/\*

u8g\_com\_arduino\_t6963.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\_CS 14

PIN\_A0 15

PIN\_RESET 16

PIN\_WR 17

PIN\_RD 18

u8g\_InitRW8Bit(u8g, dev, d0, d1, d2, d3, d4, d5, d6, d7, cs, a0, wr, rd, reset)

u8g\_InitRW8Bit(u8g, dev, 8, 9, 10, 11, 4, 5, 6, 7, 14, 15, 17, 18, 16)

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

#if defined(\_\_PIC32MX)

/\* CHIPKIT PIC32 \*/

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

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

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

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

#else

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

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

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

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

#endif

static void u8g\_com\_arduino\_t6963\_init(u8g\_t \*u8g)

{

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}

static void u8g\_com\_arduino\_t6963\_write\_data\_pin(uint8\_t pin, uint8\_t val)

{

/\* no ATOMIC protection required here, this is done by calling procedure \*/

if ( val != 0 )

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

else

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

}

static void u8g\_com\_arduino\_t6963\_set\_port\_output(void)

{

uint8\_t i;

U8G\_ATOMIC\_START();

for( i = 0; i < 8; i++ )

{

#if defined(\_\_PIC32MX)

/\* CHIPKIT PIC32 \*/

\*u8g\_mode\_port[i] |= u8g\_data\_mask[i];

#elif defined(\_\_AVR\_\_)

\*u8g\_mode\_port[i] |= u8g\_data\_mask[i];

#else

/\* TODO: use generic Arduino API \*/

\*u8g\_mode\_port[i] |= u8g\_data\_mask[i];

#endif

}

U8G\_ATOMIC\_END();

}

static void u8g\_com\_arduino\_t6963\_set\_port\_input(void)

{

uint8\_t i;

U8G\_ATOMIC\_START();

for( i = 0; i < 8; i++ )

{

#if defined(\_\_PIC32MX)

/\* CHIPKIT PIC32 \*/

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

#elif defined(\_\_AVR\_\_)

/\* avr \*/

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

\*u8g\_output\_data\_port[i] &= ~u8g\_data\_mask[i]; // no pullup

#else

/\* TODO: use generic Arduino API \*/

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

\*u8g\_output\_data\_port[i] &= ~u8g\_data\_mask[i]; // no pullup

#endif

}

U8G\_ATOMIC\_END();

}

static void u8g\_com\_arduino\_t6963\_write(u8g\_t \*u8g, uint8\_t val)

{

U8G\_ATOMIC\_START();

u8g\_com\_arduino\_t6963\_write\_data\_pin( 0, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 1, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 2, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 3, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 4, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 5, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 6, val&1 );

val >>= 1;

u8g\_com\_arduino\_t6963\_write\_data\_pin( 7, val&1 );

val >>= 1;

U8G\_ATOMIC\_END();

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_WR, 0);

u8g\_MicroDelay(); /\* 80ns, reference: t6963 datasheet \*/

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_WR, 1);

u8g\_MicroDelay(); /\* 10ns, reference: t6963 datasheet \*/

}

static uint8\_t u8g\_com\_arduino\_t6963\_read(u8g\_t \*u8g)

{

uint8\_t val = 0;

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_RD, 0);

u8g\_MicroDelay(); /\* 150ns, reference: t6963 datasheet \*/

U8G\_ATOMIC\_START();

/\* only read bits 0, 1 and 3 \*/

if ( (\*u8g\_input\_data\_port[3] & u8g\_data\_mask[3]) != 0 )

val++;

val <<= 1;

val <<= 1;

if ( (\*u8g\_input\_data\_port[1] & u8g\_data\_mask[1]) != 0 )

val++;

val <<= 1;

if ( (\*u8g\_input\_data\_port[0] & u8g\_data\_mask[0]) != 0 )

val++;

U8G\_ATOMIC\_END();

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_RD, 1);

u8g\_MicroDelay(); /\* 10ns, reference: t6963 datasheet \*/

return val;

}

#define U8G\_STATUS\_TIMEOUT 50

static uint8\_t u8g\_com\_arduino\_t6963\_until\_01\_ok(u8g\_t \*u8g)

{

long x;

u8g\_com\_arduino\_t6963\_set\_port\_input();

x = millis();

x += U8G\_STATUS\_TIMEOUT;

for(;;)

{

if ( (u8g\_com\_arduino\_t6963\_read(u8g) & 3) == 3 )

break;

if ( x < millis() )

return 0;

}

u8g\_com\_arduino\_t6963\_set\_port\_output();

return 1;

}

static uint8\_t u8g\_com\_arduino\_t6963\_until\_3\_ok(u8g\_t \*u8g)

{

long x;

u8g\_com\_arduino\_t6963\_set\_port\_input();

x = millis();

x += U8G\_STATUS\_TIMEOUT;

for(;;)

{

if ( (u8g\_com\_arduino\_t6963\_read(u8g) & 8) == 8 )

break;

if ( x < millis() )

return 0;

}

u8g\_com\_arduino\_t6963\_set\_port\_output();

return 1;

}

static uint8\_t u8g\_com\_arduino\_t6963\_write\_cmd(u8g\_t \*u8g, uint8\_t val)

{

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, 1);

if ( u8g\_com\_arduino\_t6963\_until\_01\_ok(u8g) == 0 )

return 0;

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, 1);

u8g\_com\_arduino\_t6963\_write(u8g, val);

return 1;

}

static uint8\_t u8g\_com\_arduino\_t6963\_write\_data(u8g\_t \*u8g, uint8\_t val)

{

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, 1);

if ( u8g\_com\_arduino\_t6963\_until\_01\_ok(u8g) == 0 )

return 0;

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, 0);

u8g\_com\_arduino\_t6963\_write(u8g, val);

return 1;

}

static uint8\_t u8g\_com\_arduino\_t6963\_write\_auto\_data(u8g\_t \*u8g, uint8\_t val)

{

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, 1);

if ( u8g\_com\_arduino\_t6963\_until\_3\_ok(u8g) == 0 )

return 0;

u8g\_com\_arduino\_digital\_write(u8g, U8G\_PI\_A0, 0);

u8g\_com\_arduino\_t6963\_write(u8g, val);

return 1;

}

uint8\_t u8g\_com\_arduino\_t6963\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

switch(msg)

{

case U8G\_COM\_MSG\_INIT:

u8g->pin\_list[U8G\_PI\_A0\_STATE] = 0;

u8g\_com\_arduino\_t6963\_init(u8g);

/\* setup the RW (equal to WR) pin as output and force it to high \*/

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

{

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

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

}

/\* set all pins (except WR 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, active low chip select \*/

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\_WRITE\_BYTE:

if ( u8g->pin\_list[U8G\_PI\_A0\_STATE] == 0 )

{

u8g\_com\_arduino\_t6963\_write\_data(u8g, arg\_val);

}

else

{

u8g\_com\_arduino\_t6963\_write\_cmd(u8g, arg\_val);

}

break;

case U8G\_COM\_MSG\_WRITE\_SEQ:

{

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

u8g\_com\_arduino\_t6963\_write\_cmd(u8g, 0x0b0); /\* auto write \*/

while( arg\_val > 0 )

{

if ( u8g\_com\_arduino\_t6963\_write\_auto\_data(u8g, \*ptr++) == 0 )

break;

arg\_val--;

}

u8g\_com\_arduino\_t6963\_write\_cmd(u8g, 0x0b2); /\* auto reset \*/

}

break;

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

{

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

u8g\_com\_arduino\_t6963\_write\_cmd(u8g, 0x0b0); /\* auto write \*/

while( arg\_val > 0 )

{

if ( u8g\_com\_arduino\_t6963\_write\_auto\_data(u8g, u8g\_pgm\_read(ptr)) == 0 )

break;

ptr++;

arg\_val--;

}

u8g\_com\_arduino\_t6963\_write\_cmd(u8g, 0x0b2); /\* auto reset \*/

}

break;

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

u8g->pin\_list[U8G\_PI\_A0\_STATE] = arg\_val;

//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\_t6963\_fn(u8g\_t \*u8g, uint8\_t msg, uint8\_t arg\_val, void \*arg\_ptr)

{

return 1;

}

#endif /\* ARDUINO \*/