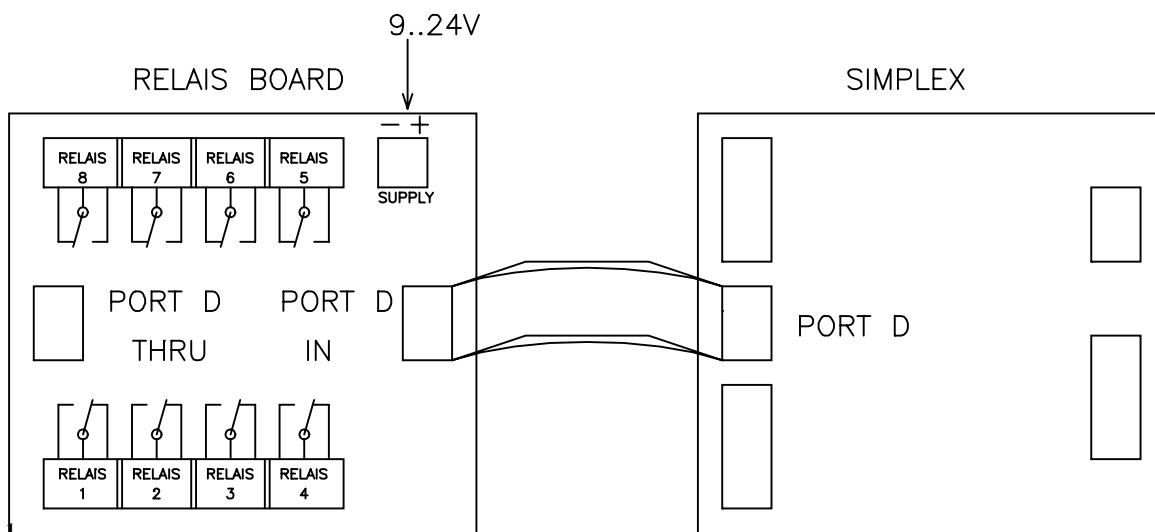


Het relaisboard wordt op de SIMPLEX aangesloten via poort D.



Poort D wordt gebruikt om de relais via de uitbreidings poort te besturen. Er kunnen meerdere relaisboards toegepast worden. De boards worden dan in serie geplaatst.

Poort D is aan de linkerkant van het relaisboard opnieuw op een connectors beschikbaar. Hierdoor kunnen andere uitbreidingen eenvoudig worden aangesloten, bijvoorbeeld een tweede relaisboard of één of meer boards met optisch geïsoleerde ingangen.

De relais worden (na gebufferd te zijn) aangestuurd vanuit de uitgangen van een schuifregister. Dit 8-bits schuifregister wordt geladen middels de uitbreidingspoort. Met deze poort kunnen bytes seriëel worden verstuurd. Wanneer het byte in het schuifregister is aangekomen, wordt met behulp van een 'strobe' lijn het nieuwe byte op de uitgangen van het schuifregister gezet.

De aansluitingen zijn als volgt:

- seriële data op poort D, bit 3
- klok op poort D, bit 4
- strobe op poort D, bit 5

Wanneer meerdere relaisboards aangesloten worden, zal het byte dat als eerste verstuurd wordt in eerste instantie terechtkomen in het board dat in de keten het dichtst bij de SIMPLEX is aangesloten. Wanneer een volgend byte verstuurd wordt, schuift het eerste byte naar het volgende board, en het nieuwe byte komt in het voorste board terecht. Zo kunnen alle boards geladen worden met een nieuw byte. Daarna kan de 'strobe' puls gegeven worden, waardoor de schuifregisters de informatie overnemen op de uitgangen.

Het volgende programma is geschikt om meerdere boards te bedienen, en houdt ook rekening met eventuele input boards. De constanten 'extra_ingangen', en 'extra_uitgangen' moeten worden ingesteld op het aantal boards met extra ingangen, resp. extra uitgangen. Voor één relaisboard stelt U 'extra_ingangen' in op 0, en 'extra_uitgangen' op 1.

Het programma bevat twee arrays van bytes ('inputs' en 'outputs'). Elk van deze arrays bevat evenveel bytes als werd ingesteld in de constanten. De subroutine 'refresh' gebruikt het 'output' array om de bytes te lezen die naar de relaisboards moeten, en vult het 'input' array met de bytes die van de ingangsboards gelezen werden.

Het eerste byte in een array is bestemd voor het achterste board in een keten (het board dat in de seriële keten het verst van de SIMPLEX verwijderd is.)

De procedure is:

- zend alle bytes van het array 'outputs' via de seriële poort naar de boards met extra uitgangen.
- maak de 'strobe' hoog, en daarna weer laag (de uitgangen worden nu overgenomen uit de

schuifregisters, en tevens worden de ingangen overgenomen in de schuifregisters.)
- lees de bytes uit de boards met extra ingangen, en sla deze op in het array 'inputs'.

```

*****
* definieren 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
*****

*****
* De uitbreidingspoort
*****
PROGRAM          space
expanstart      equ $           |het beginadres van deze module

spiss           equ bit5        |de 'strobe' lijn
spiclk          equ bit4        |klok
spimosi         equ bit3        |seriële data uit
spimiso         equ bit2        |seriële data in

extra_ingenen  equ 1           |aantal ingangs uitbreidingen van 8 bits
extra_uitgangen equ 1           |aantal uitgangs uitbreidingen van 8 bits

DATA           space
inputs         rmb extra_ingenen
outputs        rmb extra_uitgangen

* initialisatie van de uitbreidingspoort
PROGRAM          space
                ldab ddrd       |maak de pinnen uitgangen
                orab #(spiss or spiclk or spimosi)
                andb #not(spimiso)
                stab ddrd
                ldab portd      |maak de strobe een '0'
                andb #not( spiss)
                stab portd
                ldab spcr       |kies de snelheid en andere opties
                orab #(spe or mstr or cpol or cpha)
                andb #low(not( spie or dwom or spr1 or spr0))
                stab spcr
                bra expansend    |einde van de initialisatie

```

* hulproutines

* lees de ingangen, laat resultaat achter in 'inputs'

```
expansin    equ $
            pshb
            pshx
            ldab #extra_ingangen
            ldx #inputs      |wijs op plaats waar de ingangen opgeslagen
            abx
expansin0   tstb            |moeten gaan worden
            beq expansin99  |kijk of alle ingangen gedaan zijn
            pshb
            stab spdr       |verzend en lees 1 byte
expansin1   tst spsr
            bpl expansin1   |wacht tot het byte weg is, en het nieuwe
            ldab spdr       |aangekomen, sla dit dan op
            stab 0,x
            dex             |wijs op de plaats voor het volgende byte
            pulb
            decb
            bra expansin0
expansin99  pulx
            pulb
            rts
```

* schrijf de uitgangen, haal de waarden om weg te schrijven op uit 'outputs'

```
expansout   equ $
            pshb
            pshx
            ldab #extra_uitgangen
            ldx #outputs     |wijs op plaats waar de waarden voor de
expansout0  tstb            |uitgangen opgeslagen staan
            beq expansout99 |kijk of alle uitgangen gedaan zijn
            pshb
            ldab 0,x        |haal een byte voor de uitgangen op
            stab spdr       |verzend en lees 1 byte
expansout1  tst spsr
            bpl expansout1  |wacht tot het byte weg is
            inx             |wijs op de plaats voor het volgende byte
            pulb
            decb
            bra expansout0
expansout99 pulx
            pulb
            rts
```

**** Ververs de ingangen en de uitgangen. De extra ingangen worden
**** ingelezen in 'inputs', de extra uitgangen worde gevuld met de waarden
**** uit 'ouputs'.

```
refresh     equ $
            pshb
            bsr expansout   |vul de uitgangen
            ldab portd      |geef een strobe signaal
            orab #spiss
            stab portd
            andb #not(spiss)
```

```

        stab portd
        bsr expansin      |lees de ingangen
        pulb
        rts

expansend equ $          |einde van de uitbreidingspoort module

* het aantal bytes dat deze module nodig heeft in het programma gebied
expansize equ expansend-expansstart

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

DATA         space
teller      rmb 1

PROGRAM      space

delay       equ $
            ldx #30000
main01      dex
            bne main01
            rts

main0       clr teller
main1       ldab teller
            stab outputs
            jsr refresh
            inc teller
            bsr delay
            bra main1

            end

```