/\*

u8g\_com\_api.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.

\*/

#include "u8g.h"

uint8\_t u8g\_InitCom(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t clk\_cycle\_time)

{

return dev->com\_fn(u8g, U8G\_COM\_MSG\_INIT, clk\_cycle\_time, NULL);

}

void u8g\_StopCom(u8g\_t \*u8g, u8g\_dev\_t \*dev)

{

dev->com\_fn(u8g, U8G\_COM\_MSG\_STOP, 0, NULL);

}

/\* cs contains the chip number, which should be enabled \*/

void u8g\_SetChipSelect(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t cs)

{

dev->com\_fn(u8g, U8G\_COM\_MSG\_CHIP\_SELECT, cs, NULL);

}

void u8g\_SetResetLow(u8g\_t \*u8g, u8g\_dev\_t \*dev)

{

dev->com\_fn(u8g, U8G\_COM\_MSG\_RESET, 0, NULL);

}

void u8g\_SetResetHigh(u8g\_t \*u8g, u8g\_dev\_t \*dev)

{

dev->com\_fn(u8g, U8G\_COM\_MSG\_RESET, 1, NULL);

}

void u8g\_SetAddress(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t address)

{

dev->com\_fn(u8g, U8G\_COM\_MSG\_ADDRESS, address, NULL);

}

uint8\_t u8g\_WriteByte(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t val)

{

return dev->com\_fn(u8g, U8G\_COM\_MSG\_WRITE\_BYTE, val, NULL);

}

uint8\_t u8g\_WriteSequence(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t cnt, uint8\_t \*seq)

{

return dev->com\_fn(u8g, U8G\_COM\_MSG\_WRITE\_SEQ, cnt, seq);

}

uint8\_t u8g\_WriteSequenceP(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t cnt, const uint8\_t \*seq)

{

return dev->com\_fn(u8g, U8G\_COM\_MSG\_WRITE\_SEQ\_P, cnt, (void \*)seq);

}

/\*

sequence := { direct\_value | escape\_sequence }

direct\_value := 0..254

escape\_sequence := value\_255 | sequence\_end | delay | adr | cs | not\_used

value\_255 := 255 255

sequence\_end = 255 254

delay := 255 0..127

adr := 255 0x0e0 .. 0x0ef

cs := 255 0x0d0 .. 0x0df

not\_used := 255 101..254

#define U8G\_ESC\_DLY(x) 255, ((x) & 0x7f)

#define U8G\_ESC\_CS(x) 255, (0xd0 | ((x)&0x0f))

#define U8G\_ESC\_ADR(x) 255, (0xe0 | ((x)&0x0f))

#define U8G\_ESC\_VCC(x) 255, (0xbe | ((x)&0x01))

#define U8G\_ESC\_END 255, 254

#define U8G\_ESC\_255 255, 255

#define U8G\_ESC\_RST(x) 255, (0xc0 | ((x)&0x0f))

\*/

uint8\_t u8g\_WriteEscSeqP(u8g\_t \*u8g, u8g\_dev\_t \*dev, const uint8\_t \*esc\_seq)

{

uint8\_t is\_escape = 0;

uint8\_t value;

for(;;)

{

value = u8g\_pgm\_read(esc\_seq);

if ( is\_escape == 0 )

{

if ( value != 255 )

{

if ( u8g\_WriteByte(u8g, dev, value) == 0 )

return 0;

}

else

{

is\_escape = 1;

}

}

else

{

if ( value == 255 )

{

if ( u8g\_WriteByte(u8g, dev, value) == 0 )

return 0;

}

else if ( value == 254 )

{

break;

}

else if ( value >= 0x0f0 )

{

/\* not yet used, do nothing \*/

}

else if ( value >= 0xe0 )

{

u8g\_SetAddress(u8g, dev, value & 0x0f);

}

else if ( value >= 0xd0 )

{

u8g\_SetChipSelect(u8g, dev, value & 0x0f);

}

else if ( value >= 0xc0 )

{

u8g\_SetResetLow(u8g, dev);

value &= 0x0f;

value <<= 4;

value+=2;

u8g\_Delay(value);

u8g\_SetResetHigh(u8g, dev);

u8g\_Delay(value);

}

else if ( value >= 0xbe )

{

/\* not yet implemented \*/

/\* u8g\_SetVCC(u8g, dev, value & 0x01); \*/

}

else if ( value <= 127 )

{

u8g\_Delay(value);

}

is\_escape = 0;

}

esc\_seq++;

}

return 1;

}