/\*

u8g\_arduino\_fast\_parallel.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.

PIN\_D0 8

PIN\_D1 9

PIN\_D2 10

PIN\_D3 11

PIN\_D4 4

PIN\_D5 5

PIN\_D6 6

PIN\_D7 7

PIN\_CS1 14

PIN\_CS2 15

PIN\_RW 16

PIN\_DI 17

PIN\_EN 18

u8g\_Init8Bit(u8g, dev, d0, d1, d2, d3, d4, d5, d6, d7, en, cs1, cs2, di, rw, reset)

u8g\_Init8Bit(u8g, dev, 8, 9, 10, 11, 4, 5, 6, 7, 18, 14, 15, 17, 16, U8G\_PIN\_NONE)

Update for ATOMIC operation done (01 Jun 2013)

U8G\_ATOMIC\_OR(ptr, val)

U8G\_ATOMIC\_AND(ptr, val)

U8G\_ATOMIC\_START();

U8G\_ATOMIC\_END();

\*/

#include "u8g.h"

#if defined(ARDUINO)

#if ARDUINO < 100

//#include <WProgram.h>

#include <wiring\_private.h>

#include <pins\_arduino.h>

#else

#include <Arduino.h>

#endif

#define PIN\_D0 8

#define PIN\_D1 9

#define PIN\_D2 10

#define PIN\_D3 11

#define PIN\_D4 4

#define PIN\_D5 5

#define PIN\_D6 6

#define PIN\_D7 7

#define PIN\_CS1 14

#define PIN\_CS2 15

#define PIN\_RW 16

#define PIN\_DI 17

#define PIN\_EN 18

//#define PIN\_RESET

#if defined(\_\_PIC32MX)

/\* CHIPKIT PIC32 \*/

static volatile uint32\_t \*u8g\_data\_port[8];

static uint32\_t u8g\_data\_mask[8];

#else

static volatile uint8\_t \*u8g\_data\_port[8];

static uint8\_t u8g\_data\_mask[8];

#endif

static void u8g\_com\_arduino\_fast\_parallel\_init(u8g\_t \*u8g)

{

u8g\_data\_port[0] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D0]));

u8g\_data\_mask[0] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D0]);

u8g\_data\_port[1] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D1]));

u8g\_data\_mask[1] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D1]);

u8g\_data\_port[2] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D2]));

u8g\_data\_mask[2] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D2]);

u8g\_data\_port[3] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D3]));

u8g\_data\_mask[3] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D3]);

u8g\_data\_port[4] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D4]));

u8g\_data\_mask[4] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D4]);

u8g\_data\_port[5] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D5]));

u8g\_data\_mask[5] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D5]);

u8g\_data\_port[6] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D6]));

u8g\_data\_mask[6] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D6]);

u8g\_data\_port[7] = portOutputRegister(digitalPinToPort(u8g->pin\_list[U8G\_PI\_D7]));

u8g\_data\_mask[7] = digitalPinToBitMask(u8g->pin\_list[U8G\_PI\_D7]);

}

/\* atomic protection must be done by calling function \*/

static void u8g\_com\_arduino\_fast\_write\_data\_pin(uint8\_t pin, uint8\_t val)

{

if ( val != 0 )

\*u8g\_data\_port[pin] |= u8g\_data\_mask[pin];

else

\*u8g\_data\_port[pin] &= ~u8g\_data\_mask[pin];

}

void u8g\_com\_arduino\_fast\_parallel\_write(u8g\_t \*u8g, uint8\_t val)

{

U8G\_ATOMIC\_START();

u8g\_com\_arduino\_fast\_write\_data\_pin( 0, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 1, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 2, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 3, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 4, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 5, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 6, val&1 );

val >>= 1;

u8g\_com\_arduino\_fast\_write\_data\_pin( 7, val&1 );

val >>= 1;

U8G\_ATOMIC\_END();

/\* EN cycle time must be 1 micro second \*/

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

u8g\_MicroDelay(); /\* delay by 1000ns, reference: ST7920: 140ns, SBN1661: 100ns \*/

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

u8g\_10MicroDelay(); /\* ST7920 commands: 72us \*/

u8g\_10MicroDelay(); /\* ST7920 commands: 72us \*/

}

uint8\_t u8g\_com\_arduino\_fast\_parallel\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

switch(msg)

{

case U8G\_COM\_MSG\_INIT:

u8g\_com\_arduino\_fast\_parallel\_init(u8g);

/\* setup the RW pin as output and force it to low \*/

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

{

pinMode(u8g->pin\_list[U8G\_PI\_RW], OUTPUT);

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

}

/\* set all pins (except RW pin) \*/

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

break;

case U8G\_COM\_MSG\_STOP:

break;

case U8G\_COM\_MSG\_CHIP\_SELECT:

if ( arg\_val == 0 )

{

/\* disable \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS1, HIGH);

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS2, HIGH);

}

else if ( arg\_val == 1 )

{

/\* enable \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS1, LOW);

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS2, HIGH);

}

else if ( arg\_val == 2 )

{

/\* enable \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS1, HIGH);

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS2, LOW);

}

else

{

/\* enable \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS1, LOW);

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_CS2, LOW);

}

break;

case U8G\_COM\_MSG\_WRITE\_BYTE:

u8g\_com\_arduino\_fast\_parallel\_write(u8g, arg\_val);

break;

case U8G\_COM\_MSG\_WRITE\_SEQ:

{

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

while( arg\_val > 0 )

{

u8g\_com\_arduino\_fast\_parallel\_write(u8g, \*ptr++);

arg\_val--;

}

}

break;

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

{

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

while( arg\_val > 0 )

{

u8g\_com\_arduino\_fast\_parallel\_write(u8g, u8g\_pgm\_read(ptr));

ptr++;

arg\_val--;

}

}

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\_DI, arg\_val);

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;

}

return 1;

}

#else

uint8\_t u8g\_com\_arduino\_fast\_parallel\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

return 1;

}

#endif /\* ARDUINO \*/