

TSTIO -- TSX-Plus terminal in MACRO V05.05 Tuesday 17-Jan-89 14:05

Table of contents

4-	1	DLXOFF -- Transmit XOFF character
5-	1	DLXON -- Transmit XON character
6-	1	DLINT -- DL terminal output interrupt entry point
6-	45	DLSTRT -- Start output to a DL11 line
7-	1	DZDINT -- DZ terminal output interrupt entry point
8-	1	DZSTRT -- Start output to a DZ11 line
9-	1	NEDCHR -- Get next character for output
9-	14	REQCDO -- Request clock-driven output processing
10-	1	SETSPD -- Set transmit/receive speed for a line
11-	1	GETDSS/SETDSS -- Get/set data set status

```

1          .TITLE  TSTIO  -- TSX-Plus terminal interrupt driver
2 000000    .PSECT  TSTIO
3          .ENABL  LC
4          .ENABL  AMA
5          ;
6          ; TSTIO contain the interrupt processing code for sending and receiving
7          ; characters to DL11, DZ11, and DH11 lines.
8          ;
9          .DSABL  GBL
10         ;
11         ; Global definitions
12         ;
13         .GLOBL  TSTIO, DLINT, DZINT, CDSTRT, CDSTOP, NEDSOT
14         .GLOBL  CDIRTN, CDIFLG, CDORTN, CDOFLG, NEDCDI, NEDCDO
15         .GLOBL  ININT, CDSSPD, LINRTS, TTRSAV, TTINPT
16         .GLOBL  CDCLOK, CDQDSS, CDSOSS, CDSBRK, SETSPD
17         .GLOBL  CDSTR2, CDSXON, CDSXOF, DLSTRT, DZSTRT
18         .GLOBL  SILFET, TRNSTR, NEDCLO, TIOVEC, GETDSS, SETDSS
19         ;
20         ; Global references
21         ;
22         .GLOBL  $OITIM, $XCHAR
23         .GLOBL  KPAR6, LMXLN
24         .GLOBL  DHSSPD, VHSSPD, DHSBRK, VHSBRK
25         .GLOBL  LMXNUM
26         .GLOBL  LSW3, LSW5
27         .GLOBL  MXCSR, MXLNT
28         .GLOBL  MXTBUF, MXTCR, NEDCHR, PR7, PSW
29         .GLOBL  RLINE
30         .GLOBL  TBR, TRINT, TRRDY, TSR
31         .GLOBL  DHSTRT, DHSTOP, DLSBRK, DZSBRK, DZSSPD
32         .GLOBL  VHSTRT, DLCLOK, DLSSPD, $DHCDO
33         .GLOBL  DHXON, DHXOFF, VHXON, VHXOFF
34         .GLOBL  DZCLOK, DLQDSS, DZQDSS, DHQDSS, VHQDSS
35         .GLOBL  DLSOSS, DZSDSS, DHSOSS, VHSOSS
36         .GLOBL  LOUTIR, KPAR5
37         .GLOBL  LSW10
38         .GLOBL  $HISTP
39         .GLOBL  $SXOFF, LCDTYP
40         .GLOBL  $SXON
41         .GLOBL  VHSTOP, INTPRI
42         .GLOBL  DHCLOK, VHCLOK

```

```

1      ; -----
2      ; Macro definitions:
3      ;
4      ; Disable interrupts
5      ;
6      ;     .MACRO  DISABL          ;Disable interrupts
7      ;     BIS     #340, @PSW
8      ;     .ENDM  DISABL
9      ;
10     ; Enable interrupts
11     ;
12     ;     .MACRO  ENABL          ;Enable interrupts
13     ;     BIC     INTPRI, @PSW
14     ;     .ENDM  ENABL
15     ;
16     ; Call a routine in a system overlay
17     ;
18     ;     .MACRO  OCALL  ENTADD
19     ;     .IF      B, ENTADD
20     ;     .ERROR   ;OCALL without entry address
21     ;     .ENDC
22     ;     CALL    OVRHC
23     ;     .WORD   ENTADD
24     ;     .ENDM  OCALL
25     ;
26     ; The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
27     ; terminal character buffer area. The previous contents of par6 map
28     ; register are pushed on the stack and may be restored by using the
29     ; UNMAP or UNMAPX macros.
30     ; R1 must contain the line index number of the line whose buffers
31     ; are being accessed.
32     ; The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
33     ; macros is that the X-versions are more efficient but may only be
34     ; used from within interrupt service routines where we are guaranteed
35     ; to be running on the system stack.
36     ; The TTMAP and UNMAP versions of the macros must only be
37     ; used in sections of code where the interrupts are disabled.
38     ;
39     ;     .MACRO  TTMAPX
40     ;     MOV     LTPAR(R1), @KPAR6
41     ;     .ENDM  TTMAPX
42     ;
43     ;
44     ;
45     ;     .MACRO  UNMAPX
46     ;     .ENDM  UNMAPX
47     ;
48     ;     .MACRO  TTMAP
49     ;     MOV     @KPAR6, MAPHLD
50     ;     MOV     LTPAR(R1), @KPAR6
51     ;     .ENDM  TTMAP
52     ;
53     ;     .MACRO  UNMAP
54     ;     MOV     MAPHLD, @KPAR6
55     ;     .ENDM  UNMAP

```

```

1      ; -----
2      ; Address vectors used to transfer control to device specific routines
3      ; based on the device type code (CDX$xxx).
4      ; Note: the entries in these vectors must be arranged in the same order
5      ; as the CDX$xxx values defined in TSGEN.
6      ;
7      ; Table of routines to call to initiate transmission to a line
8      ;
9      ; Inputs:
10     ; R1 = Physical line index number.
11     ;
12     TSTIO:
13     CDSTR1: .WORD    DLSTRT          ; DL11
14             .WORD    DZSTRT          ; DZ11
15             .WORD    REQCD0          ; DH11
16             .WORD    REQCD0          ; DHV11
17             .WORD    DORTS           ; Professional console
18             .WORD    DORTS           ; Professional communications port
19             .WORD    DORTS           ; Professional printer port
20             .WORD    DORTS           ; Professional quad serial line unit
21     ;
22     ; Table of routines used to perform actual hardware startup
23     ; of output transmission.
24     ;
25     CDSTR2: .WORD    DORTS           ; DL11
26             .WORD    DORTS           ; DZ11
27             .WORD    DHSTRT          ; DH11
28             .WORD    VHSTRT          ; DHV11
29             .WORD    DORTS           ; Professional console
30             .WORD    DORTS           ; Professional communications port
31             .WORD    DORTS           ; Professional printer port
32             .WORD    DORTS           ; Professional quad serial line unit
33     ;
34     ; Table of routines to call to stop transmission to a line.
35     ; (These routines are used if a ctrl-S is received)
36     ;
37     ; Inputs:
38     ; R1 = Physical line index number.
39     ;
40     CDSTOP: .WORD    DORTS           ; DL11
41             .WORD    DORTS           ; DZ11
42             .WORD    DHSTOP          ; DH11
43             .WORD    VHSTOP          ; DHV11
44             .WORD    DORTS           ; Professional console
45             .WORD    DORTS           ; Professional communications port
46             .WORD    DORTS           ; Professional printer port
47             .WORD    DORTS           ; Professional quad serial line unit
48     ;
49     ; Table of routines called to stuff an XOFF (ctrl-S) into the output
50     ; stream for a line.
51     ;
52     ; Inputs:
53     ; R1 = Line index number
54     ;
55     CDSXOF: .WORD    DLXOFF          ; DL11
56             .WORD    DLXOFF          ; DZ11
57             .WORD    DHXOFF          ; DH11

```

```

58 000066 0000000      .WORD  VHXOFF      ; DHV11
59 000070 000314'      .WORD  DLXOFF      ; Professional console
60 000072 000240'      .WORD  DORTS       ; Professional communications port
61 000074 000314'      .WORD  DLXOFF      ; Professional printer port
62 000076 000314'      .WORD  DLXOFF      ; Professional quad serial line unit
63
64      ; Table of routine called to stuff an XON (ctrl-Q) into the output
65      ; stream for a line.
66
67      ; Inputs:
68      ; R1 = Line index number.
69
70 000100 000342'      CDSXON: .WORD  DLXON      ; DL11
71 000102 000342'      .WORD  DLXON      ; DZ11
72 000104 0000000      .WORD  DHXON      ; DH11
73 000106 0000000      .WORD  VHXON      ; DHV11
74 000110 000342'      .WORD  DLXON      ; Professional console
75 000112 000240'      .WORD  DORTS       ; Professional communications port
76 000114 000342'      .WORD  DLXON      ; Professional printer port
77 000116 000342'      .WORD  DLXON      ; Professional quad serial line unit
78
79      ; Table of routines to be called on a 0.5 second basis to do line checking.
80
81      ; Inputs:
82      ; R1 = Physical line index number.
83
84 000120 0000000      CDCCLOCK: .WORD  DLCLOCK      ; DL11
85 000122 0000000      .WORD  DZCLOCK      ; DZ11
86 000124 0000000      .WORD  DHCLOCK      ; DH11
87 000126 0000000      .WORD  VHCLOCK      ; DHV11
88 000130 000240'      .WORD  DORTS       ; Professional console
89 000132 000240'      .WORD  DORTS       ; Professional communications port
90 000134 000240'      .WORD  DORTS       ; Professional printer port
91 000136 000240'      .WORD  DORTS       ; Professional quad serial line unit
92
93      ; Table of routines to be called to get the data set status
94
95      ; Inputs:
96      ; R1 = Physical line index number.
97
98      ; Outputs:
99      ; R0 = Generic data set status flags (MS$xxx)
100
101 000140 0000000      CDGDSS: .WORD  DLGDSS      ; DL11
102 000142 0000000      .WORD  DZGDSS      ; DZ11
103 000144 0000000      .WORD  DHGDSS      ; DH11
104 000146 0000000      .WORD  VHGDSS      ; DHV11
105 000150 000240'      .WORD  DORTS       ; Professional console
106 000152 000240'      .WORD  DORTS       ; Professional communications port
107 000154 000240'      .WORD  DORTS       ; Professional printer port
108 000156 000240'      .WORD  DORTS       ; Professional quad serial line unit
109
110      ; Table of routines to be called to set data set status
111
112      ; Inputs:
113      ; R0 = Data set control flags (MS$xxx)
114      ; R1 = Physical line index number.

```

```

115
116 000160 000000G      ;
117 000162 000000G      ; CDSDDS: .WORD  DLSDSS          ; DL11
118 000164 000000G      ; .WORD  DZSDSS          ; DZ11
119 000166 000000G      ; .WORD  DHSDDS          ; DH11
120 000170 000240'      ; .WORD  VHSDDS          ; DHV11
121 000172 000240'      ; .WORD  DORTS          ; Professional console
122 000174 000240'      ; .WORD  DORTS          ; Professional communications port
123 000176 000240'      ; .WORD  DORTS          ; Professional printer port
124                      ; .WORD  DORTS          ; Professional quad serial line unit
125                      ;
126                      ; Table of routines to be called to control break transmission
127                      ;
128                      ; Inputs:
129                      ; RO = Break control flag (MS$BRK)
130                      ; R1 = Physical line index number.
131
132 000200 000000G      ;
133 000202 000000G      ; CDSBRK: .WORD  DLSBRK          ; DL11
134 000204 000000G      ; .WORD  DZSBRK          ; DZ11
135 000206 000000G      ; .WORD  DHSBRK          ; DH11
136 000210 000240'      ; .WORD  VHSBRK          ; DHV11
137 000212 000240'      ; .WORD  DORTS          ; Professional console
138 000214 000240'      ; .WORD  DORTS          ; Professional communications port
139 000216 000240'      ; .WORD  DORTS          ; Professional printer port
140                      ; .WORD  DORTS          ; Professional quad serial line unit
141                      ;
142                      ; Table of routines to be called to control transmission speed
143                      ;
144                      ; Inputs:
145                      ; RO = Line speed code.
146                      ; R1 = Line index number.
147
148 000220 000000G      ;
149 000222 000000G      ; CDSSPD: .WORD  DLSSPD          ; DL11
150 000224 000000G      ; .WORD  DZSSPD          ; DZ11
151 000226 000000G      ; .WORD  DHSSPD          ; DH11
152 000230 000240'      ; .WORD  VHSSPD          ; DHV11
153 000232 000240'      ; .WORD  DORTS          ; Professional console
154 000234 000240'      ; .WORD  DORTS          ; Professional communications port
155 000236 000240'      ; .WORD  DORTS          ; Professional printer port
156                      ; .WORD  DORTS          ; Professional quad serial line unit
157
158                      ;
159                      ; Dummy routine to do a simple return
160
161 000240 000207      ; DORTS: RETURN
162
163                      ;
164                      ; -----
165                      ; Pointer vector to routines in TSTIQX module that is relocated over TSINIT.
166
167 000242 000000      ; TIOVEC:
168 000244 000000      ; ININTP: .WORD  0
169 000246 000000      ; TTINPT: .WORD  0
170 000250 000000      ; SILFET: .WORD  0
171 000252 000000      ; TTRSAV: .WORD  0
172 000254 000000      ; TRNSTR: .WORD  0
173 000256 000000      ; NEDCP: .WORD  0
174 000258 000000      ; CDORTN: .WORD  0
175 000260 000000      ; CDIRTN: .WORD  0
176 000262 000000      ; SNDFRE: .WORD  0

```

```

172 000264 177777          .WORD  -1          ;Signal end of vector
173
174
175          ; -----
176          ; Data areas
177 000266 000000  MAPHLD: .WORD  0          ;TEMP CELL USED BY TMAP MACRO
178 000270 000000  NEDCDI: .WORD  0          ;Non-zero ==> Need clock driven input proc
179 000272 000000  NEDCDO: .WORD  0          ;Non-zero ==> Need clock driven output proc
180 000274 000000  NEDCLO: .WORD  0          ;Non-zero ==> Need CL output processing
181 000276 000000  NEDSOT: .WORD  0          ;Non-zero ==> Need output-low job scheduling
182
183          ; Byte data
184
185 000300      000  CDIFLG: .BYTE  0          ;Non-zero ==> Input char fork routine running
186 000301      000  CDOFLG: .BYTE  0          ;Non-zero ==> Output char fork routine running
187          ; Table used to select bit-within-word corresponding to a line
188 000302      001      002      004  MXLBIT: .BYTE  1,2,4,10,20,40,100,200
      000305      010      020      040
      000310      100      200
189
190          .EVEN
191
192          ; Ignore an interrupt
193 000312 000207  LINRTS: RETURN          ;Ignore interrupt

```


DLXOFF -- Transmit XOFF character

```

1          .SBTTL  DLXOFF -- Transmit XOFF character
2          ; -----
3          ; DLXOFF is called to transmit an XOFF (ctrl-S) character to stop
4          ; transmission to us.
5          ;
6          ; Inputs:
7          ; R1 = Line index number.
8          ;
9 000314    DLXOFF:
10         ;
11         ; Set flag saying we have stopped the sender
12         ;
13 000314    052761    000000G 000000G    BIS    ##HISTP,LSW10(R1);Set flag saying we have stopped sender
14         ;
15         ; Set flag to cause an XOFF character to be sent by the transmitter
16         ; interrupt routine.
17         ;
18 000322    052761    000000G 000000G    BIS    ##SXOFF,LSW10(R1);Set flag saying to send XOFF
19         ;
20         ; Start the transmitter
21         ;
22 000330    016100    000000G    MOV     LCDTYP(R1),R0    ;Get device type code index
23 000334    004770    000000'    CALL    @CDSTRT(R0)      ;Call routine to start the transmitter
24         ;
25         ; Finished
26         ;
27 000340    000207    RETURN

```


DLXON -- Transmit XON character

```

1          .SBTTL  DLXON  -- Transmit XON character
2          ;-----
3          ; DLXON is called to transmit an XON (ctrl-Q) character.
4          ;
5          ; Inputs:
6          ;   R1 = Line index number.
7          ;
8 000342    DLXON:
9          ;
10         ; Clear flag that says XOFF has been sent
11         ;
12 000342    042761    0000000 0000000      BIC    ##HISTP,LSW10(R1);Clear flag that says XOFF has been sent
13         ;
14         ; Set flag to cause an XON character to be sent by the transmitter
15         ; interrupt routine.
16         ;
17 000350    052761    0000000 0000000      BIS    ##SXON,LSW10(R1);Force transmission of XON
18         ;
19         ; Start the transmitter
20         ;
21 000356    016100    0000000      MOV     LCDTYP(R1),R0    ;Get device type index number
22 000362    004770    0000000      CALL    @CDSTRT(R0)      ;Call routine to start the transmitter
23         ;
24         ; Finished
25         ;
26 000366    000207      RETURN

```

DLINT -- DL terminal output interrupt entry point

```

1          .SBTTL  DLINT  -- DL terminal output interrupt entry point
2          ;-----
3          ; DLINT contains the DL device specific output interrupt service routine.
4          ; The interrupt is vectored directly to an instruction sequence of the
5          ; form
6          ;
7          ;     JSR      R4,@#DLINT
8          ;     .WORD   line_number
9          ;
10         ; Thus on entry to DLINT R4 has been saved on the stack and currently
11         ; points to a word containing the line index number.
12         ;
13         ;
14 000370 010046 DLINT:  MOV      RO,-(SP)          ; Save registers (R4 already on stack)
15         ;
16         ; Obtain the interrupting physical line number.
17         ;
18 000372 011404         MOV      (R4),R4          ; Get physical line index number
19         ;
20         ; Get next character to transmit
21         ;
22 000374 004774 000000G CALL     @LOUTIR(R4)        ; Get next character for line
23 000400 103417         BCS      DLCLR             ; Br if no character available
24         ;
25         ; Transmit the character contained in RO.
26         ;
27 000402 052764 000000G 000000G BIS      #$XCHAR,LSW3(R4)    ; Set transmitter busy flag
28 000410 042764 000000G 000000G BIC      #$OITIM,LSW5(R4)    ; Start timer to catch lost interrupts
29 000416 032774 000000G 000000G 1$:  BIT      #TRRDY,@TSR(R4)  ; Make sure transmitter is ready
30 000424 001774         BEQ      1$                ; Should never loop here
31 000426 110074 000000G         MOVB     RO,@TBR(R4)          ; Transmit the character
32         ;
33         ; Finished. Return from interrupt.
34         ;
35 000432 012600 DLRTN:  MOV      (SP)+,RO
36 000434 012604         MOV      (SP)+,R4          ; R4 was pushed by JSR instruction
37 000436 000002         RTI
38         ;
39         ; Disable transmit interrupts.
40         ;
41 000440 042764 000000G 000000G DLCLR:  BIC      #$XCHAR,LSW3(R4)    ; Set not busy transmit flag
42 000446 042774 000000G 000000G         BIC      #TRINT,@TSR(R4)    ; Disable transmit interrupt enable
43 000454 000766         BR        DLRTN             ; Return from interrupt
44         ;
45         .SBTTL  DLSTRT -- Start output to a DL11 line
46         ;-----
47         ; Start output to a DL11 line.
48         ; Enable transmit interrupts.
49         ; R1 = Physical line number
50         ;
51 000456 052771 000000G 000000G DLSTRT: BIS      #TRINT,@TSR(R1)    ; Enable transmit interrupt enable
52 000464 000207         RETURN
53

```

DZOINT -- DZ terminal output interrupt entry point

```

1          .SBTTL  DZOINT -- DZ terminal output interrupt entry point
2          ;-----
3          ; DZOINT contains the DZ device specific output interrupt service routine.
4          ; The interrupt is vectored directly to an instruction sequence of the
5          ; form
6          ;
7          ;       JSR      R5,@#DZOINT
8          ;       .WORD   mux_number
9          ;
10         ; Thus on entry to DZOINT R5 has been saved on the stack and currently
11         ; points to a word containing the mux index number.
12         ;
13         ;
14 000466 010046 DZOINT: MOV      R0,-(SP)          ; Save registers (R5 already on stack)
15 000470 010446      MOV      R4,-(SP)
16         ;
17         ; Obtain the interrupting physical line number.
18         ;
19 000472 011505      MOV      (R5),R5              ; Get mux index number
20 000474 017500 000000 MOV      @MXCSR(R5),R0        ; Get the DZ CSR status register
21 000500 100022      BPL      DZRTN                  ; Br if no line interrupted
22 000502 042700 000000C BIC      #^C<RLINE>,R0          ; Isolate the interrupting line
23 000506 000300      SWAB     R0                      ; Shift to right byte
24 000510 066500 000000G ADD      MXLNT(R5),R0          ; Point to correct MUX line table
25 000514 111004      MOVB     (R0),R4                ; Get the physical line number
26 000516 001413      BEQ      DZRTN                  ; Br if line not genned in system
27         ;
28         ; Obtain the next character for the interrupting line.
29         ;
30 000520 004774 000000G CALL     @LOUTIR(R4)          ; Get the next character for line
31 000524 103414      BCS      DZCLR                  ; Br if no character available
32         ;
33         ; Transmit the character contained in R0.
34         ;
35 000526 052764 000000G 000000G BIS      #$XCHAR,LSW3(R4)      ; Set transmitter busy flag
36 000534 042764 000000G 000000G BIC      #$OITIM,LSW5(R4)      ; Start timer to catch lost interrupts
37 000542 110075 000000G      MOVB     R0,@MXTBUF(R5)          ; Transmit the character
38         ;
39         ; Finished. Return from interrupt.
40         ;
41 000546 012604 DZRTN: MOV      (SP)+,R4
42 000550 012600      MOV      (SP)+,R0
43 000552 012605      MOV      (SP)+,R5              ; R5 was saved by JSR
44 000554 000002      RTI
45         ;
46         ; Disable transmit interrupts.
47         ;
48 000556 042764 000000G 000000G DZCLR: BIC      #$XCHAR,LSW3(R4) ; Set not busy transmit flag
49 000564 016400 000000G      MOV      LMXLN(R4),R0          ; Get number of line within MUX
50 000570 146075 000302' 000000G BICB     MXLBIT(R0),@MXTCR(R5) ; Disable transmitter interrupt
51 000576 000763      BR        DZRTN                  ; Return from interrupt

```

```

1
2
3
4
5
6
7 000600 010046
8 000602 010546
9 000604 016105 0000000
10 000610 016100 0000000
11 000614 156075 000302' 0000000
12 000622 012605
13 000624 012600
14 000626 000207

.SBTTL DZSTRT -- Start output to a DZ11 line
-----
; Start output to a DZ11 line.
; Enable transmit interrupts.
; R1 = Physical line number
;
DZSTRT: MOV R0, -(SP) ; Save registers
MOV R5, -(SP)
MOV LMXNUM(R1), R5 ; Get the MUX index number
MOV LMXLN(R1), R0 ; Get number of line within MUX
BISB MXLBIT(R0), @MXTCR(R5) ; Enable transmitter interrupt
MOV (SP)+, R5 ; Restore registers
MOV (SP)+, R0
RETURN

```

```

1          .SBTTL  NEDCHR -- Get next character for output
2          ;-----
3          ; This is a dummy resident routine which calls the relocated
4          ; NEDCHR routine in the TSTIOX module.
5          ;
6 000630 000177 177420 NEDCHR: JMP      @NEDCP          ;Call routine in TSTIOX
7          ;
8          ;-----
9          ; This is a dummy resident routine which calls the relocated
10         ; ININT routine in the TSTIOX module.
11         ;
12 000634 000177 177402 ININT:  JMP      @ININTP          ;Call routine in TSTIOX
13         ;
14         .SBTTL  REQCD0 -- Request clock-driven output processing
15         ;-----
16         ; REQCD0 is called to set a flag requesting clock-driven output
17         ; character processing.
18         ;
19         ; Inputs:
20         ; R1 = Physical line index number.
21         ;
22 000640 052761 0000000 0000000 REQCD0: BIS      ##DHCD0,LSW10(R1)    ;Request clock-driven processing
23 000646 005237 000272'      INC      NEDCDO          ;Request output processing
24 000652 000207      RETURN          ;Finished

```

SETSPD -- Set transmit/receive speed for a line

```

1          .SBTTL  SETSPD -- Set transmit/receive speed for a line
2          ;-----
3          ; SETSPD is called to set the transmit/receive speed for a line.
4          ;
5          ; Inputs:
6          ;   R0 = Speed code.
7          ;   R1 = Line index number of line being set.
8          ;
9 000654 010246 SETSPD: MOV      R2, -(SP)
10 000656 013746 000000G      MOV      @#KPAR5, -(SP) ; Save kpar 5 in case an overlay is calling us
11          ;
12          ; Call hardware dependent routine to set the speed
13          ;
14 000662 016102 000000G      MOV      LCDTYP(R1), R2 ; Get hardware device type code
15 000666 004772 000220'      CALL      @CDSSPD(R2) ; Call routine to set the speed
16          ;
17          ; Finished
18          ;
19 000672 012637 000000G      MOV      (SP)+, @#KPAR5
20 000676 012602      MOV      (SP)+, R2
21 000700 000207      RETURN

```

GETDSS/SETDSS -- Get/set data set status

```

1          .SBTTL  GETDSS/SETDSS -- Get/set data set status
2          ;-----
3          ; Inputs: R1 contains line index
4          ;           For set, options are in R0 (for @CDSDSS routine)
5          ; Outputs: For get, options are returned in R0 (by @CDGDSS routine)
6          ;
7 000702 005046  GETDSS: CLR      -(SP)          ; Say we are getting status
8 000704 000402          BR      DSSCOM          ; Br to common routine
9 000706 012746 000001 SETDSS: MOV      #1,-(SP)        ; Say we are setting data set status
10         ;
11 000712 010246  DSSCOM: MOV      R2,-(SP)
12 000714 013746 000000G      MOV      @#KPAR5,-(SP) ; Save KPAR 5 in case overlay is calling us
13         ;
14         ; Call hardware dependent routine to get current data set status
15         ;
16 000720 016102 000000G      MOV      LCDTYP(R1),R2 ; Get hardware device type code
17 000724 005766 000004      TST      4(SP)          ; Get or set data set status?
18 000730 001003          BNE      1$              ; Br if set
19 000732 004772 000140'      CALL     @CDGDSS(R2)    ; Get data set status (into R0)
20 000736 000402          BR      2$              ; Br to exit
21 000740 004772 000160' 1$:  CALL     @CDSDSS(R2)    ; Set data set status (from R0)
22         ;
23         ; Clean up and return
24         ;
25 000744 012637 000000G 2$:  MOV      (SP)+,@#KPAR5 ; Restore KPAR 5 mapping
26 000750 012602          MOV      (SP)+,R2
27 000752 005726          TST      (SP)+          ; Remove GET/SET flag from stack
28 000754 000207          RETURN
29         ;
30         .END

```

Errors detected: 0

*** Assembler statistics

```

Work file reads: 0
Work file writes: 0
Size of work file: 232 Words ( 1 Pages)
Size of core pool: 17920 Words ( 70 Pages)
Operating system: RT-11

```

```

Elapsed time: 00:00:08.52
DK: TSTIO,LP:TSTIO=DK:TSTIO/C/N:SYM

```


[illegible]

ININTP	3-163#	9-12				
INTPRI	1-41					
KPAR5	1-36	10-10	10-19*	11-12	11-25*	
KPAR6	1-23					
LCDTYP	1-39	4-22	5-21	10-14	11-16	
LINRTS	1-15	3-193#				
LMXLN	1-23	7-49	8-10			
LMXNUM	1-25	8-9				
LQUTIR	1-36	6-22	7-30			
LSW10	1-37	4-13*	4-18*	5-12*	5-17*	9-22*
LSW3	1-26	6-27*	6-41*	7-35*	7-48*	
LSW5	1-26	6-28*	7-36*			
MAPHLD	3-177#					
MXCSR	1-27	7-20				
MXLBIT	3-188#	7-50	8-11			
MXLNT	1-27	7-24				
MXTBUF	1-28	7-37*				
MXTCR	1-28	7-50*	8-11*			
NEDCDI	1-14	3-178#				
NEDCDO	1-14	3-179#	9-23*			
NEDCHR	1-28	9-6#				
NEDCLO	1-18	3-180#				
NEDCP	3-168#	9-6				
NEDSOT	1-13	3-181#				
PR7	1-28					
PSW	1-28					
REQCDO	3-15	3-16	9-22#			
RLINE	1-29	7-22				
SETDSS	1-18	11-9#				
SETSPD	1-16	10-9#				
SILFET	1-18	3-165#				
SNDFRE	3-171#					
TBR	1-30	6-31*				
TIOVEC	1-18	3-162#				
TRINT	1-30	6-42	6-51			
TRNSTR	1-18	3-167#				
TRRDY	1-30	6-29				
TSR	1-30	6-29	6-42*	6-51*		
TSTIO	1-13	3-12#				
TTINPT	1-15	3-164#				
TTRSAV	1-15	3-166#				
VHCLOK	1-42	3-87				
VHGDSS	1-34	3-104				
VHSBRK	1-24	3-134				
VHSDSS	1-35	3-119				
VHSSPD	1-24	3-149				
VHSTOP	1-41	3-43				
VHSTRT	1-32	3-28				
VHXOFF	1-33	3-58				
VHXON	1-33	3-73				

DISABL	2-6#
ENABL	2-12#
OCALL	2-18#
TTMAP	2-48#
TTMAPX	2-39#
UNMAP	2-53#
UNMAPX	2-45#