

## Table of contents

2-	1	Macros
3-	1	Data areas
4-	1	System control register
5-	1	Interrupt control values
6-	1	Video display control values
7-	1	Printer port control values
8-	1	Communications port control values
9-	1	Quad Serial Line Unit control values
10-	1	PROINI -- General initialization for Pro
11-	1	INISLT -- Set up information about option slots
12-	1	PRONOP -- Disable PRO interrupts
13-	1	PROHAN -- Initialize the PI handler
14-	1	PROLIN -- See if line is a special PRO terminal
15-	8	GPRVEC -- Get address of vector for PRO device
16-	1	GPRCSR -- Get address of CSR for PRO device
17-	1	GPRSLT -- Get Slot number for PRO device
18-	1	GETSLT -- Determine which option slot has device controller
19-	1	*** Console Control Routines ***
19-	2	PIINIT -- Initialize the console
20-	1	PISTRT -- Start output to video screen
20-	29	PIDRIV -- Clock driven PI transmitter routine
21-	1	PIGO -- Start PI transmitter
22-	1	PIVTIR -- Video end of transfer interrupt
23-	1	PINDCH -- Get next output character for console
24-	1	PIQUIT -- Check output character limit
25-	1	PIVFIR -- Video end of frame interrupt
26-	1	PIKIIR -- Keyboard input interrupt
26-	29	KBDCHR -- Process character received from keyboard
27-	1	PIKOIR -- Output interrupt for keyboard
28-	1	PIHAN -- Simulated PI handler
29-	1	*** Printer Port Control Routines ***
29-	2	PPINIT -- Initialize the printer port
30-	1	PPTINT -- Transmitter interrupt
31-	1	PPSTRT -- Start output to printer port
32-	1	PPRINT -- Receiver interrupt
33-	1	PPGDSS -- Get data set status for printer port
34-	1	PPSBRK -- Control break transmission
35-	1	PPSSPD -- Set transmission speed for printer port
36-	1	*** Communications Port Control Routines ***
36-	2	CPINIT -- Communications port initialization
37-	1	CPCINT -- Receiver/Transmitter interrupt routine
38-	1	CPIREC -- Received character from comm port
39-	1	CPITR -- Transmitter interrupt
40-	1	CPSTRT -- Start output to communications port
41-	1	CPCLOK -- Timer driven routine for communications port
42-	1	CPXOFF -- Transmit XOFF to communications port
43-	1	CPXON -- Transmit XON to communications port
44-	1	CPGDSS -- Get communications port data set status
45-	1	CPSDSS -- Set data set status for communications port
46-	1	CPSBRK -- Control break transmission on communications port
47-	1	CPSSPD -- Set speed for communications port
48-	2	*** Quad Serial Line Unit Routines ***
48-	3	QPINIT -- Initialize quad serial line unit
49-	1	QPLINE -- Initialize a line connected to quad line unit
50-	1	QPSTRT -- Start output
51-	1	QPCINT -- Interrupt from quad line unit
52-	1	QPSSPD -- Set speed for quad line port

53-	1	QPSBRK	-- Control break transmission to quad line unit
54-	1	QPGDSS	-- Get data set status for line
55-	1	QPSDSS	-- Set data set status
56-	1	QPCVLA	-- Convert line index into register addresses
57-	1	QPCVLX	-- Get address and configuration info

```

1          .TITLE  TSXPRO -- TSX-Plus PROxxx Routines
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  QBL
5 000000   .CSECT  TSXPRO
6 000000   TSXPRO:
7 000000   PROBAS:
8          ;-----
9          ; TSX-Plus interface routines for Digital PRO-3xx series
10         ; personal computers.
11         ;
12         ; Copyright (c) 1984, 1985.
13         ; S&H Computer Systems, Inc.
14         ; Nashville, Tennessee 37212
15         ; U. S. A.
16         ;
17         ; Assembly parameters
18         ;
19         ; Set QPASM to 1 to include support for quad serial line unit.
20         ; Set QPASM to 0 to exclude quad line unit code.
21         ;
22         ; QPASM = 1 ; Select quad port code
23         ; MAXPOC = 15 ; Max consecutive output chars to PI
24         ;
25         ; Global definitions
26         ;
27         .GLOBL  TSXPRO
28         .GLOBL  PROSIZ, PROLIN, PROINI, PROHAN, PIHAN, PRONOP
29         .GLOBL  PIDRIV
30         ;
31         ; Global references
32         ;
33         .GLOBL  FRKCQE, DOSCHD, CORUSR, LXCL
34         .GLOBL  LMXPRM, $XCHAR, LSW3, $OITIM, LSW5, PROSLT
35         .GLOBL  FORK, INTEN, TTINPT, RBERR, OVRRUN, FRMERR, RCVPAR
36         .GLOBL  RSR, PSW, INTPRI, CDSTRT, CDGDSS, CDSOSS, $PHONE, ILSW2
37         .GLOBL  LCDTYP, CDX$PC, CDX$PP, MS$DTR, MS$CAR, MS$RNG
38         .GLOBL  PROBRK, $DEAD, PPTERM, LSTHL, CDX$PI, $HISTP
39         .GLOBL  INVEC, LOUTIR, S9600, CDSXOF, CDSXON
40         .GLOBL  KPAR5, KPAR6, VPAR5, VPAR6, PISRT, CDCLOK, VSWPSL
41         .GLOBL  RT$BAS, IOFIN, FP$PIO, LSW10, $SXON, $SXOFF
42         .GLOBL  MS$BRK, CDSBRK, CDSSPD, VT100, ITRMTP, TTRSAV
43         .GLOBL  LP$SPD, LP$PAR, LP$ODD, LP$7BT, S4800
44         .GLOBL  RPRCSR, RPRVEC, DWTYPE, VIDCSR, CDX$QP
45         .GLOBL  SR0MMR, SR3MMR, MMENBL, EMMAP, LMXNUM, LMXLN
46         .GLOBL  MXVEC, MXCSR, PROBUF

```

Macros

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13

.SBTTL Macros

-----  
; Macro to disable interrupts

; .MACRO DISABL  
; BIS #340,@#PSW  
; .ENDM DISABL

; Macro to enable interrupts

; .MACRO ENABL  
; BIC INTPRI,@#PSW  
; .ENDM ENABL

Data areas

```

1          .SBTTL  Data areas
2          ;-----
3          ; Vector of addresses used to move information between TSXPRO and PI handler.
4          ;
5 000000  PROVEC:
6          ;
7          ; Addresses passed from the PI handler to TSXPRO
8          ;
9 000000  000000  PIINAD: .WORD  0          ;Address of initialization code
10 000002  000000  PIINWD: .WORD  0          ;Number of words of initialization code
11 000004  000000  PIXOCH: .WORD  0          ;Routine called to send a char to the video
12 000006  000000  PIXEOF: .WORD  0          ;Routine called for end-of-frame interrupts
13 000010  000000  PIXICH: .WORD  0          ;Routine to process keyboard input interrupts
14 000012  000000  PIXIOI: .WORD  0          ;Routine to process keyboard output interrupts
15 000014  000000  PIXIOQ: .WORD  0          ;Routine to process an I/O queue request
16 000016  000000  $VDCSR: .WORD  0          ;Address of video CSR address
17 000020  000000  REENAB: .WORD  0          ;Pointer to REENAB cell
18 000022  000000  PRIOO:  .WORD  0          ;Pointer to PRIOO cell
19 000024  000000  VDFLAG: .WORD  0          ;Pointer to VDFLAG cell
20 000026  000000  PIGDFL: .WORD  0          ;Pointer to PIGDFL cell (1==>PI run, 0==>stop)
21          ;
22          ; Addresses passed from TSXPRO to the PI handler
23          ;
24 000030  002156' .WORD  KBDCHR          ;Routine to process an input character
25 000032  0000000 .WORD  FORK          ;Address of system .FORK routine
26 000034  001740' .WORD  PINDCH          ;Routine to get next char to send
27 000036  001172' .WORD  GPRCSR          ;Routine to get CSR address for device
28 000040  001254' .WORD  GETSLT          ;Routine to get option slot # for device
29          ;
30          ; End of address vector
31          ;
32 000042  000000  .WORD  0          ;End of PROVEC vector
33          ;
34          ;-----
35          ; Data areas:
36          ;
37 000044  000000  PILINE: .WORD  0          ;# of line connected to Pro console
38 000046  000000  PPLINE: .WORD  0          ;# of line connected to printer port
39 000050  000000  CPLINE: .WORD  0          ;# of line connected to communications port
40 000052  000000  PIBASE: .WORD  0          ;Base 64-byte block # of PI handler
41 000054  000000  PIOIFL: .WORD  0          ;Non-zero ==> Doing PI output interrupt
42 000056  177777  PISCNT: .WORD -1          ;Counts output starts per clock period
43 000060  062550  R5OPI:  .RAD50  /PI /          ;Name of PI handler
44 000062  000000  VIDSLT: .WORD  0          ;Option slot # where video controller is
45 000064  000000  VIDVEC: .WORD  0          ;Address of video vector A
46 000066  000000  QPCSR:  .WORD  0          ;Address of base of registers for quad port
47 000070  000000  QPVEC:  .WORD  0          ;Address of vector for quad port
48 000072  000017  PIMOC:  .WORD  MAXPOC          ;Limit consecutive output chars to PI
49          ;
50          ; Table of possible video device ID values
51          ;
52 000074  001002  VIDTBL: .WORD  1002          ;350 video
53 000076  000050  .WORD  50          ;380 video
54 000100  010050  .WORD  10050          ;380 video with EBO
55 000102  002002  .WORD  2002          ;IVIS video
56          ;
57          ; 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
```

```
1          .SBTTL System control register
2          ;-----
3          ; Control values in system status register
4          ;
5          173700 PSSREG = 173700 ;Address of status register
6          ;
7          ; Flags in status register
8          ;
9          000007 SS#BNK = 7 ;Mask for memory bank information
10         000020 SS#MON = 20 ;Monitor-present flag
11         000200 SS#BRK = 200 ;Break enable flag for diagnostic port
```

Interrupt control values

```

1          .SBTTL  Interrupt control values
2          ;-----
3          ; The following values relate to the interrupt control system on the
4          ; Professional computer.
5          ;
6          ; Addresses of control registers
7          ;
8          173200 ICODR  =      173200      ; Interrupt controller 0 data register
9          173202 ICOCRSR =      173202      ; Interrupt controller 0 control register
10         173204 IC1DR   =      173204      ; Interrupt controller 1 data register
11         173206 IC1CSR  =      173206      ; Interrupt controller 1 control register
12         173210 IC2DR   =      173210      ; Interrupt controller 2 data register
13         173212 IC2CSR  =      173212      ; Interrupt controller 2 control register
14         ;
15         ; Interrupt Mode Register
16         ;
17         000001 IM$PM   =      1          ; Priority mode (0=fixed, 1=rotating)
18         000002 IM$VS   =      2          ; Vector selection
19         000004 IM$IM   =      4          ; Interrupt mode (1==>Do not interrupt)
20         000010 IM$GIP  =     10          ; Group interrupt polarity
21         000020 IM$IRP  =     20          ; Interrupt request polarity
22         000140 IM$RP   =    140          ; Mask for register preselect values
23         000200 IM$MM   =    200          ; Master mask (1==>Enable group interrupts)
24         000200 IM$STV  =    200          ; Standard value for TSX-Plus
25         ;
26         ; Command values that can be stored into CSR register
27         ;
28         000000 IM$RST  =      000          ; Reset
29         000020 IM$ZRM  =      020          ; Clear bits in IRR and IMR registers
30         000030 IM$CRM  =      030          ; Clear single bit in IRR and IMR registers
31         000040 IM$ZM   =      040          ; Clear all bits in IMR register
32         000050 IM$CM   =      050          ; Clear single bit in IMR register
33         000060 IM$OM   =      060          ; Set all bits in IMR register to one
34         000070 IM$SM   =      070          ; Set single bit in IMR register
35         000100 IM$ZR   =     100          ; Clear all bits in IRR register
36         000110 IM$CR   =     110          ; Clear single bit in IRR register
37         000120 IM$OR   =     120          ; Set all bits in IRR register to ones
38         000130 IM$SR   =     130          ; Set single bit in IRR register
39         000140 IM$CHP  =     140          ; Clear highest priority ISR bit
40         000160 IM$ZS   =     160          ; Clear ISR register to zero
41         000170 IM$CS   =     170          ; Clear single bit in ISR register
42         000200 IM$LMB  =     200          ; Load mode bits
43         000240 IM$CMB  =     240          ; Control mode bits
44         000260 IM$PR   =     260          ; Preselect IMR register
45         000300 IM$PA   =     300          ; Preselect ACR register
46         000340 IM$PRM  =     340          ; Preselect response memory
47         ;
48         ; Values used with command codes to select specific interrupts
49         ;
50         000001 IM$KBR  =      1          ; ICO - Keyboard receiver interrupt
51         000002 IM$KBT  =      2          ; ICO - Keyboard transmitter interrupt
52         000003 IM$CPD  =      3          ; ICO - Communications port data transfer
53         000004 IM$CPM  =      4          ; ICO - Communications port modem change
54         000005 IM$PPR  =      5          ; ICO - Printer port receiver interrupt
55         000006 IM$PPT  =      6          ; ICO - Printer port transmitter interrupt
56         000007 IM$CLK  =      7          ; ICO - Clock interrupt

```



Video display control values

```

1          .SBTTL  Video display control values
2          ;-----
3          ; The following values relate to the video display.
4          ;
5          ; Control flags in Constrol/status register
6          ;
7          000001 VP$LMD  =      1          ;Line mode definition
8          000002 VP$IMD  =      2          ;Interlace mode definition
9          000040 VP$OEF  =     40          ;Odd/Even frame flag
10         000100 VP$EFI  =     100         ;End of frame interrupt enable
11         000200 VP$EOF  =     200         ;End of frame flag
12         001400 VP$COD  =    1400        ;Mask for class of operation
13         002000 VP$CME  =     2000       ;Color map enable
14         010000 VP$OMP  =    10000      ;Option module presence flag
15         040000 VP$DDI  =    40000      ;Done interrupt enable
16         100000 VP$TRD  =   100000      ;Transfer done flag

```

Printer port control values

```

1          .SBTTL  Printer port control values
2          ;-----
3          ; The following values relate to the "printer port" which can optionally
4          ; be used as a time-sharing terminal under TSX-Plus.
5          ;
6          ; Register addresses and vectors
7          ;
8          173400 PP$DBR = 173400 ;Data buffer register
9          173402 PP$STR = 173402 ;Status register
10         173404 PP$MDR = 173404 ;Mode registers
11         173406 PP$CMR = 173406 ;Command register
12         177560 PP$RCS = 177560 ;Receiver CSR for maintenance mode
13         177564 PP$TCS = 177564 ;Transmitter CSR for maintenance mode
14         ;
15         000220 PP$RCV = 220 ;Receiver vector
16         000224 PP$TRV = 224 ;Transmitter vector
17         ;
18         ; Flags in the status register
19         ;
20         000001 PP$TR = 1 ;Transmitter ready
21         000002 PP$RD = 2 ;Receive done
22         000010 PP$PE = 10 ;Parity error detected
23         000020 PP$OE = 20 ;Overrun error
24         000040 PP$FE = 40 ;Framing error
25         000200 PP$DSR = 200 ;Data set ready
26         ;
27         ; Flags in mode register 1
28         ;
29         000014 PP$LEN = 14 ;Mask for character length
30         000010 PP$7BT = 10 ;7 bit characters
31         000014 PP$8BT = 14 ;8 bit characters
32         000020 PP$PAR = 20 ;Enable parity control
33         000040 PP$EVN = 40 ;1==>even parity, 0==>odd parity
34         000300 PP$SBL = 300 ;Mask for stop bit length
35         000102 PP$M1F = 102 ;Standard mode register 1 value for TSX-Plus
36         ;
37         ; Flags in mode register 2
38         ;
39         000017 PP$BRS = 17 ;Mask for baud rate select field
40         000260 PP$M2F = 260 ;Standard mode register 2 value for TSX-Plus
41         ;
42         ; Command register
43         ;
44         000001 PP$TEN = 1 ;Transmitter enable
45         000002 PP$DTR = 2 ;Data terminal ready
46         000004 PP$REN = 4 ;Receiver enable
47         000010 PP$FB = 10 ;Force break transmission
48         000020 PP$RE = 20 ;Reset error
49         000040 PP$RTS = 40 ;Request to send
50         000300 PP$DM = 300 ;Operating mode mask
51         000047 PP$CMF = 47 ;Standard command register value for TSX-Plus

```

```

1          .SBTTL  Communications port control values
2          ;-----
3          ; Control values for the communications port which may be used as
4          ; a TSX-Plus time-sharing line.
5          ;
6          ; Control registers and vectors
7          ;
8          173300 CP$DBR = 173300 ;Data buffer register
9          173302 CP$CAR = 173302 ;Control/status register A
10         173306 CP$CBR = 173306 ;Control/status register B
11         173310 CP$MOR = 173310 ;Modem control register 0
12         173312 CP$M1R = 173312 ;Modem control register 1
13         173314 CP$BRR = 173314 ;Baud rate register
14         ;
15         000210 CP$RTV = 210 ;Receive/transmit vector
16         000214 CP$MCV = 214 ;Modem change vector
17         ;
18         ; Control/status Register A
19         ;
20         000007 CP$ARP = 7 ;Mask to select read/write sub-register
21         000070 CP$ACM = 70 ;Mask for command bits
22         000300 CP$CRC = 300 ;Mask for CRC control bits
23         ;
24         ; Read/Write sub-register select values
25         ; (These values are stored into CP$ARP and CP$BRP to select sub registers)
26         ;
27         000000 CP$WRO = 0 ;Write register 0
28         000001 CP$WR1 = 1 ;Write register 1
29         000002 CP$WR2 = 2 ;Write register 2
30         000003 CP$WR3 = 3 ;Write register 3
31         000004 CP$WR4 = 4 ;Write register 4
32         000005 CP$WR5 = 5 ;Write register 5
33         000006 CP$WR6 = 6 ;Write register 6
34         000007 CP$WR7 = 7 ;Write register 7
35         000000 CP$RRO = 0 ;Read register 0
36         000001 CP$RR1 = 1 ;Read register 1
37         000002 CP$RR2 = 2 ;Read register 2
38         ;
39         ; Command values for WRO under control/status register A
40         ;
41         000010 CP$SA = 10 ;Send abort
42         000020 CP$RES = 20 ;Reset external interrupt
43         000030 CP$CR = 30 ;Channel reset
44         000040 CP$EIR = 40 ;Enable interrupt on next char received
45         000050 CP$RTI = 50 ;Reset transmitter interrupt pending
46         000060 CP$ER = 60 ;Error reset
47         000070 CP$EI = 70 ;End of interrupt
48         ;
49         ; Sub-register WR1 under control/status register A
50         ;
51         000001 CP$EIE = 1 ;External interrupt enable
52         000002 CP$TIE = 2 ;Transmitter interrupt enable
53         000030 CP$RIE = 30 ;Mask for receiver interrupt enable flags
54         ;
55         ; Sub-register WR3 under control/status register A
56         ;
57         000001 CP$REN = 1 ;Receiver enable

```

Communications port control values

```

58      000002      CP$SCL =      2      ; Sync character load inhibit
59      000004      CP$ASM =      4      ; Address search mode
60      000010      CP$RCE =     10      ; Receiver CRC enable
61      000020      CP$EHP =     20      ; Enter hunt phase
62      000300      CP$RCL =    300      ; Receiver character length
63      000100      CP$7BR =     100      ; 7 bit characters
64      000300      CP$8BR =     300      ; 8 bit characters
65      000001      CP$AW3 =      001      ; Standard value for TSX-Plus
66      ;
67      ; Sub-register WR4 under control/status register A
68      ;
69      000001      CP$PAR =      1      ; Parity enable
70      000002      CP$EVN =      2      ; 1==>even parity, 0==>odd parity
71      000014      CP$SBS =     14      ; Mask for stop bits select value
72      000060      CP$SMS =     60      ; Mask for synchronous mode control flags
73      000300      CP$CMM =    300      ; Mask for clock mode value
74      000104      CP$AW4 =     104      ; Standard value for TSX-Plus
75      ;
76      ; Sub-register WR5 under control/status register A
77      ;
78      000001      CP$TCE =      1      ; Transmitter CRC enable
79      000004      CP$CCS =      4      ; CRC polynomial select
80      000010      CP$TEN =     10      ; Transmitter enable
81      000020      CP$SB  =     20      ; Send break
82      000014      CP$LEN =     14      ; Transmitter character length
83      000040      CP$7BT =     040      ; 7 bit characters
84      000140      CP$8BT =     140      ; 8 bit characters
85      000010      CP$AW5 =     010      ; Standard value for TSX-Plus
86      ;
87      ; Sub-register RRO under control/status register A
88      ;
89      000001      CP$RCA =      1      ; Receive character available
90      000002      CP$INP =      2      ; Interrupt pending
91      000004      CP$TBM =      4      ; Transmit buffer empty
92      000020      CP$SH  =     20      ; Sync/hunt
93      000100      CP$TEM =     100      ; Transmitter underrun/end of message
94      000200      CP$BR  =     200      ; Break received
95      ;
96      ; Sub-register RR1 under control/status register A
97      ;
98      000001      CP$AS  =      1      ; All sent -- Transmitter ready for next char
99      000016      CP$RC  =     16      ; Mask for residue codes
100     000020      CP$RPE =     20      ; Received parity error
101     000040      CP$ROE =     40      ; Receiver overrun error
102     000100      CP$RFE =    100      ; Framing error
103     000200      CP$EOF =    200      ; End of frame
104     ;
105     ; Control/status Register B
106     ;
107     000007      CP$BRP =      7      ; Mask to select sub-register
108     ;
109     ; Sub-register WR1 under control/status register B
110     ;
111     000004      CP$BW1 =      4      ; Standard value for TSX-Plus
112     ;
113     ; Sub-register WR2 under control/status register B
114     ;

```

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$MCO  =      30         ;Standard value for TSX-Plus
134      ;
135      ; Modem Control Register 1
136      ;
137      000004      CP$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

```

Quad Serial Line Unit control values

```

1          .SBTTL Quad Serial Line Unit control values
2          ;-----
3          ;
4          ; Device ID for quad serial line unit
5          ;
6          000064 QP$ID = 64 ;Device ID for quad serial line unit
7          ;
8          ; Addresses of registers relative to base address for module
9          ;
10         ; Read registers
11         ;
12         000004 QPRMSR = 004 ;Module status register
13         000100 QPRMRA = 100 ;Mode register A
14         000102 QPRSRA = 102 ;Status register A
15         000106 QPRRRA = 106 ;Receiver register A
16         000112 QPRISR = 112 ;Interrupt status register
17         000120 QPRMRB = 120 ;Mode register B
18         000122 QPRSRB = 122 ;Status register B
19         000126 QPRRRB = 126 ;Receiver register B
20         000132 QPRIP = 132 ;Input port
21         ;
22         ; Write registers
23         ;
24         000102 QPRCSA = 102 ;Clock select A
25         000104 QPRCRA = 104 ;Command register A
26         000106 QPRTRA = 106 ;Transmitter register A
27         000110 QPRACR = 110 ;Auxiliary control register
28         000112 QPRIMR = 112 ;Interrupt mask register
29         000122 QPRCSB = 122 ;Clock select B
30         000126 QPRTRB = 126 ;Transmitter register B
31         000132 QPROPC = 132 ;Output port configuration register
32         000134 QPRSOP = 134 ;Set output port bits register
33         000136 QPRROP = 136 ;Reset output port bits register
34         ;
35         ; Offsets to register addresses by channel and DUART
36         ;
37         000020 QPRCOF = 020 ;Diff between channel A and B registers
38         000040 QPRUOF = 040 ;Diff between DUART 0 and DUART 1 registers
39         ;
40         ; Control flags for the quad serial line unit.
41         ;
42         ; Module status register
43         ;
44         000004 QP$22C = 4 ;2/2 configuration
45         000010 QP$40C = 10 ;4/0 configuration
46         000100 QP$IEN = 100 ;Interrupt enable
47         ;
48         ; Mode register 1
49         ;
50         000002 QP$7BR = 2 ;7 bit characters
51         000003 QP$8BR = 3 ;8 bit characters
52         000004 QP$ODD = 4 ;On ==> Odd parity
53         000020 QP$NPK = 20 ;No parity (off==>Want parity)
54         000040 QP$BEM = 40 ;Block error mode
55         000100 QP$SFL = 100 ;Interrupt when silo full
56         000200 QP$RXR = 200 ;Receiver RTS control
57         ;

```

Quad Serial Line Unit control values

```

58      ; Mode register 2
59      ;
60      000007      QP$1SB  =      7      ; One stop bit
61      000017      QP$2SR  =     17      ; Two stop bits
62      000020      QP$TCS  =     20      ; Transmitter CTS enable
63      000040      QP$TRS  =     40      ; Transmitter RTS enable
64      ;
65      ; Command register
66      ;
67      000001      QP$ERX  =      1      ; Enable receiver
68      000002      QP$DRX  =      2      ; Disable receiver
69      000004      QP$ETX  =      4      ; Enable transmitter
70      000010      QP$DTX  =     10      ; Disable transmitter
71      ;
72      ; Command values for command register
73      ;
74      000020      QP$RPR  =     20      ; Reset MR pointer
75      000040      QP$RRX  =     40      ; Reset receiver
76      000060      QP$RTX  =     60      ; Reset transmitter
77      000100      QP$RES  =    100      ; Reset error status
78      000120      QP$RBC  =    120      ; Reset break change interrupt
79      000140      QP$SBT  =    140      ; Start break transmission
80      000160      QP$EBT  =    160      ; End break transmission
81      ;
82      ; Status register
83      ;
84      000001      QP$RDN  =      1      ; Receiver ready (character received)
85      000002      QP$FFL  =      2      ; FIFO silo full
86      000004      QP$TDN  =      4      ; Transmitter ready (finished sending char)
87      000010      QP$TEM  =     10      ; Transmitter buffer empty
88      000020      QP$ROE  =     20      ; Receiver overrun error
89      000040      QP$RPE  =     40      ; Received parity error
90      000100      QP$RFE  =    100      ; Received framing error
91      000200      QP$BRK  =    200      ; Received break
92      ;
93      ; Auxiliary control register
94      ;
95      000360      QP$AWA  =    360      ; Standard value for TSX-Plus
96      ;
97      ; Interrupt status register
98      ;
99      000001      QP$TAR  =      1      ; Transmitter A finished transmission
100     000002      QP$RAR  =      2      ; Receiver A has a character
101     000004      QP$BCA  =      4      ; Receiver A break status change
102     000010      QP$CNR  =     10      ; Counter/timer ready
103     000020      QP$TBR  =     20      ; Transmitter B finished transmission
104     000040      QP$RBR  =     40      ; Receiver B has a character
105     000100      QP$BCB  =    100      ; Receiver B break status change
106     000200      QP$IPC  =    200      ; Input port status change
107     ;
108     ; Interrupt mask register
109     ;
110     000063      QP$AWI  =     63      ; Standard value for TSX-Plus

```

PROINI -- General initialization for Pro

```

1          .SBTTL  PROINI -- General initialization for Pro
2          ;-----
3          ; Perform general system initialization for PRO.
4          ;
5 000112  010146  PROINI: MOV      R1,-(SP)
6          ;
7          ; Set up information about which device is installed in each option slot
8          ;
9 000114  004737  000270'      CALL      INISLT
10         ;
11        ; Set up address of video interrupt vector and CSR
12        ;
13 000120  012701  000074'      MOV       #VIDTBL,R1      ; Point to table with video ID's
14 000124  012100          1$:  MOV       (R1)+,R0      ; Get next possible video device ID
15 000126  004737  001254'      CALL      GETSLT      ; See if we can find this ID
16 000132  103774          BCS       1$              ; Loop if not
17 000134  010037  000062'      MOV       R0,VIDSLT      ; Save video option slot number
18 000140  072027  000003          ASH       #3,R0          ; Convert slot # to interrupt address
19 000144  062700  000300          ADD       #300,R0
20 000150  010037  000064'      MOV       R0,VIDVEC      ; Save address of video int vector
21 000154  013700  000062'      MOV       VIDSLT,R0      ; Get video slot #
22 000160  072027  000007          ASH       #7,R0          ; CSR addresses are 200 apart per slot
23 000164  062700  174000          ADD       #174000,R0     ; Add address of CSR for slot 0
24 000170  010037  000000G      MOV       R0,VIDCSR     ; Set CSR address for video
25        ;
26        ; Store pointers to the routines that get CSR and Vector addresses
27        ; for PRO devices.
28        ;
29 000174  012737  001126' 000000G  MOV       #GPVVEC,RPRVEC ; Routine to get PRO vector addresses
30 000202  012737  001130' 000000G  MOV       #GPVCSR,RPRCSR ; Routine to get PRO CSR addresses
31        ;
32        ; Determine if a terminal is connected to the printer port.
33        ;
34 000210  113700  177560          MOVVB    @#PP$RCS,R0     ; Get receiver control register
35 000214  153700  177564          BISB    @#PP$TCS,R0     ; Combine flags from transmitter
36 000220  001405          BEQ     4$              ; Br if terminal not connected
37 000222  105237  000000G'      INCB    PPTERM          ; Remember terminal on printer port
38        ;
39        ; A terminal is connected to the printer port.
40        ; Enable BREAK key on diagnostic terminal to enter ODT
41        ;
42 000226  005727  000000G          TST     #PROBRK         ; Is break control wanted?
43 000232  001004          BNE     2$              ; Br if yes
44 000234  042737  000200  173700  4$:  BIC     #SS$BRK,@#PSSREG ; Disable break control
45 000242  000406          BR      5$              ;
46 000244  052737  000200  173700  2$:  BIS     #SS$BRK,@#PSSREG ; Enable break ODT entry
47 000252  012737  000067  173406          MOV     #PP$CMF!PP$RE,@#PP$CMR ; Enable printer port
48        ;
49        ; Perform initialization for quad serial line unit
50        ;
51 000260          5$:
52        .IF      NE,QPASM      ; Do if quad line support wanted
53 000260  004737  004660'      CALL    QPINIT          ; Init quad line unit
54        .ENDC      ; NE,QPASM
55        ;
56        ; Finished
57        ;

```



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

```

1          .SBTTL  INISLT -- Set up information about option slots
2          ;-----
3          ; This routine initializes a table that contains the device ID's of
4          ; devices installed in each of the PRO option slots.
5          ;
6 000270 010146 INISLT: MOV      R1, -(SP)
7 000272 010246          MOV      R2, -(SP)
8          ;
9          ; Map KPAR5 to system configuration table
10         ;
11 000274 005001          CLR      R1          ;Get byte value without sign extension
12 000276 153701 173050  BISB    @#173050, R1      ;Get 32Kb top of memory block #
13 000302 072127 000011  ASH     #9, R1          ;Convert to 64 byte block number
14 000306 162701 000200  SUB     #200, R1        ;Get # of last 8Kb block of memory
15 000312          DISABL          ;** Disable interrupts **
16 000320 013746 000000G  MOV     @#KPAR5, -(SP)    ;Save current par 5 mapping
17 000324 010137 000000G  MOV     R1, @#KPAR5      ;Map par5 to configuration table
18 000330 052737 000000G 000000G  BIS     #MMENBL, @#SR0MMR ;Turn on memory management
19 000336 052737 000000G 000000G  BIS     #EMMAP, @#SR3MMR ;Enable 22 bit addressing
20         ;
21         ; Get number of option slots
22         ;
23 000344 012702 017764G  MOV     #VPAR5+17764, R2 ;Point to cell with # option slots
24 000350 011200          MOV     (R2), R0        ;Get # of option slots
25         ;
26         ; Set up table in TSX that has device ID number of each option slot
27         ;
28 000352 012701 000000G  MOV     #PROSLT, R1      ;Point to table that will store option ID's
29 000356 005742          1$:  TST     -(R2)          ;Skip word with option status value
30 000360 014221          MOV     -(R2), (R1)+    ;Store option ID code
31 000362 077003          SOB     R0, 1$         ;Loop to get all option slot device ID's
32 000364 012711 177777  MOV     #-1, (R1)       ;Store -1 at end of table
33         ;
34         ; Restore KPAR5 mapping
35         ;
36 000370 012637 000000G  MOV     (SP)+, @#KPAR5  ;Restore par 5 mapping
37 000374 042737 000000G 000000G  BIC     #MMENBL, @#SR0MMR ;Turn off memory mapping
38 000402          ENABL          ;** Enable interrupts **
39         ;
40         ; Finished
41         ;
42 000410 012602          MOV     (SP)+, R2
43 000412 012601          MOV     (SP)+, R1
44 000414 000207          RETURN

```

PRONOP -- Disable PRO interrupts

```

1          .SBTTL  PRONOP -- Disable PRO interrupts
2          ;-----
3          ; PRONOP is called during system startup to direct PRO interrupts to
4          ; an RTI instruction so they will not enter RT-11.
5          ;
6 000416 010146 PRONOP: MOV      R1,-(SP)
7 000420 012700 000466'   MOV      #DORTI,R0          ;Get address of RTI instruction
8 000424 010037 000060   MOV      R0,@#60          ;Catch keyboard input interrupt
9 000430 010037 000200   MOV      R0,@#200        ;380 keyboard input interrupt
10 000434 010037 000064   MOV      R0,@#64         ;End-of-transfer video interrupt
11 000440 010037 000204   MOV      R0,@#204        ;Keyboard output interrupt
12 000444 010037 000230   MOV      R0,@#230        ;380 clock interrupt
13 000450 013701 000064'   MOV      VIDVEC,R1       ;Point to video interrupt vectors
14 000454 010011         MOV      R0,(R1)         ;End-of-frame video interrupt
15 000456 010061 000004   MOV      R0,4(R1)        ;end-of-transfer video interrupt
16          ;
17          ; Finished
18          ;
19 000462 012601         MOV      (SP)+,R1
20 000464 000207         RETURN
21          ;
22          ; RTI instruction used to render interrupts harmless
23          ;
24 000466 000002 DORTI:  RTI              ;Return from interrupt immediately

```

PROHAN -- Initialize the PI handler

```

1          .SBTTL  PROHAN -- Initialize the PI handler
2          ;-----
3          ; PROHAN is called to initialize the PI handler.
4          ;
5 000470 010146 PROHAN: MOV     R1,-(SP)
6 000472 010246          MOV     R2,-(SP)
7 000474 010346          MOV     R3,-(SP)
8 000476 013746 0000000  MOV     @#KPAR5,-(SP) ; Save current kernel PAR5 mapping
9 000502 013746 0000000  MOV     @#KPAR6,-(SP) ; Save current kernel PAR6 mapping
10         ;
11         ; Set up information about where the PI handler is in memory.
12         ;
13 000506 012703 0000000  MOV     #PISRT,R3      ; Point to shared run-time descriptor for PI
14 000512 016337 0000000 000052'  MOV     RT#BAS(R3),PIBASE; This is base 64-byte block # of run-time
15 000520 013737 000052' 0000000  MOV     PIBASE,@#KPAR5 ; Map to base of PI handler through PAR 5
16         ;
17         ; Transfer information between TSXPRO and the PI handler.
18         ;
19 000526 012701 0010000  MOV     #VPAR5+1000,R1 ; Point to HOKVEC vector in PI handler
20 000532 012702 0000000'  MOV     #PROVEC,R2     ; Point to TSXPRO vector
21 000536 011100          5$: MOV     (R1),R0        ; Get address of a cell in PI handler
22 000540 001404          BEQ     6$             ; Br if PI wants an address
23 000542 062700 0000000  ADD     #VPAR5,R0      ; Bias the address to be in PAR 5 region
24 000546 010012          MOV     R0,(R2)        ; Store address pointer in TSXPRO cell
25 000550 000402          BR     7$             ;
26 000552 011211          6$: MOV     (R2),(R1)     ; Pass an address to PI handler
27 000554 001402          BEQ     8$             ; Br if just hit end of lists
28 000556 022122          7$: CMP     (R1)+,(R2)+ ; Increment both list pointers
29 000560 000766          BR     5$             ; Continue processing lists
30         ;
31         ; Move the PI initialization overlay into the TSINIT work buffer.
32         ;
33 000562 013701 0000000'  8$: MOV     PIINAD,R1   ; Get address of init overlay
34 000566 162701 0000000  SUB     #VPAR5,R1      ; Subtract bias we added
35 000572 072127 177772  ASH     #-6,R1         ; Convert to 64-byte block #
36 000576 063701 000052'  ADD     PIBASE,R1      ; Get mapping base for init overlay
37 000602 010137 0000000  MOV     R1,@#KPAR6     ; Map to overlay through PAR 6
38 000606 013700 000002'  MOV     PIINWD,R0      ; Get # words in init overlay
39 000612 162700 0000000  SUB     #VPAR5,R0      ; Subtract bias we added
40 000616 012701 0000000  MOV     #VPAR6,R1      ; Get virtual address of init overlay
41 000622 013702 0000000  MOV     PROBUF,R2      ; Get address of TSINIT work buffer
42 000626 012122          4$: MOV     (R1)+,(R2)+ ; Move init code to work buffer
43 000630 077002          SOB     R0,4$         ;
44         ;
45         ; Now execute the PI initialization code that we have moved to
46         ; the TSINIT work buffer.
47         ; Note: PAR5 is now mapped to the resident portion of the PI handler.
48         ; On entry,
49         ; R3 = Virtual address of base of PI handler.
50         ; R1 = 64-byte block number of base of PI handler.
51         ;
52 000632 013701 000052'  MOV     PIBASE,R1      ; Get 64-byte block # of PI base
53 000636 012703 0000000  MOV     #VPAR5,R3      ; Get virtual address of PI base
54 000642 004777 0000000  CALL    @PROBUF        ; Call PI initialization code
55         ;
56         ; Now get some actual addresses from PI handler
57         ;

```

```

58 000646 017737 177144 000016'      MOV      @#VDCSR,#VDCSR ;Get video CSR address
59                                     ;
60                                     ; Connect to interrupt vectors
61                                     ;
62 000654 012700 000340      MOV      #340,R0        ;Get priority 7 code
63 000660 012701 000060      MOV      #60,R1        ;Get address of start of vector area
64 000664 012721 002114'     MOV      #PIKIIR,(R1)+ ;Set keyboard input interrupt routine
65 000670 010021              MOV      R0,(R1)+      ;Set interrupt service priority to 7
66 000672 012721 001560'     MOV      #PIVTIR,(R1)+ ;Set end of transfer video interrupt
67 000676 010021              MOV      R0,(R1)+      ;Set priority = 7
68 000700 012737 002214' 000204  MOV      #PIKOIR,@#204 ;Set keyboard output interrupt routine
69 000706 010037 000206      MOV      R0,@#206     ;Set priority
70                                     ;
71                                     ; Connect to 380 interrupt vectors
72                                     ;
73 000712 013737 000060 000200  MOV      @#60,@#200    ;380 keyboard input interrupt
74 000720 013737 000062 000202  MOV      @#62,@#202
75 000726 013701 000064'     MOV      VIDVEC,R1    ;Point to video interrupt vector A
76 000732 012721 002052'     MOV      #PIVFIR,(R1)+ ;Set end-of-frame video interrupt
77 000736 010021              MOV      R0,(R1)+      ;Set priority
78 000740 012721 001560'     MOV      #PIVTIR,(R1)+ ;Set end-of-transfer video interrupt
79 000744 010021              MOV      R0,(R1)+      ;Set priority
80 000746 013737 000100 000230  MOV      @#100,@#230  ;380 clock interrupt
81 000754 013737 000102 000232  MOV      @#102,@#232
82                                     ;
83                                     ; Connect to interrupt for quad serial line unit
84                                     ;
85                                     . IF      NE,QPASM      ;Assemble if quad line unit support wanted
86 000762 013703 000070'     MOV      QPVEC,R3     ;Get address of interrupt vector
87 000766 001410              BEQ      10$          ;Br if quad serial unit not installed
88 000770 012723 005354'     MOV      #QPCINT,(R3)+ ;Set PC for interrupt
89 000774 010013              MOV      R0,(R3)      ;Set PS for interrupt
90 000776 013703 000066'     MOV      QPCSR,R3     ;Get address of CSR register
91 001002 152763 000100 000004  BISB    #QP$IEN,QPRMSR(R3) ;Enable interrupts
92                                     . ENDC
93                                     ;
94                                     ; Finished
95                                     ;
96 001010 012637 0000006     10$:   MOV      (SP)+,@#KPAR6 ;Restore kernel mapping
97 001014 012637 0000006     MOV      (SP)+,@#KPAR5
98 001020 012603              MOV      (SP)+,R3
99 001022 012602              MOV      (SP)+,R2
100 001024 012601              MOV      (SP)+,R1
101 001026 000207              RETURN
    
```

PROLIN -- See if line is a special PRO terminal

```

1          .SBTTL  PROLIN -- See if line is a special PRO terminal
2          ;-----
3          ; PROLIN is called once for each DL11 type line defined in the system.
4          ; It determines if the line is a printer port or communications port
5          ; line and if so performs the initialization for the line.
6          ;
7          ; Inputs:
8          ;   R1 = Line index number.
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> This is a printer port or comm port line.
12         ;   C-flag set      ==> This is not a printer port or comm port line.
13         ;
14 001030  PROLIN:
15         ;
16         ; See if this is a line connected to the quad serial line unit
17         ;
18         .IF      NE, QPASM      ; Assemble for quad line support
19 001030  005761  0000000  TST      LMXNUM(R1)      ; Is this line on a mux?
20 001034  001403          BEQ      3$          ; Br if not
21 001036  004737  005136'  CALL     QPLINE         ; Initialize line on quad unit
22 001042  000427          BR       9$
23         .ENDC      ; NE, QPASM
24         ;
25         ; See if this is the Professional video console
26         ;
27 001044  026127  0000000  000060  3$:     CMP      INVEC(R1), #60  ; Is this the console line?
28 001052  001003          BNE      2$          ; Br if not console line
29 001054  004737  001316'  CALL     PIINIT         ; Initialize PI line
30 001060  000420          BR       9$
31         ;
32         ; Get address of receiver status register for the line
33         ;
34 001062  016100  0000000  2$:     MOV      RSR(R1), R0      ; Get address of receiver status register
35         ;
36         ; See if this is the printer port
37         ;
38 001066  020027  173400          CMP      RO, #PP$DBR      ; Printer port?
39 001072  001003          BNE      1$          ; Br if not
40 001074  004737  002324'  CALL     PPINIT         ; Initialize the printer port
41 001100  000410          BR       9$
42         ;
43         ; See if this is the communications port
44         ;
45 001102  020027  173300  1$:     CMP      RO, #CP$DBR      ; Communications port?
46 001106  001003          BNE      8$          ; Br if not
47 001110  004737  003146'  CALL     CPINIT         ; Initialize the comm port
48 001114  000402          BR       9$
49         ;
50         ; This line is not the printer port or the comm port
51         ;
52 001116  000261  8$:     SEC                          ; Signal failure on return
53 001120  000401          BR       10$
54         ;
55         ; This line was the printer port or the comm port
56         ;
57 001122  000241  9$:     CLC                          ; Signal success on return

```

```
58 ;  
59 ; Finished  
60 ;  
61 001124 00020/ 10$: RETURN
```

```

1          ;
2          ; Branch vector to entry points of PRO CSR/Vector/Slot routines
3          ;
4 001126 000402  GPVVEC: BR      GPRVEC      ;Get vector address
5 001130 000420  GPVCSR: BR      GPRCSR      ;Get CSR address
6 001132 000436  GPVSLT: BR     GPRSLT      ;Get Slot value
7
8          .SBTTL  GPRVEC -- Get address of vector for PRO device
9          ;-----
10         ; This routine is called to determine the address of the interrupt vector
11         ; for a PRO device based on the device ID.
12         ;
13         ; Inputs:
14         ; Device ID is pushed on stack before calling GPRVEC.
15         ;
16         ; Outputs:
17         ; C-flag cleared ==> Found device in tables; C-flag set ==> No such device.
18         ; Top of stack contains vector for device.
19         ;
20 001134 010046  GPRVEC: MOV     RO,-(SP)
21 001136 016600 000004  MOV     4(SP),RO      ;Get the device ID code
22 001142 004737 001254'  CALL   GETSLT        ;Determine which option slot has the dev
23 001146 103407          BCS     9$            ;Br if device not recognized
24 001150 072027 000003  ASH    #3,RO         ;Vectors are 8 bytes apart per slot
25 001154 062700 000300  ADD    #300,RO       ;Vectors for slot 0 start at 300
26 001160 000241          CLC                    ;Signal success on return
27 001162 010066 000004  MOV    RO,4(SP)      ;Place vector address on stack
28 001166 012600 9$:      MOV    (SP)+,RO
29 001170 000207          RETURN
  
```



GPRCSR -- Get address of CSR for PRO device

```

1          .SBTTL  GPRCSR -- Get address of CSR for PRO device
2          ;-----
3          ; This routine is called to determine the address of the CSR
4          ; for a PRO device based on the device ID.
5          ;
6          ; Inputs:
7          ; Device ID is pushed on stack before calling GPRCSR.
8          ;
9          ; Outputs:
10         ; C-flag cleared ==> Found device in tables; C-flag set ==> No such device.
11         ; Top of stack contains address of CSR for device.
12         ;
13 001172 010046 GPRCSR: MOV     RO,-(SP)
14 001174 016600 000004 MOV     4(SP),RO      ;Get the device ID code
15 001200 004737 001254' CALL    GETSLT       ;Determine which option slot has the dev
16 001204 103407 BCS     9$           ;Br if device not recognized
17 001206 072027 000007 ASH     #7.,RO       ;CSR address are 200 apart per slot
18 001212 062700 174000 ADD     #174000,RO    ;CSR for slot 0 starts at 174000
19 001216 000241 CLC                    ;Signal success on return
20 001220 010066 000004 MOV     RO,4(SP)      ;Place CSR address on stack
21 001224 012600 9$:   MOV     (SP)+,RO
22 001226 000207 RETURN

```

GPRSLT -- Get Slot number for PRO device

```

1          .SBTTL  GPRSLT -- Get Slot number for PRO device
2          ;-----
3          ; This routine is called to determine the slot number
4          ; for a PRO device based on the device ID.
5          ;
6          ; Inputs:
7          ; Device ID is pushed on stack before calling GPRSLT.
8          ;
9          ; Outputs:
10         ; C-flag cleared ==> Found device in tables; C-flag set ==> No such device.
11         ; Top of stack contains slot number for device.
12         ;
13 001230 010046 GPRSLT: MOV     RO,-(SP)
14 001232 016600      MOV     4(SP),RO      ;Get the device ID code
15 001236 004737 001254' CALL   GETSLT      ;Determine which option slot has the dev
16 001242 103402      BCS     9$          ;Br if device not recognized
17 001244 010066 000004      MOV     RO,4(SP)      ;Put slot number back onto stack
18 001250 012600 9$:      MOV     (SP)+,RO
19 001252 000207      RETURN

```

GETSLT --- Determine which option slot has device controller

```

1          .SBTTL  GETSLT -- Determine which option slot has device controller
2          ;-----
3          ; This routine is called to determine which PRO option slot contains the
4          ; controller for a specified device.
5          ;
6          ; Inputs:
7          ;   RO = Device ID code.
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Found the device; C-flag set ==> No such device
11         ;   RO = Slot number.
12         ;
13 001254 010146 GETSLT: MOV     R1, -(SP)
14 001256 010001      MOV     RO, R1          ; Get device ID code
15 001260 012700 0000000 MOV     #PROSLT, RO      ; Point to table with device ID codes
16 001264 021027 177777 1#:  CMP     (RO), #-1      ; Reached end of table entries?
17 001270 001407      BEQ     2#          ; Br if yes
18 001272 020120      CMP     R1, (RO)+      ; Search for device ID in table
19 001274 001373      BNE     1#          ; Loop if not found
20 001276 162700 0000020 SUB     #PROSLT+2, RO    ; Compute table offset of entry
21 001302 006200      ASR     RO          ; Convert to word index
22 001304 000241      CLC          ; Signal success on return
23 001306 000401      BR     9#
24 001310 000261 2#:  SEC          ; Signal failure on return
25 001312 012601 9#:  MOV     (SP)+, R1
26 001314 000207      RETURN

```

\*\*\* Console Control Routines \*\*\*

```

1          .SBTTL  *** Console Control Routines ***
2          .SBTTL  PIINIT --- Initialize the console
3          ;-----
4          ; PIINIT is called during system initialization to initialize the
5          ; console.
6          ;
7          ; Inputs:
8          ; R1 = Line index number.
9          ;
10         001316 PIINIT:
11         ;
12         ; Remember index number of line connected to console
13         ;
14         001316 010137 000044'      MOV     R1,PILINE      ; Save line index # of console line
15         001322 012761 0000000 0000000  MOV     #CDX#PI,LCDTYP(R1); Set type of communications controller
16         ;
17         ; Set addresses of communication port device-dependent routines
18         ;
19         001330 012737 001354' 0000000  MOV     #PISTRT,CDSTRT+CDX#PI ; Set address of start routine
20         ;
21         ; Default terminal type to VT100
22         ;
23         001336 005761 0000000      TST     ITRMTP(R1)      ; Is a terminal type specified?
24         001342 001003                      BNE     1$              ; Br if yes
25         001344 012761 0000000 0000000  MOV     #VT100,ITRMTP(R1); Set default terminal type to VT100
26         ;
27         ; Finished
28         ;
29         001352 000207      1$:      RETURN

```

PISTRT -- Start output to video screen

```

1          .SBTTL  PISTRT -- Start output to video screen
2          ;-----
3          ; PISTRT is called after a character is placed in the output ring
4          ; buffer to initiate transmission to the video screen
5          ;
6          ; Inputs:
7          ; R1 = Line index number of line being started.
8          ;
9 001354   PISTRT:
10         ;
11         ; See if we have already started transmitter during this clock period
12         ;
13 001354   005237   000056'   INC      PISCNT      ;Have we already started during clk interval?
14 001360   003005           BGT      9$           ;Br if yes
15         ;
16         ; Do not allow output to the console to consume the entire system.
17         ;
18 001362   004737   002000'   CALL     PIQUIT     ;Is it time to suspend output?
19 001366   103402           BCS      9$           ;Br if yes
20         ;
21         ; Start output
22         ;
23 001370   004737   001420'   CALL     PIGO      ;Start transmitter
24         ;
25         ; Finished
26         ;
27 001374   000207   9$:      RETURN
28         ;
29         .SBTTL  PIDRIV -- Clock driven PI transmitter routine
30         ;-----
31         ; PIDRIV is called periodically from the clock interrupt routine
32         ; to force output to the PI console.
33         ;
34 001376   PIDRIV:
35         ;
36         ; Reset output character suspend count
37         ;
38 001376   012737   000017   000072'   MOV     #MAXPOC,PIMDC ;Reset output character count
39         ;
40         ; Start output to PI
41         ;
42 001404   004737   001420'   CALL     PIGO      ;Start the transmitter
43         ;
44         ; Reset flag saying if transmitter has been started during clock interval
45         ;
46 001410   012737   177777   000056'   MOV     #-1,PISCNT   ;Reset counter
47         ;
48         ; Finished
49         ;
50 001416   000207   RETURN

```

PIC0 -- Start PI transmitter

```

1          .SBTTL  PIC0  -- Start PI transmitter
2          ;-----
3          ; PIC0 is called to enable a transfer-done interrupt which will start
4          ; transmission to the PI console.
5          ;
6 001420  013746  000000G  PIC0:  MOV    @KPAR5,-(SP)  ;Save current PAR5 mapping
7 001424  013737  000052' 000000G      MOV    PIBASE,@KPAR5 ;Set up PAR5 to map to PI
8          ;
9          ; See if output to the PI handler has been suspended
10         ;
11 001432  005777  176370      TST    @PIGOFL      ;Has PI output been suspended?
12 001436  001013      BNE    1$          ;Br if not
13         ;
14         ; Output to the PI handler has been suspended.
15         ; Consume all pending output characters and then return.
16         ;
17 001440  010446      MOV    R4,-(SP)
18 001442  013704  000044'      MOV    PILINE,R4      ;Get line index number
19 001446      DISABL          ;;;** Disable interrupts **
20 001454  004774  000000G  4$:  CALL   @LOUTIR(R4)    ;;;Get next char for PI
21 001460  103375      BCC    4$          ;;;Loop till all characters consumed
22 001462  012604      MOV    (SP)+,R4      ;;;
23 001464  000427      BR     9$          ;;;
24         ;
25         ; Output to PI handler is not suspended
26         ;
27 001466      1$:  DISABL          ;;;** Disable interrupts **
28         ;
29         ; See if we have already triggered a PI interrupt
30         ;
31 001474  005737  000054'      TST    PIOIFL      ;;;Have we already triggered an output int?
32 001500  001021      BNE    9$          ;;;Br if output interrupt active now
33 001502  005237  000054'      INC    PIOIFL      ;;;Set flag saying output int active
34 001506  010677  176312      MOV    SP,@VDFLAG    ;;;Tell VDCURS that buffer is not empty
35 001512  032777  040000 176276  BIT    #VP$DOI,@$VDCSR ;;;Is end-of-transfer interrupt enabled?
36 001520  001011      BNE    9$          ;;;Br if yes
37 001522  005777  176274      TST    @PRID0      ;;;Has video said to leave it alone?
38 001526  001003      BNE    2$          ;;;Br if yes
39 001530  052777  040000 176260  BIS    #VP$DOI,@$VDCSR ;;;Generate end-of-transfer interrupt
40 001536  012777  100000 176254  2$:  MOV    #100000,@GREENAB ;;;Tell video to turn on interrupts when ready
41         ;
42         ; Finished
43         ;
44 001544  012637  000000G  9$:  MOV    (SP)+,@KPAR5 ;;;Restore PAR5 mapping
45 001550      ENABL          ;;;** Enable interrupts **
46 001556  000207      RETURN

```

```

1          .SBTTL  PIVTIR -- Video end of transfer interrupt
2          ;-----
3          ; PIVTIR is jumped to from the video end-of-transfer interrupt.
4          ;
5 001560   PIVTIR:
6          ;
7          ; Do .INTEN to drop priority to 4
8          ;
9 001560   004537   000000G       JSR      R5,INTEN      ;Do standard interrupt entry
10 001564   000140       .WORD    140          ;Priority = 4
11 001566   010046       MOV      RO,-(SP)      ;Save RO
12          ;
13          ; Set flag saying output interrupt is in progress
14          ;
15 001570   005237   000054'       INC      PIOIFL       ;Doing output interrupt processing
16          ;
17          ; Disable transfer done interrupt
18          ;
19 001574   013737   000052' 000000G   MOV      PIBASE,@#KPAR5 ;Map kernel PAR 5 to the PI handler
20 001602   012777   100000   176210   MOV      #100000,@REENAB ;Set flag to reenale transfer-done int
21 001610   042777   040000   176200   BIC      #VP#DOI,@$VDCSR ;Disable transfer-done interrupt
22          ;
23          ; Limit the number of consecutive characters sent to the console
24          ; to prevent the PI output processing (which is compute bound) from
25          ; consuming 100% of the machine.
26          ;
27 001616   013704   000044'       MOV      PILINE,R4     ;Get line index number of console line
28 001622   004737   002000'       CALL     PIQUIT        ;Time to suspend output?
29 001626   103431       BCS     2$             ;Br if yes
30          ;
31          ; Get the next character to transmit.
32          ;
33 001630   004774   000000G       CALL     @LQUTIR(R4)   ;Get next char for line
34 001634   103423       BCS     1$             ;Br if no character available
35 001636   010005       MOV     RO,R5          ;Get char to transmit to R5
36 001640   012600       MOV     (SP)+,RO      ;Restore RO before we fork
37          ;
38          ; There is another character to transmit. Now fork
39          ;
40 001642   004537   000000G       JSR     R5,FORK        ;Fork to get to priority 0
41 001646   000000G       .WORD   FP#PIO        ;Fork priority
42 001650   010046       MOV     RO,-(SP)      ;Save RO
43          ;
44          ; Map PAR5 to the PI handler
45          ;
46 001652   013737   000052' 000000G   MOV     PIBASE,@#KPAR5 ;Map kernel PAR 5 to the PI handler
47          ;
48          ; Enter PI handler to transmit the character
49          ;
50 001660   010500       MOV     R5,RO         ;Get char to transmit
51 001662   052764   000000G 000000G   BIS     #$XCHAR,LSW3(R4);Set transmitter busy flag
52 001670   042764   000000G 000000G   BIC     #$OITIM,LSW5(R4);Start output interrupt timer
53 001676   004777   176102       CALL    @PIXOCH       ;Call routine to transmit the character
54 001702   000412       BR     9$             ;Finished
55          ;
56          ; No character is available for transmission
57          ;

```

```
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 @GREENAB ;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
```



PINDCH -- Get next output character for console

```

1          .SBTTL  PINDCH -- Get next output character for console
2          ;-----
3          ; PINDCH is called from within the VIDE0 handler to try to get another
4          ; character to send to the console terminal.
5          ;
6          ; Outputs:
7          ; C-flag cleared ==> a character is available.
8          ; C-flag set    ==> No more characters available.
9          ; RO = Character to send if C-flag is cleared.
10         ;
11 001740 010446 PINDCH: MOV     R4, -(SP)
12 001742 013704 000044'      MOV     PILINE, R4      ;Get line index number of console
13         ;
14         ; Limit the number of consecutive output characters to prevent the
15         ; P1 output processing (which is compute bound) from completely
16         ; dominating the system.
17         ;
18 001746 004737 002000'      CALL    PIQUIT      ;Time to suspend output?
19 001752 103410              BCS     9$           ;Br if yes
20         ;
21         ; Call line-dependent routine to try to get another output character
22         ;
23 001754 1#:  DISABL          ;;;** Disable interrupts for LOUTIR **
24 001762 004774 000000G      CALL    @LOUTIR(R4) ;;;Get next character for line
25 001766              ENABL          ;;;** Enable interrupts **
26         ;
27         ; Finished
28         ;
29 001774 012604 9$:  MOV     (SP)+, R4
30 001776 000207          RETURN

```

PIQUIT -- Check output character limit

```

1          .SBTTL  PIQUIT -- Check output character limit
2          ;-----
3          ; This routine causes output to the console terminal to be suspended
4          ; after a certain number of characters have been transmitted.
5          ; The output is resumed after a timed interval.
6          ; This is done to prevent the console terminal output processing (which
7          ; is compute bound) from consuming 100% of the machine.
8          ;
9          ; Outputs:
10         ; C-flag set      ==> Suspend output.
11         ; C-flag cleared ==> Continue output.
12         ;
13 002000  PIQUIT:
14         ;
15         ; See if there is any other pending system activity
16         ;
17 002000  105737  0000000  TSTB   DOSCHD      ; Scheduler cycle needed?
18 002004  001013  BNE     2$          ; Br if yes
19 002006  005737  0000000  TST   FRKQOE      ; Fork requests pending?
20 002012  001010  BNE     2$          ; Br if yes
21 002014  105737  0000000  TSTB  CORUSR      ; Is there another executing job?
22 002020  001412  BEQ     1$          ; Br if not
23 002022  013700  000044'  MOV   PILINE,RO  ; Get # of job using console terminal
24 002026  005760  0000000  TST   LXCL(RO)   ; Is that job cross-connected to CL line?
25 002032  002405  BLT     1$          ; Br if not
26         ;
27         ; There is some pending system activity.
28         ; See if it is time to suspend PI output.
29         ;
30 002034  005337  000072'  2$:   DEC   PIMOC      ; Time to suspend output?
31 002040  003002  BGT     1$          ; Br if not
32 002042  000261  SEC                      ; Signal to suspend
33 002044  000401  BR      9$          ;
34 002046  000241  1$:   CLC                      ; Signal to continue output
35         ;
36         ; Finished
37         ;
38 002050  000207  9$:   RETURN

```

PIVFIR -- Video end of frame interrupt

```

1          .SBTTL PIVFIR -- Video end of frame interrupt
2          ;-----
3          ; PIVFIR is jumped to from the interrupt vector whenever a video
4          ; end-of-frame interrupt occurs.
5          ;
6 002052 PIVFIR:
7          ;
8          ; Do .INTEN to drop priority to 4
9          ;
10 002052 004537 0000000 JSR     R5,INTEN      ;Do standard interrupt entry
11 002056 000140          .WORD   140          ;Priority = 4
12 002060 010046          MOV     R0,-(SP)
13 002062 010146          MOV     R1,-(SP)
14 002064 010246          MOV     R2,-(SP)
15 002066 010346          MOV     R3,-(SP)
16          ;
17          ; Enter overlay to do the actual processing
18          ;
19 002070 013737 000052' 0000000 MOV     PIBASE,@#KPAR5 ;Map kernel PAR 5 to the PI handler
20 002076 004777 175704          CALL   @PIXEOF        ;Do end of frame processing
21          ;
22          ; Finished
23          ;
24 002102 012603          MOV     (SP)+,R3
25 002104 012602          MOV     (SP)+,R2
26 002106 012601          MOV     (SP)+,R1
27 002110 012600          MOV     (SP)+,R0
28 002112 000207          RETURN

```

```

1          .SBTTL  PIKIIR -- Keyboard input interrupt
2          ;-----
3          ; PIKIIR is vectored to when a keyboard input interrupt occurs.
4          ;
5 002114   PIKIIR:
6          ;
7          ; Do .INTEN to drop priority to 4
8          ;
9 002114   004537   000000G       JSR      R5,INTEN       ;Do standard interrupt entry
10 002120   000140       .WORD    140           ;Priority = 4
11 002122   010046       MOV      R0,-(SP)
12 002124   010146       MOV      R1,-(SP)
13 002126   010246       MOV      R2,-(SP)
14 002130   010346       MOV      R3,-(SP)
15          ;
16          ; Enter system overlay to do the actual processing
17          ;
18 002132   013737   000052' 000000G   MOV      PIBASE,@#KPAR5 ;Map kernel PAR 5 to the PI handler
19 002140   004777   175644           CALL     @PIXICH       ;Process the input character
20          ;
21          ; Finished
22          ;
23 002144   012603       MOV      (SP)+,R3
24 002146   012602       MOV      (SP)+,R2
25 002150   012601       MOV      (SP)+,R1
26 002152   012600       MOV      (SP)+,R0
27 002154   000207       RETURN
28
29          .SBTTL  KBDCHR -- Process character received from keyboard
30          ;-----
31          ; KBDCHR is jumped to from the PI handler when a character has been
32          ; received and converted to ASCII.
33          ;
34          ; Inputs:
35          ;   R5 = Received ascii character.
36          ;
37 002156   KBDCHR:
38          ;
39          ; Save the PAR5 and PAR6 mapping for the handler, and then call
40          ; the TTINPT routine to process the received character.
41          ;
42 002156   013746   000000G       MOV      @#KPAR5,-(SP)   ;Save PAR5 and PAR6 mapping for PI
43 002162   013746   000000G       MOV      @#KPAR6,-(SP)
44 002166   010446       MOV      R4,-(SP)
45 002170   013704   000044'       MOV      PILINE,R4     ;Get line index # for console line
46 002174   004777   000000G       CALL     @TTINPT      ;Process received character
47          ;
48          ; Now restore PAR5 and PAR6 mapping and return to PI
49          ;
50 002200   012604       MOV      (SP)+,R4
51 002202   012637   000000G       MOV      (SP)+,@#KPAR6 ;Restore PAR5 and PAR6
52 002206   012637   000000G       MOV      (SP)+,@#KPAR5
53 002212   000207       RETURN               ;Return to PI

```

PIKOIR -- Output interrupt for keyboard

```

1                                     .SBTTL  PIKOIR -- Output interrupt for keyboard
2                                     ;-----
3                                     ; PIKOIR is vectored to when we receive an interrupt on completion of
4                                     ; transmitting a character to the keyboard.
5                                     ;
6 002214                               PIKOIR:
7                                     ;
8                                     ; Do .INTEN to drop priority to 4
9                                     ;
10 002214 004537 0000000             JSR      R5,INTEN           ;Do standard interrupt entry
11 002220 000140                     .WORD    140             ;Priority = 4
12 002222 010046                     MOV     R0,-(SP)
13 002224 010146                     MOV     R1,-(SP)
14 002226 010246                     MOV     R2,-(SP)
15 002230 010346                     MOV     R3,-(SP)
16                                     ;
17                                     ; Call overlay routine to do the processing
18                                     ;
19 002232 013737 000052' 0000000     MOV     PIBASE,@#KPAR5 ;Map kernel PAR 5 to the PI handler
20 002240 004777 175546             CALL    @PIXIDI       ;Call overlay processing routine
21                                     ;
22                                     ; Finished
23                                     ;
24 002244 012603                     MOV     (SP)+,R3
25 002246 012602                     MOV     (SP)+,R2
26 002250 012601                     MOV     (SP)+,R1
27 002252 012600                     MOV     (SP)+,R0
28 002254 000207                     RETURN

```

PIHAN -- Simulated PI handler

```

1          .SBTTL  PIHAN  -- Simulated PI handler
2          ;-----
3          ; The following routine is a simulated handler for the PI device.
4          ; It is used to allow SETUP to perform .SPFUN operations on
5          ; the PI device.
6          ;
7          ; Simulated handler header
8          ;
9 002256 000000  PIHAN:  .WORD  0          ;Device vector
10 002260 000014          .WORD  PIINT-    ;Offset to interrupt service entry point
11 002262 000000          .WORD  0          ;Interrupt priority
12 002264 000000  PILQE:  .WORD  0          ;Last queue element
13 002266 000000  PICQE:  .WORD  0          ;Current queue element
14 002270 000402          BR      PIENTR    ;Enter here from system I/O queueing
15          ;
16          ; Simulated interrupt point and abort entry point
17          ;
18 002272 000207          RETURN          ;Abort entry point
19 002274 000002  PIINT:  RTI          ;Fake interrupt entry point
20          ;
21          ; Process an I/O queue request directed to the PI device
22          ;
23 002276          PIENTR:
24          ;
25          ; Map PAR 5 to the real PI handler
26          ;
27 002276 013737 000052' 000000G      MOV      PIBASE,@#KPAR5 ;Map PAR 5 to PI handler
28          ;
29          ; Call real PI handler to process the .SPFUN
30          ;
31 002304 013700 002266'      MOV      PICQE,R0      ;Pass address of I/O queue element in R0
32 002310 004777 175500      CALL     @PIXIOQ      ;Enter handler to process the .SPFUN
33          ;
34          ; Finished processing the I/O queue request
35          ;
36 002314 012704 002266'      MOV      #PICQE,R4      ;Set up R4 for IOFIN
37 002320 000137 000000G      JMP      IOFIN        ;I/O operation is completed

```

\*\*\* Printer Port Control Routines \*\*\*

```

1          .SBTTL  *** Printer Port Control Routines ***
2          .SBTTL  PPINIT -- Initialize the printer port
3          ;-----
4          ; PPINIT is called during system initialization to initialize the
5          ; printer port.
6          ;
7          ; Inputs:
8          ; R1 = Line index number.
9          ;
10         002324 010246 PPINIT: MOV     R2,-(SP)
11         ;
12         ; Remember the line index number of the line connected to the printer port
13         ;
14         002326 010137 000046'      MOV     R1,PPLINE      ;Remember which line connected to printer port
15         002332 012761 000000G 000000G  MOV     #CDX$PP,LCDTYP(R1) ;Set type of communications controller
16         ;
17         ; Set addresses of printer port routines in vector of device-dependent
18         ; routines.
19         ;
20         002340 012737 002620' 000000C  MOV     #PPSTRT,CDSTRT+CDX$PP ;Set address of start routine
21         002346 012737 002772' 000000C  MOV     #PPGDSS,CDGDSS+CDX$PP ;Address of rtn to get data set status
22         002354 012737 002776' 000000C  MOV     #PPSBRK,CDSBRK+CDX$PP ;Address of rtn to control break
23         002362 012737 003024' 000000C  MOV     #PPSSPD,CDSSPD+CDX$PP ;Address of rtn to set speed
24         ;
25         ; Connect the printer port interrupts to the service routine
26         ;
27         002370 012737 002654' 000220   MOV     #PPRINT,@#PP$RCV;Connect receiver interrupt
28         002376 012737 000340 000222   MOV     #340,@#PP$RCV+2 ;Set prio = 7 on interrupt entry
29         002404 012737 002516' 000224   MOV     #PPTINT,@#PP$TRV;Connect transmitter interrupt
30         002412 012737 000340 000226   MOV     #340,@#PP$TRV+2 ;Set prio 7
31         ;
32         ; Set up mode register 1
33         ;
34         002420 005737 173406 3#:      TST     @#PP$CMR      ;Access command reg to reset mode reg ptr
35         002424 112737 000102 173404  MOVB   #PP$M1F,@#PP$MDR;Initialize values in mode register 1
36         ;
37         ; Set speed in mode register 2
38         ;
39         002432 116100 000001G      MOVB   LMXPRM+1(R1),RO ;Get line parameter values
40         002436 001007      BNE    1$             ;Br if parameter values were specified
41         002440 012700 000000G      MOV     #S9600,RO      ;Default to 9600 baud
42         002444 105737 000000G      TSTB   PPTERM        ;Is a terminal connected to printer port?
43         002450 001002      BNE    1$             ;Br if yes
44         002452 012700 000000G      MOV     #S4800,RO     ;Default to 4800 baud
45         002456 004737 003024' 1#:      CALL   PPSSPD        ;Set the speed in mode register 2
46         ;
47         ; Initialize the command register
48         ;
49         002462 012737 000067 173406  MOV     #PP$CMF!PP$RE,@#PP$CMR ;Initialize command register
50         ;
51         ; Enable receiver interrupts
52         ;
53         002470 112737 000035 173202  MOVB   #IM$CRM!IM$PPR,@#ICOC$SR ;Clear IRR&IMR flags for receiver
54         ;
55         ; Disable transmitter (set mask bit in IMR), but raise a transmitter
56         ; request so we will get an interrupt by clearing the mask flag later.
57         ;

```

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



PPTINT -- Transmitter interrupt

```

1          .SBTTL  PPTINT -- Transmitter interrupt
2          ;-----
3          ; Interrupt service routine for printer port transmitter interrupts.
4          ;
5 002516 010046 PPTINT: MOV     R0,-(SP)
6 002520 010446      MOV     R4,-(SP)
7 002522 013746 0000000  MOV     INTPRI,-(SP) ;Save current interrupt priority
8 002526 005037 0000000  CLR     INTPRI      ;Say running priority = 7
9          ;
10         ; Disable interrupts from transmitter
11         ;
12 002532 112737 000076 173202      MOVB   #IM$SM!IM$PPT,@#ICOCSR ;Set interrupt mask for transmitter
13         ;
14         ; Get number of line connected to printer port
15         ;
16 002540 013704 000046'      MOV     PPLINE,R4      ;Get index # of line connected to printer
17         ;
18         ; Say line is no longer transmitting a character
19         ;
20 002544 042764 0000000 0000000  BIC     #$XCHAR,LSW3(R4);Say line no longer transmitting a character
21         ;
22         ; Get next character to transmit
23         ;
24 002552 004774 0000000      CALL    @LOUTIR(R4)    ;Get next character to transmit
25 002556 103413      BCS    1$             ;Br if no character available
26         ;
27         ; Transmit the character contained in R0
28         ;
29 002560 052764 0000000 0000000  BIS     #$XCHAR,LSW3(R4);Set transmitter busy flag
30 002566 042764 0000000 0000000  BIC     #$DITIM,LSW5(R4);Start timer to catch lost interrupts
31 002574 110037 173400      MOVB   R0,@#PP$DBR    ;Transmit the character
32         ;
33         ; Enable transmitter interrupts
34         ;
35 002600 112737 000056 173202      MOVB   #IM$CM!IM$PPT,@#ICOCSR ;Clear interrupt mask flag
36         ;
37         ; Finished. Return from interrupt
38         ;
39 002606 012637 0000000 1$:  MOV     (SP)+,INTPRI    ;Restore interrupt priority
40 002612 012604      MOV     (SP)+,R4
41 002614 012600      MOV     (SP)+,R0
42 002616 000002      RTI

```

PPSTRT -- Start output to printer port

```

1          .SBTTL  PPSTRT -- Start output to printer port
2          ;
3          ; Subroutine called to attempt to start output for the printer port.
4          ; Enable transmitter interrupt.
5          ; R1 = Physical line number.
6          ;
7 002620 032761 0000000 0000000 PPSTRT: BIT    $$XCHAR,LSW3(R1)    ; Is the transmitter busy now?
8 002626 001011          BNE    1$                                ; Br if yes
9 002630 112737 000076 173202      MOVB   #IM$SM!IM$PPT,@#ICOCSR ; Set interrupt mask bit
10 002636 112737 000136 173202      MOVB   #IM$SR!IM$PPT,@#ICOCSR ; Set interrupt request flag
11 002644 112737 000056 173202      MOVB   #IM$CM!IM$PPT,@#ICOCSR ; Clear interrupt mask flag for xmitter
12 002652 000207          1$:   RETURN

```

```

1          .SBTTL  PPRINT -- Receiver interrupt
2          ;-----
3          ; Interrupt service routine for printer port receiver interrupt.
4          ;
5 002654   PPRINT:
6          ;
7          ; Call routine to save R0, R1, R4, and R5.
8          ;
9 002654   004577   000000G   JSR      R5,@TTRSAV      ;Save some registers
10         ;
11         ; R0, R1, R4 and R5 are now available.
12         ; Get number of line connected to printer port.
13         ;
14 002660   013704   000046'   MOV      PPLINE,R4      ;Get line index number
15         ;
16         ; Get the character from the receiver
17         ;
18 002664   113705   173400   MOVVB   @PP$DBR,R5      ;Get the received character
19 002670   042705   177400   BIC     #^C<377>,R5    ;Kill sign extension
20         ;
21         ; Set flags in R5 to simulate status flags returned by DL11 receiver
22         ;
23 002674   032737   000010   173402   BIT     #PP$PE,@PP$STR ;Was a parity error detected?
24 002702   001402                   BEQ     1$              ;Br if not
25 002704   052705   000000C   BIS     #RBERR!RCVPAR,R5;Set parity error flag
26 002710   032737   000040   173402   1$:    BIT     #PP$FE,@PP$STR ;Was a framing error detected?
27 002716   001402                   BEQ     2$              ;Br if not
28 002720   052705   000000C   BIS     #RBERR!FRMERR,R5;Set framing error flag
29 002724   032737   000020   173402   2$:    BIT     #PP$OE,@PP$STR ;Did an overrun error occur?
30 002732   001402                   BEQ     3$              ;Br if not
31 002734   052705   000000C   BIS     #RBERR!OVRRUN,R5;Set overrun error flag
32         ;
33         ; Clear error flags in receiver
34         ;
35 002740   032705   000000G   3$:    BIT     #RBERR,R5      ;Were any errors detected?
36 002744   001403                   BEQ     4$              ;Br if not
37 002746   052737   000020   173406   BIS     #PP$RE,@PP$CMR ;Clear error flags
38         ;
39         ; Call TTINPT to process the character we received
40         ;
41 002754   004777   000000G   4$:    CALL   @TTINPT      ;Enter character input routine
42 002760   000403                   BR      9$              ;Return from interrupt
43         ;
44         ; We got an interrupt but no character was available
45         ;
46 002762   112737   000115   173202   6$:    MOVVB   #IM$CR!IM$PPR,@#ICOC$SR ;Clear interrupt request flag
47         ;
48         ; Return from interrupt
49         ;
50 002770   000207   9$:    RETURN                ;Return from interrupt

```

PPGDSS -- Get data set status for printer port

```

1          .SBTTL  PPGDSS -- Get data set status for printer port
2          ;-----
3          ; This is the device-dependent routine for the printer port which
4          ; returns generic flags indicating the status of the data set for
5          ; this line.
6          ;
7          ; Inputs:
8          ;   R1 = Physical line index number
9          ;
10         ; Outputs:
11         ;   R0 = Generic status flags
12         ;
13 002772 005000 PPGDSS: CLR      R0          ; Say no ring, carrier, or DTR
14         ;
15         ; Finished
16         ;
17 002774 000207          RETURN

```

PPSBRK -- Control break transmission

```

1          .SBTTL  PPSBRK -- Control break transmission
2          ;-----
3          ; PPSBRK is called to start or stop transmitting a break character
4          ; to the printer port.
5          ;
6          ; Inputs:
7          ; R1 = Physical line index number.
8          ; R0 = Break control flag (MS$BRK)
9          ;
10         PPSBRK:
11         ;
12         ; See if we are to start or stop sending a break
13         ;
14         002776 032700 0000000  BIT    #MS$BRK,R0    ;Start or stop sending break?
15         003002 001404          BEQ    1$           ;Br to stop
16         ;
17         ; Start sending a break character
18         ;
19         003004 152737 000010 173406  BISB   #PP$FB,@#PP$CMR ;Start sending a break
20         003012 000403          BR     9$
21         ;
22         ; Stop sending a break character
23         ;
24         003014 142737 000010 173406 1$:   BICB   #PP$FB,@#PP$CMR ;Stop sending a break
25         ;
26         ; Finished
27         ;
28         003022 000207          9$:   RETURN

```

```

1          .SBTTL  PPSSPD -- Set transmission speed for printer port
2          ;-----
3          ; PPSSPD is called to set the transmission speed for the printer port.
4          ;
5          ; Inputs:
6          ;   R0 = Speed code.
7          ;   R1 = Physical line index number.
8          ;
9 003024  010346  PPSSPD: MOV      R3,-(SP)
10         ;
11         ; Save new parameter flags for line
12         ;
13 003026  110061  0000010  MOVB   R0,LMXPRM+1(R1) ; Save new flags for line
14         ;
15         ; Build value to store in mode register 1 (char length and parity)
16         ;
17 003032  012703  000102  MOV    #PP$M1F,R3    ; Get standard mode register 1 flags
18 003036  032700  0000000  BIT    #LP$7BT,R0    ; 7 bit characters wanted?
19 003042  001003  BNE    2$            ; Br if yes
20 003044  052703  000014  BIS    #PP$BBT,R3    ; Select 8 bit characters
21 003050  000402  BR     3$            ;
22 003052  052703  000010  2$:   BIS    #PP$7BT,R3 ; Select 7 bit characters
23 003056  032700  0000000  3$:   BIT    #LP$PAR,R0 ; Parity wanted?
24 003062  001407  BEQ    1$            ; Br if not
25 003064  052703  000020  BIS    #PP$PAR,R3    ; Enable parity
26 003070  032700  0000000  BIT    #LP$ODD,R0    ; Odd parity wanted?
27 003074  001002  BNE    1$            ; Br if yes
28 003076  052703  000040  BIS    #PP$EVN,R3    ; Select even parity
29 003102  1$:   DISABL ; ** Disable interrupts **
30 003110  105737  173406  TSTB  @#PP$CMR      ; Access command reg to reset mode reg ptr
31 003114  110337  173404  MOVB  R3,@#PP$MDR   ; Store value into mode register 1
32         ;
33         ; Store baud rate code into mode register 2
34         ;
35 003120  042700  0000000  BIC    #^C<LP$SPD>,R0 ; Clear all but baud rate code
36 003124  052700  000260  BIS    #PP$M2F,R0    ; Set standard flags for mode register 2
37 003130  110037  173404  MOVB  R0,@#PP$MDR   ; Store value into mode register 2
38 003134  ENABL ; ** Enable interrupts **
39         ;
40         ; Finished
41         ;
42 003142  012603  MOV    (SP)+,R3
43 003144  000207  RETURN

```

\*\*\* Communications Port Control Routines \*\*\*

```

1          .SBTTL  *** Communications Port Control Routines ***
2          .SBTTL  CPINIT -- Communications port initialization
3          ;-----
4          ; Perform initialization for the communications port.
5          ;
6          ; Inputs:
7          ; R1 = Index number of time-sharing line connected to comm port
8          ;
9 003146   CPINIT:
10         ;
11         ; Remember which line is connected to comm port
12         ;
13 003146   010137   000050'           MOV     R1,CPLINE      ;Remember line index number
14 003152   012761   000000G 000000G   MOV     #CDX$PC,LCDTYP(R1) ;Set type of communications controller
15         ;
16         ; Set addresses of communication port device-dependent routines
17         ;
18 003160   012737   003772' 000000C   MOV     #CP$STRT,CD$STRT+CDX$PC ;Set address of start routine
19 003166   012737   004060' 000000C   MOV     #CP$CLOCK,CD$CLOCK+CDX$PC ;Set address of timer-driven routine
20 003174   012737   004122' 000000C   MOV     #CP$XOFF,CDS$XOFF+CDX$PC ;Routine to send XOFF
21 003202   012737   004210' 000000C   MOV     #CP$XON,CDS$XON+CDX$PC ;Routine to send XON
22 003210   012737   004276' 000000C   MOV     #CP$GDSS,CD$GDSS+CDX$PC ;Routine to get data set status flags
23 003216   012737   004346' 000000C   MOV     #CP$SDSS,CDS$DSS+CDX$PC ;Routine to do data set control
24 003224   012737   004374' 000000C   MOV     #CP$BRK,CDS$BRK+CDX$PC ;Routine to control break transmission
25 003232   012737   004462' 000000C   MOV     #CP$SPD,CD$SPD+CDX$PC ;Routine to set transmission speed
26         ;
27         ; Connect to interrupt vectors
28         ;
29 003240   012737   003504' 000210   MOV     #CPCINT,@#CP$RTV;Receiver/transmitter interrupt
30 003246   012737   000340 000212   MOV     #340,@#CP$RTV+2 ;Set prio = 7
31 003254   012737   003504' 000214   MOV     #CPCINT,@#CP$MCV;Modem change interrupt
32 003262   012737   000340 000216   MOV     #340,@#CP$MCV+2 ;Set prio = 7
33         ;
34         ; Initialize values in CSR A
35         ;
36 003270   012700   173302           MOV     #CP$CAR,R0      ;Get pointer to CSR A
37 003274   112710   000030           MOVB   #CP$CR,(R0)     ;Reset the channel
38 003300   000240           NOP                     ;Delay time for channel clear operation
39 003302   000240           NOP
40 003304   000240           NOP
41 003306   000240           NOP
42 003310   112710   000300           MOVB   #300,(R0)      ;Reset transmitter underrun condition
43 003314   112710   000004           MOVB   #CP$WR4,(R0)   ;Select write register 4
44 003320   112710   000104           MOVB   #CP$AW4,(R0)   ;16x clock rate, 1 stop bit, asynch
45 003324   112710   000003           MOVB   #CP$WR3,(R0)   ;Select write register 3
46 003330   112710   000001           MOVB   #CP$AW3,(R0)   ;Enable receiver and set 8-bit char len
47 003334   112710   000005           MOVB   #CP$WR5,(R0)   ;Select write register 5
48 003340   112710   000010           MOVB   #CP$AW5,(R0)   ;8-bit char, enable transmitter
49 003344   112710   000002           MOVB   #CP$WR2,(R0)   ;Select write register 2
50 003350   112710   000000           MOVB   #0,(R0)        ;Store all zero in WR2
51 003354   112710   000020           MOVB   #CP$RES,(R0)   ;Reset external interrupts
52 003360   116100   000001G           MOVB   LMXPRM+1(R1),R0 ;Get flags for line
53 003364   001002           BNE    1$              ;Br if speed specified
54 003366   012700   000000G           MOV     #S9600,R0      ;Default to 9600 baud
55 003372   004737   004462'           1$:    CALL    CP$SPD     ;Set speed, parity, character length
56         ;
57         ; 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 000000@ 000000@ 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
```



```

1                                     .SBTTL  CPCINT -- Receiver/Transmitter interrupt routine
2                                     ;-----
3                                     ; CPCINT is the interrupt entry point for both receiver and transmitter
4                                     ; interrupts.
5                                     ;
6 003504 010046 CPCINT: MOV      RO,-(SP)      ;Get a work register
7 003506 112737 000002 173306      MOVVB   #CP*RR2,@#CP*CBR;Select CSR B read register 2
8 003514 113700 173306      MOVVB   @#CP*CBR,RO    ;Get the interrupt flags from RR2
9 003520 006200      ASR      RO          ;Get value as word table index
10 003522 042700 177761      BIC     #^C<16>,RO    ;Clear all but interrupt code flags
11 003526 000170 003532'      JMP     @CPIVEC(RO)   ;Jump to appropriate interrupt service rtn
12                                     ;
13                                     ; Jump vector based on interrupt type
14                                     ;
15 003532 003572' CPIVEC: .WORD   CPIERR      ;0 - Invalid
16 003534 003572'      .WORD   CPIERR      ;1 - Invalid
17 003536 003572'      .WORD   CPIERR      ;2 - Invalid
18 003540 003572'      .WORD   CPIERR      ;3 - Invalid
19 003542 003672'      .WORD   CPITR       ;4 - Transmitter buffer empty
20 003544 003564'      .WORD   CPIESC      ;5 - External status change
21 003546 003604'      .WORD   CPIREC      ;6 - Received character
22 003550 003552'      .WORD   CPISRC      ;7 - Special receiver condition
23                                     ;
24                                     ; Special receiver condition interrupt
25                                     ;
26 003552 112737 000060 173302 CPISRC: MOVVB   #CP*ER,@#CP*CAR ;Reset error status flags
27 003560 000137 003604'      JMP     CPIREC        ;Process any character we got with errors
28                                     ;
29                                     ; External status change interrupt processing
30                                     ;
31 003564 112737 000020 173302 CPIESC: MOVVB   #CP*RES,@#CP*CAR;Reset external status
32                                     ;
33                                     ; Invalid interrupt code.
34                                     ; Reset interrupt status and return.
35                                     ;
36 003572 112737 000070 173302 CPIERR: MOVVB   #CP*EI,@#CP*CAR ;Declare end of interrupt
37 003600 012600      MOV      (SP)+,RO
38 003602 000002      RTI          ;Return from interrupt

```

CPIREC -- Received character from comm port

```

1          .SBTTL  CPIREC -- Received character from comm port
2          ;-----
3          ; CPIREC is the interrupt service routine called when we get an
4          ; interrupt from the comm port saying that it has received a character.
5          ;
6          ; On entry, R0 is on the stack.
7          ;
8 003604 012600 CPIREC: MOV      (SP)+,R0      ;Restore R0
9          ;
10         ; Call routine to save R0, R1, R4, and R5
11         ;
12 003606 004577 000000G JSR      R5,@TTRSAV      ;Save registers
13         ;
14         ; R0, R1, R4 and R5 are available.
15         ; See if a framing error was detected
16         ;
17 003612 005005          CLR      R5          ;Build character and flags in R5
18 003614 112737 000000 173302 MOVB    #CP$RRO,@#CP$CAR;Select read register 0
19 003622 113704 173302 MOVB    @#CP$CAR,R4      ;Get contents of read register 0
20 003626 132704 000200 BITB    #CP$BR,R4      ;Was a break detected?
21 003632 001402 BEQ      3$          ;Br if not
22 003634 052705 000000G BIS      #FRMERR,R5      ;Set framing error flag
23 003640 112737 000020 173302 3$: MOVB    #CP$RES,@#CP$CAR;Reset register 0 flags
24         ;
25         ; Get the character from the receiver
26         ;
27 003646 153705 173300 BISB    @#CP$DBR,R5      ;Get the received character
28         ;
29         ; Reenable interrupts from receiver
30         ;
31 003652 112737 000070 173302 MOVB    #CP$EI,@#CP$CAR ;Declare end of interrupt
32         ;
33         ; Get number of line connected to communications port
34         ;
35 003660 013704 000050' MOV      CPLINE,R4      ;Get line index number
36         ;
37         ; Call TTINPT to process the character
38         ;
39 003664 004777 000000G CALL    @TTINPT          ;Enter routine to process the character
40         ;
41         ; Return from interrupt
42         ;
43 003670 000207 RETURN                                ;Return from interrupt through TTRSAV

```

CPITR -- Transmitter interrupt

```

1          .SBTTL  CPITR  -- Transmitter interrupt
2          ;-----
3          ; CPITR is the interrupt service routine for the communications port
4          ; transmitter.
5          ;
6          ; On entry, R0 is on the stack.
7          ;
8 003672  010446  CPITR:  MOV     R4,-(SP)      ;Stack R4 also
9 003674  013746  000000G  MOV     INTPRI,-(SP)  ;Save current interrupt priority
10 003700  005037  000000G  CLR     INTPRI       ;Say running interrupt priority is 7
11          ;
12          ; Say that line is no longer transmitting a character
13          ;
14 003704  013704  000050'  MOV     CPLINE,R4    ;Get # of line connected to comm port
15 003710  042764  000000G 000000G  BIC     ##XCHAR,LSW3(R4);Clear transmitting-character flag for line
16 003716  112737  000050  173302  MOVB   #CP$RTI,@#CP$CAR;Reset transmitter interrupt flag
17 003724  112737  000070  173302  MOVB   #CP$EI,@#CP$CAR ;Declare end of interrupt
18          ;
19          ; Get next character to transmit
20          ;
21 003732  004774  000000G  CALL   @LOUTIR(R4)  ;Get next char to transmit
22 003736  103410  BCS     9$          ;Br if no more chars to send
23          ;
24          ; Transmit the character
25          ;
26 003740  052764  000000G 000000G  BIS     ##XCHAR,LSW3(R4);Say a character is being transmitted
27 003746  042764  000000G 000000G  BIC     ##OITIM,LSW5(R4);Start timer to catch lost interrupts
28 003754  110037  173300  MOVB   RO,@#CP$DBR  ;Transmit the character
29          ;
30          ; Finished
31          ;
32 003760  012637  000000G  9$:    MOV     (SP)+,INTPRI ;Restore interrupt priority level
33 003764  012604  MOV     (SP)+,R4
34 003766  012600  MOV     (SP)+,RO
35 003770  000002  RTI          ;Return from interrupt

```

CPSTRT -- Start output to communications port

```

1          .SBTTL  CPSTRT -- Start output to communications port
2          ;-----
3          ;  CPSTRT is called to initiate output to the communications port.
4          ;
5          ;  R1 = Line index number of line being started.
6          ;
7 003772  010046  CPSTRT: MOV      R0,-(SP)
8 003774  010446          MOV      R4,-(SP)
9          ;
10         ;  Disable communications port interrupts
11         ;
12 003776          DISABL          ;;Disable interrupts
13         ;
14         ;  See if the communications port transmitter is already busy
15         ;
16 004004  032761  0000000 0000000  BIT      $$XCHAR,LSW3(R1)  ;;Is the transmitter busy now?
17 004012  001014          BNE      1$          ;;Br if yes
18         ;
19         ;  Transmitter is idle.
20         ;  Get next character to transmit
21         ;
22 004014  010104          MOV      R1,R4          ;;Get line # to R4 for NEDCHR
23 004016  004774  0000000  CALL     @LOUTIR(R4)  ;;Get a character to transmit
24 004022  103410          BCS     1$          ;;Br if nothing to transmit
25         ;
26         ;  Start transmission
27         ;
28 004024  052761  0000000 0000000  BIS      $$XCHAR,LSW3(R1)  ;;Say a character being transmitted
29 004032  042761  0000000 0000000  BIC      $$OITIM,LSW5(R1)  ;;Start timer to catch lost interrupt
30 004040  110037  173300  MOVB     RO,@#CP#DBR  ;;Transmit the character
31         ;
32         ;  Enable the transmitter interrupt
33         ;
34 004044          1$:  ENABL          ;Enable interrupts
35         ;
36         ;  Finished
37         ;
38 004052  012604          MOV      (SP)+,R4
39 004054  012600          MOV      (SP)+,RO
40 004056  000207          RETURN

```

```
1 .SBTTL CPCLOK -- Timer driven routine for communications port
2 ;-----
3 ; CPCLOK is a timer driven routine called periodically to check on
4 ; the status of the communications port.
5 ;
6 ; Inputs:
7 ; R1 = Line index number.
8 ;
9 004060 CPCLOK:
10 ;
11 ; Check for lost output interrupts
12 ;
13 004060 032761 0000000 0000000 BIT ##0ITIM,LSW5(R1);Have we started timer interval?
14 004066 001411 BEQ 1$ ;Br if not
15 004070 032761 0000000 0000000 BIT ##XCHAR,LSW3(R1);Are we still waiting for an interrupt?
16 004076 001410 BEQ 9$ ;Br if not
17 004100 042761 0000000 0000000 BIC ##XCHAR,LSW3(R1);Say wait is over
18 004106 004737 003772' CALL CPSTRT ;Try to restart transmitter
19 004112 052761 0000000 0000000 1$: BIS ##0ITIM,LSW5(R1);Start timed interval
20 ;
21 ; Finished
22 ;
23 004120 000207 9$: RETURN
```

```

1          .SBTTL  CPXOFF -- Transmit XOFF to communications port
2          ;-----
3          ; CPXOFF is called to stuff an XOFF character into the output stream
4          ; for the communications port.
5          ;
6          ; Inputs:
7          ; R1 = Line index number.
8          ;
9 004122   CPXOFF: DISABL          ;;; ** Disable interrupts **
10         ;
11         ; Set flag that says XOFF has been sent
12         ;
13 004130   052761  0000000 0000000   BIS    ##HISTP,LSW10(R1);; Remember that XOFF has been sent
14         ;
15         ; See if the transmitter is busy now
16         ;
17 004136   032761  0000000 0000000   BIT    ##XCHAR,LSW3(R1);; Is the transmitter busy now?
18 004144   001404          BEQ    1$          ;;; Br if not busy now
19         ;
20         ; The transmitter is busy now.
21         ; Set flag which will cause output character generator to send XOFF
22         ;
23 004146   052761  0000000 0000000   BIS    ##SXOFF,LSW10(R1);; Set flag to cause XOFF to be sent
24 004154   000411          BR     9$          ;;;
25         ;
26         ; Transmitter is not busy.
27         ; Transmit an XOFF character.
28         ;
29 004156   052761  0000000 0000000 1$:  BIS    ##XCHAR,LSW3(R1);; Say transmitter busy
30 004164   042761  0000000 0000000   BIC    ##OITIM,LSW5(R1);; Start output interrupt timer
31 004172   112737  000023  173300   MOVB   #23,@#CP#DBR    ;;; Transmit XOFF
32         ;
33         ; Finished
34         ;
35 004200   9$:  ENABL          ;;; ** Enable interrupts **
36 004206   000207          RETURN

```

```
1 .SBTTL CPXON -- Transmit XON to communications port
2 ;-----
3 ; CPXON is called to stuff an XON character into the output stream
4 ; for the communications port.
5 ;
6 ; Inputs:
7 ; R1 = Line index number.
8 ;
9 004210 CPXON: DISABL ;;;** Disable interrupts **
10 ;
11 ; Clear flag that says XOFF has been sent
12 ;
13 004216 042761 0000000 0000000 BIC ##HISTP,LSW10(R1);;Clear flag that says XOFF sent
14 ;
15 ; See if the transmitter is busy now
16 ;
17 004224 032761 0000000 0000000 BIT ##XCHAR,LSW3(R1);;Is the transmitter busy now?
18 004232 001404 BEQ 1$ ;;;Br if not busy now
19 ;
20 ; The transmitter is busy now.
21 ; Set flag which will cause output character generator to send XON
22 ;
23 004234 052761 0000000 0000000 BIS ##SXON,LSW10(R1);;Set flag to cause XON to be sent
24 004242 000411 BR 9$ ;;;
25 ;
26 ; Transmitter is not busy.
27 ; Transmit an XON character.
28 ;
29 004244 052761 0000000 0000000 1$: BIS ##XCHAR,LSW3(R1);;Say transmitter busy
30 004252 042761 0000000 0000000 BIC ##OITIM,LSW5(R1);;Start output interrupt timer
31 004260 112737 000021 173300 MOVB #21,@#CP#DBR ;;;Transmit XON
32 ;
33 ; Finished
34 ;
35 004266 9$: ENABL ;;;** Enable interrupts **
36 004274 000207 RETURN
37
38
```

```
1          . SBTTL  CPGDSS -- Get communications port data set status
2          ;-----
3          ; CPGDSS is called to get the data set (modem) status for the
4          ; communications port.
5          ;
6          ; Inputs:
7          ; R1 = Physical line index number
8          ;
9          ; Outputs:
10         ; R0 = Generic data set status flags (MS$xxx)
11         ;
12 004276 005000 CPGDSS: CLR      R0          ; Develop result in R0
13         ;
14         ; See if the phone is ringing
15         ;
16 004300 132737 000100 173312      BITB   #CP$RI,@#CP$M1R ; Is the phone ringing?
17 004306 001402      BEQ     1$          ; Br if not
18 004310 052700 0000000      BIS     #MS$RNG,R0      ; Set ringing flag
19         ;
20         ; See if carrier is detected
21         ;
22 004314 132737 000020 173312 1$:   BITB   #CP$CD,@#CP$M1R ; Do we have carrier detect?
23 004322 001402      BEQ     2$          ; Br if not
24 004324 052700 0000000      BIS     #MS$CAR,R0      ; Set carrier-detect flag
25         ;
26         ; See if Data Terminal Ready is asserted
27         ;
28 004330 132737 000020 173310 2$:   BITB   #CP$DTR,@#CP$MOR; Are we asserting DTR?
29 004336 001402      BEQ     3$          ; Br if not
30 004340 052700 0000000      BIS     #MS$DTR,R0      ; Remember DTR is asserted
31         ;
32         ; Finished
33         ;
34 004344 000207      3$:   RETURN
```



```
1 .SBTTTL CPSDSS -- Set data set status for communications port
2 ;-----
3 ; CPSDSS is called to control the data set (modem) status for the
4 ; communications port.
5 ;
6 ; Inputs:
7 ; R1 = Physical line index number
8 ; RO = Data set control flags (MS$xxx)
9 ;
10 004346 CPSDSS:
11 ;
12 ; See if we should raise or drop Data Terminal Ready
13 ;
14 004346 032700 0000000 BIT #MS$DTR,RO ;Raise or drop DTR?
15 004352 001404 BEQ 1$ ;Br to drop DTR
16 ;
17 ; Raise DTR
18 ;
19 004354 152737 000020 173310 BISB #CP$DTR,@#CP$MOR;Raise DTR
20 004362 000403 BR 9$
21 ;
22 ; Drop DTR
23 ;
24 004364 142737 000020 173310 1$: BICB #CP$DTR,@#CP$MOR;Drop DTR
25 ;
26 ; Finished
27 ;
28 004372 000207 9$: RETURN
```

```
1 .SBTTL CPSBRK -- Control break transmission on communications port
2 ;-----
3 ; CPSBRK is called to start or stop transmitting a break character to
4 ; the communications port.
5 ;
6 ; Inputs:
7 ; R0 = Break control flag (MS$BRK)
8 ; R1 = Physical line index number.
9 ;
10 004374 010246 CPSBRK: MOV R2, -(SP)
11 ;
12 ; Set standard flags for write-register 5
13 ;
14 004376 012702 000050 MOV #CP$AW5!CP$7BT, R2 ; Standard flags and 7-bit char length
15 004402 132761 000000G 000001G BITB #LP$7BT, LMXPRM+1(R1); 7-bit characters wanted?
16 004410 001002 BNE 2$ ; Br if yes
17 004412 052702 000140 BIS #CP$8BT, R2 ; Select 8-bit characters
18 ;
19 ; See if we want to start or stop sending a break
20 ;
21 004416 032700 000000G 2$: BIT #MS$BRK, R0 ; Start or stop sending break?
22 004422 001402 BEQ 1$ ; Br if stop
23 004424 052702 000020 BIS #CP$SB, R2 ; Set send-break flag
24 ;
25 ; Store parameters into control register
26 ;
27 004430 1$: DISABL ;;; ** Disable interrupts **
28 004436 112737 000005 173302 MOVB #CP$WR5, @CP$CAR ;;; Select write register 5
29 004444 110237 173302 MOVB R2, @CP$CAR ;;; Start or stop sending break
30 004450 ENABL ;;; ** Enable interrupts **
31 ;
32 ; Finished
33 ;
34 004456 012602 9$: MOV (SP)+, R2
35 004460 000207 RETURN
```

```

1          .SBTTL  CPSSPD -- Set speed for communications port
2          ;-----
3          ; CPSSPD is called to set the transmit/receive speed for
4          ; the communications port.
5          ;
6          ; Inputs:
7          ; R0 = Speed code.
8          ; R1 = Physical line index number.
9          ;
10         CPSSPD: MOV      R3, -(SP)
11         MOV      R4, -(SP)
12         MOV      R5, -(SP)
13         ;
14         ; Set baud rate in line parameter word for this line
15         ;
16         004470 110061 0000010      MOVVB   R0, LMXPRM+1(R1) ; Save new flags for line
17         ;
18         ; Store speed value into hardware control register
19         ;
20         004474 010003                MOV      R0, R3          ; Get all flags
21         004476 042703 0000000      BIC      #^C<LP$SPD>, R3 ; Clear all but speed code
22         004502 010346                MOV      R3, -(SP)      ; Save speed code
23         004504 072327 0000004      ASH     #4, R3          ; Shift speed value left 4 bits
24         004510 052603                BIS      (SP)+, R3      ; Combine with unshifted value
25         004512 110337 173314      MOVVB   R3, @#CP$BRR    ; Set baud rate in hardware register
26         ;
27         ; Set parity control
28         ;
29         004516 012704 173302      MOV      #CP$CAR, R4    ; Get pointer to CSR A
30         004522 012703 000104      MOV      #CP$AW4, R3    ; Get standard bits for WR4
31         004526 032700 0000000      BIT      #LP$PAR, R0    ; Is parity wanted?
32         004532 001407                BEQ     1$              ; Br if not
33         004534 052703 0000001      BIS      #CP$PAR, R3    ; Set parity-enable flag
34         004540 032700 0000000      BIT      #LP$ODD, R0    ; Is odd parity wanted?
35         004544 001002                BNE     1$              ; Br if yes
36         004546 052703 0000002      BIS      #CP$EVN, R3    ; Select even parity
37         004552 112714 0000004      1$:    MOVVB   #CP$WR4, (R4) ; Select write register 4
38         004556 110314                MOVVB   R3, (R4)        ; Set parity control flags
39         ;
40         ; Set character length
41         ;
42         004560 012703 0000001      MOV      #CP$AW3, R3    ; Get standard flags for WR3
43         004564 012705 0000010      MOV      #CP$AW5, R5    ; Get standard flags for WR5
44         004570 032700 0000000      BIT      #LP$7BT, R0    ; 7 bit characters wanted?
45         004574 001005                BNE     2$              ; Br if yes
46         004576 052705 0001040      BIS      #CP$8BT, R5    ; Select 8 bit characters
47         004602 052703 000300      BIS      #CP$8BR, R3
48         004606 000404                BR      3$
49         004610 052705 0000040      2$:    BIS      #CP$7BT, R5    ; Select 7 bit characters
50         004614 052703 000100      BIS      #CP$7BR, R3
51         004620                3$:    DISABL                ; ** Disable interrupts **
52         004626 112714 0000003      MOVVB   #CP$WR3, (R4)    ; Select write register 3
53         004632 110314                MOVVB   R3, (R4)        ; Set length for received characters
54         004634 112714 0000005      MOVVB   #CP$WR5, (R4)    ; Select write register 5
55         004640 110514                MOVVB   R5, (R4)        ; Set length for transmitted characters
56         004642                ENABL                ; ** Enable interrupts **
57         ;

```

```
58          ; Finished  
59          ;  
60 004650 012605      MOV      (SP)+, R5  
61 004652 012604      MOV      (SP)+, R4  
62 004654 012603      MOV      (SP)+, R3  
63 004656 000207      RETURN
```

```

1          .IF      NE,QPASM          ; Assemble if quad line unit code wanted
2          .SBTTL   ***  Quad Serial Line Unit Routines  ***
3          .SBTTL   QPINIT -- Initialize quad serial line unit
4          ;-----
5          ; QPINIT is called to initialize the quad serial line unit
6          ;
7 004660  010546  QPINIT: MOV      R5,-(SP)
8          ;
9          ; Determine if a quad line unit is installed
10         ;
11 004662  012746  000064          MOV      #QP$ID,-(SP) ; Stack device ID for quad line unit
12 004666  004737  001172'        CALL    QPRCSR ; Get CSR for quad line unit
13 004672  012605          MOV      (SP)+,R5 ; Get returned CSR address
14 004674  103014          BCC     2$ ; Br if it is installed
15         ;
16         ; There is no quad line unit installed.
17         ; Mark all lines associated with quad line unit as dead.
18         ;
19 004676  012705  0000000        MOV      #LSTHL,R5 ; Get index to last line
20 004702  005765  0000000        3$:    TST     LMXNUM(R5) ; Is this a quad unit line?
21 004706  001403          BEQ     4$ ; Br if not
22 004710  052765  0000000 0000000  BIS     ##DEAD,LSW3(R5) ; Mark line as dead
23 004716  162705  0000002        4$:    SUB     #2,R5 ; Get next line index
24 004722  003367          BGT     3$ ; Br if more lines to do
25 004724  000502          BR      9$ ; Finished
26         ;
27         ; There is a quad line unit installed
28         ;
29 004726  010537  000066'        2$:    MOV      R5,QPCSR ; Save CSR address
30 004732  132765  000004  000004  BITB   #QP$22C,QPRMSR(R5); Do we have modem control on lines 0 and 2?
31 004740  001402          BEQ     1$ ; Br if not
32 004742  105237  000110'        INCB   QPMODM ; Remember we have modem control on two lines
33 004746  142765  000100  000004  1$:    BICB   #QP$IEN,QPRMSR(R5); Clear interrupt-enable bit
34 004754  012746  000064          MOV      #QP$ID,-(SP) ; Stack device ID
35 004760  004737  001134'        CALL    QPRVEC ; Get vector address
36 004764  012637  000070'        MOV      (SP)+,QPVEC ; Save vector address
37 004770  012700  000064          MOV      #QP$ID,R0 ; Get device ID
38 004774  004737  001254'        CALL    GETSLT ; Get slot # where device is installed
39 005000  110037  000111'        MOVB   R0,QPSLOT ; Save the slot number
40         ;
41         ; Set addresses of quad line unit device-dependent routines
42         ;
43 005004  012737  005310' 000000C  MOV      #QP$STRT,CDSTRT+CDX$QP ; Start-output routine
44 005012  012737  006054' 000000C  MOV      #QP$GDSS,CDGDSS+CDX$QP ; Get data set status
45 005020  012737  006160' 000000C  MOV      #QP$SDSS,CDSDDS+CDX$QP ; Set data set status
46 005026  012737  006016' 000000C  MOV      #QP$BRK,CDSBRK+CDX$QP ; Control break transmission
47 005034  012737  005632' 000000C  MOV      #QP$SPD,CDSSPD+CDX$QP ; Set line speed
48         ;
49         ; Initialize Output Port Configuration Register
50         ;
51 005042  105065  000132          CLRB   QPROPC(R5) ; All bits are off in OPCR
52 005046  105065  000172          CLRB   QPROPC+QPRUOF(R5) ; DUART 1
53         ;
54         ; Initialize Auxiliary Control Register
55         ;
56 005052  112765  000360  000110  MOVB   #QP$AWA,QPRACR(R5) ; Initialize ACR
57 005060  112765  000360  000150  MOVB   #QP$AWA,QPRACR+QPRUOF(R5) ; DUART 1

```

QPINIT -- Initialize quad serial line unit

```

58 ;
59 ; Initialize Interrupt Mask Register
60 ;
61 005066 112765 000063 000112      MOVB    #QP$AWI,QPRIMR(R5) ;Intialize IMR
62 005074 112765 000063 000152      MOVB    #QP$AWI,QPRIMR+QPRUOF(R5) ;DUART 1
63 ;
64 ; Initialize output port
65 ;
66 005102 112765 000377 000136      MOVB    #377,QPRROP(R5) ;Reset all output port bits
67 005110 112765 000377 000176      MOVB    #377,QPRROP+QPRUOF(R5) ;DUART 1
68 ;
69 ; Initialize interrupt controller
70 ;
71 005116 012700 000030                MOV     #IM$CRM,RO        ;Get command to clear IRR and IMR int flags
72 005122 153700 000111'              BISB   QPSLOT,RO         ;Or in slot number where QP controller is
73 005126 110037 173206              MOVB   RO,@#IC1CSR      ;Clear IRR and IMR flags for quad port
74 ;
75 ; Finished
76 ;
77 005132 012605                      9$:    MOV     (SP)+,R5
78 005134 000207                      RETURN

```

QPLINE -- Initialize a line connected to quad line unit

```

1          .SBTTL  QPLINE -- Initialize a line connected to quad line unit
2          ;-----
3          ; QPLINE is called to initialize a specific line that is connected to
4          ; the quad serial line unit.
5          ;
6          ; Inputs:
7          ; R1 = Line index number.
8          ;
9 005136  010544  QPLINE: MOV      R5, -(SP)
10         ;
11         ; Say this line is controlled by quad serial line unit
12         ;
13 005140  012761  0000000 0000000  MOV      #CDX$QP, LCDTYP(R1)
14         ;
15         ; See if quad line unit is installed
16         ;
17 005146  005737  000066'  TST      QPCSR          ; Is quad line unit installed?
18 005152  001454  BEQ      9$            ; Br if not
19         ;
20         ; Remember the TSX-Plus line index number associated with this unit line
21         ;
22 005154  016105  0000000  MOV      LMXLN(R1), R5   ; Get # of line within quad unit
23 005160  110165  000104'  MOVB    R1, QPLX(R5)    ; Store TSX-Plus line index #
24 005164  016100  0000000  MOV      LMXNUM(R1), R0  ; Get mux index number
25 005170  013760  000066' 0000000  MOV      QPCSR, MXCSR(R0) ; Remember CSR address for unit
26 005176  013760  000070' 0000000  MOV      QPVEC, MXVEC(R0) ; Remember vector address of unit
27         ;
28         ; See if this line should have modem control
29         ;
30 005204  105737  000110'  TSTB    QPMDM          ; Are we configured for modem control?
31 005210  001403  BEQ      3$            ; Br if not -- No lines have modem control
32 005212  032705  000001  BIT      #1, R5         ; Is this line 0 or 2?
33 005216  001403  BEQ      4$            ; Br if yes -- Has modem control
34 005220  042761  0000000 0000000 3$: BIC     ##PHONE, ILSW2(R1); Say line does not have modem control
35         ;
36         ; Convert line number into quad unit address
37         ;
38 005226  004737  006236'  4$: CALL  QPCVLA        ; Convert line # to address
39         ;
40         ; At this point,
41         ; R5 = Base address of registers for quad unit and channel.
42         ;
43         ; Initialize mode registers 1 and 2
44         ;
45 005232  112765  000020  000104  MOVB    #QP$RPR, QPRCRA(R5); Select mode register 1
46 005240  112765  000023  000100  MOVB    #023, QPRMRA(R5); Initialize mode register 1
47 005246  112765  000007  000100  MOVB    #007, QPRMRA(R5); Initialize mode register 2
48         ;
49         ; Initialize clock select register
50         ;
51 005254  112765  000273  000102  MOVB    #273, QPRCSA(R5); Initialize to 9600 baud
52         ;
53         ; Initialize receiver and transmitter
54         ;
55 005262  112765  000052  000104  MOVB    #QP$RRX!QP$DRX!QP$DTX, QPRCRA(R5); Reset receiver + disable
56 005270  112765  000100  000104  MOVB    #QP$RES, QPRCRA(R5); Reset error status
57 005276  112765  000065  000104  MOVB    #QP$RTX!QP$ERX!QP$ETX, QPRCRA(R5); Reset transmitter + enable

```

```
58 ;  
59 ; Finished  
60 ;  
61 005304 012605 9#: MOV (SP)+,R5  
62 005306 000207 RETURN
```



QPSTRT -- Start output

```

1          .SBTTL  QPSTRT -- Start output
2          ;-----
3          ; QPSTRT is called to initiate output to a line connected to the
4          ; quad line unit.
5          ;
6          ; Inputs:
7          ; R1 = Line index number.
8          ;
9 005310 010546 QPSTRT: MOV     R5,-(SP)
10         ;
11         ; Convert line number into quad unit address
12         ;
13 005312 004737 006236'          CALL     QPCVLA          ;Convert line # to address
14         ;
15         ; At this point,
16         ; R5 = Base address of registers for quad unit and channel.
17         ;
18         ; See if the transmitter is already busy
19         ;
20 005316          DISABL          ;;;Disable interrupts
21 005324 032761 0000000 0000000  BIT     ##XCHAR,LSW3(R1);;;Is the transmitter busy now?
22 005332 001003          BNE     1$          ;;;Br if yes
23         ;
24         ; Transmitter is idle.
25         ; Enable the transmitter which will cause an interrupt.
26         ;
27 005334 112765 000004 000104          MOVB   #QP$ETX,QPRCRA(R5);;;Enable transmitter (will cause interrupt)
28         ;
29         ; Finished
30         ;
31 005342          1$:  ENABL          ;Enable interrupts
32 005350 012605          MOV     (SP)+,R5
33 005352 000207          RETURN

```

```

1          .SBTTL  QPCINT -- Interrupt from quad line unit
2          ;-----
3          ; QPCINT is the interrupt entry point for all interrupts from the
4          ; quad line unit.
5          ;
6 005354   QPCINT:
7          ;
8          ; Save registers R0, R1, R4, and R5.
9          ; TTRSAV is the standard interrupt entry routine called when processing
10         ; received character interrupts. We always call it in the case of an
11         ; interrupt from a quad unit line since we don't know whether the interrupt
12         ; is from a transmitter or receiver.
13         ;
14 005354   004577   0000000   JSR      R5,@TTRSAV      ;Save registers
15 005360   010246   MOV      R2,-(SP)
16 005362   010346   MOV      R3,-(SP)
17         ;
18         ; Disable further interrupts from the quad line unit
19         ;
20 005364   013703   000066'   MOV      QPCSR,R3      ;Get address of base of registers for DUART 0
21 005370   142763   000100   000004   BICB    #QP#IEN,QPRMSR(R3) ;Clear interrupt-enable bit
22 005376   010301   MOV      R3,R1        ;Get addr of base of registers for DUART
23         ;
24         ; Begin loop to determine which of the four lines needs service
25         ;
26 005400   005002   CLR      R2          ;Start with line # 0 (DUART 0, channel A)
27         ;
28         ; Get TSX-Plus line index number of this line
29         ;
30 005402   116204   000104'   1$:     MOVB   QPLX(R2),R4      ;Get TSX-Plus line index number
31 005406   001461   BEQ     3$           ;Br if no line connected
32         ;
33         ; See if an output interrupt occurred
34         ;
35 005410   136263   005622'   000112   BITB    QPFTXR(R2),QPRISR(R3); Is transmitter ready for another char?
36 005416   001422   BEQ     2$           ;Br if not
37         ;
38         ; The transmitter is ready for another character.
39         ; See if there is another character to transmit.
40         ;
41 005420   004774   0000000   CALL    @LOUTIR(R4)   ;Get next character to transmit
42 005424   103411   BCS     8$           ;Br if no more chars to transmit
43         ;
44         ; Transmit the character contained in R0
45         ;
46 005426   052764   0000000   0000000   BIS     ##XCHAR,LSW3(R4); Set transmitter-busy flag
47 005434   042764   0000000   0000000   BIC     ##DITIM,LSW5(R4); Start timer to catch lost interrupts
48 005442   110061   000106   MOVB    R0,QPRTRA(R1) ; Start transmitting the character
49 005446   000406   BR      2$
50         ;
51         ; There are no more characters to transmit.
52         ; Disable the transmitter
53         ;
54 005450   042764   0000000   0000000   8$:     BIC     ##XCHAR,LSW3(R4); Say transmission finished
55 005456   112761   000010   000104   MOVB    #QP#DTX,QPRCRA(R1); Disable the transmitter
56         ;
57         ; See if this line has received a character

```

```

58 ;
59 005464 136263 005626' 000112 2#: BITB QPFRXR(R2),QPRISR(R3);Is a received character ready?
60 005472 001427 ; BEQ 3# ;Br if not
61 ;
62 ; Get received character
63 ;
64 005474 116100 000102 ; MOVB QPRSRA(R1),R0 ;Get status flags for character
65 005500 116105 000106 ; MOVB QPRRRA(R1),R5 ;Get the received character
66 005504 042705 177400 ; BIC #^C<377>,R5 ;Mask character to 8 bits
67 ;
68 ; Set status flags in R5 to simulate status flags returned by DL11
69 ;
70 005510 032700 000040 ; BIT #QP$RPE,R0 ;Did we have a parity error?
71 005514 001402 ; BEQ 4# ;Br if not
72 005516 052705 000000C ; BIS #RBERR!RCVPAR,R5;Set parity-error flags
73 005522 032700 000300 4#: BIT #QP$RFE!QP$BRK,R0 ;Framing error or break?
74 005526 001402 ; BEQ 5# ;Br if not
75 005530 052705 000000C ; BIS #RBERR!FRMERR,R5;Set framing-error flags
76 005534 032700 000020 5#: BIT #QP$RCE,R0 ;Did an overrun occur?
77 005540 001402 ; BEQ 6# ;Br if not
78 005542 052705 000000C ; BIS #RBERR!OVRRUN,R5;Set overrun flags
79 ;
80 ; Call TTINPT to process the received character
81 ;
82 005546 004777 0000000 6#: CALL @TTINPT ;Enter character input routine
83 ;
84 ; Check for other lines connect to the quad unit that may need service
85 ;
86 005552 005202 3#: INC R2 ;Advance line # within quad unit (0-3)
87 005554 020227 000003 ; CMP R2,#3 ;Have we serviced all lines?
88 005560 101010 ; BHI 7# ;Br if yes
89 005562 062701 000020 ; ADD #QPRCOF,R1 ;Point to registers for next channel
90 005566 020227 000002 ; CMP R2,#2 ;Time to advance register base to DUART 1?
91 005572 001303 ; BNE 1# ;Br if not
92 005574 062703 000040 ; ADD #QPRUOF,R3 ;Point to registers for DUART 1
93 005600 000700 ; BR 1# ;Go service next line
94 ;
95 ; We have serviced all four lines.
96 ; Reenable interrupt for quad line unit.
97 ;
98 005602 013703 000066' 7#: MOV QPCSR,R3 ;Get address of base of registers for DUART 0
99 005606 152763 000100 000004 ; BISB #QP$IEN,QPRMSR(R3) ;Set interrupt-enable bit
100 ;
101 ; Finished
102 ;
103 005614 012603 ; MOV (SP)+,R3
104 005616 012602 ; MOV (SP)+,R2
105 005620 000207 ; RETURN ;Return through TTRSAV which restores regs
106 ;
107 ; Vector containing the proper bit mask to test for transmitter-ready
108 ; in the Interrupt Status Register for each line.
109 ;
110 005622 001 020 001 QPFTXR: .BYTE 1,20,1,20
111 005625 020
112 ;
113 ; 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
```

QPSSPD -- Set speed for quad line port

```

1          .SBTTL  QPSSPD -- Set speed for quad line port
2          ;-----
3          ; QPSSPD is called to set the transmit/receive speed for a line
4          ; connected to the quad serial line unit.
5          ;
6          ; Inputs:
7          ; R0 = Speed code
8          ; R1 = Physical line index number.
9          ;
10         005632  010346  QPSSPD:  MOV     R3, -(SP)
11         005634  010546          MOV     R5, -(SP)
12         ;
13         ; Save info about baud rate for this line
14         ;
15         005636  110061  0000010  MOVB   R0, LMXPRM+1(R1) ; Save info in line table
16         ;
17         ; Convert line number into quad unit address
18         ;
19         005642  004737  006236'  CALL   QPCVLA          ; Convert line # to address
20         ;
21         ; At this point,
22         ; R5 = Base address of registers for quad unit and channel.
23         ;
24         ; Set transmit/receive speed
25         ;
26         005646  010003          MOV     R0, R3          ; Get control flags
27         005650  042703  000000C  BIC    #^C<LP$SPD>, R3 ; Clear all but speed value
28         005654  116303  005776'  MOVB   QPSPD(R3), R3   ; Get speed code for quad unit
29         005660  010346          MOV     R3, -(SP)
30         005662  072327  0000004  ASH    #4, R3          ; Position speed code for receive
31         005666  052603          BIS    (SP)+, R3       ; Or in transmit speed
32         005670  110365  000102  MOVB   R3, QPRCSA(R5)  ; Set speed for line
33         ;
34         ; Set character length and parity
35         ;
36         005674  032700  0000000  BIT    #LP$7BT, R0     ; 7 bit characters wanted?
37         005700  001403          BEQ    1$,             ; Br if not
38         005702  012703  0000002  MOV    #QP$7BR, R3    ; Get 7 bit char flags
39         005706  000402          BR     2$,             ;
40         005710  012703  0000003  1$:   MOV    #QP$8BR, R3 ; Get 8 bit char flags
41         005714  032700  0000000  2$:   BIT    #LP$PAR, R0  ; Is parity wanted?
42         005720  001003          BNE    3$,             ; Br if yes
43         005722  052703  0000020  BIS    #QP$NPR, R3    ; Set no-parity flag
44         005726  000405          BR     4$,             ;
45         005730  032700  0000000  3$:   BIT    #LP$ODD, R0  ; Odd parity wanted?
46         005734  001402          BEQ    4$,             ; Br if not
47         005736  052703  0000004  BIS    #QP$ODD, R3    ; Select odd parity
48         005742          4$:   DISABL          ; ; ** Disable interrupts **
49         005750  112765  0000020  000104  MOVB   #QP$RPR, QPRCRA(R5) ; ; Select mode register 1
50         005756  110365  000100  MOVB   R3, QPRMRA(R5)  ; ; Set char length and parity
51         005762          ENABL          ; ; ** Enable interrupts **
52         ;
53         ; Finished
54         ;
55         005770  012605          MOV    (SP)+, R5
56         005772  012603          MOV    (SP)+, R3
57         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
```

QPSBRK -- Control break transmission to quad line unit

```

1          .SBTTL  QPSBRK -- Control break transmission to quad line unit
2          ;-----
3          ; QPSBRK is called to start or stop transmitting a break character to
4          ; a line connected to the quad serial line unit.
5          ;
6          ; Inputs
7          ; R0 = Break control flag (MS$BRK)
8          ; R1 = Physical line index number.
9          ;
10         006016  010546  QPSBRK: MOV      R5,-(SP)
11         ;
12         ; Convert line number to address of control register
13         ;
14         006020  004737  006236'  CALL     QPCVLA      ;Convert line # to address
15         ;
16         ; At this point,
17         ; R5 = Base address of registers for quad unit and channel.
18         ;
19         ; See if we are to start or stop sending a break
20         ;
21         006024  032700  0000000  BIT      #MS$BRK,R0    ;Start or stop sending a break?
22         006030  001404          BEQ      1$                ;Br if stop
23         ;
24         ; Start sending a break
25         ;
26         006032  112765  000140  000104  MOVB    #QP$SBT,QPRCRA(R5) ;Start break transmission
27         006040  000403          BR      9$
28         ;
29         ; Stop sending a break
30         ;
31         006042  112765  000160  000104  1$:    MOVB    #QP$EBT,QPRCRA(R5) ;Stop break transmission
32         ;
33         ; Finished
34         ;
35         006050  012605  9$:    MOV     (SP)+,R5
36         006052  000207          RETURN

```

QPGDSS --- Get data set status for line

```

1          .SBTTL  QPGDSS --- Get data set status for line
2          ;-----
3          ; QPGDSS is called to get the data set (modem) status for a line
4          ; connected to the quad serial line unit.
5          ;
6          ; Inputs:
7          ; R1 = Physical line index number.
8          ;
9          ; Outputs:
10         ; R0 = Generic data set status flags (MS#xxx)
11         ;
12 006054 010346 QPGDSS: MOV     R3,-(SP)
13 006056 010546         MOV     R5,-(SP)
14         ;
15         ; Convert line number to address of control register
16         ;
17 006060 004737 006260'         CALL    QPCVLX         ;Convert line # to address
18         ;
19         ; At this point,
20         ; R5 = Base address of registers for quad unit.
21         ; R3 = Table index whose value depends on whether we are accessing
22         ; lines 0 or 1 (DUART 0) or lines 2 and 3 (DUART 1)
23         ; and the configuration 4/0 or 2/2.
24         ;
25         ;
26         ;
27         ;
28         ;
29         ;
30         ;
31         ;
32         ; See if the phone is ringing
33         ;
34 006064 005000         CLR     R0         ;Develop flags in R0
35 006066 136365 006144' 000132 BITB   QPFRI(R3),QPRIP(R5) ;Is the RI flag set for line?
36 006074 001402         BEQ     1$         ;Br if not
37 006076 052700 000000G         BIS     #MS#RNG,R0         ;Set ring status flag for return
38         ;
39         ; See if carrier is detected
40         ;
41 006102 136365 006140' 000132 1$: BITB   QPFCD(R3),QPRIP(R5) ;Is carrier detected?
42 006110 001402         BEQ     2$         ;Br if not
43 006112 052700 000000G         BIS     #MS#CAR,R0         ;Set carrier-detected flag for return
44         ;
45         ; See if Data Terminal Ready is asserted
46         ;
47 006116 136363 006150' 006154' 2$: BITB   QPFDTR(R3),QPOP(R3) ;Is DTR asserted?
48 006124 001402         BEQ     3$         ;Br if not
49 006126 052700 000000G         BIS     #MS#DTR,R0         ;Set DTR return flag
50         ;
51         ; Finished
52         ;
53 006132 012605 3$:     MOV     (SP)+,R5
54 006134 012603         MOV     (SP)+,R3
55 006136 000207         RETURN
56         ;
57         ; Flag values to check Carrier Detect flag based on channel and config

```



```
58 ;  
59 006140 004 010 004 QPFCD: .BYTE 4,10,4,0  
   006143 000  
60 ;  
61 ; Flag values to check for line ringing  
62 ;  
63 006144 000 000 010 QPFRI: .BYTE 0,0,10,0  
   006147 000  
64 ;  
65 ; Flag values to check for (or assert) Data Terminal Ready  
66 ;  
67 006150 004 010 004 QPFDTR: .BYTE 4,10,4,0  
   006153 000  
68 ;  
69 ; Shadow cells to hold value of output port settings  
70 ;  
71 006154 000 000 000 QPOP: .BYTE 0,0,0,0  
   006157 000
```

QPSDSS -- Set data set status

```

1          .SBTTL  QPSDSS -- Set data set status
2          ;-----
3          ; QPSDSS is called to control the data set (modem) status for a line
4          ; connected to the quad line unit.
5          ;
6          ; Inputs:
7          ; R1 = Physical line index number.
8          ; R0 = Data set control flags (MS#xxx)
9          ;
10         006160 010346 QPSDSS: MOV     R3,-(SP)
11         006162 010546         MOV     R5,-(SP)
12         ;
13         ; Convert line number to address of control register
14         ;
15         006164 004737 006260'         CALL    QPCVLX         ;Convert line # to address
16         ;
17         ; At this point,
18         ; R5 = Base address of registers for quad unit.
19         ; R3 = Table index whose value depends on whether we are accessing
20         ; lines 0 or 1 (DUART 0) or lines 2 and 3 (DUART 1)
21         ; and the configuration 4/0 or 2/2.
22         ;
23         ;
24         ;
25         ;
26         ;
27         ;
28         ;
29         ;
30         ; See if we should raise or drop Data Terminal Ready
31         ;
32         006170 032700 0000000         BIT     #MS#DTR,R0         ;Raise or drop DTR?
33         006174 001407         BEQ     1$         ;Br to drop DTR
34         ;
35         ; Raise DTR
36         ;
37         006176 116365 006150' 000134         MOVB   QPFDTR(R3),QPRSDP(R5);Raise bit in output port
38         006204 156363 006150' 006154'         BISB   QPFDTR(R3),QPOP(R3);Set bit in shadow cell
39         006212 000404         BR     9$
40         ;
41         ; Drop DTR
42         ;
43         006214 116365 006150' 000136 1$:     MOVB   QPFDTR(R3),QPRSDP(R5);Drop bit in output port
44         006222 146363 006150' 006154'         BICB   QPFDTR(R3),QPOP(R3);Clear bit in shadow cell
45         ;
46         ; Finished
47         ;
48         006230 012605 9$:     MOV     (SP)+,R5
49         006232 012603         MOV     (SP)+,R3
50         006234 000207         RETURN

```

Line	Config	Value
0/2	4/0	0
1/3	4/0	1
0/2	2/2	2
1/3	2/2	3

QPCVLA -- Convert line index into register addresses

```

1          .SBTTL  QPCVLA -- Convert line index into register addresses
2          ;-----
3          ; QPCVLA is called to convert a TSX-Plus line index number of a line
4          ; connected to the quad serial line unit into an address that corresponds
5          ; to the base of the registers for the corresponding DUART and channel.
6          ;
7          ; Inputs:
8          ;   R1 = TSX-Plus line index number
9          ;
10         ; Outputs:
11         ;   R5 = Address of base of registers for DUART and channel
12         ;   (R0 is preserved)
13         ;
14 006236  QPCVLA:
15         ;
16         ; Determine which quad line this line is connected to
17         ;
18 006236  016105  0000000  MOV     LMXLN(R1),R5    ;Get line # within quad line unit (0-3)
19         ;
20         ; Compute address of registers for this line
21         ;
22 006242  116505  006254'  2#:  MOVB   QPFADR(R5),R5    ;Get offset based on DUART and channel
23 006246  063705  000066'  ADD    QPCSR,R5        ;Add base CSR address for module
24         ;
25         ; Finished
26         ;
27 006252  000207  RETURN
28         ;
29         ; Table of register address offsets based on line number
30         ;
31 006254      000      020      040  QPFADR: .BYTE  0,20,40,60
32 006257      060
          .EVEN

```

```

1          .SBTTL  QPCVLX -- Get address and configuration info
2          ;-----
3          ; QPCVLX is called to convert a TSX-Plus line number for a line connected
4          ; to the quad serial line unit into two items of information:
5          ; (1) the address of the base of the registers for the DUART to which
6          ; the line is connected; (2) an index value in the range 0 to 3 which
7          ; indicates which channel of the DUART the line is connected to and
8          ; which configuration (2/2 or 4/0) the serial line unit is in.
9          ;
10         ; Inputs:
11         ; R1 = TSX-Plus line index number.
12         ;
13         ; Outputs:
14         ; R5 = Base address of registers for DUART.
15         ; R3 = Channel and configuration index value as follows:
16         ;
17         ;      Line  Config  Value
18         ;      ----  -
19         ;      0/2    4/0     0
20         ;      1/3    4/0     1
21         ;      0/2    2/2     2
22         ;      1/3    2/2     3
23         ;
24         ; (R0 is preserved)
25         ;
26 006260  QPCVLX:
27         ;
28         ; Determine which quad line this line is connected to
29         ;
30 006260  016103  0000000  MOV     LMXLN(R1),R3    ;Get line # within quad line unit (0-3)
31         ;
32         ; Get the DUART base register address
33         ;
34 006264  116305  006314'  2$:    MOVB   QPFDAD(R3),R5    ;Get offset based on DUART 0 or 1
35 006270  063705  000066'  ADD    QPCSR,R5          ;Add CSR address of base of module
36         ;
37         ; Get index value based on line # and configuration
38         ;
39 006274  042703  177776  BIC    #^C<1>,R3        ;Get 0 for lines 0&2, 1 for lines 1&3
40 006300  105737  000110'  TSTB   QPMDM            ;Is this a 2/2 configuration?
41 006304  001402  BEQ     9$              ;Br if not
42 006306  062703  000002  ADD    #2,R3            ;Set index based on config
43         ;
44         ; Finished
45         ;
46 006312  000207  9$:    RETURN
47         ;
48         ; Table of address offsets to base registers for DUARTs based on line #
49         ;
50 006314  000 000 040  QPFDAD: .BYTE 0,0,40,40
51         ;
52         ;      .EVEN
53         ;      .ENDC ;NE,QPASM ;End of quad line unit code
54         ;
55         ; Size of TSXPRO
56         ;
56 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



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\$RRO	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\$WRO	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#				







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*					
PPGDSS	29-21	33-13#						



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#								
QPFDDAD	57-34	57-50#								
QPFDR	54-47	54-67#	55-37	55-38	55-43	55-44				
QPFRI	54-35	54-63#								
QPFRRXR	51-59	51-115#								
QPFTRX	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						

R5OPI	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					
SR0MMR	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#DEF	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						

DISABL	2-5# 52-48	11-15	21-19	21-27	23-23	35-29	40-12	42-9	43-9	46-27	47-51	50-20
ENABL	2-11#	11-38	21-45	23-25	35-38	40-34	42-35	43-35	46-30	47-56	50-31	52-51