#include <avr/pgmspace.h>

//#include <util/delay.h>

#include <stdlib.h>

#include <Wire.h>

#include "SSD1306.h"

SSD1306::SSD1306(int8\_t SCLK, int8\_t DC, int8\_t RST, int8\_t CS) {

cs = CS;

rst = RST;

dc = DC;

sclk = SCLK;

}

// initializer for I2C - we only indicate the reset pin!

SSD1306::SSD1306(int8\_t reset) {

sclk = dc = cs = -1;

rst = reset;

}

void SSD1306::begin(uint8\_t vccstate, uint8\_t i2caddr) {

\_i2caddr = i2caddr;

// set pin directions

// I2C Init

Wire.begin(); // Is this the right place for this?

// Setup reset pin direction (used by both SPI and I2C)

pinMode(rst, OUTPUT);

digitalWrite(rst, HIGH);

// VDD (3.3V) goes high at start, lets just chill for a ms

delay(1);

// bring reset low

digitalWrite(rst, LOW);

// wait 10ms

delay(10);

// bring out of reset

digitalWrite(rst, HIGH);

// turn on VCC (9V?)

#if defined SSD1306\_128\_32

// Init sequence for 128x32 OLED module

ssd1306\_command(SSD1306\_DISPLAYOFF); // 0xAE

ssd1306\_command(SSD1306\_SETDISPLAYCLOCKDIV); // 0xD5

ssd1306\_command(0x80); // the suggested ratio 0x80

ssd1306\_command(SSD1306\_SETMULTIPLEX); // 0xA8

ssd1306\_command(0x1F);

ssd1306\_command(SSD1306\_SETDISPLAYOFFSET); // 0xD3

ssd1306\_command(0x0); // no offset

ssd1306\_command(SSD1306\_SETSTARTLINE | 0x0); // line #0

ssd1306\_command(SSD1306\_CHARGEPUMP); // 0x8D

if (vccstate == SSD1306\_EXTERNALVCC)

{ ssd1306\_command(0x10); }

else

{ ssd1306\_command(0x14); }

ssd1306\_command(SSD1306\_MEMORYMODE); // 0x20

ssd1306\_command(0x00); // 0x0 act like ks0108

ssd1306\_command(SSD1306\_SEGREMAP | 0x1);

ssd1306\_command(SSD1306\_COMSCANDEC);

ssd1306\_command(SSD1306\_SETCOMPINS); // 0xDA

ssd1306\_command(0x02);

ssd1306\_command(SSD1306\_SETCONTRAST); // 0x81

ssd1306\_command(0x8F);

ssd1306\_command(SSD1306\_SETPRECHARGE); // 0xd9

if (vccstate == SSD1306\_EXTERNALVCC)

{ ssd1306\_command(0x22); }

else

{ ssd1306\_command(0xF1); }

ssd1306\_command(SSD1306\_SETVCOMDETECT); // 0xDB

ssd1306\_command(0x40);

ssd1306\_command(SSD1306\_DISPLAYALLON\_RESUME); // 0xA4

ssd1306\_command(SSD1306\_NORMALDISPLAY); // 0xA6

#endif

#if defined SSD1306\_128\_64

// Init sequence for 128x64 OLED module

ssd1306\_command(SSD1306\_DISPLAYOFF); // 0xAE

ssd1306\_command(SSD1306\_SETDISPLAYCLOCKDIV); // 0xD5

ssd1306\_command(0x80); // the suggested ratio 0x80

ssd1306\_command(SSD1306\_SETMULTIPLEX); // 0xA8

ssd1306\_command(0x3F);

ssd1306\_command(SSD1306\_SETDISPLAYOFFSET); // 0xD3

ssd1306\_command(0x0); // no offset

ssd1306\_command(SSD1306\_SETSTARTLINE | 0x0); // line #0

ssd1306\_command(SSD1306\_CHARGEPUMP); // 0x8D

if (vccstate == SSD1306\_EXTERNALVCC)

{ ssd1306\_command(0x10); }

else

{ ssd1306\_command(0x14); }

ssd1306\_command(SSD1306\_MEMORYMODE); // 0x20

ssd1306\_command(0x00); // 0x0 act like ks0108

ssd1306\_command(SSD1306\_SEGREMAP | 0x1);

ssd1306\_command(SSD1306\_COMSCANDEC);

ssd1306\_command(SSD1306\_SETCOMPINS); // 0xDA

ssd1306\_command(0x12);

ssd1306\_command(SSD1306\_SETCONTRAST); // 0x81

if (vccstate == SSD1306\_EXTERNALVCC)

{ ssd1306\_command(0x9F); }

else

{ ssd1306\_command(0xCF); }

ssd1306\_command(SSD1306\_SETPRECHARGE); // 0xd9

if (vccstate == SSD1306\_EXTERNALVCC)

{ ssd1306\_command(0x22); }

else

{ ssd1306\_command(0xF1); }

ssd1306\_command(SSD1306\_SETVCOMDETECT); // 0xDB

ssd1306\_command(0x40);

ssd1306\_command(SSD1306\_DISPLAYALLON\_RESUME); // 0xA4

ssd1306\_command(SSD1306\_NORMALDISPLAY); // 0xA6

#endif

ssd1306\_command(SSD1306\_DISPLAYON);//--turn on oled panel

// clear screen

delay(5);

ssd1306\_command(SSD1306\_SETLOWCOLUMN | 0x0); // low col = 0

ssd1306\_command(SSD1306\_SETHIGHCOLUMN | 0x0); // hi col = 0

ssd1306\_command(SSD1306\_SETSTARTLINE | 0x0); // line #0

for (byte i = 0; i < SSD1306\_LCDHEIGHT / 8; i++) {

// send a bunch of data in one xmission

ssd1306\_command(0xB0 + i);//set page address

ssd1306\_command(0);//set lower column address

ssd1306\_command(0x10);//set higher column address

for(byte j = 0; j < 8; j++){

Wire.beginTransmission(\_i2caddr);

Wire.write(0x40);

for (byte k = 0; k < SSD1306\_LCDWIDTH / 8; k++) {

Wire.write(0);

}

Wire.endTransmission();

}

}

}

void SSD1306::invertDisplay(uint8\_t i) {

if (i) {

ssd1306\_command(SSD1306\_INVERTDISPLAY);

} else {

ssd1306\_command(SSD1306\_NORMALDISPLAY);

}

}

void SSD1306::ssd1306\_command(uint8\_t c) {

// I2C

uint8\_t control = 0x00; // Co = 0, D/C = 0

Wire.beginTransmission(\_i2caddr);

Wire.write(control);

Wire.write(c);

Wire.endTransmission();

}

// startscrollright

// Activate a right handed scroll for rows start through stop

// Hint, the display is 16 rows tall. To scroll the whole display, run:

// display.scrollright(0x00, 0x0F)

void SSD1306::startscrollright(uint8\_t start, uint8\_t stop){

ssd1306\_command(SSD1306\_RIGHT\_HORIZONTAL\_SCROLL);

ssd1306\_command(0X00);

ssd1306\_command(start);

ssd1306\_command(0X00);

ssd1306\_command(stop);

ssd1306\_command(0X01);

ssd1306\_command(0XFF);

ssd1306\_command(SSD1306\_ACTIVATE\_SCROLL);

}

// startscrollleft

// Activate a right handed scroll for rows start through stop

// Hint, the display is 16 rows tall. To scroll the whole display, run:

// display.scrollright(0x00, 0x0F)

void SSD1306::startscrollleft(uint8\_t start, uint8\_t stop){

ssd1306\_command(SSD1306\_LEFT\_HORIZONTAL\_SCROLL);

ssd1306\_command(0X00);

ssd1306\_command(start);

ssd1306\_command(0X00);

ssd1306\_command(stop);

ssd1306\_command(0X01);

ssd1306\_command(0XFF);

ssd1306\_command(SSD1306\_ACTIVATE\_SCROLL);

}

// startscrolldiagright

// Activate a diagonal scroll for rows start through stop

// Hint, the display is 16 rows tall. To scroll the whole display, run:

// display.scrollright(0x00, 0x0F)

void SSD1306::startscrolldiagright(uint8\_t start, uint8\_t stop){

ssd1306\_command(SSD1306\_SET\_VERTICAL\_SCROLL\_AREA);

ssd1306\_command(0X00);

ssd1306\_command(SSD1306\_LCDHEIGHT);

ssd1306\_command(SSD1306\_VERTICAL\_AND\_RIGHT\_HORIZONTAL\_SCROLL);

ssd1306\_command(0X00);

ssd1306\_command(start);

ssd1306\_command(0X00);

ssd1306\_command(stop);

ssd1306\_command(0X01);

ssd1306\_command(SSD1306\_ACTIVATE\_SCROLL);

}

// startscrolldiagleft

// Activate a diagonal scroll for rows start through stop

// Hint, the display is 16 rows tall. To scroll the whole display, run:

// display.scrollright(0x00, 0x0F)

void SSD1306::startscrolldiagleft(uint8\_t start, uint8\_t stop){

ssd1306\_command(SSD1306\_SET\_VERTICAL\_SCROLL\_AREA);

ssd1306\_command(0X00);

ssd1306\_command(SSD1306\_LCDHEIGHT);

ssd1306\_command(SSD1306\_VERTICAL\_AND\_LEFT\_HORIZONTAL\_SCROLL);

ssd1306\_command(0X00);

ssd1306\_command(start);

ssd1306\_command(0X00);

ssd1306\_command(stop);

ssd1306\_command(0X01);

ssd1306\_command(SSD1306\_ACTIVATE\_SCROLL);

}

void SSD1306::stopscroll(void){

ssd1306\_command(SSD1306\_DEACTIVATE\_SCROLL);

}

void SSD1306::ssd1306\_data(uint8\_t c) {

// I2C

uint8\_t control = 0x40; // Co = 0, D/C = 1

Wire.beginTransmission(\_i2caddr);

Wire.write(control);

Wire.write(c);

Wire.endTransmission();

}

void SSD1306::fill(unsigned char dat)

{

unsigned char i,j;

ssd1306\_command(0x00);//set lower column address

ssd1306\_command(0x10);//set higher column address

ssd1306\_command(0xB0);//set page address

#ifdef TWBR

uint8\_t twbrbackup = TWBR;

TWBR = 18; // upgrade to 400KHz!

#endif

for (byte i=0; i<(SSD1306\_LCDHEIGHT/8); i++)

{

// send a bunch of data in one xmission

ssd1306\_command(0xB0 + i);//set page address

ssd1306\_command(0);//set lower column address

ssd1306\_command(0x10);//set higher column address

for(byte j = 0; j < 8; j++){

Wire.beginTransmission(\_i2caddr);

Wire.write(0x40);

for (byte k = 0; k < 16; k++) {

Wire.write(dat);

}

Wire.endTransmission();

}

}

#ifdef TWBR

TWBR = twbrbackup;

#endif

}

void SSD1306::draw8x8(byte\* buffer, uint8\_t x, uint8\_t y)

{

// send a bunch of data in one xmission

ssd1306\_command(0xB0 + y);//set page address

ssd1306\_command(x & 0xf);//set lower column address

ssd1306\_command(0x10 | (x >> 4));//set higher column address

Wire.beginTransmission(\_i2caddr);

Wire.write(0x40);

Wire.write(buffer, 8);

Wire.endTransmission();

}