/\*

u8g\_com\_arduino\_hw\_usart\_spi.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.

SPI Clock Cycle Type

SSD1351 50ns 20 MHz

SSD1322 300ns 3.3 MHz

SSD1327 300ns

SSD1306 300ns

ST7565 400ns 2.5 MHz

ST7920 400ns

\*/

#include "u8g.h"

#if defined(ARDUINO)

#if defined(\_\_AVR\_ATmega32U4\_\_ )

#include <avr/interrupt.h>

#include <avr/io.h>

#if ARDUINO < 100

#include <WProgram.h>

#else

#include <Arduino.h>

#endif

static uint8\_t u8g\_usart\_spi\_out(uint8\_t data)

{

/\* send data \*/

UDR1 = data;

/\* wait for empty transmit buffer \*/

while(!(UCSR1A & (1 << UDRE1)));

return UDR1;

}

uint8\_t u8g\_com\_arduino\_hw\_usart\_spi\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

switch(msg)

{

case U8G\_COM\_MSG\_STOP:

break;

case U8G\_COM\_MSG\_INIT:

/\* SCK is already an output as we overwrite TXLED \*/

u8g\_com\_arduino\_assign\_pin\_output\_high(u8g);

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS, HIGH);

// Init interface at 2MHz

UBRR1 = 0x00;

UCSR1C = (1 << UMSEL11) | (1 << UMSEL10);

UCSR1B = (1 << TXEN1);

UBRR1 = 3;

break;

case U8G\_COM\_MSG\_ADDRESS: /\* define cmd (arg\_val = 0) or data mode (arg\_val = 1) \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, arg\_val);

break;

case U8G\_COM\_MSG\_CHIP\_SELECT:

if ( arg\_val == 0 )

{

/\* disable \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS, HIGH);

}

else

{

/\* enable \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS, LOW);

}

break;

case U8G\_COM\_MSG\_RESET:

if ( u8g->pin\_list[U8G\_PI\_RESET] != U8G\_PIN\_NONE )

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_RESET, arg\_val);

break;

case U8G\_COM\_MSG\_WRITE\_BYTE:

u8g\_usart\_spi\_out(arg\_val);

break;

case U8G\_COM\_MSG\_WRITE\_SEQ:

{

register uint8\_t \*ptr = arg\_ptr;

while( arg\_val > 0 )

{

u8g\_usart\_spi\_out(\*ptr++);

arg\_val--;

}

}

break;

case U8G\_COM\_MSG\_WRITE\_SEQ\_P:

{

register uint8\_t \*ptr = arg\_ptr;

while( arg\_val > 0 )

{

u8g\_usart\_spi\_out(u8g\_pgm\_read(ptr));

ptr++;

arg\_val--;

}

}

break;

}

return 1;

}

/\* #elif defined(\_\_18CXX) || defined(\_\_PIC32MX) \*/

/\* #elif defined(\_\_arm\_\_) // Arduino Due, maybe we should better check for \_\_SAM3X8E\_\_ \*/

#else /\* \_\_AVR\_ATmega32U4\_\_ \*/

#endif /\* \_\_AVR\_ATmega32U4\_\_ \*/

#else /\* ARDUINO \*/

uint8\_t u8g\_com\_arduino\_hw\_usart\_spi\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

return 1;

}

#endif /\* ARDUINO \*/