/\*

u8g\_dev\_t6963\_128x64.c

Tested with Varitronix MGLS240128TZ

Universal 8bit Graphics Library

Copyright (c) 2013, 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.

Application Notes for the MGLS 240x128

www.baso.no/content/pdf/T6963C\_Application.pdf

Hitachi App Notes:

https://www.sparkfun.com/datasheets/LCD/Monochrome/AN-029-Toshiba\_T6963C.pdf

Notes:

The font selection pins should generate the 8x8 font.

For the MGLS240128TZ only FS1 is available on pin 18.

FS1 must be low to generate the 8x8 font.

\*/

#include "u8g.h"

#define WIDTH 128

#define HEIGHT 64

#define PAGE\_HEIGHT 16

/\* text is not used, so settings are not relevant \*/

static const uint8\_t u8g\_dev\_t6963\_128x64\_init\_seq[] PROGMEM = {

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

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

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

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

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

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

0x000, /\* low byte \*/

0x000, /\* height byte \*/

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

0x021, /\* set cursor position \*/

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

0x000, /\* low byte \*/

0x000, /\* height byte \*/

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

0x022, /\* set offset \*/

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

0x000, /\* low byte \*/

0x000, /\* height byte \*/

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

0x040, /\* text home \*/

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

WIDTH/8, /\* low byte \*/

0x000, /\* height byte \*/

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

0x041, /\* text columns \*/

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

0x000, /\* low byte \*/

0x000, /\* height byte \*/

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

0x042, /\* graphics home \*/

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

WIDTH/8, /\* low byte \*/

0x000, /\* height byte \*/

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

0x043, /\* graphics columns \*/

// mode set

// 0x080: Internal CG, OR Mode

// 0x081: Internal CG, EXOR Mode

// 0x083: Internal CG, AND Mode

// 0x088: External CG, OR Mode

// 0x089: External CG, EXOR Mode

// 0x08B: External CG, AND Mode

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

0x080, /\* mode register: OR Mode, Internal Character Mode \*/

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

// display mode

// 0x090: Display off

// 0x094: Graphic off, text on, cursor off, blink off

// 0x096: Graphic off, text on, cursor on, blink off

// 0x097: Graphic off, text on, cursor on, blink on

// 0x098: Graphic on, text off, cursor off, blink off

// 0x09a: Graphic on, text off, cursor on, blink off

// ...

// 0x09c: Graphic on, text on, cursor off, blink off

// 0x09f: Graphic on, text on, cursor on, blink on

0x098, /\* mode register: Display Mode, Graphics on, Text off, Cursor off \*/

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

0x000, /\* low byte \*/

0x000, /\* height byte \*/

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

0x024, /\* set adr pointer \*/

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

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

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

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

};

uint8\_t u8g\_dev\_t6963\_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\_NONE);

u8g\_WriteEscSeqP(u8g, dev, u8g\_dev\_t6963\_128x64\_init\_seq);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

uint8\_t y, i;

uint16\_t disp\_ram\_adr;

uint8\_t \*ptr;

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

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

u8g\_SetChipSelect(u8g, dev, 1);

y = pb->p.page\_y0;

ptr = pb->buf;

disp\_ram\_adr = WIDTH/8;

disp\_ram\_adr \*= y;

for( i = 0; i < PAGE\_HEIGHT; i ++ )

{

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

u8g\_WriteByte(u8g, dev, disp\_ram\_adr&255 ); /\* address low byte \*/

u8g\_WriteByte(u8g, dev, disp\_ram\_adr>>8 ); /\* address hight byte \*/

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

u8g\_WriteByte(u8g, dev, 0x024 ); /\* set adr ptr \*/

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

ptr += WIDTH/8;

disp\_ram\_adr += WIDTH/8;

}

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

u8g\_SetChipSelect(u8g, dev, 0);

}

break;

}

return u8g\_dev\_pb16h1\_base\_fn(u8g, dev, msg, arg);

}

// U8G\_PB\_DEV(u8g\_dev\_t6963\_128x64\_8bit, WIDTH, HEIGHT, PAGE\_HEIGHT, u8g\_dev\_t6963\_128x64\_fn, U8G\_COM\_T6963);

uint8\_t u8g\_dev\_t6963\_128x64\_2x\_bw\_buf[WIDTH/8\*PAGE\_HEIGHT] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_t6963\_128x64\_2x\_bw\_pb = { {PAGE\_HEIGHT, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_t6963\_128x64\_2x\_bw\_buf};

u8g\_dev\_t u8g\_dev\_t6963\_128x64\_8bit = { u8g\_dev\_t6963\_128x64\_fn, &u8g\_dev\_t6963\_128x64\_2x\_bw\_pb, U8G\_COM\_T6963 };