/\*

u8g\_circle.c

Utility to draw empty and filled circles.

Universal 8bit Graphics Library

Copyright (c) 2011, bjthom@gmail.com

u8g\_DrawCircle & u8g\_DrawDisc by 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.

Addition to the U8G Library 02/25/12

\*/

#include "u8g.h"

#ifdef OLD\_CODE

void circ\_upperRight(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0) {

u8g\_DrawPixel(u8g, x0 + x, y0 - y);

u8g\_DrawPixel(u8g, x0 + y, y0 - x);

}

void circ\_upperLeft(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0) {

u8g\_DrawPixel(u8g, x0 - x, y0 - y);

u8g\_DrawPixel(u8g, x0 - y, y0 - x);

}

void circ\_lowerRight(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0) {

u8g\_DrawPixel(u8g, x0 + x, y0 + y);

u8g\_DrawPixel(u8g, x0 + y, y0 + x);

}

void circ\_lowerLeft(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0) {

u8g\_DrawPixel(u8g, x0 - x, y0 + y);

u8g\_DrawPixel(u8g, x0 - y, y0 + x);

}

void circ\_all(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0) {

circ\_upperRight(u8g, x, y, x0, y0);

circ\_upperLeft(u8g, x, y, x0, y0);

circ\_lowerRight(u8g, x, y, x0, y0);

circ\_lowerLeft(u8g, x, y, x0, y0);

}

void u8g\_DrawEmpCirc(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rad, uint8\_t option)

{

if ( u8g\_IsBBXIntersection(u8g, x0-rad-1, y0-rad-1, 2\*rad+1, 2\*rad+1) == 0)

return;

int f = 1 - rad;

int ddF\_x = 1;

int ddF\_y = -2\*rad;

uint8\_t x = 0;

uint8\_t y = rad;

void ( \*circ\_util )(u8g\_t \*, u8g\_uint\_t, u8g\_uint\_t, u8g\_uint\_t, u8g\_uint\_t);

switch (option)

{

case U8G\_CIRC\_UPPER\_RIGHT:

u8g\_DrawPixel(u8g, x0, y0 - rad);

u8g\_DrawPixel(u8g, x0 + rad, y0);

circ\_util = circ\_upperRight;

break;

case U8G\_CIRC\_UPPER\_LEFT:

u8g\_DrawPixel(u8g, x0, y0 - rad);

u8g\_DrawPixel(u8g, x0 - rad, y0);

circ\_util = circ\_upperLeft;

break;

case U8G\_CIRC\_LOWER\_RIGHT:

u8g\_DrawPixel(u8g, x0, y0 + rad);

u8g\_DrawPixel(u8g, x0 + rad, y0);

circ\_util = circ\_lowerRight;

break;

case U8G\_CIRC\_LOWER\_LEFT:

u8g\_DrawPixel(u8g, x0, y0 + rad);

u8g\_DrawPixel(u8g, x0 - rad, y0);

circ\_util = circ\_lowerLeft;

break;

default:

case U8G\_CIRC\_ALL:

u8g\_DrawPixel(u8g, x0, y0 + rad);

u8g\_DrawPixel(u8g, x0, y0 - rad);

u8g\_DrawPixel(u8g, x0 + rad, y0);

u8g\_DrawPixel(u8g, x0 - rad, y0);

circ\_util = circ\_all;

break;

}

while( x < y )

{

if(f >= 0)

{

y--;

ddF\_y += 2;

f += ddF\_y;

}

x++;

ddF\_x += 2;

f += ddF\_x;

circ\_util(u8g, x, y, x0, y0);

}

}

void u8g\_DrawFillCirc(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rad, uint8\_t option)

{

if ( u8g\_IsBBXIntersection(u8g, x0-rad-1, y0-rad-1, 2\*rad+1, 2\*rad+1) == 0)

return;

int f = 1 - rad;

int ddF\_x = 1;

int ddF\_y = -2\*rad;

uint8\_t x = 0;

uint8\_t y = rad;

// Draw vertical diameter at the horiz. center

// u8g\_DrawVLine(u8g, x0, y0 - rad, 2\*rad+1);

if (option == U8G\_CIRC\_UPPER\_LEFT || option == U8G\_CIRC\_UPPER\_RIGHT) {

u8g\_DrawVLine(u8g, x0, y0 - rad, rad+1);

}

else if (option == U8G\_CIRC\_LOWER\_LEFT || option == U8G\_CIRC\_LOWER\_RIGHT) {

u8g\_DrawVLine(u8g, x0, y0, rad+1);

}

else {

u8g\_DrawVLine(u8g, x0, y0 - rad, 2\*rad+1);

}

while( x < y )

{

if(f >= 0)

{

y--;

ddF\_y += 2;

f += ddF\_y;

}

x++;

ddF\_x += 2;

f += ddF\_x;

//Draw vertical lines from one point to another

switch (option)

{

case U8G\_CIRC\_UPPER\_RIGHT:

u8g\_DrawVLine(u8g, x0+x, y0-y, y+1);

u8g\_DrawVLine(u8g, x0+y, y0-x, x+1);

break;

case U8G\_CIRC\_UPPER\_LEFT:

u8g\_DrawVLine(u8g, x0-x, y0-y, y+1);

u8g\_DrawVLine(u8g, x0-y, y0-x, x+1);

break;

case U8G\_CIRC\_LOWER\_RIGHT:

u8g\_DrawVLine(u8g, x0+x, y0, y+1);

u8g\_DrawVLine(u8g, x0+y, y0, x+1);

break;

case U8G\_CIRC\_LOWER\_LEFT:

u8g\_DrawVLine(u8g, x0-x, y0, y+1);

u8g\_DrawVLine(u8g, x0-y, y0, x+1);

break;

case U8G\_CIRC\_ALL:

u8g\_DrawVLine(u8g, x0+x, y0-y, 2\*y+1);

u8g\_DrawVLine(u8g, x0-x, y0-y, 2\*y+1);

u8g\_DrawVLine(u8g, x0+y, y0-x, 2\*x+1);

u8g\_DrawVLine(u8g, x0-y, y0-x, 2\*x+1);

break;

}

}

}

#endif

/\*=========================================================================\*/

static void u8g\_draw\_circle\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option) U8G\_NOINLINE;

static void u8g\_draw\_circle\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option)

{

/\* upper right \*/

if ( option & U8G\_DRAW\_UPPER\_RIGHT )

{

u8g\_DrawPixel(u8g, x0 + x, y0 - y);

u8g\_DrawPixel(u8g, x0 + y, y0 - x);

}

/\* upper left \*/

if ( option & U8G\_DRAW\_UPPER\_LEFT )

{

u8g\_DrawPixel(u8g, x0 - x, y0 - y);

u8g\_DrawPixel(u8g, x0 - y, y0 - x);

}

/\* lower right \*/

if ( option & U8G\_DRAW\_LOWER\_RIGHT )

{

u8g\_DrawPixel(u8g, x0 + x, y0 + y);

u8g\_DrawPixel(u8g, x0 + y, y0 + x);

}

/\* lower left \*/

if ( option & U8G\_DRAW\_LOWER\_LEFT )

{

u8g\_DrawPixel(u8g, x0 - x, y0 + y);

u8g\_DrawPixel(u8g, x0 - y, y0 + x);

}

}

void u8g\_draw\_circle(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rad, uint8\_t option)

{

u8g\_int\_t f;

u8g\_int\_t ddF\_x;

u8g\_int\_t ddF\_y;

u8g\_uint\_t x;

u8g\_uint\_t y;

f = 1;

f -= rad;

ddF\_x = 1;

ddF\_y = 0;

ddF\_y -= rad;

ddF\_y \*= 2;

x = 0;

y = rad;

u8g\_draw\_circle\_section(u8g, x, y, x0, y0, option);

while ( x < y )

{

if (f >= 0)

{

y--;

ddF\_y += 2;

f += ddF\_y;

}

x++;

ddF\_x += 2;

f += ddF\_x;

u8g\_draw\_circle\_section(u8g, x, y, x0, y0, option);

}

}

void u8g\_DrawCircle(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rad, uint8\_t option)

{

/\* check for bounding box \*/

{

u8g\_uint\_t radp, radp2;

radp = rad;

radp++;

radp2 = radp;

radp2 \*= 2;

if ( u8g\_IsBBXIntersection(u8g, x0-radp, y0-radp, radp2, radp2) == 0)

return;

}

/\* draw circle \*/

u8g\_draw\_circle(u8g, x0, y0, rad, option);

}

static void u8g\_draw\_disc\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option) U8G\_NOINLINE;

static void u8g\_draw\_disc\_section(u8g\_t \*u8g, u8g\_uint\_t x, u8g\_uint\_t y, u8g\_uint\_t x0, u8g\_uint\_t y0, uint8\_t option)

{

/\* upper right \*/

if ( option & U8G\_DRAW\_UPPER\_RIGHT )

{

u8g\_DrawVLine(u8g, x0+x, y0-y, y+1);

u8g\_DrawVLine(u8g, x0+y, y0-x, x+1);

}

/\* upper left \*/

if ( option & U8G\_DRAW\_UPPER\_LEFT )

{

u8g\_DrawVLine(u8g, x0-x, y0-y, y+1);

u8g\_DrawVLine(u8g, x0-y, y0-x, x+1);

}

/\* lower right \*/

if ( option & U8G\_DRAW\_LOWER\_RIGHT )

{

u8g\_DrawVLine(u8g, x0+x, y0, y+1);

u8g\_DrawVLine(u8g, x0+y, y0, x+1);

}

/\* lower left \*/

if ( option & U8G\_DRAW\_LOWER\_LEFT )

{

u8g\_DrawVLine(u8g, x0-x, y0, y+1);

u8g\_DrawVLine(u8g, x0-y, y0, x+1);

}

}

void u8g\_draw\_disc(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rad, uint8\_t option)

{

u8g\_int\_t f;

u8g\_int\_t ddF\_x;

u8g\_int\_t ddF\_y;

u8g\_uint\_t x;

u8g\_uint\_t y;

f = 1;

f -= rad;

ddF\_x = 1;

ddF\_y = 0;

ddF\_y -= rad;

ddF\_y \*= 2;

x = 0;

y = rad;

u8g\_draw\_disc\_section(u8g, x, y, x0, y0, option);

while ( x < y )

{

if (f >= 0)

{

y--;

ddF\_y += 2;

f += ddF\_y;

}

x++;

ddF\_x += 2;

f += ddF\_x;

u8g\_draw\_disc\_section(u8g, x, y, x0, y0, option);

}

}

void u8g\_DrawDisc(u8g\_t \*u8g, u8g\_uint\_t x0, u8g\_uint\_t y0, u8g\_uint\_t rad, uint8\_t option)

{

/\* check for bounding box \*/

{

u8g\_uint\_t radp, radp2;

radp = rad;

radp++;

radp2 = radp;

radp2 \*= 2;

if ( u8g\_IsBBXIntersection(u8g, x0-radp, y0-radp, radp2, radp2) == 0)

return;

}

/\* draw disc \*/

u8g\_draw\_disc(u8g, x0, y0, rad, option);

}