/\*

u8g\_dev\_ht1632.c

1-Bit (BW) Driver for HT1632 controller

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.

U8G\_PIN\_NONE can be used as argument

uint8\_t u8g\_InitSPI(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t sck, uint8\_t mosi, uint8\_t cs, uint8\_t a0, uint8\_t reset)

{

...

u8g->pin\_list[U8G\_PI\_SCK] = sck;

u8g->pin\_list[U8G\_PI\_MOSI] = mosi;

u8g->pin\_list[U8G\_PI\_CS] = cs;

u8g->pin\_list[U8G\_PI\_A0] = a0;

u8g->pin\_list[U8G\_PI\_RESET] = reset;

mapping

#define DATA\_PIN --> U8G\_PI\_MOSI

#define WR\_PIN --> U8G\_PI\_SCK

#define CS\_PIN --> U8G\_PI\_CS

U8G\_PI\_A0 --> not used

U8G\_PI\_RESET --> not used

Usage:

u8g\_InitSPI(&u8g, &u8g\_dev\_ht1632\_24x16, WR\_PIN, DATA\_IN, CS\_PIN, U8G\_PIN\_NONE, U8G\_PIN\_NONE)

\*/

#include "u8g.h"

#define WIDTH 24

#define HEIGHT 16

#define PAGE\_HEIGHT 16

/\* http://forum.arduino.cc/index.php?topic=168537.0 \*/

#define HT1632\_CMD\_SYSDIS 0x00 // CMD= 0000-0000-x Turn off oscil

#define HT1632\_CMD\_SYSON 0x01 // CMD= 0000-0001-x Enable system oscil

#define HT1632\_CMD\_LEDOFF 0x02 // CMD= 0000-0010-x LED duty cycle gen off

#define HT1632\_CMD\_LEDON 0x03 // CMD= 0000-0011-x LEDs ON

#define HT1632\_CMD\_BLOFF 0x08 // CMD= 0000-1000-x Blink OFF

#define HT1632\_CMD\_BLON 0x09 // CMD= 0000-1001-x Blink On

#define HT1632\_CMD\_SLVMD 0x10 // CMD= 0001-00xx-x Slave Mode

#define HT1632\_CMD\_MSTMD 0x14 // CMD= 0001-01xx-x Master Mode

#define HT1632\_CMD\_RCCLK 0x18 // CMD= 0001-10xx-x Use on-chip clock

#define HT1632\_CMD\_EXTCLK 0x1C // CMD= 0001-11xx-x Use external clock

#define HT1632\_CMD\_COMS00 0x20 // CMD= 0010-ABxx-x commons options

#define HT1632\_CMD\_COMS01 0x24 // CMD= 0010-ABxx-x commons options

#define HT1632\_CMD\_COMS10 0x28 // CMD= 0010-ABxx-x commons options

#define HT1632\_CMD\_COMS11 0x2C // P-MOS OUTPUT AND 16COMMON OPTION

#define HT1632\_CMD\_PWM 0xA0 // CMD= 101x-PPPP-x PWM duty cycle

#define HT1632\_ID\_CMD 4 /\* ID = 100 - Commands \*/

#define HT1632\_ID\_RD 6 /\* ID = 110 - Read RAM \*/

#define HT1632\_ID\_WR 5 /\* ID = 101 - Write RAM \*/

#define HT1632\_ID\_LEN 3 // IDs are 3 bits

#define HT1632\_CMD\_LEN 8 // CMDs are 8 bits

#define HT1632\_DATA\_LEN 8 // Data are 4\*2 bits

#define HT1632\_ADDR\_LEN 7 // Address are 7 bits

#if defined(ARDUINO)

#if ARDUINO < 100

#include <WProgram.h>

#else

#include <Arduino.h>

#endif

//#define WR\_PIN 3

//#define DATA\_PIN 2

//#define CS\_PIN 4

void ht1632\_write\_data\_MSB(u8g\_t \*u8g, uint8\_t cnt, uint8\_t data, uint8\_t extra)

{

int8\_t i;

uint8\_t data\_pin = u8g->pin\_list[U8G\_PI\_MOSI];

uint8\_t wr\_pin = u8g->pin\_list[U8G\_PI\_SCK];

for(i = cnt - 1; i >= 0; i--)

{

if ((data >> i) & 1)

{

digitalWrite(data\_pin, HIGH);

}

else

{

digitalWrite(data\_pin, LOW);

}

digitalWrite(wr\_pin, LOW);

u8g\_MicroDelay();

digitalWrite(wr\_pin, HIGH);

u8g\_MicroDelay();

}

// Send an extra bit

if (extra)

{

digitalWrite(data\_pin, HIGH);

digitalWrite(wr\_pin, LOW);

u8g\_MicroDelay();

digitalWrite(wr\_pin, HIGH);

u8g\_MicroDelay();

}

}

void ht1632\_write\_data(u8g\_t \*u8g, uint8\_t cnt, uint8\_t data)

{

uint8\_t i;

uint8\_t data\_pin = u8g->pin\_list[U8G\_PI\_MOSI];

uint8\_t wr\_pin = u8g->pin\_list[U8G\_PI\_SCK];

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

{

if ((data >> i) & 1) {

digitalWrite(data\_pin, HIGH);

}

else {

digitalWrite(data\_pin, LOW);

}

digitalWrite(wr\_pin, LOW);

u8g\_MicroDelay();

digitalWrite(wr\_pin, HIGH);

u8g\_MicroDelay();

}

}

void ht1632\_init(u8g\_t \*u8g)

{

//uint8\_t i;

uint8\_t data\_pin = u8g->pin\_list[U8G\_PI\_MOSI];

uint8\_t wr\_pin = u8g->pin\_list[U8G\_PI\_SCK];

uint8\_t cs\_pin = u8g->pin\_list[U8G\_PI\_CS];

pinMode(data\_pin, OUTPUT);

pinMode(wr\_pin, OUTPUT);

pinMode(cs\_pin, OUTPUT);

digitalWrite(data\_pin, HIGH);

digitalWrite(wr\_pin, HIGH);

digitalWrite(cs\_pin, HIGH);

digitalWrite(cs\_pin, LOW);

/\* init display once after startup \*/

ht1632\_write\_data\_MSB(u8g, 3, HT1632\_ID\_CMD, false); // IDs are 3 bits

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_SYSDIS, true); // 8 bits

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_SYSON, true); // 8 bits

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_COMS11, true); // 8 bits

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_LEDON, true); // 8 bits

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_BLOFF, true); // 8 bits

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_PWM+15, true); // 8 bits

digitalWrite(cs\_pin, HIGH);

/\* removed following (debug) code \*/

/\*

digitalWrite(cs\_pin, LOW);

ht1632\_write\_data\_MSB(u8g, 3, HT1632\_ID\_WR, false); // Send "write to display" command

ht1632\_write\_data\_MSB(u8g, 7, 0, false);

for(i = 0; i<48; ++i)

{

ht1632\_write\_data(u8g, 8, 0xFF);

}

digitalWrite(cs\_pin, HIGH);

\*/

}

/\*

page: 0=data contain lines 0..16, 1=data contain lines 16..32 (a 24x16 display will only have page 0)

cnt: width of the display

data: pointer to a buffer with 2\*cnt bytes.

\*/

void ht1632\_transfer\_data(u8g\_t \*u8g, uint8\_t page, uint8\_t cnt, uint8\_t \*data)

{

uint8\_t addr;

uint8\_t cs\_pin = u8g->pin\_list[U8G\_PI\_CS];

/\* send data to the ht1632 \*/

digitalWrite(cs\_pin, LOW);

ht1632\_write\_data\_MSB(u8g, 3, HT1632\_ID\_WR, false); // Send "write to display" command

ht1632\_write\_data\_MSB(u8g, 7, page\*2\*cnt, false);

// Operating in progressive addressing mode

for (addr = 0; addr < cnt; addr++)

{

ht1632\_write\_data(u8g, 8, data[addr]);

ht1632\_write\_data(u8g, 8, data[addr+cnt]);

}

digitalWrite(cs\_pin, HIGH);

}

/\* value is between 0...15 \*/

void ht1632\_set\_contrast(u8g\_t \*u8g, uint8\_t value)

{

uint8\_t cs\_pin = u8g->pin\_list[U8G\_PI\_CS];

digitalWrite(cs\_pin, LOW);

ht1632\_write\_data\_MSB(u8g, 3, HT1632\_ID\_CMD, false);

ht1632\_write\_data\_MSB(u8g, 8, HT1632\_CMD\_PWM + value, false);

digitalWrite(cs\_pin, HIGH);

}

#else

void ht1632\_init(u8g\_t \*u8g)

{

}

void ht1632\_transfer\_data(u8g\_t \*u8g, uint8\_t page, uint8\_t cnt, uint8\_t \*data)

{

}

void ht1632\_set\_contrast(u8g\_t \*u8g, uint8\_t value)

{

}

#endif /\* ARDUINO \*/

uint8\_t u8g\_dev\_ht1632\_24x16\_fn(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t msg, void \*arg)

{

switch(msg)

{

case U8G\_DEV\_MSG\_INIT:

ht1632\_init(u8g);

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

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

/\* current page: pb->p.page \*/

/\* ptr to the buffer: pb->buf \*/

ht1632\_transfer\_data(u8g, pb->p.page, WIDTH, pb->buf);

}

break;

case U8G\_DEV\_MSG\_CONTRAST:

/\* values passed to SetContrast() are between 0 and 255, scale down to 0...15 \*/

ht1632\_set\_contrast(u8g, (\*(uint8\_t \*)arg) >> 4);

return 1;

}

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

}

uint8\_t u8g\_dev\_ht1632\_24x16\_buf[WIDTH\*2] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_ht1632\_24x16\_pb = { {16, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_ht1632\_24x16\_buf};

u8g\_dev\_t u8g\_dev\_ht1632\_24x16 = { u8g\_dev\_ht1632\_24x16\_fn, &u8g\_dev\_ht1632\_24x16\_pb, u8g\_com\_null\_fn };