/\*

u8g\_clip.c

procedures for clipping

taken over from procs in u8g\_pb.c

Universal 8bit Graphics Library

Copyright (c) 2012, 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.

Notes

This is one of the most critical parts of u8glib. It must be fast, but still reliable.

Based on the intersection program (see tools folder), there is minimized version of

the condition for the intersaction test:

minimized version

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

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

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

It includes the assumption, that a1 <= a2 is always true (correct, because

a1, a2 are the page dimensions.

The direct implementation of the above result is done in:

uint8\_t u8g\_is\_intersection\_boolean(u8g\_uint\_t a0, u8g\_uint\_t a1, u8g\_uint\_t v0, u8g\_uint\_t v1)

However, this is slower than a decision tree version:

static uint8\_t u8g\_is\_intersection\_decision\_tree(u8g\_uint\_t a0, u8g\_uint\_t a1, u8g\_uint\_t v0, u8g\_uint\_t v1)

Also suprising is, that the macro implementation is slower than the inlined version.

The decision tree is based on the expansion of the truth table.

\*/

#include "u8g.h"

#ifdef \_\_GNUC\_\_

#define U8G\_ALWAYS\_INLINE \_\_inline\_\_ \_\_attribute\_\_((always\_inline))

#else

#define U8G\_ALWAYS\_INLINE

#endif

/\*

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

\*/

#ifdef OLD\_CODE\_WHICH\_IS\_TOO\_SLOW

static uint8\_t u8g\_is\_intersection\_boolean(u8g\_uint\_t a0, u8g\_uint\_t a1, u8g\_uint\_t v0, u8g\_uint\_t v1)

{

uint8\_t c1, c2, c3, tmp;

c1 = v0 <= a1;

c2 = v1 >= a0;

c3 = v0 > v1;

tmp = c1;

c1 &= c2;

c2 &= c3;

c3 &= tmp;

c1 |= c2;

c1 |= c3;

return c1 & 1;

}

#endif

#define U8G\_IS\_INTERSECTION\_MACRO(a0,a1,v0,v1) ((uint8\_t)( (v0) <= (a1) ) ? ( ( (v1) >= (a0) ) ? ( 1 ) : ( (v0) > (v1) ) ) : ( ( (v1) >= (a0) ) ? ( (v0) > (v1) ) : ( 0 ) ))

//static uint8\_t u8g\_is\_intersection\_decision\_tree(u8g\_uint\_t a0, u8g\_uint\_t a1, u8g\_uint\_t v0, u8g\_uint\_t v1) U8G\_ALWAYS\_INLINE;

static uint8\_t U8G\_ALWAYS\_INLINE u8g\_is\_intersection\_decision\_tree(u8g\_uint\_t a0, u8g\_uint\_t a1, u8g\_uint\_t v0, u8g\_uint\_t v1)

{

/\* surprisingly the macro leads to larger code \*/

/\* return U8G\_IS\_INTERSECTION\_MACRO(a0,a1,v0,v1); \*/

if ( v0 <= a1 )

{

if ( v1 >= a0 )

{

return 1;

}

else

{

if ( v0 > v1 )

{

return 1;

}

else

{

return 0;

}

}

}

else

{

if ( v1 >= a0 )

{

if ( v0 > v1 )

{

return 1;

}

else

{

return 0;

}

}

else

{

return 0;

}

}

}

uint8\_t u8g\_IsBBXIntersection(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t w, u8g\_uint\_t h)

{

register u8g\_uint\_t tmp;

tmp = y;

tmp += h;

tmp--;

if ( u8g\_is\_intersection\_decision\_tree(u8g->current\_page.y0, u8g->current\_page.y1, y, tmp) == 0 )

return 0;

tmp = x;

tmp += w;

tmp--;

return u8g\_is\_intersection\_decision\_tree(u8g->current\_page.x0, u8g->current\_page.x1, x, tmp);

}