3.3.3 Ansteuerung eines grafischen LCD über RS232 vom PC

Für die grafische LCD TLX-711A wird mit einem 68HC11 eine RS232 Schnittstelle erzeugt. Die Schnittstelle ist auf 9600 Baud und 8 Datenbits ohne Parität eingestellt. Die Synchronisation mit dem PC erfolgt durch das DTR-Signal vom Zwerg11a. Das DTR-Signal ist nur aktiv, wenn sich das Steuerprogramm im 68HC11 in dem Unterprogramm zur Eingabe eines Zeichens von der RS232-Schnittstelle befindet. Die LCD TLX-711A wird vom Steuerprogramm so initialisiert, daß ein grafisches Display und ein alphanumerisches Display übereinander liegen und die Pixel aus beiden Anzeigen mit einer EXOR-Verknüpfung angezeigt werden. Dadurch ist eine Inversdarstellung von Texten möglich, wenn die Pixel des grafischen Displays alle gesetzt sind. Das Steuerprogramm im 68HC11 wertet folgende Steuerzeichen im gesendeten Textstring aus:

- CR ($0d) Es wird an den Anfang der nächsten Zeile gesprungen.
- LF ($0a) wird ignoriert.
- BS ($08) setzt die Schreibposition für das nächste Zeichen um 1 zurück.
- FF ($0c) löscht das alphanumerische Display.
- ESC ($1b) Beginn einer 3 Bytefole zur Ansteuerung der Pixel des grafischen Displays.

Die ESC-Sequenzen haben folgenden binären Aufbau:

ESC	Y	X
0001 1011	PCYY YYYY	XXXX XXXX

Nach dem Steuerbyte ESC folgen zwei Byte, in denen in P der Wert des Pixel festgelegt wird und in den Y und X Bits der beiden Folgebytes wird die Position des Pixels übergeben. Die Position Y=0 und X=0 befindet sich oben links auf der LCD TLX-711A. Wenn das Bit C=1 ist, sind die Angaben in den Y und X Bits unbedeutend. Ein gesetztes Bit C bewirkt ein setzen aller Pixel der LCD mit dem Wert in P. Damit ist ein löschen des grafischen Displays möglich. Der Wertebereich für Y ist 0 ... 63 und für X 0 ... 239. Die Liste des Steuerprogramm für den 68HC11 ist unten abgedruckt.

file: a_g_lcd.a

```
*               ******************************
*               * 8*40 LCD mit Font 6*8*
*               *            &                 *
*               *   4*240 dots grafisch    *
*               ******************************
```

file: c:\prak\zwerg11a\lib.a\hc11.h

```
0000b600  EEPROM      equ     $b600           eeprom area


00001000  REG         equ     $1000           register base
```

```
00001000 porta    equ REG+$00 port a data reg.
00001002 pioc     equ REG+$02 parallel i/o control reg.
00001003 portc    equ REG+$03 port c data reg.
00001004 portb    equ REG+$04 port b data reg.
00001005 portcl   equ REG+$05 port c latched data reg.
00001007 ddrc     equ REG+$07 port c data direction reg.
00001008 portd    equ REG+$08 port d data reg.
00001009 ddrd     equ REG+$09 port d data direction reg.
0000100a porte    equ REG+$0a port e data reg.
0000100b cforc    equ REG+$0b timer compare force reg.
0000100c oc1m     equ REG+$0c output compare 1 mask reg.
0000100d oc1d     equ REG+$0d output compare 1 data reg.
0000100e tcnt     equ REG+$0e timer count reg. 16 bit
0000100e tcnth    equ REG+$0e timer count reg. high
0000100f tcntl    equ REG+$0f timer count reg. low
00001010 tic1h    equ REG+$10 timer input capture reg. high
00001011 tic1l    equ REG+$11 tic1 low
00001012 tic2h    equ REG+$12 tic2 high
00001013 tic2l    equ REG+$13 tic2 low
00001014 tic3h    equ REG+$14 tic3 high
00001015 tic3l    equ REG+$15 tic3 low
00001016 toc1h    equ REG+$16 timer output compare reg. high
00001017 toc1l    equ REG+$17 toc1 low
00001018 toc2h    equ REG+$18 toc2 high
00001019 toc2l    equ REG+$19 toc2 low
0000101a toc3h    equ REG+$1a toc3 high
0000101b toc3l    equ REG+$1b toc3 low
0000101c toc4h    equ REG+$1c toc4 high
0000101d toc4l    equ REG+$1d toc4 low
0000101e toc5h    equ REG+$1e toc5 high
0000101f toc5l    equ REG+$1f toc5 low
00001020 tctl1    equ REG+$20 timer control reg. 1
00001021 tctl2    equ REG+$21 timer control reg. 2
00001022 tmsk1    equ REG+$22 main timer interrupt mask reg.1
00001023 tflg1    equ REG+$23 main timer interrupt flag reg.1
00001024 tmsk2    equ REG+$24 misk. timer interrupt mask reg.2
00001025 tflg2    equ REG+$25 misc. timer interrupt flag
00001026 pactl    equ REG+$26 pulse accumulator control reg.
00001027 pacnt    equ REG+$27 pulse accumulator count reg.
00001028 spcr     equ REG+$28 spi control reg.
00001029 spsr     equ REG+$29 spi status reg.
0000102a spdr     equ REG+$2a spi data reg.
0000102b baud     equ REG+$2b sci baud rate control reg.
0000102c sccr1    equ REG+$2c sci control reg. 1
0000102d sccr2    equ REG+$2d sci control reg. 2
0000102e scsr     equ REG+$2e sci status reg.
0000102f scdr     equ REG+$2f sci data reg.
00001030 adctl    equ REG+$30 a/d control/status reg.
00001031 adr1     equ REG+$31 a/d result reg. 1
00001032 adr2     equ REG+$32 a/d result reg. 2
00001033 adr3     equ REG+$33 a/d result reg. 3
00001034 adr4     equ REG+$34 a/d result reg. 4
00001035 bprot    equ REG+$35 block protect reg.
00001039 option   equ REG+$39 system configuration options
0000103a coprst   equ REG+$3a arm/reset cop timer circuitry
0000103b pprog    equ REG+$3b eeprom programming reg.
0000103c hprio    equ REG+$3c highest priority interrupt and misc.
0000103d init     equ REG+$3d ram and i/o mapping reg.
0000103e test1    equ REG+$3e factory test reg.
0000103f config   equ REG+$3f configuration control reg.
```

```
00000000 oporta    equ  $00   port a data reg.
00000002 opioc     equ  $02   parallel i/o control reg.
00000003 oportc    equ  $03   port c data reg.
00000004 oportb    equ  $04   port b data reg.
00000005 oportcl   equ  $05   port c latched data reg.
00000007 oddrc     equ  $07   port c data direction reg.
00000008 oportd    equ  $08   port d data reg.
00000009 oddrd     equ  $09   port d data direction reg.
0000000a oporte    equ  $0a   port e data reg.
0000000b ocforc    equ  $0b   timer compare force reg.
0000000c ooc1m     equ  $0c   output compare 1 mask reg.
0000000d ooc1d     equ  $0d   output compare 1 data reg.
0000000e otcnth    equ  $0e   timer count reg. high
0000000f otcntl    equ  $0f   timer count reg. low
00000010 otic1h    equ  $10   timer input capture reg. high
00000011 otic1l    equ  $11   tic1 low
00000012 otic2h    equ  $12   tic2 high
00000013 otic2l    equ  $13   tic2 low
00000014 otic3h    equ  $14   tic3 high
00000015 otic3l    equ  $15   tic3 low
00000016 otoc1h    equ  $16   timer output compare reg. high
00000017 otoc1l    equ  $17   toc1 low
00000018 otoc2h    equ  $18   toc2 high
00000019 otoc2l    equ  $19   toc2 low
0000001a otoc3h    equ  $1a   toc3 high
0000001b otoc3l    equ  $1b   toc3 low
0000001c otoc4h    equ  $1c   toc4 high
0000001d otoc4l    equ  $1d   toc4 low
0000001e otoc5h    equ  $1e   toc5 high
0000001f otoc5l    equ  $1f   toc5 low
00000020 otctl1    equ  $20   timer control reg. 1
00000021 otctl2    equ  $21   timer control reg. 2
00000022 otmsk1    equ  $22   main timer interrupt mask reg.1
00000023 otflg1    equ  $23   main timer interrupt flag reg.1
00000024 otmsk2    equ  $24   misk. timer interrupt mask reg.2
00000025 otflg2    equ  $25   misc. timer interrupt flag
00000026 opactl    equ  $26   pulse accumulator control reg.
00000027 opacnt    equ  $27   pulse accumulator count reg.
00000028 ospcr     equ  $28   spi control reg.
00000029 ospsr     equ  $29   spi status reg.
0000002a ospdr     equ  $2a   spi data reg.
0000002b obaud     equ  $2b   sci baud rate control reg.
0000002c osccr1    equ  $2c   sci control reg. 1
0000002d osccr2    equ  $2d   sci control reg. 2
0000002e oscsr     equ  $2e   sci status reg.
0000002f oscdr     equ  $2f   sci data reg.
00000030 oadctl    equ  $30   a/d control/status reg.
00000031 oadr1     equ  $31   a/d result reg. 1
00000032 oadr2     equ  $32   a/d result reg. 2
00000033 oadr3     equ  $33   a/d result reg. 3
00000034 oadr4     equ  $34   a/d result reg. 4
00000035 obprot    equ  $35   block protect reg.
00000039 ooption   equ  $39   system configuration options
0000003a ocoprst   equ  $3a   arm/reset cop timer circuitry
0000003b opprog    equ  $3b   eeprom programming reg.
0000003c ohprio    equ  $3c   highest priority interrupt and misc.
0000003d oinit     equ  $3d   ram and i/o mapping reg.
0000003e otest1    equ  $3e   factory test reg.
0000003f oconfig   equ  $3f   configuration control reg.
```

file: a_g_lcd.a

```
                                bss

0000  00000001        row             ds.b    1
0001  00000001        column          ds.b    1
0002  00000001        setreset        ds.b    1               * 1/0 - Bit setzen/rücksetzen
0003  00000001        mem             ds.b    1               * Arbeitszelle für Pixel


      00000028  maxcol          equ     40              * Anzahl Spalten
      00000140  maxpos          equ     320             * Anz.von Cursorpos.
      00000006  widthcol                equ     6               * Spaltenbreite
      000000ef  xmax            equ     239
      000000ee  xmax1           equ     xmax-1
      0000003f  ymax            equ     63


                                text
      0000b600                  org     EEPROM
b600  8e00ff    start           lds     #$00ff          * Stack = $00ff
b603  8620                      ldaa    #$20
file: a_g_lcd.a

b605  b71009                    staa    ddrd

*****************  * SCI initialisieren *
b608  bdb7b3                    jsr     sciini

*****************  * LCD initialisieren *
b60b  bdb6b7                    jsr     lcd2ini         * Init.-Routine
b60e  ce1000                    ldx     #REG
b611  bdb757                    jsr     cls             * TEXT-Bildschirm löschen
b614  7f0002                    clr     setreset        * Pixel löschen
b617  bdb79f                    jsr     clsg            * Grafik-Bild löschen


b61a  cc0000    wrap            ldd     #0
b61d  bdb6f1    wrpos           jsr     lcd2pos         * write position
b620  bdb69a    loop            jsr     cupos           * write CU-position
b623  36                        psha
b624  bdb7c6                    jsr     sciget          * wait for key
b627  810c                      cmpa    #$0c
b629  2746                      beq     ff
b62b  810d                      cmpa    #$0d
b62d  271b                      beq     cr
b62f  810a                      cmpa    #$0a
b631  2730                      beq     lf
b633  8108                      cmpa    #$08
b635  272f                      beq     bs
b637  811b                      cmpa    #$1b
b639  273c                      beq     esc
b63b  bdb748                    jsr     lcd2put         * put to lcd+INC wr-pos
b63e  32                        pula
b63f  c30001                    addd    #1
b642  1a830140                  cpd     #maxpos
b646  26d8                      bne     loop
b648  20d0                      bra     wrap
```

```
**********  * new line for cr

b64a  7c0000   cr          inc     row
b64d  32                   pula
b64e  4f                   clra
b64f  5f                   clrb
b650  c30028   cr1         addd    #maxcol
b653  7a0000               dec     row
b656  26f8                 bne     cr1
b658  1a830140       cr2           cpd     #maxpos
b65c  2dbf                 blt     wrpos
b65e  830140               subd    #maxpos
b661  20ba     cr3         bra     wrpos

**********  * lf ignore

b663  32       lf          pula
b664  20f2                 bra     cr2

**********  * bs

b666  32       bs          pula
b667  830001               subd    #1
b66a  2ab1                 bpl     wrpos
b66c  cc013f               ldd     #maxpos-1
b66f  20ac                 bra     wrpos
:
**********  * ff

b671  32       ff          pula
b672  bdb757               jsr     cls
b675  20a6                 bra     wrpos

**********  * esc -> Pixel setzen/rücksetzen

b677  37       esc         pshb
b678  bdb7c6               jsr     sciget      * Y-Position (Bit8 = Pixel)
b67b  16                   tab
b67c  4f                   clra
b67d  c580                 bitb    #$80
b67f  2702                 beq     res
b681  8601                 ldaa    #1
b683  9702     res         staa    <setreset   * Pixel setzen = Linien Zeichnen
b685  bdb7c6               jsr     sciget      * X-Position
b688  c540                 bitb    #$40
b68a  2705                 beq     nocl
b68c  bdb79f               jsr     clsg
b68f  2005                 bra     eclsg
b691  c43f     nocl        andb    #$3f
b693  bdb770               jsr     pixel
b696  33       eclsg       pulb
b697  32                   pula
b698  2083                 bra     wrpos
```

```
**********  * Unterprogramme *

**********  * char.-adr. to CU-pos
**********  * ACCD = char.+adr for write position

b69a  36        cupos           psha
b69b  37                        pshb
b69c  c6ff                      ldab    #-1
b69e  d700                      stab    <row
b6a0  33                        pulb
b6a1  37                        pshb
b6a2  7c0000    cu1             inc     row
b6a5  830028                    subd    #maxcol
b6a8  2af8                      bpl     cu1
b6aa  c30028                    addd    #maxcol
b6ad  d701                      stab    <column
b6af  9600                      ldaa    <row
b6b1  bdb703                    jsr     lcd2cus
b6b4  33                        pulb
b6b5  32                        pula
b6b6  39                        rts
file: lcd2x.a

********** LCD2 module **********************************************************************
*                                                                                             *
* Date        Version What                                                        Who      *
* 25.10.91      1.00    create                                                   he       *
* ------------------------------------------------------------------------------------------*
*                         - Notes -                                                           *
*                                                                                             *
* Wie LCD1.A, aber für grafische LCDisplays.                                                  *
*                                                                                             *
* ACHTUNG:                                                                                    *
*  - lcd2ini braucht die Anzahl der Spalten der verwendeten Anzeige                           *
*  - lcd2dat/cmd haben zusätzlich den gewünschten Status als Parameter.                       *
* ------------------------------------------------------------------------------------------*
*                         - Usage -                                                           *
*                                                                                             *
* include in application                                                                      *
* ------------------------------------------------------------------------------------------*
* Copyright (C) MCT Paul & Scherer Mikrocomputertechnik GmbH Berlin                           *
*********************************************************************************************

****** init LCD (no register save)
*
b6b7  ceb6d4    lcd2ini         ldx     #lcditb         * init table address    *
b6ba  e600      lcdi1           ldab    0,x             * get # of parameters   *
b6bc  08        lcdi2           inx                     * next...               *
b6bd  a600                      ldaa    0,x             * get parameter/cmd     *
b6bf  5a                        decb                    * parameter?            *
b6c0  2b08                      bmi     lcdi3           * no, write cmd         *
b6c2  37                        pshb
b6c3  c603                      ldab    #3              * requested status      *
b6c5  8d54                      bsr     lcd2dat         * write parameter       *
b6c7  33                        pulb
b6c8  20f2                      bra     lcdi2           * more parameters...    *
b6ca  c603      lcdi3           ldab    #3              * requested status      *
b6cc  8d47                      bsr     lcd2cmd         * write cmd             *
b6ce  08                        inx                     * next...               *
b6cf  6d00                      tst     0,x             * end?                  *
b6d1  2ae7                      bpl     lcdi1           * no, more cmds...      *
b6d3  39                        rts
```

```
b6d4  0081      lcditb          dc.b    0,$81           * mode = ROMzg TX xor Gr
b6d6  02001024                  dc.b    2,$00,$10,$24   * Adr.-pointer =$1000
b6da  02000021                  dc.b    2,$00,$00,$21   * CU  -pointer =0
b6de  02001042                  dc.b    2,$00,$10,$42   * Graphic Home Addr. = $1000
b6e2  02280043                  dc.b    2,$28,$00,$43   * Column = 40 zu 6 Bit
b6e6  02000040                  dc.b    2,$00,$00,$40   * Text Home Addr.= 0
b6ea  02280041                  dc.b    2,$28,$00,$41   *   Column = 40
b6ee  009c                      dc.b    0,$9c           * Text,Graphic = ON
b6f0  ff                        dc.b    -1

****** set write position
*         ACCD = adress (see LCD data sheet)
*
b6f1  37        lcd2pos         pshb
b6f2  36                        psha
b6f3  17                        tba
b6f4  c603                      ldab    #3              * requested status      *
b6f6  8d23                      bsr     lcd2dat         * address LSB           *
b6f8  32                        pula
b6f9  8d20                      bsr     lcd2dat         * address MSB           *
b6fb  36                        psha
b6fc  8624                      ldaa    #$24
b6fe  8d15                      bsr     lcd2cmd         * address pointer set   *
b700  32                        pula
b701  33                        pulb
b702  39                        rts

****** set CU position
*         ACCA = CU-position ROW (see LCD data sheet)
*         ACCB = CU-position COLUMN
*
b703  37        lcd2cus         pshb
b704  36                        psha
b705  17                        tba
b706  c603                      ldab    #3              * requested status      *
b708  8d11                      bsr     lcd2dat         * COLUMN position       *
b70a  32                        pula
b70b  8d0e                      bsr     lcd2dat         * ROW    position       *
b70d  36                        psha
b70e  8621                      ldaa    #$21
b710  8d03                      bsr     lcd2cmd         * address pointer set   *
b712  32                        pula
b713  33                        pulb
b714  39                        rts

****** write cmd to LCD
*         ACCA = cmd
*         ACCB = requested status (see LCD data sheet)
*
b715  37        lcd2cmd         pshb
b716  36                        psha
b717  8621                      ldaa    #$21            * prepare cmd write     *
b719  2004                      bra     lcdd1
```

```
****** write data to LCD
*         ACCA = data
*         ACCB = requested status (see LCD data sheet)
*
b71b  37        lcd2dat         pshb
b71c  36                        psha
b71d  8620                      ldaa    #$20            * prepare data write    *
b71f  7f1007    lcdd1           clr     ddrc            * portc all inputs              *
b722  36                        psha
b723  8671      lcdd2           ldaa    #$71
b725  b71004                    staa    portb           * status                *
b728  8611                      ldaa    #$11
b72a  b71004                    staa    portb           * read enable           *
b72d  17                        tba
b72e  b41003                    anda    portc
b731  11                        cba                     * requested status?     *
b732  26ef                      bne     lcdd2           * no, try again...      *
b734  33                        pulb
b735  f71004                    stab    portb           * cmd/data write enable *
b738  86ff                      ldaa    #$ff
b73a  b71007                    staa    ddrc            * portc all outputs     *
b73d  32                        pula
b73e  b71003                    staa    portc           * cmd/data out          *
b741  ca70                      orab    #$70
b743  f71004                    stab    portb           * latch cmd/data                *
b746  33                        pulb
b747  39                        rts

****** put char in ACCA to lcd (address incremented)
*
b748  36        lcd2put         psha
b749  37                        pshb
b74a  c603                      ldab    #3              * requested status      *
b74c  8020                      suba    #" "            * blank is code 0       *
b74e  8dcb                      bsr     lcd2dat
b750  86c0                      ldaa    #$c0
b752  8dc1                      bsr     lcd2cmd         * data writeaddr incr   *
b754  33                        pulb
b755  32                        pula
b756  39                        rts
file: cls_lcd2.a

**** CLS for LCD
*
b757  cc0000    cls             ldd     #0
b75a  bdb6f1                    jsr     lcd2pos
b75d  cc0140                    ldd     #maxpos
b760  36        cls1            psha
b761  8620                      ldaa    #" "
b763  bdb748                    jsr     lcd2put
b766  32                        pula
b767  830001                    subd    #1
b76a  26f4                      bne     cls1
b76c  bdb6f1                    jsr     lcd2pos
b76f  39                        rts
file: a_g_lcd.a
```

```
****************** * Pixel setzen/rücksetzen *              0------------> 239=xmax
****************** * A = X-Pos.   von 0 ... 239             |
****************** * B = Y-Pos.   von 0 ... 63              |
****************** * (setreset) = 1/0 setzen/rücksetzen     v 63=ymax

b770  36         pixel          psha                       * X-Pos. merken

b771  8628                      ldaa    #maxcol            * A = maxcol für Pixel
b773  3d                        mul                        * D = A*B = maxcol*ZeileY = Zeiger
b774  c31000                    addd    #$1000             * Anf.-Adr. Pixelram
b777  188f                      xgdy                       * IY = Zeiger auf Y-Pos.

b779  4f                        clra
b77a  33                        pulb                       * D = X-Pos.
b77b  3c                        pshx
b77c  ce0006                    ldx     #widthcol          * IX = Spaltenbreite
b77f  02                        idiv                       * IX = D/IX ; D=Rest
b780  8f                        xgdx                       * IXlow jetzt in B
b781  183a                      aby                        * Pixeladresse in IY bilden
b783  8f                        xgdx                       * in B ist Rest
b784  8605                      ldaa    #widthcol-1
b786  10                        sba                        * Bits 0 ... 5
b787  38                        pulx

b788  36                        psha                       * Bitposition merken

****************** * Address Pointer zu LCD senden *

b789  188f                      xgdy                       *  D = Position
b78b  bdb6f1                    jsr     lcd2pos

****************** * Bit-Set Commando aufbauen und senden *
b78e  32                        pula                       * A = Bitpos.(untere 3 Bit )
b78f  8af8                      oraa    #$f8               * Bit setzen annehmen (4.Bit )
b791  d602                      ldab    <setreset          * B = setreset
b793  c401                      andb    #1                 * test ob setzen gewollt
b795  2602                      bne     setbit             * wenn ja, weiter bei setbit
b797  84f7                      anda    #$f7               * wenn nein, 4.Bit in A rücksetzen
b799  c603       setbit         ldab    #3                 * B = 3, Status LCD
b79b  7eb715                    jmp     lcd2cmd            * Bit setzen Commando

****************** * Grafik-Bildschirm löschen *
b79e  c640       clsg           ldab    #ymax+1            * Y - Position für Löschen
b7a0  86f0       clr0           ldaa    #xmax+1            * X - Position für Löschen
b7a2  36         clr1           psha
b7a3  37                        pshb
b7a4  4a                        deca                       * X von 239 ... 0
b7a5  5a                        decb                       * Y von  63 ... 0
b7a6  bdb770                    jsr     pixel
b7a9  33                        pulb
b7aa  32                        pula
b7ab  4a                        deca
b7ac  26f4                      bne     clr1
b7ae  5a                        decb
b7af  26ef                      bne     clr0
b7b1  39                        rts
```

file: sci_dtr.a

```
************** SCI module *******************************************************************
*                                                                                          *
* Date          Version What                                                    Who        *
* 25.10.91      1.00    create                                                  he         *
* -----------------------------------------------------------------------------------------*
*                                   - Notes -                                              *
*                                                                                          *
* Dieses Modul stellt Funktionen für die serielle Schnittstelle (SCI) des                  *
* HC11 zur Verfügung.                                                                      *
* -----------------------------------------------------------------------------------------*
*                                   - Usage -                                              *
*                                                                                          *
* include in application                                                                   *
* -----------------------------------------------------------------------------------------*
* Copyright (C) MCT Paul & Scherer Mikrocomputertechnik GmbH Berlin                             *
********************************************************************************************

****** init sci (no register save)
*
b7b2  8630      sciini  ldaa    #$30
b7b4  b7102b            staa    baud                    * 9600 baud         *
b7b7  8608              ldaa    #$8
b7b9  ba1000            oraa    porta
b7bc  b71000            staa    porta                   * DTR Busy set      *
b7bf  860c              ldaa    #$0c
b7c1  b7102d            staa    sccr2                   * enable sci        *
b7c4  39                rts

****** tst if char from sci available
*     return ACCA = 0 if not
*
*               scitst  ldaa    scsr
*                       anda    #$20                    * receive buf full? *
*                       rts                             * return 0 if not   *

****** get char from sci to ACCA
*
b7c5  b6102e    sciget  ldaa    scsr
b7c8  8420              anda    #$20                    * receive buf full? *
b7ca  260a              bne     scig1                   * yes               *
b7cc  86f7              ldaa    #$f7
b7ce  b41000            anda    porta
b7d1  b71000            staa    porta                   * DTR not Busy set  *
b7d4  20ef              bra     sciget
b7d6  8608      scig1   ldaa    #$8
b7d8  ba1000            oraa    porta
b7db  b71000            staa    porta                   * DTR Busy set      *
b7de  b6102f            ldaa    scdr                    * read from buf     *
b7e1  39                rts

****** put char in ACCA to sci
*
*               sciput  tst     scsr                    * transmit buf empty? *
*                       bpl     sciput                  * no, try again...    *
*                       staa    scdr                    * write to buf        *
*                       rts
```