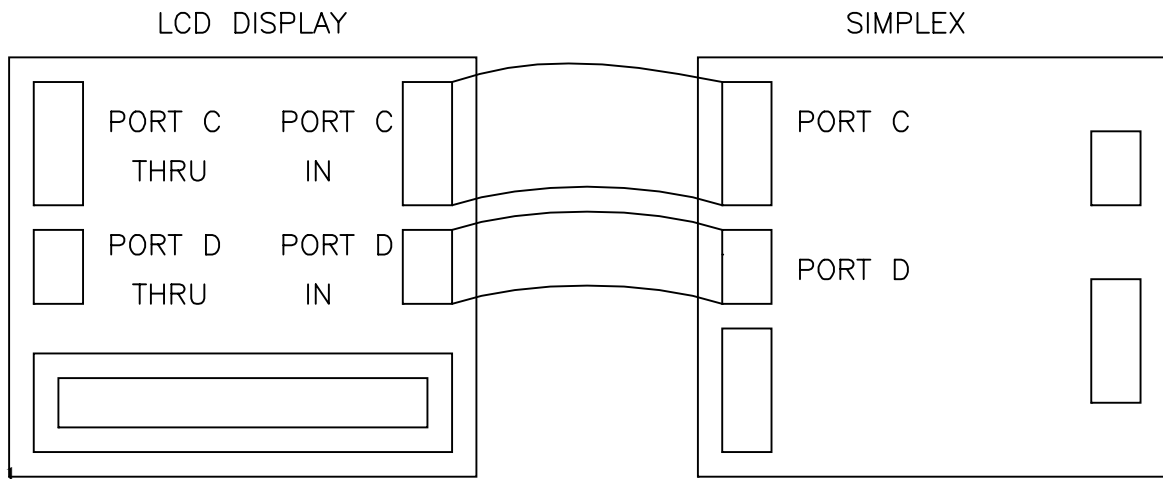


Het LCD wordt op de SIMPLEX aangesloten via poort C en poort D.



Poort D wordt alleen gebruikt om het LCD van voeding te voorzien. Voor de besturing en het uitlezen van het toetsenbord is poort C gebruikt.

Beide poorten zijn aan de linkerkant van het LCD opnieuw op connectors beschikbaar. Hierdoor kunnen andere uitbreidingen eenvoudig worden aangesloten.

Het LCD is een intelligente module, die wordt aangestuurd door middel van poort C. Het LCD beschikt over een 4-bits databus, en enkele controle signalen. De aansluitingen zijn als volgt:

- Adreslijn aangesloten op poort C, bit1
- Read/write aangesloten op poort C, bit 2
- Enable aangesloten op poort C, bit 3
- Databus bit4, aangesloten op poort C, bit4
- Databus bit5, aangesloten op poort C, bit5
- Databus bit6, aangesloten op poort C, bit6
- Databus bit7, aangesloten op poort C, bit7

LET OP: De bits PC7, PC6, PC5 en PC4 worden ook gebruikt voor het toetsenbord.

Het LCD beschikt over twee registers. De keuze tussen de registers wordt gemaakt middels de adreslijn (poort C, bit 1.) Als de adreslijn een '0' is, wordt het instructieregister geactiveerd, als de adreslijn een '1' is, wordt het dataregister geactiveerd.

Elk van de registers kan gelezen en geschreven worden. Lezen of schrijven wordt bepaald door read/write (poort C, bit2.) Een '1' op read/write betekent lezen van een register, een '0' op de read/write lijn betekent schrijven.

Bij elke lees- of schrijf actie moet een puls op de enable-lijn gegeven worden.

Bij lezen: Zet eerst de adreslijn goed, en de read/write lijn op '1'. Zorg ervoor dat bit4 t/m bit7 van poort C als ingang geprogrammeerd staan. Maak dan de enable-lijn hoog. Nu zet het LCD de data op zijn databus. De data kan via poort C gelezen worden. Daarna moet de enable-lijn weer laag gemaakt worden.

Bij schrijven: Zet de adreslijn goed, en de read/write lijn op '0'. Plaats de data voor het LCD op bit4 t/m bit7 van poort C, en zorg ervoor dat deze bits als uitgang geprogrammeerd staan. Maak dan de enable-lijn hoog, en daarna weer laag. Het LCD neemt de data over.

Hoewel het LCD slechts een 4-bits databus heeft, zijn zowel het instructie- als het dataregister 8 bits breed. Dit betekent dat lezen en schrijven van de registers steeds in twee slagen moet gebeuren. Eerst wordt het meest significante nibble (4 bits) van het register gelezen (geschreven), en daarna het minst significante nibble.

Het instructie register wordt gebruikt om het display opdrachten te geven. De opdrachten kunnen zijn:

- Clear screen.
- Display aan, uit.
- Cursor aan, uit.
- Cursor knipperen.
- Schuif de tekst naar links als er een karakter geschreven wordt.
- Schuif de tekst naar rechts als er een karakter geschreven wordt.
- Positioneer de cursor.

Tekens worden op het display gezet door deze één voor één naar het dataregister te schrijven.

Het volgende programma laat zien hoe het LCD geprogrammeerd moet worden. Voor enkele commando's is een aparte subroutine geschreven. Andere commando's zijn reeds gedefiniëerd, en kunnen aan het display gegeven worden met behulp van de routine 'lcdcommand'.

Om de cursor aan te zetten, gebruikt U bijvoorbeeld:

```
ldab #(lcdonoff or lddispon or lcdcuron)
jsr lcdcommand
```

en wanneer de cursor moet knipperen:

```
ldab #(lcdonoff or lddispon or lcdcuron or lcdblinkon)
jsr lcdcommand
```

Tekst wordt op het display gezet door middel van de subroutine 'lcdputchar' (voor één teken), of 'lcdpstring' (voor een string.)

De cursor kan op een bepaalde plaats van het display gezet worden via de subroutine 'lcdputputcursor'. Dit is de plaats waar het volgende teken geplaatst zal worden (ongeacht of de cursor aan- of uit staat.)

```

*****
* definiëren van de geheugen map
*****
        incl "map512.asm"

*****
* start van het programma
*****
PROGRAM          space          |kies het programma-gebied

reset            equ $           |na reset begint de micro op deze plaats
                lds #stackend    |begin met de stackpointer te laden

                ldx #databeg     |en zet dan het volledige datagebied op 00
clearram        clr 0,x
                inx
                cpx #dataend
                bls clearram

                ldx #regsbeg     |laat IX op de bank met I/O registers wijzen

*****
* modules
*****

*****
* LCD display
*****
PROGRAM          space
lcdstart        equ $           |het beginadres van deze module

lcdbusybit     equ bit7        |is een '1' als het lcd bezig is

* LCD commando codes
ldcclearcode   equ $01         |maakt het display leeg
lcdreturncode  equ $02         |zet cursor op locatie 0
lcdshiftright  equ $07         |schuif display naar rechts
lcdshiftright  equ $05         |schuif display naar links
lcdnoshift     equ $06         |display niet schuiven
lcdonoff       equ $08         |basis voor on/off control
lcdddispon     equ bit2        |combineer met lcdonoff voor display 'aan'
lcdcuron       equ bit1        |combineer met lcdonoff voor cursor 'aan'
lcdblinkon     equ bit0        |combineer met lcdonoff voor cursor knipperen
lcdsetaddress  equ $80         |combineer met adres ( < $80)

PROGRAM          space
                bra lcdinit

* lees het instructieregister van het lcd in (B) (alleen de
* 4 meest significante bits zijn geldig).
lcdlees        equ $
                ldab #bit2      |maak read-write lijn een '1'
                stab portc
                orab #bit3      |maak het enable signaal een '1'
                stab portc
                nop
                ldab portc      |lees de data
                pshb

```

```

ldab #bit2      |maak enable signaal laag, laat
stab portc     |read-write op een '1'
pulb
rts

```

* stuur data van (B) naar het LCD (alleen de 4 meest significante bits)

```

lcdzend      equ $
psha
pshb
andb #not(bit2) |read-write lijn moet een '0' zijn
ldaa ddrcc    |maak de datalijnen uitgangen
oraa #(bit7 or bit6 or bit5 or bit4)
staa ddrcc
stab portc    |zet de inhoud van (B) op poort C
orab #bit3    |enable-lijn is aangesloten op bit3
stab portc    |geef een enable-puls
andb #not( bit3)
stab portc
anda #low(not( bit7 or bit6 or bit5 or bit4))
staa ddrcc    |maak data-lijnen weer inputs
pulb
pula
rts

```

* wacht tot het lcd klaar is met het vorige commando

```

lcdwait      equ $
pshb          |bewaars register
lcdwait0     bsr lcdlees
pshb
bsr lcdlees
pulb
andb #lcdbusybit
bne lcdwait0 |wacht totdat het busy-bit een '0' is
pulb          |haal register terug van de stack
rts

```

* stuur een commando naar het lcd, commando in (B)

```

lcdcommand  equ $
psha
tpa
psha
sei          |blokkeer interrupts tijdelijk
bsr lcdwait |wacht totdat het lcd klaar is
pshb
andb #$F0
bsr lcdzend |verstuur het meest significante nibble
pulb
pshb
lslb
lslb
lslb
lslb
bsr lcdzend |en dan het minst significante nibble
pula
tap
pula
pulb
rts

```

* wachtlus voor timing tijdens initialisatie

```
lcdw      equ $
          pshx
          ldx #0
lcdw0     dex
          bne lcdw0
          pulx
          rts
```

* initialisatie van het LCD

```
lcdinit      ldab portc |zet enable-lijn op '0'
             andb #not( bit3)
             stab portc
             ldab ddrcc |en kies dan in- en uitgangen op poort C
             orab #(bit3 or bit2 or bit1)
             andb #low(not(bit7 or bit6 or bit5 or bit4))
             stab ddrcc
             bsr lcdw    |wacht totdat de LCD gereset is
             ldaa #3
lcdinit0     ldab #$30
             bsr lcdzend |initialiseer display driver I.C. op het lcd
             bsr lcdw    |wacht totdat het command uitgevoerd is
             deca        |dit commando moet meermaals gegeven worden
             bne lcdinit0
             ldab #$20   |kies nu 4-bits interface
             bsr lcdzend
             bsr lcdw
             ldab #$28   |kies 2-regelig display
             bsr lcdcommand
             bsr lcddisploff
             bsr lcdclearscreen
             ldab #lcdnoshift
             bsr lcdcommand |geen display shift
             jsr lcddisplon
             bra lcdend   |einde van de initialisatie
```

***** LCD routines

* maak het LCD scherm schoon

```
lcdclearscreen equ $
              pshb
              ldab #lcdclearcode
              bsr lcdcommand
              pulb
              rts
```

```

* Display aanzetten
lcddisplon equ $
    pshb
    ldab #lcdonoff+lcddispon
    bsr lcdcommand
    pulb
    rts

* Display uitzetten
lcddisploff equ $
    pshb
    ldab #lcdonoff
    bsr lcdcommand
    pulb
    rts

* Zet cursor op locatie X, Y. (B) = Y + 2*X {y=0..1}
lcdsetcursor equ $
    psha
    pshb
    tba
    anda #%11111110
    lsra
    bitb #bit0
    beq lcdsetcursor0
    adda #$40
lcdsetcursor0 tab
    orab #lcdsetaddress
    bsr lcdcommand
    pulb
    pula
    rts

* Zet een character op het lcd, character verwacht in (B)
lcdputchar equ $
    psha
    tpa
    psha
    sei          |blokkeer interrupts tijdelijk
    pshb
    jsr lcdwait
    andb #$F0
    orab #bit1
    jsr lcdzend |verstuur het meest significante nibble
    pulb
    pshb
    lslb
    lslb
    lslb
    lslb
    orab #bit1
    jsr lcdzend |en dan het minst significante nibble
    pula
    tap
    pula
    pulb
    rts

```

```
* Stuur een string naar het LCD, X wijst op string die met 00 is afgesloten
lcdpstring equ $
           pshb
lcdpstring0 ldab 0,x
            beq lcdpstring9
            bsr lcdputchar
            inx
            bra lcdpstring0
lcdpstring9 pulb
           rts

lcdend          equ $          |einde van de lcd module

* het aantal bytes dat deze module nodig heeft in het programma gebied
lcdsize        equ lcdend-lcdstart
```

```

*****
* het hoofdprogramma
*****
PROGRAM          space
                 bra main0

st1              fcc 'SIMPLEX',0
st2              fcc 'LCD',0

delay           equ $
                 ldab #1
main00          ldx #0
main01          dex
                 bne main01
                 decb
                 bne main00
                 rts

main0           ldab #(0 + 2* 4)
                 jsr lcdsetcursor
                 ldx #st1
                 jsr lcdpstring
                 ldab #(1 + 2* 6)
                 jsr lcdsetcursor
                 ldx #st2
                 jsr lcdpstring
main10          jsr delay
                 jsr lcddisploff
                 jsr delay
                 jsr lcddisplon
                 bra main10

end

```