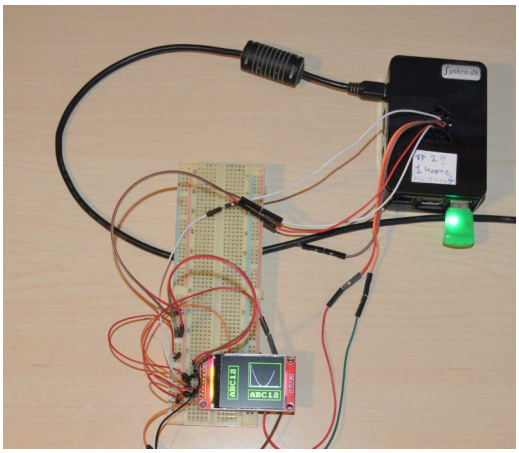
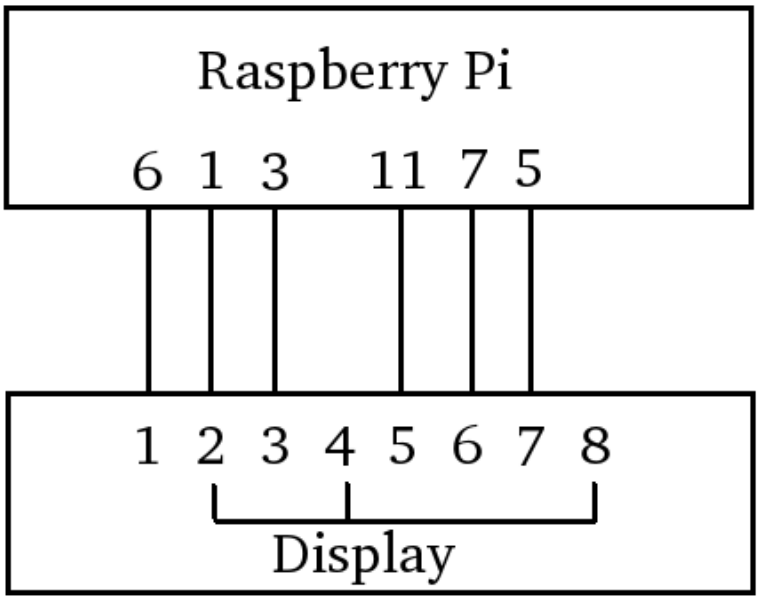
Test af 128x160 display

På en Raspberry Pi





Oversigt: Jeg beskriver hvordan man skriver til displayet og hvordan man får det til det. Jeg demonstrerer i C hvordan man skaber en simpel font, og hvordan man skaber simple grafiske funktioner, som at trække en streg.

Denne og andre kan findes på: https://synkro.dk/bog
DB: Henrik Kressner
Dec 2020

Table of Contents

1. Indledning....2
2. Konstruktionen....3
3. Initialisering....4
4 Hvis noget på displayet....5
Bilag A Kildetekst....6

1. Indledning

Dette er mine foreløbige erfaringer med et 128x160display.

Displayet er udstyret med en Siltronix ST7735S, der udgør interfacet mellem selve displayet og processoren. I displayet er der kun pixel, man skal selv skabe tal, bogstaver og grafik helt fra bunden.

Inspiration til den benyttede font har jeg fået fra en gammel 80’er computer, jeg har ikke interesse i at skabe flotte fonte.

Displayet findes i flere versioner, alt efter leverandør. Det betyder ben konfigurationen og navnene på benene varierer, fra leverandør til leverandør. Mit kommer fra AZDelivery.

Displayet kommer på et print og er også udstyret med et interface til et SD kort, det vil jeg ikke komme ind på i dette notat.

Displayet kan angiveligt ”rulle” (scrolle), det har jeg ikke arbejdet med.

Displayet kan angiveligt kører SPI, det er nu ikke helt problemfrit.

Testprogram (kildetekst) er vist i Bilag A

Jeg skifter mellem at skrive på engelsk og dansk, alt efter hvad jeg finder passende hvor det bliver brugt.

Henrik Kressner
December 2020

2. Konstruktionen

Mit display er leveret på et print med 8 bens interface, jeg har forbundet dem således til en RPi.

Pin display	Pin RPi	Kommentar
1	6	Ground
2	1	Vcc (3.3V) Max er 4,8V
3	3	CS Chip Select aktiv low.
4	1	Reset (Forbind til Vcc)
5	11	AO If = 1 → Der kommer en kommando : If = 0 → Der kommer data
6	7	SDA
7	5	SCK
8	1	LED/Baggrundslys (Forbind til Vcc)

I følge manualen på Siltronix ST7735S skulle det være muligt at sende både parallelt og serielt, til og fra displayet. Det ser ud til det eneste der er implementeret på den udgave jeg har, er det der kaldes ”4-line Serial Interface”, hvilket betyder, at udover Vcc og ground, skal der benyttes SC (Chip enable), AO (data/command flag), SCK (serial clock) og SDA (serial data input til display)

I følge manualen på Siltronix ST7735S skulle det være muligt at læse fra displayet, efter mange test må jeg konkludere det ikke er implenteret i min udgave. Når man forsøger at læse fra enheden, kommer der kun ettaller, uanset om man bruger 3-line Serial Interface eller 4-line Serial Interface.

Rx problem: Kan det skyldes jeg bruger GPIO 2 og 3, det er I2C med 1,8K pull ops ? (Det er CE og SCK, ikke data, så betyder det noget?)

Kan jeg få 3-line Serial Interface til at virke ?
- Der begynder at tegne sig noget, men det ses ikke at være dokumenteret.

Hvad sker der når CS er high, er der noget 3 statet for så kan en bit hvis undgås ?
- Nej, det kaldes pause og ser ud til at være nødvendig, selv om manualen siger dte modsatte.

3. Initialisering

Selv om enheden startes korrekt op, som beskrevet i manualen side 23, skal det initialiseres. Jeg har derfor skrevet en initialseringsfunktion, det er blot manualen side 45 (Initializing by Instruction), skrevet i C.

```
void Reset()
{
	Tx(0x30,0);
	usleep(4500);
	Tx(0x30,0);
	usleep(4500);
	Tx(0x30,0);
	usleep(4500);

	// Display 2 linier, 8 bit, 5x8
	Tx(0x38,0);
	usleep(4500);
	Tx(0x08,0);	// Display off
	usleep(4500);
	Tx(0x01,0);	// Display Clear
	usleep(4500);
	Tx(0x05,0);	// Entry Mode Set
	usleep(4500);
}
```

Tabel 3.1

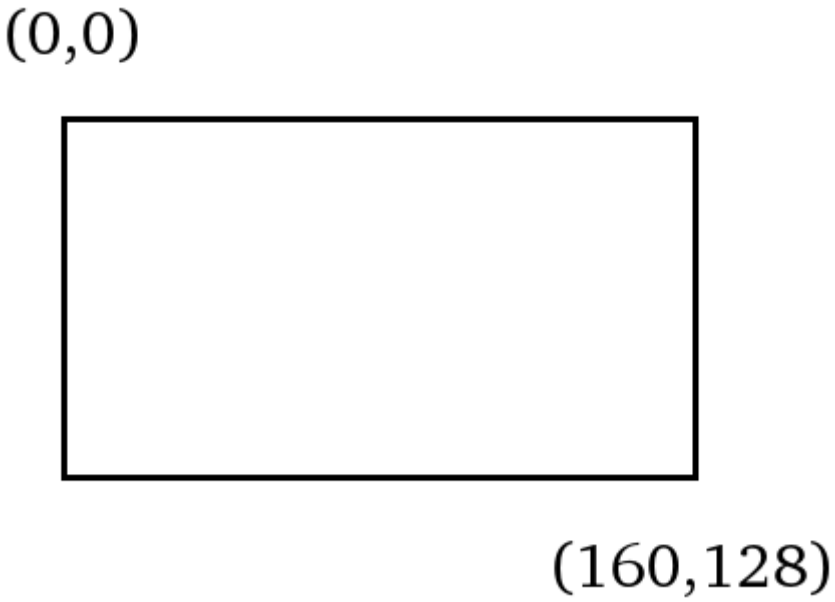
Den sidste instruks (Entry Mode Set) betyder der kan sendes data til displayt.

4 Vis noget på displayet

Alt på displayet foregår i vinduer, som er et sammenhængene dataareal i enhedens hukommelse. Det mindste vindue er en pixel og det største vindue er det samlede display. Hvis man skriver ud over vinduets dataareal, wrappes det omkring og begynder forfra.

Default vindue er det samlede display.

Et vindue defineres med de to kommandoer 0x2A (Column Address Set) og 0x2B (Row Address Set). Begge komandoer tager som argument en startadresse på 16 bit, og en slutadresse på 16 bit. Der er i alt 160 x adresser og 128 y adresser.



For at få noget på displayet, skal der sendes data til det, det gør funktionen Tx:

```
void Tx(char data, int kommando)
```

Hvor data er den byte der skal sendes til displayet og kommando er nul (0) eller et (1), hvor nul betyder det er data, og et betyder der kommer en kommando. For at se hvilke kommandoer der kan sendes, se tabel 15 i afsnit 10.1 på side 104.

Hvis man vil tegne et rektankel på (10,10) til (20,20) kan det gøres ved at kalde SetRektangel således:

```
SetRektangel(10,10,20,20,color);
```

Hvor color er et array af 6 byte der definerer farver, se senere.

Man kan slette displayet efter at have initialiseret det, ved at skrive fra startadresse (nul) til slutadressen.hvis man ønsker en bestemt farve kan det gøre på denne måde

```
for(i=0; i < BREDDE*HOJDE; i++) {
        Tx(0x0,0);      // Roed
        Tx(0x0,0);      // Groen
        Tx(0x0,0);      // Blaa
}
```

Nul er ingenting hviket betyder skærmen bliver sort, hvis man ønsker skærmen hvid skal alle farver sættes til 0xFF o.s.v.

Enheden starter default op i et mode der kræver 6 byt til hver farve, hver af de 6 bit ”sidder” i et 8 bit register. Jeg har derfor lavet en variabel på 6 byte, 3 byte til forgrundsfarven og 3 byte til baggrundsfarven. (De 2 ”øverste” (MSB) i hver byte ignoreres.)

```
char color[6];
```

Forgrundsfarven er de 3 første byte, baggrundsfarven de 3 følgende.

Bilag A Kildetekst

Kildeteksten består af to filer, spi.c og font1.h.

```
// font1.h
// Danner et array med bitstrukturen for alle ASCII tegn
// (c) Synkro Engineering 2020

#if !defined(SIGNS)
#define SIGNS

#define MAXSET = 1024         // 128*8


// Matrix er på 8*8 bit.

      char FONT1[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  // Space
                  0x18, 0x18, 0x18, 0x18, 0x18, 0x00, 0x18, 0x00, // !
                  0x6C, 0x6C, 0x6C, 0X00, 0X00, 0X00, 0X00, 0X00, // "
                  0x6C, 0x6C, 0xFE, 0x6C, 0xFE, 0x6C, 0x6C, 0x00, // #
                  0x18, 0x3E, 0x58, 0x3C, 0x1A, 0x7C, 0x18, 0x00,// $
                  0x00, 0xC6, 0xCC, 0x18, 0x30, 0x66, 0xC6, 0x00,// %
                  0x38, 0x68, 0x38, 0x76, 0xDC, 0xCC, 0x76, 0x00,// &
                  0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,// '
                  0x0C, 0x18, 0x30, 0x30, 0x30, 0x18, 0x0C, 0x00,// (
                  0x30, 0x18, 0x0C, 0x0C, 0x0C, 0x18, 0x30, 0x00,// )
                  0x00, 0x66, 0x3C, 0xFF, 0x3C, 0x66, 0x00, 0x00,// *
                  0x00, 0x18, 0x18, 0x7E, 0x18, 0x18, 0x00, 0x00,// +
                  0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x30,// ,
                  0x00, 0x00, 0x00, 0x7E, 0x00, 0x00, 0x00, 0x00,// -
                  0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00,// .
                  0x06, 0x0C, 0x18, 0x30, 0x60, 0xC0, 0x80, 0x00,// /
                  0x7C, 0xC6, 0xCE, 0xD6, 0xE6, 0xC6, 0x7C, 0x00,// 0
                  0x18, 0x38, 0x18, 0x18, 0x18, 0x18, 0x7E, 0x00,// 1
                  0x3C, 0x66, 0x06, 0x3C, 0x60, 0x66, 0x7E, 0x00,// 2
                  0x3C, 0x46, 0x06, 0x1C, 0x06, 0x66, 0x3C, 0x00,// 3
                  0x18, 0x38, 0x58, 0x98, 0xFE, 0x18, 0x3C, 0x00,// 4
                  0x7E, 0x62, 0x60, 0x3C, 0x06, 0x66, 0x3C, 0x00,// 5
                  0x3C, 0x66, 0x60, 0x3C, 0x66, 0x66, 0x3C, 0x00,// 6
                  0x7C, 0x06, 0x06, 0x0C, 0x18, 0x18, 0x18, 0x00,// 7
                  0x3C, 0x66, 0x66, 0x3C, 0x66, 0x66, 0x3C, 0x00,// 8
                  0x3C, 0x66, 0x66, 0x3E, 0x06, 0x66, 0x3C, 0x00,// 9
                  0x00, 0x00, 0x18, 0x18, 0x00, 0x18, 0x18, 0x00,// :
                  0x00, 0x00, 0x18, 0x18, 0x00, 0x18, 0x18, 0x30,// ;
                  0x0C, 0x18, 0x30, 0x60, 0x30, 0x18, 0x0C, 0x00,// <
                  0x00, 0x00, 0x7E, 0x00, 0x00, 0x7E, 0x00, 0x00,// =
                  0x60, 0x30, 0x18, 0x0C, 0x18, 0x30, 0x60, 0x00,// >
                  0x3C, 0x66, 0x06, 0x0C, 0x18, 0x00, 0x18, 0x00,// ?
                  0x7C, 0xC6, 0xDE, 0xD6, 0xDE, 0xC0, 0x7C, 0x00,// @
                  0x18, 0x3C, 0x66, 0x66, 0x7E, 0x66, 0x66, 0x00,// A
                  0xFC, 0x66, 0x66, 0x7C, 0x66, 0x66, 0xFC, 0x00,// B
                  0x3C, 0x66, 0xC0, 0xC0, 0xC0, 0x66, 0x3C, 0x00,// C
                  0xF8, 0x6C, 0x66, 0x66, 0x66, 0x6C, 0xF8, 0x00,// D
```

```
                    0xFE, 0x62, 0x68, 0x78, 0x68, 0x62, 0xFE, 0x00,// E
                    0xFE, 0x62, 0x68, 0x78, 0x68, 0x60, 0xF0, 0x00,// F
                    0x3C, 0x66, 0xC0, 0xC0, 0xCE, 0xC6, 0x7E, 0x00,// G
                    0x66, 0x66, 0x66, 0x7E, 0x66, 0x66, 0x66, 0x00,// H
                    0x7E, 0x18, 0x18, 0x18, 0x18, 0x18, 0x7E, 0x00,// I
                    0x1E, 0x0C, 0x0C, 0x0c, 0xCC, 0xCC, 0x78, 0x00,// J
                    0xE6, 0x66, 0x6C, 0x78, 0x6C, 0x66, 0xE6, 0x00,// K
                    0xF0, 0x60, 0x60, 0x60, 0x62, 0x66, 0xFE, 0x00,// L
                    0xC6, 0xEE, 0xFE, 0xFE, 0xD6, 0xC6, 0xC6, 0x00,// M
                    0xC6, 0xE6, 0xF6, 0xDE, 0xCE, 0xC6, 0xC6, 0x00,// N
                    0x38, 0x6C, 0xC6, 0xC6, 0xC6, 0x6C, 0x38, 0x00,// O
                    0xFC, 0x66, 0x66, 0x7C, 0x60, 0x60, 0xF0, 0x00,// P
                    0x38, 0x6C, 0xC6, 0xC6, 0xDA, 0xCC, 0x76, 0x00,// Q
                    0xFC, 0x66, 0x66, 0x7C, 0x6C, 0x66, 0xE2, 0x00,// R
                    0x3C, 0x66, 0x60, 0x3C, 0x06, 0x66, 0x3C, 0x00,// S
                    0x7E, 0x5A, 0x18, 0x18, 0x18, 0x18, 0x3C, 0x00,// T
                    0x66, 0x66, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x00,// U
                    0x66, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x18, 0x00,// V
                    0xD6, 0xD6, 0xD6, 0xD6, 0xFE, 0xEE, 0xC6, 0x00,// W
                    0xC6, 0x6C, 0x38, 0x38, 0x6C, 0xC6, 0xC6, 0x00,// X
                    0x66, 0x66, 0x66, 0x3C, 0x18, 0x18, 0x3C, 0x00,// Y
                    0xFE, 0xC6, 0x8C, 0x18, 0x32, 0x66, 0xFE, 0x00,// Z
                    0x3C, 0x30, 0x30, 0x30, 0x30, 0x30, 0x3C, 0x00,// [
                    0xC0, 0x60, 0x30, 0x18, 0x0C, 0x06, 0x02, 0x00,// \
                    0x3C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x3C, 0x00,// ]
                    0x18, 0x3C, 0x7E, 0x18, 0x18, 0x18, 0x18, 0x00,// ^
                    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,// _
                    0x00, 0x30, 0x18, 0x0C, 0x00, 0x00, 0x00, 0x00,// '
                    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //

// Her skal de små bogstaver så komme.

                    0x00, 0x00, 0x78, 0x0C, 0x7C, 0xCC, 0x76, 0x00,// a
                    0xE0, 0x60, 0x7C, 0x66, 0x66, 0x66, 0xBC, 0x00,// b
                    0x00, 0x00, 0x3C, 0x66, 0x60, 0x66, 0x3C, 0x00,// c
                    0x1C, 0x0C, 0x7C, 0xCC, 0xCC, 0xCC, 0x76, 0x00,// d
                    0x00, 0x00, 0x3C, 0x66, 0x7E, 0x60, 0x3C, 0x00,// e
                    0x1C, 0x36, 0x30, 0x78, 0x30, 0x30, 0x78, 0x00,// f
                    0x00, 0x00, 0x3E, 0x66, 0x66, 0x3E, 0x06, 0x7C,// g
                    0xE0, 0x60, 0x6C, 0x76, 0x66, 0x66, 0xE6, 0x00,// h
                    0x18, 0x00, 0x38, 0x18, 0x18, 0x18, 0x3C, 0x00,// i
                    0x02, 0x00, 0x0E, 0x06, 0x06, 0x66, 0x66, 0x3C,// j
                    0xE0, 0x60, 0x66, 0x6C, 0x78, 0x6C, 0x66, 0x00,// k
                    0x38, 0x18, 0x18, 0x18, 0x18, 0x18, 0x3C, 0x00,// l
                    0x00, 0x00, 0x6C, 0xFE, 0xD6, 0xD6, 0xC6, 0x00,// m
                    0x00, 0x00, 0xD8, 0x66, 0x66, 0x66, 0x66, 0x00,// n
                    0x00, 0x00, 0x3C, 0x66, 0x66, 0x66, 0x3C, 0x00,// o
                    0x00, 0x00, 0xDC, 0x66, 0x66, 0x7C, 0x60, 0xF0,// p
                    0x00, 0x00, 0x76, 0xCC, 0xCC, 0x7C, 0x0C, 0x1E,// q
                    0x00, 0x00, 0xD8, 0x6C, 0x60, 0x60, 0xF0, 0x00,// r
                    0x00, 0x00, 0x3C, 0x60, 0x3C, 0x06, 0x7C, 0x00,// s
                    0x00, 0x30, 0x7C, 0x30, 0x30, 0x36, 0x1C, 0x00,// t
                    0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3E, 0x00,// u
                    0x00, 0x00, 0x66, 0x66, 0x66, 0x3C, 0x18, 0x00,// v
                    0x00, 0x00, 0xC6, 0xD6, 0xD6, 0xFE, 0x6C, 0x00,// w
                    0x00, 0x00, 0xC6, 0x6C, 0x38, 0x6C, 0xC6, 0x00,// x
```

```
                0x00, 0x00, 0x66, 0x66, 0x66, 0x3E, 0x06, 0x7C,// y
                0x00, 0x7E, 0x4C, 0x18, 0x30, 0x7E, 0x00, 0x00,// z
                0x0E, 0x18, 0x18, 0x70, 0x18, 0x18, 0x0E, 0x00,// {
                0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x00,// |
                0x70, 0x18, 0x18, 0x0E, 0x18, 0x18, 0x70, 0x00,// }
                0x66, 0x66, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // "
                0x10, 0x38, 0x6C, 0xC6, 0x00, 0x00, 0x00, 0x00, // ^
                0x3C, 0x66, 0x60, 0xF8, 0x60, 0x66, 0xFE, 0x00,// £
                0x3C, 0x44, 0xBA, 0xA2, 0xBA, 0x44, 0x3C, 0x00,// cr
                0x7E, 0xA4, 0x24, 0x24, 0x24, 0x24, 0x24, 0x00,// pi
                0x1E, 0x30, 0x30, 0x7C, 0x30, 0x30, 0xE0, 0x00
};

#endif
```

```
// Filnavn = spi.c
//  Use a RPi 2 with 128x160 pixel SPI display with ST7735? controller Other RPi
//  version, look comment below Done by Henrik Kressner 2020 This is free software,
//  use at your own risc.

// ToDo:


//  To compile (must be root) :  cc spi.c -o spi
//  To run (must be root): spi


#define VERSION "0.29"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
#include <time.h>
#include "font1.h"
#include <math.h>



//  Access from ARM Running Linux
//  For Raspberry Pi 2 and Pi 3, change BCM2708_PERI_BASE to 0x3F000000  ???????
//  My RPi 2 runs with 0x200000 I think RPI3 should run at 0x3F000000 ???
#define BCM2708_PERI_BASE        0x20000000
#define GPIO_BASE                (BCM2708_PERI_BASE + 0x200000) /* GPIO
controller */
#define PAGE_SIZE (4*1024)
#define BLOCK_SIZE (4*1024)

# define CS 2     // CS
# define CLK 3    // SCK
# define DAT 4    // SOA
# define A0 17    // AO
# define BREDDE 129
# define HOJDE 159


int  mem_fd;
char *gpio_mem, *gpio_map;
char *spi0_mem, *spi0_map;


// I/O access
volatile unsigned *gpio;
```

```
// GPIO setup macros. Always use INP_GPIO(x) before using OUT_GPIO(x)
// or SET_GPIO_ALT(x,y)
#define INP_GPIO(g) *(gpio+((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio+((g)/10)) |=  (1<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio+(((g)/10))) |= (((a)<=3?(a)+4:(a)==4?
3:2)<<(((g)%10)*3))

#define GPIO_SET *(gpio+7)  // sets   bits which are 1 ignores bits which are 0
#define GPIO_CLR *(gpio+10) // clears bits which are 1 ignores bits which are 0
#define GET_GPIO(g) (*(gpio+13)&(1<<g)) // 0 if LOW, (1<<g) if HIGH


void setup_io()
{
   /* open /dev/mem */
   if ((mem_fd = open("/dev/mem", O_RDWR|O_SYNC) ) < 0) {
      printf("can't open /dev/mem \n");
      exit (-1);
   }
   /* mmap GPIO */
   // Allocate MAP block
   if ((gpio_mem = malloc(BLOCK_SIZE + (PAGE_SIZE-1))) == NULL) {
      printf("allocation error \n");
      exit (-1);
   }
   // Make sure pointer is on 4K boundary
   if ((unsigned long)gpio_mem % PAGE_SIZE)
     gpio_mem += PAGE_SIZE - ((unsigned long)gpio_mem % PAGE_SIZE);
   // Now map it
   gpio_map = (unsigned char *)mmap(
      (caddr_t)gpio_mem,
      BLOCK_SIZE,
      PROT_READ|PROT_WRITE,
      MAP_SHARED|MAP_FIXED,
      mem_fd,
      GPIO_BASE
   );

   if ((long)gpio_map < 0) {
      printf("mmap error %d\n", (int)gpio_map);
      exit (-1);
   }
   // Always use volatile pointer!
   gpio = (volatile unsigned *)gpio_map;
} // setup_io
```

```
void help()
{
      printf("\nSPI (128x160) display tester %s\n", VERSION);
      printf("");
      printf("\n");
      printf("");
      printf("\n");
      printf("");
      printf("");
      printf("\n");
      printf("DB Henrik Kressner 2020.\n");
}

void Rx(char *data)

// Ser ikke ud til at være mulig
// https://forums.adafruit.com/viewtopic.php?f=47&t=24796

{
      int i;

        INP_GPIO(CS);
        OUT_GPIO(CS);
        INP_GPIO(CLK);
        OUT_GPIO(CLK);
        INP_GPIO(DAT);          // VIGTIGT 3 state
        INP_GPIO(A0);
        OUT_GPIO(A0);

      GPIO_SET = 1<<A0;
        GPIO_CLR = 1<<CS;       // CS Low, tid ligegyldig
      usleep(1);
        for (i=0; i<8; i++)
        {
                if (GET_GPIO(DAT))
                        data[0] = data[0] | 8;
                else
                        data[0] = data[0] | 0;
                data[0] = data[0] >> 1;
                usleep(1);      // Vent 10 nS
                GPIO_SET = 1<<CLK;      // Clock op
                usleep(1);      // vent 60 nS
                GPIO_CLR = 1<<CLK;
        }
        usleep(1);              // Vent 65 nS
        GPIO_SET = 1<<CS;       // CS high
}
```

```
void Tx(char data, int kommando)
{
      int i;

        INP_GPIO(CS);
        OUT_GPIO(CS);
        INP_GPIO(CLK);
        OUT_GPIO(CLK);
      GPIO_CLR = 1<<CLK;
        INP_GPIO(DAT);
        OUT_GPIO(DAT);
        INP_GPIO(A0);
        OUT_GPIO(A0);
      GPIO_CLR = 1<<DAT;      // Skal starte på nul
      GPIO_CLR = 1<<CS; // CS Low, tid ligegyldig
        if (kommando == 1)
            GPIO_CLR = 1<<A0;
      else
            GPIO_SET = 1<<A0;
      for (i=0; i<8; i++)
        {
                if (0x80 & data)
                        GPIO_SET = 1<<DAT;
                else
                        GPIO_CLR = 1<<DAT;

                data = data << 1;
            GPIO_SET = 1<<CLK;      // Clock op
            GPIO_CLR = 1<<CLK;
        }
      GPIO_SET = 1<<CS; // CS high
      INP_GPIO(DAT);          // VIGTIGT 3 state
}

void SetVindu(char x1, char y1, char x2, char y2)
{
        Tx(0x2A,1);       // Coloumn Address Set
        Tx(0,0);
        Tx(x1,0);
        Tx(0,0);
        Tx(x2,0);

        Tx(0x2B,1);     // Row Address Set
        Tx(0,0);
        Tx(y1,0);
        Tx(0,0);
        Tx(y2,0);

       Tx(0x2C,1);     // Memory Write}
}
```

```
void SetPix(char x, char y, char *color)
{
      SetVindu(x,y,1,1);
      Tx(color[0],0);     // Roed
      Tx(color[1],0);     // Groen
        Tx(color[2],0);     // Blaa
}

void SletPix(char x, char y, char *color)
{
        SetVindu(x,y,1,1);
        Tx(color[3],0);     // Roed
        Tx(color[4],0);     // Groen
        Tx(color[5],0);     // Blaa
}

void Streg(char x, char y, char l, char h, char *color)
{
        int i,j;

      if (l>h)    // Vandret
              for(i=x; i < x+l+1; i++)
                  SetPix(i,y,color);
      else        // Lodret
                for(i=y; i < y+h+1; i++)
                  SetPix(x,i,color);
}

void SetRektangel(char x, char y, char l, char h, char *color)
{
        int i,j;

      Streg(x,y,l,1,color);
        Streg(x,y,1,h,color);
        Streg(x+l,y,1,h,color);
        Streg(x,y+h,l,1,color);
}
```

```
void SetTegn90(char tegn, int x, int y, char *color, int skala)
{
        int i,j;
      char filter = 0x80;

      tegn = tegn - 'A' + 33;
        SetVindu(x,y,x+8*skala-1,y+8*skala-1);
        for (j=0; j<8; j++)     {
                for (y=0; y<skala; y++) {
                        for(i=7; i >= 0; i--)  {
                                for (x=0; x<skala; x++)
                                        if (FONT1[i + tegn*8] & filter) {
                                                Tx(color[0],0);     // Roed
                                                Tx(color[1],0);     // Groen
                                                Tx(color[2],0);     // Blaa
                                        }
                                        else  {
                                                Tx(color[3],0);     // Roed
                                                Tx(color[4],0);     // Groen
                                                Tx(color[5],0);     // Blaa
                                        }
                  }  // i
                }  // y
            filter = filter >> 1;
        } // j
}


void SetChar(char tegn, int x, int y, char *color, int skala)
{
        int i,is,j,js;
        char filter;

        tegn = tegn - 'A' + 33;
        for (j=0; j<8; j++)
        {
            for (js=0; js<skala; js++)
            {
            filter = 0x80;
                for(i=0; i < 7; i++)
            {
                    for (is=0; is<skala; is++)
                        if (FONT1[j + tegn*8] & filter)
                             SetPix(x+i*skala+is,y+j*skala+js,color);
                        else
                            SletPix(x+i*skala+is,y+j*skala+js,color);
                    filter = filter >> 1;
            }  // i
            }  // js
        }  // j
}
```

```
void WriteLine(char *s, char x, char y, char *color, char skala)
{
      int i, tmp;

      tmp = strlen(s);
        for (i=0; i<tmp; i++)
          SetChar(s[i],x+i*8*skala,y,color,skala);

}

void WriteLine90(char *s, char x, char y, char *color, char skala)
{
        int i,tmp;

      tmp = strlen(s) - 1;
        for (i=tmp; i>=0; i--)
            SetTegn90(s[i],x,y+i*8*skala,color,skala);
}

void SlaaEnStreg(char x1, char y1, char x2, char y2, char *color)
{
      int a,x,y;

      if ( (x2>x1) && (y2>y1) )     {
            a = (x2-x1)/(y2-y1);
            for (x=0; x<x2-x1; x++)
            {
                  y = x + y1;
                  SetPix(x+x1,y,color);
            }
      } // if
      if (x1 == x2)
            Streg(x1, y1, 1, y2-y1, color);
        if (y1 == y2)
            Streg(x1, y1, x2-x1, 1, color);
}


void Initialiser()
{
        // Init
        Tx(0x01,1);     // Software reset
        usleep(150);
        Tx(0x11,1);     // Sleep out
        usleep(150);
        Tx(0x29,1);     // Display on
        SetVindu(0,0,BREDDE,HOJDE);
}
```

```
void main(int a, char *s[])
{
      int i;
      int SKALA = 2;
      char tmp[] = "ABC12";
      char color[6];

      help();
      setup_io();
      switch (a)
      {
            case 1: printf("0 %s\n",s[a-1]); break;
                case 2: printf("1 %s\n",s[a-1]); break;
                case 3: printf("2 %s\n",s[a-1]); break;
                case 4: printf("3 %s\n",s[a-1]); break;
      }

      Initialiser();

      for(i=0; i < BREDDE*HOJDE; i++)     {
                Tx(0x0,0);    // Roed
                Tx(0x0,0);    // Groen
            Tx(0x0,0);  // Blaa
      }

      color[0] = 0x00; color[1] = 0xFF; color[2] = 0x00;
        color[3] = 0x00; color[4] = 0x00; color[5] = 0x00;

        SetRektangel(15,15,8*SKALA + 75,8*SKALA + 10, color);
      WriteLine(tmp,20,20,color,SKALA);

        SetRektangel(15,70,8*SKALA + 10,8*SKALA + 71, color);
      WriteLine90(tmp,20,74,color, SKALA);

      SetRektangel(50,70,8*SKALA + 50,8*SKALA + 71, color);

      color[0] = 0xFF; color[1] = 0xFF; color[2] = 0xFF;
      SlaaEnStreg(52,75,52,152,color);
        SlaaEnStreg(52,75,102+8,75,color);

      for (i=-30; i<30; i++)
              SlaaEnStreg(52+0.05*i*i, i+110, 53+0.05*i*i, i+111,color);

}
```