/\*

u8g\_dev\_st7920\_192x32.c

Universal 8bit Graphics Library

Copyright (c) 2011, 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.

\*/

#include "u8g.h"

#define WIDTH 192

#define HEIGHT 32

/\* init sequence from https://github.com/adafruit/ST7565-LCD/blob/master/ST7565/ST7565.cpp \*/

static const uint8\_t u8g\_dev\_st7920\_192x32\_init\_seq[] PROGMEM = {

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

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

U8G\_ESC\_RST(15), /\* do reset low pulse with (15\*16)+2 milliseconds (=maximum delay)\*/

U8G\_ESC\_DLY(100), /\* 8 Dez 2012: additional delay 100 ms because of reset\*/

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

U8G\_ESC\_DLY(50), /\* delay 50 ms \*/

0x038, /\* 8 Bit interface (DL=1), basic instruction set (RE=0) \*/

0x00c, /\* display on, cursor & blink off; 0x08: all off \*/

0x006, /\* Entry mode: Cursor move to right ,DDRAM address counter (AC) plus 1, no shift \*/

0x002, /\* disable scroll, enable CGRAM adress \*/

0x001, /\* clear RAM, needs 1.6 ms \*/

U8G\_ESC\_DLY(100), /\* delay 10 ms \*/

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

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

};

uint8\_t u8g\_dev\_st7920\_192x32\_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\_400NS);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7920\_192x32\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

uint8\_t y, i;

uint8\_t \*ptr;

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

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

u8g\_SetChipSelect(u8g, dev, 1);

y = pb->p.page\_y0;

ptr = pb->buf;

for( i = 0; i < 8; i ++ )

{

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

u8g\_WriteByte(u8g, dev, 0x03e ); /\* enable extended mode \*/

u8g\_WriteByte(u8g, dev, 0x080 | y ); /\* y pos \*/

u8g\_WriteByte(u8g, dev, 0x080 ); /\* set x pos to 0\*/

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

u8g\_WriteSequence(u8g, dev, WIDTH/8, ptr);

ptr += WIDTH/8;

y++;

}

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

}

return u8g\_dev\_pb8h1\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_st7920\_192x32\_4x\_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\_400NS);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7920\_192x32\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

uint8\_t y, i;

uint8\_t \*ptr;

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

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

u8g\_SetChipSelect(u8g, dev, 1);

y = pb->p.page\_y0;

ptr = pb->buf;

for( i = 0; i < 32; i ++ )

{

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

u8g\_WriteByte(u8g, dev, 0x03e ); /\* enable extended mode \*/

u8g\_WriteByte(u8g, dev, 0x080 | y ); /\* y pos \*/

u8g\_WriteByte(u8g, dev, 0x080 ); /\* set x pos to 0\*/

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

u8g\_WriteSequence(u8g, dev, WIDTH/8, ptr);

ptr += WIDTH/8;

y++;

}

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

}

return u8g\_dev\_pb32h1\_base\_fn(u8g, dev, msg, arg);

}

U8G\_PB\_DEV(u8g\_dev\_st7920\_192x32\_sw\_spi, WIDTH, HEIGHT, 8, u8g\_dev\_st7920\_192x32\_fn, U8G\_COM\_ST7920\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_st7920\_192x32\_hw\_spi, WIDTH, HEIGHT, 8, u8g\_dev\_st7920\_192x32\_fn, U8G\_COM\_ST7920\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_st7920\_192x32\_8bit, WIDTH, HEIGHT, 8, u8g\_dev\_st7920\_192x32\_fn, U8G\_COM\_FAST\_PARALLEL);

#define QWIDTH (WIDTH\*4)

uint8\_t u8g\_dev\_st7920\_192x32\_4x\_buf[QWIDTH] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_st7920\_192x32\_4x\_pb = { {32, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_st7920\_192x32\_4x\_buf};

u8g\_dev\_t u8g\_dev\_st7920\_192x32\_4x\_sw\_spi = { u8g\_dev\_st7920\_192x32\_4x\_fn, &u8g\_dev\_st7920\_192x32\_4x\_pb, U8G\_COM\_ST7920\_SW\_SPI };

u8g\_dev\_t u8g\_dev\_st7920\_192x32\_4x\_hw\_spi = { u8g\_dev\_st7920\_192x32\_4x\_fn, &u8g\_dev\_st7920\_192x32\_4x\_pb, U8G\_COM\_ST7920\_HW\_SPI };

u8g\_dev\_t u8g\_dev\_st7920\_192x32\_4x\_8bit = { u8g\_dev\_st7920\_192x32\_4x\_fn, &u8g\_dev\_st7920\_192x32\_4x\_pb, U8G\_COM\_FAST\_PARALLEL };