/\*

u8g\_dev\_st7565\_dogm128.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

const uint8\_t u8g\_dev\_st7565\_dogm128\_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 \*/

0x040, /\* set display start line \*/

0x0a1, /\* ADC set to reverse \*/

0x0c0, /\* common output mode: set scan direction normal operation \*/

0x0a6, /\* display normal (none reverse) \*/

0x0a2, /\* LCD bias 1/9 \*/

0x02f, /\* all power control circuits on \*/

0x0f8, /\* set booster ratio to \*/

0x000, /\* 4x \*/

0x027, /\* set V0 voltage resistor ratio to large \*/

0x081, /\* set contrast \*/

0x018, /\* contrast value, EA default: 0x016 \*/

0x0ac, /\* indicator \*/

0x000, /\* disable \*/

0x0a4, /\* normal display (not all on) \*/

0x0af, /\* display on \*/

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

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

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

};

static const uint8\_t u8g\_dev\_st7565\_dogm128\_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 \*/

};

static const uint8\_t u8g\_dev\_st7565\_dogm128\_sleep\_on[] PROGMEM = {

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

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

0x0ac, /\* static indicator off \*/

0x000, /\* indicator register set (not sure if this is required) \*/

0x0ae, /\* display off \*/

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

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

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

};

static const uint8\_t u8g\_dev\_st7565\_dogm128\_sleep\_off[] PROGMEM = {

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

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

0x0a4, /\* all points off \*/

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\_st7565\_dogm128\_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\_st7565\_dogm128\_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\_st7565\_dogm128\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | pb->p.page); /\* select current page (ST7565R) \*/

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\_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) >> 2);

u8g\_SetChipSelect(u8g, dev, 0);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7565\_dogm128\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7565\_dogm128\_sleep\_off);

return 1;

}

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

}

uint8\_t u8g\_dev\_st7565\_dogm128\_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\_400NS);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7565\_dogm128\_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\_st7565\_dogm128\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | (2\*pb->p.page)); /\* select current page (ST7565R) \*/

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\_st7565\_dogm128\_data\_start);

u8g\_WriteByte(u8g, dev, 0x0b0 | (2\*pb->p.page+1)); /\* select current page (ST7565R) \*/

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\_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) >> 2);

u8g\_SetChipSelect(u8g, dev, 0);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_ON:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7565\_dogm128\_sleep\_on);

return 1;

case U8G\_DEV\_MSG\_SLEEP\_OFF:

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_st7565\_dogm128\_sleep\_off);

return 1;

}

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

}

U8G\_PB\_DEV(u8g\_dev\_st7565\_dogm128\_sw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_st7565\_dogm128\_fn, U8G\_COM\_SW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_st7565\_dogm128\_hw\_spi, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_st7565\_dogm128\_fn, U8G\_COM\_HW\_SPI);

U8G\_PB\_DEV(u8g\_dev\_st7565\_dogm128\_parallel, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_st7565\_dogm128\_fn, U8G\_COM\_PARALLEL);

uint8\_t u8g\_dev\_st7565\_dogm128\_2x\_buf[WIDTH\*2] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_st7565\_dogm128\_2x\_pb = { {16, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_st7565\_dogm128\_2x\_buf};

u8g\_dev\_t u8g\_dev\_st7565\_dogm128\_2x\_sw\_spi = { u8g\_dev\_st7565\_dogm128\_2x\_fn, &u8g\_dev\_st7565\_dogm128\_2x\_pb, U8G\_COM\_SW\_SPI };

u8g\_dev\_t u8g\_dev\_st7565\_dogm128\_2x\_hw\_spi = { u8g\_dev\_st7565\_dogm128\_2x\_fn, &u8g\_dev\_st7565\_dogm128\_2x\_pb, U8G\_COM\_HW\_SPI };

u8g\_dev\_t u8g\_dev\_st7565\_dogm128\_2x\_parallel = { u8g\_dev\_st7565\_dogm128\_2x\_fn, &u8g\_dev\_st7565\_dogm128\_2x\_pb, U8G\_COM\_PARALLEL };