/\*

u8g\_dev\_flipdisc.c

1-Bit (BW) Driver for flip disc matrix

2x 7 pixel height

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"

#define WIDTH 28

#define HEIGHT 14

#define PAGE\_HEIGHT 14

/\*

Write data to the flip disc matrix.

This procedure must be implemented by the user.

Arguments:

id: Id for the matrix. Currently always 0.

page: A page has a height of 14 pixel. For a matrix with HEIGHT == 14 this will be always 0

width: The width of the flip disc matrix. Always equal to WIDTH

row1: first data line (7 pixel per byte)

row2: first data line (7 pixel per byte)

\*/

void writeFlipDiscMatrix(uint8\_t id, uint8\_t page, uint8\_t width, uint8\_t \*row1, uint8\_t \*row2);

void (\*u8g\_write\_flip\_disc\_matrix)(uint8\_t id, uint8\_t page, uint8\_t width, uint8\_t \*row1, uint8\_t \*row2);

void u8g\_SetFlipDiscCallback(u8g\_t \*u8g, void (\*cb)(uint8\_t id, uint8\_t page, uint8\_t width, uint8\_t \*row1, uint8\_t \*row2))

{

u8g\_write\_flip\_disc\_matrix = cb;

}

uint8\_t u8g\_dev\_flipdisc\_2x7\_bw\_fn(u8g\_t \*u8g, u8g\_dev\_t \*dev, uint8\_t msg, void \*arg)

{

switch(msg)

{

case U8G\_DEV\_MSG\_INIT:

break;

case U8G\_DEV\_MSG\_STOP:

break;

case U8G\_DEV\_MSG\_PAGE\_NEXT:

{

u8g\_pb\_t \*pb = (u8g\_pb\_t \*)(dev->dev\_mem);

/\* current page: pb->p.page \*/

/\* ptr to the buffer: pb->buf \*/

(\*u8g\_write\_flip\_disc\_matrix)(0, pb->p.page, WIDTH, pb->buf, (uint8\_t \*)(pb->buf)+WIDTH);

}

break;

case U8G\_DEV\_MSG\_CONTRAST:

return 1;

}

return u8g\_dev\_pb14v1\_base\_fn(u8g, dev, msg, arg);

}

uint8\_t u8g\_dev\_flipdisc\_2x7\_bw\_buf[WIDTH\*2] U8G\_NOCOMMON ;

u8g\_pb\_t u8g\_dev\_flipdisc\_2x7\_bw\_pb = { {16, HEIGHT, 0, 0, 0}, WIDTH, u8g\_dev\_flipdisc\_2x7\_bw\_buf};

u8g\_dev\_t u8g\_dev\_flipdisc\_2x7 = { u8g\_dev\_flipdisc\_2x7\_bw\_fn, &u8g\_dev\_flipdisc\_2x7\_bw\_pb, u8g\_com\_null\_fn };