libg15render

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 g15canvas Struct Reference

This structure holds the data need to render objects to the LCD screen.

```
#include <libg15render.h>
```

Data Fields

- unsigned char **buffer** [**G15_BUFFER_LEN**]
- FT_Library **ftLib**
- int **mode_cache**
- int **mode_reverse**
- int **mode_xor**
- FT_Face **ttf_face** [**G15_MAX_FACE**][sizeof(FT_Face)]
- int **ttf_fontsize** [**G15_MAX_FACE**]

3.1.1 Detailed Description

This structure holds the data need to render objects to the LCD screen.

Definition at line 36 of file libg15render.h.

3.1.2 Field Documentation

3.1.2.1 unsigned char g15canvas::buffer[G15_BUFFER_LEN]

g15canvas::buffer (p. 5)[] is a buffer holding the pixel data to be sent to the LCD.

Definition at line 39 of file libg15render.h.

Referenced by g15r_clearScreen(), g15r_getPixel(), g15r_initCanvas(), g15r_loadWbmpSplash(), and g15r_set↩
Pixel().

3.1.2.2 FT_Library g15canvas::ftLib

Definition at line 47 of file libg15render.h.

Referenced by draw_ttf_char(), g15r_initCanvas(), and g15r_ttfLoad().

3.1.2.3 int g15canvas::mode_cache

g15canvas::mode_cache (p. 6) can be used to determine whether caching should be used in an application.

Definition at line 43 of file libg15render.h.

Referenced by g15r_initCanvas().

3.1.2.4 int g15canvas::mode_reverse

g15canvas::mode_reverse (p. 6) determines whether color values passed to g15r_setPixel are reversed.

Definition at line 45 of file libg15render.h.

Referenced by g15r_initCanvas(), and g15r_setPixel().

3.1.2.5 int g15canvas::mode_xor

g15canvas::mode_xor (p. 6) determines whether xor processing is used in g15r_setPixel.

Definition at line 41 of file libg15render.h.

Referenced by g15r_initCanvas(), and g15r_setPixel().

3.1.2.6 FT_Face g15canvas::ttf_face[G15_MAX_FACE][sizeof(FT_Face)]

Definition at line 48 of file libg15render.h.

Referenced by g15r_ttfLoad(), and g15r_ttfPrint().

3.1.2.7 int g15canvas::ttf_fontsize[G15_MAX_FACE]

Definition at line 49 of file libg15render.h.

Referenced by g15r_ttfLoad(), and g15r_ttfPrint().

The documentation for this struct was generated from the following file:

- **libg15render.h**

Chapter 4

File Documentation

4.1 config.h File Reference

Macros

- #define **HAVE_DLFCN_H** 1
- #define **HAVE_FT2BUILD_H** 1
- #define **HAVE_INTTYPES_H** 1
- #define **HAVE_LIBG15** 1
- #define **HAVE_LIBM** 1
- #define **HAVE_MEMORY_H** 1
- #define **HAVE_MEMSET** 1
- #define **HAVE_STDINT_H** 1
- #define **HAVE_STDLIB_H** 1
- #define **HAVE_STRING_H** 1
- #define **HAVE_STRINGS_H** 1
- #define **HAVE_SYS_STAT_H** 1
- #define **HAVE_SYS_TYPES_H** 1
- #define **HAVE_UNISTD_H** 1
- #define **PACKAGE** "libg15render"
- #define **PACKAGE_BUGREPORT** "mirabeaj@gmail.com"
- #define **PACKAGE_NAME** "libg15render"
- #define **PACKAGE_STRING** "libg15render 1.2"
- #define **PACKAGE_TARNAME** "libg15render"
- #define **PACKAGE_VERSION** "1.2"
- #define **STDC_HEADERS** 1
- #define **TTF_SUPPORT** 1
- #define **VERSION** "1.2"

4.1.1 Macro Definition Documentation

4.1.1.1 #define HAVE_DLFCN_H 1

Definition at line 5 of file config.h.

4.1.1.2 #define HAVE_FT2BUILD_H 1

Definition at line 8 of file config.h.

4.1.1.3 #define HAVE_INTTYPES_H 1

Definition at line 11 of file config.h.

4.1.1.4 #define HAVE_LIBG15 1

Definition at line 14 of file config.h.

4.1.1.5 #define HAVE_LIBM 1

Definition at line 17 of file config.h.

4.1.1.6 #define HAVE_MEMORY_H 1

Definition at line 20 of file config.h.

4.1.1.7 #define HAVE_MEMSET 1

Definition at line 23 of file config.h.

4.1.1.8 #define HAVE_STDINT_H 1

Definition at line 26 of file config.h.

4.1.1.9 #define HAVE_STDLIB_H 1

Definition at line 29 of file config.h.

4.1.1.10 #define HAVE_STRING_H 1

Definition at line 35 of file config.h.

4.1.1.11 #define HAVE_STRINGS_H 1

Definition at line 32 of file config.h.

4.1.1.12 #define HAVE_SYS_STAT_H 1

Definition at line 38 of file config.h.

4.1.1.13 #define HAVE_SYS_TYPES_H 1

Definition at line 41 of file config.h.

4.1.1.14 #define HAVE_UNISTD_H 1

Definition at line 44 of file config.h.

4.1.1.15 #define PACKAGE "libg15render"

Definition at line 47 of file config.h.

4.1.1.16 #define PACKAGE_BUGREPORT "mirabeaj@gmail.com"

Definition at line 50 of file config.h.

4.1.1.17 #define PACKAGE_NAME "libg15render"

Definition at line 53 of file config.h.

4.1.1.18 #define PACKAGE_STRING "libg15render 1.2"

Definition at line 56 of file config.h.

4.1.1.19 #define PACKAGE_TARNAME "libg15render"

Definition at line 59 of file config.h.

4.1.1.20 #define PACKAGE_VERSION "1.2"

Definition at line 62 of file config.h.

4.1.1.21 #define STDC_HEADERS 1

Definition at line 65 of file config.h.

4.1.1.22 #define TTF_SUPPORT 1

Definition at line 68 of file config.h.

4.1.1.23 #define VERSION "1.2"

Definition at line 71 of file config.h.

4.2 libg15render.h File Reference

```
#include <string.h>
#include <ft2build.h>
```

Data Structures

- struct **g15canvas**

  This structure holds the data need to render objects to the LCD screen.

Macros

- #define **BYTE_SIZE** 8
- #define **G15_BUFFER_LEN** 1048
- #define **G15_COLOR_BLACK** 1
- #define **G15_COLOR_WHITE** 0
- #define **G15_LCD_HEIGHT** 43
- #define **G15_LCD_OFFSET** 32
- #define **G15_LCD_WIDTH** 160
- #define **G15_MAX_FACE** 5
- #define **G15_PIXEL_FILL** 1
- #define **G15_PIXEL_NOFILL** 0
- #define **G15_TEXT_LARGE** 2
- #define **G15_TEXT_MED** 1
- #define **G15_TEXT_SMALL** 0

Typedefs

- typedef struct **g15canvas** **g15canvas**

  This structure holds the data need to render objects to the LCD screen.

Functions

- void **g15r_clearScreen** (**g15canvas** ∗canvas, int color)

  Fills the screen with pixels of color.
- void **g15r_drawBar** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int color, int num, int max, int type)

  Draws a completion bar.
- void **g15r_drawBigNum** (**g15canvas** ∗canvas, unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2, int color, int num)

  Draw a large number.
- void **g15r_drawCircle** (**g15canvas** ∗canvas, int x, int y, int r, int fill, int color)

  Draws a circle centered at (x, y) with a radius of r.
- void **g15r_drawIcon** (**g15canvas** ∗canvas, char ∗buf, int my_x, int my_y, int width, int height)

  Draw an icon to the screen from a wbmp buffer.
- void **g15r_drawLine** (**g15canvas** ∗canvas, int px1, int py1, int px2, int py2, const int color)

  Draws a line from (px1, py1) to (px2, py2)
- void **g15r_drawRoundBox** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int fill, int color)

  Draws a box with rounded corners bounded by (x1, y1) and (x2, y2)
- void **g15r_drawSprite** (**g15canvas** ∗canvas, char ∗buf, int my_x, int my_y, int width, int height, int start_x, int start_y, int total_width)

  Draw a sprite to the screen from a wbmp buffer.
- int **g15r_getPixel** (**g15canvas** ∗canvas, unsigned int x, unsigned int y)

  Gets the value of the pixel at (x, y)
- void **g15r_initCanvas** (**g15canvas** ∗canvas)

  Clears the canvas and resets the mode switches.
- int **g15r_loadWbmpSplash** (**g15canvas** ∗canvas, char ∗filename)

  Draw a splash screen from 160x43 wbmp file.
- char ∗ **g15r_loadWbmpToBuf** (char ∗filename, int ∗img_width, int ∗img_height)

  Load a wbmp file into a buffer.
- void **g15r_pixelBox** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int color, int thick, int fill)

  Draws a box bounded by (x1, y1) and (x2, y2)
- void **g15r_pixelOverlay** (**g15canvas** ∗canvas, int x1, int y1, int width, int height, short colormap[ ])

  Overlays a bitmap of size width x height starting at (x1, y1)
- void **g15r_pixelReverseFill** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int fill, int color)

  Fills an area bounded by (x1, y1) and (x2, y2)
- void **g15r_renderCharacterLarge** (**g15canvas** ∗canvas, int x, int y, unsigned char character, unsigned int sx, unsigned int sy)

  Renders a character in the large font at (x, y)
- void **g15r_renderCharacterMedium** (**g15canvas** ∗canvas, int x, int y, unsigned char character, unsigned int sx, unsigned int sy)

  Renders a character in the meduim font at (x, y)
- void **g15r_renderCharacterSmall** (**g15canvas** ∗canvas, int x, int y, unsigned char character, unsigned int sx, unsigned int sy)

  Renders a character in the small font at (x, y)
- void **g15r_renderString** (**g15canvas** ∗canvas, unsigned char stringOut[ ], int row, int size, unsigned int sx, unsigned int sy)

  Renders a string with font size in row.
- void **g15r_setPixel** (**g15canvas** ∗canvas, unsigned int x, unsigned int y, int val)

  Sets the value of the pixel at (x, y)
- void **g15r_ttfLoad** (**g15canvas** ∗canvas, char ∗fontname, int fontsize, int face_num)

  Loads a font through the FreeType2 library.
- void **g15r_ttfPrint** (**g15canvas** ∗canvas, int x, int y, int fontsize, int face_num, int color, int center, char ∗print_string)

  Prints a string in a given font.

Variables

- unsigned char **fontdata_6x4** [ ]

  Font data for the small (6x4) font.
- unsigned char **fontdata_7x5** [ ]

  Font data for the medium (7x5) font.
- unsigned char **fontdata_8x8** [ ]

  Font data for the large (8x8) font.

4.2.1 Macro Definition Documentation

4.2.1.1 #define BYTE_SIZE 8

Definition at line 21 of file libg15render.h.

Referenced by g15r_drawIcon(), g15r_drawSprite(), g15r_getPixel(), g15r_loadWbmpToBuf(), and g15r_setPixel().

4.2.1.2 #define G15_BUFFER_LEN 1048

Definition at line 22 of file libg15render.h.

Referenced by g15r_clearScreen(), g15r_initCanvas(), and g15r_loadWbmpSplash().

4.2.1.3 #define G15_COLOR_BLACK 1

Definition at line 27 of file libg15render.h.

Referenced by g15r_drawRoundBox(), g15r_pixelOverlay(), g15r_renderCharacterLarge(), g15r_render↩
CharacterMedium(), and g15r_renderCharacterSmall().

4.2.1.4 #define G15_COLOR_WHITE 0

Definition at line 26 of file libg15render.h.

Referenced by g15r_drawRoundBox(), g15r_pixelOverlay(), g15r_renderCharacterLarge(), g15r_render↩
CharacterMedium(), and g15r_renderCharacterSmall().

4.2.1.5 #define G15_LCD_HEIGHT 43

Definition at line 24 of file libg15render.h.

Referenced by g15r_getPixel(), and g15r_setPixel().

4.2.1.6 #define G15_LCD_OFFSET 32

Definition at line 23 of file libg15render.h.

4.2.1.7 #define G15_LCD_WIDTH 160

Definition at line 25 of file libg15render.h.

Referenced by g15r_getPixel(), and g15r_setPixel().

4.2.1.8 #define G15_MAX_FACE 5

Definition at line 33 of file libg15render.h.

Referenced by g15r_ttfLoad().

4.2.1.9 #define G15_PIXEL_FILL 1

Definition at line 32 of file libg15render.h.

4.2.1.10 #define G15_PIXEL_NOFILL 0

Definition at line 31 of file libg15render.h.

4.2.1.11 #define G15_TEXT_LARGE 2

Definition at line 30 of file libg15render.h.

Referenced by g15r_renderString().

4.2.1.12 #define G15_TEXT_MED 1

Definition at line 29 of file libg15render.h.

Referenced by g15r_renderString().

4.2.1.13 #define G15_TEXT_SMALL 0

Definition at line 28 of file libg15render.h.

Referenced by g15r_renderString().

4.2.2 Typedef Documentation

4.2.2.1 typedef struct g15canvas g15canvas

This structure holds the data need to render objects to the LCD screen.

4.2.3 Function Documentation

4.2.3.1 void g15r_clearScreen (g15canvas * *canvas,* int *color*)

Fills the screen with pixels of color.

Clears the screen and fills it with pixels of color

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
color	Screen will be filled with this color.

Definition at line 80 of file screen.c.

References g15canvas::buffer, and G15_BUFFER_LEN.

```
{
  memset (canvas->buffer, (color ? 0xFF : 0), G15_BUFFER_LEN);
}
```

4.2.3.2 void g15r_drawBar (g15canvas ∗ *canvas*, int *x1*, int *y1*, int *x2*, int *y2*, int *color*, int *num*, int *max*, int *type*)

Draws a completion bar.

Given a maximum value, and a value between 0 and that maximum value, calculate and draw a bar showing that percentage.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the bar.
y1	Defines uppermost bound of the bar.
x2	Defines rightmost bound of the bar.
y2	Defines bottommost bound of the bar.
color	The bar will be drawn this color.
num	Number of units relative to max filled.
max	Number of units equal to 100% filled.
type	Type of bar. 1=solid bar, 2=solid bar with border, 3 = solid bar with I-frame.

Definition at line 337 of file pixel.c.

References g15r_drawLine(), and g15r_pixelBox().

```
{
  float len, length;
  int x;
  if (max == 0)
    return;
  if (num > max)
    num = max;

  if (type == 2)
    {
      y1 += 2;
      y2 -= 2;
      x1 += 2;
      x2 -= 2;
    }

  len = ((float) max / (float) num);
  length = (x2 - x1) / len;

  if (type == 1)
    {
      g15r_pixelBox (canvas, x1, y1 - type, x2, y2 + type, color ^ 1, 1, 1);
      g15r_pixelBox (canvas, x1, y1 - type, x2, y2 + type, color, 1, 0);
    }
  else if (type == 2)
    {
      g15r_pixelBox (canvas, x1 - 2, y1 - type, x2 + 2, y2 + type, color ^ 1,
                     1, 1);
      g15r_pixelBox (canvas, x1 - 2, y1 - type, x2 + 2, y2 + type, color, 1,
                     0);
    }
  else if (type == 3)
    {
      g15r_drawLine (canvas, x1, y1 - type, x1, y2 + type, color);
      g15r_drawLine (canvas, x2, y1 - type, x2, y2 + type, color);
      g15r_drawLine (canvas, x1, y1 + ((y2 - y1) / 2), x2,
                     y1 + ((y2 - y1) / 2), color);
    }
  g15r_pixelBox (canvas, x1, y1, (int) ceil (x1 + length), y2, color, 1, 1);
}
```

4.2.3.3 void g15r_drawBigNum (**g15canvas** ∗ *canvas,* unsigned int *x1,* unsigned int *y1,* unsigned int *x2,* unsigned int *y2,* int *color,* int *num*)

Draw a large number.

Draw a large number to a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the number.
y1	Defines uppermost bound of the number.
x2	Defines rightmost bound of the number.
y2	Defines bottommost bound of the number.
num	The number to be drawn.

Definition at line 545 of file pixel.c.

References g15r_pixelBox().

```
{
    x1 += 2;
    x2 -= 2;

    switch(num){
        case 0:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1 +5, y1 +5, x2 -5, y2 - 6, 1 - color, 1, 1);
            break;
        case 1:
            g15r_pixelBox (canvas, x2-5, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1, x2 -5, y2, 1 - color, 1, 1);
            break;
        case 2:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+5, x2 -5, y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+((y2/2)+3), x2 , y2-6, 1 - color, 1, 1);
            break;
        case 3:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+5, x2 -5, y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
            break;
        case 4:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2 -5, y2, 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1, x2-5 , y1+((y2/2)-3), 1 - color, 1, 1);
            break;
        case 5:
```

```
      g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
      g15r_pixelBox (canvas, x1+5, y1+5, x2 , y1+((y2/2)-3), 1 - color, 1, 1);
      g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
      break;
    case 6:
      g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
      g15r_pixelBox (canvas, x1+5, y1+5, x2 , y1+((y2/2)-3), 1 - color, 1, 1);
      g15r_pixelBox (canvas, x1+5, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
      break;
    case 7:
      g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
      g15r_pixelBox (canvas, x1, y1+5, x2 -5, y2, 1 - color, 1, 1);
      break;
    case 8:
      g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
      g15r_pixelBox (canvas, x1+5, y1+5, x2-5 , y1+((y2/2)-3), 1 - color, 1, 1);
      g15r_pixelBox (canvas, x1+5, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
      break;
    case 9:
      g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
      g15r_pixelBox (canvas, x1+5, y1+5, x2-5 , y1+((y2/2)-3), 1 - color, 1, 1);
      g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2-5 , y2, 1 - color, 1, 1);
      break;
    case 10:
      g15r_pixelBox (canvas, x2-5, y1+5, x2, y1+10 , color, 1, 1);
      g15r_pixelBox (canvas, x2-5, y2-10, x2, y2-5 , color, 1, 1);
      break;
    case 11:
      g15r_pixelBox (canvas, x1, y1+((y2/2)-2), x2, y1+((y2/2)+2), color, 1, 1);
      break;
    case 12:
      g15r_pixelBox (canvas, x2-5, y2-5, x2, y2 , color, 1, 1);
      break;
  }
}
```

4.2.3.4 void g15r_drawCircle (**g15canvas** ∗ *canvas,* int *x,* int *y,* int *r,* int *fill,* int *color*)

Draws a circle centered at (x, y) with a radius of r.

Draws a circle centered at (x, y) with a radius of r.

The circle will be filled if fill != 0.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	Defines horizontal center of the circle.
y	Defines vertical center of circle.
r	Defines radius of circle.
fill	The circle will be filled with color if fill != 0.
color	Lines defining the circle will be drawn this color.

Definition at line 203 of file pixel.c.

References g15r_drawLine(), and g15r_setPixel().

```
{
 int xx, yy, dd;

 xx = 0;
 yy = r;
 dd = 2 * (1 - r);

 while (yy >= 0)
  {
   if (!fill)
    {
          g15r_setPixel (canvas, x + xx, y - yy, color);
          g15r_setPixel (canvas, x + xx, y + yy, color);
          g15r_setPixel (canvas, x - xx, y - yy, color);
          g15r_setPixel (canvas, x - xx, y + yy, color);
        }
      else
        {
          g15r_drawLine (canvas, x - xx, y - yy, x + xx, y - yy, color);
          g15r_drawLine (canvas, x - xx, y + yy, x + xx, y + yy, color);
        }
      if (dd + yy > 0)
        {
          yy--;
          dd = dd - (2 * yy + 1);
        }
      if (xx > dd)
        {
          xx++;
          dd = dd + (2 * xx + 1);
        }
    }
}
```

4.2.3.5 void g15r_drawIcon (**g15canvas** * *canvas,* char * *buf,* int *my_x,* int *my_y,* int *width,* int *height*)

Draw an icon to the screen from a wbmp buffer.

Draw an icon to a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated in is found.
buf	A pointer to the buffer holding the icon to be displayed.
my_x	Leftmost boundary of image.
my_y	Topmost boundary of image.
width	Width of the image in buf.
height	Height of the image in buf.

Definition at line 411 of file pixel.c.

References BYTE_SIZE, and g15r_setPixel().

```
{
    int y,x,val;
    unsigned int pixel_offset = 0;
    unsigned int byte_offset, bit_offset;

    for (y=0; y < height - 1; y++)
      for (x=0; x < width - 1; x++)
        {
                pixel_offset = y * width + x;
                byte_offset = pixel_offset / BYTE_SIZE;
                bit_offset = 7 - (pixel_offset % BYTE_SIZE);

                val = (buf[byte_offset] & (1 << bit_offset)) >> bit_offset;
                g15r_setPixel (canvas, x + my_x, y + my_y, val);
        }
}
```

4.2.3.6 void g15r_drawLine (**g15canvas** * *canvas,* int *px1,* int *py1,* int *px2,* int *py2,* const int *color*)

Draws a line from (px1, py1) to (px2, py2)

A line of color is drawn from (px1, py1) to (px2, py2).

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
px1	X component of point 1.
py1	Y component of point 1.
px2	X component of point 2.
py2	Y component of point 2.
color	Line will be drawn this color.

Definition at line 99 of file pixel.c.

References g15r_setPixel(), and swap().

Referenced by g15r_drawBar(), g15r_drawCircle(), g15r_drawRoundBox(), and g15r_pixelBox().

```
{
  /*
   * Bresenham's Line Algorithm
   * http://en.wikipedia.org/wiki/Bresenham's_algorithm
   */

  int steep = 0;

  if (abs (py2 - py1) > abs (px2 - px1))
    steep = 1;

  if (steep)
    {
      swap (&px1, &py1);
      swap (&px2, &py2);
    }

  if (px1 > px2)
    {
      swap (&px1, &px2);
      swap (&py1, &py2);
    }

  int dx = px2 - px1;
  int dy = abs (py2 - py1);

  int error = 0;
  int y = py1;
  int ystep = (py1 < py2) ? 1 : -1;
  int x = 0;

  for (x = px1; x <= px2; ++x)
    {
      if (steep)
        g15r_setPixel (canvas, y, x, color);
      else
        g15r_setPixel (canvas, x, y, color);

      error += dy;
      if (2 * error >= dx)
        {
          y += ystep;
          error -= dx;
        }
    }
}
```

4.2.3.7 void g15r_drawRoundBox (**g15canvas** * *canvas*, int *x1*, int *y1*, int *x2*, int *y2*, int *fill*, int *color*)

Draws a box with rounded corners bounded by (x1, y1) and (x2, y2)

Draws a rounded box around the area bounded by (x1, y1) and (x2, y2).

The box will be filled if fill != 0.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the box.
y1	Defines uppermost bound of the box.
x2	Defines rightmost bound of the box.
y2	Defines bottommost bound of the box.
fill	The box will be filled with color if fill != 0.
color	Lines defining the box will be drawn this color.

Definition at line 252 of file pixel.c.

References G15_COLOR_BLACK, G15_COLOR_WHITE, g15r_drawLine(), and g15r_setPixel().

```
{
  int y, shave = 3;

  if (shave > (x2 - x1) / 2)
    shave = (x2 - x1) / 2;
  if (shave > (y2 - y1) / 2)
    shave = (y2 - y1) / 2;

  if ((x1 != x2) && (y1 != y2))
    {
      if (fill)
        {
          g15r_drawLine (canvas, x1 + shave, y1, x2 - shave, y1, color);
          for (y = y1 + 1; y < y1 + shave; y++)
            g15r_drawLine (canvas, x1 + 1, y, x2 - 1, y, color);
          for (y = y1 + shave; y <= y2 - shave; y++)
            g15r_drawLine (canvas, x1, y, x2, y, color);
          for (y = y2 - shave + 1; y < y2; y++)
            g15r_drawLine (canvas, x1 + 1, y, x2 - 1, y, color);
          g15r_drawLine (canvas, x1 + shave, y2, x2 - shave, y2, color);
          if (shave == 4)
            {
              g15r_setPixel (canvas, x1 + 1, y1 + 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
              g15r_setPixel (canvas, x1 + 1, y2 - 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
              g15r_setPixel (canvas, x2 - 1, y1 + 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
              g15r_setPixel (canvas, x2 - 1, y2 - 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
            }
        }
      else
        {
          g15r_drawLine (canvas, x1 + shave, y1, x2 - shave, y1, color);
          g15r_drawLine (canvas, x1, y1 + shave, x1, y2 - shave, color);
          g15r_drawLine (canvas, x2, y1 + shave, x2, y2 - shave, color);
          g15r_drawLine (canvas, x1 + shave, y2, x2 - shave, y2, color);
          if (shave > 1)
            {
              g15r_drawLine (canvas, x1 + 1, y1 + 1, x1 + shave - 1, y1 + 1,
                             color);
              g15r_drawLine (canvas, x2 - shave + 1, y1 + 1, x2 - 1, y1 + 1,
                             color);
              g15r_drawLine (canvas, x1 + 1, y2 - 1, x1 + shave - 1, y2 - 1,
                             color);
              g15r_drawLine (canvas, x2 - shave + 1, y2 - 1, x2 - 1, y2 - 1,
                             color);
              g15r_drawLine (canvas, x1 + 1, y1 + 1, x1 + 1, y1 + shave - 1,
                             color);
              g15r_drawLine (canvas, x1 + 1, y2 - 1, x1 + 1, y2 - shave + 1,
                             color);
              g15r_drawLine (canvas, x2 - 1, y1 + 1, x2 - 1, y1 + shave - 1,
                             color);
            g15r_drawLine (canvas, x2 - 1, y2 - 1, x2 - 1, y2 - shave + 1,
                           color);
        }
      }
    }
}
```

4.2.3.8 void g15r_drawSprite (g15canvas ∗ *canvas,* char ∗ *buf,* int *my_x,* int *my_y,* int *width,* int *height,* int *start_x,* int *start_y,* int *total_width*)

Draw a sprite to the screen from a wbmp buffer.

Draw a sprite to a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated in is found.
buf	A pointer to the buffer holding a set of sprites.
my_x	Leftmost boundary of image.
my_y	Topmost boundary of image.
width	Width of the sprite.
height	Height of the sprite.
start_x	X offset for reading sprite from buf.
start_y	Y offset for reading sprite from buf.
total_width	Width of the set of sprites held in buf.

Definition at line 443 of file pixel.c.

References BYTE_SIZE, and g15r_setPixel().

```
{
    int y,x,val;
    unsigned int pixel_offset = 0;
    unsigned int byte_offset, bit_offset;

    for (y=0; y < height - 1; y++)
      for (x=0; x < width - 1; x++)
        {
                pixel_offset = (y + start_y) * total_width + (x + start_x);
                byte_offset = pixel_offset / BYTE_SIZE;
                bit_offset = 7 - (pixel_offset % BYTE_SIZE);

                val = (buf[byte_offset] & (1 << bit_offset)) >> bit_offset;
                g15r_setPixel (canvas, x + my_x, y + my_y, val);
        }
}
```

4.2.3.9 int g15r_getPixel (g15canvas ∗ *canvas,* unsigned int *x,* unsigned int *y*)

Gets the value of the pixel at (x, y)

Retrieves the value of the pixel at (x, y)

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	X offset for pixel to be retrieved.
y	Y offset for pixel to be retrieved.

Definition at line 29 of file screen.c.

References g15canvas::buffer, BYTE_SIZE, G15_LCD_HEIGHT, and G15_LCD_WIDTH.

Referenced by g15r_pixelReverseFill(), and g15r_setPixel().

```
{
  if (x >= G15_LCD_WIDTH || y >= G15_LCD_HEIGHT)
    return 0;

  unsigned int pixel_offset = y * G15_LCD_WIDTH + x;
  unsigned int byte_offset = pixel_offset / BYTE_SIZE;
  unsigned int bit_offset = 7 - (pixel_offset % BYTE_SIZE);

  return (canvas->buffer[byte_offset] & (1 << bit_offset)) >> bit_offset;
}
```

4.2.3.10 void g15r_initCanvas (**g15canvas** * *canvas*)

Clears the canvas and resets the mode switches.

Clears the screen and resets the mode values for a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct

Definition at line 91 of file screen.c.

References g15canvas::buffer, g15canvas::ftLib, G15_BUFFER_LEN, g15canvas::mode_cache, g15canvas::mode_reverse, and g15canvas::mode_xor.

```
{
  memset (canvas->buffer, 0, G15_BUFFER_LEN);
  canvas->mode_cache = 0;
  canvas->mode_reverse = 0;
  canvas->mode_xor = 0;
#ifdef TTF_SUPPORT
  if (FT_Init_FreeType (&canvas->ftLib))
    printf ("Freetype couldnt initialise\n");
#endif
}
```

4.2.3.11 int g15r_loadWbmpSplash (**g15canvas** * *canvas*, char * *filename*)

Draw a splash screen from 160x43 wbmp file.

wbmp splash screen loader - assumes image is 160x43

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
filename	A string holding the path to the wbmp to be displayed.

Definition at line 387 of file pixel.c.

References g15canvas::buffer, G15_BUFFER_LEN, and g15r_loadWbmpToBuf().

```
{
    int width=0, height=0;
    char *buf;

    buf = g15r_loadWbmpToBuf(filename,
                             &width,
                             &height);

    memcpy (canvas->buffer, buf, G15_BUFFER_LEN);
    return 0;
}
```

4.2.3.12 char* g15r_loadWbmpToBuf (char * *filename,* int * *img_width,* int * *img_height*)

Load a wbmp file into a buffer.

basic wbmp loader - loads a wbmp image into a buffer.

Parameters

filename	A string holding the path to the wbmp to be loaded.
img_width	A pointer to an int that will hold the image width on return.
img_height	A pointer to an int that will hold the image height on return.

Definition at line 469 of file pixel.c.

References BYTE_SIZE.

Referenced by g15r_loadWbmpSplash().

```
{
    int wbmp_fd;
    int retval;
    int x,y,val;
    char *buf;
    unsigned int buflen,header=4;
    unsigned char headerbytes[5];
    unsigned int pixel_offset = 0;
    unsigned int byte_offset, bit_offset;

    wbmp_fd=open(filename,O_RDONLY);
    if(!wbmp_fd){
        return NULL;
    }

    retval=read(wbmp_fd,headerbytes,5);

    if(retval){
        if (headerbytes[2] & 1) {
            *img_width = ((unsigned char)headerbytes[2] ^ 1) | (unsigned char)headerbytes[3];
            *img_height = headerbytes[4];
            header = 5;
        } else {
            *img_width = headerbytes[2];
            *img_height = headerbytes[3];
        }

        int byte_width = *img_width / 8;
        if (*img_width %8)
          byte_width++;

        buflen = byte_width * (*img_height);

        buf = (char *)malloc (buflen);
        if (buf == NULL)
          return NULL;
```

```

      if (header == 4)
        buf[0]=headerbytes[4];

      retval=read(wbmp_fd,buf+(5-header),buflen);

      close(wbmp_fd);
  }

  /* now invert the image */
  for (y = 0; y < *img_height; y++)
    for (x = 0; x < *img_width; x++)
      {
              pixel_offset = y * (*img_width) + x;
              byte_offset = pixel_offset / BYTE_SIZE;
              bit_offset = 7 - (pixel_offset % BYTE_SIZE);

              val = (buf[byte_offset] & (1 << bit_offset)) >> bit_offset;

              if (!val)
                buf[byte_offset] = buf[byte_offset] | 1 << bit_offset;
              else
                buf[byte_offset] = buf[byte_offset] & ~(1 << bit_offset);
      }

  return buf;
}
```

4.2.3.13 void g15r_pixelBox (**g15canvas** ∗ *canvas,* int *x1,* int *y1,* int *x2,* int *y2,* int *color,* int *thick,* int *fill*)

Draws a box bounded by (x1, y1) and (x2, y2)

Draws a box around the area bounded by (x1, y1) and (x2, y2).

The box will be filled if fill != 0 and the sides will be thick pixels wide.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the box.
y1	Defines uppermost bound of the box.
x2	Defines rightmost bound of the box.
y2	Defines bottommost bound of the box.
color	Lines defining the box will be drawn this color.
thick	Lines defining the box will be this many pixels thick.
fill	The box will be filled with color if fill != 0.

Definition at line 163 of file pixel.c.

References g15r_drawLine(), and g15r_setPixel().

Referenced by g15r_drawBar(), and g15r_drawBigNum().

```
{
  int i = 0;
  for (i = 0; i < thick; ++i)
    {
      g15r_drawLine (canvas, x1, y1, x2, y1, color);    /* Top    */
      g15r_drawLine (canvas, x1, y1, x1, y2, color);    /* Left   */
      g15r_drawLine (canvas, x2, y1, x2, y2, color);    /* Right  */
      g15r_drawLine (canvas, x1, y2, x2, y2, color);    /* Bottom */
      x1++;
      y1++;
      x2--;
      y2--;
  }

 int x = 0, y = 0;

 if (fill)
  {
   for (x = x1; x <= x2; ++x)
    for (y = y1; y <= y2; ++y)
     g15r_setPixel (canvas, x, y, color);
  }

}
```

4.2.3.14 void g15r_pixelOverlay (**g15canvas** ∗ *canvas,* int *x1,* int *y1,* int *width,* int *height,* short *colormap[ ]*)

Overlays a bitmap of size width x height starting at (x1, y1)

A 1-bit bitmap defined in colormap[] is drawn to the canvas with an upper left corner at (x1, y1) and a lower right corner at (x1+width, y1+height).

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines the leftmost bound of the area to be drawn.
y1	Defines the uppermost bound of the area to be drawn.
width	Defines the width of the bitmap to be drawn.
height	Defines the height of the bitmap to be drawn.
colormap	An array containing width∗height entries of value 0 for pixel off or != 0 for pixel on.

Definition at line 74 of file pixel.c.

References G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

```
{
 int i = 0;

 for (i = 0; i < (width * height); ++i)
  {
   int color = (colormap[i] ? G15_COLOR_BLACK : G15_COLOR_WHITE);
   int x = x1 + i % width;
   int y = y1 + i / width;
   g15r_setPixel (canvas, x, y, color);
  }
}
```

4.2.3.15 void g15r_pixelReverseFill (**g15canvas** ∗ *canvas,* int *x1,* int *y1,* int *x2,* int *y2,* int *fill,* int *color*)

Fills an area bounded by (x1, y1) and (x2, y2)

The area with an upper left corner at (x1, y1) and lower right corner at (x2, y2) will be filled with color if fill>0 or the current contents of the area will be reversed if fill==0.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of area to be filled.
y1	Defines uppermost bound of area to be filled.
x2	Defines rightmost bound of area to be filled.
y2	Defines bottommost bound of area to be filled.
fill	Area will be filled with color if fill != 0, else contents of area will have color values reversed.
color	If fill != 0, then area will be filled if color == 1 and emptied if color == 0.

Definition at line 45 of file pixel.c.

References g15r_getPixel(), and g15r_setPixel().

```
{
  int x = 0;
  int y = 0;

  for (x = x1; x <= x2; ++x)
    {
      for (y = y1; y <= y2; ++y)
        {
          if (!fill)
            color = !g15r_getPixel (canvas, x, y);
          g15r_setPixel (canvas, x, y, color);
        }
    }
}
```

4.2.3.16 void g15r_renderCharacterLarge (g15canvas ∗ *canvas*, int *x*, int *y*, unsigned char *character*, unsigned int *sx*, unsigned int *sy*)

Renders a character in the large font at (x, y)

Definition at line 22 of file text.c.

References fontdata_8x8, G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

Referenced by g15r_renderString().

```
{
  int helper = character * 8;   /* for our font which is 8x8 */

  int top_left_pixel_x = sx + col * (8);        /* 1 pixel spacing */
  int top_left_pixel_y = sy + row * (8);        /* once again 1 pixel spacing */

  int x, y;
  for (y = 0; y < 8; ++y)
    {
      for (x = 0; x < 8; ++x)
        {
          char font_entry = fontdata_8x8[helper + y];

          if (font_entry & 1 << (7 - x))
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_BLACK);
          else
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_WHITE);

        }
    }
}
```

4.2.3.17 void g15r_renderCharacterMedium (g15canvas ∗ *canvas*, int *x*, int *y*, unsigned char *character*, unsigned int *sx*, unsigned int *sy*)

Renders a character in the meduim font at (x, y)

Definition at line 50 of file text.c.

References fontdata_7x5, G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

Referenced by g15r_renderString().

```
{
  int helper = character * 7 * 5;       /* for our font which is 6x4 */

  int top_left_pixel_x = sx + col * (5);        /* 1 pixel spacing */
  int top_left_pixel_y = sy + row * (7);        /* once again 1 pixel spacing */

  int x, y;
  for (y = 0; y < 7; ++y)
    {
      for (x = 0; x < 5; ++x)
        {
          char font_entry = fontdata_7x5[helper + y * 5 + x];
          if (font_entry)
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_BLACK);
          else
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_WHITE);

        }
    }
}
```

4.2.3.18 void g15r_renderCharacterSmall (g15canvas ∗ *canvas,* int *x,* int *y,* unsigned char *character,* unsigned int *sx,* unsigned int *sy*)

Renders a character in the small font at (x, y)

Definition at line 77 of file text.c.

References fontdata_6x4, G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

Referenced by g15r_renderString().

```
{
  int helper = character * 6 * 4;       /* for our font which is 6x4 */

  int top_left_pixel_x = sx + col * (4);        /* 1 pixel spacing */
  int top_left_pixel_y = sy + row * (6);        /* once again 1 pixel spacing */

  int x, y;
  for (y = 0; y < 6; ++y)
    {
      for (x = 0; x < 4; ++x)
        {
          char font_entry = fontdata_6x4[helper + y * 4 + x];
          if (font_entry)
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_BLACK);
          else
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_WHITE);

        }
    }
}
```

4.2.3.19 void g15r_renderString (g15canvas ∗ *canvas,* unsigned char *stringOut[ ],* int *row,* int *size,* unsigned int *sx,* unsigned int *sy*)

Renders a string with font size in row.

Definition at line 104 of file text.c.

References G15_TEXT_LARGE, G15_TEXT_MED, G15_TEXT_SMALL, g15r_renderCharacterLarge(), g15r_↩ renderCharacterMedium(), and g15r_renderCharacterSmall().

```
{

  int i = 0;
  for (i; stringOut[i] != NULL; ++i)
    {
      switch (size)
        {
        case G15_TEXT_SMALL:
          {
            g15r_renderCharacterSmall (canvas, i, row, stringOut[i], sx, sy);
            break;
          }
        case G15_TEXT_MED:
          {
            g15r_renderCharacterMedium (canvas, i, row, stringOut[i], sx, sy);
            break;
          }
        case G15_TEXT_LARGE:
          {
            g15r_renderCharacterLarge (canvas, i, row, stringOut[i], sx, sy);
            break;
          }
        default:
          break;
        }
    }

}
```

4.2.3.20 void g15r_setPixel (**g15canvas** ∗ *canvas,* unsigned int *x,* unsigned int *y,* int *val*)

Sets the value of the pixel at (x, y)

Sets the value of the pixel at (x, y)

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	X offset for pixel to be set.
y	Y offset for pixel to be set.
val	Value to which pixel should be set.

Definition at line 50 of file screen.c.

References g15canvas::buffer, BYTE_SIZE, G15_LCD_HEIGHT, G15_LCD_WIDTH, g15r_getPixel(), g15canvas↩ ::mode_reverse, and g15canvas::mode_xor.

Referenced by draw_ttf_char(), g15r_drawCircle(), g15r_drawIcon(), g15r_drawLine(), g15r_drawRoundBox(), g15r_drawSprite(), g15r_pixelBox(), g15r_pixelOverlay(), g15r_pixelReverseFill(), g15r_renderCharacterLarge(), g15r_renderCharacterMedium(), and g15r_renderCharacterSmall().

```
{
  if (x >= G15_LCD_WIDTH || y >= G15_LCD_HEIGHT)
    return;

  unsigned int pixel_offset = y * G15_LCD_WIDTH + x;
  unsigned int byte_offset = pixel_offset / BYTE_SIZE;
  unsigned int bit_offset = 7 - (pixel_offset % BYTE_SIZE);

  if (canvas->mode_xor)
    val ^= g15r_getPixel (canvas, x, y);
  if (canvas->mode_reverse)
    val = !val;

  if (val)
    canvas->buffer[byte_offset] =
      canvas->buffer[byte_offset] | 1 << bit_offset;
  else
    canvas->buffer[byte_offset] =
      canvas->buffer[byte_offset] & ~(1 << bit_offset);

}
```

4.2.3.21 void g15r_ttfLoad (**g15canvas** ∗ *canvas,* char ∗ *fontname,* int *fontsize,* int *face_num*)

Loads a font through the FreeType2 library.

Load a font for use with FreeType2 font support

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
fontname	Absolute pathname to font file to be loaded.
fontsize	Size in points for font to be loaded.
face_num	Slot into which font face will be loaded.

Definition at line 145 of file text.c.

References g15canvas::ftLib, G15_MAX_FACE, g15canvas::ttf_face, and g15canvas::ttf_fontsize.

```
{
  int errcode = 0;

  if (face_num < 0)
    face_num = 0;
  if (face_num > G15_MAX_FACE)
    face_num = G15_MAX_FACE;

  if (canvas->ttf_fontsize[face_num])
    FT_Done_Face (canvas->ttf_face[face_num][0]);       /* destroy the last face */

  if (!canvas->ttf_fontsize[face_num] && !fontsize)
    canvas->ttf_fontsize[face_num] = 10;
  else
    canvas->ttf_fontsize[face_num] = fontsize;

  errcode =
    FT_New_Face (canvas->ftLib, fontname, 0, &canvas->ttf_face[face_num][0]);
  if (errcode)
    {
      canvas->ttf_fontsize[face_num] = 0;
    }
  else
    {
      if (canvas->ttf_fontsize[face_num]
          && FT_IS_SCALABLE (canvas->ttf_face[face_num][0]))
        errcode =
          FT_Set_Char_Size (canvas->ttf_face[face_num][0], 0,
                            canvas->ttf_fontsize[face_num] * 64, 90, 0);
    }
}
```

4.2.3.22 void g15r_ttfPrint (**g15canvas** ∗ *canvas,* int *x,* int *y,* int *fontsize,* int *face_num,* int *color,* int *center,* char ∗ *print_string*)

Prints a string in a given font.

Render a string with a FreeType2 font

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	initial x position for string.
y	initial y position for string.
fontsize	Size of string in points.
face_num	Font to be used is loaded in this slot.
color	Text will be drawn this color.
center	Text will be centered if center == 1 and right justified if center == 2.
print_string	Pointer to the string to be printed.

Definition at line 283 of file text.c.

References calc_ttf_centering(), calc_ttf_right_justify(), calc_ttf_true_ypos(), draw_ttf_str(), g15canvas::ttf_face, and g15canvas::ttf_fontsize.

```
{
  int errcode = 0;

  if (canvas->ttf_fontsize[face_num])
    {
      if (fontsize > 0 && FT_IS_SCALABLE (canvas->ttf_face[face_num][0]))
        {
          canvas->ttf_fontsize[face_num] = fontsize;
          int errcode =
            FT_Set_Pixel_Sizes (canvas->ttf_face[face_num][0], 0,
                                canvas->ttf_fontsize[face_num]);
          if (errcode)
            printf ("Trouble setting the Glyph size!\n");
        }
      y =
        calc_ttf_true_ypos (canvas->ttf_face[face_num][0], y,
                            canvas->ttf_fontsize[face_num]);
      if (center == 1)
        x = calc_ttf_centering (canvas->ttf_face[face_num][0], print_string);
      else if (center == 2)
        x = calc_ttf_right_justify (canvas->ttf_face[face_num][0], print_string);
      draw_ttf_str (canvas, print_string, x, y, color,
                    canvas->ttf_face[face_num][0]);
    }
}
```

4.2.4 Variable Documentation

4.2.4.1 unsigned char fontdata_6x4[ ]

Font data for the small (6x4) font.

Referenced by g15r_renderCharacterSmall().

4.2.4.2 unsigned char fontdata_7x5[ ]

Font data for the medium (7x5) font.

Referenced by g15r_renderCharacterMedium().

4.2.4.3 unsigned char fontdata_8x8[ ]

Font data for the large (8x8) font.

Referenced by g15r_renderCharacterLarge().

4.3 pixel.c File Reference

```
#include <fcntl.h>
#include "libg15render.h"
```

Functions

- void **g15r_drawBar** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int color, int num, int max, int type)
  Draws a completion bar.
- void **g15r_drawBigNum** (**g15canvas** ∗canvas, unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2, int color, int num)
  Draw a large number.
- void **g15r_drawCircle** (**g15canvas** ∗canvas, int x, int y, int r, int fill, int color)
  Draws a circle centered at (x, y) with a radius of r.
- void **g15r_drawIcon** (**g15canvas** ∗canvas, char ∗buf, int my_x, int my_y, int width, int height)
  Draw an icon to the screen from a wbmp buffer.
- void **g15r_drawLine** (**g15canvas** ∗canvas, int px1, int py1, int px2, int py2, const int color)
  Draws a line from (px1, py1) to (px2, py2)
- void **g15r_drawRoundBox** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int fill, int color)
  Draws a box with rounded corners bounded by (x1, y1) and (x2, y2)
- void **g15r_drawSprite** (**g15canvas** ∗canvas, char ∗buf, int my_x, int my_y, int width, int height, int start_x, int start_y, int total_width)
  Draw a sprite to the screen from a wbmp buffer.
- int **g15r_loadWbmpSplash** (**g15canvas** ∗canvas, char ∗filename)
  Draw a splash screen from 160x43 wbmp file.
- char ∗ **g15r_loadWbmpToBuf** (char ∗filename, int ∗img_width, int ∗img_height)
  Load a wbmp file into a buffer.
- void **g15r_pixelBox** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int color, int thick, int fill)
  Draws a box bounded by (x1, y1) and (x2, y2)
- void **g15r_pixelOverlay** (**g15canvas** ∗canvas, int x1, int y1, int width, int height, short colormap[ ])
  Overlays a bitmap of size width x height starting at (x1, y1)
- void **g15r_pixelReverseFill** (**g15canvas** ∗canvas, int x1, int y1, int x2, int y2, int fill, int color)
  Fills an area bounded by (x1, y1) and (x2, y2)
- void **swap** (int ∗x, int ∗y)

4.3.1 Function Documentation

4.3.1.1 void g15r_drawBar (g15canvas ∗ *canvas,* int *x1,* int *y1,* int *x2,* int *y2,* int *color,* int *num,* int *max,* int *type*)

Draws a completion bar.

Given a maximum value, and a value between 0 and that maximum value, calculate and draw a bar showing that percentage.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the bar.
y1	Defines uppermost bound of the bar.
x2	Defines rightmost bound of the bar.
y2	Defines bottommost bound of the bar.
color	The bar will be drawn this color.
num	Number of units relative to max filled.
max	Number of units equal to 100% filled.
type	Type of bar. 1=solid bar, 2=solid bar with border, 3 = solid bar with I-frame.

Definition at line 337 of file pixel.c.

References g15r_drawLine(), and g15r_pixelBox().

```
{
  float len, length;
  int x;
  if (max == 0)
    return;
  if (num > max)
    num = max;

  if (type == 2)
    {
      y1 += 2;
      y2 -= 2;
      x1 += 2;
      x2 -= 2;
    }

  len = ((float) max / (float) num);
  length = (x2 - x1) / len;

  if (type == 1)
    {
      g15r_pixelBox (canvas, x1, y1 - type, x2, y2 + type, color ^ 1, 1, 1);
      g15r_pixelBox (canvas, x1, y1 - type, x2, y2 + type, color, 1, 0);
    }
  else if (type == 2)
    {
      g15r_pixelBox (canvas, x1 - 2, y1 - type, x2 + 2, y2 + type, color ^ 1,
                     1, 1);
      g15r_pixelBox (canvas, x1 - 2, y1 - type, x2 + 2, y2 + type, color, 1,
                     0);
    }
  else if (type == 3)
    {
      g15r_drawLine (canvas, x1, y1 - type, x1, y2 + type, color);
      g15r_drawLine (canvas, x2, y1 - type, x2, y2 + type, color);
      g15r_drawLine (canvas, x1, y1 + ((y2 - y1) / 2), x2,
                     y1 + ((y2 - y1) / 2), color);
    }
  g15r_pixelBox (canvas, x1, y1, (int) ceil (x1 + length), y2, color, 1, 1);
}
```

4.3.1.2 void g15r_drawBigNum (g15canvas ∗ *canvas,* unsigned int *x1,* unsigned int *y1,* unsigned int *x2,* unsigned int *y2,* int *color,* int *num*)

Draw a large number.

Draw a large number to a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the number.
y1	Defines uppermost bound of the number.
x2	Defines rightmost bound of the number.
y2	Defines bottommost bound of the number.
num	The number to be drawn.

Definition at line 545 of file pixel.c.

References g15r_pixelBox().

```
{
    x1 += 2;
    x2 -= 2;

    switch(num){
        case 0:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1 +5, y1 +5, x2 -5, y2 - 6, 1 - color, 1, 1);
            break;
        case 1:
            g15r_pixelBox (canvas, x2-5, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1, x2 -5, y2, 1 - color, 1, 1);
            break;
        case 2:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+5, x2 -5, y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+((y2/2)+3), x2 , y2-6, 1 - color, 1, 1);
            break;
        case 3:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+5, x2 -5, y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
            break;
        case 4:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2 -5, y2, 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1, x2-5 , y1+((y2/2)-3), 1 - color, 1, 1);
            break;
        case 5:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+5, x2 , y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
            break;
        case 6:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+5, x2 , y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
            break;
        case 7:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+5, x2 -5, y2, 1 - color, 1, 1);
            break;
        case 8:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+5, x2-5 , y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+((y2/2)+3), x2-5 , y2-6, 1 - color, 1, 1);
            break;
        case 9:
            g15r_pixelBox (canvas, x1, y1, x2, y2 , color, 1, 1);
            g15r_pixelBox (canvas, x1+5, y1+5, x2-5 , y1+((y2/2)-3), 1 - color, 1, 1);
            g15r_pixelBox (canvas, x1, y1+((y2/2)+3), x2-5 , y2, 1 - color, 1, 1);
            break;
        case 10:
            g15r_pixelBox (canvas, x2-5, y1+5, x2, y1+10 , color, 1, 1);
            g15r_pixelBox (canvas, x2-5, y2-10, x2, y2-5 , color, 1, 1);
            break;
        case 11:
            g15r_pixelBox (canvas, x1, y1+((y2/2)-2), x2, y1+((y2/2)+2), color, 1, 1);
            break;
        case 12:
            g15r_pixelBox (canvas, x2-5, y2-5, x2, y2 , color, 1, 1);
            break;
    }
}
```

4.3.1.3 void g15r_drawCircle (**g15canvas** ∗ *canvas,* int *x,* int *y,* int *r,* int *fill,* int *color*)

Draws a circle centered at (x, y) with a radius of r.

Draws a circle centered at (x, y) with a radius of r.

The circle will be filled if fill != 0.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	Defines horizontal center of the circle.
y	Defines vertical center of circle.
r	Defines radius of circle.
fill	The circle will be filled with color if fill != 0.
color	Lines defining the circle will be drawn this color.

Definition at line 203 of file pixel.c.

References g15r_drawLine(), and g15r_setPixel().

```
{
  int xx, yy, dd;

  xx = 0;
  yy = r;
  dd = 2 * (1 - r);

  while (yy >= 0)
    {
      if (!fill)
        {
          g15r_setPixel (canvas, x + xx, y - yy, color);
          g15r_setPixel (canvas, x + xx, y + yy, color);
          g15r_setPixel (canvas, x - xx, y - yy, color);
          g15r_setPixel (canvas, x - xx, y + yy, color);
        }
      else
        {
          g15r_drawLine (canvas, x - xx, y - yy, x + xx, y - yy, color);
          g15r_drawLine (canvas, x - xx, y + yy, x + xx, y + yy, color);
        }
      if (dd + yy > 0)
        {
          yy--;
          dd = dd - (2 * yy + 1);
        }
      if (xx > dd)
        {
          xx++;
          dd = dd + (2 * xx + 1);
        }
    }
}
```

4.3.1.4 void g15r_drawIcon (**g15canvas** ∗ *canvas,* char ∗ *buf,* int *my_x,* int *my_y,* int *width,* int *height*)

Draw an icon to the screen from a wbmp buffer.

Draw an icon to a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated in is found.
buf	A pointer to the buffer holding the icon to be displayed.
my_x	Leftmost boundary of image.
my_y	Topmost boundary of image.
width	Width of the image in buf.
height	Height of the image in buf.

Definition at line 411 of file pixel.c.

References BYTE_SIZE, and g15r_setPixel().

```
{
    int y,x,val;
    unsigned int pixel_offset = 0;
    unsigned int byte_offset, bit_offset;

    for (y=0; y < height - 1; y++)
      for (x=0; x < width - 1; x++)
        {
                pixel_offset = y * width + x;
                byte_offset = pixel_offset / BYTE_SIZE;
                bit_offset = 7 - (pixel_offset % BYTE_SIZE);

                val = (buf[byte_offset] & (1 << bit_offset)) >> bit_offset;
                g15r_setPixel (canvas, x + my_x, y + my_y, val);
        }
}
```

4.3.1.5 void g15r_drawLine (**g15canvas** * *canvas*, int *px1*, int *py1*, int *px2*, int *py2*, const int *color*)

Draws a line from (px1, py1) to (px2, py2)

A line of color is drawn from (px1, py1) to (px2, py2).

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
px1	X component of point 1.
py1	Y component of point 1.
px2	X component of point 2.
py2	Y component of point 2.
color	Line will be drawn this color.

Definition at line 99 of file pixel.c.

References g15r_setPixel(), and swap().

Referenced by g15r_drawBar(), g15r_drawCircle(), g15r_drawRoundBox(), and g15r_pixelBox().

```
{
  /*
   * Bresenham's Line Algorithm
   * http://en.wikipedia.org/wiki/Bresenham's_algorithm
   */

  int steep = 0;

  if (abs (py2 - py1) > abs (px2 - px1))
    steep = 1;

  if (steep)
    {
      swap (&px1, &py1);
      swap (&px2, &py2);
    }

  if (px1 > px2)
    {
      swap (&px1, &px2);
      swap (&py1, &py2);
    }

  int dx = px2 - px1;
  int dy = abs (py2 - py1);

  int error = 0;
  int y = py1;
  int ystep = (py1 < py2) ? 1 : -1;
  int x = 0;

  for (x = px1; x <= px2; ++x)
    {
      if (steep)
        g15r_setPixel (canvas, y, x, color);
      else
        g15r_setPixel (canvas, x, y, color);

      error += dy;
      if (2 * error >= dx)
        {
          y += ystep;
          error -= dx;
        }
    }
}
```

4.3.1.6 void g15r_drawRoundBox (g15canvas * *canvas*, int *x1*, int *y1*, int *x2*, int *y2*, int *fill*, int *color*)

Draws a box with rounded corners bounded by (x1, y1) and (x2, y2)

Draws a rounded box around the area bounded by (x1, y1) and (x2, y2).

The box will be filled if fill != 0.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the box.
y1	Defines uppermost bound of the box.
x2	Defines rightmost bound of the box.
y2	Defines bottommost bound of the box.
fill	The box will be filled with color if fill != 0.
color	Lines defining the box will be drawn this color.

Definition at line 252 of file pixel.c.

References G15_COLOR_BLACK, G15_COLOR_WHITE, g15r_drawLine(), and g15r_setPixel().

```
{
  int y, shave = 3;

  if (shave > (x2 - x1) / 2)
    shave = (x2 - x1) / 2;
  if (shave > (y2 - y1) / 2)
    shave = (y2 - y1) / 2;

  if ((x1 != x2) && (y1 != y2))
    {
      if (fill)
        {
          g15r_drawLine (canvas, x1 + shave, y1, x2 - shave, y1, color);
          for (y = y1 + 1; y < y1 + shave; y++)
            g15r_drawLine (canvas, x1 + 1, y, x2 - 1, y, color);
          for (y = y1 + shave; y <= y2 - shave; y++)
            g15r_drawLine (canvas, x1, y, x2, y, color);
          for (y = y2 - shave + 1; y < y2; y++)
            g15r_drawLine (canvas, x1 + 1, y, x2 - 1, y, color);
          g15r_drawLine (canvas, x1 + shave, y2, x2 - shave, y2, color);
          if (shave == 4)
            {
              g15r_setPixel (canvas, x1 + 1, y1 + 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
              g15r_setPixel (canvas, x1 + 1, y2 - 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
              g15r_setPixel (canvas, x2 - 1, y1 + 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
              g15r_setPixel (canvas, x2 - 1, y2 - 1,
                             color ==
                             G15_COLOR_WHITE ? G15_COLOR_BLACK :
                             G15_COLOR_WHITE);
            }
        }
      else
        {
          g15r_drawLine (canvas, x1 + shave, y1, x2 - shave, y1, color);
          g15r_drawLine (canvas, x1, y1 + shave, x1, y2 - shave, color);
          g15r_drawLine (canvas, x2, y1 + shave, x2, y2 - shave, color);
          g15r_drawLine (canvas, x1 + shave, y2, x2 - shave, y2, color);
          if (shave > 1)
            {
              g15r_drawLine (canvas, x1 + 1, y1 + 1, x1 + shave - 1, y1 + 1,
                             color);
              g15r_drawLine (canvas, x2 - shave + 1, y1 + 1, x2 - 1, y1 + 1,
                             color);
              g15r_drawLine (canvas, x1 + 1, y2 - 1, x1 + shave - 1, y2 - 1,
                             color);
              g15r_drawLine (canvas, x2 - shave + 1, y2 - 1, x2 - 1, y2 - 1,
                             color);
              g15r_drawLine (canvas, x1 + 1, y1 + 1, x1 + 1, y1 + shave - 1,
                             color);
              g15r_drawLine (canvas, x1 + 1, y2 - 1, x1 + 1, y2 - shave + 1,
                             color);
              g15r_drawLine (canvas, x2 - 1, y1 + 1, x2 - 1, y1 + shave - 1,
                             color);
              g15r_drawLine (canvas, x2 - 1, y2 - 1, x2 - 1, y2 - shave + 1,
                             color);
            }
        }
    }
}
```

4.3.1.7 void g15r_drawSprite (**g15canvas** ∗ *canvas,* char ∗ *buf,* int *my_x,* int *my_y,* int *width,* int *height,* int *start_x,* int *start_y,* int *total_width*)

Draw a sprite to the screen from a wbmp buffer.

Draw a sprite to a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated in is found.
buf	A pointer to the buffer holding a set of sprites.

Parameters

my_x	Leftmost boundary of image.
my_y	Topmost boundary of image.
width	Width of the sprite.
height	Height of the sprite.
start_x	X offset for reading sprite from buf.
start_y	Y offset for reading sprite from buf.
total_width	Width of the set of sprites held in buf.

Definition at line 443 of file pixel.c.

References BYTE_SIZE, and g15r_setPixel().

```
{
    int y,x,val;
    unsigned int pixel_offset = 0;
    unsigned int byte_offset, bit_offset;

    for (y=0; y < height - 1; y++)
      for (x=0; x < width - 1; x++)
        {
                pixel_offset = (y + start_y) * total_width + (x + start_x);
                byte_offset = pixel_offset / BYTE_SIZE;
                bit_offset = 7 - (pixel_offset % BYTE_SIZE);

                val = (buf[byte_offset] & (1 << bit_offset)) >> bit_offset;
                g15r_setPixel (canvas, x + my_x, y + my_y, val);
        }
}
```

4.3.1.8 int g15r_loadWbmpSplash (**g15canvas** ∗ *canvas,* char ∗ *filename*)

Draw a splash screen from 160x43 wbmp file.

wbmp splash screen loader - assumes image is 160x43

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
filename	A string holding the path to the wbmp to be displayed.

Definition at line 387 of file pixel.c.

References g15canvas::buffer, G15_BUFFER_LEN, and g15r_loadWbmpToBuf().

```
{
    int width=0, height=0;
    char *buf;

    buf = g15r_loadWbmpToBuf(filename,
                             &width,
                             &height);

    memcpy (canvas->buffer, buf, G15_BUFFER_LEN);
    return 0;
}
```

4.3.1.9 char∗ g15r_loadWbmpToBuf (char ∗ *filename*, int ∗ *img_width*, int ∗ *img_height*)

Load a wbmp file into a buffer.

basic wbmp loader - loads a wbmp image into a buffer.

Parameters

filename	A string holding the path to the wbmp to be loaded.
img_width	A pointer to an int that will hold the image width on return.
img_height	A pointer to an int that will hold the image height on return.

Definition at line 469 of file pixel.c.

References BYTE_SIZE.

Referenced by g15r_loadWbmpSplash().

```
{
    int wbmp_fd;
    int retval;
    int x,y,val;
    char *buf;
    unsigned int buflen,header=4;
    unsigned char headerbytes[5];
    unsigned int pixel_offset = 0;
    unsigned int byte_offset, bit_offset;

    wbmp_fd=open(filename,O_RDONLY);
    if(!wbmp_fd){
        return NULL;
    }

    retval=read(wbmp_fd,headerbytes,5);

    if(retval){
        if (headerbytes[2] & 1) {
            *img_width = ((unsigned char)headerbytes[2] ^ 1) | (unsigned char)headerbytes[3];
            *img_height = headerbytes[4];
            header = 5;
        } else {
            *img_width = headerbytes[2];
            *img_height = headerbytes[3];
        }

        int byte_width = *img_width / 8;
        if (*img_width %8)
          byte_width++;

        buflen = byte_width * (*img_height);

        buf = (char *)malloc (buflen);
        if (buf == NULL)
          return NULL;

        if (header == 4)
          buf[0]=headerbytes[4];

        retval=read(wbmp_fd,buf+(5-header),buflen);

        close(wbmp_fd);
    }

    /* now invert the image */
    for (y = 0; y < *img_height; y++)
      for (x = 0; x < *img_width; x++)
        {
                pixel_offset = y * (*img_width) + x;
                byte_offset = pixel_offset / BYTE_SIZE;
                bit_offset = 7 - (pixel_offset % BYTE_SIZE);

                val = (buf[byte_offset] & (1 << bit_offset)) >> bit_offset;

                if (!val)
```

```
                  buf[byte_offset] = buf[byte_offset] | 1 << bit_offset;
                else
                  buf[byte_offset] = buf[byte_offset] & ~(1 << bit_offset);
        }

    return buf;
}
```

4.3.1.10 void g15r_pixelBox (**g15canvas** ∗ *canvas,* int *x1,* int *y1,* int *x2,* int *y2,* int *color,* int *thick,* int *fill*)

Draws a box bounded by (x1, y1) and (x2, y2)

Draws a box around the area bounded by (x1, y1) and (x2, y2).

The box will be filled if fill != 0 and the sides will be thick pixels wide.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of the box.
y1	Defines uppermost bound of the box.
x2	Defines rightmost bound of the box.
y2	Defines bottommost bound of the box.
color	Lines defining the box will be drawn this color.
thick	Lines defining the box will be this many pixels thick.
fill	The box will be filled with color if fill != 0.

Definition at line 163 of file pixel.c.

References g15r_drawLine(), and g15r_setPixel().

Referenced by g15r_drawBar(), and g15r_drawBigNum().

```
{
  int i = 0;
  for (i = 0; i < thick; ++i)
    {
      g15r_drawLine (canvas, x1, y1, x2, y1, color);    /* Top    */
      g15r_drawLine (canvas, x1, y1, x1, y2, color);    /* Left   */
      g15r_drawLine (canvas, x2, y1, x2, y2, color);    /* Right  */
      g15r_drawLine (canvas, x1, y2, x2, y2, color);    /* Bottom */
      x1++;
      y1++;
      x2--;
      y2--;
    }

  int x = 0, y = 0;

  if (fill)
    {
      for (x = x1; x <= x2; ++x)
        for (y = y1; y <= y2; ++y)
          g15r_setPixel (canvas, x, y, color);
    }

}
```

4.3.1.11 void g15r_pixelOverlay (**g15canvas** ∗ *canvas,* int *x1,* int *y1,* int *width,* int *height,* short *colormap[ ]*)

Overlays a bitmap of size width x height starting at (x1, y1)

A 1-bit bitmap defined in colormap[] is drawn to the canvas with an upper left corner at (x1, y1) and a lower right corner at (x1+width, y1+height).

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines the leftmost bound of the area to be drawn.
y1	Defines the uppermost bound of the area to be drawn.
width	Defines the width of the bitmap to be drawn.
height	Defines the height of the bitmap to be drawn.
colormap	An array containing width∗height entries of value 0 for pixel off or != 0 for pixel on.

Definition at line 74 of file pixel.c.

References G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

```
{
  int i = 0;

  for (i = 0; i < (width * height); ++i)
    {
      int color = (colormap[i] ? G15_COLOR_BLACK : G15_COLOR_WHITE);
      int x = x1 + i % width;
      int y = y1 + i / width;
      g15r_setPixel (canvas, x, y, color);
    }
}
```

4.3.1.12 void g15r_pixelReverseFill (**g15canvas** ∗ *canvas,* int *x1,* int *y1,* int *x2,* int *y2,* int *fill,* int *color*)

Fills an area bounded by (x1, y1) and (x2, y2)

The area with an upper left corner at (x1, y1) and lower right corner at (x2, y2) will be filled with color if fill>0 or the current contents of the area will be reversed if fill==0.

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x1	Defines leftmost bound of area to be filled.
y1	Defines uppermost bound of area to be filled.
x2	Defines rightmost bound of area to be filled.
y2	Defines bottommost bound of area to be filled.
fill	Area will be filled with color if fill != 0, else contents of area will have color values reversed.
color	If fill != 0, then area will be filled if color == 1 and emptied if color == 0.

Definition at line 45 of file pixel.c.

References g15r_getPixel(), and g15r_setPixel().

```
{
  int x = 0;
  int y = 0;

  for (x = x1; x <= x2; ++x)
    {
      for (y = y1; y <= y2; ++y)
        {
          if (!fill)
            color = !g15r_getPixel (canvas, x, y);
          g15r_setPixel (canvas, x, y, color);
        }
    }
}
```

4.3.1.13 void swap (int ∗ *x,* int ∗ *y*)

Definition at line 23 of file pixel.c.

Referenced by g15r_drawLine().

```
{
  int tmp;

  tmp = *x;
  *x = *y;
  *y = tmp;
}
```

4.4 screen.c File Reference

```
#include "libg15render.h"
```

Functions

- void **g15r_clearScreen** (**g15canvas** ∗canvas, int color)

  Fills the screen with pixels of color.
- int **g15r_getPixel** (**g15canvas** ∗canvas, unsigned int x, unsigned int y)

  Gets the value of the pixel at (x, y)
- void **g15r_initCanvas** (**g15canvas** ∗canvas)

  Clears the canvas and resets the mode switches.
- void **g15r_setPixel** (**g15canvas** ∗canvas, unsigned int x, unsigned int y, int val)

  Sets the value of the pixel at (x, y)

4.4.1 Function Documentation

4.4.1.1 void g15r_clearScreen (g15canvas ∗ *canvas,* int *color*)

Fills the screen with pixels of color.

Clears the screen and fills it with pixels of color

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
color	Screen will be filled with this color.

Definition at line 80 of file screen.c.

References g15canvas::buffer, and G15_BUFFER_LEN.

```
{
  memset (canvas->buffer, (color ? 0xFF : 0), G15_BUFFER_LEN);
}
```

4.4.1.2 int g15r_getPixel (**g15canvas** ∗ *canvas,* unsigned int *x,* unsigned int *y*)

Gets the value of the pixel at (x, y)

Retrieves the value of the pixel at (x, y)

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	X offset for pixel to be retrieved.
y	Y offset for pixel to be retrieved.

Definition at line 29 of file screen.c.

References g15canvas::buffer, BYTE_SIZE, G15_LCD_HEIGHT, and G15_LCD_WIDTH.

Referenced by g15r_pixelReverseFill(), and g15r_setPixel().

```
{
  if (x >= G15_LCD_WIDTH || y >= G15_LCD_HEIGHT)
    return 0;

  unsigned int pixel_offset = y * G15_LCD_WIDTH + x;
  unsigned int byte_offset = pixel_offset / BYTE_SIZE;
  unsigned int bit_offset = 7 - (pixel_offset % BYTE_SIZE);

  return (canvas->buffer[byte_offset] & (1 << bit_offset)) >> bit_offset;
}
```

4.4.1.3 void g15r_initCanvas (**g15canvas** ∗ *canvas*)

Clears the canvas and resets the mode switches.

Clears the screen and resets the mode values for a canvas

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct

Definition at line 91 of file screen.c.

References g15canvas::buffer, g15canvas::ftLib, G15_BUFFER_LEN, g15canvas::mode_cache, g15canvas↩ ::mode_reverse, and g15canvas::mode_xor.

```
{
  memset (canvas->buffer, 0, G15_BUFFER_LEN);
  canvas->mode_cache = 0;
  canvas->mode_reverse = 0;
  canvas->mode_xor = 0;
#ifdef TTF_SUPPORT
  if (FT_Init_FreeType (&canvas->ftLib))
    printf ("Freetype couldnt initialise\n");
#endif
}
```

4.4.1.4 void g15r_setPixel (**g15canvas** ∗ *canvas,* unsigned int *x,* unsigned int *y,* int *val*)

Sets the value of the pixel at (x, y)

Sets the value of the pixel at (x, y)

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	X offset for pixel to be set.
y	Y offset for pixel to be set.
val	Value to which pixel should be set.

Definition at line 50 of file screen.c.

References g15canvas::buffer, BYTE_SIZE, G15_LCD_HEIGHT, G15_LCD_WIDTH, g15r_getPixel(), g15canvas↩ ::mode_reverse, and g15canvas::mode_xor.

Referenced by draw_ttf_char(), g15r_drawCircle(), g15r_drawIcon(), g15r_drawLine(), g15r_drawRoundBox(), g15r_drawSprite(), g15r_pixelBox(), g15r_pixelOverlay(), g15r_pixelReverseFill(), g15r_renderCharacterLarge(), g15r_renderCharacterMedium(), and g15r_renderCharacterSmall().

```
{
  if (x >= G15_LCD_WIDTH || y >= G15_LCD_HEIGHT)
    return;

  unsigned int pixel_offset = y * G15_LCD_WIDTH + x;
  unsigned int byte_offset = pixel_offset / BYTE_SIZE;
  unsigned int bit_offset = 7 - (pixel_offset % BYTE_SIZE);

  if (canvas->mode_xor)
    val ^= g15r_getPixel (canvas, x, y);
  if (canvas->mode_reverse)
    val = !val;

  if (val)
    canvas->buffer[byte_offset] =
      canvas->buffer[byte_offset] | 1 << bit_offset;
  else
    canvas->buffer[byte_offset] =
      canvas->buffer[byte_offset] & ~(1 << bit_offset);

}
```

4.5 text.c File Reference

```
#include "libg15render.h"
```

Functions

- int **calc_ttf_centering** (FT_Face face, char ∗str)
- int **calc_ttf_right_justify** (FT_Face face, char ∗str)
- int **calc_ttf_totalstringwidth** (FT_Face face, char ∗str)
- int **calc_ttf_true_ypos** (FT_Face face, int y, int ttf_fontsize)
- void **draw_ttf_char** (**g15canvas** ∗canvas, FT_Bitmap charbitmap, unsigned char character, int x, int y, int color)
- void **draw_ttf_str** (**g15canvas** ∗canvas, char ∗str, int x, int y, int color, FT_Face face)
- void **g15r_renderCharacterLarge** (**g15canvas** ∗canvas, int col, int row, unsigned char character, unsigned int sx, unsigned int sy)

  Renders a character in the large font at (x, y)
- void **g15r_renderCharacterMedium** (**g15canvas** ∗canvas, int col, int row, unsigned char character, unsigned int sx, unsigned int sy)

  Renders a character in the meduim font at (x, y)
- void **g15r_renderCharacterSmall** (**g15canvas** ∗canvas, int col, int row, unsigned char character, unsigned int sx, unsigned int sy)

  Renders a character in the small font at (x, y)
- void **g15r_renderString** (**g15canvas** ∗canvas, unsigned char stringOut[ ], int row, int size, unsigned int sx, unsigned int sy)

  Renders a string with font size in row.
- void **g15r_ttfLoad** (**g15canvas** ∗canvas, char ∗fontname, int fontsize, int face_num)

  Loads a font through the FreeType2 library.
- void **g15r_ttfPrint** (**g15canvas** ∗canvas, int x, int y, int fontsize, int face_num, int color, int center, char ∗print_string)

  Prints a string in a given font.

4.5.1 Function Documentation

4.5.1.1 int calc_ttf_centering (FT_Face *face,* char ∗ *str*)

Definition at line 209 of file text.c.

References calc_ttf_totalstringwidth().

Referenced by g15r_ttfPrint().

```
{
  int leftpos;

  leftpos = 80 - (calc_ttf_totalstringwidth (face, str) / 2);
  if (leftpos < 1)
    leftpos = 1;

  return leftpos;
}
```

4.5.1.2 int calc_ttf_right_justify (FT_Face *face,* char ∗ *str*)

Definition at line 221 of file text.c.

References calc_ttf_totalstringwidth().

Referenced by g15r_ttfPrint().

```
{
  int leftpos;

  leftpos = 160 - calc_ttf_totalstringwidth (face, str);
  if (leftpos < 1)
    leftpos = 1;

  return leftpos;
}
```

4.5.1.3 int calc_ttf_totalstringwidth (FT_Face *face,* char ∗ *str*)

Definition at line 191 of file text.c.

Referenced by calc_ttf_centering(), and calc_ttf_right_justify().

```
{
  FT_GlyphSlot slot = face->glyph;
  FT_UInt glyph_index;
  int i, errcode;
  unsigned int len = strlen (str);
  int width = 0;

  for (i = 0; i < len; i++)
    {
      glyph_index = FT_Get_Char_Index (face, str[i]);
      errcode = FT_Load_Glyph (face, glyph_index, 0);
      width += slot->advance.x >> 6;
    }
  return width;
}
```

4.5.1.4 int calc_ttf_true_ypos (FT_Face *face,* int *y,* int *ttf_fontsize*)

Definition at line 179 of file text.c.

Referenced by g15r_ttfPrint().

```
{

  if (!FT_IS_SCALABLE (face))
    ttf_fontsize = face->available_sizes->height;

  y += ttf_fontsize * .75;

  return y;
}
```

4.5.1.5 void draw_ttf_char (g15canvas ∗ *canvas,* FT_Bitmap *charbitmap,* unsigned char *character,* int *x,* int *y,* int *color*)

Definition at line 233 of file text.c.

References g15canvas::ftLib, and g15r_setPixel().

Referenced by draw_ttf_str().

```
{
  FT_Int char_x, char_y, p, q;
  FT_Int x_max = x + charbitmap.width;
  FT_Int y_max = y + charbitmap.rows;
  static FT_Bitmap tmpbuffer;

  /* convert to 8bit format.. */
  FT_Bitmap_Convert (canvas->ftLib, &charbitmap, &tmpbuffer, 1);

  for (char_y = y, q = 0; char_y < y_max; char_y++, q++)
      for (char_x = x, p = 0; char_x < x_max; char_x++, p++)
          if (tmpbuffer.buffer[q * tmpbuffer.width + p])
            g15r_setPixel (canvas, char_x, char_y, color);
}
```

4.5.1.6 void draw_ttf_str (g15canvas ∗ *canvas,* char ∗ *str,* int *x,* int *y,* int *color,* FT_Face *face*)

Definition at line 251 of file text.c.

References draw_ttf_char().

Referenced by g15r_ttfPrint().

```
{
  FT_GlyphSlot slot = face->glyph;
  int i, errcode;
  unsigned int len = strlen (str);

  for (i = 0; i < len; i++)
    {
      errcode =
        FT_Load_Char (face, str[i],
                      FT_LOAD_RENDER | FT_LOAD_MONOCHROME |
                      FT_LOAD_TARGET_MONO);
      draw_ttf_char (canvas, slot->bitmap, str[i], x + slot->bitmap_left,
                     y - slot->bitmap_top, color);
      x += slot->advance.x >> 6;
    }
}
```

4.5.1.7 void g15r_renderCharacterLarge (g15canvas ∗ *canvas,* int *col,* int *row,* unsigned char *character,* unsigned int *sx,* unsigned int *sy*)

Renders a character in the large font at (x, y)

Definition at line 22 of file text.c.

References fontdata_8x8, G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

Referenced by g15r_renderString().

```
{
  int helper = character * 8;   /* for our font which is 8x8 */

  int top_left_pixel_x = sx + col * (8);        /* 1 pixel spacing */
  int top_left_pixel_y = sy + row * (8);        /* once again 1 pixel spacing */

  int x, y;
  for (y = 0; y < 8; ++y)
    {
      for (x = 0; x < 8; ++x)
        {
          char font_entry = fontdata_8x8[helper + y];

          if (font_entry & 1 << (7 - x))
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_BLACK);
          else
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_WHITE);

        }
    }
}
```

4.5.1.8 void g15r_renderCharacterMedium (g15canvas * *canvas*, int *col*, int *row*, unsigned char *character*, unsigned int *sx*, unsigned int *sy*)

Renders a character in the meduim font at (x, y)

Definition at line 50 of file text.c.

References fontdata_7x5, G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

Referenced by g15r_renderString().

```
{
  int helper = character * 7 * 5;       /* for our font which is 6x4 */

  int top_left_pixel_x = sx + col * (5);        /* 1 pixel spacing */
  int top_left_pixel_y = sy + row * (7);        /* once again 1 pixel spacing */

  int x, y;
  for (y = 0; y < 7; ++y)
    {
      for (x = 0; x < 5; ++x)
        {
          char font_entry = fontdata_7x5[helper + y * 5 + x];
          if (font_entry)
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_BLACK);
          else
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_WHITE);

        }
    }
}
```

4.5.1.9 void g15r_renderCharacterSmall (g15canvas * *canvas*, int *col*, int *row*, unsigned char *character*, unsigned int *sx*, unsigned int *sy*)

Renders a character in the small font at (x, y)

Definition at line 77 of file text.c.

References fontdata_6x4, G15_COLOR_BLACK, G15_COLOR_WHITE, and g15r_setPixel().

Referenced by g15r_renderString().

```
{
  int helper = character * 6 * 4;       /* for our font which is 6x4 */

  int top_left_pixel_x = sx + col * (4);        /* 1 pixel spacing */
  int top_left_pixel_y = sy + row * (6);        /* once again 1 pixel spacing */

  int x, y;
  for (y = 0; y < 6; ++y)
    {
      for (x = 0; x < 4; ++x)
        {
          char font_entry = fontdata_6x4[helper + y * 4 + x];
          if (font_entry)
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_BLACK);
          else
            g15r_setPixel (canvas, top_left_pixel_x + x, top_left_pixel_y + y,
                           G15_COLOR_WHITE);

        }
    }
}
```

4.5.1.10 void g15r_renderString (**g15canvas** ∗ *canvas,* unsigned char *stringOut[ ],* int *row,* int *size,* unsigned int *sx,* unsigned int *sy*)

Renders a string with font size in row.

Definition at line 104 of file text.c.

References G15_TEXT_LARGE, G15_TEXT_MED, G15_TEXT_SMALL, g15r_renderCharacterLarge(), g15r_↩
renderCharacterMedium(), and g15r_renderCharacterSmall().

```
{

  int i = 0;
  for (i; stringOut[i] != NULL; ++i)
    {
      switch (size)
        {
        case G15_TEXT_SMALL:
          {
            g15r_renderCharacterSmall (canvas, i, row, stringOut[i], sx, sy);
            break;
          }
        case G15_TEXT_MED:
          {
            g15r_renderCharacterMedium (canvas, i, row, stringOut[i], sx, sy);
            break;
          }
        case G15_TEXT_LARGE:
          {
            g15r_renderCharacterLarge (canvas, i, row, stringOut[i], sx, sy);
            break;
          }
        default:
          break;
        }
    }

}
```

4.5.1.11 void g15r_ttfLoad (**g15canvas** ∗ *canvas,* char ∗ *fontname,* int *fontsize,* int *face_num*)

Loads a font through the FreeType2 library.

Load a font for use with FreeType2 font support

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
fontname	Absolute pathname to font file to be loaded.
fontsize	Size in points for font to be loaded.
face_num	Slot into which font face will be loaded.

Definition at line 145 of file text.c.

References g15canvas::ftLib, G15_MAX_FACE, g15canvas::ttf_face, and g15canvas::ttf_fontsize.

```
{
  int errcode = 0;

  if (face_num < 0)
    face_num = 0;
  if (face_num > G15_MAX_FACE)
    face_num = G15_MAX_FACE;

  if (canvas->ttf_fontsize[face_num])
    FT_Done_Face (canvas->ttf_face[face_num][0]);       /* destroy the last face */

  if (!canvas->ttf_fontsize[face_num] && !fontsize)
    canvas->ttf_fontsize[face_num] = 10;
  else
    canvas->ttf_fontsize[face_num] = fontsize;

  errcode =
    FT_New_Face (canvas->ftLib, fontname, 0, &canvas->ttf_face[face_num][0]);
  if (errcode)
    {
      canvas->ttf_fontsize[face_num] = 0;
    }
  else
    {
      if (canvas->ttf_fontsize[face_num]
          && FT_IS_SCALABLE (canvas->ttf_face[face_num][0]))
        errcode =
          FT_Set_Char_Size (canvas->ttf_face[face_num][0], 0,
                            canvas->ttf_fontsize[face_num] * 64, 90, 0);
    }
}
```

4.5.1.12 void g15r_ttfPrint (**g15canvas** ∗ *canvas,* int *x,* int *y,* int *fontsize,* int *face_num,* int *color,* int *center,* char ∗ *print_string*)

Prints a string in a given font.

Render a string with a FreeType2 font

Parameters

canvas	A pointer to a **g15canvas** (p. 5) struct in which the buffer to be operated on is found.
x	initial x position for string.
y	initial y position for string.
fontsize	Size of string in points.
face_num	Font to be used is loaded in this slot.
color	Text will be drawn this color.
center	Text will be centered if center == 1 and right justified if center == 2.
print_string	Pointer to the string to be printed.

Definition at line 283 of file text.c.

References calc_ttf_centering(), calc_ttf_right_justify(), calc_ttf_true_ypos(), draw_ttf_str(), g15canvas::ttf_face, and g15canvas::ttf_fontsize.

```
{
  int errcode = 0;

  if (canvas->ttf_fontsize[face_num])
    {
      if (fontsize > 0 && FT_IS_SCALABLE (canvas->ttf_face[face_num][0]))
        {
          canvas->ttf_fontsize[face_num] = fontsize;
          int errcode =
            FT_Set_Pixel_Sizes (canvas->ttf_face[face_num][0], 0,
                                canvas->ttf_fontsize[face_num]);
          if (errcode)
            printf ("Trouble setting the Glyph size!\n");
        }
      y =
        calc_ttf_true_ypos (canvas->ttf_face[face_num][0], y,
                            canvas->ttf_fontsize[face_num]);
      if (center == 1)
```

```
        x = calc_ttf_centering (canvas->ttf_face[face_num][0], print_string);
      else if (center == 2)
        x = calc_ttf_right_justify (canvas->ttf_face[face_num][0], print_string);
      draw_ttf_str (canvas, print_string, x, y, color,
                    canvas->ttf_face[face_num][0]);
    }
}
```

Index