Table of contents

Table of contents

```
                                        .TITLE  TSXPRO -- TSX-Plus PROxxx Routines
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
000000                                  .CSECT  TSXPRO
000000                          TSXPRO:
000000                          PROBAS:
                                ;-------------------------------------------------------------------------
                                ;  TSX-Plus interface routines for Digital PRO-3xx series
                                ;  personal computers.
                                ;
                                ;  Copyright (c) 1984, 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee  37212
                                ;  U.S.A.
                                ;
                                ;  Assembly parameters
                                ;
                                ;  Set QPASM to 1 to include support for quad serial line unit.
                                ;  Set QPASM to 0 to exclude quad line unit code.
                                ;
000001                          QPASM   =       1               ;Select quad port code
000017                          MAXPOC  =       15.             ;Max consecutive output chars to PI
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  TSXPRO
                                        .GLOBL  PROSIZ,PROLIN,PROINI,PROHAN,PIHAN,PRONOP
                                        .GLOBL  PIDRIV
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  FRKCQE,DOSCHD,CORUSR,LXCL
                                        .GLOBL  LMXPRM,$XCHAR,LSW3,$OITIM,LSW5,PROSLT
                                        .GLOBL  FORK,INTEN,TTINPT,RBERR,OVRRUN,FRMERR,RCVPAR
                                        .GLOBL  RSR,PSW,INTPRI,CDSTRT,CDGDSS,CDSDSS,$PHONE,ILSW2
                                        .GLOBL  LCDTYP,CDX$PC,CDX$PP,MS$DTR,MS$CAR,MS$RNG
                                        .GLOBL  PROBRK,$DEAD,PPTERM,LSTHL,CDX$PI,$HISTP
                                        .GLOBL  INVEC,LOUTIR,S9600,CDSXOF,CDSXON
                                        .GLOBL  KPAR5,KPAR6,VPAR5,VPAR6,PISRT,CDCLOK,VSWPSL
                                        .GLOBL  RT$BAS,IOFIN,FP$PIO,LSW10,$SXON,$SXOFF
                                        .GLOBL  MS$BRK,CDSBRK,CDSSPD,VT100,ITRMTP,TTRSAV
                                        .GLOBL  LP$SPD,LP$PAR,LP$ODD,LP$7BT,S4800
                                        .GLOBL  RPRCSR,RPRVEC,DWTYPE,VIDCSR,CDX$QP
                                        .GLOBL  SROMMR,SR3MMR,MMENBL,EMMAP,LMXNUM,LMXLN
                                        .GLOBL  MXVEC,MXCSR,PROBUF
```

```
                        .SBTTL  Macros
                ;-------------------------------------------------------------------------------
                ;  Macro to disable interrupts
                ;
                        .MACRO  DISABL
                        BIS     #340,@#PSW
                        .ENDM   DISABL
                ;
                ;  Macro to enable interrupts
                ;
                        .MACRO  ENABL
                        BIC     INTPRI,@#PSW
                        .ENDM   ENABL
```

```
                                        .SBTTL  Data areas
                                ;-------------------------------------------------------------------------
                                ;  Vector of addresses used to move information between TSXPRO and PI handler.
                                ;
000000                          PROVEC:
                                ;
                                ;  Addresses passed from the PI handler to TSXPRO
                                ;
000000  000000                  PIINAD: .WORD   0               ;Address of initialization code
000002  000000                  PIINWD: .WORD   0               ;Number of words of initialization code
000004  000000                  PIXOCH: .WORD   0               ;Routine called to send a char to the video
000006  000000                  PIXEOF: .WORD   0               ;Routine called for end-of-frame interrupts
000010  000000                  PIXICH: .WORD   0               ;Routine to process keyboard input interrupts
000012  000000                  PIXIOI: .WORD   0               ;Routine to process keyboard output interrupts
000014  000000                  PIXIOQ: .WORD   0               ;Routine to process an I/O queue request
000016  000000                  $VDCSR: .WORD   0               ;Address of video CSR address
000020  000000                  REENAB: .WORD   0               ;Pointer to REENAB cell
000022  000000                  PRIOO:  .WORD   0               ;Pointer to PRIOO cell
000024  000000                  VDFLAG: .WORD   0               ;Pointer to VDFLAG cell
000026  000000                  PIGOFL: .WORD   0               ;Pointer to PIGOFL cell (1==>PI run, 0==>stop)
                                ;
                                ;  Addresses passed from TSXPRO to the PI handler
                                ;
000030  002156'                          .WORD   KBDCHR          ;Routine to process an input character
000032  0000000                         .WORD   FORK            ;Address of system .FORK routine
000034  001740'                          .WORD   PINDCH          ;Routine to get next char to send
000036  001172'                          .WORD   GPRCSR          ;Routine to get CSR address for device
000040  001254'                          .WORD   GETSLT          ;Routine to get option slot # for device
                                ;
                                ;  End of address vector
                                ;
000042  000000                          .WORD   0               ;End of PROVEC vector
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Data areas:
                                ;
000044  000000                  PILINE: .WORD   0               ;# of line connected to Pro console
000046  000000                  PPLINE: .WORD   0               ;# of line connected to printer port
000050  000000                  CPLINE: .WORD   0               ;# of line connected to communications port
000052  000000                  PIBASE: .WORD   0               ;Base 64-byte block # of PI handler
000054  000000                  PIOIFL: .WORD   0               ;Non-zero ==> Doing PI output interrupt
000056  177777                  PISCNT: .WORD   -1              ;Counts output starts per clock period
000060  062550                  R5OPI:  .RAD50  /PI /           ;Name of PI handler
000062  000000                  VIDSLT: .WORD   0               ;Option slot # where video controller is
000064  000000                  VIDVEC: .WORD   0               ;Address of video vector A
000066  000000                  QPCSR:  .WORD   0               ;Address of base of registers for quad port
000070  000000                  QPVEC:  .WORD   0               ;Address of vector for quad port
000072  000017                  PIMOC:  .WORD   MAXPOC          ;Limit consecutive output chars to PI
                                ;
                                ;  Table of possible video device ID values
                                ;
000074  001002                  VIDTBL: .WORD   1002            ;350 video
000076  000050                          .WORD   50              ;380 video
000100  010050                          .WORD   10050           ;380 video with EBO
000102  002002                          .WORD   2002            ;IVIS video
                                ;
                                ;  Byte data
```

Data areas

```
      58                                        ;
      59 000104      000      000      000     QPLX:    .BYTE   0,0,0,0          ;Line #'s connected to each quad port line
         000107      000
      60 000110      000                       QPMODM:  .BYTE   0                ;0==>4/0 configuration, 1==>2/2 configuration
      61 000111      000                       QPSLOT:  .BYTE   0                ;Slot # where quad port controller installed
      62                                                .EVEN
```

```
                                        .SBTTL  System control register
                                ;-------------------------------------------------------------------------
                                ;  Control values in system status register
                                ;
000000 173700                   PSSREG  =       173700          ;Address of status register
                                ;
                                ;  Flags in status register
                                ;
000007                          SS$BNK  =       7               ;Mask for memory bank information
000020                          SS$MON  =       20              ;Monitor-present flag
000200                          SS$BRK  =       200             ;Break enable flag for diagnostic port
```

```
                                        .SBTTL  Interrupt control values
                                ;-----------------------------------------------------------------------
                                ;  The following values relate to the interrupt control system on the
                                ;  Professional computer.
                                ;
                                ;  Addresses of control registers
                                ;
173200                          ICODR   =       173200          ;Interrupt controller 0 data register
173202                          ICOCSR  =       173202          ;Interrupt controller 0 control register
173204                          IC1DR   =       173204          ;Interrupt controller 1 data register
173206                          IC1CSR  =       173206          ;Interrupt controller 1 control register
173210                          IC2DR   =       173210          ;Interrupt controller 2 data register
173212                          IC2CSR  =       173212          ;Interrupt controller 2 control register
                                ;
                                ;  Interrupt Mode Register
                                ;
000001                          IM$PM   =       1               ;Priority mode (0=fixed, 1=rotating)
000002                          IM$VS   =       2               ;Vector selection
000004                          IM$IM   =       4               ;Interrupt mode (1==>Do not interrupt)
000010                          IM$GIP  =       10              ;Group interrupt polarity
000020                          IM$IRP  =       20              ;Interrupt request polarity
000140                          IM$RP   =       140             ;Mask for register preselect values
000200                          IM$MM   =       200             ;Master mask (1==>Enable group interrupts)
000200                          IM$STV  =       200             ;Standard value for TSX-Plus
                                ;
                                ;  Command values that can be stored into CSR register
                                ;
000000                          IM$RST  =       000             ;Reset
000020                          IM$ZRM  =       020             ;Clear bits in IRR and IMR registers
000030                          IM$CRM  =       030             ;Clear single bit in IRR and IMR registers
000040                          IM$ZM   =       040             ;Clear all bits in IMR register
000050                          IM$CM   =       050             ;Clear single bit in IMR register
000060                          IM$OM   =       060             ;Set all bits in IMR register to one
000070                          IM$SM   =       070             ;Set single bit in IMR register
000100                          IM$ZR   =       100             ;Clear all bits in IRR register
000110                          IM$CR   =       110             ;Clear single bit in IRR register
000120                          IM$OR   =       120             ;Set all bits in IRR register to ones
000130                          IM$SR   =       130             ;Set single bit in IRR register
000140                          IM$CHP  =       140             ;Clear highest priority ISR bit
000160                          IM$ZS   =       160             ;Clear ISR register to zero
000170                          IM$CS   =       170             ;Clear single bit in ISR register
000200                          IM$LMB  =       200             ;Load mode bits
000240                          IM$CMB  =       240             ;Control mode bits
000260                          IM$PR   =       260             ;Preselect IMR register
000300                          IM$PA   =       300             ;Preselect ACR register
000340                          IM$PRM  =       340             ;Preselect response memory
                                ;
                                ;  Values used with command codes to select specific interrupts
                                ;
000001                          IM$KBR  =       1               ;IC0 - Keyboard receiver interrupt
000002                          IM$KBT  =       2               ;IC0 - Keyboard transmitter interrupt
000003                          IM$CPD  =       3               ;IC0 - Communications port data transfer
000004                          IM$CPM  =       4               ;IC0 - Communications port modem change
000005                          IM$PPR  =       5               ;IC0 - Printer port receiver interrupt
000006                          IM$PPT  =       6               ;IC0 - Printer port transmitter interrupt
000007                          IM$CLK  =       7               ;IC0 - Clock interrupt
```

```
                                .SBTTL  Video display control values
                        ;-----------------------------------------------------------------------
                        ;  The following values relate to the video display.
                        ;
                        ;  Control flags in Constrol/status register
                        ;
000001          VP$LMD  =       1               ;Line mode definition
000002          VP$IMD  =       2               ;Interlace mode definition
000040          VP$OEF  =       40              ;Odd/Even frame flag
000100          VP$EFI  =       100             ;End of frame interrupt enable
000200          VP$EOF  =       200             ;End of frame flag
001400          VP$COO  =       1400            ;Mask for class of operation
002000          VP$CME  =       2000            ;Color map enable
010000          VP$OMP  =       10000           ;Option module presence flag
040000          VP$DOI  =       40000           ;Done interrupt enable
100000          VP$TRD  =       100000          ;Transfer done flag
```

```
                                        .SBTTL  Printer port control values
                                ;-----------------------------------------------------------------------
                                ;  The following values relate to the "printer port" which can optionally
                                ;  be used as a time-sharing terminal under TSX-Plus.
                                ;
                                ;  Register addresses and vectors
                                ;
000000  173400                  PP$DBR  =       173400          ;Data buffer register
000002  173402                  PP$STR  =       173402          ;Status register
000004  173404                  PP$MDR  =       173404          ;Mode registers
000006  173406                  PP$CMR  =       173406          ;Command register
000010  177560                  PP$RCS  =       177560          ;Receiver CSR for maintenance mode
000014  177564                  PP$TCS  =       177564          ;Transmitter CSR for maintenance mode
                                ;
000020  000220                  PP$RCV  =       220             ;Receiver vector
000024  000224                  PP$TRV  =       224             ;Transmitter vector
                                ;
                                ;  Flags in the status register
                                ;
000001  000001                  PP$TR   =       1               ;Transmitter ready
000002  000002                  PP$RD   =       2               ;Receive done
000004  000010                  PP$PE   =       10              ;Parity error detected
000006  000020                  PP$OE   =       20              ;Overrun error
000010  000040                  PP$FE   =       40              ;Framing error
000012  000200                  PP$DSR  =       200             ;Data set ready
                                ;
                                ;  Flags in mode register 1
                                ;
000000  000014                  PP$LEN  =       14              ;Mask for character length
000000  000010                  PP$7BT  =       10              ;7 bit characters
000000  000014                  PP$8BT  =       14              ;8 bit characters
000002  000020                  PP$PAR  =       20              ;Enable parity control
000004  000040                  PP$EVN  =       40              ;1==>even parity, 0==>odd parity
000006  000300                  PP$SBL  =       300             ;Mask for stop bit length
000010  000102                  PP$M1F  =       102             ;Standard mode register 1 value for TSX-Plus
                                ;
                                ;  Flags in mode register 2
                                ;
000000  000017                  PP$BRS  =       17              ;Mask for baud rate select field
000010  000260                  PP$M2F  =       260             ;Standard mode register 2 value for TSX-Plus
                                ;
                                ;  Command register
                                ;
000000  000001                  PP$TEN  =       1               ;Transmitter enable
000001  000002                  PP$DTR  =       2               ;Data terminal ready
000002  000004                  PP$REN  =       4               ;Receiver enable
000003  000010                  PP$FB   =       10              ;Force break transmission
000004  000020                  PP$RE   =       20              ;Reset error
000005  000040                  PP$RTS  =       40              ;Request to send
000006  000300                  PP$OM   =       300             ;Operating mode mask
000010  000047                  PP$CMF  =       47              ;Standard command register value for TSX-Plus
```

```
                                .SBTTL  Communications port control values
                        ;-----------------------------------------------------------------------
                        ;  Control values for the communications port which may be used as
                        ;  a TSX-Plus time-sharing line.
                        ;
                        ;  Control registers and vectors
                        ;
173300                  CP$DBR  =       173300          ;Data buffer register
173302                  CP$CAR  =       173302          ;Control/status register A
173306                  CP$CBR  =       173306          ;Control/status register B
173310                  CP$MOR  =       173310          ;Modem control register 0
173312                  CP$M1R  =       173312          ;Modem control register 1
173314                  CP$BRR  =       173314          ;Baud rate register
                        ;
000210                  CP$RTV  =       210             ;Receive/transmit vector
000214                  CP$MCV  =       214             ;Modem change vector
                        ;
                        ;  Control/status Register A
                        ;
000007                  CP$ARP  =       7               ;Mask to select read/write sub-register
000070                  CP$ACM  =       70              ;Mask for command bits
000300                  CP$CRC  =       300             ;Mask for CRC control bits
                        ;
                        ;  Read/Write sub-register select values
                        ;  (These values are stored into CP$ARP and CP$BRP to select sub registers)
                        ;
000000                  CP$WR0  =       0               ;Write register 0
000001                  CP$WR1  =       1               ;Write register 1
000002                  CP$WR2  =       2               ;Write register 2
000003                  CP$WR3  =       3               ;Write register 3
000004                  CP$WR4  =       4               ;Write register 4
000005                  CP$WR5  =       5               ;Write register 5
000006                  CP$WR6  =       6               ;Write register 6
000007                  CP$WR7  =       7               ;Write register 7
000000                  CP$RR0  =       0               ;Read register 0
000001                  CP$RR1  =       1               ;Read register 1
000002                  CP$RR2  =       2               ;Read register 2
                        ;
                        ;  Command values for WR0 under control/status register A
                        ;
000010                  CP$SA   =       10              ;Send abort
000020                  CP$RES  =       20              ;Reset external interrupt
000030                  CP$CR   =       30              ;Channel reset
000040                  CP$EIR  =       40              ;Enable interrupt on next char received
000050                  CP$RTI  =       50              ;Reset transmitter interrupt pending
000060                  CP$ER   =       60              ;Error reset
000070                  CP$EI   =       70              ;End of interrupt
                        ;
                        ;  Sub-register WR1 under control/status register A
                        ;
000001                  CP$EIE  =       1               ;External interrupt enable
000002                  CP$TIE  =       2               ;Transmitter interrupt enable
000030                  CP$RIE  =       30              ;Mask for receiver interrupt enable flags
                        ;
                        ;  Sub-register WR3 under control/status register A
                        ;
000001                  CP$REN  =       1               ;Receiver enable
```

Communications port control values

```
                                CP$SCL  =       2               ;Sync character load inhibit
                                CP$ASM  =       4               ;Address search mode
                                CP$RCE  =       10              ;Receiver CRC enable
                                CP$EHP  =       20              ;Enter hunt phase
                                CP$RCL  =       300             ;Receiver character length
                                CP$7BR  =       100             ;7 bit characters
                                CP$8BR  =       300             ;8 bit characters
                                CP$AW3  =       001             ;Standard value for TSX-Plus
                                ;
                                ;  Sub-register WR4 under control/status register A
                                ;
                                CP$PAR  =       1               ;Parity enable
                                CP$EVN  =       2               ;1==>even parity, 0==>odd parity
                                CP$SBS  =       14              ;Mask for stop bits select value
                                CP$SMS  =       60              ;Mask for synchronous mode control flags
                                CP$CMM  =       300             ;Mask for clock mode value
                                CP$AW4  =       104             ;Standard value for TSX-Plus
                                ;
                                ;  Sub-register WR5 under control/status register A
                                ;
                                CP$TCE  =       1               ;Transmitter CRC enable
                                CP$CCS  =       4               ;CRC polynomial select
                                CP$TEN  =       10              ;Transmitter enable
                                CP$SB   =       20              ;Send break
                                CP$LEN  =       14              ;Transmitter character length
                                CP$7BT  =       040             ;7 bit characters
                                CP$8BT  =       140             ;8 bit characters
                                CP$AW5  =       010             ;Standard value for TSX-Plus
                                ;
                                ;  Sub-register RR0 under control/status register A
                                ;
                                CP$RCA  =       1               ;Receive character available
                                CP$INP  =       2               ;Interrupt pending
                                CP$TBM  =       4               ;Transmit buffer empty
                                CP$SH   =       20              ;Sync/hunt
                                CP$TEM  =       100             ;Transmitter underrun/end of message
                                CP$BR   =       200             ;Break received
                                ;
                                ;  Sub-register RR1 under control/status register A
                                ;
                                CP$AS   =       1               ;All sent -- Transmitter ready for next char
                                CP$RC   =       16              ;Mask for residue codes
                                CP$RPE  =       20              ;Received parity error
                                CP$ROE  =       40              ;Receiver overrun error
                                CP$RFE  =       100             ;Framing error
                                CP$EOF  =       200             ;End of frame
                                ;
                                ;  Control/status Register B
                                ;
                                CP$BRP  =       7               ;Mask to select sub-register
                                ;
                                ;  Sub-register WR1 under control/status register B
                                ;
                                CP$BW1  =       4               ;Standard value for TSX-Plus
                                ;
                                ;  Sub-register WR2 under control/status register B
                                ;
```

Communications port control values

```
    115         077777                  CP$BW2  =       77777           ;Standard value for TSX-Plus
    116                                 ;
    117                                 ;  Sub-register RR2 under control/status register B
    118                                 ;
    119         000000                  CP$ITE  =       0               ;Transmitter buffer empty
    120         000024                  CP$IES  =       24              ;External status change
    121         000030                  CP$IRC  =       30              ;Receiver character available
    122         000034                  CP$ISR  =       34              ;Special receiver condition
    123                                 ;
    124                                 ;  Modem Control Register 0
    125                                 ;
    126         000001                  CP$LL   =       1               ;Local loopback
    127         000002                  CP$RL   =       2               ;Remote loopback
    128         000004                  CP$SRS  =       4               ;Signaling rate select
    129         000010                  CP$RTS  =       10              ;Request to send
    130         000020                  CP$DTR  =       20              ;Data terminal ready
    131         000140                  CP$CS   =       140             ;Mask for clock source values
    132         000200                  CP$MM   =       200             ;Maintenance mode
    133         000030                  CP$MC0  =       30              ;Standard value for TSX-Plus
    134                                 ;
    135                                 ;  Modem Control Register 1
    136                                 ;
    137         000004                  CP$SMI  =       4               ;Speed mode indicator
    138         000010                  CP$TI   =       10              ;Test indicator
    139         000020                  CP$CD   =       20              ;Carrier detect
    140         000040                  CP$CTS  =       40              ;Clear to send
    141         000100                  CP$RI   =       100             ;Ring indicator
    142         000200                  CP$DSR  =       200             ;Data set ready
```

```
                                .SBTTL  Quad Serial Line Unit control values
                        ;-----------------------------------------------------------------------
                        ;
                        ;  Device ID for quad serial line unit
                        ;
000064                  QP$ID   =       64              ;Device ID for quad serial line unit
                        ;
                        ;  Addresses of registers relative to base address for module
                        ;
                        ;  Read registers
                        ;
000004                  QPRMSR  =       004             ;Module status register
000100                  QPRMRA  =       100             ;Mode register A
000102                  QPRSRA  =       102             ;Status register A
000106                  QPRRRA  =       106             ;Receiver register A
000112                  QPRISR  =       112             ;Interrupt status register
000120                  QPRMRB  =       120             ;Mode register B
000122                  QPRSRB  =       122             ;Status register B
000126                  QPRRRB  =       126             ;Receiver register B
000132                  QPRIP   =       132             ;Input port
                        ;
                        ;  Write registers
                        ;
000102                  QPRCSA  =       102             ;Clock select A
000104                  QPRCRA  =       104             ;Command register A
000106                  QPRTRA  =       106             ;Transmitter register A
000110                  QPRACR  =       110             ;Auxiliary control register
000112                  QPRIMR  =       112             ;Interrupt mask register
000122                  QPRCSB  =       122             ;Clock select B
000126                  QPRTRB  =       126             ;Transmitter register B
000132                  QPROPC  =       132             ;Output port configuration register
000134                  QPRSOP  =       134             ;Set output port bits register
000136                  QPRROP  =       136             ;Reset output port bits register
                        ;
                        ;  Offsets to register addresses by channel and DUART
                        ;
000020                  QPRCOF  =       020             ;Diff between channel A and B registers
000040                  QPRUOF  =       040             ;Diff between DUART 0 and DUART 1 registers
                        ;
                        ;  Control flags for the quad serial line unit.
                        ;
                        ;  Module status register
                        ;
000004                  QP$22C  =       4               ;2/2 configuration
000010                  QP$40C  =       10              ;4/0 configuration
000100                  QP$IEN  =       100             ;Interrupt enable
                        ;
                        ;  Mode register 1
                        ;
000002                  QP$7BR  =       2               ;7 bit characters
000003                  QP$8BR  =       3               ;8 bit characters
000004                  QP$ODD  =       4               ;On ==> Odd parity
000020                  QP$NPR  =       20              ;No parity (off==>Want parity)
000040                  QP$BEM  =       40              ;Block error mode
000100                  QP$SFL  =       100             ;Interrupt when silo full
000200                  QP$RXR  =       200             ;Receiver RTS control
                        ;
```

Quad Serial Line Unit control values

```
                                ;  Mode register 2
                                ;
000007                          QP$1SB  =       7               ;One stop bit
000017                          QP$2SR  =       17              ;Two stop bits
000020                          QP$TCS  =       20              ;Transmitter CTS enable
000040                          QP$TRS  =       40              ;Transmitter RTS enable
                                ;
                                ;  Command register
                                ;
000001                          QP$ERX  =       1               ;Enable receiver
000002                          QP$DRX  =       2               ;Disable receiver
000004                          QP$ETX  =       4               ;Enable transmitter
000010                          QP$DTX  =       10              ;Disable transmitter
                                ;
                                ;  Command values for command register
                                ;
000020                          QP$RPR  =       20              ;Reset MR pointer
000040                          QP$RRX  =       40              ;Reset receiver
000060                          QP$RTX  =       60              ;Reset transmitter
000100                          QP$RES  =       100             ;Reset error status
000120                          QP$RBC  =       120             ;Reset break change interrupt
000140                          QP$SBT  =       140             ;Start break transmission
000160                          QP$EBT  =       160             ;End break transmission
                                ;
                                ;  Status register
                                ;
000001                          QP$RDN  =       1               ;Receiver ready (character received)
000002                          QP$FFL  =       2               ;FIFO silo full
000004                          QP$TDN  =       4               ;Transmitter ready (finished sending char)
000010                          QP$TEM  =       10              ;Transmitter buffer empty
000020                          QP$ROE  =       20              ;Receiver overrun error
000040                          QP$RPE  =       40              ;Received parity error
000100                          QP$RFE  =       100             ;Received framing error
000200                          QP$BRK  =       200             ;Received break
                                ;
                                ;  Auxiliary control register
                                ;
000360                          QP$AWA  =       360             ;Standard value for TSX-Plus
                                ;
                                ;  Interrupt status register
                                ;
000001                          QP$TAR  =       1               ;Transmitter A finished transmission
000002                          QP$RAR  =       2               ;Receiver A has a character
000004                          QP$BCA  =       4               ;Receiver A break status change
000010                          QP$CNR  =       10              ;Counter/timer ready
000020                          QP$TBR  =       20              ;Transmitter B finished transmission
000040                          QP$RBR  =       40              ;Receiver B has a character
000100                          QP$BCB  =       100             ;Receiver B break status change
000200                          QP$IPC  =       200             ;Input port status change
                                ;
                                ;  Interrupt mask register
                                ;
000063                          QP$AWI  =       063             ;Standard value for TSX-Plus
```

```
                                        .SBTTL  PROINI -- General initialization for Pro
                                ;-------------------------------------------------------------------
                                ;  Perform general system initialization for PRO.
                                ;
000112  010146                 PROINI: MOV     R1,-(SP)
                                ;
                                ;  Set up information about which device is installed in each option slot
                                ;
000114  004737  000270'                 CALL    INISLT
                                ;
                                ;  Set up address of video interrupt vector and CSR
                                ;
000120  012701  000074'                 MOV     #VIDTBL,R1              ;Point to table with video ID's
000124  012100                  1$:     MOV     (R1)+,R0                ;Get next possible video device ID
000126  004737  001254'                 CALL    GETSLT                  ;See if we can find this ID
000132  103774                         BCS     1$                      ;Loop if not
000134  010037  000062'                 MOV     R0,VIDSLT               ;Save video option slot number
000140  072027  000003                 ASH     #3,R0                   ;Convert slot # to interrupt address
000144  062700  000300                 ADD     #300,R0
000150  010037  000064'                 MOV     R0,VIDVEC               ;Save address of video int vector
000154  013700  000062'                 MOV     VIDSLT,R0               ;Get video slot #
000160  072027  000007                 ASH     #7,R0                   ;CSR addresses are 200 apart per slot
000164  062700  174000                 ADD     #174000,R0              ;Add address of CSR for slot 0
000170  010037  000000G                MOV     R0,VIDCSR               ;Set CSR address for video
                                ;
                                ;  Store pointers to the routines that get CSR and Vector addresses
                                ;  for PRO devices.
                                ;
000174  012737  001126' 000000G         MOV     #GPVVEC,RPRVEC          ;Routine to get PRO vector addresses
000202  012737  001130' 000000G         MOV     #GPVCSR,RPRCSR          ;Routine to get PRO CSR addresses
                                ;
                                ;  Determine if a terminal is connected to the printer port.
                                ;
000210  113700  177560                  MOVB    @#PP$RCS,R0             ;Get receiver control register
000214  153700  177564                  BISB    @#PP$TCS,R0             ;Combine flags from transmitter
000220  001405                          BEQ     4$                      ;Br if terminal not connected
000222  105237  000000G-                INCB    PPTERM                  ;Remember terminal on printer port
                                ;
                                ;  A terminal is connected to the printer port.
                                ;  Enable BREAK key on diagnostic terminal to enter ODT
                                ;
000226  005727  000000G                 TST     #PROBRK                 ;Is break control wanted?
000232  001004                         BNE     2$                      ;Br if yes
000234  042737  000200  173700  4$:     BIC     #SS$BRK,@#PSSREG        ;Disable break control
000242  000406                         BR      5$
000244  052737  000200  173700  2$:     BIS     #SS$BRK,@#PSSREG        ;Enable break ODT entry
000252  012737  000067  173406          MOV     #PP$CMF!PP$RE,@#PP$CMR  ;Enable printer port
                                ;
                                ;  Perform initialization for quad serial line unit
                                ;
000260                          5$:
                                        .IF     NE,QPASM                ;Do if quad line support wanted
000260  004737  004660'                 CALL    QPINIT                  ;Init quad line unit
                                        .ENDC   ;NE,QPASM
                                ;
                                ;  Finished
                                ;
```

```
     58 000264  012601                    9$:     MOV     (SP)+,R1
     59 000266  000207                            RETURN
```

```
                                        .SBTTL  INISLT -- Set up information about option slots
                                ;-----------------------------------------------------------------------
                                ;  This routine initializes a table that contains the device ID's of
                                ;  devices installed in each of the PRO option slots.
                                ;
000270  010146                 INISLT: MOV     R1,-(SP)
000272  010246                         MOV     R2,-(SP)
                                ;
                                ;  Map KPAR5 to system configuration table
                                ;
000274  005001                         CLR     R1                      ;Get byte value without sign extension
000276  153701  173050                 BISB    @#173050,R1             ;Get 32Kb top of memory block #
000302  072127  000011                 ASH     #9.,R1                  ;Convert to 64 byte block number
000306  162701  000200                 SUB     #200,R1                 ;Get # of last 8Kb block of memory
000312                                 DISABL                          ;;;** Disable interrupts **
000320  013746  0000000                MOV     @#KPAR5,-(SP)           ;;;Save current par 5 mapping
000324  010137  0000000                MOV     R1,@#KPAR5              ;;;Map par5 to configuration table
000330  052737  0000000 0000000        BIS     #MMENBL,@#SROMMR        ;;;Turn on memory management
000336  052737  0000000 0000000        BIS     #EMMAP,@#SR3MMR         ;;;Enable 22 bit addressing
                                ;
                                ;  Get number of option slots
                                ;
000344  012702  0177640                MOV     #VPAR5+17764,R2         ;;;Point to cell with # option slots
000350  011200                         MOV     (R2),R0                 ;;;Get # of option slots
                                ;
                                ;  Set up table in TSX that has device ID number of each option slot
                                ;
000352  012701  0000000                MOV     #PROSLT,R1              ;;;Point to table that will store option ID's
000356  005742                 1$:     TST     -(R2)                   ;;;Skip word with option status value
000360  014221                         MOV     -(R2),(R1)+             ;;;Store option ID code
000362  077003                         SOB     R0,1$                   ;;;Loop to get all option slot device ID's
000364  012711  177777                 MOV     #-1,(R1)                ;;;Store -1 at end of table
                                ;
                                ;  Restore KPAR5 mapping
                                ;
000370  012637  0000000                MOV     (SP)+,@#KPAR5           ;;;Restore par 5 mapping
000374  042737  0000000 0000000        BIC     #MMENBL,@#SROMMR        ;;;Turn off memory mapping
000402                                 ENABL                           ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
000410  012602                         MOV     (SP)+,R2
000412  012601                         MOV     (SP)+,R1
000414  000207                         RETURN
```

```
                                        .SBTTL  PRONOP -- Disable PRO interrupts
                                ;-----------------------------------------------------------------------
                                ;  PRONOP is called during system startup to direct PRO interrupts to
                                ;  an RTI instruction so they will not enter RT-11.
                                ;
000416  010146                  PRONOP: MOV     R1,-(SP)
000420  012700  000466'                 MOV     #DORTI,R0               ;Get address of RTI instruction
000424  010037  000060                  MOV     R0,@#60                 ;Catch keyboard input interrupt
000430  010037  000200                  MOV     R0,@#200                ;380 keyboard input interrupt
000434  010037  000064                  MOV     R0,@#64                 ;End-of-transfer video interrupt
000440  010037  000204                  MOV     R0,@#204                ;Keyboard output interrupt
000444  010037  000230                  MOV     R0,@#230                ;380 clock interrupt
000450  013701  000064'                 MOV     VIDVEC,R1               ;Point to video interrupt vectors
000454  010011                          MOV     R0,(R1)                 ;End-of-frame video interrupt
000456  010061  000004                  MOV     R0,4(R1)                ;end-of-transfer video interrupt
                                ;
                                ;  Finished
                                ;
000462  012601                          MOV     (SP)+,R1
000464  000207                          RETURN
                                ;
                                ;  RTI instruction used to render interrupts harmless
                                ;
000466  000002                  DORTI:  RTI                             ;Return from interrupt immediately
```

```
                                        .SBTTL  PROHAN -- Initialize the PI handler
                                ;-------------------------------------------------------------------
                                ;  PROHAN is called to initialize the PI handler.
                                ;
000470  010146                  PROHAN: MOV     R1,-(SP)
000472  010246                          MOV     R2,-(SP)
000474  010346                          MOV     R3,-(SP)
000476  013746  000000G                 MOV     @#KPAR5,-(SP)   ;Save current kernel PAR5 mapping
000502  013746  000000G                 MOV     @#KPAR6,-(SP)   ;Save current kernel PAR6 mapping
                                ;
                                ;  Set up information about where the PI handler is in memory.
                                ;
000506  012703  000000G                 MOV     #PISRT,R3       ;Point to shared run-time descriptor for PI
000512  016337  000000G 000052'         MOV     RT$BAS(R3),PIBASE;This is base 64-byte block # of run-time
000520  013737  000052' 000000G         MOV     PIBASE,@#KPAR5  ;Map to base of PI handler through PAR 5
                                ;
                                ;  Transfer information between TSXPRO and the PI handler.
                                ;
000526  012701  001000G                 MOV     #VPAR5+1000,R1  ;Point to HOKVEC vector in PI handler
000532  012702  000000'                 MOV     #PROVEC,R2      ;Point to TSXPRO vector
000536  011100                  5$:     MOV     (R1),R0         ;Get address of a cell in PI handler
000540  001404                          BEQ     6$              ;Br if PI wants an address
000542  062700  000000G                 ADD     #VPAR5,R0       ;Bias the address to be in PAR 5 region
000546  010012                          MOV     R0,(R2)         ;Store address pointer in TSXPRO cell
000550  000407                          BR      7$
000552  011211                  6$:     MOV     (R2),(R1)       ;Pass an address to PI handler
000554  001402                          BEQ     8$              ;Br if just hit end of lists
000556  022122                  7$:     CMP     (R1)+,(R2)+     ;Increment both list pointers
000560  000766                          BR      5$              ;Continue processing lists
                                ;
                                ;  Move the PI initialization overlay into the TSINIT work buffer.
                                ;
000562  013701  000000'         8$:     MOV     PIINAD,R1       ;Get address of init overlay
000566  162701  000000G                 SUB     #VPAR5,R1       ;Subtract bias we added
000572  072127  177772                  ASH     #-6,R1          ;Convert to 64-byte block #
000576  063701  000052'                 ADD     PIBASE,R1       ;Get mapping base for init overlay
000602  010137  000000G                 MOV     R1,@#KPAR6      ;Map to overlay through PAR 6
000606  013700  000002'                 MOV     PIINWD,R0       ;Get # words in init overlay
000612  162700  000000G                 SUB     #VPAR5,R0       ;Subtract bias we added
000616  012701  000000G                 MOV     #VPAR6,R1       ;Get virtual address of init overlay
000622  013702  000000G                 MOV     PROBUF,R2       ;Get address of TSINIT work buffer
000626  012122                  4$:     MOV     (R1)+,(R2)+     ;Move init code to work buffer
000630  077002                          SOB     R0,4$
                                ;
                                ;  Now execute the PI initialization code that we have moved to
                                ;  the TSINIT work buffer.
                                ;  Note: PAR5 is now mapped to the resident portion of the PI handler.
                                ;  On entry,
                                ;    R3 = Virtual address of base of PI handler.
                                ;    R1 = 64-byte block number of base of PI handler.
                                ;
000632  013701  000052'                 MOV     PIBASE,R1       ;Get 64-byte block # of PI base
000636  012703  000000G                 MOV     #VPAR5,R3       ;Get virtual address of PI base
000642  004777  000000G                 CALL    @PROBUF         ;Call PI initialization code
                                ;
                                ;  Now get some actual addresses from PI handler
                                ;
```

```
                                        MOV     @#VDCSR,$VDCSR  ;Get video CSR address
                                ;
                                ;  Connect to interrupt vectors
                                ;
                                        MOV     #340,R0         ;Get priority 7 code
                                        MOV     #60,R1          ;Get address of start of vector area
                                        MOV     #PIKIIR,(R1)+   ;Set keyboard input interrupt routine
                                        MOV     R0,(R1)+        ;Set interrupt service priority to 7
                                        MOV     #PIVTIR,(R1)+   ;Set end of transfer video interrupt
                                        MOV     R0,(R1)+        ;Set priority = 7
                                        MOV     #PIKOIR,@#204   ;Set keyboard output interrupt routine
                                        MOV     R0,@#206        ;Set priority
                                ;
                                ;  Connect to 380 interrupt vectors
                                ;
                                        MOV     @#60,@#200      ;380 keyboard input interrupt
                                        MOV     @#62,@#202
                                        MOV     VIDVEC,R1       ;Point to video interrupt vector A
                                        MOV     #PIVFIR,(R1)+   ;Set end-of-frame video interrupt
                                        MOV     R0,(R1)+        ;Set priority
                                        MOV     #PIVTIR,(R1)+   ;Set end-of-transfer video interrupt
                                        MOV     R0,(R1)+        ;Set priority
                                        MOV     @#100,@#230     ;380 clock interrupt
                                        MOV     @#102,@#232
                                ;
                                ;  Connect to interrupt for quad serial line unit
                                ;
                                        .IF     NE,QPASM        ;Assemble if quad line unit support wanted
                                        MOV     QPVEC,R3        ;Get address of interrupt vector
                                        BEQ     10$             ;Br if quad serial unit not installed
                                        MOV     #QPCINT,(R3)+   ;Set PC for interrupt
                                        MOV     R0,(R3)         ;Set PS for interrupt
                                        MOV     QPCSR,R3        ;Get address of CSR register
                                        BISB    #QP$IEN,QPRMSR(R3) ;Enable interrupts
                                        .ENDC   ;NE,QPASM
                                ;
                                ;  Finished
                                ;
                                10$:    MOV     (SP)+,@#KPAR6   ;Restore kernel mapping
                                        MOV     (SP)+,@#KPAR5
                                        MOV     (SP)+,R3
                                        MOV     (SP)+,R2
                                        MOV     (SP)+,R1
                                        RETURN
```

```
                                        .SBTTL  PROLIN -- See if line is a special PRO terminal
                                ;-----------------------------------------------------------------------
                                ;  PROLIN is called once for each DL11 type line defined in the system.
                                ;  It determines if the line is a printer port or communications port
                                ;  line and if so performs the initialization for the line.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> This is a printer port or comm port line.
                                ;   C-flag set     ==> This is not a printer port or comm port line.
                                ;
001030                          PROLIN:
                                ;
                                ;  See if this is a line connected to the quad serial line unit
                                ;
                                        .IF     NE,QPASM        ;Assemble for quad line support
001030  005761  0000000         TST     LMXNUM(R1)      ;Is this line on a mux?
001034  001403                  BEQ     3$              ;Br if not
001036  004737  005136'         CALL    QPLINE          ;Initialize line on quad unit
001042  000427                  BR      9$
                                        .ENDC   ;NE,QPASM
                                ;
                                ;  See if this is the Professional video console
                                ;
001044  026127  0000000 000060  3$:     CMP     INVEC(R1),#60   ;Is this the console line?
001052  001003                  BNE     2$              ;Br if not console line
001054  004737  001316'         CALL    PIINIT          ;Initialize PI line
001060  000420                  BR      9$
                                ;
                                ;  Get address of receiver status register for the line
                                ;
001062  016100  0000000         2$:     MOV     RSR(R1),R0      ;Get address of receiver status register
                                ;
                                ;  See if this is the printer port
                                ;
001066  020027  173400          CMP     R0,#PP$DBR      ;Printer port?
001072  001003                  BNE     1$              ;Br if not
001074  004737  002324'         CALL    PPINIT          ;Initialize the printer port
001100  000410                  BR      9$
                                ;
                                ;  See if this is the communications port
                                ;
001102  020027  173300          1$:     CMP     R0,#CP$DBR      ;Communications port?
001106  001003                  BNE     8$              ;Br if not
001110  004737  003146'         CALL    CPINIT          ;Initialize the comm port
001114  000402                  BR      9$
                                ;
                                ;  This line is not the printer port or the comm port
                                ;
001116  000261                  8$:     SEC                     ;Signal failure on return
001120  000401                  BR      10$
                                ;
                                ;  This line was the printer port or the comm port
                                ;
001122  000241                  9$:     CLC                     ;Signal success on return
```

```
                                        ;
                                        ;  Finished
                                        ;
001124  00020/                          10$:    RETURN
```

```
                                ;
                                ;  Branch vector to entry points of PRO CSR/Vector/Slot routines
                                ;
001126  000402                  GPVVEC: BR      GPRVEC          ;Get vector address
001130  000420                  GPVCSR: BR      GPRCSR          ;Get CSR address
001132  000436                  GPVSLT: BR      GPRSLT          ;Get Slot value

                                        .SBTTL  GPRVEC -- Get address of vector for PRO device
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to determine the address of the interrupt vector
                                ;  for a PRO device based on the device ID.
                                ;
                                ;  Inputs:
                                ;   Device ID is pushed on stack before calling GPRVEC.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found device in tables; C-flag set ==> No such device.
                                ;   Top of stack contains vector for device.
                                ;
001134  010046                  GPRVEC: MOV     R0,-(SP)
001136  016600  000004                  MOV     4(SP),R0        ;Get the device ID code
001142  004737  001254'                 CALL    GETSLT          ;Determine which option slot has the dev
001146  103407                          BCS     9$              ;Br if device not recognized
001150  072027  000003                  ASH     #3,R0           ;Vectors are 8 bytes apart per slot
001154  062700  000300                  ADD     #300,R0         ;Vectors for slot 0 start at 300
001160  000241                          CLC                     ;Signal success on return
001162  010066  000004                  MOV     R0,4(SP)        ;Place vector address on stack
001166  012600                  9$:     MOV     (SP)+,R0
001170  000207                          RETURN
```

```
                                        .SBTTL  GPRCSR -- Get address of CSR for PRO device
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to determine the address of the CSR
                                ;  for a PRO device based on the device ID.
                                ;
                                ;  Inputs:
                                ;   Device ID is pushed on stack before calling GPRCSR.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found device in tables; C-flag set ==> No such device.
                                ;   Top of stack contains address of CSR for device.
                                ;
001172  010046                  GPRCSR: MOV     R0,-(SP)
001174  016600  000004                  MOV     4(SP),R0        ;Get the device ID code
001200  004737  001254'                 CALL    GETSLT          ;Determine which option slot has the dev
001204  103407                          BCS     9$              ;Br if device not recognized
001206  072027  000007                  ASH     #7.,R0          ;CSR address are 200 apart per slot
001212  062700  174000                  ADD     #174000,R0      ;CSR for slot 0 starts at 174000
001216  000241                          CLC                     ;Signal success on return
001220  010066  000004                  MOV     R0,4(SP)        ;Place CSR address on stack
001224  012600                  9$:     MOV     (SP)+,R0
001226  000207                          RETURN
```

```
                                        .SBTTL  GPRSLT -- Get Slot number for PRO device
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to determine the slot number
                                ;  for a PRO device based on the device ID.
                                ;
                                ;  Inputs:
                                ;   Device ID is pushed on stack before calling GPRSLT.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found device in tables; C-flag set ==> No such device.
                                ;   Top of stack contains slot number for device.
                                ;
001230  010046                 GPRSLT: MOV     R0,-(SP)
001232  016600  000004                 MOV     4(SP),R0        ;Get the device ID code
001236  004737  001254'                CALL    GETSLT          ;Determine which option slot has the dev
001242  103402                         BCS     9$              ;Br if device not recognized
001244  010066  000004                 MOV     R0,4(SP)        ;Put slot number back onto stack
001250  012600                 9$:     MOV     (SP)+,R0
001252  000207                         RETURN
```

```
                                        .SBTTL  GETSLT -- Determine which option slot has device controller
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to determine which PRO option slot contains the
                                ;  controller for a specified device.
                                ;
                                ;  Inputs:
                                ;   R0 = Device ID code.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found the device; C-flag set ==> No such device
                                ;   R0 = Slot number.
                                ;
001254  010146                  GETSLT: MOV     R1,-(SP)
001256  010001                          MOV     R0,R1           ;Get device ID code
001260  012700  0000000                 MOV     #PROSLT,R0      ;Point to table with device ID codes
001264  021027  177777          1$:     CMP     (R0),#-1        ;Reached end of table entries?
001270  001407                          BEQ     2$              ;Br if yes
001272  020120                          CMP     R1,(R0)+        ;Search for device ID in table
001274  001373                          BNE     1$              ;Loop if not found
001276  162700  0000020                 SUB     #PROSLT+2,R0    ;Compute table offset of entry
001302  006200                          ASR     R0              ;Convert to word index
001304  000241                          CLC                     ;Signal success on return
001306  000401                          BR      9$
001310  000261                  2$:     SEC                     ;Signal failure on return
001312  012601                  9$:     MOV     (SP)+,R1
001314  000207                          RETURN
```

```
                        .SBTTL  ***  Console Control Routines  ***
                        .SBTTL  PIINIT -- Initialize the console
                ;-----------------------------------------------------------------------
                ;  PIINIT is called during system initialization to initialize the
                ;  console.
                ;
                ;  Inputs:
                ;   R1 = Line index number.
                ;
001316          PIINIT:
                ;
                ;  Remember index number of line connected to console
                ;
001316  010137  000044'                 MOV     R1,PILINE       ;Save line index # of console line
001322  012761  0000000 0000000        MOV     #CDX$PI,LCDTYP(R1);Set type of communications controller
                ;
                ;  Set addresses of communication port device-dependent routines
                ;
001330  012737  001354' 0000000        MOV     #PISTRT,CDSTRT+CDX$PI ;Set address of start routine
                ;
                ;  Default terminal type to VT100
                ;
001336  005761  0000000                TST     ITRMTP(R1)      ;Is a terminal type specified?
001342  001003                         BNE     1$              ;Br if yes
001344  012761  0000000 0000000        MOV     #VT100,ITRMTP(R1);Set default terminal type to VT100
                ;
                ;  Finished
                ;
001352  000207          1$:            RETURN
```

```
                                        .SBTTL  PISTRT -- Start output to video screen
                                ;-------------------------------------------------------------------
                                ;  PISTRT is called after a character is placed in the output ring
                                ;  buffer to initiate transmission to the video screen
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number of line being started.
                                ;
001354                          PISTRT:
                                ;
                                ;  See if we have already started transmitter during this clock period
                                ;
001354  005237  000056'               INC     PISCNT          ;Have we already started during clk interval?
001360  003005                        BGT     9$              ;Br if yes
                                ;
                                ;  Do not allow output to the console to consume the entire system.
                                ;
001362  004737  002000'               CALL    PIQUIT          ;Is it time to suspend output?
001366  103402                        BCS     9$              ;Br if yes
                                ;
                                ;  Start output
                                ;
001370  004737  001420'               CALL    PIGO            ;Start transmitter
                                ;
                                ;  Finished
                                ;
001374  000207                 9$:     RETURN

                                        .SBTTL  PIDRIV -- Clock driven PI transmitter routine
                                ;-------------------------------------------------------------------
                                ;  PIDRIV is called periodically from the clock interrupt routine
                                ;  to force output to the PI console.
                                ;
001376                          PIDRIV:
                                ;
                                ;  Reset output character suspend count
                                ;
001376  012737  000017  000072'       MOV     #MAXPOC,PIMOC   ;Reset output character count
                                ;
                                ;  Start output to PI
                                ;
001404  004737  001420'               CALL    PIGO            ;Start the transmitter
                                ;
                                ;  Reset flag saying if transmitter has been started during clock interval
                                ;
001410  012737  177777  000056'       MOV     #-1,PISCNT      ;Reset counter
                                ;
                                ;  Finished
                                ;
001416  000207                        RETURN
```

```
                                        .SBTTL  PIGO   -- Start PI transmitter
                                ;-------------------------------------------------------------------
                                ;  PIGO is called to enable a transfer-done interrupt which will start
                                ;  transmission to the PI console.
                                ;
001420  013746  000000G         PIGO:   MOV     @#KPAR5,-(SP)   ;Save current PAR5 mapping
001424  013737  000052' 000000G         MOV     PIBASE,@#KPAR5  ;Set up PAR5 to map to PI
                                ;
                                ;  See if output to the PI handler has been suspended
                                ;
001432  005777  176370                  TST     @PIGOFL         ;Has PI output been suspended?
001436  001013                          BNE     1$              ;Br if not
                                ;
                                ;  Output to the PI handler has been suspended.
                                ;  Consume all pending output characters and then return.
                                ;
001440  010446                          MOV     R4,-(SP)
001442  013704  000044'                 MOV     PILINE,R4       ;Get line index number
001446                                  DISABL                  ;;;** Disable interrupts **
001454  004774  000000G         4$:     CALL    @LOUTIR(R4)     ;;;Get next char for PI
001460  103375                          BCC     4$              ;;;Loop till all characters consumed
001462  012604                          MOV     (SP)+,R4        ;;;
001464  000427                          BR      9$              ;;;
                                ;
                                ;  Output to PI handler is not suspended
                                ;
001466                          1$:     DISABL                  ;;;** Disable interrupts **
                                ;
                                ;  See if we have already triggered a PI interrupt
                                ;
001474  005737  000054'                 TST     PIOIFL          ;;;Have we already triggered an output int?
001500  001021                          BNE     9$              ;;;Br if output interrupt active now
001502  005237  000054'                 INC     PIOIFL          ;;;Set flag saying output int active
001506  010677  176312                  MOV     SP,@VDFLAG      ;;;Tell VDCURS that buffer is not empty
001512  032777  040000  176276         BIT     #VP$DOI,@$VDCSR ;;;Is end-of-transfer interrupt enabled?
001520  001011                          BNE     9$              ;;;Br if yes
001522  005777  176274                  TST     @PRIOO          ;;;Has video said to leave it alone?
001526  001003                          BNE     2$              ;;;Br if yes
001530  052777  040000  176260         BIS     #VP$DOI,@$VDCSR ;;;Generate end-of-transfer interrupt
001536  012777  100000  176254 2$:     MOV     #100000,@REENAB ;;;Tell video to turn on interrupts when ready
                                ;
                                ;  Finished
                                ;
001544  012637  000000G         9$:     MOV     (SP)+,@#KPAR5   ;;;Restore PAR5 mapping
001550                                  ENABL                   ;;;** Enable interrupts **
001556  000207                          RETURN
```

```
                                        .SBTTL  PIVTIR -- Video end of transfer interrupt
                                ;-----------------------------------------------------------------------
                                ;  PIVTIR is jumped to from the video end-of-transfer interrupt.
                                ;
001560                          PIVTIR:
                                ;
                                ;  Do .INTEN to drop priority to 4
                                ;
001560  004537  000000G               JSR     R5,INTEN        ;Do standard interrupt entry
001564  000140                        .WORD   140             ;Priority = 4
001566  010046                        MOV     R0,-(SP)        ;Save R0
                                ;
                                ;  Set flag saying output interrupt is in progress
                                ;
001570  005237  000054'               INC     PIOIFL          ;Doing output interrupt processing
                                ;
                                ;  Disable transfer done interrupt
                                ;
001574  013737  000052' 000000G       MOV     PIBASE,@#KPAR5  ;Map kernel PAR 5 to the PI handler
001602  012777  100000  176210        MOV     #100000,@REENAB ;Set flag to reenable transfer-done int
001610  042777  040000  176200        BIC     #VP$DOI,@$VDCSR ;Disable transfer-done interrupt
                                ;
                                ;  Limit the number of consecutive characters sent to the console
                                ;  to prevent the PI output processing (which is compute bound) from
                                ;  consuming 100% of the machine.
                                ;
001616  013704  000044'               MOV     PILINE,R4       ;Get line index number of console line
001622  004737  002000'               CALL    PIQUIT          ;Time to suspend output?
001626  103431                        BCS     2$              ;Br if yes
                                ;
                                ;  Get the next character to transmit.
                                ;
001630  004774  000000G               CALL    @LOUTIR(R4)     ;Get next char for line
001634  103423                        BCS     1$              ;Br if no character available
001636  010005                        MOV     R0,R5           ;Get char to transmit to R5
001640  012600                        MOV     (SP)+,R0        ;Restore R0 before we fork
                                ;
                                ;  There is another character to transmit.  Now fork
                                ;
001642  004537  000000G               JSR     R5,FORK         ;Fork to get to priority 0
001646  000000G                       .WORD   FP$PIO          ;Fork priority
001650  010046                        MOV     R0,-(SP)        ;Save R0
                                ;
                                ;  Map PAR5 to the PI handler
                                ;
001652  013737  000052' 000000G       MOV     PIBASE,@#KPAR5  ;Map kernel PAR 5 to the PI handler
                                ;
                                ;  Enter PI handler to transmit the character
                                ;
001660  010500                        MOV     R5,R0           ;Get char to transmit
001662  052764  000000G 000000G       BIS     #$XCHAR,LSW3(R4);Set transmitter busy flag
001670  042764  000000G 000000G       BIC     #$OITIM,LSW5(R4);Start output interrupt timer
001676  004777  176102                CALL    @PIXOCH         ;Call routine to transmit the character
001702  000412                        BR      9$              ;Finished
                                ;
                                ;  No character is available for transmission
                                ;
```

```
     58 001704  012737  000017  000072' 1$:     MOV     #MAXPOC,PIMDC   ;Reset output char limit count
     59 001712  042764  000000G 000000G 2$:     BIC     #$XCHAR,LSW3(R4);Say transmitter is not busy
     60 001720  005077  176100                   CLR     @VDFLAG         ;Tell VDCURS that buffer is empty
     61 001724  005077  176070                   CLR     @REENAB         ;Tell video not to turn on interrupts
     62                                ;
     63                                ;  Finished
     64                                ;
     65 001730  005037  000054'        9$:     CLR     PIOIFL          ;Say output interrupt not in progress
     66 001734  012600                          MOV     (SP)+,R0
     67 001736  000207                          RETURN
```

```
                                        .SBTTL  PINDCH -- Get next output character for console
                                ;-----------------------------------------------------------------------
                                ;  PINDCH is called from within the VIDEO handler to try to get another
                                ;  character to send to the console terminal.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> a character is available.
                                ;   C-flag set     ==> No more characters available.
                                ;   R0 = Character to send if C-flag is cleared.
                                ;
001740  010446                  PINDCH: MOV     R4,-(SP)
001742  013704  000044'                 MOV     PILINE,R4       ;Get line index number of console
                                ;
                                ;  Limit the number of consecutive output characters to prevent the
                                ;  P1 output processing (which is compute bound) from completely
                                ;  dominating the system.
                                ;
001746  004737  002000'                 CALL    PIQUIT          ;Time to suspend output?
001752  103410                          BCS     9$              ;Br if yes
                                ;
                                ;  Call line-dependent routine to try to get another output character
                                ;
001754                          1$:     DISABL                  ;;;** Disable interrupts for LOUTIR **
001762  004774  0000000                 CALL    @LOUTIR(R4)     ;;;Get next character for line
001766                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
001774  012604                  9$:     MOV     (SP)+,R4
001776  000207                          RETURN
```

```
                                        .SBTTL  PIQUIT -- Check output character limit
                                ;-----------------------------------------------------------------------
                                ;  This routine causes output to the console terminal to be suspended
                                ;  after a certain number of characters have been transmitted.
                                ;  The output is resumed after a timed interval.
                                ;  This is done to prevent the console terminal output processing (which
                                ;  is compute bound) from consuming 100% of the machine.
                                ;
                                ;  Outputs:
                                ;   C-flag set     ==> Suspend output.
                                ;   C-flag cleared ==> Continue output.
                                ;
002000                          PIQUIT:
                                ;
                                ;  See if there is any other pending system activity
                                ;
002000  105737  0000000                 TSTB    DOSCHD          ;Scheduler cycle needed?
002004  001013                         BNE     2$              ;Br if yes
002006  005737  0000000                 TST     FRKCQE          ;Fork requests pending?
002012  001010                         BNE     2$              ;Br if yes
002014  105737  0000000                 TSTB    CORUSR          ;Is there another executing job?
002020  001412                         BEQ     1$              ;Br if not
002022  013700  000044'                 MOV     PILINE,R0       ;Get # of job using console terminal
002026  005760  0000000                 TST     LXCL(R0)        ;Is that job cross-connected to CL line?
002032  002405                         BLT     1$              ;Br if not
                                ;
                                ;  There is some pending system activity.
                                ;  See if it is time to suspend PI output.
                                ;
002034  005337  000072'        2$:     DEC     PIMOC           ;Time to suspend output?
002040  003002                         BGT     1$              ;Br if not
002042  000261                         SEC                     ;Signal to suspend
002044  000401                         BR      9$
002046  000241                 1$:     CLC                     ;Signal to continue output
                                ;
                                ;  Finished
                                ;
002050  000207                 9$:     RETURN
```

```
                                        .SBTTL  PIVFIR -- Video end of frame interrupt
                                ;-----------------------------------------------------------------------
                                ;  PIVFIR is jumped to from the interrupt vector whenever a video
                                ;  end-of-frame interrupt occurs.
                                ;
002052                          PIVFIR:
                                ;
                                ;  Do .INTEN to drop priority to 4
                                ;
002052  004537  000000G                 JSR     R5,INTEN        ;Do standard interrupt entry
002056  000140                         .WORD   140             ;Priority = 4
002060  010046                         MOV     R0,-(SP)
002062  010146                         MOV     R1,-(SP)
002064  010246                         MOV     R2,-(SP)
002066  010346                         MOV     R3,-(SP)
                                ;
                                ;  Enter overlay to do the actual processing
                                ;
002070  013737  000052' 000000G         MOV     PIBASE,@#KPAR5  ;Map kernel PAR 5 to the PI handler
002076  004777  175704                  CALL    @PIXEOF         ;Do end of frame processing
                                ;
                                ;  Finished
                                ;
002102  012603                         MOV     (SP)+,R3
002104  012602                         MOV     (SP)+,R2
002106  012601                         MOV     (SP)+,R1
002110  012600                         MOV     (SP)+,R0
002112  000207                         RETURN
```

```
                                        .SBTTL  PIKIIR -- Keyboard input interrupt
                                ;-----------------------------------------------------------------------
                                ;  PIKIIR is vectored to when a keyboard input interrupt occurs.
                                ;
002114                          PIKIIR:
                                ;
                                ;  Do .INTEN to drop priority to 4
                                ;
002114  004537  0000000G                JSR     R5,INTEN        ;Do standard interrupt entry
002120  000140                          .WORD   140             ;Priority = 4
002122  010046                          MOV     R0,-(SP)
002124  010146                          MOV     R1,-(SP)
002126  010246                          MOV     R2,-(SP)
002130  010346                          MOV     R3,-(SP)
                                ;
                                ;  Enter system overlay to do the actual processing
                                ;
002132  013737  000052' 0000000G        MOV     PIBASE,@#KPAR5  ;Map kernel PAR 5 to the PI handler
002140  004777  175644                  CALL    @PIXICH         ;Process the input character
                                ;
                                ;  Finished
                                ;
002144  012603                          MOV     (SP)+,R3
002146  012602                          MOV     (SP)+,R2
002150  012601                          MOV     (SP)+,R1
002152  012600                          MOV     (SP)+,R0
002154  000207                          RETURN

                                        .SBTTL  KBDCHR -- Process character received from keyboard
                                ;-----------------------------------------------------------------------
                                ;  KBDCHR is jumped to from the PI handler when a character has been
                                ;  received and converted to ASCII.
                                ;
                                ;  Inputs:
                                ;   R5 = Received ascii character.
                                ;
002156                          KBDCHR:
                                ;
                                ;  Save the PAR5 and PAR6 mapping for the handler, and then call
                                ;  the TTINPT routine to process the received character.
                                ;
002156  013746  0000000G                MOV     @#KPAR5,-(SP)   ;Save PAR5 and PAR6 mapping for PI
002162  013746  0000000G                MOV     @#KPAR6,-(SP)
002166  010446                          MOV     R4,-(SP)
002170  013704  000044'                 MOV     PILINE,R4       ;Get line index # for console line
002174  004777  0000000G                CALL    @TTINPT         ;Process received character
                                ;
                                ;  Now restore PAR5 and PAR6 mapping and return to PI
                                ;
002200  012604                          MOV     (SP)+,R4
002202  012637  0000000G                MOV     (SP)+,@#KPAR6   ;Restore PAR5 and PAR6
002206  012637  0000000G                MOV     (SP)+,@#KPAR5
002212  000207                          RETURN                  ;Return to PI
```

```
                                        .SBTTL  PIKOIR -- Output interrupt for keyboard
                                ;-----------------------------------------------------------------------
                                ;  PIKOIR is vectored to when we receive an interrupt on completion of
                                ;  transmitting a character to the keyboard.
                                ;
002214                          PIKOIR:
                                ;
                                ;  Do .INTEN to drop priority to 4
                                ;
002214  004537  000000G                 JSR     R5,INTEN        ;Do standard interrupt entry
002220  000140                          .WORD   140             ;Priority = 4
002222  010046                          MOV     R0,-(SP)
002224  010146                          MOV     R1,-(SP)
002226  010246                          MOV     R2,-(SP)
002230  010346                          MOV     R3,-(SP)
                                ;
                                ;  Call overlay routine to do the processing
                                ;
002232  013737  000052' 000000G         MOV     PIBASE,@#KPAR5  ;Map kernel PAR 5 to the PI handler
002240  004777  175546                  CALL    @PIXIOI         ;Call overlay processing routine
                                ;
                                ;  Finished
                                ;
002244  012603                          MOV     (SP)+,R3
002246  012602                          MOV     (SP)+,R2
002250  012601                          MOV     (SP)+,R1
002252  012600                          MOV     (SP)+,R0
002254  000207                          RETURN
```

```
                                        .SBTTL  PIHAN  -- Simulated PI handler
                                ;-------------------------------------------------------------------
                                ;  The following routine is a simulated handler for the PI device.
                                ;  It is used to allow SETUP to perform .SPFUN operations on
                                ;  the PI device.
                                ;
                                ;  Simulated handler header
                                ;
002256  000000                  PIHAN:  .WORD   0               ;Device vector
002260  000014                          .WORD   PIINT-.         ;Offset to interrupt service entry point
002262  000000                          .WORD   0               ;Interrupt priority
002264  000000                  PILQE:  .WORD   0               ;Last queue element
002266  000000                  PICQE:  .WORD   0               ;Current queue element
002270  000402                          BR      PIENTR          ;Enter here from system I/O queueing
                                ;
                                ;  Simulated interrupt point and abort entry point
                                ;
002272  000207                          RETURN                  ;Abort entry point
002274  000002                  PIINT:  RTI                     ;Fake interrupt entry point
                                ;
                                ;  Process an I/O queue request directed to the PI device
                                ;
002276                          PIENTR:
                                ;
                                ;  Map PAR 5 to the real PI handler
                                ;
002276  013737  000052' 000000G         MOV     PIBASE,@#KPAR5  ;Map PAR 5 to PI handler
                                ;
                                ;  Call real PI handler to process the .SPFUN
                                ;
002304  013700  002266'                 MOV     PICQE,R0        ;Pass address of I/O queue element in R0
002310  004777  175500                  CALL    @PIXIOQ         ;Enter handler to process the .SPFUN
                                ;
                                ;  Finished processing the I/O queue request
                                ;
002314  012704  002266'                 MOV     #PICQE,R4       ;Set up R4 for IOFIN
002320  000137  000000G                 JMP     IOFIN           ;I/O operation is completed
```

```
                                        .SBTTL  *** Printer Port Control Routines  ***
                                        .SBTTL  PPINIT -- Initialize the printer port
                                ;-----------------------------------------------------------------------
                                ;  PPINIT is called during system initialization to initialize the
                                ;  printer port.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
002324  010246                  PPINIT: MOV     R2,-(SP)
                                ;
                                ;  Remember the line index number of the line connected to the printer port
                                ;
002326  010137  000046'                 MOV     R1,PPLINE       ;Remember which line connected to printer port
002332  012761  000000G 000000G         MOV     #CDX$PP,LCDTYP(R1) ;Set type of communications controller
                                ;
                                ;  Set addresses of printer port routines in vector of device-dependent
                                ;  routines.
                                ;
002340  012737  002620' 000000C         MOV     #PPSTRT,CDSTRT+CDX$PP ;Set address of start routine
002346  012737  002772' 000000C         MOV     #PPGDSS,CDGDSS+CDX$PP ;Address of rtn to get data set status
002354  012737  002776' 000000C         MOV     #PPSBRK,CDSBRK+CDX$PP ;Address of rtn to control break
002362  012737  003024' 000000C         MOV     #PPSSPD,CDSSPD+CDX$PP ;Address of rtn to set speed
                                ;
                                ;  Connect the printer port interrupts to the service routine
                                ;
002370  012737  002654' 000220          MOV     #PPRINT,@#PP$RCV;Connect receiver interrupt
002376  012737  000340  000222          MOV     #340,@#PP$RCV+2 ;Set prio = 7 on interrupt entry
002404  012737  002516' 000224          MOV     #PPTINT,@#PP$TRV;Connect transmitter interrupt
002412  012737  000340  000226          MOV     #340,@#PP$TRV+2 ;Set prio 7
                                ;
                                ;  Set up mode register 1
                                ;
002420  005737  173406          3$:     TST     @#PP$CMR        ;Access command reg to reset mode reg ptr
002424  112737  000102  173404          MOVB    #PP$M1F,@#PP$MDR;Initialize values in mode register 1
                                ;
                                ;  Set speed in mode register 2
                                ;
002432  116100  0000010                 MOVB    LMXPRM+1(R1),R0 ;Get line parameter values
002436  001007                          BNE     1$              ;Br if parameter values were specified
002440  012700  0000000                 MOV     #S9600,R0       ;Default to 9600 baud
002444  105737  0000000                 TSTB    PPTERM          ;Is a terminal connected to printer port?
002450  001002                          BNE     1$              ;Br if yes
002452  012700  0000000                 MOV     #S4800,R0       ;Default to 4800 baud
002456  004737  003024'          1$:     CALL    PPSSPD          ;Set the speed in mode register 2
                                ;
                                ;  Initialize the command register
                                ;
002462  012737  000067  173406          MOV     #PP$CMF!PP$RE,@#PP$CMR ;Initialize command register
                                ;
                                ;  Enable receiver interrupts
                                ;
002470  112737  000035  173202          MOVB    #IM$CRM!IM$PPR,@#ICOCSR ;Clear IRR&IMR flags for receiver
                                ;
                                ;  Disable transmitter (set mask bit in IMR), but raise a transmitter
                                ;  request so we will get an interrupt by clearing the mask flag later.
                                ;
```

PPINIT -- Initialize the printer port

```
     58 002476  112737  000076  173202          MOVB    #IM$SM!IM$PPT,@#ICOCSR ;Set interrupt mask flag in IMR
     59 002504  112737  000136  173202          MOVB    #IM$SR!IM$PPT,@#ICOCSR ;Raise interrupt request flag in IRR
     60                                ;
     61                                ;  Finished
     62                                ;
     63 002512  012602                 9$:     MOV     (SP)+,R2
     64 002514  000207                         RETURN
```

```
                        .SBTTL  PPTINT -- Transmitter interrupt
                ;-----------------------------------------------------------------------
                ;  Interrupt service routine for printer port transmitter interrupts.
                ;
002516  010046          PPTINT: MOV     R0,-(SP)
002520  010446                  MOV     R4,-(SP)
002522  013746  000000G         MOV     INTPRI,-(SP)    ;Save current interrupt priority
002526  005037  000000G         CLR     INTPRI          ;Say running priority = 7
                ;
                ;  Disable interrupts from transmitter
                ;
002532  112737  000076  173202      MOVB    #IM$SM!IM$PPT,@#ICOCSR ;Set interrupt mask for transmitter
                ;
                ;  Get number of line connected to printer port
                ;
002540  013704  000046'             MOV     PPLINE,R4       ;Get index # of line connected to printer
                ;
                ;  Say line is no longer transmitting a character
                ;
002544  042764  000000G 000000G     BIC     #$XCHAR,LSW3(R4);Say line no longer transmitting a character
                ;
                ;  Get next character to transmit
                ;
002552  004774  000000G             CALL    @LOUTIR(R4)     ;Get next character to transmit
002556  103413                      BCS     1$              ;Br if no character available
                ;
                ;  Transmit the character contained in R0
                ;
002560  052764  000000G 000000G     BIS     #$XCHAR,LSW3(R4);Set transmitter busy flag
002566  042764  000000G 000000G     BIC     #$OITIM,LSW5(R4);Start timer to catch lost interrupts
002574  110037  173400              MOVB    R0,@#PP$DBR     ;Transmit the character
                ;
                ;  Enable transmitter interrupts
                ;
002600  112737  000056  173202      MOVB    #IM$CM!IM$PPT,@#ICOCSR ;Clear interrupt mask flag
                ;
                ;  Finished.  Return from interrupt
                ;
002606  012637  000000G     1$:     MOV     (SP)+,INTPRI    ;Restore interrupt priority
002612  012604                      MOV     (SP)+,R4
002614  012600                      MOV     (SP)+,R0
002616  000002                      RTI
```

```
                                        .SBTTL  PPSTRT -- Start output to printer port
                                ;
                                ;  Subroutine called to attempt to start output for the printer port.
                                ;  Enable transmitter interrupt.
                                ;  R1 = Physical line number.
                                ;
002620  032761  0000000 0000000 PPSTRT: BIT     #$XCHAR,LSW3(R1)        ;Is the transmitter busy now?
002626  001011                          BNE     1$                      ;Br if yes
002630  112737  000076  173202          MOVB    #IM$SM!IM$PPT,@#ICOCSR  ;Set interrupt mask bit
002636  112737  000136  173202          MOVB    #IM$SR!IM$PPT,@#ICOCSR  ;Set interrupt request flag
002644  112737  000056  173202          MOVB    #IM$CM!IM$PPT,@#ICOCSR  ;Clear interrupt mask flag for xmitter
002652  000207                  1$:     RETURN
```

```
                        .SBTTL  PPRINT -- Receiver interrupt
                ;-----------------------------------------------------------------------
                ;  Interrupt service routine for printer port receiver interrupt.
                ;
002654          PPRINT:
                ;
                ;  Call routine to save R0, R1, R4, and R5.
                ;
002654  004577  000000G         JSR     R5,@TTRSAV      ;Save some registers
                ;
                ;  R0, R1, R4 and R5 are now available.
                ;  Get number of line connected to printer port.
                ;
002660  013704  000046'         MOV     PPLINE,R4       ;Get line index number
                ;
                ;  Get the character from the receiver
                ;
002664  113705  173400          MOVB    @#PP$DBR,R5     ;Get the received character
002670  042705  177400          BIC     #^C<377>,R5     ;Kill sign extension
                ;
                ;  Set flags in R5 to simulate status flags returned by DL11 receiver
                ;
002674  032737  000010  173402  1$:     BIT     #PP$PE,@#PP$STR ;Was a parity error detected?
002702  001402                  BEQ     1$              ;Br if not
002704  052705  000000C         BIS     #RBERR!RCVPAR,R5;Set parity error flag
002710  032737  000040  173402  2$:     BIT     #PP$FE,@#PP$STR ;Was a framing error detected?
002716  001402                  BEQ     2$              ;Br if not
002720  052705  000000C         BIS     #RBERR!FRMERR,R5;Set framing error flag
002724  032737  000020  173402  3$:     BIT     #PP$OE,@#PP$STR ;Did an overrun error occur?
002732  001402                  BEQ     3$              ;Br if not
002734  052705  000000C         BIS     #RBERR!OVRRUN,R5;Set overrun error flag
                ;
                ;  Clear error flags in receiver
                ;
002740  032705  0000000         4$:     BIT     #RBERR,R5       ;Were any errors detected?
002744  001403                  BEQ     5$              ;Br if not
002746  052737  000020  173406  5$:     BIS     #PP$RE,@#PP$CMR ;Clear error flags
                ;
                ;  Call TTINPT to process the character we received
                ;
002754  004777  0000000         6$:     CALL    @TTINPT         ;Enter character input routine
002760  000403                  BR      9$              ;Return from interrupt
                ;
                ;  We got an interrupt but no character was available
                ;
002762  112737  000115  173202  8$:     MOVB    #IM$CR!IM$PPR,@#ICOCSR ;Clear interrupt request flag
                ;
                ;  Return from interrupt
                ;
002770  000207                  9$:     RETURN                  ;Return from interrupt
```

```
                                        .SBTTL  PPGDSS -- Get data set status for printer port
                                ;-----------------------------------------------------------------------
                                ;  This is the device-dependent routine for the printer port which
                                ;  returns generic flags indicating the status of the data set for
                                ;  this line.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number
                                ;
                                ;  Outputs:
                                ;   R0 = Generic status flags
                                ;
002772  005000                  PPGDSS: CLR     R0              ;Say no ring, carrier, or DTR
                                ;
                                ;  Finished
                                ;
002774  000207                          RETURN
```

```
                                        .SBTTL  PPSBRK -- Control break transmission
                                ;-----------------------------------------------------------------------
                                ;  PPSBRK is called to start or stop transmitting a break character
                                ;  to the printer port.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;   R0 = Break control flag (MS$BRK)
                                ;
002776                          PPSBRK:
                                ;
                                ;  See if we are to start or stop sending a break
                                ;
002776  032700  0000000                 BIT     #MS$BRK,R0      ;Start or stop sending break?
003002  001404                          BEQ     1$              ;Br to stop
                                ;
                                ;  Start sending a break character
                                ;
003004  152737  000010  173406          BISB    #PP$FB,@#PP$CMR ;Start sending a break
003012  000403                          BR      9$
                                ;
                                ;  Stop sending a break character
                                ;
003014  142737  000010  173406  1$:     BICB    #PP$FB,@#PP$CMR ;Stop sending a break
                                ;
                                ;  Finished
                                ;
003022  000207                  9$:     RETURN
```

```
                                        .SBTTL  PPSSPD -- Set transmission speed for printer port
                                ;-----------------------------------------------------------------------
                                ;  PPSSPD is called to set the transmission speed for the printer port.
                                ;
                                ;  Inputs:
                                ;   R0 = Speed code.
                                ;   R1 = Physical line index number.
                                ;
003024  010346                  PPSSPD: MOV     R3,-(SP)
                                ;
                                ;  Save new parameter flags for line
                                ;
003026  110061  000010                  MOVB    R0,LMXPRM+1(R1) ;Save new flags for line
                                ;
                                ;  Build value to store in mode register 1 (char length and parity)
                                ;
003032  012703  000102                  MOV     #PP$M1F,R3      ;Get standard mode register 1 flags
003036  032700  000000                  BIT     #LP$7BT,R0      ;7 bit characters wanted?
003042  001003                          BNE     2$              ;Br if yes
003044  052703  000014                  BIS     #PP$8BT,R3      ;Select 8 bit characters
003050  000402                          BR      3$
003052  052703  000010          2$:     BIS     #PP$7BT,R3      ;Select 7 bit characters
003056  032700  000000          3$:     BIT     #LP$PAR,R0      ;Parity wanted?
003062  001407                          BEQ     1$              ;Br if not
003064  052703  000020                  BIS     #PP$PAR,R3      ;Enable parity
003070  032700  000000                  BIT     #LP$ODD,R0      ;Odd parity wanted?
003074  001002                          BNE     1$              ;Br if yes
003076  052703  000040                  BIS     #PP$EVN,R3      ;Select even parity
003102                          1$:     DISABL                  ;;;** Disable interrupts **
003110  105737  173406                  TSTB    @#PP$CMR        ;;;Access command reg to reset mode reg ptr
003114  110337  173404                  MOVB    R3,@#PP$MDR     ;;;Store value into mode register 1
                                ;
                                ;  Store baud rate code into mode register 2
                                ;
003120  042700  000000C                 BIC     #^C<LP$SPD>,R0  ;;;Clear all but baud rate code
003124  052700  000260                  BIS     #PP$M2F,R0      ;;;Set standard flags for mode register 2
003130  110037  173404                  MOVB    R0,@#PP$MDR     ;;;Store value into mode register 2
003134                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
003142  012603                          MOV     (SP)+,R3
003144  000207                          RETURN
```

```
                        .SBTTL  *** Communications Port Control Routines  ***
                        .SBTTL  CPINIT -- Communications port initialization
                ;-----------------------------------------------------------------------
                ;  Perform initialization for the communications port.
                ;
                ;  Inputs:
                ;   R1 = Index number of time-sharing line connected to comm port
                ;
003146          CPINIT:
                ;
                ;  Remember which line is connected to comm port
                ;
003146 010137 000050'           MOV     R1,CPLINE       ;Remember line index number
003152 012761 000000G 000000G   MOV     #CDX$PC,LCDTYP(R1) ;Set type of communications controller
                ;
                ;  Set addresses of communication port device-dependent routines
                ;
003160 012737 003772' 000000C   MOV     #CPSTRT,CDSTRT+CDX$PC ;Set address of start routine
003166 012737 004060' 000000C   MOV     #CPCLOK,CDCLOK+CDX$PC ;Set address of timer-driven routine
003174 012737 004122' 000000C   MOV     #CPXOFF,CDSXOF+CDX$PC ;Routine to send XOFF
003202 012737 004210' 000000C   MOV     #CPXON,CDSXON+CDX$PC  ;Routine to send XON
003210 012737 004276' 000000C   MOV     #CPGDSS,CDGDSS+CDX$PC ;Routine to get data set status flags
003216 012737 004346' 000000C   MOV     #CPSDSS,CDSDSS+CDX$PC ;Routine to do data set control
003224 012737 004374' 000000C   MOV     #CPSBRK,CDSBRK+CDX$PC ;Routine to control break transmission
003232 012737 004462' 000000C   MOV     #CPSSPD,CDSSPD+CDX$PC ;Routine to set transmission speed
                ;
                ;  Connect to interrupt vectors
                ;
003240 012737 003504' 000210    MOV     #CPCINT,@#CP$RTV;Receiver/transmitter interrupt
003246 012737 000340  000212    MOV     #340,@#CP$RTV+2 ;Set prio = 7
003254 012737 003504' 000214    MOV     #CPCINT,@#CP$MCV;Modem change interrupt
003262 012737 000340  000216    MOV     #340,@#CP$MCV+2 ;Set prio = 7
                ;
                ;  Initialize values in CSR A
                ;
003270 012700 173302            MOV     #CP$CAR,R0      ;Get pointer to CSR A
003274 112710 000030            MOVB    #CP$CR,(R0)     ;Reset the channel
003300 000240                   NOP                     ;Delay time for channel clear operation
003302 000240                   NOP
003304 000240                   NOP
003306 000240                   NOP
003310 112710 000300            MOVB    #300,(R0)       ;Reset transmitter underrun condition
003314 112710 000004            MOVB    #CP$WR4,(R0)    ;Select write register 4
003320 112710 000104            MOVB    #CP$AW4,(R0)    ;16x clock rate, 1 stop bit, asynch
003324 112710 000003            MOVB    #CP$WR3,(R0)    ;Select write register 3
003330 112710 000001            MOVB    #CP$AW3,(R0)    ;Enable receiver and set 8-bit char len
003334 112710 000005            MOVB    #CP$WR5,(R0)    ;Select write register 5
003340 112710 000010            MOVB    #CP$AW5,(R0)    ;8-bit char, enable transmitter
003344 112710 000002            MOVB    #CP$WR2,(R0)    ;Select write register 2
003350 112710 000000            MOVB    #0,(R0)         ;Store all zero in WR2
003354 112710 000020            MOVB    #CP$RES,(R0)    ;Reset external interrupts
003360 116100 000001G           MOVB    LMXPRM+1(R1),R0 ;Get flags for line
003364 001002                   BNE     1$              ;Br if speed specified
003366 012700 000000G           MOV     #S9600,R0       ;Default to 9600 baud
003372 004737 004462'   1$:     CALL    CPSSPD          ;Set speed, parity, character length
                ;
                ;  Initialize values in CSR B
```

```
      58                                        ;
      59 003376  012700  173306                        MOV     #CP$CBR,R0      ;Get pointer to CSR B
      60 003402  112710  000030                        MOVB    #CP$CR,(R0)     ;Reset the channel
      61 003406  112710  000002                        MOVB    #CP$WR2,(R0)    ;Select write register 2
      62 003412  112710  000000                        MOVB    #0,(R0)         ;Set 0 as vector base
      63 003416  112710  000001                        MOVB    #CP$WR1,(R0)    ;Select write register 1
      64 003422  112710  000004                        MOVB    #4,(R0)         ;Store value specified in ref manual
      65                                        ;
      66                                        ;  Enable interrupts
      67                                        ;
      68 003426  112737  000033  173202                MOVB    #IM$CRM!IM$CPD,@#ICOCSR ;Clear interrupt mask
      69 003434  112737  000000  173310                MOVB    #0,@#CP$MOR     ;Initialize modem control register 0
      70 003442  112737  000001  173302                MOVB    #CP$WR1,@#CP$CAR ;Select CSR A write register 1
      71 003450  112737  000032  173302                MOVB    #CP$RIE!CP$TIE,@#CP$CAR ;Enable receiver and transmitter ints
      72 003456  152737  000010  173310                BISB    #CP$RTS,@#CP$MOR;Set RTS
      73 003464  032761  0000000 0000006              BIT     #$PHONE,ILSW2(R1);Is this a dial-up line?
      74 003472  001003                                BNE     2$              ;Br if yes -- don't set DTR yet, wait for ring
      75 003474  152737  000020  173310                BISB    #CP$DTR,@#CP$MOR;Set DTR
      76                                        ;
      77                                        ;  Finished
      78                                        ;
      79 003502  000207                         2$:    RETURN
```

```
                                        .SBTTL  CPCINT -- Receiver/Transmitter interrupt routine
                                ;-----------------------------------------------------------------------
                                ;  CPCINT is the interrupt entry point for both receiver and transmitter
                                ;  interrupts.
                                ;
003504  010046                 CPCINT: MOV     R0,-(SP)        ;Get a work register
003506  112737  000002  173306         MOVB    #CP$RR2,@#CP$CBR;Select CSR B read register 2
003514  113700  173306                 MOVB    @#CP$CBR,R0     ;Get the interrupt flags from RR2
003520  006200                         ASR     R0              ;Get value as word table index
003522  042700  177761                 BIC     #^C<16>,R0      ;Clear all but interrupt code flags
003526  000170  003532'                JMP     @CPIVEC(R0)     ;Jump to appropriate interrupt service rtn
                                ;
                                ;  Jump vector based on interrupt type
                                ;
003532  003572'                 CPIVEC: .WORD   CPIERR          ;0 - Invalid
003534  003572'                         .WORD   CPIERR          ;1 - Invalid
003536  003572'                         .WORD   CPIERR          ;2 - Invalid
003540  003572'                         .WORD   CPIERR          ;3 - Invalid
003542  003672'                         .WORD   CPITR           ;4 - Transmitter buffer emtpy
003544  003564'                         .WORD   CPIESC          ;5 - External status change
003546  003604'                         .WORD   CPIREC          ;6 - Received character
003550  003552'                         .WORD   CPISRC          ;7 - Special receiver condition
                                ;
                                ;  Special receiver condition interrupt
                                ;
003552  112737  000060  173302 CPISRC: MOVB    #CP$ER,@#CP$CAR ;Reset error status flags
003560  000137  003604'                JMP     CPIREC          ;Process any character we got with errors
                                ;
                                ;  External status change interrupt processing
                                ;
003564  112737  000020  173302 CPIESC: MOVB    #CP$RES,@#CP$CAR;Reset external status
                                ;
                                ;  Invalid interrupt code.
                                ;  Reset interrupt status and return.
                                ;
003572  112737  000070  173302 CPIERR: MOVB    #CP$EI,@#CP$CAR ;Declare end of interrupt
003600  012600                         MOV     (SP)+,R0
003602  000002                         RTI                     ;Return from interrupt
```

```
                                        .SBTTL  CPIREC -- Received character from comm port
                                ;-----------------------------------------------------------------------
                                ;  CPIREC is the interrupt service routine called when we get an
                                ;  interrupt from the comm port saying that it has received a character.
                                ;
                                ;  On entry, R0 is on the stack.
                                ;
003604  012600                  CPIREC: MOV     (SP)+,R0        ;Restore R0
                                ;
                                ;  Call routine to save R0, R1, R4, and R5
                                ;
003606  004577  0000000'              JSR     R5,@TTRSAV      ;Save registers
                                ;
                                ;  R0, R1, R4 and R5 are available.
                                ;  See if a framing error was detected
                                ;
003612  005005                        CLR     R5              ;Build character and flags in R5
003614  112737  000000  173302        MOVB    #CP$RR0,@#CP$CAR;Select read register 0
003622  113704  173302               MOVB    @#CP$CAR,R4     ;Get contents of read register 0
003626  132704  000200               BITB    #CP$BR,R4       ;Was a break detected?
003632  001402                       BEQ     3$              ;Br if not
003634  052705  0000000'             BIS     #FRMERR,R5      ;Set framing error flag
003640  112737  000020  173302  3$:  MOVB    #CP$RES,@#CP$CAR;Reset register 0 flags
                                ;
                                ;  Get the character from the receiver
                                ;
003646  153705  173300                BISB    @#CP$DBR,R5     ;Get the received character
                                ;
                                ;  Reenable interrupts from receiver
                                ;
003652  112737  000070  173302        MOVB    #CP$EI,@#CP$CAR ;Declare end of interrupt
                                ;
                                ;  Get number of line connected to communications port
                                ;
003660  013704  000050'               MOV     CPLINE,R4       ;Get line index number
                                ;
                                ;  Call TTINPT to process the character
                                ;
003664  004777  0000000'              CALL    @TTINPT         ;Enter routine to process the character
                                ;
                                ;  Return from interrupt
                                ;
003670  000207                        RETURN                  ;Return from interrupt through TTRSAV
```

```
                                        .SBTTL  CPITR  -- Transmitter interrupt
                                ;-----------------------------------------------------------------
                                ;  CPITR is the interrupt service routine for the communications port
                                ;  transmitter.
                                ;
                                ;  On entry, R0 is on the stack.
                                ;
003672  010446                  CPITR:  MOV     R4,-(SP)        ;Stack R4 also
003674  013746  0000000G                MOV     INTPRI,-(SP)    ;Save current interrupt priority
003700  005037  0000000G                CLR     INTPRI          ;Say running interrupt priority is 7
                                ;
                                ;  Say that line is no longer transmitting a character
                                ;
003704  013704  000050'                 MOV     CPLINE,R4       ;Get # of line connected to comm port
003710  042764  0000000G 0000000G       BIC     #$XCHAR,LSW3(R4);Clear transmitting-character flag for line
003716  112737  000050  173302          MOVB    #CP$RTI,@#CP$CAR;Reset transmitter interrupt flag
003724  112737  000070  173302          MOVB    #CP$EI,@#CP$CAR ;Declare end of interrupt
                                ;
                                ;  Get next character to transmit
                                ;
003732  004774  0000000G                CALL    @LOUTIR(R4)     ;Get next char to transmit
003736  103410                          BCS     9$              ;Br if no more chars to send
                                ;
                                ;  Transmit the character
                                ;
003740  052764  0000000G 0000000G       BIS     #$XCHAR,LSW3(R4);Say a character is being transmitted
003746  042764  0000000G 0000000G       BIC     #$OITIM,LSW5(R4);Start timer to catch lost interrupts
003754  110037  173300                  MOVB    R0,@#CP$DBR     ;Transmit the character
                                ;
                                ;  Finished
                                ;
003760  012637  0000000G        9$:     MOV     (SP)+,INTPRI    ;Restore interrupt priority level
003764  012604                          MOV     (SP)+,R4
003766  012600                          MOV     (SP)+,R0
003770  000002                          RTI                     ;Return from interrupt
```

```
                                        .SBTTL  CPSTRT -- Start output to communications port
                                ;-----------------------------------------------------------------
                                ;  CPSTRT is called to initiate output to the communications port.
                                ;
                                ;  R1 = Line index number of line being started.
                                ;
003772  010046                  CPSTRT: MOV     R0,-(SP)
003774  010446                          MOV     R4,-(SP)
                                ;
                                ;  Disable communications port interrupts
                                ;
003776                              DISABL                          ;;;Disable interrupts
                                ;
                                ;  See if the communications port transmitter is already busy
                                ;
004004  032761  0000000 0000000     BIT     #$XCHAR,LSW3(R1)        ;;;Is the transmitter busy now?
004012  001014                      BNE     1$                      ;;;Br if yes
                                ;
                                ;  Transmitter is idle.
                                ;  Get next character to transmit
                                ;
004014  010104                      MOV     R1,R4                   ;;;Get line # to R4 for NEDCHR
004016  004774  0000000             CALL    @LOUTIR(R4)             ;;;Get a character to transmit
004022  103410                      BCS     1$                      ;;;Br if nothing to transmit
                                ;
                                ;  Start transmission
                                ;
004024  052761  0000000 0000000     BIS     #$XCHAR,LSW3(R1)        ;;;Say a character being transmitted
004032  042761  0000000 0000000     BIC     #$OITIM,LSW5(R1)        ;;;Start timer to catch lost interrupt
004040  110037  173300              MOVB    R0,@#CP$DBR             ;;;Transmit the character
                                ;
                                ;  Enable the transmitter interrupt
                                ;
004044                          1$: ENABL                           ;Enable interrupts
                                ;
                                ;  Finished
                                ;
004052  012604                      MOV     (SP)+,R4
004054  012600                      MOV     (SP)+,R0
004056  000207                      RETURN
```

```
                                        .SBTTL  CPCLOK -- Timer driven routine for communications port
                                ;-----------------------------------------------------------------------
                                ;  CPCLOK is a timer driven routine called periodically to check on
                                ;  the status of the communications port.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
004060                          CPCLOK:
                                ;
                                ;  Check for lost output interrupts
                                ;
004060  032761  0000000G 0000000G       BIT     #$OITIM,LSW5(R1);Have we started timer interval?
004066  001411                          BEQ     1$              ;Br if not
004070  032761  0000000G 0000000G       BIT     #$XCHAR,LSW3(R1);Are we still waiting for an interrupt?
004076  001410                          BEQ     9$              ;Br if not
004100  042761  0000000G 0000000G       BIC     #$XCHAR,LSW3(R1);Say wait is over
004106  004737  003772'                 CALL    CPSTRT          ;Try to restart transmitter
004112  052761  0000000G 0000000G 1$:   BIS     #$OITIM,LSW5(R1);Start timed interval
                                ;
                                ;  Finished
                                ;
004120  000207                  9$:     RETURN
```

```
                                        .SBTTL  CPXOFF -- Transmit XOFF to communications port
                                ;-----------------------------------------------------------------------
                                ;  CPXOFF is called to stuff an XOFF character into the output stream
                                ;  for the communications port.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
004122                          CPXOFF: DISABL                  ;;;** Disable interrupts **
                                ;
                                ;  Set flag that says XOFF has been sent
                                ;
004130  052761  0000000 0000000         BIS     #$HISTP,LSW10(R1);;;Remember that XOFF has been sent
                                ;
                                ;  See if the transmitter is busy now
                                ;
004136  032761  0000000 0000000         BIT     #$XCHAR,LSW3(R1);;;Is the transmitter busy now?
004144  001404                          BEQ     1$              ;;;Br if not busy now
                                ;
                                ;  The transmitter is busy now.
                                ;  Set flag which will cause output character generator to send XOFF
                                ;
004146  052761  0000000 0000000         BIS     #$SXOFF,LSW10(R1);;;Set flag to cause XOFF to be sent
004154  000411                          BR      9$              ;;;
                                ;
                                ;  Transmitter is not busy.
                                ;  Transmit an XOFF character.
                                ;
004156  052761  0000000 0000000 1$:     BIS     #$XCHAR,LSW3(R1);;;Say transmitter busy
004164  042761  0000000 0000000         BIC     #$OITIM,LSW5(R1);;;Start output interrupt timer
004172  112737  000023  173300          MOVB    #23,@#CP$DBR    ;;;Transmit XOFF
                                ;
                                ;  Finished
                                ;
004200                          9$:     ENABL                   ;** Enable interrupts **
004206  000207                          RETURN
```

```
                                        .SBTTL  CPXON  -- Transmit XON to communications port
                                ;-----------------------------------------------------------------------
                                ;  CPXON is called to stuff an XON character into the output stream
                                ;  for the communications port.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
004210                          CPXON:  DISABL                  ;;;** Disable interrupts **
                                ;
                                ;  Clear flag that says XOFF has been sent
                                ;
004216  042761  000000G 000000G         BIC     #$HISTP,LSW10(R1);;;Clear flag that says XOFF sent
                                ;
                                ;  See if the transmitter is busy now
                                ;
004224  032761  000000G 000000G         BIT     #$XCHAR,LSW3(R1);;;Is the transmitter busy now?
004232  001404                          BEQ     1$              ;;;Br if not busy now
                                ;
                                ;  The transmitter is busy now.
                                ;  Set flag which will cause output character generator to send XON
                                ;
004234  052761  000000G 000000G         BIS     #$SXON,LSW10(R1);;;Set flag to cause XON to be sent
004242  000411                          BR      9$              ;;;
                                ;
                                ;  Transmitter is not busy.
                                ;  Transmit an XON character.
                                ;
004244  052761  000000G 000000G 1$:     BIS     #$XCHAR,LSW3(R1);;;Say transmitter busy
004252  042761  000000G 000000G         BIC     #$OITIM,LSW5(R1);;;Start output interrupt timer
004260  112737  000021  173300          MOVB    #21,@#CP$DBR    ;;;Transmit XON
                                ;
                                ;  Finished
                                ;
004266                          9$:     ENABL                   ;** Enable interrupts **
004274  000207                          RETURN
```

```
                                        .SBTTL  CPGDSS -- Get communications port data set status
                                ;-----------------------------------------------------------------------
                                ;  CPGDSS is called to get the data set (modem) status for the
                                ;  communications port.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number
                                ;
                                ;  Outputs:
                                ;   R0 = Generic data set status flags (MS$xxx)
                                ;
004276  005000                  CPGDSS: CLR     R0              ;Develop result in R0
                                ;
                                ;  See if the phone is ringing
                                ;
004300  132737  000100  173312          BITB    #CP$RI,@#CP$M1R ;Is the phone ringing?
004306  001402                          BEQ     1$              ;Br if not
004310  052700  000000G               BIS     #MS$RNG,R0      ;Set ringing flag
                                ;
                                ;  See if carrier is detected
                                ;
004314  132737  000020  173312  1$:     BITB    #CP$CD,@#CP$M1R ;Do we have carrier detect?
004322  001402                          BEQ     2$              ;Br if not
004324  052700  000000G               BIS     #MS$CAR,R0      ;Set carrier-detect flag
                                ;
                                ;  See if Data Terminal Ready is asserted
                                ;
004330  132737  000020  173310  2$:     BITB    #CP$DTR,@#CP$MOR;Are we asserting DTR?
004336  001402                          BEQ     3$              ;Br if not
004340  052700  000000G               BIS     #MS$DTR,R0      ;Remember DTR is asserted
                                ;
                                ;  Finished
                                ;
004344  000207                  3$:     RETURN
```

```
                                        .SBTTL  CPSDSS -- Set data set status for communications port
                                ;-----------------------------------------------------------------------
                                ;  CPSDSS is called to control the data set (modem) status for the
                                ;  communications port.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number
                                ;   R0 = Data set control flags (MS$xxx)
                                ;
004346                          CPSDSS:
                                ;
                                ;  See if we should raise or drop Data Terminal Ready
                                ;
004346  032700  0000000               BIT     #MS$DTR,R0      ;Raise or drop DTR?
004352  001404                        BEQ     1$              ;Br to drop DTR
                                ;
                                ;  Raise DTR
                                ;
004354  152737  000020  173310        BISB    #CP$DTR,@#CP$MOR;Raise DTR
004362  000403                        BR      9$
                                ;
                                ;  Drop DTR
                                ;
004364  142737  000020  173310  1$:   BICB    #CP$DTR,@#CP$MOR;Drop DTR
                                ;
                                ;  Finished
                                ;
004372  000207                  9$:   RETURN
```

```
                                        .SBTTL  CPSBRK -- Control break transmission on communications port
                                ;-----------------------------------------------------------------------
                                ;  CPSBRK is called to start or stop transmitting a break character to
                                ;  the communications port.
                                ;
                                ;  Inputs:
                                ;   R0 = Break control flag (MS$BRK)
                                ;   R1 = Physical line index number.
                                ;
004374  010246                  CPSBRK: MOV     R2,-(SP)
                                ;
                                ;  Set standard flags for write-register 5
                                ;
004376  012702  000050                  MOV     #CP$AW5!CP$7BT,R2 ;Standard flags and 7-bit char length
004402  132761  0000000 0000010        BITB    #LP$7BT,LMXPRM+1(R1);7-bit characters wanted?
004410  001002                          BNE     2$                ;Br if yes
004412  052702  000140                  BIS     #CP$8BT,R2        ;Select 8-bit characters
                                ;
                                ;  See if we want to start or stop sending a break
                                ;
004416  032700  0000000         2$:     BIT     #MS$BRK,R0        ;Start or stop sending break?
004422  001402                          BEQ     1$                ;Br if stop
004424  052702  000020                  BIS     #CP$SB,R2         ;Set send-break flag
                                ;
                                ;  Store parameters into control register
                                ;
004430                          1$:     DISABL                    ;;;** Disable interrupts **
004436  112737  000005  173302          MOVB    #CP$WR5,@#CP$CAR  ;;;Select write register 5
004444  110237  173302                  MOVB    R2,@#CP$CAR       ;;;Start or stop sending break
004450                                  ENABL                     ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
004456  012602                  9$:     MOV     (SP)+,R2
004460  000207                          RETURN
```

```
                                        .SBTTL  CPSSPD -- Set speed for communications port
                                ;-----------------------------------------------------------------
                                ;  CPSSPD is called to set the transmit/receive speed for
                                ;  the communications port.
                                ;
                                ;  Inputs:
                                ;   R0 = Speed code.
                                ;   R1 = Physical line index number.
                                ;
004462  010346                  CPSSPD: MOV     R3,-(SP)
004464  010446                          MOV     R4,-(SP)
004466  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set baud rate in line parameter word for this line
                                ;
004470  110061  000001G                 MOVB    R0,LMXPRM+1(R1) ;Save new flags for line
                                ;
                                ;  Store speed value into hardware control register
                                ;
004474  010003                          MOV     R0,R3           ;Get all flags
004476  042703  000000C                 BIC     #^C<LP$SPD>,R3  ;Clear all but speed code
004502  010346                          MOV     R3,-(SP)        ;Save speed code
004504  072327  000004                  ASH     #4,R3           ;Shift speed value left 4 bits
004510  052603                          BIS     (SP)+,R3        ;Combine with unshifted value
004512  110337  173314                  MOVB    R3,@#CP$BRR     ;Set baud rate in hardware register
                                ;
                                ;  Set parity control
                                ;
004516  012704  173302                  MOV     #CP$CAR,R4      ;Get pointer to CSR A
004522  012703  000104                  MOV     #CP$AW4,R3      ;Get standard bits for WR4
004526  032700  0000000                 BIT     #LP$PAR,R0      ;Is parity wanted?
004532  001407                          BEQ     1$              ;Br if not
004534  052703  000001                  BIS     #CP$PAR,R3      ;Set parity-enable flag
004540  032700  0000000                 BIT     #LP$ODD,R0      ;Is odd parity wanted?
004544  001002                          BNE     1$              ;Br if yes
004546  052703  000002                  BIS     #CP$EVN,R3      ;Select even parity
004552  112714  000004          1$:     MOVB    #CP$WR4,(R4)    ;Select write register 4
004556  110314                          MOVB    R3,(R4)         ;Set parity control flags
                                ;
                                ;  Set character length
                                ;
004560  012703  000001                  MOV     #CP$AW3,R3      ;Get standard flags for WR3
004564  012705  000010                  MOV     #CP$AW5,R5      ;Get standard flags for WR5
004570  032700  0000000                 BIT     #LP$7BT,R0      ;7 bit characters wanted?
004574  001005                          BNE     2$              ;Br if yes
004576  052705  000140                  BIS     #CP$8BT,R5      ;Select 8 bit characters
004602  052703  000300                  BIS     #CP$8BR,R3
004606  000404                          BR      3$
004610  052705  000040          2$:     BIS     #CP$7BT,R5      ;Select 7 bit characters
004614  052703  000100                  BIS     #CP$7BR,R3
004620                          3$:     DISABL                  ;;;** Disable interrupts **
004626  112714  000003                  MOVB    #CP$WR3,(R4)    ;;;Select write register 3
004632  110314                          MOVB    R3,(R4)         ;;;Set length for received characters
004634  112714  000005                  MOVB    #CP$WR5,(R4)    ;;;Select write register 5
004640  110514                          MOVB    R5,(R4)         ;;;Set length for transmitted characters
004642                                  ENABL                   ;** Enable interrupts **
                                ;
```

```
                                        ;  Finished
                                        ;
004650  012605                                MOV     (SP)+,R5
004652  012604                                MOV     (SP)+,R4
004654  012603                                MOV     (SP)+,R3
004656  000207                                RETURN
```

```
                                        .IF     NE,QPASM        ;Assemble if quad line unit code wanted
                                        .SBTTL  *** Quad Serial Line Unit Routines ***
                                        .SBTTL  QPINIT -- Initialize quad serial line unit
                                ;-----------------------------------------------------------------------
                                ;  QPINIT is called to initialize the quad serial line unit
                                ;
004660  010546                  QPINIT: MOV     R5,-(SP)
                                ;
                                ;  Determine if a quad line unit is installed
                                ;
004662  012746  000064                  MOV     #QP$ID,-(SP)    ;Stack device ID for quad line unit
004666  004737  001172'                 CALL    QPRCSR          ;Get CSR for quad line unit
004672  012605                          MOV     (SP)+,R5        ;Get returned CSR address
004674  103014                          BCC     2$              ;Br if it is installed
                                ;
                                ;  There is no quad line unit installed.
                                ;  Mark all lines associated with quad line unit as dead.
                                ;
004676  012705  000000G                 MOV     #LSTHL,R5       ;Get index to last line
004702  005765  000000G         3$:     TST     LMXNUM(R5)      ;Is this a quad unit line?
004706  001403                          BEQ     4$              ;Br if not
004710  052765  000000G 000000G         BIS     #$DEAD,LSW3(R5) ;Mark line as dead
004716  162705  000002          4$:     SUB     #2,R5           ;Get next line index
004722  003367                          BGT     3$              ;Br if more lines to do
004724  000502                          BR      9$              ;Finished
                                ;
                                ;  There is a quad line unit installed
                                ;
004726  010537  000066'          2$:     MOV     R5,QPCSR        ;Save CSR address
004732  132765  000004  000004          BITB    #QP$22C,QPRMSR(R5);Do we have modem control on lines 0 and 2?
004740  001402                          BEQ     1$              ;Br if not
004742  105237  000110'                 INCB    QPMODM          ;Remember we have modem control on two lines
004746  142765  000100  000004  1$:     BICB    #QP$IEN,QPRMSR(R5);Clear interrupt-enable bit
004754  012746  000064                  MOV     #QP$ID,-(SP)    ;Stack device ID
004760  004737  001134'                 CALL    QPRVEC          ;Get vector address
004764  012637  000070'                 MOV     (SP)+,QPVEC     ;Save vector address
004770  012700  000064                  MOV     #QP$ID,R0       ;Get device ID
004774  004737  001254'                 CALL    GETSLT          ;Get slot # where device is installed
005000  110037  000111'                 MOVB    R0,QPSLOT       ;Save the slot number
                                ;
                                ;  Set addresses of quad line unit device-dependent routines
                                ;
005004  012737  005310' 000000C         MOV     #QPSTRT,CDSTRT+CDX$QP ;Start-output routine
005012  012737  006054' 000000C         MOV     #QPGDSS,CDGDSS+CDX$QP ;Get data set status
005020  012737  006160' 000000C         MOV     #QPSDSS,CDSDSS+CDX$QP ;Set data set status
005026  012737  006016' 000000C         MOV     #QPSBRK,CDSBRK+CDX$QP ;Control break transmission
005034  012737  005632' 000000C         MOV     #QPSSPD,CDSSPD+CDX$QP ;Set line speed
                                ;
                                ;  Initialize Output Port Configuration Register
                                ;
005042  105065  000132                  CLRB    QPROPC(R5)      ;All bits are off in OPCR
005046  105065  000172                  CLRB    QPROPC+QPRUOF(R5) ;DUART 1
                                ;
                                ;  Initialize Auxiliary Control Register
                                ;
005052  112765  000360  000110          MOVB    #QP$AWA,QPRACR(R5) ;Initialize ACR
005060  112765  000360  000150          MOVB    #QP$AWA,QPRACR+QPRUOF(R5) ;DUART 1
```

```
                                        ;
                                        ;   Initialize Interrupt Mask Register
                                        ;
005066  112765  000063  000112                MOVB    #QP$AWI,QPRIMR(R5) ;Intialize IMR
005074  112765  000063  000152                MOVB    #QP$AWI,QPRIMR+QPRUOF(R5) ;DUART 1
                                        ;
                                        ;   Initialize output port
                                        ;
005102  112765  000377  000136                MOVB    #377,QPRROP(R5) ;Reset all output port bits
005110  112765  000377  000176                MOVB    #377,QPRROP+QPRUOF(R5) ;DUART 1
                                        ;
                                        ;   Initialize interrupt controller
                                        ;
005116  012700  000030                        MOV     #IM$CRM,R0      ;Get command to clear IRR and IMR int flags
005122  153700  000111'                      BISB    QPSLOT,R0       ;Or in slot number where QP controller is
005126  110037  173206                        MOVB    R0,@#IC1CSR     ;Clear IRR and IMR flags for quad port
                                        ;
                                        ;   Finished
                                        ;
005132  012605                        9$:     MOV     (SP)+,R5
005134  000207                                RETURN
```

```
                                        .SBTTL  QPLINE -- Initialize a line connected to quad line unit
                                ;-----------------------------------------------------------------------
                                ;  QPLINE is called to initialize a specific line that is connected to
                                ;  the quad serial line unit.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
005136  010546                 QPLINE: MOV     R5,-(SP)
                                ;
                                ;  Say this line is controlled by quad serial line unit
                                ;
005140  012761  000000G 000000G         MOV     #CDX$QP,LCDTYP(R1)
                                ;
                                ;  See if quad line unit is installed
                                ;
005146  005737  000066'                 TST     QPCSR           ;Is quad line unit installed?
005152  001454                         BEQ     9$              ;Br if not
                                ;
                                ;  Remember the TSX-Plus line index number associated with this unit line
                                ;
005154  016105  000000G                 MOV     LMXLN(R1),R5    ;Get # of line within quad unit
005160  110165  000104'                 MOVB    R1,QPLX(R5)     ;Store TSX-Plus line index #
005164  016100  000000G                 MOV     LMXNUM(R1),R0   ;Get mux index number
005170  013760  000066' 000000G         MOV     QPCSR,MXCSR(R0) ;Remember CSR address for unit
005176  013760  000070' 000000G         MOV     QPVEC,MXVEC(R0) ;Remember vector address of unit
                                ;
                                ;  See if this line should have modem control
                                ;
005204  105737  000110'                 TSTB    QPMODM          ;Are we configured for modem control?
005210  001403                         BEQ     3$              ;Br if not -- No lines have modem control
005212  032705  000001                 BIT     #1,R5           ;Is this line 0 or 2?
005216  001403                         BEQ     4$              ;Br if yes -- Has modem control
005220  042761  000000G 000000G 3$:    BIC     #$PHONE,ILSW2(R1);Say line does not have modem control
                                ;
                                ;  Convert line number into quad unit address
                                ;
005226  004737  006236'         4$:    CALL    QPCVLA          ;Convert line # to address
                                ;
                                ;  At this point,
                                ;   R5 = Base address of registers for quad unit and channel.
                                ;
                                ;  Initialize mode registers 1 and 2
                                ;
005232  112765  000020  000104         MOVB    #QP$RPR,QPRCRA(R5);Select mode register 1
005240  112765  000023  000100         MOVB    #023,QPRMRA(R5) ;Initialize mode register 1
005246  112765  000007  000100         MOVB    #007,QPRMRA(R5) ;Initialize mode register 2
                                ;
                                ;  Initialize clock select register
                                ;
005254  112765  000273  000102         MOVB    #273,QPRCSA(R5) ;Initialize to 9600 baud
                                ;
                                ;  Initialize receiver and transmitter
                                ;
005262  112765  000052  000104         MOVB    #QP$RRX!QP$DRX!QP$DTX,QPRCRA(R5) ;Reset receiver + disable
005270  112765  000100  000104         MOVB    #QP$RES,QPRCRA(R5) ;Reset error status
005276  112765  000065  000104         MOVB    #QP$RTX!QP$ERX!QP$ETX,QPRCRA(R5) ;Reset transmitter + enable
```

```
                                        ;
                                        ;  Finished
                                        ;
005304  012605                          9$:     MOV     (SP)+,R5
005306  000207                                  RETURN
```

```
                                        .SBTTL  QPSTRT -- Start output
                                ;-----------------------------------------------------------------------
                                ;  QPSTRT is called to initiate output to a line connected to the
                                ;  quad line unit.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;
005310  010546                  QPSTRT: MOV     R5,-(SP)
                                ;
                                ;  Convert line number into quad unit address
                                ;
005312  004737  006236'                 CALL    QPCVLA          ;Convert line # to address
                                ;
                                ;  At this point,
                                ;   R5 = Base address of registers for quad unit and channel.
                                ;
                                ;  See if the transmitter is already busy
                                ;
005316                                  DISABL                  ;;;Disable interrupts
005324  032761  000000G 000000G         BIT     #$XCHAR,LSW3(R1);;;Is the transmitter busy now?
005332  001003                          BNE     1$              ;;;Br if yes
                                ;
                                ;  Transmitter is idle.
                                ;  Enable the transmitter which will cause an interrupt.
                                ;
005334  112765  000004  000104        MOVB    #QP$ETX,QPRCRA(R5);;;Enable transmitter (will cause interrupt)
                                ;
                                ;  Finished
                                ;
005342                          1$:     ENABL                   ;Enable interrupts
005350  012605                          MOV     (SP)+,R5
005352  000207                          RETURN
```

```
                                        .SBTTL  QPCINT -- Interrupt from quad line unit
                                ;-----------------------------------------------------------------------
                                ;  QPCINT is the interrupt entry point for all interrupts from the
                                ;  quad line unit.
                                ;
005354                          QPCINT:
                                ;
                                ;  Save registers R0, R1, R4, and R5.
                                ;  TTRSAV is the standard interrupt entry routine called when processing
                                ;  received character interrupts. We always call it in the case of an
                                ;  interrupt from a quad unit line since we don't know whether the interrupt
                                ;  is from a transmitter or receiver.
                                ;
005354  004577  0000000               JSR     R5,@TTRSAV      ;Save registers
005360  010246                        MOV     R2,-(SP)
005362  010346                        MOV     R3,-(SP)
                                ;
                                ;  Disable further interrupts from the quad line unit
                                ;
005364  013703  000066'               MOV     QPCSR,R3        ;Get address of base of registers for DUART 0
005370  142763  000100  000004        BICB    #QP$IEN,QPRMSR(R3) ;Clear interrupt-enable bit
005376  010301                        MOV     R3,R1           ;Get addr of base of registers for DUART
                                ;
                                ;  Begin loop to determine which of the four lines needs service
                                ;
005400  005002                        CLR     R2              ;Start with line # 0 (DUART 0, channel A)
                                ;
                                ;  Get TSX-Plus line index number of this line
                                ;
005402  116204  000104'         1$:   MOVB    QPLX(R2),R4     ;Get TSX-Plus line index number
005406  001461                        BEQ     3$              ;Br if no line connected
                                ;
                                ;  See if an output interrupt occurred
                                ;
005410  136263  005622' 000112        BITB    QPFTXR(R2),QPRISR(R3);Is transmitter ready for another char?
005416  001422                        BEQ     2$              ;Br if not
                                ;
                                ;  The transmitter is ready for another character.
                                ;  See if there is another character to transmit.
                                ;
005420  004774  0000000               CALL    @LOUTIR(R4)     ;Get next character to transmit
005424  103411                        BCS     8$              ;Br if no more chars to transmit
                                ;
                                ;  Transmit the character contained in R0
                                ;
005426  052764  000000@ 000000@       BIS     #$XCHAR,LSW3(R4);Set transmitter-busy flag
005434  042764  000000@ 000000@       BIC     #$OITIM,LSW5(R4);Start timer to catch lost interrupts
005442  110061  000106                MOVB    R0,QPRTRA(R1)   ;Start transmitting the character
005446  000406                        BR      2$
                                ;
                                ;  There are no more characters to transmit.
                                ;  Disable the transmitter
                                ;
005450  042764  000000@ 000000@ 8$:   BIC     #$XCHAR,LSW3(R4);Say transmission finished
005456  112761  000010  000104        MOVB    #QP$DTX,QPRCRA(R1) ;Disable the transmitter
                                ;
                                ;  See if this line has received a character
```

```
                                        ;
005464  136263  005626' 000112  2$:     BITB    QPFRXR(R2),QPRISR(R3);Is a received character ready?
005472  001427                          BEQ     3$              ;Br if not
                                        ;
                                        ;  Get received character
                                        ;
005474  116100  000102                  MOVB    QPRSRA(R1),R0   ;Get status flags for character
005500  116105  000106                  MOVB    QPRRRA(R1),R5   ;Get the received character
005504  042705  177400                  BIC     #^C<377>,R5     ;Mask character to 8 bits
                                        ;
                                        ;  Set status flags in R5 to simulate status flags returned by DL11
                                        ;
005510  032700  000040                  BIT     #QP$RPE,R0      ;Did we have a parity error?
005514  001402                          BEQ     4$              ;Br if not
005516  052705  000000C                 BIS     #RBERR!RCVPAR,R5;Set parity-error flags
005522  032700  000300          4$:     BIT     #QP$RFE!QP$BRK,R0 ;Framing error or break?
005526  001402                          BEQ     5$              ;Br if not
005530  052705  000000C                 BIS     #RBERR!FRMERR,R5;Set framing-error flags
005534  032700  000020          5$:     BIT     #QP$ROE,R0      ;Did an overrun occur?
005540  001402                          BEQ     6$              ;Br if not
005542  052705  000000C                 BIS     #RBERR!OVRRUN,R5;Set overrun flags
                                        ;
                                        ;  Call TTINPT to process the received character
                                        ;
005546  004777  0000000         6$:     CALL    @TTINPT         ;Enter character input routine
                                        ;
                                        ;  Check for other lines connect to the quad unit that may need service
                                        ;
005552  005202                  3$:     INC     R2              ;Advance line # within quad unit (0-3)
005554  020227  000003                  CMP     R2,#3           ;Have we serviced all lines?
005560  101010                          BHI     7$              ;Br if yes
005562  062701  000020                  ADD     #QPRCOF,R1      ;Point to registers for next channel
005566  020227  000002                  CMP     R2,#2           ;Time to advance register base to DUART 1?
005572  001303                          BNE     1$              ;Br if not
005574  062703  000040                  ADD     #QPRUOF,R3      ;Point to registers for DUART 1
005600  000700                          BR      1$              ;Go service next line
                                        ;
                                        ;  We have serviced all four lines.
                                        ;  Reenable interrupt for quad line unit.
                                        ;
005602  013703  000066'          7$:     MOV     QPCSR,R3        ;Get address of base of registers for DUART 0
005606  152763  000100  000004          BISB    #QP$IEN,QPRMSR(R3) ;Set interrupt-enable bit
                                        ;
                                        ;  Finished
                                        ;
005614  012603                          MOV     (SP)+,R3
005616  012602                          MOV     (SP)+,R2
005620  000207                          RETURN                  ;Return through TTRSAV which restors regs
                                        ;
                                        ;  Vector containing the proper bit mask to test for transmitter-ready
                                        ;  in the Interrupt Status Register for each line.
                                        ;
005622      001     020     001  QPFTXR: .BYTE   1,20,1,20
005625      020
                                        ;
                                        ;  Vector containing the proper bit mask to test for receiver-done
                                        ;  in the Interrupt Status Register for each line.
```

```
    114                                         ;
    115 005626      002     040     002  QPFRXR: .BYTE   2,40,2,40
        005631      040
    116                                         .EVEN
```

```
                                        .SBTTL  QPSSPD -- Set speed for quad line port
                                ;-----------------------------------------------------------------
                                ;  QPSSPD is called to set the transmit/receive speed for a line
                                ;  connected to the quad serial line unit.
                                ;
                                ;  Inputs:
                                ;   R0 = Speed code
                                ;   R1 = Physical line index number.
                                ;
005632  010346                 QPSSPD: MOV     R3,-(SP)
005634  010546                         MOV     R5,-(SP)
                                ;
                                ;  Save info about baud rate for this line
                                ;
005636  110061  0000010                 MOVB    R0,LMXPRM+1(R1) ;Save info in line table
                                ;
                                ;  Convert line number into quad unit address
                                ;
005642  004737  006236'                 CALL    QPCVLA          ;Convert line # to address
                                ;
                                ;  At this point,
                                ;   R5 = Base address of registers for quad unit and channel.
                                ;
                                ;  Set transmit/receive speed
                                ;
005646  010003                          MOV     R0,R3           ;Get control flags
005650  042703  000000C                 BIC     #^C<LP$SPD>,R3  ;Clear all but speed value
005654  116303  005776'                 MOVB    QPSPD(R3),R3    ;Get speed code for quad unit
005660  010346                          MOV     R3,-(SP)
005662  072327  000004                  ASH     #4,R3           ;Position speed code for receive
005666  052603                          BIS     (SP)+,R3        ;Or in transmit speed
005670  110365  000102                  MOVB    R3,QPRCSA(R5)   ;Set speed for line
                                ;
                                ;  Set character length and parity
                                ;
005674  032700  0000000                 BIT     #LP$7BT,R0      ;7 bit characters wanted?
005700  001403                          BEQ     1$              ;Br if not
005702  012703  000002                  MOV     #QP$7BR,R3      ;Get 7 bit char flags
005706  000402                          BR      2$
005710  012703  000003          1$:     MOV     #QP$8BR,R3      ;Get 8 bit char flags
005714  032700  0000000          2$:     BIT     #LP$PAR,R0      ;Is parity wanted?
005720  001003                          BNE     3$              ;Br if yes
005722  052703  000020                  BIS     #QP$NPR,R3      ;Set no-parity flag
005726  000405                          BR      4$
005730  032700  0000000          3$:     BIT     #LP$ODD,R0      ;Odd parity wanted?
005734  001402                          BEQ     4$              ;Br if not
005736  052703  000004                  BIS     #QP$ODD,R3      ;Select odd parity
005742                          4$:     DISABL                  ;;;** Disable interrupts **
005750  112765  000020  000104          MOVB    #QP$RPR,QPRCRA(R5) ;;;Select mode register 1
005756  110365  000100                  MOVB    R3,QPRMRA(R5)   ;;;Set char length and parity
005762                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Finished
                                ;
005770  012605                          MOV     (SP)+,R5
005772  012603                          MOV     (SP)+,R3
005774  000207                          RETURN
```

QPSSPD -- Set speed for quad line port

```
      58                                        ;
      59                                        ;  Vector of values to convert TSX-Plus standard transmit/receive
      60                                        ;  speed codes into codes for the quad serial line unit.
      61                                        ;
      62 005776      000     000     001  QPSPD:   .BYTE   0.,0.,1.,2.,3.,4.,5.,6.,10.,7.,8.,0.,9.,0.,11.,12.
         006001      002     003     004
         006004      005     006     012
         006007      007     010     000
         006012      011     000     013
         006015      014
      63                                        .EVEN
```

```
                                        .SBTTL  QPSBRK -- Control break transmission to quad line unit
                                ;-----------------------------------------------------------------------
                                ;  QPSBRK is called to start or stop transmitting a break character to
                                ;  a line connected to the quad serial line unit.
                                ;
                                ;  Inputs
                                ;   R0 = Break control flag (MS$BRK)
                                ;   R1 = Physical line index number.
                                ;
006016  010546                  QPSBRK: MOV     R5,-(SP)
                                ;
                                ;  Convert line number to address of control register
                                ;
006020  004737  006236'                 CALL    QPCVLA          ;Convert line # to address
                                ;
                                ;  At this point,
                                ;   R5 = Base address of registers for quad unit and channel.
                                ;
                                ;  See if we are to start or stop sending a break
                                ;
006024  032700  0000000                 BIT     #MS$BRK,R0      ;Start or stop sending a break?
006030  001404                          BEQ     1$              ;Br if stop
                                ;
                                ;  Start sending a break
                                ;
006032  112765  000140  000104          MOVB    #QP$SBT,QPRCRA(R5) ;Start break transmission
006040  000403                          BR      9$
                                ;
                                ;  Stop sending a break
                                ;
006042  112765  000160  000104  1$:     MOVB    #QP$EBT,QPRCRA(R5) ;Stop break transmission
                                ;
                                ;  Finished
                                ;
006050  012605                  9$:     MOV     (SP)+,R5
006052  000207                          RETURN
```

```
                                        .SBTTL  QPGDSS -- Get data set status for line
                                ;-----------------------------------------------------------------
                                ;  QPGDSS is called to get the data set (modem) status for a line
                                ;  connected to the quad serial line unit.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;
                                ;  Outputs:
                                ;   R0 = Generic data set status flags (MS$xxx)
                                ;
006054  010346                 QPGDSS: MOV     R3,-(SP)
006056  010546                         MOV     R5,-(SP)
                                ;
                                ;  Convert line number to address of control register
                                ;
006060  004737  006260'                 CALL    QPCVLX          ;Convert line # to address
                                ;
                                ;  At this point,
                                ;    R5 = Base address of registers for quad unit.
                                ;    R3 = Table index whose value depends on whether we are accessing
                                ;         lines 0 or 1 (DUART 0) or lines 2 and 3 (DUART 1)
                                ;         and the configuration 4/0 or 2/2.
                                ;
                                ;         Line  Config  Value
                                ;         ----  ------  -----
                                ;          0/2    4/0     0
                                ;          1/3    4/0     1
                                ;          0/2    2/2     2
                                ;          1/3    2/2     3
                                ;
                                ;  See if the phone is ringing
                                ;
006064  005000                          CLR     R0              ;Develop flags in R0
006066  136365  006144' 000132          BITB    QPFRI(R3),QPRIP(R5) ;Is the RI flag set for line?
006074  001402                          BEQ     1$              ;Br if not
006076  052700  0000000                 BIS     #MS$RNG,R0      ;Set ring status flag for return
                                ;
                                ;  See if carrier is detected
                                ;
006102  136365  006140' 000132  1$:     BITB    QPFCD(R3),QPRIP(R5) ;Is carrier detected?
006110  001402                          BEQ     2$              ;Br if not
006112  052700  0000000                 BIS     #MS$CAR,R0      ;Set carrier-detected flag for return
                                ;
                                ;  See if Data Terminal Ready is asserted
                                ;
006116  136363  006150' 006154' 2$:     BITB    QPFDTR(R3),QPOP(R3) ;Is DTR asserted?
006124  001402                          BEQ     3$              ;Br if not
006126  052700  0000000                 BIS     #MS$DTR,R0      ;Set DTR return flag
                                ;
                                ;  Finished
                                ;
006132  012605                  3$:     MOV     (SP)+,R5
006134  012603                          MOV     (SP)+,R3
006136  000207                          RETURN
                                ;
                                ;  Flag values to check Carrier Detect flag based on channel and config
```

QPGDSS -- Get data set status for line

```
                                                ;
006140  004     010     004     QPFCD:  .BYTE   4,10,4,0
006143  000
                                                ;
                                                ;  Flag values to check for line ringing
                                                ;
006144  000     000     010     QPFRI:  .BYTE   0,0,10,0
006147  000
                                                ;
                                                ;  Flag values to check for (or assert) Data Terminal Ready
                                                ;
006150  004     010     004     QPFDTR: .BYTE   4,10,4,0
006153  000
                                                ;
                                                ;  Shadow cells to hold value of output port settings
                                                ;
006154  000     000     000     QPOP:   .BYTE   0,0,0,0
006157  000
```

```
                                        .SBTTL  QPSDSS -- Set data set status
                                ;-----------------------------------------------------------------------
                                ;  QPSDSS is called to control the data set (modem) status for a line
                                ;  connected to the quad line unit.
                                ;
                                ;  Inputs:
                                ;   R1 = Physical line index number.
                                ;   R0 = Data set control flags (MS$xxx)
                                ;
006160  010346                 QPSDSS: MOV     R3,-(SP)
006162  010546                         MOV     R5,-(SP)
                                ;
                                ;  Convert line number to address of control register
                                ;
006164  004737  006260'                 CALL    QPCVLX          ;Convert line # to address
                                ;
                                ;  At this point,
                                ;    R5 = Base address of registers for quad unit.
                                ;    R3 = Table index whose value depends on whether we are accessing
                                ;         lines 0 or 1 (DUART 0) or lines 2 and 3 (DUART 1)
                                ;         and the configuration 4/0 or 2/2.
                                ;
                                ;         Line  Config  Value
                                ;         ----  ------  -----
                                ;          0/2    4/0     0
                                ;          1/3    4/0     1
                                ;          0/2    2/2     2
                                ;          1/3    2/2     3
                                ;
                                ;  See if we should raise or drop Data Terminal Ready
                                ;
006170  032700  0000000                 BIT     #MS$DTR,R0      ;Raise or drop DTR?
006174  001407                         BEQ     1$              ;Br to drop DTR
                                ;
                                ;  Raise DTR
                                ;
006176  116365  006150' 000134          MOVB    QPFDTR(R3),QPRSOP(R5);Raise bit in output port
006204  156363  006150' 006154'        BISB    QPFDTR(R3),QPOP(R3);Set bit in shadow cell
006212  000406                         BR      9$
                                ;
                                ;  Drop DTR
                                ;
006214  116365  006150' 000136  1$:     MOVB    QPFDTR(R3),QPRROP(R5);Drop bit in output port
006222  146363  006150' 006154'        BICB    QPFDTR(R3),QPOP(R3);Clear bit in shadow cell
                                ;
                                ;  Finished
                                ;
006230  012605                 9$:     MOV     (SP)+,R5
006232  012603                         MOV     (SP)+,R3
006234  000207                         RETURN
```

```
                                        .SBTTL  QPCVLA -- Convert line index into register addresses
                                ;-----------------------------------------------------------------
                                ;  QPCVLA is called to convert a TSX-Plus line index number of a line
                                ;  connected to the quad serial line unit into an address that corresponds
                                ;  to the base of the registers for the corresponding DUART and channel.
                                ;
                                ;  Inputs:
                                ;   R1 = TSX-Plus line index number
                                ;
                                ;  Outputs:
                                ;   R5 = Address of base of registers for DUART and channel
                                ;   (R0 is preserved)
                                ;
006236                          QPCVLA:
                                ;
                                ;  Determine which quad line this line is connected to
                                ;
006236  016105  0000006               MOV     LMXLN(R1),R5    ;Get line # within quad line unit (0-3)
                                ;
                                ;  Compute address of registers for this line
                                ;
006242  116505  006254'         2$:     MOVB    QPFADR(R5),R5   ;Get offset based on DUART and channel
006246  063705  000066'                 ADD     QPCSR,R5        ;Add base CSR address for module
                                ;
                                ;  Finished
                                ;
006252  000207                          RETURN
                                ;
                                ;  Table of register address offsets based on line number
                                ;
006254      000     020     040 QPFADR: .BYTE   0,20,40,60
006257      060
                                        .EVEN
```

```
                                        .SBTTL  QPCVLX -- Get address and configuration info
                                ;-------------------------------------------------------------
                                ;  QPCVLX is called to convert a TSX-Plus line number for a line connected
                                ;  to the quad serial line unit into two items of information:
                                ;  (1) the address of the base of the registers for the DUART to which
                                ;  the line is connected; (2) an index value in the range 0 to 3 which
                                ;  indicates which channel of the DUART the line is connected to and
                                ;  which configuration (2/2 or 4/0) the serial line unit is in.
                                ;
                                ;  Inputs:
                                ;   R1 = TSX-Plus line index number.
                                ;
                                ;  Outputs:
                                ;   R5 = Base address of registers for DUART.
                                ;   R3 = Channel and configuration index value as follows:
                                ;
                                ;         Line  Config  Value
                                ;         ----  ------  -----
                                ;          0/2    4/0      0
                                ;          1/3    4/0      1
                                ;          0/2    2/2      2
                                ;          1/3    2/2      3
                                ;
                                ;   (R0 is preserved)
                                ;
006260                          QPCVLX:
                                ;
                                ;  Determine which quad line this line is connected to
                                ;
006260  016103  000000                  MOV     LMXLN(R1),R3    ;Get line # within quad line unit (0-3)
                                ;
                                ;  Get the DUART base register address
                                ;
006264  116305  006314'         2$:     MOVB    QPFDAD(R3),R5   ;Get offset based on DUART 0 or 1
006270  063705  000066'                 ADD     QPCSR,R5        ;Add CSR address of base of module
                                ;
                                ;  Get index value based on line # and configuration
                                ;
006274  042703  177776                  BIC     #^C<1>,R3       ;Get 0 for lines 0&2, 1 for lines 1&3
006300  105737  000110'                 TSTB    QPMODM          ;Is this a 2/2 configuration?
006304  001402                          BEQ     9$              ;Br if not
006306  062703  000002                  ADD     #2,R3           ;Set index based on config
                                ;
                                ;  Finished
                                ;
006312  000207                  9$:     RETURN
                                ;
                                ;  Table of address offsets to base registers for DUARTs based on line #
                                ;
006314      000     000     040 QPFDAD: .BYTE   0,0,40,40
    006317      040
                                        .EVEN
                                        .ENDC   ;NE,QPASM       ;End of quad line unit code
                                ;
                                ;  Size of TSXPRO
                                ;
            006320              PROSIZ  =       .-PROBAS        ;Size of TSXPRO
```

```
     57          000001                                  .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  152 Words  ( 1 Pages)
Size of core pool:  17720 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:38.45
DK:TSXPRO,LP:TSXPRO=DK:TSXPRO.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

```
$DEAD     1-38      48-22
$HISTP    1-38      42-13     43-13
$OITIM    1-34      22-52     30-30     39-27     40-29     41-13     41-19     42-30     43-30     51-47
$PHONE    1-36      36-73     49-34
$SXOFF    1-41      42-23
$SXON     1-41      43-23
$VDCSR    3-16#     13-58     13-58*    21-35     21-39*    22-21*
$XCHAR    1-34      22-51     22-59     30-20     30-29     31-7      39-15     39-26     40-16     40-28     41-15     41-17
         42-17      42-29     43-17     43-29     50-21     51-46     51-54
CDCLOK    1-40      36-19*
CDGDSS    1-36      29-21*    36-22*    48-44*
CDSBRK    1-42      29-22*    36-24*    48-46*
CDSDSS    1-36      36-23*    48-45*
CDSSPD    1-42      29-23*    36-25*    48-47*
CDSTRT    1-36      19-19*    29-20*    36-18*    48-43*
CDSXOF    1-39      36-20*
CDSXON    1-39      36-21*
CDX$PC    1-37      36-14     36-18*    36-19*    36-20*    36-21*    36-22*    36-23*    36-24*    36-25*
CDX$PI    1-38      19-15     19-19*
CDX$PP    1-37      29-15     29-20*    29-21*    29-22*    29-23*
CDX$QP    1-44      48-43*    48-44*    48-45*    48-46*    48-47*    49-13
CORUSR    1-33      24-21
CP$7BR    8-63#     47-50
CP$7BT    8-83#     46-14     47-49
CP$8BR    8-64#     47-47
CP$8BT    8-84#     46-17     47-46
CP$ACM    8-21#
CP$ARP    8-20#
CP$AS     8-98#
CP$ASM    8-59#
CP$AW3    8-65#     36-46     47-42
CP$AW4    8-74#     36-44     47-30
CP$AW5    8-85#     36-48     46-14     47-43
CP$BR     8-94#     38-20
CP$BRP    8-107#
CP$BRR    8-13#     47-25*
CP$BW1    8-111#
CP$BW2    8-115#
CP$CAR    8-9#      36-36     36-70*    36-71*    37-26*    37-31*    37-36*    38-18*    38-19     38-23*    38-31*    39-16*
         39-17*     46-28*    46-29*    47-29
CP$CBR    8-10#     36-59     37-7*     37-8
CP$CCS    8-79#
CP$CD     8-139#    44-22
CP$CMM    8-73#
CP$CR     8-43#     36-37     36-60
CP$CRC    8-22#
CP$CS     8-131#
CP$CTS    8-140#
CP$DBR    8-8#      14-45     38-27     39-28*    40-30*    42-31*    43-31*
CP$DSR    8-142#
CP$DTR    8-130#    36-75     44-28     45-19     45-24
CP$EHP    8-61#
CP$EI     8-47#     37-36     38-31     39-17
CP$EIE    8-51#
CP$EIR    8-44#
CP$EOF    8-103#
```

Cross reference table (CREF V05.04)

```
CP$ER     8-46#     37-26
CP$EVN    8-70#     47-36
CP$IES    8-120#
CP$INP    8-90#
CP$IRC    8-121#
CP$ISR    8-122#
CP$ITE    8-119#
CP$LEN    8-82#
CP$LL     8-126#
CP$MOR    8-11#     36-69*    36-72*    36-75*    44-28     45-19*    45-24*
CP$M1R    8-12#     44-16     44-22
CP$MCO    8-133#
CP$MCV    8-16#     36-31*    36-32*
CP$MM     8-132#
CP$PAR    8-69#     47-33
CP$RC     8-99#
CP$RCA    8-89#
CP$RCE    8-60#
CP$RCL    8-62#
CP$REN    8-57#
CP$RES    8-42#     36-51     37-31     38-23
CP$RFE    8-102#
CP$RI     8-141#    44-16
CP$RIE    8-53#     36-71
CP$RL     8-127#
CP$ROE    8-101#
CP$RPE    8-100#
CP$RR0    8-35#     38-18
CP$RR1    8-36#
CP$RR2    8-37#     37-7
CP$RTI    8-45#     39-16
CP$RTS    8-129#    36-72
CP$RTV    8-15#     36-29*    36-30*
CP$SA     8-41#
CP$SB     8-81#     46-23
CP$SBS    8-71#
CP$SCL    8-58#
CP$SH     8-92#
CP$SMI    8-137#
CP$SMS    8-72#
CP$SRS    8-128#
CP$TBM    8-91#
CP$TCE    8-78#
CP$TEM    8-93#
CP$TEN    8-80#
CP$TI     8-138#
CP$TIE    8-52#     36-71
CP$WR0    8-27#
CP$WR1    8-28#     36-63     36-70
CP$WR2    8-29#     36-49     36-61
CP$WR3    8-30#     36-45     47-52
CP$WR4    8-31#     36-43     47-37
CP$WR5    8-32#     36-47     46-28     47-54
CP$WR6    8-33#
CP$WR7    8-34#
CPCINT   36-29      36-31     37-6#
```

Cross reference table (CREF V05.04)

```
CPCLOK  36-19    41-9#
CPGDSS  36-22    44-12#
CPIERR  37-15    37-16    37-17    37-18    37-36#
CPIESC  37-20    37-31#
CPINIT  14-47    36-9#
CPIREC  37-21    37-27    38-8#
CPISRC  37-22    37-26#
CPITR   37-19    39-8#
CPIVEC  37-11    37-15#
CPLINE   3-39#   36-13#   38-35    39-14
CPSBRK  36-24    46-10#
CPSDSS  36-23    45-10#
CPSSPD  36-25    36-55    47-10#
CPSTRT  36-18    40-7#    41-18
CPXOFF  36-20    42-9#
CPXON   36-21    43-9#
DORTI   12-7     12-24#
DOSCHD   1-33    24-17
DWTYPE   1-44
EMMAP    1-45    11-19
FORK     1-35     3-25    22-40
FP$PIO   1-41    22-41
FRKCQE   1-33    24-19
FRMERR   1-35    32-28    38-22    51-75
GETSLT   3-28    10-15    15-22    16-15    17-15    18-13#   48-38
GPRCSR   3-27    15-5     16-13#   48-12
GPRSLT  15-6     17-13#
GPRVEC  15-4     15-20#   48-35
GPVCSR  10-30    15-5#
GPVSLT  15-6#
GPVVEC  10-29    15-4#
ICOCSR   5-9#    29-53*   29-58*   29-59*   30-12*   30-35*   31-9*    31-10*   31-11*   32-46*   36-68*
ICODR    5-8#
IC1CSR   5-11#   48-73*
IC1DR    5-10#
IC2CSR   5-13#
IC2DR    5-12#
ILSW2    1-36    36-73    49-34*
IM$CHP   5-39#
IM$CLK   5-56#
IM$CM    5-32#   30-35    31-11
IM$CMB   5-43#
IM$CPD   5-52#   36-68
IM$CPM   5-53#
IM$CR    5-36#   32-46
IM$CRM   5-30#   29-53    36-68    48-71
IM$CS    5-41#
IM$GIP   5-20#
IM$IM    5-19#
IM$IRP   5-21#
IM$KBR   5-50#
IM$KBT   5-51#
IM$LMB   5-42#
IM$MM    5-23#
IM$OM    5-33#
IM$OR    5-37#
```

Cross reference table (CREF V05.04)

```
IM$PA     5-45#
IM$PM     5-17#
IM$PPR    5-54#    29-53    32-46
IM$PPT    5-55#    29-58    29-59    30-12    30-35    31-9     31-10    31-11
IM$PR     5-44#
IM$PRM    5-46#
IM$RP     5-22#
IM$RST    5-28#
IM$SM     5-34#    29-58    30-12    31-9
IM$SR     5-38#    29-59    31-10
IM$STV    5-24#
IM$VS     5-18#
IM$ZM     5-31#
IM$ZR     5-35#
IM$ZRM    5-29#
IM$ZS     5-40#
INISLT   10-9      11-6#
INTEN     1-35     22-9     25-10    26-9     27-10
INTPRI    1-36     11-38    21-45    23-25    30-7     30-8*    30-39*   35-38    39-9     39-10*   39-32*   40-34
         42-35     43-35    46-30    47-56    50-31    52-51
INVEC     1-39     14-27
IOFIN     1-41     28-37
ITRMTP    1-42     19-23    19-25*
KBDCHR    3-24     26-37#
KPAR5     1-40     11-16    11-17*   11-36*   13-8     13-15*   13-97*   21-6     21-7*    21-44*   22-19*   22-46*
         25-19*    26-18*   26-42    26-52*   27-19*   28-27*
KPAR6     1-40     13-9     13-37*   13-96*   26-43    26-51*
LCDTYP    1-37     19-15*   29-15*   36-14*   49-13*
LMXLN     1-45     49-22    56-18    57-30
LMXNUM    1-45     14-19    48-20    49-24
LMXPRM    1-34     29-39    35-13*   36-52    46-15    47-16*   52-15*
LOUTIR    1-39     21-20    22-33    23-24    30-24    39-21    40-23    51-41
LP$7BT    1-43     35-18    46-15    47-44    52-36
LP$ODD    1-43     35-26    47-34    52-45
LP$PAR    1-43     35-23    47-31    52-41
LP$SPD    1-43     35-35    47-21    52-27
LSTHL     1-38     48-19
LSW10     1-41     42-13*   42-23*   43-13*   43-23*
LSW3      1-34     22-51*   22-59*   30-20*   30-29*   31-7     39-15*   39-26*   40-16    40-28*   41-15    41-17*
         42-17     42-29*   43-17    43-29*   48-22*   50-21    51-46*   51-54*
LSW5      1-34     22-52*   30-30*   39-27*   40-29*   41-13    41-19*   42-30*   43-30*   51-47*
LXCL      1-33     24-24
MAXPOC    1-23#     3-48    20-38    22-58
MMENBL    1-45     11-18    11-37
MS$BRK    1-42     34-14    46-21    53-21
MS$CAR    1-37     44-24    54-43
MS$DTR    1-37     44-30    45-14    54-49    55-32
MS$RNG    1-37     44-18    54-37
MXCSR     1-46     49-25*
MXVEC     1-46     49-26*
OVRRUN    1-35     32-31    51-78
PIBASE    3-40#    13-14*   13-15    13-36    13-52    21-7     22-19    22-46    25-19    26-18    27-19    28-27
PICQE    28-13#    28-31    28-36
PIDRIV    1-29     20-34#
PIENTR   28-14     28-23#
PIGO     20-23     20-42    21-6#
```

Cross reference table (CREF V05.04)

```
PIGOFL    3-20#     21-11
PIHAN     1-28      28-9#
PIINAD    3-9#      13-33
PIINIT   14-29      19-10#
PIINT    28-10      28-19#
PIINWD    3-10#     13-38
PIKIIR   13-64      26-5#
PIKOIR   13-68      27-6#
PILINE    3-37#     19-14*    21-18     22-27     23-12     24-23     26-45
PILQE    28-12#
PIMOC     3-48#     20-38*    22-58*    24-30*
PINDCH    3-26      23-11#
PIOIFL    3-41#     21-31     21-33*    22-15*    22-65*
PIQUIT   20-18      22-28     23-18     24-13#
PISCNT    3-42#     20-13*    20-46*
PISRT     1-40      13-13
PISTRT   19-19      20-9#
PIVFIR   13-76      25-6#
PIVTIR   13-66      13-78     22-5#
PIXEOF    3-12#     25-20
PIXICH    3-13#     26-19
PIXIOI    3-14#     27-20
PIXIOQ    3-15#     28-32
PIXOCH    3-11#     22-53
PP$7BT    7-30#     35-22
PP$8BT    7-31#     35-20
PP$BRS    7-39#
PP$CMF    7-51#     10-47     29-49
PP$CMR    7-11#     10-47*    29-34     29-49*    32-37*    34-19*    34-24*    35-30
PP$DBR    7-8#      14-38     30-31*    32-18
PP$DSR    7-25#
PP$DTR    7-45#
PP$EVN    7-33#     35-28
PP$FB     7-47#     34-19     34-24
PP$FE     7-24#     32-26
PP$LEN    7-29#
PP$M1F    7-35#     29-35     35-17
PP$M2F    7-40#     35-36
PP$MDR    7-10#     29-35*    35-31*    35-37*
PP$OE     7-23#     32-29
PP$OM     7-50#
PP$PAR    7-32#     35-25
PP$PE     7-22#     32-23
PP$RCS    7-12#     10-34
PP$RCV    7-15#     29-27*    29-28*
PP$RD     7-21#
PP$RE     7-48#     10-47     29-49     32-37
PP$REN    7-46#
PP$RTS    7-49#
PP$SBL    7-34#
PP$STR    7-9#      32-23     32-26     32-29
PP$TCS    7-13#     10-35
PP$TEN    7-44#
PP$TR     7-20#
PP$TRV    7-16#     29-29*    29-30*
PPQDSS   29-21      33-13#
```

Cross reference table (CREF V05.04)

```
PPINIT  14-40    29-10#
PPLINE   3-38#   29-14*   30-16    32-14
PPRINT  29-27    32-5#
PPSBRK  29-22    34-10#
PPSSPD  29-23    29-45    35-9#
PPSTRT  29-20    31-7#
PPTERM   1-38    10-37*   29-42
PPTINT  29-29    30-5#
PRIOO    3-18#   21-37
PROBAS   1-7#    57-56
PROBRK   1-38    10-42
PROBUF   1-46    13-41    13-54
PROHAN   1-28    13-5#
PROINI   1-28    10-5#
PROLIN   1-28    14-14#
PRONOP   1-28    12-6#
PROSIZ   1-28    57-56#
PROSLT   1-34    11-28    18-15    18-20
PROVEC   3-5#    13-20
PSSREG   4-5#    10-44*   10-46*
PSW      1-36    11-15*   11-38*   21-19*   21-27*   21-45*   23-23*   23-25*   35-29*   35-38*   40-12*   40-34*
        42-9*    42-35*   43-9*    43-35*   46-27*   46-30*   47-51*   47-56*   50-20*   50-31*   52-48*   52-51*
QP$1SB   9-60#
QP$22C   9-44#   48-30
QP$2SB   9-61#
QP$40C   9-45#
QP$7BR   9-50#   52-38
QP$8BR   9-51#   52-40
QP$AWA   9-95#   48-56    48-57
QP$AWI   9-110#  48-61    48-62
QP$BCA   9-101#
QP$BCB   9-105#
QP$BEM   9-54#
QP$BRK   9-91#   51-73
QP$CNR   9-102#
QP$DRX   9-68#   49-55
QP$DTX   9-70#   49-55    51-55
QP$EBT   9-80#   53-31
QP$ERX   9-67#   49-57
QP$ETX   9-69#   49-57    50-27
QP$FFL   9-85#
QP$ID    9-6#    48-11    48-34    48-37
QP$IEN   9-46#   13-91    48-33    51-21    51-99
QP$IPC   9-106#
QP$NPR   9-53#   52-43
QP$ODD   9-52#   52-47
QP$RAR   9-100#
QP$RBC   9-78#
QP$RBR   9-104#
QP$RDN   9-84#
QP$RES   9-77#   49-56
QP$RFE   9-90#   51-73
QP$ROE   9-88#   51-76
QP$RPE   9-89#   51-70
QP$RPR   9-74#   49-45    52-49
QP$RRX   9-75#   49-55
```

Cross reference table (CREF V05.04)

```
QP$RTX    9-76#     49-57
QP$RXR    9-56#
QP$SBI    9-79#     53-26
QP$SFL    9-55#
QP$TAR    9-99#
QP$TBR    9-103#
QP$TCS    9-62#
QP$TDN    9-86#
QP$TEM    9-87#
QP$TRS    9-63#
QPASM     1-22#     10-52     13-85     14-18     48-1
QPCINT   13-88      51-6#
QPCSR     3-46#     13-90     48-29*    49-17     49-25     51-20     51-98     56-23     57-35
QPCVLA   49-38      50-13     52-19     53-14     56-14#
QPCVLX   54-17      55-15     57-26#
QPFADR   56-22      56-31#
QPFCD    54-41      54-59#
QPFDAD   57-34      57-50#
QPFDTR   54-47      54-67#    55-37     55-38     55-43     55-44
QPFRI    54-35      54-63#
QPFRXR   51-59      51-115#
QPFTXR   51-35      51-110#
QPGDSS   48-44      54-12#
QPINIT   10-53      48-7#
QPLINE   14-21      49-9#
QPLX      3-59#     49-23*    51-30
QPMODM    3-60#     48-32*    49-30     57-40
QPOP     54-47      54-71#    55-38*    55-44*
QPRACR    9-27#     48-56*    48-57*
QPRCOF    9-37#     51-89
QPRCRA    9-25#     49-45*    49-55*    49-56*    49-57*    50-27*    51-55*    52-49*    53-26*    53-31*
QPRCSA    9-24#     49-51*    52-32*
QPRCSB    9-29#
QPRIMR    9-28#     48-61*    48-62*
QPRIP     9-20#     54-35     54-41
QPRISR    9-16#     51-35     51-59
QPRMRA    9-13#     49-46*    49-47*    52-50*
QPRMRB    9-17#
QPRMSR    9-12#     13-91*    48-30     48-33*    51-21*    51-99*
QPROPC    9-31#     48-51*    48-52*
QPRROP    9-33#     48-66*    48-67*    55-43*
QPRRRA    9-15#     51-65
QPRRRB    9-19#
QPRSOP    9-32#     55-37*
QPRSRA    9-14#     51-64
QPRSRB    9-18#
QPRTRA    9-26#     51-48*
QPRTRB    9-30#
QPRUOF    9-38#     48-52*    48-57*    48-62*    48-67*    51-92
QPSBRK   48-46      53-10#
QPSDSS   48-45      55-10#
QPSLOT    3-61#     48-39*    48-72
QPSPD    52-28      52-62#
QPSSPD   48-47      52-10#
QPSTRT   48-43      50-9#
QPVEC     3-47#     13-86     48-36*    49-26
```

Cross reference table (CREF V05.04)

```
R50PI     3-43#
RBERR     1-35      32-25     32-28     32-31     32-35     51-72     51-75     51-78
RCVPAR    1-35      32-25     51-72
REENAB    3-17#     21-40*    22-20*    22-61*
RPRCSR    1-44      10-30*
RPRVEC    1-44      10-29*
RSR       1-36      14-34
RT$BAS    1-41      13-14
S4800     1-43      29-44
S9600     1-39      29-41     36-54
SROMMR    1-45      11-18*    11-37*
SR3MMR    1-45      11-19*
SS$BNK    4-9#
SS$BRK    4-11#     10-44     10-46
SS$MON    4-10#
TSXPRO    1-6#       1-27
TTINPT    1-35      26-46     32-41     38-39     51-82
TTRSAV    1-42      32-9      38-12     51-14
VDFLAG    3-19#     21-34*    22-60*
VIDCSR    1-44      10-24*
VIDSLT    3-44#     10-17*    10-21
VIDTBL    3-52#     10-13
VIDVEC    3-45#     10-20*    12-13     13-75
VP$CME    6-13#
VP$COO    6-12#
VP$DOI    6-15#     21-35     21-39     22-21
VP$EFI    6-10#
VP$EOF    6-11#
VP$IMD    6-8#
VP$LMD    6-7#
VP$OEF    6-9#
VP$OMP    6-14#
VP$TRD    6-16#
VPAR5     1-40      11-23     13-19     13-23     13-34     13-39     13-53
VPAR6     1-40      13-40
VSWPSL    1-40
VT100     1-42      19-25
```

Cross reference table (CREF V05.04)