4);Is buffer nearly full?
000530  101021                 BHI     3$              ;Br if not
000532  001404                 BEQ     2$              ;Always send ctrl-S at set point
000534  032764  000001  0000000 BIT     #1,LHIRBS(R4)   ;Send ctrl-S on every other char if < set pt
000542  001014                 BNE     3$              ;Br if not time to send ctrl-S
000544  016403  0000000 2$:     MOV     LCLUNT(R4),R3   ;Is this line a CL unit?
000550  002404                 BLT     4$              ;Br if not
000552  032763  0000000 0000000 BIT     #CO$BNO,CL$OPT(R3);Are we in binary output mode?
000560  001005                 BNE     3$              ;Br if yes -- Cannot send XOFF
000562  010401         4$:     MOV     R4,R1           ;Get line index number to R1
000564  016103  0000000         MOV     LCDTYP(R1),R3   ;Get hardware device type index
000570  004773  0000000         CALL    @CDSXOF(R3)     ;Call routine to stuff an XOFF character
                ;
                ;  Finished
                ;
000574  012603  3$:     MOV     (SP)+,R3
000576  012601         MOV     (SP)+,R1
000600  000207         RETURN
```

```
                                        .SBTTL  SILFET -- Fetch a character from the input silo
                                ;-----------------------------------------------------------------------
                                ;  SILFET is called to fetch the next character from the input silo.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> A character is available.
                                ;   C-flag set     ==> No more characters in silo.
                                ;   R0 = Character gotten from silo.
                                ;
000602  010346                  SILFET: MOV     R3,-(SP)
000604  005046                          CLR     -(SP)           ;Will restore this into R0
                                ;
                                ;  See if there are any characters in the silo
                                ;
000606  026161  000000G 000000G         CMP     LHIRBS(R1),LHIRBA(R1);Are there any chars in the silo?
000614  001452                          BEQ     8$              ;Br if silo is empty
                                ;
                                ;  There are characters in the silo.
                                ;  Get the next character out of the silo.
                                ;
000616  016103  000000G                 MOV     LHIRBG(R1),R3   ;Get pointer to next character
000622  112316                          MOVB    (R3)+,(SP)      ;Fetch char from silo
000624  020361  000000G                 CMP     R3,LHIRBE(R1)   ;Did we go beyond end of silo?
000630  103402                          BLO     1$              ;Br if not
000632  016103  000000G                 MOV     LHIRBB(R1),R3   ;Wrap around to front of silo
000636  010361  000000G         1$:     MOV     R3,LHIRBG(R1)   ;Save new pointer into silo
000642  005261  000000G                 INC     LHIRBS(R1)      ;Say there is another free space in silo
                                ;
                                ;  If we have transmitted an XOFF because the silo was nearly full,
                                ;  transmit an XON now if we have nearly emptied the silo.
                                ;
000646  032761  000000G 000000G         BIT     #$HISTP,LSW10(R1);Have we transmitted an XOFF?
000654  001416                          BEQ     2$              ;Br if not
000656  016103  000000G                 MOV     LHIRBA(R1),R3   ;Get total silo size
000662  166103  000000G                 SUB     LHIRBS(R1),R3   ;Compute # chars in silo now
000666  120361  000001G                 CMPB    R3,LHIRBC+1(R1) ;Is it time to send XON?
000672  101007                          BHI     2$              ;Br if not
000674  042761  000000G 000000G         BIC     #$HISTP,LSW10(R1);Say XOFF has been cancelled
000702  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get device type code
000706  004770  000000G                 CALL    @CDSXON(R0)     ;Call routine to transmit an XON
                                ;
                                ;  See if we need to translate this character
                                ;
000712  016103  000000G         2$:     MOV     LCXTBL(R1),R3   ;Get pointer to translation table
000716  001407                          BEQ     3$              ;Br if there is no translation table
000720  012300                  4$:     MOV     (R3)+,R0        ;Get next entry from translation table
000722  001405                          BEQ     3$              ;Br if hit end of table
000724  000300                          SWAB    R0              ;Get external char to low order byte
000726  121600                          CMPB    (SP),R0         ;Does this match received character?
000730  001373                          BNE     4$              ;Loop if not
000732  000300                          SWAB    R0              ;Get internal char to low order byte
000734  110016                          MOVB    R0,(SP)         ;Replace external char with internal char
                                ;
                                ;  We got a character
```

```
                                        ;
000736  000241                  3$:     CLC                     ;Signal that we got a character
000740  000401                          BR      9$
                                        ;
                                        ;  There are no characters available
                                        ;
000742  000261                  8$:     SEC                     ;Signal that no characters are available
                                        ;
                                        ;  Finished
                                        ;
000744  012600                  9$:     MOV     (SP)+,R0
000746  012603                          MOV     (SP)+,R3
000750  000207                          RETURN
```

```
                                        .SBTTL  TTRSAV -- Save registers for TT input interrupt
                                ;-----------------------------------------------------------------------
                                ;  TTRSAV should be called from all terminal input interrupt reception
                                ;  routines.  It saves registers R0, R1, and R4 and sets up the stack
                                ;  so that the RTI instruction that would normally return from the
                                ;  interrupt will return to this routine to restore the registers before
                                ;  returning from the interrupt.
                                ;  This is done to allow us to defer doing a .INTEN until later in the
                                ;  character processing.
                                ;
                                ;  Form of call:
                                ;
                                ;       JSR     R5,TTRSAV
                                ;
                                ;  Inputs:
                                ;   Interrupted PC and PSW must be on the top of the stack at the
                                ;   time that the JSR is executed.
                                ;
000752                          TTRSAV:
                                ;
                                ;  On entry, the following items are on the stack: R5, PC, and PSW.
                                ;  Push R0, R1, and R4.
                                ;
000752  010046                         MOV     R0,-(SP)
000754  010146                         MOV     R1,-(SP)
000756  010446                         MOV     R4,-(SP)
000760  013746  0000000                MOV     @#INTPRI,-(SP)  ;Save current running interrupt priority
                                ;
                                ;  Say our running interrupt priority is 7
                                ;
000764  005037  0000000                CLR     @#INTPRI        ;Say running priority = 7
                                ;
                                ;  Now call calling routine as a coroutine
                                ;
000770  004715                         CALL    (R5)            ;Call calling routine as a coroutine
                                ;
                                ;  Finished character input interrupt processing.
                                ;  Restore values we pushed.
                                ;
000772  012637  0000000                MOV     (SP)+,@#INTPRI  ;Restore running interrupt priority
000776  012604                         MOV     (SP)+,R4
001000  012601                         MOV     (SP)+,R1
001002  012600                         MOV     (SP)+,R0
001004  012605                         MOV     (SP)+,R5
                                ;
                                ;  At this point, only the interrupt PC and PSW are left on the stack.
                                ;  See if we need to queue a fork request for input character processing.
                                ;
001006  105767  177014                 TSTB    TTIFRK          ;Do we need to queue a fork request?
001012  001420                         BEQ     9$              ;Br if not
001014  105067  177006                 CLRB    TTIFRK          ;Clear fork request flag
001020  105737  0000000                TSTB    @#CDIFLG        ;Is character processing already active?
001024  001013                         BNE     9$              ;Br if yes
001026  105237  0000000                INCB    @#CDIFLG        ;Say we are doing input processing now
                                ;
                                ;  Do a .INTEN as if we were accepting the interrupt initially
                                ;  (The stack is set up as if an interrupt just occurred)
```

```
                                        ;
001032  004537  0000000                 JSR     R5,@#INTEN      ;Do Inten
001036  000100                          .WORD   100             ;Priority = 5
                                        ;
                                        ;  Now fork and call TT input processing routine
                                        ;
001040  004537  0000000                 JSR     R5,@#FORK       ;Do .FORK
001044  0000000                         .WORD   FP$CDI          ;Fork priority
001046  004767  000004                  CALL    CIPFRK          ;Call TT input processing routine
001052  000207                          RETURN                  ;Return to inten routine which exits from int
                                        ;
                                        ;  Return from interrupt
                                        ;
001054  000002                  9$:    RTI                     ;Return from interrupt
```

```
                                        .SBTTL  CIPFRK -- Fork level input character processing
                                ;-----------------------------------------------------------------------
                                ;  CIPFRK is called at fork level to do input character processing.
                                ;
001056  010146                 CIPFRK: MOV     R1,-(SP)
001060  010446                         MOV     R4,-(SP)
                                ;
                                ;  Begin loop to service each line
                                ;
001062  012701  000002                  MOV     #2,R1           ;Start with line 1
                                ;
                                ;  See if this routine needs input character processing
                                ;
001066  032761  0000000 0000000 1$:     BIT     #$HARD,LSW3(R1) ;Is this a real line?
001074  001435                         BEQ     3$              ;Br if not
001076                                  DISABL                  ;;;** Disable interrupts **
001104  032761  0000000 0000000        BIT     #$NDICP,LSW10(R1);;;Does this line need input char processing?
001112  001423                         BEQ     2$              ;;;Br if not
001114  032761  0000000 0000000        BIT     #$ICPFK,LSW10(R1);;;Are we doing processing now?
001122  001017                         BNE     2$              ;;;Br if yes
                                ;
                                ;  This line does need input character processing
                                ;
001124  052761  0000000 0000000        BIS     #$ICPFK,LSW10(R1);;;Set flag saying processing taking place
001132                                  ENABL                   ;** Enable interrupts **
001140  042761  0000000 0000000        BIC     #$NDICP,LSW10(R1);Say input character processing has been done
001146  010104                         MOV     R1,R4           ;Get line index number in R4 also
001150  004771  0000000                 CALL    @LINIR(R1)      ;Call input processing routine for line
001154  042761  0000000 0000000        BIC     #$ICPFK,LSW10(R1);Say no longer doing input processing
                                ;
                                ;  See if there are more lines to check
                                ;
001162                          2$:     ENABL                   ;** Enable interrupts **
001170  062701  000002          3$:     ADD     #2,R1           ;Get next line index number
001174  020127  0000000                 CMP     R1,#LSTHL       ;Have we done all lines?
001200  101732                         BLOS    1$              ;Loop if more to service
                                ;
                                ;  Say fork level input processing is finished
                                ;
001202  105037  0000000                 CLRB    @#CDIFLG        ;Say input processing finished
                                ;
                                ;  Finished
                                ;
001206  012604                          MOV     (SP)+,R4
001210  012601                          MOV     (SP)+,R1
001212  000207                          RETURN
```

```
                                        .SBTTL  GOTXOF -- Process received XOFF character
                                ;-----------------------------------------------------------------------
                                ;  GOTXOF is called when we receive an XOFF (ctrl-S) character.
                                ;  It suspends transmission.
                                ;
                                ;  Inputs:
                                ;   R4 = Line index number.
                                ;
001214  010146                  GOTXOF: MOV     R1,-(SP)
001216  010401                          MOV     R4,R1           ;Carry line index number in R1
                                ;
                                ;  Set flag saying transmission is to be suspended
                                ;
001220  052761  000000G 000000G         BIS     #$CTRLS,LSW3(R1);Set flag saying ctrl-S received
                                ;
                                ;  Call device-dependent routine to stop the transmitter
                                ;
001226  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get comm device type code
001232  004770  000000G                 CALL    @CDSTOP(R0)     ;Stop the transmitter
                                ;
                                ;  Finished
                                ;
001236  012601                          MOV     (SP)+,R1
001240  000207                          RETURN
```

```
                                        .SBTTL  GOTXON -- Process received XON character
                                ;-----------------------------------------------------------------------
                                ;  GOTXON is called when we receive an XON (ctrl-Q) character.
                                ;
                                ;  Inputs:
                                ;   R4 = Line index number.
                                ;
001242  010146                  GOTXON: MOV     R1,-(SP)
001244  010401                          MOV     R4,R1           ;Get line index number to R1
                                ;
                                ;  Clear flag that says output is suspended
                                ;
001246  042761  000000G 000000G         BIC     #$CTRLS,LSW3(R1);Cancel control-S output suspension
                                ;
                                ;  Call routine to restart output to the line
                                ;
001254  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get device type code
001260  004770  000000G                 CALL    @CDSTRT(R0)     ;Call routine to start transmitter
                                ;
                                ;  Finished
                                ;
001264  012601                          MOV     (SP)+,R1
001266  000207                          RETURN
```

```
                                        .SBTTL  TRNSTR -- Start line transmitter
                                ;-----------------------------------------------------------------------
                                ;  TRNSTR is called each time a character is added to a terminal's
                                ;  output buffer to try to start the transmitter for the terminal.
                                ;  All registers are preserved.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number
                                ;
001270  010046                  TRNSTR: MOV     R0,-(SP)
001272  010146                          MOV     R1,-(SP)
001274  016101  000000G                 MOV     LNPRIM(R1),R1   ;Get primary line index number
                                ;
                                ;  Never try to start output for a detached job
                                ;
001300  032761  000000G 000000G         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
001306  001014                          BNE     9$              ;Br if yes -- No output to detached jobs
                                ;
                                ;  Don't start output if output has been suspended.
                                ;
001310  032761  000000C 000000G         BIT     #<$CTRLS!$DEAD>,LSW3(R1);Line suspended or dead?
001316  001010                          BNE     9$              ;Br if yes
001320  032761  000000G 000000G         BIT     #$HARD,LSW3(R1) ;Is hardware connected to this line?
001326  001404                          BEQ     9$              ;Br if not
                                ;
                                ;  Line is able to receive a character.
                                ;  Call device-dependent routine to start output.
                                ;
001330  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get comm device type index
001334  004770  000000G                 CALL    @CDSTRT(R0)     ;Call routine to start transmitter for line
                                ;
                                ;  Finished
                                ;
001340  012601                  9$:     MOV     (SP)+,R1
001342  012600                          MOV     (SP)+,R0
001344  000207                          RETURN
```

```
                                        .SBTTL  NEDCHR -- Get next char for device driver
                                ;-----------------------------------------------------------------
                                ;  NEDCHR is called from a terminal device driver to get the
                                ;  next char that is to be sent to the terminal.
                                ;
                                ;  Inputs:
                                ;   R4 = Physical line number for which a character is being gotten.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Another character is available.
                                ;   C-flag set     ==> No more characters available.
                                ;   R0 = Character to send if C-flag cleared.
                                ;
001346  010146                  NEDCHR: MOV     R1,-(SP)
001350  010246                          MOV     R2,-(SP)
001352  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if we need to send an XOFF (ctrl-S)
                                ;
001354  032764  000000G 000000G         BIT     #$SXOFF,LSW10(R4);Do we need to send an XOFF?
001362  001406                          BEQ     12$             ;Br if not
001364  042764  000000G 000000G         BIC     #$SXOFF,LSW10(R4);Clear flag saying XOFF needed
001372  112700  000000G                 MOVB    #CTRLS,R0       ;Get ctrl-S character
001376  000556                          BR      15$             ;Go send it
                                ;
                                ;  See if we need to send an XON (ctrl-Q)
                                ;
001400  032764  000000G 000000G 12$:    BIT     #$SXON,LSW10(R4);Do we need to send an XON?
001406  001406                          BEQ     13$             ;Br if not
001410  042764  000000G 000000G         BIC     #$SXON,LSW10(R4);Clear flag saying XON needed
001416  112700  000000G                 MOVB    #CTRLQ,R0       ;Get ctrl-Q character
001422  000544                          BR      15$             ;Go send it
                                ;
                                ;  Get virtual line number to R1
                                ;
001424  116401  000000G         13$:    MOVB    LNMAP(R4),R1    ;Get virtual line number for this phys line
                                ;
                                ;  Determine if output to line is suspended due to Control-S character
                                ;
001430  032761  000000G 000000G         BIT     #$INIT,LSW(R1)  ;Has line been initialized yet?
001436  001540                          BEQ     8$              ;Br if line is not initialized
001440  032764  000000G 000000G         BIT     #$CTRLS,LSW3(R4);Is output to line suspended?
001446  001134                          BNE     8$              ;Br if yes
                                ;
                                ;  See if the next character is coming from a message buffer
                                ;
001450  016402  000000G         2$:     MOV     LMSGBF(R4),R2   ;Is output coming from a message buffer?
001454  001414                          BEQ     3$              ;Br if not
001456  016200  000000G                 MOV     SB$PNT(R2),R0   ;Get address of next character in buffer
001462  005262  000000G                 INC     SB$PNT(R2)      ;Advance the character pointer
001466  112000                          MOVB    (R0)+,R0        ;Get the character from the buffer
001470  001110                          BNE     5$              ;Br if we got a character to send
001472  016264  000000G 000000G         MOV     SB$LNK(R2),LMSGBF(R4) ;Remove this element from list for job
001500  004767  000246                  CALL    SNDFRE          ;Free the message buffer
001504  000761                          BR      2$              ;See if there is another to send
                                ;
                                ;  See if line is cross connected to a CL line
```

```
                                        ;
001506  016402  0000000         3$:     MOV     LXCL(R4),R2     ;Is line cross connected to a CL line?
001512  002406                         BLT     14$             ;Br if not
001514  016201  0000000                 MOV     CL$LIX(R2),R1   ;Get # of line CL unit is using
001520  004767  177056                 CALL    SILFET          ;Try to get char from input silo
001524  103072                         BCC     5$              ;Br if got a character
001526  000504                         BR      8$              ;No character to transmit
                                        ;
                                        ;  Determine how many characters are in terminal output buffer for line
                                        ;
001530  016100  0000000         14$:    MOV     LOTSIZ(R1),R0   ;Get size of output character buffer
001534  166100  0000000                 SUB     LOTSPC(R1),R0   ;Determine number of characters
001540  001477                         BEQ     8$              ;Br if there are no characters in output buf
                                        ;
                                        ;  If we are about to run out of characters, see if we need
                                        ;  to give the job an execution priority boost
                                        ;
001542  026127  0000000 0000000         CMP     LSTATE(R1),#S$OTFN ;Is job already in a high-priority state?
001550  101433                         BLOS    10$             ;Br if yes -- no need to boost priority
001552  005761  0000000                 TST     LITIME(R1)      ;Is this an interactive job?
001556  001004                         BNE     11$             ;Br if yes
001560  026127  0000000 0000000         CMP     LSTATE(R1),#S$OTLO ;Non-interactive high priority state?
001566  101424                         BLOS    10$             ;Br if already in high prio state
001570  020027  000005          11$:    CMP     R0,#5.          ;Are we really about to run out of chars?
001574  103410                         BLO     6$              ;Br if yes -- Definitely give boost
001576  026127  0000000 0000000         CMP     LSTATE(R1),#S$OTWT ;Is job waiting for output buffer space?
001604  001015                         BNE     10$             ;Br if not
001606  020027  0000000                 CMP     R0,#OTRASZ      ;Are we down to reactivation point?
001612  003012                         BGT     10$             ;Br if not
001614  000404                         BR      7$              ;Br if yes -- reactivate user
001616  026127  0000000 0000000 6$:    CMP     LSTATE(R1),#S$$RUN ;Is job in a run or wait state?
001624  101005                         BHI     10$             ;Br if in wait state -- Don't boost
001626  052761  0000000 0000000 7$:    BIS     #$SOTFN,LSW7(R1);Set flag to cause clock int to boost priority
001634  005237  0000000                 INC     @#NEDSOT        ;Say we need job scheduler processing
                                        ;
                                        ;  Send next character from normal terminal output buffer
                                        ;
001640  016102  0000000         10$:    MOV     LOTPNT(R1),R2   ;Get pointer to next character in buffer
001644                                  TTMAP                   ;Map kernel PAR 6 to TT buffer for job
001660  112200                         MOVB    (R2)+,R0        ;Get next character from terminal buffer
001662                                  UNMAP                   ;Remap kernel PAR 6
001670  005261  0000000                 INC     LOTSPC(R1)      ;Say there is one more char space in buffer
001674  020261  0000000                 CMP     R2,LOTEND(R1)   ;Did we just go beyond end of buffer?
001700  103402                         BLO     4$              ;Br if not
001702  016102  0000000                 MOV     LOTBUF(R1),R2   ;Wrap around to front of buffer
001706  010261  0000000         4$:     MOV     R2,LOTPNT(R1)   ;Save updated buffer pointer
                                        ;
                                        ;  There is a character to send.
                                        ;  See if we need to do any character translation.
                                        ;
001712  016402  0000000         5$:     MOV     LCXTBL(R4),R2   ;Is there a character translation table?
001716  001406                         BEQ     15$             ;Br if not
001720  012201                 16$:    MOV     (R2)+,R1        ;Get entry from translation table
001722  001404                         BEQ     15$             ;Br if hit end of table
001724  120001                         CMPB    R0,R1           ;Compare character with that in table
001726  001374                         BNE     16$             ;Br if no match
001730  000301                         SWAB    R1              ;Get replacement char to low order
```

```
    115 001732  110100                          MOVB    R1,R0           ;Get replacement character
    116                                 ;
    117                                 ;  We have a character to send in R0
    118                                 ;
    119 001734  000241                  15$:    CLC                     ;Signal success on return
    120 001736  000401                          BR      9$
    121                                 ;
    122                                 ;  No characters left to send
    123                                 ;
    124 001740  000261                  8$:     SEC                     ;Signal failure on return
    125                                 ;
    126                                 ;  Finished
    127                                 ;
    128 001742  012604                  9$:     MOV     (SP)+,R4
    129 001744  012602                          MOV     (SP)+,R2
    130 001746  012601                          MOV     (SP)+,R1
    131 001750  000207                          RETURN
```

```
                                        ;
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;  SNDFRE is called to return a system message buffer to the free list.
                                        ;
                                        ;  Inputs:
                                        ;   R2 = Address of system message buffer to be freed.
                                        ;   Interrupts must be disabled when this routine is called.
                                        ;
                                             .GLOBL  SNMSHD, SB$LNK, NMUMB, MBFFLG

001752                                  SNDFRE:
                                        ;
                                        ;  Return buffer to free list.
                                        ;
001752  013762  0000000 0000000              MOV     @#SNMSHD,SB$LNK(R2);Add buffer to free list
001760  010237  0000000                      MOV     R2,@#SNMSHD
001764  005237  0000000                      INC     @#NMUMB         ;Increment # free message buffers
                                        ;
                                        ;  Set flag saying a message buffer has been freed so clock routine
                                        ;  will restart any waiting jobs.
                                        ;
001770  005237  0000000                      INC     @#MBFFLG        ;Set flag for clock routine
                                        ;
                                        ;  Finished
                                        ;
001774  000207                               RETURN
```

```
                                        .SBTTL  CDORTN -- Clock driven output character processing
                                ;-----------------------------------------------------------------------
                                ;  CDORTN is called on every clock tick (50/60 Hz) to do clock
                                ;  driven output character processing.
                                ;
001776  010146                  CDORTN: MOV     R1,-(SP)
002000  005037  000000G                 CLR     @#NEDCDO        ;Say don't need output processing
                                ;
                                ;  Do processing for CL lines
                                ;
002004  005737  000000G                 TST     @#NEDCLO        ;Need CL output processing?
002010  001404                          BEQ     1$              ;Br if not
002012  005037  000000G                 CLR     @#NEDCLO        ;Reset output-processing-needed flag
002016  004737  000000G                 CALL    @#CLTIMR        ;Call routine for CL lines
                                ;
                                ;  Do output character processing for DH11 and DHV11 lines
                                ;
002022  012701  000000G         1$:     MOV     #LSTHL,R1       ;Get index to last line
002026  032761  000000G 000000G 2$:     BIT     #$DHCDO,LSW10(R1);Is clock-driven processing needed for line?
002034  001415                          BEQ     4$              ;Br if not
002036  026127  000000G 000000G         CMP     LCDTYP(R1),#CDX$VH;Is this a DHV11 line?
002044  001404                          BEQ     3$              ;Br if yes
002046  026127  000000G 000000G         CMP     LCDTYP(R1),#CDX$DH;Is this a DH11 line?
002054  001005                          BNE     4$              ;Br if not
002056  042761  000000G 000000G 3$:     BIC     #$DHCDO,LSW10(R1);Clear processing-needed flag for line
002064  004737  000000G                 CALL    @#DHTIMR        ;Call clock driven routine for DH
002070  162701  000002          4$:     SUB     #2,R1           ;Get next line index number
002074  003354                          BGT     2$              ;Loop if more lines to check
                                ;
                                ;  Say processing is finished
                                ;
002076  105037  000000G                 CLRB    @#CDOFLG        ;Say processing is finished
                                ;
                                ;  Finished
                                ;
002102  012601                          MOV     (SP)+,R1
002104  000207                          RETURN
```

```
                                        .SBTTL  CDIRTN -- Clock driven input character processing
                                ;-------------------------------------------------------------------
                                ;  CDURTN is called on every clock tick (50/60 Hz) to do clock
                                ;  driven input character processing.
                                ;
002106  010146                 CDIRTN: MOV     R1,-(SP)
002110  010446                         MOV     R4,-(SP)
002112  005037  000000G                CLR     @#NEDCDI        ;Say don't need input processing
                                ;
                                ;  Begin loop to service each line
                                ;
002116  012701  000002                 MOV     #2,R1           ;Start with line 1
                                ;
                                ;  See if this routine needs input character processing
                                ;
002122  032761  000000G 000000G 1$:    BIT     #$HARD,LSW3(R1) ;Is this a real line?
002130  001440                         BEQ     3$              ;Br if not
002132                                 DISABL                  ;;;** Disable interrupts **
002140  032761  000000G 000000G        BIT     #$NDICP,LSW10(R1);;;Does this line need input char processing?
002146  001423                         BEQ     2$              ;;;Br if not
002150  032761  000000G 000000G        BIT     #$ICPFK,LSW10(R1);;;Are we doing processing now?
002156  001017                         BNE     2$              ;;;Br if yes
                                ;
                                ;  This line does need input character processing
                                ;
002160  052761  000000G 000000G        BIS     #$ICPFK,LSW10(R1);;;Set flag saying processing taking place
002166                                 ENABL                   ;** Enable interrupts **
002174  042761  000000G 000000G        BIC     #$NDICP,LSW10(R1);Say input character processing has been done
002202  010104                         MOV     R1,R4           ;Get line index number in R4 also
002204  004771  000000G                CALL    @LINIR(R1)      ;Call input processing routine for line
002210  042761  000000G 000000G        BIC     #$ICPFK,LSW10(R1);Say no longer doing input processing
002216  042761  000000G 000000G 2$:    BIC     #$DNICP,LSW10(R1);Say input processing done for clock cycle
                                ;
                                ;  See if there are more lines to check
                                ;
002224                                 ENABL                   ;** Enable interrupts **
002232  062701  000002         3$:    ADD     #2,R1           ;Get next line index number
002236  020127  000000G                CMP     R1,#LSTHL       ;Have we done all lines?
002242  101727                         BLOS    1$              ;Loop if more to service
                                ;
                                ;  Say clock driven input processing is finished
                                ;
002244  105037  000000G                CLRB    @#CDIFLG        ;Say clock driven input processing finished
                                ;
                                ;  Finished
                                ;
002250  012604                         MOV     (SP)+,R4
002252  012601                         MOV     (SP)+,R1
002254  000207                         RETURN
        000001                         .END
```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
Work file writes: 0

```
Size of work file:  248 Words  ( 1 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:42.43
DK:TSTIOX,LP:TSTIOX=DK:TSTIOX.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

```
$8BIT    1-26     5-35
$CTRLS   1-24    10-14    11-13    12-21    13-42
$DEAD    1-29    12-21
$DETCH   1-24    12-16
$DHCDO   1-26    15-19    15-25
$DNICP   1-31     5-78     5-84    16-32
$HARD    1-24     9-14    12-23    16-16
$HISTP   1-30     7-35     7-41
$ICPFK   1-29     5-80     9-19     9-24     9-29    16-21    16-26    16-31
$IITIM   1-26     5-16
$INIT    1-25    13-40
$NDICP   1-28     5-72     5-74     9-17     9-26    16-19    16-28
$PAGE    1-27     5-50     5-61
$RBRK    1-28     5-22
$SOTFN   1-27    13-90
$SXOFF   1-26    13-20    13-22
$SXON    1-25    13-28    13-30
CDIFLG   1-28     8-52     8-54*    9-40*   16-43*
CDIRTN   3-17    16-6#
CDOFLG   1-23    15-32*
CDORTN   3-16    15-6#
CDSTOP   1-30    10-19
CDSTRT   1-30    11-18    12-30
CDSXOF   1-22     6-38
CDSXON   1-22     7-43
CDX$DH   1-25    15-23
CDX$VH   1-22    15-21
CIPFRK   8-66     9-5#
CL$LIX   1-20    13-61
CL$OPT   1-29     5-30     5-32     6-34
CLTIMR   1-22    15-14
CO$8BT   1-24     5-32
CO$BNI   1-31     5-30
CO$BNO   1-31     6-34
CTRLQ    1-22     5-57    13-31
CTRLS    1-21     5-46    13-23
DHTIMR   1-27    15-26
FASTIN   1-30     5-76
FORK     1-28     8-64
FP$CDI   1-28     8-65
FRMERR   1-25     5-20
GOTXOF   5-52    10-9#
GOTXON   5-63    11-8#
ININT    3-10     4-6#
INITFL   1-24     5-82
INTEN    1-28     8-59
INTPRI   1-23     8-27     8-31*    8-40*    9-25    9-33    16-27    16-36
IRINGP   5-68     6-10#
KPAR6    1-31    13-96    13-96*   13-98*
LCDTYP   1-21     6-37     7-42    10-18    11-17    12-29    15-21    15-23
LCLUNT   1-21     5-28     6-32
LCXTBL   1-20     7-47    13-108
LHIRBA   1-24     7-18     7-37
LHIRBB   1-23     6-21     7-28
LHIRBC   1-24     6-27     7-39
LHIRBE   1-23     6-19     7-26
```

Cross reference table (CREF V05.04)

```
LHIRBG    1-24      7-24      7-29*
LHIRBP    1-23      6-17      6-22*
LHIRBS    1-23      6-15      6-23*     6-27      6-30      7-18      7-30*     7-38
LINIR     1-21      9-28     16-30
LINNUM    1-32      4-15      4-20*
LITIME    1-26     13-77
LMSGBF    1-22     13-47     13-53*
LNMAP     1-23     13-36
LNPRIM    1-30     12-12
LOTBUF    1-21     13-102
LOTEND    1-27     13-100
LOTPNT    1-22     13-95     13-103*
LOTSIZ    1-21     13-68
LOTSPC    1-31     13-69     13-99*
LSTATE    1-30     13-75     13-79     13-83     13-88
LSTHL     1-27      9-35     15-18     16-38
LSW       1-25     12-16     13-40
LSW10     1-30      5-22*     5-72      5-74*     5-78      5-80      5-84*     7-35      7-41*     9-17      9-19      9-24*
          9-26*     9-29*    13-20     13-22*    13-28     13-30*    15-19     15-25*    16-19     16-21     16-26*    16-28*
         16-31*    16-32*
LSW2      1-29      5-35      5-50      5-61
LSW3      1-26      9-14     10-14*    11-13*    12-21     12-23     13-42     16-16
LSW5      1-26      5-16*
LSW7      1-25     13-90*
LTTPAR    1-31     13-96
LXCL      1-22     13-59
MAPHLD    3-22#    13-96*    13-98
MBFFLG    1-28     14-10     14-23*
MUXNUM    1-32      4-35      4-36*
MXLNT     1-32      4-43
MXRBUF    1-32      4-38
NEDCDI    1-21      5-75*    16-8*
NEDCDO    1-21     15-7*
NEDCHR    3-15     13-14#
NEDCLO    1-29     15-11     15-13*
NEDSOT    1-29     13-91*
NMUMB     1-27     14-10     14-18*
OTRASZ    1-23     13-85
OVRHC     1-20
PSW       1-31      9-16*     9-25*     9-33*    16-18*    16-27*    16-36*
RBR       1-32      4-25
RCVDON    1-32      4-23
RDINT     1-32      4-22      4-26      4-29
RSR       1-32      4-22*     4-23      4-26*     4-29*
S$$RUN    1-26     13-88
S$OTFN    1-28     13-75
S$OTLO    1-27     13-79
S$OTWT    1-25     13-83
SB$LNK    1-27     13-53     14-10     14-16*
SB$PNT    1-29     13-49     13-50*
SILFET    3-12      7-13#    13-62
SNDFRE    3-18     13-54     14-12#
SNMSHD    1-29     14-10     14-16     14-17*
TRNSTR    3-14     12-10#
TSTIOX    1-16      3-6#      3-10      3-11      3-12      3-13      3-14      3-15      3-16      3-17      3-18
TTIFRK    3-24#     5-85*     8-49      8-51*
```

Cross reference table (CREF V05.04)

TTINPT	3-11	4-27	4-46	5-11#
TTRSAV	3-13	4-10	8-19#	

Cross reference table (CREF V05.04)

DISABL	2-6#	9-16	16-18		
ENABL	2-12#	9-25	9-33	16-27	16-36
OCALL	2-18#				
TTMAP	2-48#	13-96			
TTMAPX	2-39#				
UNMAP	2-53#	13-98			
UNMAPX	2-45#				