/\*

u8g\_com\_api\_16gr.c

Extension of the com api for devices with 16 graylevels (4 bit per pixel).

This should fit to the 8h and 16h architectures (pb8v1, pb8v2, pb16v1, pb16v2),

mainly intended for SSD OLEDs

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"

/\* interpret b as a monochrome bit pattern, write value 15 for high bit and value 0 for a low bit \*/

/\* topbit (msb) is sent last \*/

/\* example: b = 0x083 will send 0xff, 0x00, 0x00, 0xf0 \*/

uint8\_t u8g\_WriteByteBWTo16GrDevice(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t b)

{

static uint8\_t buf[4];

static uint8\_t map[4] = { 0, 0x00f, 0x0f0, 0x0ff };

buf [3] = map[b & 3];

b>>=2;

buf [2] = map[b & 3];

b>>=2;

buf [1] = map[b & 3];

b>>=2;

buf [0] = map[b & 3];

return dev->com\_fn(u8g, U8G\_COM\_MSG\_WRITE\_SEQ, 4, buf);

}

uint8\_t u8g\_WriteSequenceBWTo16GrDevice(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t cnt, uint8\_t \*ptr)

{

do

{

if ( u8g\_WriteByteBWTo16GrDevice(u8g, dev, \*ptr++) == 0 )

return 0;

cnt--;

} while( cnt != 0 );

return 1;

}

/\* interpret b as a 4L bit pattern, write values 0x000, 0x004, 0x008, 0x00c \*/

uint8\_t u8g\_WriteByte4LTo16GrDevice(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t b)

{

//static uint8\_t map[16] = { 0x000, 0x004, 0x008, 0x00c, 0x040, 0x044, 0x048, 0x04c, 0x080, 0x084, 0x088, 0x08c, 0x0c0, 0x0c4, 0x0c8, 0x0cc};

//static uint8\_t map[16] = { 0x000, 0x004, 0x00a, 0x00f, 0x040, 0x044, 0x04a, 0x04f, 0x0a0, 0x0a4, 0x0aa, 0x0af, 0x0f0, 0x0f4, 0x0fa, 0x0ff};

static uint8\_t map[16] = { 0x000, 0x040, 0x0a0, 0x0f0, 0x004, 0x044, 0x0a4, 0x0f4, 0x00a, 0x04a, 0x0aa, 0x0fa, 0x00f, 0x04f, 0x0af, 0x0ff};

uint8\_t bb;

bb = b;

bb &= 15;

b>>=4;

dev->com\_fn(u8g, U8G\_COM\_MSG\_WRITE\_BYTE, map[bb], NULL);

return dev->com\_fn(u8g, U8G\_COM\_MSG\_WRITE\_BYTE, map[b], NULL);

}

uint8\_t u8g\_WriteSequence4LTo16GrDevice(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t cnt, uint8\_t \*ptr)

{

do

{

if ( u8g\_WriteByte4LTo16GrDevice(u8g, dev, \*ptr++) == 0 )

return 0;

cnt--;

} while( cnt != 0 );

return 1;

}