/\*

u8g\_dev\_ssd1327\_96x96\_gr.c

2-Bit (graylevel) Driver for SSD1327 Controller (OLED Display)

Tested with Seedstudio 96x96 Oled (LY120)

http://www.seeedstudio.com/wiki/index.php?title=Twig\_-\_OLED\_96x96

Universal 8bit Graphics Library

Copyright (c) 2012, olikraus@gmail.com

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,

are permitted provided that the following conditions are met:

\* Redistributions of source code must retain the above copyright notice, this list

of conditions and the following disclaimer.

\* Redistributions in binary form must reproduce the above copyright notice, this

list of conditions and the following disclaimer in the documentation and/or other

materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND

CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,

INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR

CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,

STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF

ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SSD130x Monochrom OLED Controller

SSD131x Character OLED Controller

SSD132x Graylevel OLED Controller

SSD1331 Color OLED Controller

\*/

#include "u8g.h"

#define WIDTH 96

#define HEIGHT 96

#define XOFFSET 8

/\*

http://www.seeedstudio.com/wiki/index.php?title=Twig\_-\_OLED\_96x96

\*/

static const uint8\_t u8g\_dev\_ssd1327\_2bit\_96x96\_init\_seq[] PROGMEM = {

U8G\_ESC\_DLY(10), /\* delay 10 ms \*/

U8G\_ESC\_CS(0), /\* disable chip \*/

U8G\_ESC\_ADR(0), /\* instruction mode \*/

U8G\_ESC\_RST(1), /\* do reset low pulse with (1\*16)+2 milliseconds \*/

U8G\_ESC\_CS(1), /\* enable chip \*/

0x0fd, 0x012, /\* unlock display, usually not required because the display is unlocked after reset \*/

0x0ae, /\* display off, sleep mode \*/

0x0a8, 0x05f, /\* multiplex ratio: 0x05f \* 1/64 duty \*/

0x0a1, 0x000, /\* display start line \*/

0x0a2, 0x060, /\* display offset, shift mapping ram counter \*/

//0x0a2, 0x04c, /\* NHD: display offset, shift mapping ram counter \*/

0x0a0, 0x046, /\* remap configuration, vertical address increment, enable nibble remap (upper nibble is left) \*/

//0x0a0, 0x056, /\* NHD: remap configuration, vertical address increment, enable nibble remap (upper nibble is left) \*/

0x0ab, 0x001, /\* Enable internal VDD regulator (RESET) \*/

0x081, 0x053, /\* contrast, brightness, 0..128, Newhaven: 0x040, LY120 0x053, 0x070 seems also ok \*/

0x0b1, 0x051, /\* phase length \*/

0x0b3, 0x001, /\* set display clock divide ratio/oscillator frequency \*/

0x0b9, /\* use linear lookup table \*/

#if 0

0x0b8, /\* set gray scale table \*/

//0x01, 0x011, 0x022, 0x032, 0x043, 0x054, 0x065, 0x076,

0x01, 0x011, 0x022, 0x032, 0x043, 0x054, 0x077, 0x077, // 4L mode uses 0, 2, 4, 7

#endif

0x0bc, 0x008, /\* pre-charge voltage level \*/

0x0be, 0x007, /\* VCOMH voltage \*/

0x0b6, 0x001, /\* second precharge \*/

0x0d5, 0x062, /\* enable second precharge, internal vsl (bit0 = 0) \*/

#if 0

// the following commands are not used by the SeeedGrayOLED sequence \*/

0x0ad, 0x002, /\* master configuration: disable embedded DC-DC, enable internal VCOMH \*/

0x086, /\* full current range (0x084, 0x085, 0x086) \*/

0x0b2, 0x051, /\* frame frequency (row period) \*/

0x0b4, 0x002, /\* set pre-charge compensation level (not documented in the SDD1325 datasheet, but used in the NHD init seq.) \*/

0x0b0, 0x028, /\* enable pre-charge compensation (not documented in the SDD1325 datasheet, but used in the NHD init seq.) \*/

0x0bf, 0x002|0x00d, /\* VSL voltage level (not documented in the SDD1325 datasheet, but used in the NHD init seq.) \*/

#endif

0x0a5, /\* all pixel on \*/

//0x02e, /\* no scroll (according to SeeedGrayOLED sequence) \*/

0x0af, /\* display on \*/

U8G\_ESC\_DLY(100), /\* delay 100 ms \*/

0x0a4, /\* normal display mode \*/

U8G\_ESC\_DLY(100), /\* delay 100 ms \*/

0x0a5, /\* all pixel on \*/

0x0af, /\* display on \*/

U8G\_ESC\_DLY(100), /\* delay 100 ms \*/

0x0a4, /\* normal display mode \*/

0x015, /\* column address... \*/

0x008, /\* start at column 8, special for the LY120 ??? \*/

0x037, /\* end at column 55, note: there are two pixel in one column \*/

0x075, /\* row address... \*/

0x008,

0x05f,

U8G\_ESC\_ADR(1), /\* data mode \*/

0x000f, 0x000f, 0x0000, 0x0000, 0x000f,0x000f,0x0000,0x0000,

0x000f, 0x000f, 0x0000, 0x0000, 0x000f,0x000f,0x0000,0x0000,

0x000f, 0x000f, 0x0000, 0x0000, 0x000f,0x000f,0x0000,0x0000,

0x000f, 0x000f, 0x0000, 0x0000, 0x000f,0x000f,0x0000,0x0000,

U8G\_ESC\_CS(0), /\* disable chip \*/

U8G\_ESC\_END /\* end of sequence \*/

};

static const uint8\_t u8g\_dev\_ssd1327\_2bit\_96x96\_prepare\_page\_seq[] PROGMEM = {

U8G\_ESC\_ADR(0), /\* instruction mode \*/

U8G\_ESC\_CS(1), /\* enable chip \*/

0x015, /\* column address... \*/

XOFFSET, /\* start at column 8, special for the LY120 ??? \*/

0x037, /\* end at column 55, note: there are two pixel in one column \*/

0x075, /\* row address... \*/

U8G\_ESC\_END /\* end of sequence \*/

};

static void u8g\_dev\_ssd1327\_2bit\_prepare\_page(u8g\_t \*u8g, u8g\_dev\_t \*dev)

{

uint8\_t page = ((u8g\_pb\_t \*)(dev->dev\_mem))->p.page;

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1327\_2bit\_96x96\_prepare\_page\_seq);

page <<= 2;

u8g\_WriteByte(u8g, dev, page); /\* start at the selected page \*/

page += 3;

u8g\_WriteByte(u8g, dev, page); /\* end within the selected page \*/

u8g\_SetAddress(u8g, dev, 1); /\* data mode \*/

}

static void u8g\_dev\_ssd1327\_2bit\_2x\_prepare\_page(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t is\_odd)

{

uint8\_t page = ((u8g\_pb\_t \*)(dev->dev\_mem))->p.page;

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1327\_2bit\_96x96\_prepare\_page\_seq);

page <<= 1;

page += is\_odd;

page <<= 2;

u8g\_WriteByte(u8g, dev, page); /\* start at the selected page \*/

page += 3;

u8g\_WriteByte(u8g, dev, page); /\* end within the selected page \*/

u8g\_SetAddress(u8g, dev, 1); /\* data mode \*/

}

/\* assumes row autoincrement and activated nibble remap \*/

static void u8g\_dev\_ssd1327\_2bit\_write\_4\_pixel(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t left, uint8\_t right)

{

uint8\_t d, tmp, cnt;

static uint8\_t buf[4];

buf[0] = 0;

buf[1] = 0;

buf[2] = 0;

buf[3] = 0;

cnt = 0;

do

{

if ( left == 0 && right == 0 )

break;

d = left;

d &= 3;

d <<= 4;

tmp = right;

tmp &= 3;

d |= tmp;

d <<= 2;

buf[cnt] = d;

left >>= 2;

right >>= 2;

cnt++;

}while ( cnt < 4 );

u8g\_WriteSequence(u8g, dev, 4, buf);

}

static void u8g\_dev\_ssd1327\_2bit\_write\_buffer(u8g\_t \*u8g, u8g\_dev\_t \*dev)

{

uint8\_t cnt, left, right;

uint8\_t \*ptr;

u8g\_pb\_t \*pb = (u8g\_pb\_t \*)(dev->dev\_mem);

cnt = pb->width;

cnt >>= 1;

ptr = pb->buf;

do

{

left = \*ptr++;

right = \*ptr++;

u8g\_dev\_ssd1327\_2bit\_write\_4\_pixel(u8g, dev, left, right);

cnt--;

} while( cnt > 0 );

}

static void u8g\_dev\_ssd1327\_2bit\_2x\_write\_buffer(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t is\_odd)

{

uint8\_t cnt, left, right;

uint8\_t \*ptr;

u8g\_pb\_t \*pb = (u8g\_pb\_t \*)(dev->dev\_mem);

ptr = pb->buf;

cnt = pb->width;

if ( is\_odd )

ptr += cnt;

cnt >>= 1;

do

{

left = \*ptr++;

right = \*ptr++;

u8g\_dev\_ssd1327\_2bit\_write\_4\_pixel(u8g, dev, left, right);

cnt--;

} while( cnt > 0 );

}

uint8\_t u8g\_dev\_ssd1327\_96x96\_gr\_fn(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t msg, void \*arg)

{

switch(msg)

{

case U8G\_DEV\_MSG\_INIT:

u8g\_InitCom(u8g, dev, U8G\_SPI\_CLK\_CYCLE\_300NS);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1327\_2bit\_96x96\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

u8g\_dev\_ssd1327\_2bit\_prepare\_page(u8g, dev);

u8g\_dev\_ssd1327\_2bit\_write\_buffer(u8g, dev);

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_CONTRAST:

u8g\_SetChipSelect(u8g, dev, 1);

u8g\_SetAddress(u8g, dev, 0); /\* instruction mode \*/

u8g\_WriteByte(u8g, dev, 0x081);

u8g\_WriteByte(u8g, dev, (\*(uint8\_t \*)arg) >> 1);

u8g\_SetChipSelect(u8g, dev, 0);

return 1;

}

return u8g\_dev\_pb8v2\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_ssd1327\_96x96\_2x\_gr\_fn(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t msg, void \*arg)

{

switch(msg)

{

case U8G\_DEV\_MSG\_INIT:

u8g\_InitCom(u8g, dev, U8G\_SPI\_CLK\_CYCLE\_300NS);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1327\_2bit\_96x96\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

u8g\_dev\_ssd1327\_2bit\_2x\_prepare\_page(u8g, dev, 0);

u8g\_dev\_ssd1327\_2bit\_2x\_write\_buffer(u8g, dev, 0);

u8g\_dev\_ssd1327\_2bit\_2x\_prepare\_page(u8g, dev, 1);

u8g\_dev\_ssd1327\_2bit\_2x\_write\_buffer(u8g, dev, 1);

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_CONTRAST:

u8g\_SetChipSelect(u8g, dev, 1);

u8g\_SetAddress(u8g, dev, 0); /\* instruction mode \*/

u8g\_WriteByte(u8g, dev, 0x081);

u8g\_WriteByte(u8g, dev, (\*(uint8\_t \*)arg) >> 1);

u8g\_SetChipSelect(u8g, dev, 0);

return 1;

}

return u8g\_dev\_pb16v2\_base\_fn(u8g, dev, msg, arg);

}

U8G\_PB\_DEV(u8g\_dev\_ssd1327\_96x96\_gr\_sw\_spi , WIDTH, HEIGHT, 4, u8g\_dev\_ssd1327\_96x96\_gr\_fn, U8G\_COM\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ssd1327\_96x96\_gr\_hw\_spi , WIDTH, HEIGHT, 4, u8g\_dev\_ssd1327\_96x96\_gr\_fn, U8G\_COM\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ssd1327\_96x96\_gr\_i2c , WIDTH, HEIGHT, 4, u8g\_dev\_ssd1327\_96x96\_gr\_fn, U8G\_COM\_SSD\_I2C);

#define DWIDTH (2\*WIDTH)

uint8\_t u8g\_dev\_ssd1327\_96x96\_2x\_buf[DWIDTH] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_ssd1327\_96x96\_2x\_pb = { {8, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_ssd1327\_96x96\_2x\_buf};

u8g\_dev\_t u8g\_dev\_ssd1327\_96x96\_2x\_gr\_sw\_spi = { u8g\_dev\_ssd1327\_96x96\_2x\_gr\_fn, &u8g\_dev\_ssd1327\_96x96\_2x\_pb, U8G\_COM\_SW\_SPI };

u8g\_dev\_t u8g\_dev\_ssd1327\_96x96\_2x\_gr\_hw\_spi = { u8g\_dev\_ssd1327\_96x96\_2x\_gr\_fn, &u8g\_dev\_ssd1327\_96x96\_2x\_pb, U8G\_COM\_HW\_SPI };

u8g\_dev\_t u8g\_dev\_ssd1327\_96x96\_2x\_gr\_i2c = { u8g\_dev\_ssd1327\_96x96\_2x\_gr\_fn, &u8g\_dev\_ssd1327\_96x96\_2x\_pb, U8G\_COM\_SSD\_I2C };