/\*

u8g\_pb.c

common procedures for the page buffer

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"

void u8g\_pb\_Clear(u8g\_pb\_t \*b)

{

uint8\_t \*ptr = (uint8\_t \*)b->buf;

uint8\_t \*end\_ptr = ptr;

end\_ptr += b->width;

do

{

\*ptr++ = 0;

} while( ptr != end\_ptr );

}

/\* the following procedure does not work. why? Can be checked with descpic \*/

/\*

void u8g\_pb\_Clear(u8g\_pb\_t \*b)

{

uint8\_t \*ptr = (uint8\_t \*)b->buf;

uint8\_t cnt = b->width;

do

{

\*ptr++ = 0;

cnt--;

} while( cnt != 0 );

}

\*/

/\*

intersection assumptions:

a1 <= a2 is always true

\*/

/\*

minimized version

---1----0 1 b1 <= a2 && b1 > b2

-----1--0 1 b2 >= a1 && b1 > b2

---1-1--- 1 b1 <= a2 && b2 >= a1

\*/

/\*

uint8\_t u8g\_pb8v1\_IsYIntersection\_\_\_Old(u8g\_pb\_t \*b, u8g\_uint\_t v0, u8g\_uint\_t v1)

{

uint8\_t c0, c1, c;

c0 = v0 <= b->p.page\_y1;

c1 = v1 >= b->p.page\_y0;

c = v0 > v1;

if ( c0 && c1 ) return 1;

if ( c0 && c ) return 1;

if ( c1 && c ) return 1;

return 0;

}

\*/

uint8\_t u8g\_pb\_IsYIntersection(u8g\_pb\_t \*pb, u8g\_uint\_t v0, u8g\_uint\_t v1)

{

uint8\_t c1, c2, c3, tmp;

c1 = v0 <= pb->p.page\_y1;

c2 = v1 >= pb->p.page\_y0;

c3 = v0 > v1;

/\*

if ( c1 && c2 )

return 1;

if ( c1 && c3 )

return 1;

if ( c2 && c3 )

return 1;

return 0;

\*/

tmp = c1;

c1 &= c2;

c2 &= c3;

c3 &= tmp;

c1 |= c2;

c1 |= c3;

return c1 & 1;

}

uint8\_t u8g\_pb\_IsXIntersection(u8g\_pb\_t \*b, u8g\_uint\_t v0, u8g\_uint\_t v1)

{

uint8\_t /\*c0, c1, \*/ c2, c3;

/\*

conditions: b->p.page\_y0 < b->p.page\_y1

there are no restriction on v0 and v1. If v0 > v1, then warp around unsigned is assumed

\*/

/\*

c0 = v0 < 0;

c1 = v1 < 0;

\*/

c2 = v0 > b->width;

c3 = v1 > b->width;

/\*if ( c0 && c1 ) return 0;\*/

if ( c2 && c3 ) return 0;

/\*if ( c1 && c2 ) return 0;\*/

return 1;

}

uint8\_t u8g\_pb\_IsIntersection(u8g\_pb\_t \*pb, u8g\_dev\_arg\_bbx\_t \*bbx)

{

u8g\_uint\_t tmp;

tmp = bbx->y;

tmp += bbx->h;

tmp--;

if ( u8g\_pb\_IsYIntersection(pb, bbx->y, tmp) == 0 )

return 0;

/\* maybe this one can be skiped... probability is very high to have an intersection, so it would be ok to always return 1 \*/

tmp = bbx->x;

tmp += bbx->w;

tmp--;

return u8g\_pb\_IsXIntersection(pb, bbx->x, tmp);

}

void u8g\_pb\_GetPageBox(u8g\_pb\_t \*pb, u8g\_box\_t \*box)

{

box->x0 = 0;

box->y0 = pb->p.page\_y0;

box->x1 = pb->width;

box->x1--;

box->y1 = pb->p.page\_y1;

}

uint8\_t u8g\_pb\_Is8PixelVisible(u8g\_pb\_t \*b, u8g\_dev\_arg\_pixel\_t \*arg\_pixel)

{

u8g\_uint\_t v0, v1;

v0 = arg\_pixel->y;

v1 = v0;

switch( arg\_pixel->dir )

{

case 0:

break;

case 1:

v1 += 8; /\* this is independent from the page height \*/

break;

case 2:

break;

case 3:

v0 -= 8;

break;

}

return u8g\_pb\_IsYIntersection(b, v0, v1);

}

uint8\_t u8g\_pb\_WriteBuffer(u8g\_pb\_t \*b, u8g\_t \*u8g, u8g\_dev\_t \*dev)

{

return u8g\_WriteSequence(u8g, dev, b->width, b->buf);

}