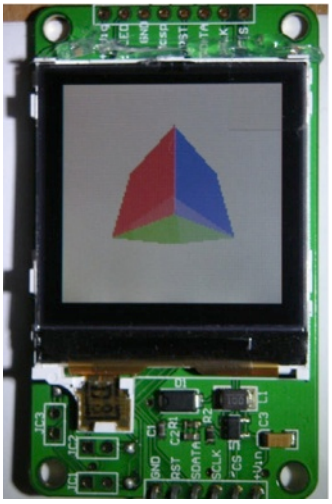
Opgave Tussentijdse Oefeningen Jaarproject I
Reeks 4: Lcd Interface & Files

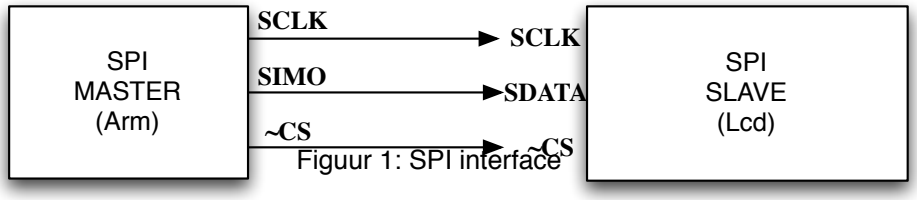


1 Introductie

In deze oefening zal je je LCD display leren aansturen. Je controleert deze display door er instructies naar te sturen. Deze instructies worden bit per bit verzonden. Wanneer instructies op deze manier verzonden worden spreken we van een serieel protocol. Het versturen van instructies over een serieel protocol komt vaak voor. Je gebruikt bijvoorbeeld een USB (Universal Serial Bus) poort om je fototoestel aan je computer te koppelen. In deze oefening zullen we een serieel protocol implementeren om er voor te zorgen dat de microcontroller kan “praten” met ons LCD scherm.

1.1 SPI

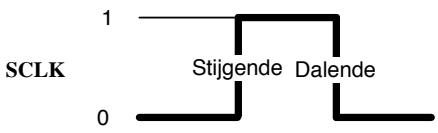
Het serieel protocol begrepen door het LCD scherm heet SPI (Serial Peripheral Interface Bus). Je display heeft drie pinnetjes die nodig zijn om te communiceren met je processor.



Figuur 1: SPI interface

Het eerste pinnetje SCLK (Serial Clock) geeft het ritme aan waarmee data doorgestuurd wordt. Door dit pinnetje net zoals in het blink voorbeeld in -en uit te schakelen weet je display de snelheid waarmee je processor data doorstuurt. Je pinnetje in en -uit schakelen zorgt ervoor dat er een puls wordt gestuurd. Zo een puls heeft een stijgende flank en een dalende flank, dit is weergeven op figuur 2. De display zal data net na de stijgende flank inlezen.

```
...
(set-pin sclk)
(clear-pin sclk)
...
```



Figuur 2: SCLK puls

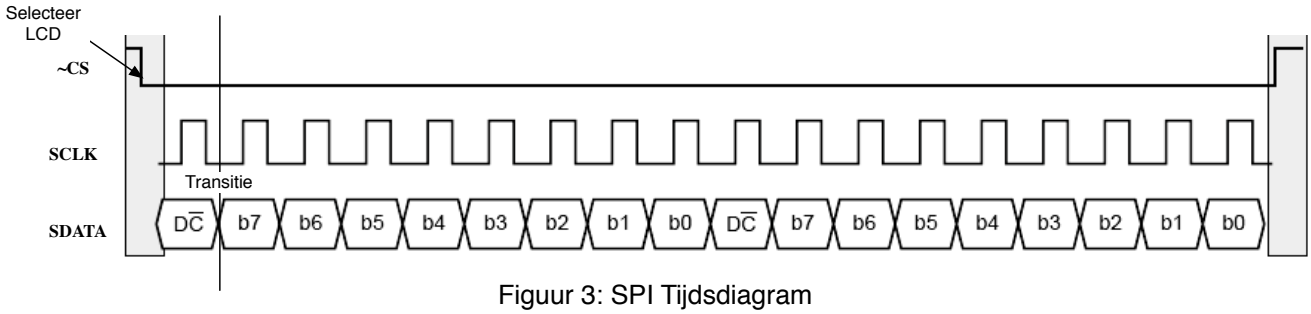
SCLK werkt samen met het tweede pinnetje SDATA(Serial Data). Dit pinnetje zal hoog of laag gemaakt worden door de microcontroller om respectievelijk een nul of een één door te sturen. De waarde van deze pin wordt dus uitgelezen door de display net na de stijgende flank van SCLK.

~CS (Chip select) dient om je display te selecteren. Net zoals je meerdere apparaten kan aansluiten op je computer, kan je ook meerdere apparaten verbinden met je

microcontroller. In SPI moet je hiervoor eerst het gewenste apparaat selecteren. Dit doe je door ~CS laag te maken.

2 Data doorsturen

Nu we de individuele pinnetjes gezien hebben, kunnen we instructies doorsturen naar onze display. De instructies bestaan uit een commando en data deel. Een commando is bijvoorbeeld DISPON dat de display aanschakelt. Een voorbeeld van data is een pixel coördinaat. Je kan dit vergelijken met een functieoproep waar je de naam (commando) eerst geeft en daarna de argumenten (data). Om een commando naar je display te sturen stuur je eerst een nul bit gevolgd door acht bits die aangeven welk commando je wil sturen. Om data naar je display te sturen, stuur je eerst een één gevolgd door de acht bits die je data voorstellen. Dit is weergeven in figuur 3.



Figuur 3: SPI Tijdsdiagram

Kort samengevat moeten we eerst onze display selecteren door ~CS laag te maken. Dan onze eerste bit op het pinnetje SDATA plaatsen om aan te geven dat we data of een commando sturen. Dan sturen we een puls op SCLK. Op dit moment zal de display de bit die we op pinnetje SDATA hebben geplaatst lezen. Vervolgens zetten we bit zeven en sturen we de volgende puls op SCLK. Dit herhalen we tot bit nul. Heel deze sequentie herhalen we voor elke instructie die we willen doorsturen. Op het einde plaatsen we ~CS weer op één zodanig dat onze display niet meer geselecteerd is.

In Scheme ziet het doorsturen van een enkele bit er zo uit:

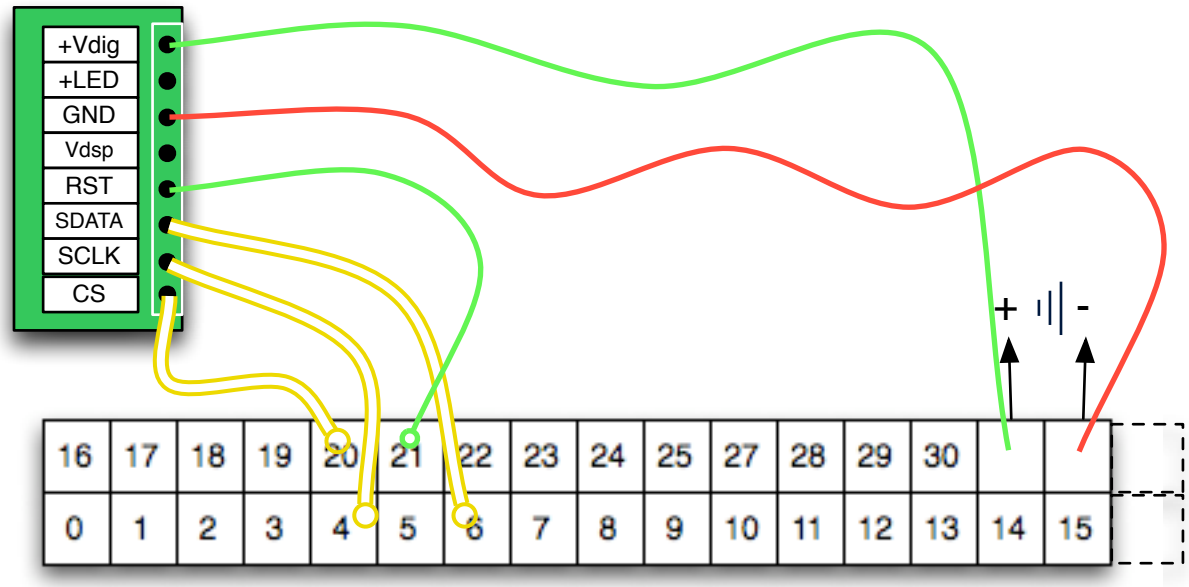
```
(define (send-bit bit)
  (clear-pin sclk)
  (if (eq? bit 0)
    (clear-pin sdata)
    (set-pin sdata))
  (set-pin sclk))
```

Als we een enkele byte willen doorsturen, moeten we specificeren of het om een commando of data gaat. In het protocol doen we dit door de byte voor te gaan met een 0 (commando) of 1 (data).

```
(define (send-byte cmd data)
  (clear-pin cs)
  ;; send the command bit
  (send-bit cmd)
  ;; send the byte
  (let loop ((bitnr 7))
    (if (>= bitnr 0)
      (begin
        (send-bit (get-bit data bitnr))
        (loop (- bitnr 1)))))
  (set-pin cs))
```

Je kan de hele code vinden op de website.

Om de LCD aan te sluiten op het bordje heb je zes pinnen nodig: drie voor de net besproken communicatie, één voor reset en twee voor stroom. Alhoewel je normaal kan kiezen welke pinnen je voor communicatie en reset gebruikt, gaat de ingebakken code van `send-byte` ervan uit dat je voor de volgende pin configuratie gekozen hebt.



RST --> P0.21
SDATA --> P0.06
SCLK --> P0.04
CS --> P0.20

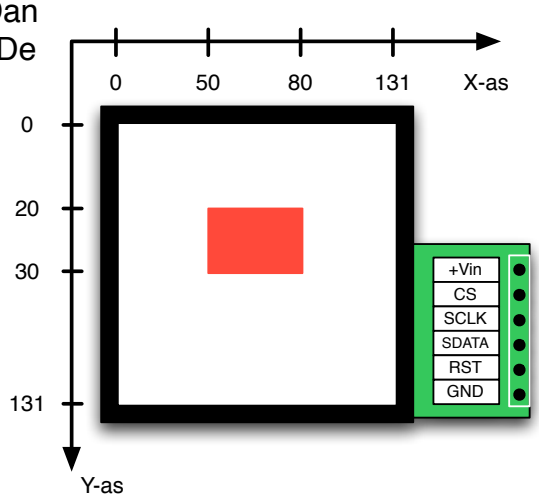
Test of het LCD scherm werkt door de gegeven code in te laden en het volgende op te roepen.

```
> (init-lcd)
```

Dit start het lcd scherm op waarna je een scherm zou moeten krijgen met daarop willekeurige kleuren. In de volgende sectie zullen we hierop tekenen.

3 Tekenen op het LCD scherm

Tekenen naar het LCD scherm kan je maar op één manier doen. Je zegt eerst op welke rechthoekige plaats op het scherm je zal tekenen. Dan zeg je dat je klaar bent om de pixels door te sturen. De volgende code maakt het LCD scherm klaar om een rechthoek te tekenen van 10 pixels hoog en 30 breed.




```
(send-command PASET 20 30)   ; y-coordinaten
(send-command CASET 50 80)   ; x-coordinaten
(send-command RAMWR)         ; ram write
```

De variabelen `PASET` , `CASET` zijn commando's om respectievelijk de rijen pixels en kolommen pixels te selecteren. Het commando `RAMWR` kondigt het zenden van pixels aan.key

Nu verwacht het scherm een reeks pixels in een welbepaalde sequentie. Het vullen van deze rechthoek is strikt van links naar rechts, van boven naar beneden. Je moet exact evenveel pixels doorsturen als er in deze rechthoek zijn. Dit kan je snel nagaan door de breedte te vermenigvuldigen met de hoogte; 10 * 30 = 300 in dit geval.

Om een enkele pixel door te sturen heb je één byte nodig. De LCD is namelijk ingesteld in 8-bit RGB mode, wat je een kleurenpallet geeft van 256 kleuren.

De volgende code zal 300 keer de kleur rood doorsturen.

```
(define colour-red #xe0)
(define (fill num-pixels)
  (if (> num-pixels 0)
    (begin
      (send-data colour-red)
      (fill (- num-pixels 1)))))
(fill (* 10 30))
```

Je bent niet verplicht de rechthoek uniform te kleuren, maar je moet wel een hele rechthoek per keer tekenen.

De LCD code op de website bevat functies die de bovenstaande functionaliteit reeds voor jou implementeert. Met de functie `(fill-rectangle x y width height colour)` kan je rechtstreeks een vlakke rechthoek in een uniforme kleur op het LCD scherm tekenen.

```
> (init-lcd)  ; start the LCD
> (fill-rectangle 50 20 10 30 colour-red)
```

Gebruik `fill-rectangle` om de bovenstaande rode rechthoek van 10 op 30 te tekenen. Je kan heel je scherm schoonvegen door een rechthoek te tekenen die het hele oppervlak bestrijkt met `(fill-rectangle x y width height colour-white)`.

4 Bestanden en Opstarten

De uitgedeelde OLIMEX bordjes bevatten flash geheugen dat kan gebruikt worden om informatie permanent weg te schrijven. Dit kunnen we gebruiken om '*library*' code die je vaak nodig hebt te onthouden tussen resets in.

In deze oefening zullen we een '*boot*' file aanmaken om de LCD code van de website te onthouden.

Je opent een file met de volgende code.

```
> (define boot-file (open-output-file "boot"))
```

Naar een file schrijven doe je op dezelfde manier als in standaard (R5RS) Scheme, met de functies `read` en `write`.

```
> (write '(define test-var 42) boot-file)
```

Nu hebben we deze code symbolisch opgeslagen in de file. Als je klaar bent met schrijven naar een bestand, sluit je dit best.

```
> (close-output-port boot-file)
```

Let op dat we een *output file* gebruikten om te schrijven. Dit zijn files waar je strikt naar schrijft. Als we van dezelfde file willen lezen, moeten we deze openen met `open-output-file`. De volgende code leest de inhoud van de bovenstaande file.

```
> (define boot-file (open-input-file "boot"))
> (read boot-file)
(define test-var 42)
```

Hierna sluiten we de file terug met:

```
> (close-input-port boot-file)
```

Wat we uitgelezen hebben is een lijst van symbolen, deze code is namelijk nog niet geëvalueerd. We evalueren de code met de standaard functie `eval`. Nu zien we dat de ingeladen variabele gekend is.
Let op, na je een file helemaal hebt ingelezen met `read` zal een volgende `read` niets kunnen lezen. Je bent namelijk al op het einde van de file. Om een file terug uit te lezen moet je deze sluiten en heropenen.

```
> (eval (read boot-file))
> test-var
42
> (close-input-port boot-file)
```

Herhaal deze procedure van schrijven en lezen met de LCD code die je op de website vindt. Trek na het schrijven de stekker uit het bordje zodat je zeker bent dat het geheugen gewist is. Laad daarna de file in en test de LCD code door iets op het scherm te tekenen.

nb. Als je verschillende lijnen code naar file schrijft, let dan op dat je dit doet met een begin: (write '(begin ...) my-file)