/\*

u8g\_dev\_ssd1306\_128x64.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 128

#define HEIGHT 64

#define PAGE\_HEIGHT 8

/\* init sequence adafruit 128x64 OLED (NOT TESTED) \*/

static const uint8\_t u8g\_dev\_ssd1306\_128x64\_adafruit1\_init\_seq[] PROGMEM = {

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 \*/

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

0x0d5, 0x080, /\* clock divide ratio (0x00=1) and oscillator frequency (0x8) \*/

0x0a8, 0x03f, /\* \*/

0x0d3, 0x000, /\* \*/

0x040, /\* start line \*/

0x08d, 0x010, /\* [1] charge pump setting (p62): 0x014 enable, 0x010 disable \*/

0x020, 0x000, /\* \*/

0x0a1, /\* segment remap a0/a1\*/

0x0c8, /\* c0: scan dir normal, c8: reverse \*/

0x0da, 0x012, /\* com pin HW config, sequential com pin config (bit 4), disable left/right remap (bit 5) \*/

0x081, 0x09f, /\* [1] set contrast control \*/

0x0d9, 0x022, /\* [1] pre-charge period 0x022/f1\*/

0x0db, 0x040, /\* vcomh deselect level \*/

0x02e, /\* 2012-05-27: Deactivate scroll \*/

0x0a4, /\* output ram to display \*/

0x0a6, /\* none inverted normal display mode \*/

0x0af, /\* display on \*/

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

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

};

/\* init sequence adafruit 128x64 OLED (NOT TESTED) \*/

static const uint8\_t u8g\_dev\_ssd1306\_128x64\_adafruit2\_init\_seq[] PROGMEM = {

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 \*/

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

0x0d5, 0x080, /\* clock divide ratio (0x00=1) and oscillator frequency (0x8) \*/

0x0a8, 0x03f, /\* \*/

0x0d3, 0x000, /\* \*/

0x040, /\* start line \*/

0x08d, 0x014, /\* [2] charge pump setting (p62): 0x014 enable, 0x010 disable \*/

0x020, 0x000, /\* \*/

0x0a1, /\* segment remap a0/a1\*/

0x0c8, /\* c0: scan dir normal, c8: reverse \*/

0x0da, 0x012, /\* com pin HW config, sequential com pin config (bit 4), disable left/right remap (bit 5) \*/

0x081, 0x0cf, /\* [2] set contrast control \*/

0x0d9, 0x0f1, /\* [2] pre-charge period 0x022/f1\*/

0x0db, 0x040, /\* vcomh deselect level \*/

0x02e, /\* 2012-05-27: Deactivate scroll \*/

0x0a4, /\* output ram to display \*/

0x0a6, /\* none inverted normal display mode \*/

0x0af, /\* display on \*/

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

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

};

/\* init sequence adafruit 128x64 OLED (NOT TESTED), like adafruit3, but with page addressing mode \*/

static const uint8\_t u8g\_dev\_ssd1306\_128x64\_adafruit3\_init\_seq[] PROGMEM = {

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 \*/

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

0x0d5, 0x080, /\* clock divide ratio (0x00=1) and oscillator frequency (0x8) \*/

0x0a8, 0x03f, /\* \*/

0x0d3, 0x000, /\* \*/

0x040, /\* start line \*/

0x08d, 0x014, /\* [2] charge pump setting (p62): 0x014 enable, 0x010 disable \*/

0x020, 0x002, /\* 2012-05-27: page addressing mode \*/

0x0a1, /\* segment remap a0/a1\*/

0x0c8, /\* c0: scan dir normal, c8: reverse \*/

0x0da, 0x012, /\* com pin HW config, sequential com pin config (bit 4), disable left/right remap (bit 5) \*/

0x081, 0x0cf, /\* [2] set contrast control \*/

0x0d9, 0x0f1, /\* [2] pre-charge period 0x022/f1\*/

0x0db, 0x040, /\* vcomh deselect level \*/

0x02e, /\* 2012-05-27: Deactivate scroll \*/

0x0a4, /\* output ram to display \*/

0x0a6, /\* none inverted normal display mode \*/

0x0af, /\* display on \*/

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

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

};

/\* init sequence Univision datasheet (NOT TESTED) \*/

static const uint8\_t u8g\_dev\_ssd1306\_128x64\_univision\_init\_seq[] PROGMEM = {

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 \*/

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

0x0d5, 0x080, /\* clock divide ratio (0x00=1) and oscillator frequency (0x8) \*/

0x0a8, 0x03f, /\* multiplex ratio \*/

0x0d3, 0x000, /\* display offset \*/

0x040, /\* start line \*/

0x08d, 0x010, /\* charge pump setting (p62): 0x014 enable, 0x010 disable \*/

0x0a1, /\* segment remap a0/a1\*/

0x0c8, /\* c0: scan dir normal, c8: reverse \*/

0x0da, 0x012, /\* com pin HW config, sequential com pin config (bit 4), disable left/right remap (bit 5) \*/

0x081, 0x09f, /\* set contrast control \*/

0x0d9, 0x022, /\* pre-charge period \*/

0x0db, 0x040, /\* vcomh deselect level \*/

0x022, 0x000, /\* page addressing mode WRONG: 3 byte cmd! \*/

0x0a4, /\* output ram to display \*/

0x0a6, /\* none inverted normal display mode \*/

0x0af, /\* display on \*/

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

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

};

/\* select one init sequence here \*/

//#define u8g\_dev\_ssd1306\_128x64\_init\_seq u8g\_dev\_ssd1306\_128x64\_univision\_init\_seq

//#define u8g\_dev\_ssd1306\_128x64\_init\_seq u8g\_dev\_ssd1306\_128x64\_adafruit1\_init\_seq

// 26. Apr 2014: in this thead: http://forum.arduino.cc/index.php?topic=234930.msg1696754;topicseen#msg1696754

// it is mentiond, that adafruit2\_init\_seq works better --> this will be used by the ssd1306\_adafruit device

//#define u8g\_dev\_ssd1306\_128x64\_init\_seq u8g\_dev\_ssd1306\_128x64\_adafruit2\_init\_seq

#define u8g\_dev\_ssd1306\_128x64\_init\_seq u8g\_dev\_ssd1306\_128x64\_adafruit3\_init\_seq

static const uint8\_t u8g\_dev\_ssd1306\_128x64\_data\_start[] PROGMEM = {

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

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

0x010, /\* set upper 4 bit of the col adr to 0 \*/

0x000, /\* set lower 4 bit of the col adr to 0 \*/

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

};

/\* the sh1106 is compatible to the ssd1306, but is 132x64. display seems to be centered \*/

static const uint8\_t u8g\_dev\_sh1106\_128x64\_data\_start[] PROGMEM = {

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

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

0x010, /\* set upper 4 bit of the col adr to 0 \*/

0x002, /\* set lower 4 bit of the col adr to 2 (centered display with sh1106) \*/

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

};

static const uint8\_t u8g\_dev\_ssd13xx\_sleep\_on[] PROGMEM = {

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

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

0x0ae, /\* display off \*/

U8G\_ESC\_CS(0), /\* disable chip, bugfix 12 nov 2014 \*/

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

};

static const uint8\_t u8g\_dev\_ssd13xx\_sleep\_off[] PROGMEM = {

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

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

0x0af, /\* display on \*/

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

U8G\_ESC\_CS(0), /\* disable chip, bugfix 12 nov 2014 \*/

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

};

uint8\_t u8g\_dev\_ssd1306\_128x64\_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\_ssd1306\_128x64\_adafruit2\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

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

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1306\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | pb->p.page); /\* select current page (SSD1306) \*/

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

if ( u8g\_pb\_WriteBuffer(pb, u8g, dev) == 0 )

return 0;

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_off);

return 1;

}

return u8g\_dev\_pb8v1\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_ssd1306\_adafruit\_128x64\_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\_ssd1306\_128x64\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

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

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1306\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | pb->p.page); /\* select current page (SSD1306) \*/

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

if ( u8g\_pb\_WriteBuffer(pb, u8g, dev) == 0 )

return 0;

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_off);

return 1;

}

return u8g\_dev\_pb8v1\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_sh1106\_128x64\_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\_ssd1306\_128x64\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

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

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_sh1106\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | pb->p.page); /\* select current page (SSD1306) \*/

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

if ( u8g\_pb\_WriteBuffer(pb, u8g, dev) == 0 )

return 0;

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_off);

return 1;

}

return u8g\_dev\_pb8v1\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_ssd1306\_128x64\_2x\_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\_ssd1306\_128x64\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

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

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1306\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | (pb->p.page\*2)); /\* select current page (SSD1306) \*/

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

u8g\_WriteSequence(u8g, dev, pb->width, pb->buf);

u8g\_SetChipSelect(u8g, dev, 0);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd1306\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | (pb->p.page\*2+1)); /\* select current page (SSD1306) \*/

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

u8g\_WriteSequence(u8g, dev, pb->width, (uint8\_t \*)(pb->buf)+pb->width);

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_off);

return 1;

}

return u8g\_dev\_pb16v1\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_sh1106\_128x64\_2x\_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\_ssd1306\_128x64\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

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

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_sh1106\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | (pb->p.page\*2)); /\* select current page (SSD1306) \*/

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

u8g\_WriteSequence(u8g, dev, pb->width, pb->buf);

u8g\_SetChipSelect(u8g, dev, 0);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_sh1106\_128x64\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | (pb->p.page\*2+1)); /\* select current page (SSD1306) \*/

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

u8g\_WriteSequence(u8g, dev, pb->width, (uint8\_t \*)(pb->buf)+pb->width);

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_ssd13xx\_sleep\_off);

return 1;

}

return u8g\_dev\_pb16v1\_base\_fn(u8g, dev, msg, arg);

}

U8G\_PB\_DEV(u8g\_dev\_ssd1306\_128x64\_sw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ssd1306\_128x64\_fn, U8G\_COM\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ssd1306\_128x64\_hw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ssd1306\_128x64\_fn, U8G\_COM\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ssd1306\_128x64\_i2c, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ssd1306\_128x64\_fn, U8G\_COM\_SSD\_I2C);

U8G\_PB\_DEV(u8g\_dev\_ssd1306\_adafruit\_128x64\_sw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ssd1306\_adafruit\_128x64\_fn, U8G\_COM\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ssd1306\_adafruit\_128x64\_hw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ssd1306\_adafruit\_128x64\_fn, U8G\_COM\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_ssd1306\_adafruit\_128x64\_i2c, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_ssd1306\_adafruit\_128x64\_fn, U8G\_COM\_SSD\_I2C);

uint8\_t u8g\_dev\_ssd1306\_128x64\_2x\_buf[WIDTH\*2] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_ssd1306\_128x64\_2x\_pb = { {16, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_ssd1306\_128x64\_2x\_buf};

u8g\_dev\_t u8g\_dev\_ssd1306\_128x64\_2x\_sw\_spi = { u8g\_dev\_ssd1306\_128x64\_2x\_fn, &u8g\_dev\_ssd1306\_128x64\_2x\_pb, U8G\_COM\_SW\_SPI };

u8g\_dev\_t u8g\_dev\_ssd1306\_128x64\_2x\_hw\_spi = { u8g\_dev\_ssd1306\_128x64\_2x\_fn, &u8g\_dev\_ssd1306\_128x64\_2x\_pb, U8G\_COM\_HW\_SPI };

u8g\_dev\_t u8g\_dev\_ssd1306\_128x64\_2x\_i2c = { u8g\_dev\_ssd1306\_128x64\_2x\_fn, &u8g\_dev\_ssd1306\_128x64\_2x\_pb, U8G\_COM\_SSD\_I2C };

U8G\_PB\_DEV(u8g\_dev\_sh1106\_128x64\_sw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_sh1106\_128x64\_fn, U8G\_COM\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_sh1106\_128x64\_hw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_sh1106\_128x64\_fn, U8G\_COM\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_sh1106\_128x64\_i2c, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_sh1106\_128x64\_fn, U8G\_COM\_SSD\_I2C);

uint8\_t u8g\_dev\_sh1106\_128x64\_2x\_buf[WIDTH\*2] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_sh1106\_128x64\_2x\_pb = { {16, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_sh1106\_128x64\_2x\_buf};

u8g\_dev\_t u8g\_dev\_sh1106\_128x64\_2x\_sw\_spi = { u8g\_dev\_sh1106\_128x64\_2x\_fn, &u8g\_dev\_sh1106\_128x64\_2x\_pb, U8G\_COM\_SW\_SPI };

u8g\_dev\_t u8g\_dev\_sh1106\_128x64\_2x\_hw\_spi = { u8g\_dev\_sh1106\_128x64\_2x\_fn, &u8g\_dev\_sh1106\_128x64\_2x\_pb, U8G\_COM\_HW\_SPI };

u8g\_dev\_t u8g\_dev\_sh1106\_128x64\_2x\_i2c = { u8g\_dev\_sh1106\_128x64\_2x\_fn, &u8g\_dev\_sh1106\_128x64\_2x\_pb, U8G\_COM\_SSD\_I2C };