gfx mono

Generated by Doxygen 1.8.11

Contents

1 License **1**

2 Examples for GFX Mono Library **3**

2.1 Quick Start Guide for the mono graphics service . . . 3

2.1.1 Basic usage of the graphics service . . . 3

2.1.2 Usage steps . . . 3

2.1.2.1 Example code . . . 3

2.1.2.2 Workflow . . . 4

2.2 Quick Start Guide for the mono font service . . . 4

2.2.1 Basic usage of the graphics service . . . 4

2.2.2 Dependencies . . . 4

2.2.3 Usage steps . . . 4

2.2.3.1 Example code . . . 4

2.2.3.2 Workflow . . . 5

3 Module Index **7**

3.1 Modules . . . 7

4 Class Index **9**

4.1 Class List . . . 9

5 File Index **11**

5.1 File List . . . 11

6 Module Documentation 13

6.1 Monochrome graphical display system . . . 13

6.1.1 Detailed Description . . . 15

6.1.2 Examples . . . 15

6.1.3 API Overview . . . 15

6.1.4 Macro Definition Documentation . . . 15

6.1.4.1 GFX_BOTTOMHALF . . . 15

6.1.4.2 GFX_LEFTHALF . . . 16

6.1.4.3 gfx_mono_draw_circle . . . 16

6.1.4.4 gfx_mono_draw_filled_circle . . . 16

6.1.4.5 gfx_mono_draw_filled_rect . . . 17

6.1.4.6 gfx_mono_draw_horizontal_line . . . 17

6.1.4.7 gfx_mono_draw_line . . . 18

6.1.4.8 gfx_mono_draw_rect . . . 18

6.1.4.9 gfx_mono_draw_vertical_line . . . 18

6.1.4.10 GFX_OCTANT0 . . . 18

6.1.4.11 GFX_OCTANT1 . . . 19

6.1.4.12 GFX_OCTANT2 . . . 19

6.1.4.13 GFX_OCTANT3 . . . 19

6.1.4.14 GFX_OCTANT4 . . . 19

6.1.4.15 GFX_OCTANT5 . . . 19

6.1.4.16 GFX_OCTANT6 . . . 19

6.1.4.17 GFX_OCTANT7 . . . 20

6.1.4.18 GFX_QUADRANT0 . . . 20

6.1.4.19 GFX_QUADRANT1 . . . 20

6.1.4.20 GFX_QUADRANT2 . . . 20

6.1.4.21 GFX_QUADRANT3 . . . 20

6.1.4.22 GFX_RIGHTHALF . . . 20

6.1.4.23 GFX_TOPHALF . . . 21

6.1.4.24 GFX_WHOLE . . . 21

6.1.5 Typedef Documentation . . . 21
6.1.5.1 gfx_coord_t . . . 21
6.1.5.2 gfx_mono_color_t . . . 21
6.1.6 Enumeration Type Documentation . . . 21
6.1.6.1 gfx_mono_bitmap_type . . . 21
6.1.6.2 gfx_mono_color . . . 21
6.2 Framebuffer . . . 22
6.2.1 Detailed Description . . . 22
6.2.2 Function Documentation . . . 22
6.2.2.1 gfx_mono_framebuffer_draw_pixel(gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color) . . . 22
6.2.2.2 gfx_mono_framebuffer_get_byte(gfx_coord_t page, gfx_coord_t column) . . . 24
6.2.2.3 gfx_mono_framebuffer_get_page(gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width) . . . 24
6.2.2.4 gfx_mono_framebuffer_get_pixel(gfx_coord_t x, gfx_coord_t y) . . . 25
6.2.2.5 gfx_mono_framebuffer_mask_byte(gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_mask, gfx_mono_color_t color) . . . 26
6.2.2.6 gfx_mono_framebuffer_put_byte(gfx_coord_t page, gfx_coord_t column, uint8_t data) . . . 27
6.2.2.7 gfx_mono_framebuffer_put_page(gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width) . . . 28
6.2.2.8 gfx_mono_set_framebuffer(uint8_t *framebuffer) . . . 28
6.3 Generic monochrome graphic primitives . . . 30
6.3.1 Detailed Description . . . 31
6.3.2 Function Documentation . . . 31
6.3.2.1 gfx_mono_generic_draw_circle(gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t octant_mask) . . . 31
6.3.2.2 gfx_mono_generic_draw_filled_circle(gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t quadrant_mask) . . . 32
6.3.2.3 gfx_mono_generic_draw_filled_rect(gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color) . . . 34
6.3.2.4 gfx_mono_generic_draw_horizontal_line(gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_mono_color color) . . . 35
6.3.2.5 gfx_mono_generic_draw_line(gfx_coord_t x1, gfx_coord_t y1, gfx_coord_t x2, gfx_coord_t y2, enum gfx_mono_color color) . . . 36

6.3.2.6 gfx_mono_generic_draw_rect(gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color) . . . 37

6.3.2.7 gfx_mono_generic_draw_vertical_line(gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_mono_color color) . . . 38

6.3.2.8 gfx_mono_generic_put_bitmap(struct gfx_mono_bitmap *bitmap, gfx_coord_t x, gfx_coord_t y) . . . 39

6.4 GFX Mono Font Library . . . 40

6.4.1 Detailed Description . . . 40

6.4.2 API Overview . . . 41

6.4.3 Enumeration Type Documentation . . . 41

6.4.3.1 font_data_type . . . 41

6.4.4 Function Documentation . . . 41

6.4.4.1 gfx_mono_draw_char(const char c, const gfx_coord_t x, const gfx_coord_t y, const struct font *font) . . . 41

6.4.4.2 gfx_mono_draw_progmem_string(char PROGMEM_PTR_T str, gfx_coord_t x, gfx_coord_t y, const struct font *font) . . . 42

6.4.4.3 gfx_mono_draw_string(const char *str, const gfx_coord_t x, const gfx_coord_t y, const struct font *font) . . . 43

6.4.4.4 gfx_mono_get_progmem_string_bounding_box(char PROGMEM_PTR_T str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height) . . . 44

6.4.4.5 gfx_mono_get_string_bounding_box(char const *str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height) . . . 45

6.5 2832HSWEG04 graphic library abstraction . . . 47

6.5.1 Detailed Description . . . 48

6.5.2 Macro Definition Documentation . . . 48

6.5.2.1 gfx_mono_draw_pixel . . . 48

6.5.2.2 gfx_mono_get_byte . . . 48

6.5.2.3 gfx_mono_get_page . . . 48

6.5.2.4 gfx_mono_get_pixel . . . 48

6.5.2.5 gfx_mono_init . . . 49

6.5.2.6 GFX_MONO_LCD_FRAMEBUFFER_SIZE . . . 49

6.5.2.7 GFX_MONO_LCD_HEIGHT . . . 49

6.5.2.8 GFX_MONO_LCD_PAGES . . . 49

6.5.2.9 GFX_MONO_LCD_PIXELS_PER_BYTE . . . 49

6.5.2.10 GFX_MONO_LCD_WIDTH . . . 49
6.5.2.11 gfx_mono_mask_byte . . . 50
6.5.2.12 gfx_mono_put_bitmap . . . 50
6.5.2.13 gfx_mono_put_byte . . . 50
6.5.2.14 gfx_mono_put_framebuffer . . . 50
6.5.2.15 gfx_mono_put_page . . . 50
6.5.3 Function Documentation . . . 50
6.5.3.1 gfx_mono_ssd1306_draw_pixel(gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color) . . . 50
6.5.3.2 gfx_mono_ssd1306_get_byte(gfx_coord_t page, gfx_coord_t column) . . . 51
6.5.3.3 gfx_mono_ssd1306_get_page(gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width) . . . 52
6.5.3.4 gfx_mono_ssd1306_get_pixel(gfx_coord_t x, gfx_coord_t y) . . . 53
6.5.3.5 gfx_mono_ssd1306_init(void) . . . 54
6.5.3.6 gfx_mono_ssd1306_mask_byte(gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_mask, gfx_mono_color_t color) . . . 55
6.5.3.7 gfx_mono_ssd1306_put_byte(gfx_coord_t page, gfx_coord_t column, uint8_t data, bool force) . . . 55
6.5.3.8 gfx_mono_ssd1306_put_framebuffer(void) . . . 57
6.5.3.9 gfx_mono_ssd1306_put_page(gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width) . . . 57
6.6 SSD1306 OLED Controller Low-level driver . . . 59
6.6.1 Detailed Description . . . 60
6.6.2 Dependencies . . . 60
6.6.3 Macro Definition Documentation . . . 60
6.6.3.1 SSD1306_CMD_ACTIVATE_SCROLL . . . 60
6.6.3.2 SSD1306_CMD_COL_ADD_SET_LSB . . . 60
6.6.3.3 SSD1306_CMD_COL_ADD_SET_MSB . . . 61
6.6.3.4 SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_LEFT . . . 61
6.6.3.5 SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_RIGHT . . . 61
6.6.3.6 SSD1306_CMD_DEACTIVATE_SCROLL . . . 61
6.6.3.7 SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRAM_ON . . . 61
6.6.3.8 SSD1306_CMD_ENTIRE_DISPLAY_ON . . . 61

6.6.3.9 SSD1306_CMD_NOP . . . 61
6.6.3.10 SSD1306_CMD_SCROLL_H_LEFT . . . 61
6.6.3.11 SSD1306_CMD_SCROLL_H_RIGHT . . . 61
6.6.3.12 SSD1306_CMD_SET_CHARGE_PUMP_SETTING . . . 61
6.6.3.13 SSD1306_CMD_SET_COLUMN_ADDRESS . . . 62
6.6.3.14 SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN . . . 62
6.6.3.15 SSD1306_CMD_SET_COM_OUTPUT_SCAN_UP . . . 62
6.6.3.16 SSD1306_CMD_SET_COM_PINS . . . 62
6.6.3.17 SSD1306_CMD_SET_CONTRAST_CONTROL_FOR_BANK0 . . . 62
6.6.3.18 SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_RATIO . . . 62
6.6.3.19 SSD1306_CMD_SET_DISPLAY_OFF . . . 62
6.6.3.20 SSD1306_CMD_SET_DISPLAY_OFFSET . . . 62
6.6.3.21 SSD1306_CMD_SET_DISPLAY_ON . . . 62
6.6.3.22 SSD1306_CMD_SET_DISPLAY_START_LINE . . . 63
6.6.3.23 SSD1306_CMD_SET_INVERSE_DISPLAY . . . 63
6.6.3.24 SSD1306_CMD_SET_MEMORY_ADDRESSING_MODE . . . 63
6.6.3.25 SSD1306_CMD_SET_MULTIPLEX_RATIO . . . 63
6.6.3.26 SSD1306_CMD_SET_NORMAL_DISPLAY . . . 63
6.6.3.27 SSD1306_CMD_SET_PAGE_ADDRESS . . . 63
6.6.3.28 SSD1306_CMD_SET_PAGE_START_ADDRESS . . . 63
6.6.3.29 SSD1306_CMD_SET_PRE_CHARGE_PERIOD . . . 63
6.6.3.30 SSD1306_CMD_SET_SEGMENT_RE_MAP_COL0_SEG0 . . . 63
6.6.3.31 SSD1306_CMD_SET_SEGMENT_RE_MAP_COL127_SEG0 . . . 64
6.6.3.32 SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL . . . 64
6.6.3.33 SSD1306_CMD_SET_VERTICAL_SCROLL_AREA . . . 64
6.6.4 Function Documentation . . . 64
6.6.4.1 ssd1306_init(void) . . . 64
6.6.4.2 ssd1306_write_command(uint8_t command) . . . 66
6.6.4.3 ssd1306_write_data(uint8_t data) . . . 66
6.6.5 Variable Documentation . . . 67
6.6.5.1 ssd1306_master . . . 67
6.6.5.2 ssd1306_slave . . . 67
6.7 Gfx_sysfont . . . 68
6.7.1 Detailed Description . . . 68
6.7.2 Macro Definition Documentation . . . 68
6.7.2.1 SYSFONT_DEFINE_GLYPHS . . . 68
6.7.2.2 SYSFONT_FIRSTCHAR . . . 68
6.7.2.3 SYSFONT_HEIGHT . . . 68
6.7.2.4 SYSFONT_LASTCHAR . . . 68
6.7.2.5 SYSFONT_LINESPACING . . . 68
6.7.2.6 SYSFONT_WIDTH . . . 68
6.7.2.7 USE_FONT_BPMONO_10x16 . . . 68

7 Class Documentation **69**

7.1 font Struct Reference 69

7.1.1 Detailed Description 69

7.1.2 Member Data Documentation 69

7.1.2.1 data 69

7.1.2.2 first_char 69

7.1.2.3 height 70

7.1.2.4 last_char 70

7.1.2.5 progmem 70

7.1.2.6 type 70

7.1.2.7 width 70

7.2 gfx_mono_bitmap Struct Reference 70

7.2.1 Detailed Description 71

7.2.2 Member Data Documentation 71

7.2.2.1 data 71

7.2.2.2 height 71

7.2.2.3 pixmap 71

7.2.2.4 progmem 71

7.2.2.5 type 72

7.2.2.6 width 72

8 File Documentation **73**

8.1 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_definitions.h File Reference . . . 73

8.1.1 Macro Definition Documentation . . . 74

8.1.1.1 GFX_DATA_CMD_SEL_CLEAR . . . 74

8.1.1.2 GFX_DATA_CMD_SEL_SET . . . 74

8.1.1.3 GFX_DELAY_FUNCTION . . . 74

8.1.1.4 GFX_DISPLAY_RESET_CLEAR . . . 74

8.1.1.5 GFX_DISPLAY_RESET_SET . . . 74

8.1.1.6 GFX_DISPLAY_SS_N_CLEAR . . . 74

8.1.1.7 GFX_DISPLAY_SS_N_SET . . . 74

8.1.1.8 GFX_MONO_UG_2832HSWEG04 . . . 75

8.1.1.9 GFX_SPI_IS_BUSY . . . 75

8.1.1.10 GFX_SPI_WRITE_FUNCTION . . . 75

8.1.1.11 PRINTF_BLOCKING . . . 75

8.2 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono.h File Reference . . . 75

8.2.1 Detailed Description . . . 77

8.2.2 Macro Definition Documentation . . . 77

8.2.2.1 PROGMEM_DECLARE . . . 77

8.2.2.2 PROGMEM_PTR_T . . . 77

8.2.2.3 PROGMEM_READ_BYTE . . . 77

8.2.2.4 PROGMEM_STRING_T . . . 77

8.2.2.5 PROGMEM_T . . . 77

8.3 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_framebuffer.c File Reference 77

8.3.1 Detailed Description . . . 78

8.4 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_framebuffer.h File Reference 78

8.4.1 Detailed Description . . . 79

8.5 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.c File Reference . 80

8.5.1 Detailed Description . . . 80

8.6 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.h File Reference . 81

8.6.1 Detailed Description . . . 82

8.7 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.c File Reference . . . 82
8.7.1 Detailed Description . . . 83
8.7.2 Macro Definition Documentation . . . 83
8.7.2.1 CONFIG_FONT_PIXELS_PER_BYTE . . . 83
8.7.2.2 EXTMEM_BUF_SIZE . . . 83
8.8 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.h File Reference . . . 84
8.8.1 Detailed Description . . . 85
8.9 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_ug_2832hsweg04.c File Reference . . . 85
8.9.1 Detailed Description . . . 86
8.9.2 Macro Definition Documentation . . . 86
8.9.2.1 CONFIG_SSD1306_FRAMEBUFFER . . . 86
8.9.3 Variable Documentation . . . 87
8.9.3.1 framebuffer . . . 87
8.10 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_ug_2832hsweg04.h File Reference . . . 87
8.10.1 Detailed Description . . . 89
8.11 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_ssd1306.c File Reference . . . 89
8.11.1 Detailed Description . . . 90
8.12 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_ssd1306.h File Reference . . . 90
8.12.1 Detailed Description . . . 92
8.13 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_sysfont.c File Reference . . . 92
8.13.1 Detailed Description . . . 92
8.13.2 Variable Documentation . . . 93
8.13.2.1 sysfont . . . 93
8.13.2.2 SYSFONT_DEFINE_GLYPHS . . . 93
8.14 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_sysfont.h File Reference . . . 93
8.14.1 Detailed Description . . . 94

Index **95**

Chapter 1

License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCL↩ UDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PAR↩ TICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOO↩ DS 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.

Chapter 2

Examples for GFX Mono Library

This is a list of the available Quick Start guides (QSGs) and example applications for Monochrome graphical display system. QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that QSGs can be compiled as a standalone application or be added to the user application.

- Quick Start Guide for the mono graphics service
- Quick Start Guide for the mono font service
- asfdoc_common2_gfx_mono_sysfont_example
- asfdoc_common2_gfx_mono_spinner_example

2.1 Quick Start Guide for the mono graphics service

This is the quick start guide for the Monochrome Graphics service, with step-by-step instructions on how to configure and use it for a specific use case.

2.1.1 Basic usage of the graphics service

This use case will demonstrate initializing the mono graphics service and then draw a black line on the screen from coordinates X=10, Y=10 to X=20, Y=20.

2.1.2 Usage steps

2.1.2.1 Example code

Add to, e.g., the main function in the application C-file:

```
system_init();
gfx_mono_init();
gfx_mono_draw_line(10, 10, 20, 20, GFX_PIXEL_SET);
```

2.1.2.2 Workflow

1. Initialize system:
   - `system_init();`
2. Initialize monochrome graphics service
   - `gfx_mono_init();`
     Note
     - This will call the init function for the low level display controller driver and intialize the screen to a cleared background.
3. Draw a line from 10,10 to 20,20:

- `gfx_mono_draw_line(10, 10, 20, 20, GFX_PIXEL_SET);`

  Note

- This uses GFX_PIXEL_SET to set the display pixels on the line; other options can be found in gfx_mono_color.

2.2 Quick Start Guide for the mono font service

This is the quick start guide for the GFX Mono Font Library with step-by-step instructions on how to configure and use it for a specific use case.

2.2.1 Basic usage of the graphics service

This use case will demonstrate initializing the mono graphics service and then draw a "Hello world!" sting on the display.

2.2.2 Dependencies

In order to use this quick start, the following dependencies are needed:

- asfdoc_samd20_system_group
- GFX Mono Font Library
- conf_sysfont.h Containing the actual font.

2.2.3 Usage steps

2.2.3.1 Example code

Add to, e.g., the main function in the application C-file:

```
system_init();
gfx_mono_init();
gfx_mono_draw_string("Hello world!",0, 0, &sysfont);
while (1) {
}
```

2.2.3.2 Workflow

1. Initialize system:
   - `system_init();`
2. Initialize monochrome graphics service
   - `gfx_mono_init();`

     Note
   - This will call the init function for the low level display controller driver and intialize the screen to a cleared background.
3. Draw a string on the screen starting at pixel (0,0)

- `gfx_mono_draw_string("Hello world!",0, 0, &sysfont);`

  Note
- This uses conf_sysfont.h where sysfont is defines to give the font to be used on the screen.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

- Monochrome graphical display system . . . 13
  - Framebuffer . . . 22
  - Generic monochrome graphic primitives . . . 30
  - GFX Mono Font Library . . . 40
  - 2832HSWEG04 graphic library abstraction . . . 47
- SSD1306 OLED Controller Low-level driver . . . 59
- Gfx_sysfont . . . 68

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

font 69
gfx_mono_bitmap
Storage structure for bitmap pixel data and metadata 70

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

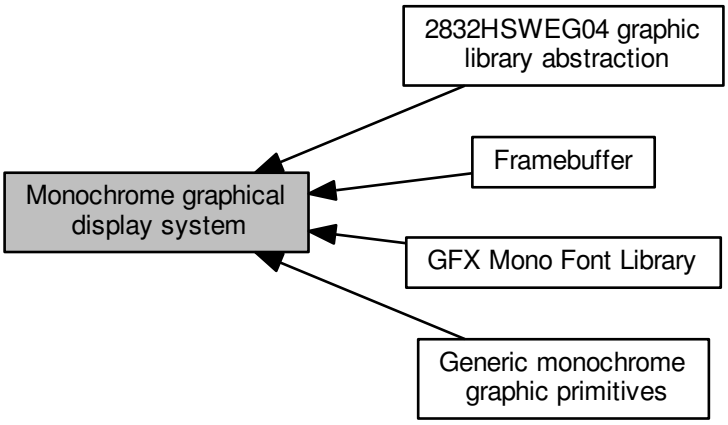
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_definitions.h 73
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono.h
Monochrome graphic library API header file 75
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_framebuffer.c
Local framebuffer 77
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_framebuffer.h
Monochrome graphic library framebuffer device 78
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.c
Generic monochrome LCD graphic primitives 80
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.h
Generic monochrome LCD graphic primitives 81
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.c
Font and text drawing routines 82
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.h
Monochrome graphic library API header file 84
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_ug_2832hsweg04.c
Haven Display UG 2832HSWEG04 display glue code for display controller 85
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_ug_2832hsweg04.h
Haven Display UG 2832HSWEG04 display glue code for display controller 87
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_ssd1306.c
SSD1306 OLED display controller driver 89
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_ssd1306.h
SSD1306 OLED display controller driver 90
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_sysfont.c
Graphical font support 92
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_sysfont.h
Default configurations for sysfont 93

Chapter 6

Module Documentation

6.1 Monochrome graphical display system

Collaboration diagram for Monochrome graphical display system:



Modules

- Framebuffer
- Generic monochrome graphic primitives
- GFX Mono Font Library
- 2832HSWEG04 graphic library abstraction

Typedefs

- typedef uint8_t gfx_mono_color_t
- typedef uint8_t gfx_coord_t

Enumerations

- enum gfx_mono_color { GFX_PIXEL_CLR = 0, GFX_PIXEL_SET = 1, GFX_PIXEL_XOR = 2 }
- enum gfx_mono_bitmap_type { GFX_MONO_BITMAP_RAM, GFX_MONO_BITMAP_PROGMEM }

Circle Sector Definitions

- #define GFX_OCTANT0 (1 << 0)
- #define GFX_OCTANT1 (1 << 1)
- #define GFX_OCTANT2 (1 << 2)
- #define GFX_OCTANT3 (1 << 3)
- #define GFX_OCTANT4 (1 << 4)
- #define GFX_OCTANT5 (1 << 5)
- #define GFX_OCTANT6 (1 << 6)
- #define GFX_OCTANT7 (1 << 7)
- #define GFX_QUADRANT0 (GFX_OCTANT0 | GFX_OCTANT1)
- #define GFX_QUADRANT1 (GFX_OCTANT2 | GFX_OCTANT3)
- #define GFX_QUADRANT2 (GFX_OCTANT4 | GFX_OCTANT5)
- #define GFX_QUADRANT3 (GFX_OCTANT6 | GFX_OCTANT7)
- #define GFX_LEFTHALF (GFX_QUADRANT3 | GFX_QUADRANT0)
- #define GFX_TOPHALF (GFX_QUADRANT0 | GFX_QUADRANT1)
- #define GFX_RIGHTHALF (GFX_QUADRANT1 | GFX_QUADRANT2)
- #define GFX_BOTTOMHALF (GFX_QUADRANT2 | GFX_QUADRANT3)
- #define GFX_WHOLE 0xFF

Graphic Drawing Primitives

- #define gfx_mono_draw_horizontal_line(x, y, length, color) gfx_mono_generic_draw_horizontal_line(x, y, length, color)

  Draw a horizontal line, one pixel wide.
- #define gfx_mono_draw_vertical_line(x, y, length, color) gfx_mono_generic_draw_vertical_line(x, y, length, color)

  Draw a vertical line, one pixel wide.
- #define gfx_mono_draw_line(x1, y1, x2, y2, color) gfx_mono_generic_draw_line(x1, y1, x2, y2, color)

  Draw a line between two arbitrary points.
- #define gfx_mono_draw_rect(x, y, width, height, color) gfx_mono_generic_draw_rect(x, y, width, height, color)

  Draw an outline of a rectangle.
- #define gfx_mono_draw_filled_rect(x, y, width, height, color)

  Draw a filled rectangle.
- #define gfx_mono_draw_circle(x, y, radius, color, octant_mask)

  Draw an outline of a circle or arc.
- #define gfx_mono_draw_filled_circle(x, y, radius, color, quadrant_mask)

  Draw a filled circle or sector.

6.1.1 Detailed Description

See Quick Start Guide for the mono graphics service.

This library provides an interface to drawing graphics on monochrome graphical displays

The graphics drivers consists of the following:

- Display driver interface (gfx_mono.h)
- General graphics drawing primitives (gfx_mono_generic.h)
- Display specific implementation (ex. gfx_mono_ug_2832hsweg04.h)

The generic drawing primitives is a library of functions for drawing graphics primitives such as lines, rectangles and circles. It uses other functions implemented by the display driver for drawing the primitives. The implementation of these functions can optionally be used by a display driver, but if the hardware of the display allows faster handling of any of the primitives, the display driver can implement it directly.

The display specific drivers provides an interface to the graphical display. It implements the low level communication with the display hardware, putting pixels on the display and drawing primitives such as lines, circles and rectangles. Depending on the display driver implementation, drawing the graphics primitives might be handled by the generic graphics drawing primitives rather than the display driver itself.

6.1.2 Examples

The following examples are available for the driver:

- Quick Start Guide for the mono graphics service
- asfdoc_common2_gfx_mono_sysfont_example
- asfdoc_common2_gfx_mono_spinner_example

6.1.3 API Overview

Note

The functions in the library are not interrupt safe.

6.1.4 Macro Definition Documentation

6.1.4.1 #define GFX_BOTTOMHALF (GFX_QUADRANT2 | GFX_QUADRANT3)

Bitmask for drawing bottom half of circle.

Definition at line 169 of file gfx_mono.h.

6.1.4.2 #define GFX_LEFTHALF (GFX_QUADRANT3 | GFX_QUADRANT0)

Bitmask for drawing left half of circle.

Definition at line 163 of file gfx_mono.h.

6.1.4.3 #define gfx_mono_draw_circle(*x, y, radius, color, octant_mask*)

Value:

```
gfx_mono_generic_draw_circle(x, y, radius, color, \
        octant_mask)
```

Draw an outline of a circle or arc.

The radius is the distance from the center to the circumference, which means that the total width or height of a circle will be (radius∗2+1).

The octant_mask parameter is a bitmask that decides which octants of the circle to draw. Use the GFX_OCTANTn, GFX_QUADRANTn, GFX_xHALF and GFX_WHOLE constants and OR them together if required. Radius equal to zero gives a single pixel.

Parameters

in	*x*	X coordinate of center.
in	*y*	Y coordinate of center.
in	*radius*	Circle radius in pixels.
in	*color*	Pixel operation.
in	*octant_mask*	Bitmask indicating which octants to draw.

Definition at line 100 of file gfx_mono_ug_2832hsweg04.h.

6.1.4.4 #define gfx_mono_draw_filled_circle(*x, y, radius, color, quadrant_mask*)

Value:

```
gfx_mono_generic_draw_filled_circle(x, y, radius, \
        color, quadrant_mask)
```

Draw a filled circle or sector.

The radius is the distance from the center to the circumference, which means that the total width or height of a circle will be (radius∗2+1).

The quadrant_mask parameter is a bitmask that decides which quadrants of the circle to draw. Use the GFX_↩ QUADRANTn, GFX_xHALF and GFX_WHOLE constants and OR them together if required. Radius equal to zero gives a single pixel.

Note

This function only supports quadrants while gfx_draw_circle() supports octants. This is to improve performance on drawing filled circles.

Parameters

in	*x*	X coordinate of center.
in	*y*	Y coordinate of center.
in	*radius*	Circle radius in pixels.
in	*color*	Pixel operation.
in	*quadrant_mask*	Bitmask indicating which quadrants to draw.

Definition at line 104 of file gfx_mono_ug_2832hsweg04.h.

6.1.4.5 #define gfx_mono_draw_filled_rect(*x, y, width, height, color*)

Value:

```
gfx_mono_generic_draw_filled_rect(x, y, width,
      height, \
        color)
```

Draw a filled rectangle.

Parameters

in	*x*	X coordinate of the left side.
in	*y*	Y coordinate of the top side.
in	*width*	Width of the rectangle.
in	*height*	Height of the rectangle.
in	*color*	Pixel operation of the line

Definition at line 96 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_draw_char(), and ssd1306_init().

6.1.4.6 #define gfx_mono_draw_horizontal_line(*x, y, length, color*) gfx_mono_generic_draw_horizontal_line(x, y, length, color)

Draw a horizontal line, one pixel wide.

Parameters

in	*x*	X coordinate of leftmost pixel.
in	*y*	Y coordinate of the line.
in	*length*	Length of the line in pixels.
in	*color*	Pixel operation of the line.

Definition at line 84 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_generic_draw_filled_rect(), and gfx_mono_generic_draw_rect().

6.1.4.7 #define gfx_mono_draw_line(*x1, y1, x2, y2, color*) gfx_mono_generic_draw_line(x1, y1, x2, y2, color)

Draw a line between two arbitrary points.

Parameters

in	*x1*	Start X coordinate.
in	*y1*	Start Y coordinate.
in	*x2*	End X coordinate.
in	*y2*	End Y coordinate.
in	*color*	Pixel operation of the line.

Definition at line 90 of file gfx_mono_ug_2832hsweg04.h.

6.1.4.8 #define gfx_mono_draw_rect(*x, y, width, height, color*) gfx_mono_generic_draw_rect(x, y, width, height, color)

Draw an outline of a rectangle.

Parameters

in	*x*	X coordinate of the left side.
in	*y*	Y coordinate of the top side.
in	*width*	Width of the rectangle.
in	*height*	Height of the rectangle.
in	*color*	Pixel operation of the line.

Definition at line 93 of file gfx_mono_ug_2832hsweg04.h.

6.1.4.9 #define gfx_mono_draw_vertical_line(*x, y, length, color*) gfx_mono_generic_draw_vertical_line(x, y, length, color)

Draw a vertical line, one pixel wide.

Parameters

in	*x*	X coordinate of the line.
in	*y*	Y coordinate of the topmost pixel.
in	*length*	Length of the line in pixels.
in	*color*	Pixel operation of the line.

Definition at line 87 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_generic_draw_filled_circle(), and gfx_mono_generic_draw_rect().

6.1.4.10 #define GFX_OCTANT0 (1 << 0)

Bitmask for drawing circle octant 0.

Definition at line 137 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.11 #define GFX_OCTANT1 (1 << 1)

Bitmask for drawing circle octant 1.

Definition at line 139 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.12 #define GFX_OCTANT2 (1 << 2)

Bitmask for drawing circle octant 2.

Definition at line 141 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.13 #define GFX_OCTANT3 (1 << 3)

Bitmask for drawing circle octant 3.

Definition at line 143 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.14 #define GFX_OCTANT4 (1 << 4)

Bitmask for drawing circle octant 4.

Definition at line 145 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.15 #define GFX_OCTANT5 (1 << 5)

Bitmask for drawing circle octant 5.

Definition at line 147 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.16 #define GFX_OCTANT6 (1 << 6)

Bitmask for drawing circle octant 6.

Definition at line 149 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.17 #define GFX_OCTANT7 (1 << 7)

Bitmask for drawing circle octant 7.

Definition at line 151 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_circle().

6.1.4.18 #define GFX_QUADRANT0 (GFX_OCTANT0 | GFX_OCTANT1)

Bitmask for drawing circle quadrant 0.

Definition at line 154 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_filled_circle().

6.1.4.19 #define GFX_QUADRANT1 (GFX_OCTANT2 | GFX_OCTANT3)

Bitmask for drawing circle quadrant 1.

Definition at line 156 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_filled_circle().

6.1.4.20 #define GFX_QUADRANT2 (GFX_OCTANT4 | GFX_OCTANT5)

Bitmask for drawing circle quadrant 2.

Definition at line 158 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_filled_circle().

6.1.4.21 #define GFX_QUADRANT3 (GFX_OCTANT6 | GFX_OCTANT7)

Bitmask for drawing circle quadrant 3.

Definition at line 160 of file gfx_mono.h.

Referenced by gfx_mono_generic_draw_filled_circle().

6.1.4.22 #define GFX_RIGHTHALF (GFX_QUADRANT1 | GFX_QUADRANT2)

Bitmask for drawing right half of circle.

Definition at line 167 of file gfx_mono.h.

6.1.4.23 #define GFX_TOPHALF (GFX_QUADRANT0 | GFX_QUADRANT1)

Bitmask for drawing top half of circle.

Definition at line 165 of file gfx_mono.h.

6.1.4.24 #define GFX_WHOLE 0xFF

Bitmask for drawing whole circle.

Definition at line 172 of file gfx_mono.h.

6.1.5 Typedef Documentation

6.1.5.1 typedef uint8_t gfx_coord_t

Definition at line 100 of file gfx_mono.h.

6.1.5.2 typedef uint8_t gfx_mono_color_t

Definition at line 99 of file gfx_mono.h.

6.1.6 Enumeration Type Documentation

6.1.6.1 enum gfx_mono_bitmap_type

Bitmap types

Enumerator

GFX_MONO_BITMAP_RAM Bitmap stored in SRAM

GFX_MONO_BITMAP_PROGMEM Bitmap stored in progmem

Definition at line 113 of file gfx_mono.h.

```
113                          {
115     GFX_MONO_BITMAP_RAM,
117     GFX_MONO_BITMAP_PROGMEM
118 };
```

6.1.6.2 enum gfx_mono_color

Pixel operations

Enumerator

GFX_PIXEL_CLR Pixel is cleared

GFX_PIXEL_SET Pixel is set on screen (OR)

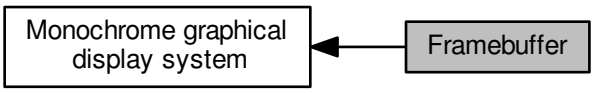
GFX_PIXEL_XOR Pixel is XORed

Definition at line 103 of file gfx_mono.h.

```
103                     {
105     GFX_PIXEL_CLR = 0,
107     GFX_PIXEL_SET = 1,
109     GFX_PIXEL_XOR = 2,
110 };
```

6.2 Framebuffer

Collaboration diagram for Framebuffer:



Functions

- void gfx_mono_set_framebuffer (uint8_t *framebuffer)

  Set the LCD framebuffer.
- void gfx_mono_framebuffer_put_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Put a page from RAM to the framebuffer.
- void gfx_mono_framebuffer_get_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Read a page from the framebuffer.
- void gfx_mono_framebuffer_draw_pixel (gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color)

  Draw pixel to framebuffer.
- uint8_t gfx_mono_framebuffer_get_pixel (gfx_coord_t x, gfx_coord_t y)

  Get the pixel value at x,y in framebuffer.
- void gfx_mono_framebuffer_put_byte (gfx_coord_t page, gfx_coord_t column, uint8_t data)

  Put a byte to the framebuffer.
- uint8_t gfx_mono_framebuffer_get_byte (gfx_coord_t page, gfx_coord_t column)

  Get a byte from the framebuffer.
- void gfx_mono_framebuffer_mask_byte (gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_↩ mask, gfx_mono_color_t color)

  Read/Modify/Write a byte in the framebuffer.

6.2.1 Detailed Description

This module provides read/write from and to a framebuffer in RAM. This is needed when using a controller that does not provide a way to read back data from the LCD controller memory. In this case we need to buffer the data in a local framebuffer to allow manipulation on pixel level. It is generally not recommended to access the framebuffer directly; this is handled by the graphic driver when needed.

6.2.2 Function Documentation

6.2.2.1 void gfx_mono_framebuffer_draw_pixel (gfx_coord_t *x*, gfx_coord_t *y*, gfx_mono_color_t *color*)

Draw pixel to framebuffer.

Parameters

in	*x*	X coordinate of the pixel
in	*y*	Y coordinate of the pixel
in	*color*	Pixel operation

Definition at line 130 of file gfx_mono_framebuffer.c.

References gfx_mono_framebuffer_get_byte(), gfx_mono_framebuffer_put_byte(), GFX_MONO_LCD_HEIGHT, GFX_MONO_LCD_PIXELS_PER_BYTE, GFX_MONO_LCD_WIDTH, GFX_PIXEL_CLR, GFX_PIXEL_SET, and GFX_PIXEL_XOR.

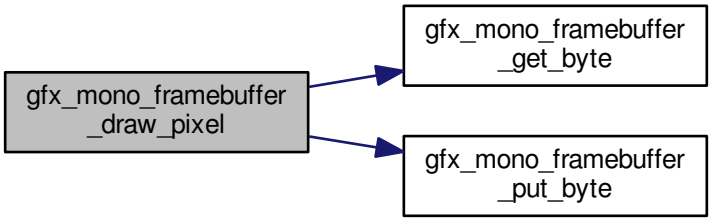
```
131                                 {
132     uint8_t page;
133     uint8_t pixel_mask;
134     uint8_t pixel_value;
135
136     /* Discard pixels drawn outside the screen */
137     if ((x > GFX_MONO_LCD_WIDTH - 1) || (y >
      GFX_MONO_LCD_HEIGHT - 1)) {
138         return;
139     }
140
141     page = y / GFX_MONO_LCD_PIXELS_PER_BYTE;
142     pixel_mask = (1 << (y - (page * 8)));
143
144     /*
145      * Read the page containing the pixel in interest, then perform the

146      * requested action on this pixel before writing the page back to the

147      * display.

148      */
149     pixel_value = gfx_mono_framebuffer_get_byte(page, x);
150
151     switch (color) {
152         case GFX_PIXEL_SET:
153             pixel_value |= pixel_mask;
154             break;
155
156         case GFX_PIXEL_CLR:
157             pixel_value &= ~pixel_mask;
158             break;
159
160         case GFX_PIXEL_XOR:
161             pixel_value ^= pixel_mask;
162             break;
163
164         default:
165             break;
166     }
167
168     gfx_mono_framebuffer_put_byte(page, x, pixel_value);
169 }
```

Here is the call graph for this function:



6.2.2.2 uint8_t gfx_mono_framebuffer_get_byte (gfx_coord_t *page*, gfx_coord_t *column*)

Get a byte from the framebuffer.

Parameters

in	*page*	Page address
in	*column*	Page offset (x coordinate)

Returns

data from LCD controller or framebuffer.

The following code will read the first byte of the framebuffer

```
1 data = gfx_mono_framebuffer_get_byte(0, 0);
```

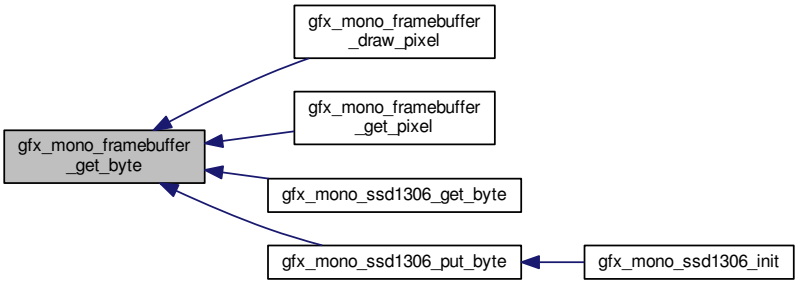
Definition at line 227 of file gfx_mono_framebuffer.c.

References GFX_MONO_LCD_WIDTH.

Referenced by gfx_mono_framebuffer_draw_pixel(), gfx_mono_framebuffer_get_pixel(), gfx_mono_ssd1306_get↩_byte(), and gfx_mono_ssd1306_put_byte().

```
227                                                                    {
228     return *(fbpointer + (page * GFX_MONO_LCD_WIDTH) + column);
229 }
```

Here is the caller graph for this function:



6.2.2.3 void gfx_mono_framebuffer_get_page (gfx_mono_color_t * *data*, gfx_coord_t *page*, gfx_coord_t *column*, gfx_coord_t *width*)

Read a page from the framebuffer.

Parameters

in	*data*	Pointer where to store the read data
in	*page*	Page address
in	*column*	Offset into page (x coordinate)
in	*width*	Number of bytes to be read

The following example will read back the first 128 bytes (first page) from the framebuffer:

```
1 gfx_mono_framebuffer_get_page(read_buffer, 0, 0, 128);
```

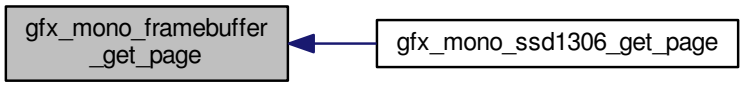
Definition at line 113 of file gfx_mono_framebuffer.c.

References GFX_MONO_LCD_WIDTH.

Referenced by gfx_mono_ssd1306_get_page().

```
114                                                 {
115     gfx_coord_t *framebuffer_pt = fbpointer +
116             ((page * GFX_MONO_LCD_WIDTH) + column);
117     do {
118         *data++ = *framebuffer_pt++;
119     } while (--width > 0);
120 }
```

Here is the caller graph for this function:



6.2.2.4 uint8_t gfx_mono_framebuffer_get_pixel (gfx_coord_t *x*, gfx_coord_t *y*)

Get the pixel value at x,y in framebuffer.

Parameters

in	*x*	X coordinate of pixel
in	*y*	Y coordinate of pixel

Returns

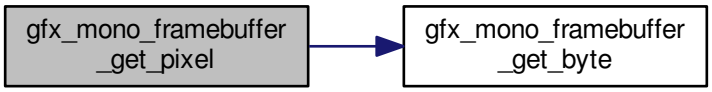
Non zero value if pixel is set.

Definition at line 179 of file gfx_mono_framebuffer.c.

References gfx_mono_framebuffer_get_byte(), GFX_MONO_LCD_HEIGHT, GFX_MONO_LCD_PIXELS_PER_↩ BYTE, and GFX_MONO_LCD_WIDTH.

```
179                                                                     {
180     uint8_t page;
181     uint8_t pixel_mask;
182
183     if ((x > GFX_MONO_LCD_WIDTH - 1) || (y >
      GFX_MONO_LCD_HEIGHT - 1)) {
184         return 0;
185     }
186
187     page = y / GFX_MONO_LCD_PIXELS_PER_BYTE;
188     pixel_mask = (1 << (y - (page * 8)));
189
190     return gfx_mono_framebuffer_get_byte(page, x) & pixel_mask;
191 }
```

Here is the call graph for this function:



6.2.2.5 void gfx_mono_framebuffer_mask_byte (gfx_coord_t *page*, gfx_coord_t *column*, gfx_mono_color_t *pixel_mask*, gfx_mono_color_t *color*)

Read/Modify/Write a byte in the framebuffer.

This function will read the byte from the framebuffer and do a mask operation on the byte according to the pixel operation selected by the color argument and the pixel mask provided.

Parameters

in	*page*	Page address
in	*column*	Page offset (x coordinate)
in	*pixel_mask*	Mask for pixel operation
in	*color*	Pixel operation

A small example that will XOR the first byte of the framebuffer with 0xAA

```
1 gfx_mono_framebuffer_mask_byte(0,0,0xAA,GFX_PIXEL_XOR);
```

Definition at line 248 of file gfx_mono_framebuffer.c.

References gfx_mono_get_byte, gfx_mono_put_byte, GFX_PIXEL_CLR, GFX_PIXEL_SET, and GFX_PIXEL_X↩ OR.

```
249                                                                 {
250     gfx_mono_color_t temp;
251
252     temp = gfx_mono_get_byte(page, column);
253
254     switch (color) {
255         case GFX_PIXEL_SET:
256             temp |= pixel_mask;
257             break;
258
259         case GFX_PIXEL_CLR:
260             temp &= ~pixel_mask;
261             break;
262
263         case GFX_PIXEL_XOR:
264             temp ^= pixel_mask;
265             break;
266     }
267
268     gfx_mono_put_byte(page, column, temp);
269 }
```

6.2.2.6 void gfx_mono_framebuffer_put_byte (gfx_coord_t *page*, gfx_coord_t *column*, uint8_t *data*)

Put a byte to the framebuffer.

Parameters

in	*page*	Page address
in	*column*	Page offset (x coordinate)
in	*data*	Data to be written

This example will put the value 0xFF to the first byte in the framebuffer

```
1 gfx_mono_framebuffer_put_byte(0, 0, 0xFF);
```

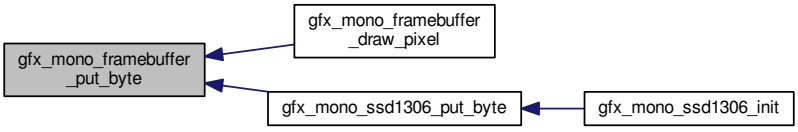
Definition at line 205 of file gfx_mono_framebuffer.c.

References GFX_MONO_LCD_FRAMEBUFFER_SIZE, and GFX_MONO_LCD_WIDTH.

Referenced by gfx_mono_framebuffer_draw_pixel(), and gfx_mono_ssd1306_put_byte().

```
206                         {
207     uint8_t *fBufferBegin = fbpointer;
208     uint8_t *fBufferEnd = fbpointer + GFX_MONO_LCD_FRAMEBUFFER_SIZE;
209     uint8_t *fbpointerTemp = (fbpointer + (page * GFX_MONO_LCD_WIDTH) + column);
210     if ((fbpointerTemp >= fBufferBegin) && (fbpointerTemp <= fBufferEnd)) {
211         *(fbpointer + (page * GFX_MONO_LCD_WIDTH) + column) =
      data;
212     }
213 }
```

Here is the caller graph for this function:



6.2.2.7 void gfx_mono_framebuffer_put_page (gfx_mono_color_t * *data*, gfx_coord_t *page*, gfx_coord_t *column*, gfx_coord_t *width*)

Put a page from RAM to the framebuffer.

Parameters

in	*data*	Pointer to data to be written
in	*page*	Page address
in	*column*	Offset into page (x coordinate)
in	*width*	Number of bytes to be written.

The following example will write 32 bytes from data_buf to the page 0, column 10 (byte 10 to 42 in the framebuffer).

```
1 gfx_mono_framebuffer_put_page(data_buf, 0, 10, 32);
```

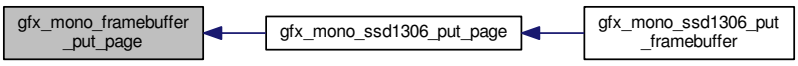
Definition at line 82 of file gfx_mono_framebuffer.c.

References GFX_MONO_LCD_FRAMEBUFFER_SIZE, and GFX_MONO_LCD_WIDTH.

Referenced by gfx_mono_ssd1306_put_page().

```
83                                                  {
84      gfx_mono_color_t *data_pt = data;
85      gfx_coord_t *framebuffer_pt = fbpointer +
86              ((page * GFX_MONO_LCD_WIDTH) + column);
87
88      do {
89          uint8_t *fBufferBegin = fbpointer;
90          uint8_t *fBufferEnd = fbpointer + GFX_MONO_LCD_FRAMEBUFFER_SIZE;
91
92          if ((framebuffer_pt >= fBufferBegin) && (framebuffer_pt <= fBufferEnd)) {
93              *framebuffer_pt++ = *data_pt++;
94          }
95
96      } while (--width > 0);
97 }
```

Here is the caller graph for this function:



6.2.2.8 void gfx_mono_set_framebuffer (uint8_t * *framebuffer*)

Set the LCD framebuffer.

Parameters

in	*framebuffer*	A pointer to an allocated area of RAM that can hold the framebuffer.

A small example:

```
1 uint8_t framebuffer[FRAMEBUFFER_SIZE];
2 gfx_mono_set_framebuffer(framebuffer);
```

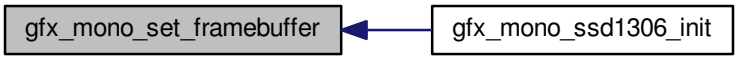
Definition at line 64 of file gfx_mono_framebuffer.c.

References framebuffer.

Referenced by gfx_mono_ssd1306_init().

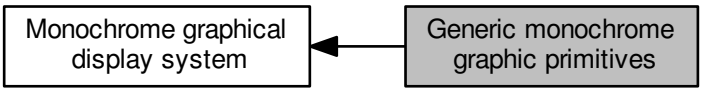
```
64                                                  {
65     fbpointer = framebuffer;
66 }
```

Here is the caller graph for this function:



6.3 Generic monochrome graphic primitives

Collaboration diagram for Generic monochrome graphic primitives:



Classes

- struct gfx_mono_bitmap

  Storage structure for bitmap pixel data and metadata.

Functions

- void gfx_mono_generic_draw_vertical_line (gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_mono_color color)

  Draw a vertical line, one pixel wide (generic implementation)
- void gfx_mono_generic_draw_line (gfx_coord_t x1, gfx_coord_t y1, gfx_coord_t x2, gfx_coord_t y2, enum gfx_mono_color color)

  Draw a line between two arbitrary points (generic implementation).
- void gfx_mono_generic_draw_rect (gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color)

  Draw an outline of a rectangle (generic implementation).
- void gfx_mono_generic_draw_filled_rect (gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color)

  Draw a filled rectangle (generic implementation).
- void gfx_mono_generic_draw_circle (gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t octant_mask)

  Draw an outline of a circle or arc (generic implementation).
- void gfx_mono_generic_draw_filled_circle (gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t quadrant_mask)

  Draw a filled circle or sector (generic implementation).
- void gfx_mono_generic_put_bitmap (struct gfx_mono_bitmap *bitmap, gfx_coord_t x, gfx_coord_t y)

  Put bitmap from FLASH or RAM to display.

- void gfx_mono_generic_draw_horizontal_line (gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_mono_color color)

  Draw a horizontal line, one pixel wide (generic implementation)

6.3.1 Detailed Description

This is a service providing generic implementations of graphic primitives

- Horizontal line
- Vertical line
- Line
- Circle (filled/not filled)
- Rectangle (filled/not filled)

it also provides functionality to draw a bitmap to the graphic memory.

These functions are made available if the graphic hardware being used do not implement the functionality in hardware. This is true in most cases.

This service is included as a requirement for a hardware specific component that uses these functions, and provides a asfdoc_common2_draw_pixel function.

6.3.2 Function Documentation

6.3.2.1 void gfx_mono_generic_draw_circle (gfx_coord_t *x,* gfx_coord_t *y,* gfx_coord_t *radius,* enum gfx_mono_color *color,* uint8_t *octant_mask*)

Draw an outline of a circle or arc (generic implementation).

The radius is the distance from the center to the circumference, which means that the total width or height of a circle will be (radius∗2+1).

The octant_mask parameter is a bitmask that decides which octants of the circle to draw. Use the GFX_OCTANTn, GFX_QUADRANTn, GFX_xHALF and GFX_WHOLE constants and OR them together if required. Radius equal to zero gives a single pixel.

Parameters

in	*x*	X coordinate of center.
in	*y*	Y coordinate of center.
in	*radius*	Circle radius in pixels.
in	*color*	Pixel operation.
in	*octant_mask*	Bitmask indicating which octants to draw.

Definition at line 311 of file gfx_mono_generic.c.

References gfx_mono_draw_pixel, GFX_OCTANT0, GFX_OCTANT1, GFX_OCTANT2, GFX_OCTANT3, GFX_↩ OCTANT4, GFX_OCTANT5, GFX_OCTANT6, and GFX_OCTANT7.

```
313                                {
314     gfx_coord_t offset_x;
315     gfx_coord_t offset_y;
316     int16_t error;
```

```
317
318     /* Draw only a pixel if radius is zero. */
319     if (radius == 0) {
320         gfx_mono_draw_pixel(x, y, color);
321         return;
322     }
323
324     /* Set up start iterators. */
325     offset_x = 0;
326     offset_y = radius;
327     error = 3 - 2 * radius;
328
329     /* Iterate offsetX from 0 to radius. */
330     while (offset_x <= offset_y) {
331         /* Draw one pixel for each octant enabled in octant_mask. */
332         if (octant_mask & GFX_OCTANT0) {
333             gfx_mono_draw_pixel(x + offset_y, y - offset_x, color);
334         }
335
336         if (octant_mask & GFX_OCTANT1) {
337             gfx_mono_draw_pixel(x + offset_x, y - offset_y, color);
338         }
339
340         if (octant_mask & GFX_OCTANT2) {
341             gfx_mono_draw_pixel(x - offset_x, y - offset_y, color);
342         }
343
344         if (octant_mask & GFX_OCTANT3) {
345             gfx_mono_draw_pixel(x - offset_y, y - offset_x, color);
346         }
347
348         if (octant_mask & GFX_OCTANT4) {
349             gfx_mono_draw_pixel(x - offset_y, y + offset_x, color);
350         }
351
352         if (octant_mask & GFX_OCTANT5) {
353             gfx_mono_draw_pixel(x - offset_x, y + offset_y, color);
354         }
355
356         if (octant_mask & GFX_OCTANT6) {
357             gfx_mono_draw_pixel(x + offset_x, y + offset_y, color);
358         }
359
360         if (octant_mask & GFX_OCTANT7) {
361             gfx_mono_draw_pixel(x + offset_y, y + offset_x, color);
362         }
363
364         /* Update error value and step offset_y when required. */
365         if (error < 0) {
366             error += ((offset_x << 2) + 6);
367         } else {
368             error += (((offset_x - offset_y) << 2) + 10);
369             --offset_y;
370         }
371
372         /* Next X. */
373         ++offset_x;
374     }
375 }
```

6.3.2.2 void gfx_mono_generic_draw_filled_circle (gfx_coord_t *x,* gfx_coord_t *y,* gfx_coord_t *radius,* enum gfx_mono_color *color,* uint8_t *quadrant_mask*)

Draw a filled circle or sector (generic implementation).

The radius is the distance from the center to the circumference, which means that the total width or height of a circle will be (radius∗2+1).

The quadrant_mask parameter is a bitmask that decides which quadrants of the circle to draw. Use the GFX_↩ QUADRANTn, GFX_xHALF and GFX_WHOLE constants and OR them together if required. Radius equal to zero gives a single pixel.

Note

This function only supports quadrants while gfx_draw_circle() supports octants. This is to improve performance on drawing filled circles.

Parameters

in	*x*	X coordinate of center.
in	*y*	Y coordinate of center.
in	*radius*	Circle radius in pixels.
in	*color*	Pixel operation.
in	*quadrant_mask*	Bitmask indicating which quadrants to draw.

Definition at line 399 of file gfx_mono_generic.c.

References gfx_mono_draw_pixel, gfx_mono_draw_vertical_line, GFX_QUADRANT0, GFX_QUADRANT1, GFX_QUADRANT2, and GFX_QUADRANT3.

```
401                                 {
402     gfx_coord_t offset_x;
403     gfx_coord_t offset_y;
404     int16_t error;
405
406     /* Draw only a pixel if radius is zero. */
407     if (radius == 0) {
408         gfx_mono_draw_pixel(x, y, color);
409         return;
410     }
411
412     /* Set up start iterators. */
413     offset_x = 0;
414     offset_y = radius;
415     error = 3 - 2 * radius;
416
417     /* Iterate offset_x from 0 to radius. */
418     while (offset_x <= offset_y) {
419         /* Draw vertical lines tracking each quadrant. */
420         if (quadrant_mask & GFX_QUADRANT0) {
421             gfx_mono_draw_vertical_line(x + offset_y,
422                     y - offset_x, offset_x + 1, color);
423             gfx_mono_draw_vertical_line(x + offset_x,
424                     y - offset_y, offset_y + 1, color);
425         }
426
427         if (quadrant_mask & GFX_QUADRANT1) {
428             gfx_mono_draw_vertical_line(x - offset_y,
429                     y - offset_x, offset_x + 1, color);
430             gfx_mono_draw_vertical_line(x - offset_x,
431                     y - offset_y, offset_y + 1, color);
432         }
433
434         if (quadrant_mask & GFX_QUADRANT2) {
435             gfx_mono_draw_vertical_line(x - offset_y,
436                     y, offset_x + 1, color);
437             gfx_mono_draw_vertical_line(x - offset_x,
438                     y, offset_y + 1, color);
439         }
440
441         if (quadrant_mask & GFX_QUADRANT3) {
442             gfx_mono_draw_vertical_line(x + offset_y,
443                     y, offset_x + 1, color);
444             gfx_mono_draw_vertical_line(x + offset_x,
445                     y, offset_y + 1, color);
446         }
447
448         /* Update error value and step offset_y when required. */
449         if (error < 0) {
450             error += ((offset_x << 2) + 6);
451         } else {
452             error += (((offset_x - offset_y) << 2) + 10);
453             --offset_y;
454         }
455
456         /* Next X. */
457         ++offset_x;
458     }
459 }
```

6.3.2.3 void gfx_mono_generic_draw_filled_rect (gfx_coord_t *x*, gfx_coord_t *y*, gfx_coord_t *width*, gfx_coord_t *height*, enum gfx_mono_color *color*)

Draw a filled rectangle (generic implementation).

Parameters

in	*x*	X coordinate of the left side.
in	*y*	Y coordinate of the top side.
in	*width*	Width of the rectangle.
in	*height*	Height of the rectangle.
in	*color*	Pixel operation of the line

Definition at line 280 of file gfx_mono_generic.c.

References gfx_mono_draw_horizontal_line.

```
282                                         {
283     if (height == 0) {
284         /* Nothing to do. Move along. */
285         return;
286     }
287
288     while (height-- > 0) {
289         gfx_mono_draw_horizontal_line(x, y + height,
      width, color);
290     }
291 }
```

6.3.2.4 void gfx_mono_generic_draw_horizontal_line (gfx_coord_t *x*, gfx_coord_t *y*, gfx_coord_t *length*, enum gfx_mono_color *color*)

Draw a horizontal line, one pixel wide (generic implementation)

Note

This function does a very simple bounds checking that does not check if the line is placed outside the screen. If you supply an x- or y-coordinate outside the display the behaviour is undefined, and you risk overwriting portions of internal SRAM.

Parameters

in	*x*	X coordinate of leftmost pixel.
in	*y*	Y coordinate of the line.
in	*length*	Length of the line in pixels.
in	*color*	Pixel operation of the line.

Definition at line 67 of file gfx_mono_generic.c.

References gfx_mono_get_byte, GFX_MONO_LCD_WIDTH, gfx_mono_put_byte, GFX_PIXEL_CLR, GFX_PIX↩EL_SET, and GFX_PIXEL_XOR.

```
68                                                      {
69     uint8_t page;
70     uint8_t pixelmask;
71     uint8_t temp;
72
73     if (x > GFX_MONO_LCD_WIDTH) {
74         return;
75     }
```

```
76
77      /* Clip line length if too long */
78      if (x + length > GFX_MONO_LCD_WIDTH) {
79          length = GFX_MONO_LCD_WIDTH - x;
80      }
81
82      page = y / 8;
83      pixelmask = (1 << (y - (page * 8)));
84
85      if (length == 0) {
86          /* Nothing to do. Move along. */
87          return;
88      }
89
90      switch (color) {
91          case GFX_PIXEL_SET:
92              while (length-- > 0) {
93                  temp = gfx_mono_get_byte(page, x + length);
94                  temp |= pixelmask;
95                  gfx_mono_put_byte(page, x + length, temp);
96              }
97              break;
98
99          case GFX_PIXEL_CLR:
100             while (length-- > 0) {
101                 temp = gfx_mono_get_byte(page, x + length);
102                 temp &= ~pixelmask;
103                 gfx_mono_put_byte(page, x + length, temp);
104             }
105             break;
106
107         case GFX_PIXEL_XOR:
108             while (length-- > 0) {
109                 temp = gfx_mono_get_byte(page, x + length);
110                 temp ^= pixelmask;
111                 gfx_mono_put_byte(page, x + length, temp);
112             }
113             break;
114
115         default:
116             break;
117     }
118 }
```

6.3.2.5 void gfx_mono_generic_draw_line (gfx_coord_t *x1*, gfx_coord_t *y1*, gfx_coord_t *x2*, gfx_coord_t *y2*, enum gfx_mono_color *color*)

Draw a line between two arbitrary points (generic implementation).

Parameters

in	*x1*	Start X coordinate.
in	*y1*	Start Y coordinate.
in	*x2*	End X coordinate.
in	*y2*	End Y coordinate.
in	*color*	Pixel operation of the line.

Definition at line 183 of file gfx_mono_generic.c.

References gfx_mono_draw_pixel.

```
185                                     {
186     uint8_t i;
187     uint8_t x;
188     uint8_t y;
189     int8_t xinc;
190     int8_t yinc;
191     int8_t dx;
192     int8_t dy;
193     int8_t e;
```

```
194
195     /* swap x1,y1  with x2,y2 */
196     if (x1 > x2) {
197         dx = x1;
198         x1 = x2;
199         x2 = dx;
200         dy = y1;
201         y1 = y2;
202         y2 = dy;
203     }
204
205     dx = x2 - x1;
206     dy = y2 - y1;
207
208     x = x1;
209     y = y1;
210
211     if (dx < 0) {
212         xinc = -1;
213         dx = -dx;
214     } else {
215         xinc = 1;
216     }
217
218     if (dy < 0) {
219         yinc = -1;
220         dy = -dy;
221     } else {
222         yinc = 1;
223     }
224
225     if (dx > dy) {
226         e = dy - dx;
227         for (i = 0; i <= dx; i++) {
228             gfx_mono_draw_pixel(x, y, color);
229             if (e >= 0) {
230                 e -= dx;
231                 y += yinc;
232             }
233
234             e += dy;
235             x += xinc;
236         }
237     } else {
238         e = dx - dy;
239         for (i = 0; i <= dy; i++) {
240             gfx_mono_draw_pixel(x, y, color);
241             if (e >= 0) {
242                 e -= dy;
243                 x += xinc;
244             }
245
246             e += dx;
247             y += yinc;
248         }
249     }
250 }
```

6.3.2.6 void gfx_mono_generic_draw_rect (gfx_coord_t *x*, gfx_coord_t *y*, gfx_coord_t *width*, gfx_coord_t *height*, enum gfx_mono_color *color*)

Draw an outline of a rectangle (generic implementation).

Parameters

in	*x*	X coordinate of the left side.
in	*y*	Y coordinate of the top side.
in	*width*	Width of the rectangle.
in	*height*	Height of the rectangle.
in	*color*	Pixel operation of the line.

Definition at line 261 of file gfx_mono_generic.c.

References gfx_mono_draw_horizontal_line, and gfx_mono_draw_vertical_line.

```
263                                     {
264     gfx_mono_draw_horizontal_line(x, y, width, color);
265     gfx_mono_draw_horizontal_line(x, y + height - 1,
      width, color);
266
267     gfx_mono_draw_vertical_line(x, y, height, color);
268     gfx_mono_draw_vertical_line(x + width - 1, y,
      height, color);
269 }
```

6.3.2.7 void gfx_mono_generic_draw_vertical_line (gfx_coord_t *x*, gfx_coord_t *y*, gfx_coord_t *length*, enum gfx_mono_color *color*)

Draw a vertical line, one pixel wide (generic implementation)

Note

This function does a very simple bounds checking that does not check if the line is placed outside the screen. If you supply an x- or y-coordinate outside the display the behaviour is undefined, and you risk overwriting portions of internal SRAM.

Parameters

in	*x*	X coordinate of the line.
in	*y*	Y coordinate of the topmost pixel.
in	*length*	Length of the line in pixels.
in	*color*	Pixel operation of the line.

Definition at line 133 of file gfx_mono_generic.c.

References gfx_mono_draw_pixel, GFX_MONO_LCD_HEIGHT, and gfx_mono_mask_byte.

```
134                                                     {
135     if (length == 0) {
136         return;
137     }
138
139     gfx_coord_t y2 = y + length - 1;
140
141     if (y == y2) {
142         gfx_mono_draw_pixel(x, y, color);
143         return;
144     }
145
146     if (y2 >= GFX_MONO_LCD_HEIGHT - 1) {
147         y2 = GFX_MONO_LCD_HEIGHT - 1;
148     }
149
150     gfx_coord_t y1page = y / 8;
151     gfx_coord_t y2page = y2 / 8;
152
153     uint8_t y1bitpos = y & 0x07;
154     uint8_t y2bitpos = y2 & 0x07;
155
156     uint8_t y1pixelmask = 0xFF << y1bitpos;
157     uint8_t y2pixelmask = 0xFF >> (7 - y2bitpos);
158
159     /* The pixels are on the same page; combine masks */
160     if (y1page == y2page) {
161         uint8_t pixelmask = y1pixelmask & y2pixelmask;
162         gfx_mono_mask_byte(y1page, x, pixelmask, color);
163     } else {
```

```
164         gfx_mono_mask_byte(y1page, x, y1pixelmask, color);
165
166         while (++y1page < y2page) {
167             gfx_mono_mask_byte(y1page, x, 0xFF, color);
168         }
169
170         gfx_mono_mask_byte(y2page, x, y2pixelmask, color);
171     }
172 }
```

6.3.2.8 void gfx_mono_generic_put_bitmap (struct gfx_mono_bitmap * *bitmap,* gfx_coord_t *x,* gfx_coord_t *y*)

Put bitmap from FLASH or RAM to display.

This function will output bitmap data from FLASH or RAM. The bitmap y-coordinate will be aligned with display pages, rounded down. Ie: placing a bitmap at x=10, y=5 will put the bitmap at x = 10,y = 0 and placing a bitmap at x = 10, y = 10 will put the bitmap at x = 10, y = 8

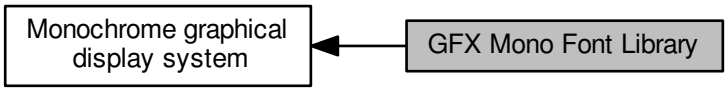
Definition at line 470 of file gfx_mono_generic.c.

References gfx_mono_bitmap::data, GFX_MONO_BITMAP_PROGMEM, GFX_MONO_BITMAP_RAM, gfx_↩ mono_put_byte, gfx_mono_put_page, gfx_mono_bitmap::height, gfx_mono_bitmap::pixmap, gfx_mono_bitmap↩ ::progmem, PROGMEM_READ_BYTE, gfx_mono_bitmap::type, and gfx_mono_bitmap::width.

```
471                         {
472     gfx_coord_t num_pages = bitmap->height / 8;
473     gfx_coord_t page = y / 8;
474     gfx_coord_t column;
475     gfx_coord_t i;
476     gfx_mono_color_t temp;
477
478     switch (bitmap->type) {
479         case GFX_MONO_BITMAP_PROGMEM:
480             for (i = 0; i < num_pages; i++) {
481                 for (column = 0; column < bitmap->width; column++) {
482                     temp = PROGMEM_READ_BYTE(bitmap->data.
    progmem
483                             + (i * bitmap->width)
484                             + column);
485                     gfx_mono_put_byte(i + page, column + x, temp);
486                 }
487             }
488             break;
489
490         case GFX_MONO_BITMAP_RAM:
491             for (i = 0; i < num_pages; i++) {
492                 gfx_mono_put_page(bitmap->data.pixmap
493                         + (i * bitmap->width), page + i, x,
494                         bitmap->width);
495             }
496             break;
497
498         default:
499             break;
500     }
501 }
```

6.4 GFX Mono Font Library

Collaboration diagram for GFX Mono Font Library:



Classes

- struct font

Enumerations

- enum font_data_type { FONT_LOC_PROGMEM }

  Valid storage locations for font data.

Strings and characters located in RAM

- void gfx_mono_draw_char (const char c, const gfx_coord_t x, const gfx_coord_t y, const struct font *font)

  Draws a character to the display.
- void gfx_mono_draw_string (const char *str, const gfx_coord_t x, const gfx_coord_t y, const struct font *font)

  Draws a string to the display.
- void gfx_mono_get_string_bounding_box (char const *str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height)

  Computes the bounding box of a string.

Strings located in flash

- void gfx_mono_draw_progmem_string (char PROGMEM_PTR_T str, gfx_coord_t x, gfx_coord_t y, const struct font *font)

  Draws a string located in program memory to the display.
- void gfx_mono_get_progmem_string_bounding_box (char PROGMEM_PTR_T str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height)

  Computes the bounding box of a string located in program memory.

6.4.1 Detailed Description

This modules provides functionality for outputting a monochrome font to a display.

6.4.2 API Overview

6.4.3 Enumeration Type Documentation

6.4.3.1 enum font_data_type

Valid storage locations for font data.

Add support for fonts in regular ram

Enumerator

FONT_LOC_PROGMEM Font data stored in program/flash memory.

Definition at line 71 of file gfx_mono_text.h.

```
71                          {
73          FONT_LOC_PROGMEM,
74      };
```

6.4.4 Function Documentation

6.4.4.1 void gfx_mono_draw_char (const char *c*, const gfx_coord_t *x*, const gfx_coord_t *y*, const struct font * *font*)

Draws a character to the display.

Parameters

in	*c*	Character to be drawn
in	*x*	X coordinate on screen.
in	*y*	Y coordinate on screen.
in	*font*	Font to draw character in

Definition at line 222 of file gfx_mono_text.c.

References FONT_LOC_PROGMEM, gfx_mono_draw_filled_rect, GFX_PIXEL_CLR, font::height, font::type, and font::width.

Referenced by gfx_mono_draw_progmem_string(), and gfx_mono_draw_string().

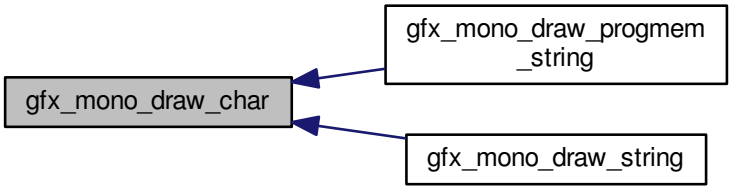
```
223                                 {
224      gfx_mono_draw_filled_rect(x, y, font->width, font->
     height,
225              GFX_PIXEL_CLR);
226
227      switch (font->type) {
228          case FONT_LOC_PROGMEM:
229              gfx_mono_draw_char_progmem(c, x, y, font);
230              break;
231
232 #ifdef CONFIG_HUGEMEM

233          case FONT_LOC_HUGEMEM:
234              gfx_mono_draw_char_hugemem(c, x, y, font);
235              break;
236
```

```
237 #endif

238         default:
239             /* Unsupported mode, call assert */
240             assert(false);
241             break;
242     }
243 }
```

Here is the caller graph for this function:



6.4.4.2 void gfx_mono_draw_progmem_string (char PROGMEM_PTR_T *str*, gfx_coord_t *x*, gfx_coord_t *y*, const struct font * *font*)

Draws a string located in program memory to the display.

This function will draw a string located in program memory to the display, this differs from gfx_mono_draw_string() by using constant string data from the program memory instead of string data in RAM.

Using program memory for constant strings will reduce the applications need for RAM, and thus lower the overall size footprint.

Parameters

in	*str*	Pointer to string located in program memory
in	*x*	X coordinate on screen.
in	*y*	Y coordinate on screen.
in	*font*	Font to draw string in

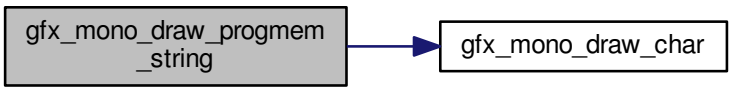
Definition at line 294 of file gfx_mono_text.c.

References gfx_mono_draw_char(), font::height, PROGMEM_PTR_T, PROGMEM_READ_BYTE, and font::width.

```
295                                                 {
296     char temp_char;
297
298     /* Sanity check on parameters, assert if str or font is NULL. */
299     assert(str != NULL);
300     assert(font != NULL);
301
302     /* Save X in order to know where to return to on CR. */
303     const gfx_coord_t start_of_string_position_x = x;
304
```

```
305     /* Draw characters until trailing null byte */
306     temp_char = PROGMEM_READ_BYTE((uint8_t PROGMEM_PTR_T) str);
307
308     while (temp_char) {
309         /* Handle '\n' as newline, draw normal characters. */
310         if (temp_char == '\n') {
311             x = start_of_string_position_x;
312             y += font->height + 1;
313         } else if (temp_char == '\r') {
314             /* Skip '\r' characters. */
315         } else {
316             gfx_mono_draw_char(temp_char, x, y, font);
317             x += font->width;
318         }
319
320         temp_char = PROGMEM_READ_BYTE((uint8_t PROGMEM_PTR_T) (++str));
321     }
322 }
```

Here is the call graph for this function:



6.4.4.3 void gfx_mono_draw_string (const char * *str*, gfx_coord_t *x*, gfx_coord_t *y*, const struct font * *font*)

Draws a string to the display.

This function will draw a string located in memory to the display.

Parameters

in	*str*	Pointer to string
in	*x*	X coordinate on screen.
in	*y*	Y coordinate on screen.
in	*font*	Font to draw string in

Definition at line 255 of file gfx_mono_text.c.

References gfx_mono_draw_char(), font::height, and font::width.

```
256                                 {
257     /* Save X in order to know where to return to on CR. */
258     const gfx_coord_t start_of_string_position_x = x;
259
260     /* Sanity check on parameters, assert if str or font is NULL. */
261     assert(str != NULL);
262     assert(font != NULL);
263
264     /* Draw characters until trailing null byte */
265     do {
266         /* Handle '\n' as newline, draw normal characters. */
267         if (*str == '\n') {
268             x = start_of_string_position_x;
```

```
269             y += font->height + 1;
270         } else if (*str == '\r') {
271             /* Skip '\r' characters. */
272         } else {
273             gfx_mono_draw_char(*str, x, y, font);
274             x += font->width;
275         }
276     } while (*(++str));
277 }
```

Here is the call graph for this function:

6.4.4.4 void gfx_mono_get_progmem_string_bounding_box (char PROGMEM_PTR_T *str*, const struct font * *font*, gfx_coord_t * *width*, gfx_coord_t * *height*)

Computes the bounding box of a string located in program memory.

Note

If string is empty the returned width will be 1 pixel and the height equal to the font height.

Parameters

in	*str*	String in program memory to calculate bounding box for
in	*font*	Font used
in	*width*	Pointer to width result
in	*height*	Pointer to height result

Definition at line 380 of file gfx_mono_text.c.

References font::height, PROGMEM_PTR_T, PROGMEM_READ_BYTE, and font::width.

```
382                             {
383     gfx_coord_t font_width = font->width;
384     gfx_coord_t font_height = font->height;
385
386     char temp_char;
387     gfx_coord_t max_width = 1;
388     gfx_coord_t max_height = font_height;
389     gfx_coord_t x = 0;
390
391     /* Sanity check on parameters, assert if str or font is NULL. */
392     assert(str != NULL);
393     assert(font != NULL);
394
395     /* Handle each character until trailing null byte */
396     temp_char = PROGMEM_READ_BYTE((uint8_t PROGMEM_PTR_T) str);
397
398     while (temp_char) {
```

```
399         /* Handle '\n' as newline, draw normal characters. */
400         if (temp_char == '\n') {
401             x = 0;
402             max_height += font_height;
403         } else if (*str == '\r') {
404             /* Skip '\r' characters. */
405         } else {
406             x += font_width;
407             if (x > max_width) {
408                 max_width = x;
409             }
410         }
411
412         temp_char = PROGMEM_READ_BYTE((uint8_t PROGMEM_PTR_T) (++str));
413     }
414
415     /* Return values through references */
416     *width = max_width;
417     *height = max_height;
418 }
```

6.4.4.5 void gfx_mono_get_string_bounding_box (const char * *str*, const struct font * *font*, gfx_coord_t * *width*, gfx_coord_t * *height*)

Computes the bounding box of a string.

Note

If string is empty the returned width will be 1 pixel and the height equal to the font height.

Parameters

in	*str*	String to calculate bounding box for
in	*font*	Font used
in	*width*	Pointer to width result
in	*height*	Pointer to height result

Definition at line 335 of file gfx_mono_text.c.

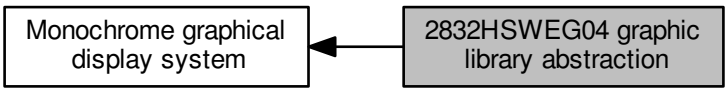
References font::height, and font::width.

```
336                                                 {
337     gfx_coord_t font_width = font->width;
338     gfx_coord_t font_height = font->height;
339
340     gfx_coord_t max_width = 1;
341     gfx_coord_t max_height = font_height;
342     gfx_coord_t x = 0;
343
344     /* Sanity check on parameters, assert if str or font is NULL. */
345     assert(str != NULL);
346     assert(font != NULL);
347
348     /* Handle each character until trailing null byte */
349     do {
350         /* Handle '\n' as newline, draw normal characters. */
351         if (*str == '\n') {
352             x = 0;
353             max_height += font_height;
354         } else if (*str == '\r') {
355             /* Skip '\r' characters. */
356         } else {
357             x += font_width;
358             if (x > max_width) {
359                 max_width = x;
360             }
361         }
```

```
362     } while (*(++str));
363
364     /* Return values through references */
365     *width = max_width;
366     *height = max_height;
367 }
```

6.5 2832HSWEG04 graphic library abstraction

Collaboration diagram for 2832HSWEG04 graphic library abstraction:



Macros

- #define GFX_MONO_LCD_WIDTH 128
- #define GFX_MONO_LCD_HEIGHT 32
- #define GFX_MONO_LCD_PIXELS_PER_BYTE 8
- #define GFX_MONO_LCD_PAGES
- #define GFX_MONO_LCD_FRAMEBUFFER_SIZE
- #define gfx_mono_put_bitmap(bitmap, x, y) gfx_mono_generic_put_bitmap(bitmap, x, y)
- #define gfx_mono_draw_pixel(x, y, color) gfx_mono_ssd1306_draw_pixel(x, y, color)
- #define gfx_mono_get_pixel(x, y) gfx_mono_ssd1306_get_pixel(x, y)
- #define gfx_mono_init() gfx_mono_ssd1306_init()
- #define gfx_mono_put_page(data, page, column, width) gfx_mono_ssd1306_put_page(data, page, column, width)
- #define gfx_mono_get_page(data, page, column, width) gfx_mono_ssd1306_get_page(data, page, column, width)
- #define gfx_mono_put_byte(page, column, data) gfx_mono_ssd1306_put_byte(page, column, data, false)
- #define gfx_mono_get_byte(page, column) gfx_mono_ssd1306_get_byte(page, column)
- #define gfx_mono_mask_byte(page, column, pixel_mask, color) gfx_mono_ssd1306_mask_byte(page, column, pixel_mask, color)
- #define gfx_mono_put_framebuffer() gfx_mono_ssd1306_put_framebuffer()

Functions

- void gfx_mono_ssd1306_put_framebuffer (void)

  Put framebuffer to LCD controller.
- void gfx_mono_ssd1306_put_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Put a page from RAM to display controller.
- void gfx_mono_ssd1306_get_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Read a page from the LCD controller.
- void gfx_mono_ssd1306_init (void)

  Initialize SSD1306 controller and LCD display. It will also write the graphic controller RAM to all zeroes.
- void gfx_mono_ssd1306_draw_pixel (gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color)

  Draw pixel to screen.
- uint8_t gfx_mono_ssd1306_get_pixel (gfx_coord_t x, gfx_coord_t y)

  Get the pixel value at x,y.

- void gfx_mono_ssd1306_put_byte (gfx_coord_t page, gfx_coord_t column, uint8_t data, bool force)

  Put a byte to the display controller RAM.
- uint8_t gfx_mono_ssd1306_get_byte (gfx_coord_t page, gfx_coord_t column)

  Get a byte from the display controller RAM.
- void gfx_mono_ssd1306_mask_byte (gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_mask, gfx_mono_color_t color)

  Read/Modify/Write a byte on the display controller.

6.5.1 Detailed Description

This module is an abstraction layer between the graphic library and the 2832HSWEG04 monochrome LCD display connected to a SSD1306 LCD controller.

As the controller does not provide any hardware accelerated graphic, all the graphic primitives are provided by the Generic monochrome graphic primitives service.

Note

Do not call the gfx_mono_ssd1306_ functions directly. use the gfx_mono names that are defined in this header and documented in Monochrome graphical display system . Ie. gfx_mono_draw_pixel() should be used, not gfx_mono_ssd1306_draw_pixel()

6.5.2 Macro Definition Documentation

6.5.2.1 #define gfx_mono_draw_pixel(*x, y, color*) gfx_mono_ssd1306_draw_pixel(x, y, color)

Definition at line 111 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_generic_draw_circle(), gfx_mono_generic_draw_filled_circle(), gfx_mono_generic_draw_line(), and gfx_mono_generic_draw_vertical_line().

6.5.2.2 #define gfx_mono_get_byte(*page, column*) gfx_mono_ssd1306_get_byte(page, column)

Definition at line 129 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_framebuffer_mask_byte(), gfx_mono_generic_draw_horizontal_line(), gfx_mono_ssd1306_draw_pixel(), gfx_mono_ssd1306_get_pixel(), and gfx_mono_ssd1306_mask_byte().

6.5.2.3 #define gfx_mono_get_page(*data, page, column, width*) gfx_mono_ssd1306_get_page(data, page, column, width)

Definition at line 123 of file gfx_mono_ug_2832hsweg04.h.

6.5.2.4 #define gfx_mono_get_pixel(*x, y*) gfx_mono_ssd1306_get_pixel(x, y)

Definition at line 114 of file gfx_mono_ug_2832hsweg04.h.

6.5.2.5 #define gfx_mono_init( ) gfx_mono_ssd1306_init()

Definition at line 117 of file gfx_mono_ug_2832hsweg04.h.

6.5.2.6 #define GFX_MONO_LCD_FRAMEBUFFER_SIZE

Value:

```
((GFX_MONO_LCD_WIDTH * \
    GFX_MONO_LCD_HEIGHT) / GFX_MONO_LCD_PIXELS_PER_BYTE)
```

Definition at line 81 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_framebuffer_put_byte(), gfx_mono_framebuffer_put_page(), and gfx_mono_ssd1306_↩ init().

6.5.2.7 #define GFX_MONO_LCD_HEIGHT 32

Definition at line 76 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_framebuffer_draw_pixel(), gfx_mono_framebuffer_get_pixel(), gfx_mono_generic_↩ draw_vertical_line(), gfx_mono_ssd1306_draw_pixel(), and gfx_mono_ssd1306_get_pixel().

6.5.2.8 #define GFX_MONO_LCD_PAGES

Value:

```
(GFX_MONO_LCD_HEIGHT / \
    GFX_MONO_LCD_PIXELS_PER_BYTE)
```

Definition at line 79 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_ssd1306_init(), and gfx_mono_ssd1306_put_framebuffer().

6.5.2.9 #define GFX_MONO_LCD_PIXELS_PER_BYTE 8

Definition at line 78 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_framebuffer_draw_pixel(), gfx_mono_framebuffer_get_pixel(), gfx_mono_ssd1306_↩ draw_pixel(), and gfx_mono_ssd1306_get_pixel().

6.5.2.10 #define GFX_MONO_LCD_WIDTH 128

Definition at line 74 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_framebuffer_draw_pixel(), gfx_mono_framebuffer_get_byte(), gfx_mono_framebuffer_↩ get_page(), gfx_mono_framebuffer_get_pixel(), gfx_mono_framebuffer_put_byte(), gfx_mono_framebuffer_put_↩ page(), gfx_mono_generic_draw_horizontal_line(), gfx_mono_ssd1306_draw_pixel(), gfx_mono_ssd1306_get_↩ pixel(), gfx_mono_ssd1306_init(), and gfx_mono_ssd1306_put_framebuffer().

6.5.2.11 #define gfx_mono_mask_byte(*page, column, pixel_mask, color*) gfx_mono_ssd1306_mask_byte(page, column, pixel_mask, color)

Definition at line 132 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_generic_draw_vertical_line().

6.5.2.12 #define gfx_mono_put_bitmap(*bitmap, x, y*) gfx_mono_generic_put_bitmap(bitmap, x, y)

Definition at line 108 of file gfx_mono_ug_2832hsweg04.h.

6.5.2.13 #define gfx_mono_put_byte(*page, column, data*) gfx_mono_ssd1306_put_byte(page, column, data, false)

Definition at line 126 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_framebuffer_mask_byte(), gfx_mono_generic_draw_horizontal_line(), gfx_mono_↩ generic_put_bitmap(), gfx_mono_ssd1306_draw_pixel(), and gfx_mono_ssd1306_mask_byte().

6.5.2.14 #define gfx_mono_put_framebuffer( ) gfx_mono_ssd1306_put_framebuffer()

Definition at line 135 of file gfx_mono_ug_2832hsweg04.h.

6.5.2.15 #define gfx_mono_put_page(*data, page, column, width*) gfx_mono_ssd1306_put_page(data, page, column, width)

Definition at line 120 of file gfx_mono_ug_2832hsweg04.h.

Referenced by gfx_mono_generic_put_bitmap().

6.5.3 Function Documentation

6.5.3.1 void gfx_mono_ssd1306_draw_pixel (gfx_coord_t *x,* gfx_coord_t *y,* gfx_coord_t *color*)

Draw pixel to screen.

Parameters

in	*x*	X coordinate of the pixel
in	*y*	Y coordinate of the pixel
in	*color*	Pixel operation

The following will set the pixel at x=10,y=10:

```
1 gfx_mono_ssd1306_draw_pixel(10, 10, GFX_PIXEL_SET);
```

The following example will clear the pixel at x=10,y=10:

```
1 gfx_mono_ssd1306_draw_pixel(10, 10, GFX_PIXEL_CLR);
```

And the following will toggle the pixel at x=10,y=10:

```
1 gfx_mono_ssd1306_draw_pixel(10, 10, GFX_PIXEL_XOR);
```

Definition at line 136 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_get_byte, GFX_MONO_LCD_HEIGHT, GFX_MONO_LCD_PIXELS_PER_BYTE, GFX_↩ MONO_LCD_WIDTH, gfx_mono_put_byte, GFX_PIXEL_CLR, GFX_PIXEL_SET, and GFX_PIXEL_XOR.

```
137                             {
138     uint8_t page;
139     uint8_t pixel_mask;
140     uint8_t pixel_value;
141
142     /* Discard pixels drawn outside the screen */
143     if ((x > GFX_MONO_LCD_WIDTH - 1) || (y >
      GFX_MONO_LCD_HEIGHT - 1)) {
144         return;
145     }
146
147     page = y / GFX_MONO_LCD_PIXELS_PER_BYTE;
148     pixel_mask = (1 << (y - (page * 8)));
149
150     /*
151      * Read the page containing the pixel in interest, then perform the
152      * requested action on this pixel before writing the page back to the
153      * display.
154      */
155     pixel_value = gfx_mono_get_byte(page, x);
156
157     switch (color) {
158         case GFX_PIXEL_SET:
159             pixel_value |= pixel_mask;
160             break;
161
162         case GFX_PIXEL_CLR:
163             pixel_value &= ~pixel_mask;
164             break;
165
166         case GFX_PIXEL_XOR:
167             pixel_value ^= pixel_mask;
168             break;
169
170         default:
171             break;
172     }
173
174     gfx_mono_put_byte(page, x, pixel_value);
175 }
```

6.5.3.2 uint8_t gfx_mono_ssd1306_get_byte (gfx_coord_t *page*, gfx_coord_t *column*)

Get a byte from the display controller RAM.

If the LCD controller is accessed by the SPI interface we cannot read the data. In this case return the data from the local framebuffer instead.

Parameters

in	*page*	Page address
in	*column*	Page offset (x coordinate)

Returns

data from LCD controller or framebuffer.

The following code will read the first byte from the display memory or the local framebuffer if direct read is not possible. The data represents the pixels from x = 0 and y = 0 to y = 7.

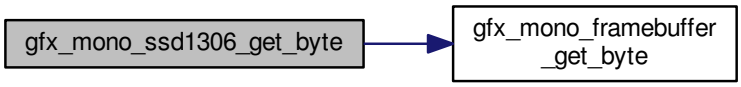
```
1 data = gfx_mono_ssd1306_get_byte(0, 0);
```

Definition at line 318 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_framebuffer_get_byte().

```
318                                                                    {
319 #ifdef CONFIG_SSD1306_FRAMEBUFFER
320     return gfx_mono_framebuffer_get_byte(page, column);
321
322 #else
323     ssd1306_set_page_address(page);
324     ssd1306_set_column_address(column);
325
326     return ssd1306_read_data();
327
328 #endif
329 }
```

Here is the call graph for this function:



6.5.3.3 void gfx_mono_ssd1306_get_page (gfx_mono_color_t * *data*, gfx_coord_t *page*, gfx_coord_t *column*, gfx_coord_t *width*)

Read a page from the LCD controller.

If the LCD controller is accessed by the SPI interface we cannot read data directly from the controller. In that case we will read the data from the local framebuffer instead.

Parameters

in	*data*	Pointer where to store the read data
in	*page*	Page address
in	*column*	Offset into page (x coordinate)
in	*width*	Number of bytes to be read

The following example will read back the first 128 bytes (first page) from the display memory:

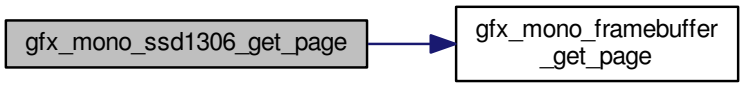
```
1 gfx_mono_ssd1306_get_page(read_buffer, 0, 0, 128);
```

Definition at line 254 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_framebuffer_get_page().

```
255                                                 {
256 #ifdef CONFIG_SSD1306_FRAMEBUFFER
257     gfx_mono_framebuffer_get_page(data, page, column,
      width);
258 #else
259     ssd1306_set_page_address(page);
260     ssd1306_set_column_address(column);
261
262     do {
263         *data++ = ssd1306_read_data();
264     } while (--width);
265 #endif
266 }
```

Here is the call graph for this function:



6.5.3.4 uint8_t gfx_mono_ssd1306_get_pixel (gfx_coord_t *x*, gfx_coord_t *y*)

Get the pixel value at x,y.

Parameters

in	*x*	X coordinate of pixel
in	*y*	Y coordinate of pixel

Returns

Non zero value if pixel is set.

The following example will read the pixel value from x=10,y=10:

```
1 pixelval = gfx_mono_ssd1306_get_pixel(10,10);
```

Definition at line 189 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_get_byte, GFX_MONO_LCD_HEIGHT, GFX_MONO_LCD_PIXELS_PER_BYTE, and GF↩X_MONO_LCD_WIDTH.

```
189                                                                 {
190     uint8_t page;
191     uint8_t pixel_mask;
192
193     if ((x > GFX_MONO_LCD_WIDTH - 1) || (y >
      GFX_MONO_LCD_HEIGHT - 1)) {
194         return 0;
195     }
196
197     page = y / GFX_MONO_LCD_PIXELS_PER_BYTE;
198     pixel_mask = (1 << (y - (page * 8)));
199
200     return gfx_mono_get_byte(page, x) & pixel_mask;
201 }
```

6.5.3.5 void gfx_mono_ssd1306_init (void)

Initialize SSD1306 controller and LCD display. It will also write the graphic controller RAM to all zeroes.

Note

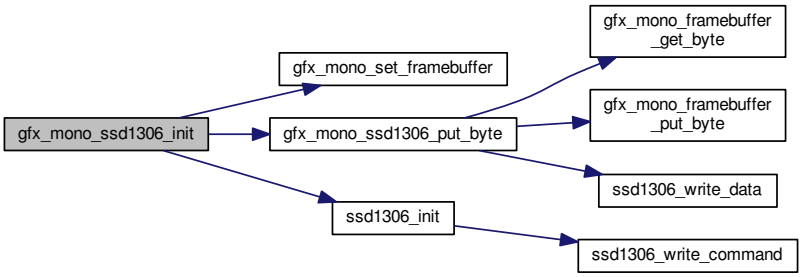
This function will clear the contents of the display.

Definition at line 62 of file gfx_mono_ug_2832hsweg04.c.

References framebuffer, GFX_MONO_LCD_FRAMEBUFFER_SIZE, GFX_MONO_LCD_PAGES, GFX_MONO_↩ LCD_WIDTH, gfx_mono_set_framebuffer(), gfx_mono_ssd1306_put_byte(), and ssd1306_init().

```
62                                  {
63     uint8_t page;
64     uint8_t column;
65
66 #ifdef CONFIG_SSD1306_FRAMEBUFFER
67     uint32_t ix;
68     for (ix = 0; ix < GFX_MONO_LCD_FRAMEBUFFER_SIZE; ix++) {
69         framebuffer[ix] = 0x00;
70     }
71
72     gfx_mono_set_framebuffer(framebuffer);
73 #endif
74
75     /* Initialize the low-level display controller. */
76     ssd1306_init();
77
78     /* Set display to output data from line 0 */
79     ssd1306_set_display_start_line_address(0);
80
81     /* Clear the contents of the display.
82      * If using a framebuffer (SPI interface) it will both clear the
83      * controller memory and the framebuffer.
84      */
85     for (page = 0; page < GFX_MONO_LCD_PAGES; page++) {
86         for (column = 0; column < GFX_MONO_LCD_WIDTH; column++) {
87             gfx_mono_ssd1306_put_byte(page, column, 0x00, true);
88         }
89     }
90 }
```

Here is the call graph for this function:



6.5.3.6 void gfx_mono_ssd1306_mask_byte (gfx_coord_t *page*, gfx_coord_t *column*, gfx_mono_color_t *pixel_mask*, gfx_mono_color_t *color*)

Read/Modify/Write a byte on the display controller.

This function will read the byte from the display controller (or the framebuffer if we cannot read directly from the controller) and do a mask operation on the byte according to the pixel operation selected by the color argument and the pixel mask provided.

Parameters

in	*page*	Page address
in	*column*	Page offset (x coordinate)
in	*pixel_mask*	Mask for pixel operation
in	*color*	Pixel operation

A small example that will XOR the first byte of display memory with 0xAA

```
1 gfx_mono_ssd1306_mask_byte(0,0,0xAA,GFX_PIXEL_XOR);
```

Definition at line 349 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_get_byte, gfx_mono_put_byte, GFX_PIXEL_CLR, GFX_PIXEL_SET, and GFX_PIXEL_X↩ OR.

```
350                                                                 {
351     gfx_mono_color_t temp = gfx_mono_get_byte(page, column);
352
353     switch (color) {
354         case GFX_PIXEL_SET:
355             temp |= pixel_mask;
356             break;
357
358         case GFX_PIXEL_CLR:
359             temp &= ~pixel_mask;
360             break;
361
362         case GFX_PIXEL_XOR:
363             temp ^= pixel_mask;
364             break;
365
366         default:
367             break;
368     }
369
370     gfx_mono_put_byte(page, column, temp);
371 }
```

6.5.3.7 void gfx_mono_ssd1306_put_byte (gfx_coord_t *page*, gfx_coord_t *column*, uint8_t *data*, bool *force*)

Put a byte to the display controller RAM.

If the LCD controller is accessed by the SPI interface we will also put the data to the local framebuffer.

Parameters

in	*page*	Page address
in	*column*	Page offset (x coordinate)
in	*data*	Data to be written
in	*force*	Forces the write

This example will put the value 0xFF to the first byte in the display memory setting a 8 pixel high column of pixels in the upper left corner of the display.

```
1 gfx_mono_ssd1306_put_byte(0, 0, 0xFF, false);
```

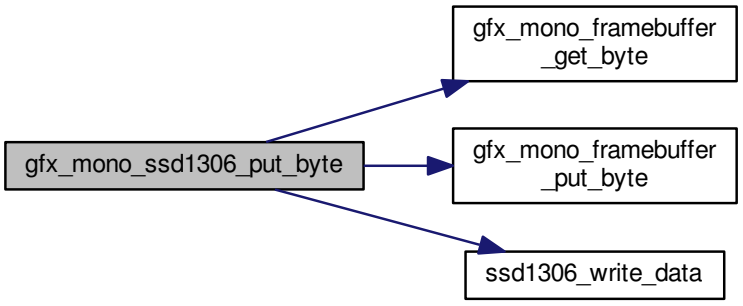
Definition at line 286 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_framebuffer_get_byte(), gfx_mono_framebuffer_put_byte(), and ssd1306_write_data().

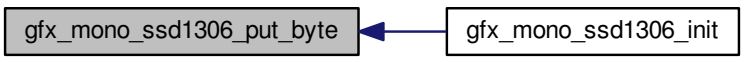
Referenced by gfx_mono_ssd1306_init().

```
287                                 {
288 #ifdef CONFIG_SSD1306_FRAMEBUFFER
289     if (!force && data == gfx_mono_framebuffer_get_byte(page, column)) {
290         return;
291     }
292     gfx_mono_framebuffer_put_byte(page, column,
    data);
293 #endif
294
295     ssd1306_set_page_address(page);
296     ssd1306_set_column_address(column);
297
298     ssd1306_write_data(data);
299 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.8 void gfx_mono_ssd1306_put_framebuffer (void)

Put framebuffer to LCD controller.

This function will output the complete framebuffer from RAM to the LCD controller.

Note

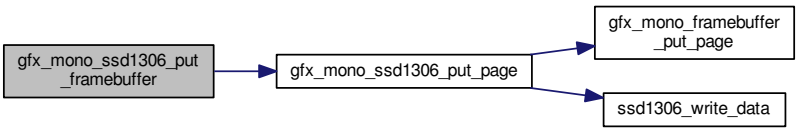
This is done automatically if using the graphic primitives. Only needed if you are manipulating the framebuffer directly in your code.

Definition at line 103 of file gfx_mono_ug_2832hsweg04.c.

References framebuffer, GFX_MONO_LCD_PAGES, GFX_MONO_LCD_WIDTH, and gfx_mono_ssd1306_put_↩page().

```
103                                             {
104     uint8_t page;
105
106     for (page = 0; page < GFX_MONO_LCD_PAGES; page++) {
107         ssd1306_set_page_address(page);
108         ssd1306_set_column_address(0);
109         gfx_mono_ssd1306_put_page(framebuffer
110                 + (page * GFX_MONO_LCD_WIDTH), page, 0,
111                 GFX_MONO_LCD_WIDTH);
112     }
113 }
```

Here is the call graph for this function:



6.5.3.9 void gfx_mono_ssd1306_put_page (gfx_mono_color_t * *data*, gfx_coord_t *page*, gfx_coord_t *column*, gfx_coord_t *width*)

Put a page from RAM to display controller.

If the controller is accessed by the SPI interface, we can not read back data from the LCD controller RAM. Because of this all data that is written to the LCD controller in this mode is also written to a framebuffer in MCU RAM.

Parameters

in	*data*	Pointer to data to be written
in	*page*	Page address
in	*column*	Offset into page (x coordinate)
in	*width*	Number of bytes to be written.

The following example will write 32 bytes from data_buf to the page 0, column 10. This will place data_buf in the rectangle x1=10,y1=0,x2=42,y2=8 (10 pixels from the upper left corner of the screen):

```
1 gfx_mono_ssd1306_put_page(data_buf, 0, 10, 32);
```

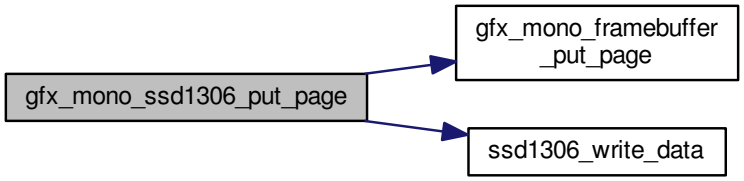
Definition at line 223 of file gfx_mono_ug_2832hsweg04.c.

References gfx_mono_framebuffer_put_page(), and ssd1306_write_data().

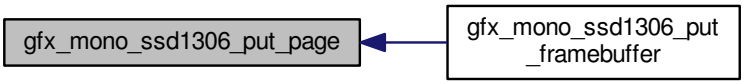
Referenced by gfx_mono_ssd1306_put_framebuffer().

```
224                                                 {
225 #ifdef CONFIG_SSD1306_FRAMEBUFFER
226     gfx_mono_framebuffer_put_page(data, page, column,
      width);
227 #endif
228     ssd1306_set_page_address(page);
229     ssd1306_set_column_address(column);
230
231     do {
232         ssd1306_write_data(*data++);
233     } while (--width);
234 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6 SSD1306 OLED Controller Low-level driver

Variables

- struct spi_module ssd1306_master
- struct spi_slave_inst ssd1306_slave

Fundamental Command defines

- #define SSD1306_CMD_COL_ADD_SET_LSB(column) (0x00 | (column))
- #define SSD1306_CMD_COL_ADD_SET_MSB(column) (0x10 | (column))
- #define SSD1306_CMD_SET_MEMORY_ADDRESSING_MODE 0x20
- #define SSD1306_CMD_SET_COLUMN_ADDRESS 0x21
- #define SSD1306_CMD_SET_PAGE_ADDRESS 0x22
- #define SSD1306_CMD_SET_DISPLAY_START_LINE(line) (0x40 | (line))
- #define SSD1306_CMD_SET_CONTRAST_CONTROL_FOR_BANK0 0x81
- #define SSD1306_CMD_SET_CHARGE_PUMP_SETTING 0x8D
- #define SSD1306_CMD_SET_SEGMENT_RE_MAP_COL0_SEG0 0xA0
- #define SSD1306_CMD_SET_SEGMENT_RE_MAP_COL127_SEG0 0xA1
- #define SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRAM_ON 0xA4
- #define SSD1306_CMD_ENTIRE_DISPLAY_ON 0xA5
- #define SSD1306_CMD_SET_NORMAL_DISPLAY 0xA6
- #define SSD1306_CMD_SET_INVERSE_DISPLAY 0xA7
- #define SSD1306_CMD_SET_MULTIPLEX_RATIO 0xA8
- #define SSD1306_CMD_SET_DISPLAY_ON 0xAF
- #define SSD1306_CMD_SET_DISPLAY_OFF 0xAE
- #define SSD1306_CMD_SET_PAGE_START_ADDRESS(page) (0xB0 | (page))
- #define SSD1306_CMD_SET_COM_OUTPUT_SCAN_UP 0xC0
- #define SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN 0xC8
- #define SSD1306_CMD_SET_DISPLAY_OFFSET 0xD3
- #define SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_RATIO 0xD5
- #define SSD1306_CMD_SET_PRE_CHARGE_PERIOD 0xD9
- #define SSD1306_CMD_SET_COM_PINS 0xDA
- #define SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL 0xDB
- #define SSD1306_CMD_NOP 0xE3

Graphic Acceleration Command defines

- #define SSD1306_CMD_SCROLL_H_RIGHT 0x26
- #define SSD1306_CMD_SCROLL_H_LEFT 0x27
- #define SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_RIGHT 0x29
- #define SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_LEFT 0x2A
- #define SSD1306_CMD_DEACTIVATE_SCROLL 0x2E
- #define SSD1306_CMD_ACTIVATE_SCROLL 0x2F
- #define SSD1306_CMD_SET_VERTICAL_SCROLL_AREA 0xA3

OLED controller write and read functions

- void ssd1306_write_command (uint8_t command)

  Writes a command to the display controller.
- void ssd1306_write_data (uint8_t data)

  Write data to the display controller.

Initialization

- void ssd1306_init (void)

  Initialize the OLED controller.

6.6.1 Detailed Description

This is a low level driver for the SSD1306 OLED controller through 4-wire SPI. It provides basic functions for initializing and writing to the OLED controller. In addition to hardware control and use of the OLED controller internal functions.

Before writing data to the display call ssd1306_init() which will set up the physical interface and the OLED. A file named conf_ssd1306.h is needed to define which interface to use. For more information see the Interface selection section. In addition one also need to define the pins SSD1306_DC_PIN, SSD1306_CS_PIN and SSD1306_RES↩_PIN and the display SSD1306_CLOCK_SPEED.

Warning

This driver is not reentrant and can not be used in interrupt\ service routines without extra care.

An example conf_ssd1306.h file could look like

```
// interface selection
#define SSD1306_SPI           SERCOM2

#define SSD1306_CLOCK_SPEED          1000000

#define SSD1306_DC_PIN               PIN_PB24
#define SSD1306_CS_PIN               PIN_PB27
#define SSD1306_RES_PIN              PIN_PA17
```

6.6.2 Dependencies

This driver depends on the following modules:

- asfdoc_sam0_port_group for IO port control.
- asfdoc_sam0_system_group for getting system clock speeds for init functions.
- asfdoc_sam0_sercom_spi_group for communication with the OLED controller
- asfdoc_sam0_sercom_spi_group for communication with the OLED controller

6.6.3 Macro Definition Documentation

6.6.3.1 #define SSD1306_CMD_ACTIVATE_SCROLL 0x2F

Definition at line 134 of file gfx_ssd1306.h.

6.6.3.2 #define SSD1306_CMD_COL_ADD_SET_LSB(*column*) (0x00 | (column))

Definition at line 100 of file gfx_ssd1306.h.

6.6.3.3 #define SSD1306_CMD_COL_ADD_SET_MSB(*column*) (0x10 | (column))

Definition at line 101 of file gfx_ssd1306.h.

6.6.3.4 #define SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_LEFT 0x2A

Definition at line 132 of file gfx_ssd1306.h.

6.6.3.5 #define SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_RIGHT 0x29

Definition at line 131 of file gfx_ssd1306.h.

6.6.3.6 #define SSD1306_CMD_DEACTIVATE_SCROLL 0x2E

Definition at line 133 of file gfx_ssd1306.h.

6.6.3.7 #define SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRAM_ON 0xA4

Definition at line 110 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.8 #define SSD1306_CMD_ENTIRE_DISPLAY_ON 0xA5

Definition at line 111 of file gfx_ssd1306.h.

6.6.3.9 #define SSD1306_CMD_NOP 0xE3

Definition at line 125 of file gfx_ssd1306.h.

6.6.3.10 #define SSD1306_CMD_SCROLL_H_LEFT 0x27

Definition at line 130 of file gfx_ssd1306.h.

6.6.3.11 #define SSD1306_CMD_SCROLL_H_RIGHT 0x26

Definition at line 129 of file gfx_ssd1306.h.

6.6.3.12 #define SSD1306_CMD_SET_CHARGE_PUMP_SETTING 0x8D

Definition at line 107 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.13 #define SSD1306_CMD_SET_COLUMN_ADDRESS 0x21

Definition at line 103 of file gfx_ssd1306.h.

6.6.3.14 #define SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN 0xC8

Definition at line 119 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.15 #define SSD1306_CMD_SET_COM_OUTPUT_SCAN_UP 0xC0

Definition at line 118 of file gfx_ssd1306.h.

6.6.3.16 #define SSD1306_CMD_SET_COM_PINS 0xDA

Definition at line 123 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.17 #define SSD1306_CMD_SET_CONTRAST_CONTROL_FOR_BANK0 0x81

Definition at line 106 of file gfx_ssd1306.h.

6.6.3.18 #define SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_RATIO 0xD5

Definition at line 121 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.19 #define SSD1306_CMD_SET_DISPLAY_OFF 0xAE

Definition at line 116 of file gfx_ssd1306.h.

6.6.3.20 #define SSD1306_CMD_SET_DISPLAY_OFFSET 0xD3

Definition at line 120 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.21 #define SSD1306_CMD_SET_DISPLAY_ON 0xAF

Definition at line 115 of file gfx_ssd1306.h.

6.6.3.22 #define SSD1306_CMD_SET_DISPLAY_START_LINE(*line*) (0x40 | (line))

Definition at line 105 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.23 #define SSD1306_CMD_SET_INVERSE_DISPLAY 0xA7

Definition at line 113 of file gfx_ssd1306.h.

6.6.3.24 #define SSD1306_CMD_SET_MEMORY_ADDRESSING_MODE 0x20

Definition at line 102 of file gfx_ssd1306.h.

6.6.3.25 #define SSD1306_CMD_SET_MULTIPLEX_RATIO 0xA8

Definition at line 114 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.26 #define SSD1306_CMD_SET_NORMAL_DISPLAY 0xA6

Definition at line 112 of file gfx_ssd1306.h.

6.6.3.27 #define SSD1306_CMD_SET_PAGE_ADDRESS 0x22

Definition at line 104 of file gfx_ssd1306.h.

6.6.3.28 #define SSD1306_CMD_SET_PAGE_START_ADDRESS(*page*) (0xB0 | (page))

Definition at line 117 of file gfx_ssd1306.h.

6.6.3.29 #define SSD1306_CMD_SET_PRE_CHARGE_PERIOD 0xD9

Definition at line 122 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.30 #define SSD1306_CMD_SET_SEGMENT_RE_MAP_COL0_SEG0 0xA0

Definition at line 108 of file gfx_ssd1306.h.

6.6.3.31 #define SSD1306_CMD_SET_SEGMENT_RE_MAP_COL127_SEG0 0xA1

Definition at line 109 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.32 #define SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL 0xDB

Definition at line 124 of file gfx_ssd1306.h.

Referenced by ssd1306_init().

6.6.3.33 #define SSD1306_CMD_SET_VERTICAL_SCROLL_AREA 0xA3

Definition at line 135 of file gfx_ssd1306.h.

6.6.4 Function Documentation

6.6.4.1 void ssd1306_init (void)

Initialize the OLED controller.

Call this function to initialize the hardware interface and the OLED controller. When initialization is done the display is turned on and ready to receive data.

Definition at line 57 of file gfx_ssd1306.c.

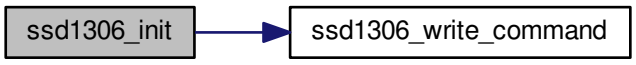
References GFX_DISPLAY_RESET_SET, gfx_mono_draw_filled_rect, GFX_PIXEL_CLR, GFX_PIXEL_SET, SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRAM_ON, SSD1306_CMD_SET_CHARGE_PUMP_SETTING, SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN, SSD1306_CMD_SET_COM_PINS, SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_RATIO, SSD1306_CMD_SET_DISPLAY_OFFSET, SSD1306_CMD_SET_DISPLAY_START_LINE, SSD1306_CMD_SET_MULTIPLEX_RATIO, SSD1306_CMD_SET_PRE_CHARGE_PERIOD, SSD1306_CMD_SET_SEGMENT_RE_MAP_COL127_SEG0, SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL, and ssd1306_write_command().

Referenced by gfx_mono_ssd1306_init().

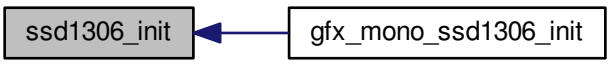
```
57                          {
58     // Initialize the interface
59     //  ssd1306_interface_init();    // --> already set by H3
60
61     // Do a hard reset of the OLED display controller
62     ssd1306_hard_reset();
63
64     // Set the reset pin to the default state
65     GFX_DISPLAY_RESET_SET();
66
67     // 1/32 Duty (0x0F~0x3F)
68     ssd1306_write_command(SSD1306_CMD_SET_MULTIPLEX_RATIO
       );
69     ssd1306_write_command(0x1F);
70
71     // Shift Mapping RAM Counter (0x00~0x3F)
72     ssd1306_write_command(SSD1306_CMD_SET_DISPLAY_OFFSET
       );
73     ssd1306_write_command(0x00);
74
75     // Set Mapping RAM Display Start Line (0x00~0x3F)
76     ssd1306_write_command(
```

```
      SSD1306_CMD_SET_DISPLAY_START_LINE(0x00));
77
78     // Set Column Address 0 Mapped to SEG0
79     ssd1306_write_command(
      SSD1306_CMD_SET_SEGMENT_RE_MAP_COL127_SEG0);
80
81     // Set COM/Row Scan Scan from COM63 to 0
82     ssd1306_write_command(
      SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN);
83
84     // Set COM Pins hardware configuration
85     ssd1306_write_command(SSD1306_CMD_SET_COM_PINS);
86     ssd1306_write_command(0x02);
87
88     ssd1306_set_contrast(0x8F);
89
90     // Disable Entire display On
91     ssd1306_write_command(
      SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRAM_ON);
92
93     ssd1306_display_invert_disable();
94
95     // Set Display Clock Divide Ratio / Oscillator Frequency (Default => 0x80)
96     ssd1306_write_command(
      SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_RATIO);
97     ssd1306_write_command(0x80);
98
99     // Enable charge pump regulator
100     ssd1306_write_command(
      SSD1306_CMD_SET_CHARGE_PUMP_SETTING);
101     ssd1306_write_command(0x14);
102
103     // Set VCOMH Deselect Level
104     ssd1306_write_command(
      SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL);
105     ssd1306_write_command(0x40); // Default => 0x20 (0.77*VCC)
106
107     // Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
108     ssd1306_write_command(SSD1306_CMD_SET_PRE_CHARGE_PERIOD
      );
109     ssd1306_write_command(0xF1);
110
111     ssd1306_display_on();
112
113     gfx_mono_draw_filled_rect(0, 0, 128, 32,
      GFX_PIXEL_SET);
114     gfx_mono_draw_filled_rect(0, 0, 128, 32,
      GFX_PIXEL_CLR);
115 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.4.2 void ssd1306_write_command (uint8_t *command*)

Writes a command to the display controller.

This functions pull pin D/C# low before writing to the controller. Different data write function is called based on the selected interface.

Parameters

command	the command to write

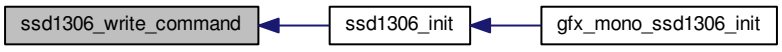
Definition at line 125 of file gfx_ssd1306.c.

References GFX_DATA_CMD_SEL_CLEAR, GFX_DISPLAY_SS_N_CLEAR, GFX_DISPLAY_SS_N_SET, GFX_SPI_IS_BUSY, and GFX_SPI_WRITE_FUNCTION.

Referenced by ssd1306_init().

```
125                                             {
126     GFX_DISPLAY_SS_N_CLEAR();
127     GFX_DATA_CMD_SEL_CLEAR();
128     GFX_SPI_WRITE_FUNCTION(&command, 1);
129     while (GFX_SPI_IS_BUSY());
130     GFX_DISPLAY_SS_N_SET();
131 }
```

Here is the caller graph for this function:



6.6.4.3 void ssd1306_write_data (uint8_t *data*)

Write data to the display controller.

This functions sets the pin D/C# before writing to the controller. Different data write function is called based on the selected interface.

Parameters

data	the data to write

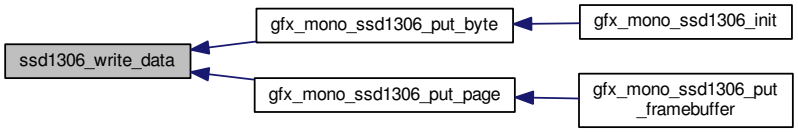
Definition at line 141 of file gfx_ssd1306.c.

References GFX_DATA_CMD_SEL_SET, GFX_DISPLAY_SS_N_CLEAR, GFX_DISPLAY_SS_N_SET, GFX_SPI_IS_BUSY, and GFX_SPI_WRITE_FUNCTION.

Referenced by gfx_mono_ssd1306_put_byte(), and gfx_mono_ssd1306_put_page().

```
141                                     {
142     GFX_DISPLAY_SS_N_CLEAR();
143     GFX_DATA_CMD_SEL_SET();
144     GFX_SPI_WRITE_FUNCTION(&data, 1);
145     while (GFX_SPI_IS_BUSY());
146     GFX_DISPLAY_SS_N_SET();
147 }
```

Here is the caller graph for this function:



6.6.5 Variable Documentation

6.6.5.1 struct spi_module ssd1306_master

6.6.5.2 struct spi_slave_inst ssd1306_slave

6.7 Gfx_sysfont

Macros

- #define USE_FONT_BPMONO_10x16
- #define SYSFONT_WIDTH 10
- #define SYSFONT_HEIGHT 16
- #define SYSFONT_LINESPACING 8
- #define SYSFONT_FIRSTCHAR ((uint8_t)' ')
- #define SYSFONT_LASTCHAR ((uint8_t)'}')
- #define SYSFONT_DEFINE_GLYPHS

6.7.1 Detailed Description

6.7.2 Macro Definition Documentation

6.7.2.1 #define SYSFONT_DEFINE_GLYPHS

Define variable containing the font

Definition at line 71 of file gfx_sysfont.h.

6.7.2.2 #define SYSFONT_FIRSTCHAR ((uint8_t)' ')

First character defined.

Definition at line 66 of file gfx_sysfont.h.

6.7.2.3 #define SYSFONT_HEIGHT 16

Height of each glyph, excluding spacer line.

Definition at line 62 of file gfx_sysfont.h.

6.7.2.4 #define SYSFONT_LASTCHAR ((uint8_t)'}')

Last character defined.

Definition at line 68 of file gfx_sysfont.h.

6.7.2.5 #define SYSFONT_LINESPACING 8

Line height.

Definition at line 64 of file gfx_sysfont.h.

6.7.2.6 #define SYSFONT_WIDTH 10

Width of each glyph, including spacer column.

Definition at line 60 of file gfx_sysfont.h.

6.7.2.7 #define USE_FONT_BPMONO_10x16

Definition at line 53 of file gfx_sysfont.h.

Chapter 7

Class Documentation

7.1 font Struct Reference

`#include <gfx_mono_text.h>`

Public Attributes

- enum font_data_type type
- union {
  uint8_t PROGMEM_PTR_T progmem
  } data

- uint8_t width
- uint8_t height
- uint8_t first_char
- uint8_t last_char

7.1.1 Detailed Description

Storage structure for font meta data.

Definition at line 77 of file gfx_mono_text.h.

7.1.2 Member Data Documentation

7.1.2.1 union { ... } font::data

7.1.2.2 uint8_t font::first_char

ASCII value of first character in font set.

Definition at line 97 of file gfx_mono_text.h.

7.1.2.3 uint8_t font::height

Height of one font character, in pixels.

Definition at line 95 of file gfx_mono_text.h.

Referenced by gfx_mono_draw_char(), gfx_mono_draw_progmem_string(), gfx_mono_draw_string(), gfx_mono↩_get_progmem_string_bounding_box(), and gfx_mono_get_string_bounding_box().

7.1.2.4 uint8_t font::last_char

ASCII value of last character in the set.

Definition at line 99 of file gfx_mono_text.h.

7.1.2.5 uint8_t PROGMEM_PTR_T font::progmem

Pointer to where the binary font data is stored. This variable is accessed either through hugemem or progmem depending on the value of *type*.

Definition at line 90 of file gfx_mono_text.h.

7.1.2.6 enum font_data_type font::type

Type of storage used for binary font data. See font_data_type.

Definition at line 79 of file gfx_mono_text.h.

Referenced by gfx_mono_draw_char().

7.1.2.7 uint8_t font::width

Width of one font character, in pixels.

Definition at line 93 of file gfx_mono_text.h.

Referenced by gfx_mono_draw_char(), gfx_mono_draw_progmem_string(), gfx_mono_draw_string(), gfx_mono↩_get_progmem_string_bounding_box(), and gfx_mono_get_string_bounding_box().

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

- /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.h

7.2 gfx_mono_bitmap Struct Reference

Storage structure for bitmap pixel data and metadata.

`#include <gfx_mono_generic.h>`

Public Attributes

- gfx_coord_t width
- gfx_coord_t height
- enum gfx_mono_bitmap_type type
- union {
  gfx_mono_color_t * pixmap
  gfx_mono_color_t PROGMEM_T * progmem
  } data

7.2.1 Detailed Description

Storage structure for bitmap pixel data and metadata.

Definition at line 80 of file gfx_mono_generic.h.

7.2.2 Member Data Documentation

7.2.2.1 union { ... } gfx_mono_bitmap::data

Referenced by gfx_mono_generic_put_bitmap().

7.2.2.2 gfx_coord_t gfx_mono_bitmap::height

Height of bitmap

Definition at line 84 of file gfx_mono_generic.h.

Referenced by gfx_mono_generic_put_bitmap().

7.2.2.3 gfx_mono_color_t* gfx_mono_bitmap::pixmap

Pointer to pixels for bitmap stored in RAM

Definition at line 90 of file gfx_mono_generic.h.

Referenced by gfx_mono_generic_put_bitmap().

7.2.2.4 gfx_mono_color_t PROGMEM_T* gfx_mono_bitmap::progmem

Pointer to pixels for bitmap stored in progmem

Definition at line 92 of file gfx_mono_generic.h.

Referenced by gfx_mono_generic_put_bitmap().

7.2.2.5 enum gfx_mono_bitmap_type gfx_mono_bitmap::type

Bitmap type

Definition at line 86 of file gfx_mono_generic.h.

Referenced by gfx_mono_generic_put_bitmap().

7.2.2.6 gfx_coord_t gfx_mono_bitmap::width

Width of bitmap

Definition at line 82 of file gfx_mono_generic.h.

Referenced by gfx_mono_generic_put_bitmap().

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

- /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.h

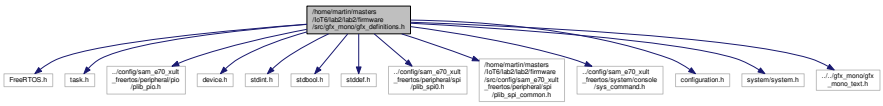
Chapter 8

File Documentation

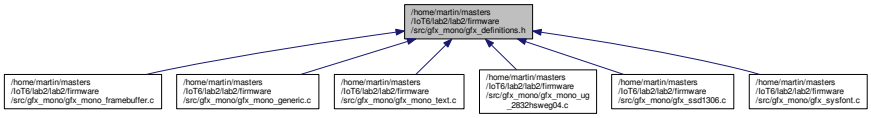
8.1 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_definitions.h File Reference

```
#include "FreeRTOS.h"
#include "task.h"
#include "../config/sam_e70_xult_freertos/peripheral/pio/plib_pio.h"
#include "../config/sam_e70_xult_freertos/peripheral/spi/plib_spi0.h"
#include "../config/sam_e70_xult_freertos/system/console/sys_command.h"
#include "../../gfx_mono/gfx_mono_text.h"
```

Include dependency graph for gfx_definitions.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define GFX_DELAY_FUNCTION(x) vTaskDelay(pdMS_TO_TICKS(x))
- #define GFX_SPI_WRITE_FUNCTION(x, y) SPI0_Write(x, y)
- #define GFX_SPI_IS_BUSY() SPI0_IsBusy()
- #define GFX_DISPLAY_RESET_CLEAR() DISPLAY_RESET_Clear()
- #define GFX_DISPLAY_RESET_SET() DISPLAY_RESET_Set()
- #define GFX_DISPLAY_SS_N_CLEAR() DISPLAY_SS_N_Clear()
- #define GFX_DISPLAY_SS_N_SET() DISPLAY_SS_N_Set()
- #define GFX_DATA_CMD_SEL_CLEAR() DATA_CMD_SEL_Clear()
- #define GFX_DATA_CMD_SEL_SET() DATA_CMD_SEL_Set()
- #define PRINTF_BLOCKING(fmt, ...)
- #define GFX_MONO_UG_2832HSWEG04

8.1.1 Macro Definition Documentation

8.1.1.1 #define GFX_DATA_CMD_SEL_CLEAR( ) DATA_CMD_SEL_Clear()

Definition at line 29 of file gfx_definitions.h.

Referenced by ssd1306_write_command().

8.1.1.2 #define GFX_DATA_CMD_SEL_SET( ) DATA_CMD_SEL_Set()

Definition at line 30 of file gfx_definitions.h.

Referenced by ssd1306_write_data().

8.1.1.3 #define GFX_DELAY_FUNCTION(*x*) vTaskDelay(pdMS_TO_TICKS(x))

Descriptive File Name

Microchip global definitions for gfx mono library

Definition at line 22 of file gfx_definitions.h.

8.1.1.4 #define GFX_DISPLAY_RESET_CLEAR( ) DISPLAY_RESET_Clear()

Definition at line 25 of file gfx_definitions.h.

8.1.1.5 #define GFX_DISPLAY_RESET_SET( ) DISPLAY_RESET_Set()

Definition at line 26 of file gfx_definitions.h.

Referenced by ssd1306_init().

8.1.1.6 #define GFX_DISPLAY_SS_N_CLEAR( ) DISPLAY_SS_N_Clear()

Definition at line 27 of file gfx_definitions.h.

Referenced by ssd1306_write_command(), and ssd1306_write_data().

8.1.1.7 #define GFX_DISPLAY_SS_N_SET( ) DISPLAY_SS_N_Set()

Definition at line 28 of file gfx_definitions.h.

Referenced by ssd1306_write_command(), and ssd1306_write_data().

8.1.1.8 #define GFX_MONO_UG_2832HSWEG04

Definition at line 40 of file gfx_definitions.h.

8.1.1.9 #define GFX_SPI_IS_BUSY( ) SPI0_IsBusy()

Definition at line 24 of file gfx_definitions.h.

Referenced by ssd1306_write_command(), and ssd1306_write_data().

8.1.1.10 #define GFX_SPI_WRITE_FUNCTION(*x, y*) SPI0_Write(x, y)

Definition at line 23 of file gfx_definitions.h.

Referenced by ssd1306_write_command(), and ssd1306_write_data().

8.1.1.11 #define PRINTF_BLOCKING(*fmt, ...*)

Value:

```
while(SYS_CMD_READY_TO_WRITE()); \
    SYS_CMD_PRINT(fmt, ##__VA_ARGS__);
```

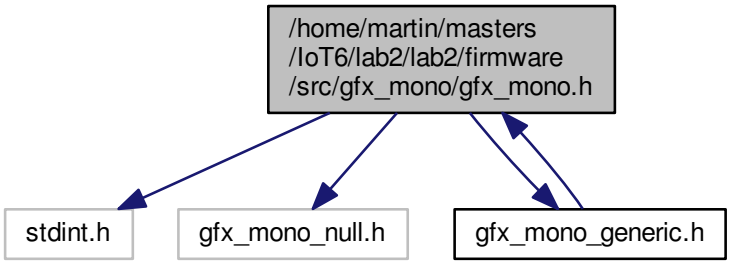
Definition at line 36 of file gfx_definitions.h.

8.2 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono.h File Reference

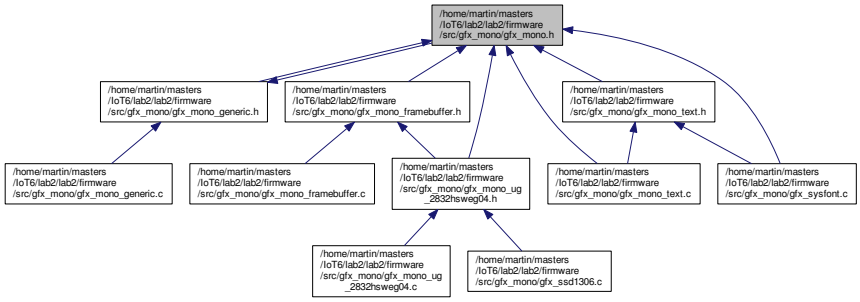
Monochrome graphic library API header file.

```
#include <stdint.h>
#include "gfx_mono_null.h"
#include "gfx_mono_generic.h"
```

Include dependency graph for gfx_mono.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define PROGMEM_DECLARE(type, name) const type name
- #define PROGMEM_T const
- #define PROGMEM_PTR_T const *
- #define PROGMEM_READ_BYTE(x) *(x)
- #define PROGMEM_STRING_T const char*

Circle Sector Definitions

- #define GFX_OCTANT0 (1 << 0)
- #define GFX_OCTANT1 (1 << 1)
- #define GFX_OCTANT2 (1 << 2)
- #define GFX_OCTANT3 (1 << 3)
- #define GFX_OCTANT4 (1 << 4)
- #define GFX_OCTANT5 (1 << 5)
- #define GFX_OCTANT6 (1 << 6)
- #define GFX_OCTANT7 (1 << 7)
- #define GFX_QUADRANT0 (GFX_OCTANT0 | GFX_OCTANT1)
- #define GFX_QUADRANT1 (GFX_OCTANT2 | GFX_OCTANT3)
- #define GFX_QUADRANT2 (GFX_OCTANT4 | GFX_OCTANT5)
- #define GFX_QUADRANT3 (GFX_OCTANT6 | GFX_OCTANT7)
- #define GFX_LEFTHALF (GFX_QUADRANT3 | GFX_QUADRANT0)
- #define GFX_TOPHALF (GFX_QUADRANT0 | GFX_QUADRANT1)
- #define GFX_RIGHTHALF (GFX_QUADRANT1 | GFX_QUADRANT2)
- #define GFX_BOTTOMHALF (GFX_QUADRANT2 | GFX_QUADRANT3)
- #define GFX_WHOLE 0xFF

Typedefs

- typedef uint8_t gfx_mono_color_t
- typedef uint8_t gfx_coord_t

Enumerations

- enum gfx_mono_color { GFX_PIXEL_CLR = 0, GFX_PIXEL_SET = 1, GFX_PIXEL_XOR = 2 }
- enum gfx_mono_bitmap_type { GFX_MONO_BITMAP_RAM, GFX_MONO_BITMAP_PROGMEM }

8.2.1 Detailed Description

Monochrome graphic library API header file.

8.2.2 Macro Definition Documentation

8.2.2.1 #define PROGMEM_DECLARE(*type, name*) const type name

Definition at line 53 of file gfx_mono.h.

8.2.2.2 #define PROGMEM_PTR_T const *

Definition at line 55 of file gfx_mono.h.

Referenced by gfx_mono_draw_progmem_string(), and gfx_mono_get_progmem_string_bounding_box().

8.2.2.3 #define PROGMEM_READ_BYTE(*x*) *(x)

Definition at line 56 of file gfx_mono.h.

Referenced by gfx_mono_draw_progmem_string(), gfx_mono_generic_put_bitmap(), and gfx_mono_get_↩ progmem_string_bounding_box().

8.2.2.4 #define PROGMEM_STRING_T const char*

Definition at line 57 of file gfx_mono.h.

8.2.2.5 #define PROGMEM_T const

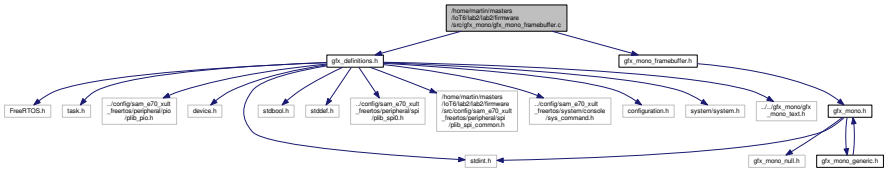
Definition at line 54 of file gfx_mono.h.

8.3 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_framebuffer.c File Reference

Local framebuffer.

```
#include "gfx_definitions.h"
#include "gfx_mono_framebuffer.h"
```

Include dependency graph for gfx_mono_framebuffer.c:



Functions

- void gfx_mono_set_framebuffer (uint8_t *framebuffer)
  Set the LCD framebuffer.
- void gfx_mono_framebuffer_put_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t column, gfx_coord_t width)
  Put a page from RAM to the framebuffer.
- void gfx_mono_framebuffer_get_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t column, gfx_coord_t width)
  Read a page from the framebuffer.
- void gfx_mono_framebuffer_draw_pixel (gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color)
  Draw pixel to framebuffer.
- uint8_t gfx_mono_framebuffer_get_pixel (gfx_coord_t x, gfx_coord_t y)
  Get the pixel value at x,y in framebuffer.
- void gfx_mono_framebuffer_put_byte (gfx_coord_t page, gfx_coord_t column, uint8_t data)
  Put a byte to the framebuffer.
- uint8_t gfx_mono_framebuffer_get_byte (gfx_coord_t page, gfx_coord_t column)
  Get a byte from the framebuffer.
- void gfx_mono_framebuffer_mask_byte (gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_↩ mask, gfx_mono_color_t color)
  Read/Modify/Write a byte in the framebuffer.

8.3.1 Detailed Description

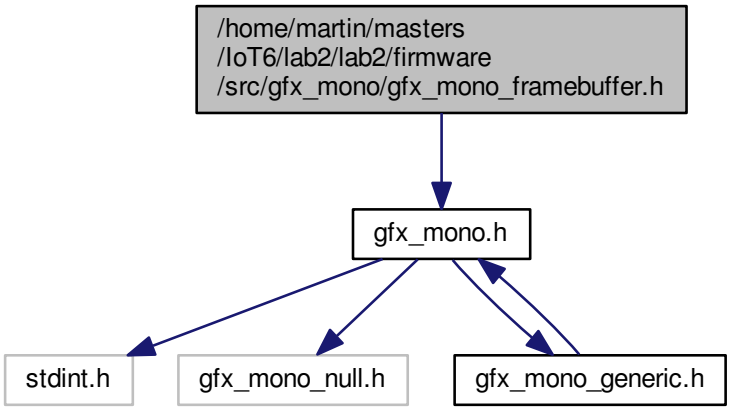
Local framebuffer.

8.4 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_framebuffer.h File Reference

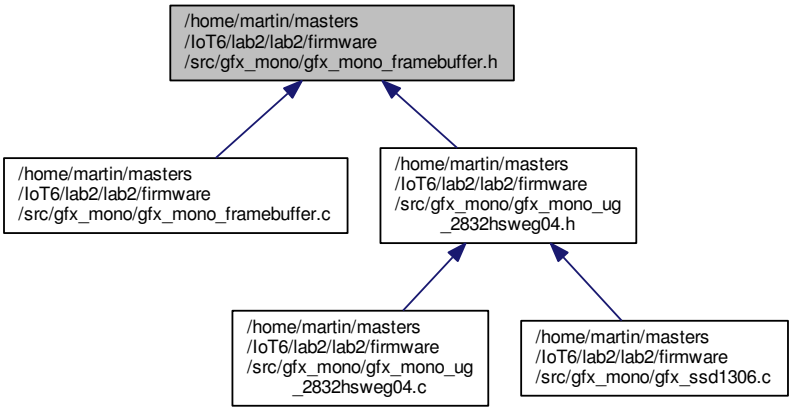
Monochrome graphic library framebuffer device.

```
#include "gfx_mono.h"
```

Include dependency graph for gfx_mono_framebuffer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void gfx_mono_set_framebuffer (uint8_t *framebuffer)

  Set the LCD framebuffer.
- void gfx_mono_framebuffer_put_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Put a page from RAM to the framebuffer.
- void gfx_mono_framebuffer_get_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Read a page from the framebuffer.
- void gfx_mono_framebuffer_draw_pixel (gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color)

  Draw pixel to framebuffer.
- uint8_t gfx_mono_framebuffer_get_pixel (gfx_coord_t x, gfx_coord_t y)

  Get the pixel value at x,y in framebuffer.
- void gfx_mono_framebuffer_put_byte (gfx_coord_t page, gfx_coord_t column, uint8_t data)

  Put a byte to the framebuffer.
- uint8_t gfx_mono_framebuffer_get_byte (gfx_coord_t page, gfx_coord_t column)

  Get a byte from the framebuffer.
- void gfx_mono_framebuffer_mask_byte (gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_↩ mask, gfx_mono_color_t color)

  Read/Modify/Write a byte in the framebuffer.

8.4.1 Detailed Description

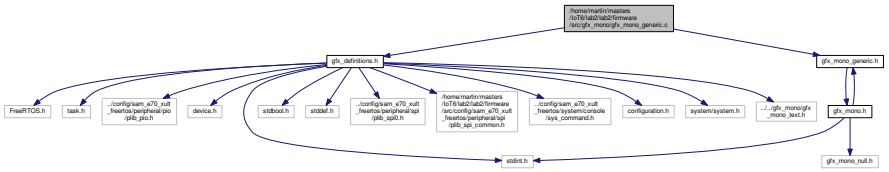
Monochrome graphic library framebuffer device.

8.5 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.c File Reference

Generic monochrome LCD graphic primitives.

```
#include "gfx_definitions.h"
#include "gfx_mono_generic.h"
```

Include dependency graph for gfx_mono_generic.c:



Functions

- void gfx_mono_generic_draw_horizontal_line (gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx↩_mono_color color)

  Draw a horizontal line, one pixel wide (generic implementation)
- void gfx_mono_generic_draw_vertical_line (gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_↩mono_color color)

  Draw a vertical line, one pixel wide (generic implementation)
- void gfx_mono_generic_draw_line (gfx_coord_t x1, gfx_coord_t y1, gfx_coord_t x2, gfx_coord_t y2, enum gfx_mono_color color)

  Draw a line between two arbitrary points (generic implementation).
- void gfx_mono_generic_draw_rect (gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color)

  Draw an outline of a rectangle (generic implementation).
- void gfx_mono_generic_draw_filled_rect (gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_↩t height, enum gfx_mono_color color)

  Draw a filled rectangle (generic implementation).
- void gfx_mono_generic_draw_circle (gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t octant_mask)

  Draw an outline of a circle or arc (generic implementation).
- void gfx_mono_generic_draw_filled_circle (gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_↩mono_color color, uint8_t quadrant_mask)

  Draw a filled circle or sector (generic implementation).
- void gfx_mono_generic_put_bitmap (struct gfx_mono_bitmap *bitmap, gfx_coord_t x, gfx_coord_t y)

  Put bitmap from FLASH or RAM to display.

8.5.1 Detailed Description

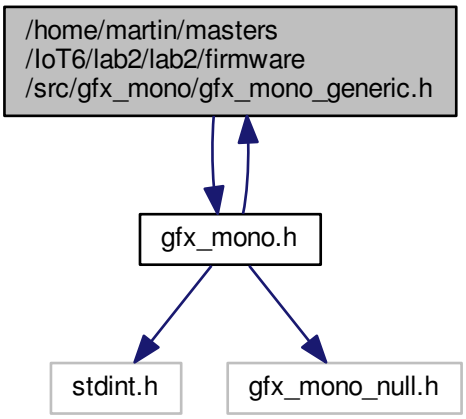
Generic monochrome LCD graphic primitives.

8.6 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_generic.h File Reference

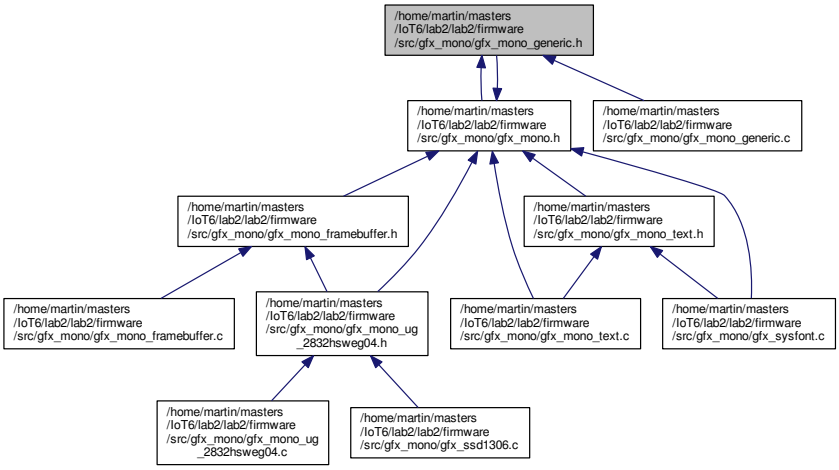
Generic monochrome LCD graphic primitives.

```
#include "gfx_mono.h"
```

Include dependency graph for gfx_mono_generic.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct gfx_mono_bitmap

  Storage structure for bitmap pixel data and metadata.

Functions

- void gfx_mono_generic_draw_vertical_line (gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_mono_color color)

  Draw a vertical line, one pixel wide (generic implementation)
- void gfx_mono_generic_draw_line (gfx_coord_t x1, gfx_coord_t y1, gfx_coord_t x2, gfx_coord_t y2, enum gfx_mono_color color)

  Draw a line between two arbitrary points (generic implementation).
- void gfx_mono_generic_draw_rect (gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color)

  Draw an outline of a rectangle (generic implementation).
- void gfx_mono_generic_draw_filled_rect (gfx_coord_t x, gfx_coord_t y, gfx_coord_t width, gfx_coord_t height, enum gfx_mono_color color)

  Draw a filled rectangle (generic implementation).
- void gfx_mono_generic_draw_circle (gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t octant_mask)

  Draw an outline of a circle or arc (generic implementation).
- void gfx_mono_generic_draw_filled_circle (gfx_coord_t x, gfx_coord_t y, gfx_coord_t radius, enum gfx_mono_color color, uint8_t quadrant_mask)

  Draw a filled circle or sector (generic implementation).
- void gfx_mono_generic_put_bitmap (struct gfx_mono_bitmap *bitmap, gfx_coord_t x, gfx_coord_t y)

  Put bitmap from FLASH or RAM to display.

- void gfx_mono_generic_draw_horizontal_line (gfx_coord_t x, gfx_coord_t y, gfx_coord_t length, enum gfx_mono_color color)

  Draw a horizontal line, one pixel wide (generic implementation)

8.6.1 Detailed Description

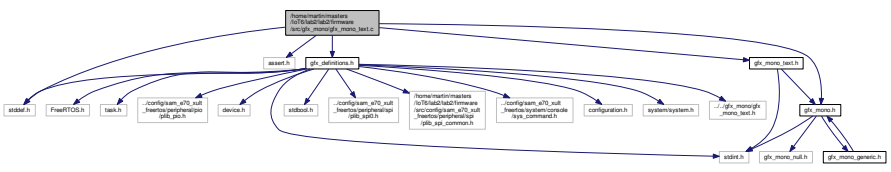
Generic monochrome LCD graphic primitives.

8.7 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.c File Reference

Font and text drawing routines.

```
#include "stddef.h"
#include "assert.h"
#include "gfx_definitions.h"
#include "gfx_mono.h"
#include "gfx_mono_text.h"
```

Include dependency graph for gfx_mono_text.c:



Macros

- #define CONFIG_FONT_PIXELS_PER_BYTE 8
- #define EXTMEM_BUF_SIZE 20

Functions

- void gfx_mono_draw_char (const char c, const gfx_coord_t x, const gfx_coord_t y, const struct font *font)

  Draws a character to the display.
- void gfx_mono_draw_string (const char *str, gfx_coord_t x, gfx_coord_t y, const struct font *font)

  Draws a string to the display.
- void gfx_mono_draw_progmem_string (char PROGMEM_PTR_T str, gfx_coord_t x, gfx_coord_t y, const struct font *font)

  Draws a string located in program memory to the display.
- void gfx_mono_get_string_bounding_box (const char *str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height)

  Computes the bounding box of a string.
- void gfx_mono_get_progmem_string_bounding_box (char PROGMEM_PTR_T str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height)

  Computes the bounding box of a string located in program memory.

8.7.1 Detailed Description

Font and text drawing routines.

8.7.2 Macro Definition Documentation

8.7.2.1 #define CONFIG_FONT_PIXELS_PER_BYTE 8

Definition at line 54 of file gfx_mono_text.c.

8.7.2.2 #define EXTMEM_BUF_SIZE 20

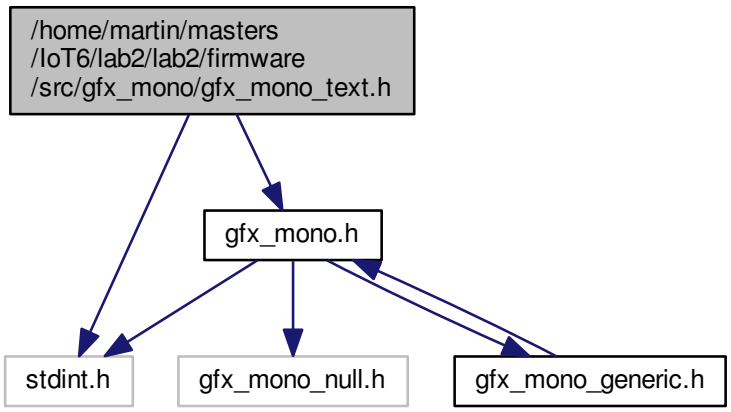
Definition at line 57 of file gfx_mono_text.c.

8.8 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_text.h File Reference

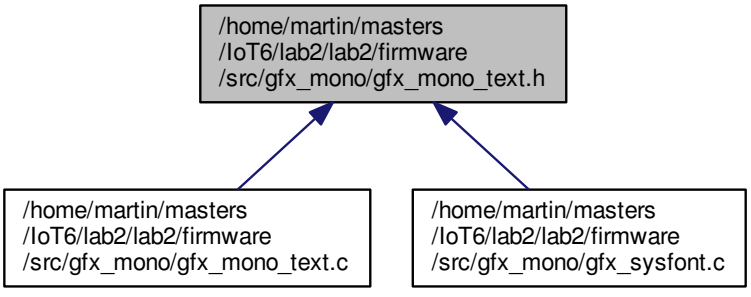
Monochrome graphic library API header file.

```
#include <stdint.h>
#include "gfx_mono.h"
```

Include dependency graph for gfx_mono_text.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct font

Enumerations

- enum font_data_type { FONT_LOC_PROGMEM }

  Valid storage locations for font data.

Functions

Strings and characters located in RAM

- void gfx_mono_draw_char (const char c, const gfx_coord_t x, const gfx_coord_t y, const struct font *font)

  Draws a character to the display.
- void gfx_mono_draw_string (const char *str, const gfx_coord_t x, const gfx_coord_t y, const struct font *font)

  Draws a string to the display.
- void gfx_mono_get_string_bounding_box (char const *str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height)

  Computes the bounding box of a string.

Strings located in flash

- void gfx_mono_draw_progmem_string (char PROGMEM_PTR_T str, gfx_coord_t x, gfx_coord_t y, const struct font *font)

  Draws a string located in program memory to the display.
- void gfx_mono_get_progmem_string_bounding_box (char PROGMEM_PTR_T str, const struct font *font, gfx_coord_t *width, gfx_coord_t *height)

  Computes the bounding box of a string located in program memory.

8.8.1 Detailed Description

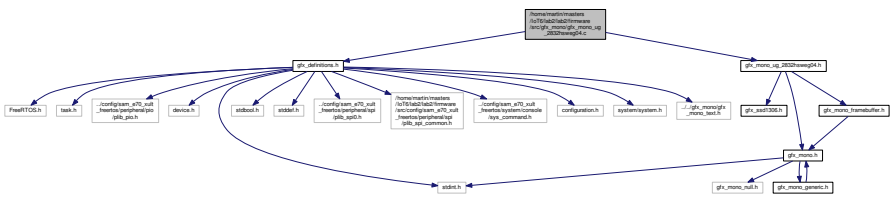
Monochrome graphic library API header file.

8.9 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_ug_2832hsweg04.c File Reference

Haven Display UG 2832HSWEG04 display glue code for display controller.

```
#include "gfx_definitions.h"
#include "gfx_mono_ug_2832hsweg04.h"
```

Include dependency graph for gfx_mono_ug_2832hsweg04.c:



Macros

- #define CONFIG_SSD1306_FRAMEBUFFER

Functions

- void gfx_mono_ssd1306_init (void)

  Initialize SSD1306 controller and LCD display. It will also write the graphic controller RAM to all zeroes.
- void gfx_mono_ssd1306_put_framebuffer (void)

  Put framebuffer to LCD controller.
- void gfx_mono_ssd1306_draw_pixel (gfx_coord_t x, gfx_coord_t y, gfx_coord_t color)

  Draw pixel to screen.
- uint8_t gfx_mono_ssd1306_get_pixel (gfx_coord_t x, gfx_coord_t y)

  Get the pixel value at x,y.
- void gfx_mono_ssd1306_put_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t column, gfx_coord_t width)

  Put a page from RAM to display controller.
- void gfx_mono_ssd1306_get_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t column, gfx_coord_t width)

  Read a page from the LCD controller.
- void gfx_mono_ssd1306_put_byte (gfx_coord_t page, gfx_coord_t column, uint8_t data, bool force)

  Put a byte to the display controller RAM.
- uint8_t gfx_mono_ssd1306_get_byte (gfx_coord_t page, gfx_coord_t column)

  Get a byte from the display controller RAM.
- void gfx_mono_ssd1306_mask_byte (gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_mask, gfx_mono_color_t color)

  Read/Modify/Write a byte on the display controller.

Variables

- uint8_t framebuffer [GFX_MONO_LCD_FRAMEBUFFER_SIZE]

8.9.1 Detailed Description

Haven Display UG 2832HSWEG04 display glue code for display controller.

8.9.2 Macro Definition Documentation

8.9.2.1 #define CONFIG_SSD1306_FRAMEBUFFER

Definition at line 50 of file gfx_mono_ug_2832hsweg04.c.

8.9.3 Variable Documentation

8.9.3.1 uint8_t framebuffer[GFX_MONO_LCD_FRAMEBUFFER_SIZE]

Definition at line 53 of file gfx_mono_ug_2832hsweg04.c.

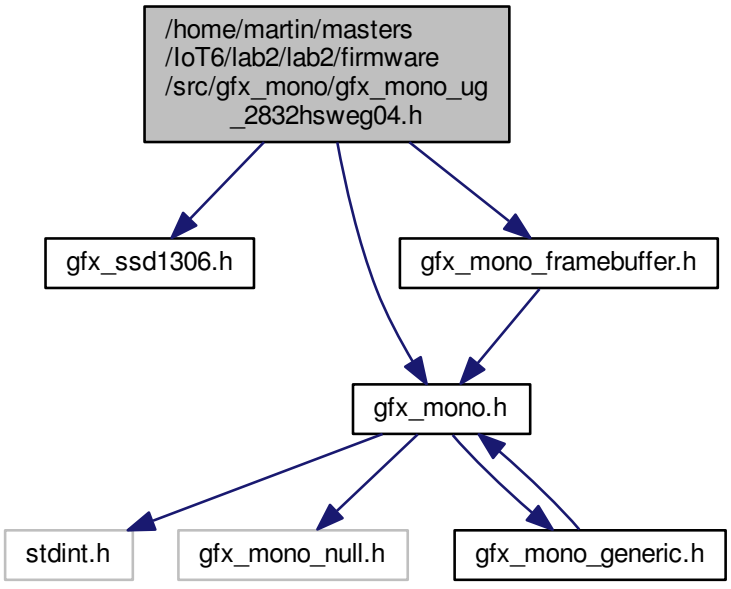
Referenced by gfx_mono_set_framebuffer(), gfx_mono_ssd1306_init(), and gfx_mono_ssd1306_put_framebuffer().

8.10 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_mono_ug_2832hsweg04.h File Reference

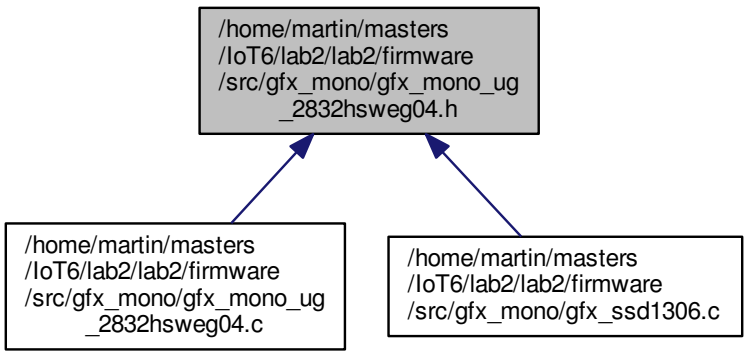
Haven Display UG 2832HSWEG04 display glue code for display controller.

```
#include "gfx_ssd1306.h"
#include "gfx_mono.h"
#include "gfx_mono_framebuffer.h"
```

Include dependency graph for gfx_mono_ug_2832hsweg04.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define GFX_MONO_LCD_WIDTH 128
- #define GFX_MONO_LCD_HEIGHT 32
- #define GFX_MONO_LCD_PIXELS_PER_BYTE 8
- #define GFX_MONO_LCD_PAGES
- #define GFX_MONO_LCD_FRAMEBUFFER_SIZE
- #define gfx_mono_put_bitmap(bitmap, x, y) gfx_mono_generic_put_bitmap(bitmap, x, y)
- #define gfx_mono_draw_pixel(x, y, color) gfx_mono_ssd1306_draw_pixel(x, y, color)
- #define gfx_mono_get_pixel(x, y) gfx_mono_ssd1306_get_pixel(x, y)
- #define gfx_mono_init() gfx_mono_ssd1306_init()
- #define gfx_mono_put_page(data, page, column, width) gfx_mono_ssd1306_put_page(data, page, column, width)
- #define gfx_mono_get_page(data, page, column, width) gfx_mono_ssd1306_get_page(data, page, column, width)
- #define gfx_mono_put_byte(page, column, data) gfx_mono_ssd1306_put_byte(page, column, data, false)
- #define gfx_mono_get_byte(page, column) gfx_mono_ssd1306_get_byte(page, column)
- #define gfx_mono_mask_byte(page, column, pixel_mask, color) gfx_mono_ssd1306_mask_byte(page, column, pixel_mask, color)
- #define gfx_mono_put_framebuffer() gfx_mono_ssd1306_put_framebuffer()

Graphic Drawing Primitives

- #define gfx_mono_draw_horizontal_line(x, y, length, color) gfx_mono_generic_draw_horizontal_line(x, y, length, color)

  Draw a horizontal line, one pixel wide.
- #define gfx_mono_draw_vertical_line(x, y, length, color) gfx_mono_generic_draw_vertical_line(x, y, length, color)

  Draw a vertical line, one pixel wide.
- #define gfx_mono_draw_line(x1, y1, x2, y2, color) gfx_mono_generic_draw_line(x1, y1, x2, y2, color)

  Draw a line between two arbitrary points.
- #define gfx_mono_draw_rect(x, y, width, height, color) gfx_mono_generic_draw_rect(x, y, width, height, color)

  Draw an outline of a rectangle.
- #define gfx_mono_draw_filled_rect(x, y, width, height, color)

Draw a filled rectangle.

- #define gfx_mono_draw_circle(x, y, radius, color, octant_mask)

  Draw an outline of a circle or arc.
- #define gfx_mono_draw_filled_circle(x, y, radius, color, quadrant_mask)

  Draw a filled circle or sector.

Functions

- void gfx_mono_ssd1306_put_framebuffer (void)

  Put framebuffer to LCD controller.
- void gfx_mono_ssd1306_put_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Put a page from RAM to display controller.
- void gfx_mono_ssd1306_get_page (gfx_mono_color_t *data, gfx_coord_t page, gfx_coord_t page_offset, gfx_coord_t width)

  Read a page from the LCD controller.
- void gfx_mono_ssd1306_init (void)

  Initialize SSD1306 controller and LCD display. It will also write the graphic controller RAM to all zeroes.
- void gfx_mono_ssd1306_draw_pixel (gfx_coord_t x, gfx_coord_t y, gfx_mono_color_t color)

  Draw pixel to screen.
- uint8_t gfx_mono_ssd1306_get_pixel (gfx_coord_t x, gfx_coord_t y)

  Get the pixel value at x,y.
- void gfx_mono_ssd1306_put_byte (gfx_coord_t page, gfx_coord_t column, uint8_t data, bool force)

  Put a byte to the display controller RAM.
- uint8_t gfx_mono_ssd1306_get_byte (gfx_coord_t page, gfx_coord_t column)

  Get a byte from the display controller RAM.
- void gfx_mono_ssd1306_mask_byte (gfx_coord_t page, gfx_coord_t column, gfx_mono_color_t pixel_mask, gfx_mono_color_t color)

  Read/Modify/Write a byte on the display controller.

8.10.1 Detailed Description

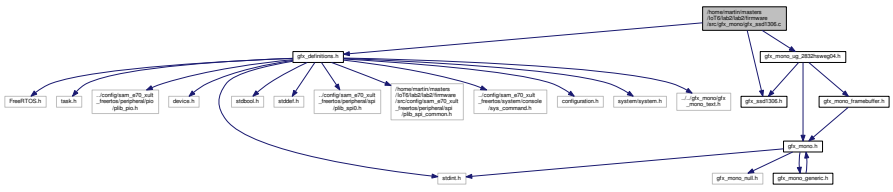
Haven Display UG 2832HSWEG04 display glue code for display controller.

8.11 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_ssd1306.c File Reference

SSD1306 OLED display controller driver.

```
#include "gfx_definitions.h"
#include "gfx_ssd1306.h"
#include "gfx_mono_ug_2832hsweg04.h"
```

Include dependency graph for gfx_ssd1306.c:



Functions

- void ssd1306_init (void)

  Initialize the OLED controller.
- void ssd1306_write_command (uint8_t command)

  Writes a command to the display controller.
- void ssd1306_write_data (uint8_t data)

  Write data to the display controller.

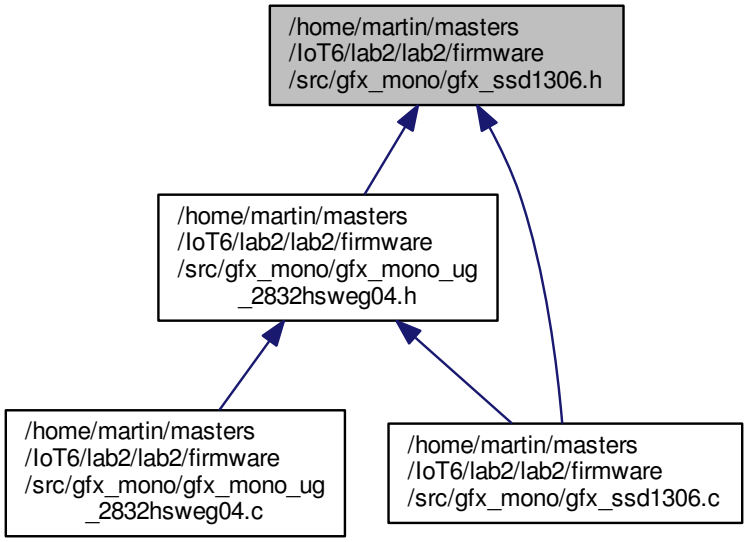
8.11.1 Detailed Description

SSD1306 OLED display controller driver.

8.12 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_ssd1306.h File Reference

SSD1306 OLED display controller driver.

This graph shows which files directly or indirectly include this file:



Macros

Fundamental Command defines

- #define SSD1306_CMD_COL_ADD_SET_LSB(column) (0x00 | (column))
- #define SSD1306_CMD_COL_ADD_SET_MSB(column) (0x10 | (column))
- #define SSD1306_CMD_SET_MEMORY_ADDRESSING_MODE 0x20
- #define SSD1306_CMD_SET_COLUMN_ADDRESS 0x21
- #define SSD1306_CMD_SET_PAGE_ADDRESS 0x22
- #define SSD1306_CMD_SET_DISPLAY_START_LINE(line) (0x40 | (line))
- #define SSD1306_CMD_SET_CONTRAST_CONTROL_FOR_BANK0 0x81
- #define SSD1306_CMD_SET_CHARGE_PUMP_SETTING 0x8D
- #define SSD1306_CMD_SET_SEGMENT_RE_MAP_COL0_SEG0 0xA0
- #define SSD1306_CMD_SET_SEGMENT_RE_MAP_COL127_SEG0 0xA1
- #define SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRAM_ON 0xA4
- #define SSD1306_CMD_ENTIRE_DISPLAY_ON 0xA5
- #define SSD1306_CMD_SET_NORMAL_DISPLAY 0xA6
- #define SSD1306_CMD_SET_INVERSE_DISPLAY 0xA7
- #define SSD1306_CMD_SET_MULTIPLEX_RATIO 0xA8
- #define SSD1306_CMD_SET_DISPLAY_ON 0xAF
- #define SSD1306_CMD_SET_DISPLAY_OFF 0xAE
- #define SSD1306_CMD_SET_PAGE_START_ADDRESS(page) (0xB0 | (page))
- #define SSD1306_CMD_SET_COM_OUTPUT_SCAN_UP 0xC0
- #define SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN 0xC8
- #define SSD1306_CMD_SET_DISPLAY_OFFSET 0xD3
- #define SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_RATIO 0xD5
- #define SSD1306_CMD_SET_PRE_CHARGE_PERIOD 0xD9
- #define SSD1306_CMD_SET_COM_PINS 0xDA
- #define SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL 0xDB
- #define SSD1306_CMD_NOP 0xE3

Graphic Acceleration Command defines

- #define SSD1306_CMD_SCROLL_H_RIGHT 0x26
- #define SSD1306_CMD_SCROLL_H_LEFT 0x27
- #define SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_RIGHT 0x29
- #define SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_H_LEFT 0x2A
- #define SSD1306_CMD_DEACTIVATE_SCROLL 0x2E
- #define SSD1306_CMD_ACTIVATE_SCROLL 0x2F
- #define SSD1306_CMD_SET_VERTICAL_SCROLL_AREA 0xA3

Functions

OLED controller write and read functions

- void ssd1306_write_command (uint8_t command)
  Writes a command to the display controller.
- void ssd1306_write_data (uint8_t data)
  Write data to the display controller.

OLED Controller reset

Sleep control

Address setup for the OLED

Display hardware control

Initialization

- void ssd1306_init (void)
  Initialize the OLED controller.

Variables

- struct spi_module ssd1306_master
- struct spi_slave_inst ssd1306_slave

8.12.1 Detailed Description

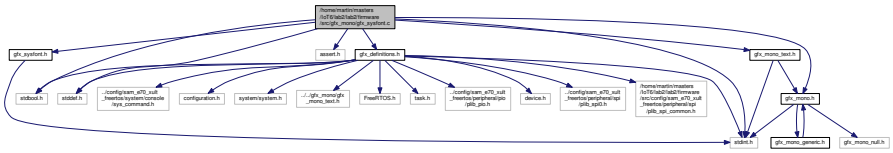
SSD1306 OLED display controller driver.

8.13 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_sysfont.c File Reference

Graphical font support.

```
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <assert.h>
#include "gfx_definitions.h"
#include "gfx_mono_text.h"
#include "gfx_sysfont.h"
#include "gfx_mono.h"
```

Include dependency graph for gfx_sysfont.c:



Variables

- SYSFONT_DEFINE_GLYPHS
- struct font sysfont

  Initialize a basic system font.

8.13.1 Detailed Description

Graphical font support.

8.13.2 Variable Documentation

8.13.2.1 struct font sysfont

Initial value:

```
= {
    .type = FONT_LOC_PROGMEM,
    .width = SYSFONT_WIDTH,
    .height = SYSFONT_HEIGHT,
    .first_char = SYSFONT_FIRSTCHAR,
    .last_char = SYSFONT_LASTCHAR,
    .data =
    {
        .progmem = sysfont_glyphs,
    },
}
```

Initialize a basic system font.

This initializes a basic system font globally usable by the application.

Definition at line 64 of file gfx_sysfont.c.

8.13.2.2 SYSFONT_DEFINE_GLYPHS

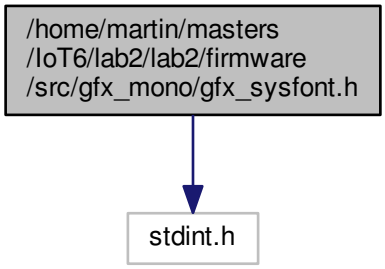
Definition at line 57 of file gfx_sysfont.c.

8.14 /home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx_mono/gfx_sysfont.h File Reference

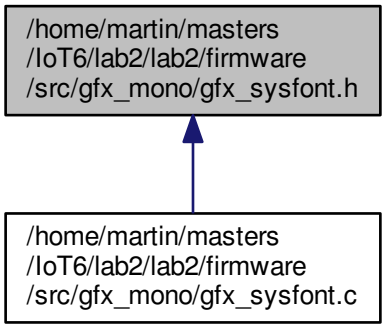
Default configurations for sysfont.

```
#include <stdint.h>
```

Include dependency graph for gfx_sysfont.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define USE_FONT_BPMONO_10x16
- #define SYSFONT_WIDTH 10
- #define SYSFONT_HEIGHT 16
- #define SYSFONT_LINESPACING 8
- #define SYSFONT_FIRSTCHAR ((uint8_t)' ')
- #define SYSFONT_LASTCHAR ((uint8_t)'}')
- #define SYSFONT_DEFINE_GLYPHS

8.14.1 Detailed Description

Default configurations for sysfont.

Index

/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_definitions.h, 73
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono.h, 75
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_framebuffer.c, 77
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_framebuffer.h, 78
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_generic.c, 80
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_generic.h, 81
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_text.c, 82
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_text.h, 84
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_ug_2832hsweg04.c, 85
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_mono_ug_2832hsweg04.h, 87
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_ssd1306.c, 89
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_ssd1306.h, 90
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_sysfont.c, 92
/home/martin/masters/IoT6/lab2/lab2/firmware/src/gfx↩
_mono/gfx_sysfont.h, 93
2832HSWEG04 graphic library abstraction, 47
GFX_MONO_LCD_FRAMEBUFFER_SIZE, 49
GFX_MONO_LCD_HEIGHT, 49
GFX_MONO_LCD_PAGES, 49
GFX_MONO_LCD_PIXELS_PER_BYTE, 49
GFX_MONO_LCD_WIDTH, 49
gfx_mono_draw_pixel, 48
gfx_mono_get_byte, 48
gfx_mono_get_page, 48
gfx_mono_get_pixel, 48
gfx_mono_init, 48
gfx_mono_mask_byte, 49
gfx_mono_put_bitmap, 50
gfx_mono_put_byte, 50
gfx_mono_put_framebuffer, 50
gfx_mono_put_page, 50
gfx_mono_ssd1306_draw_pixel, 50
gfx_mono_ssd1306_get_byte, 51
gfx_mono_ssd1306_get_page, 52
gfx_mono_ssd1306_get_pixel, 53
gfx_mono_ssd1306_init, 54
gfx_mono_ssd1306_mask_byte, 55
gfx_mono_ssd1306_put_byte, 55
gfx_mono_ssd1306_put_framebuffer, 56
gfx_mono_ssd1306_put_page, 57

CONFIG_FONT_PIXELS_PER_BYTE
gfx_mono_text.c, 83
CONFIG_SSD1306_FRAMEBUFFER
gfx_mono_ug_2832hsweg04.c, 86

data
font, 69
gfx_mono_bitmap, 71

EXTMEM_BUF_SIZE
gfx_mono_text.c, 83

FONT_LOC_PROGMEM
GFX Mono Font Library, 41
first_char
font, 69
font, 69
data, 69
first_char, 69
height, 69
last_char, 70
progmem, 70
type, 70
width, 70
font_data_type
GFX Mono Font Library, 41
Framebuffer, 22
gfx_mono_framebuffer_draw_pixel, 22
gfx_mono_framebuffer_get_byte, 23
gfx_mono_framebuffer_get_page, 24
gfx_mono_framebuffer_get_pixel, 25
gfx_mono_framebuffer_mask_byte, 26
gfx_mono_framebuffer_put_byte, 27
gfx_mono_framebuffer_put_page, 27
gfx_mono_set_framebuffer, 28
framebuffer
gfx_mono_ug_2832hsweg04.c, 87

GFX Mono Font Library, 40
FONT_LOC_PROGMEM, 41
font_data_type, 41
gfx_mono_draw_char, 41
gfx_mono_draw_progmem_string, 42
gfx_mono_draw_string, 43
gfx_mono_get_progmem_string_bounding_box,
44

gfx_mono_get_string_bounding_box, 45
GFX_BOTTOMHALF
Monochrome graphical display system, 15
GFX_DATA_CMD_SEL_CLEAR
gfx_definitions.h, 74
GFX_DATA_CMD_SEL_SET
gfx_definitions.h, 74
GFX_DELAY_FUNCTION
gfx_definitions.h, 74
GFX_DISPLAY_RESET_CLEAR
gfx_definitions.h, 74
GFX_DISPLAY_RESET_SET
gfx_definitions.h, 74
GFX_DISPLAY_SS_N_CLEAR
gfx_definitions.h, 74
GFX_DISPLAY_SS_N_SET
gfx_definitions.h, 74
GFX_LEFTHALF
Monochrome graphical display system, 15
GFX_MONO_BITMAP_PROGMEM
Monochrome graphical display system, 21
GFX_MONO_BITMAP_RAM
Monochrome graphical display system, 21
GFX_MONO_LCD_FRAMEBUFFER_SIZE
2832HSWEG04 graphic library abstraction, 49
GFX_MONO_LCD_HEIGHT
2832HSWEG04 graphic library abstraction, 49
GFX_MONO_LCD_PAGES
2832HSWEG04 graphic library abstraction, 49
GFX_MONO_LCD_PIXELS_PER_BYTE
2832HSWEG04 graphic library abstraction, 49
GFX_MONO_LCD_WIDTH
2832HSWEG04 graphic library abstraction, 49
GFX_MONO_UG_2832HSWEG04
gfx_definitions.h, 74
GFX_OCTANT0
Monochrome graphical display system, 18
GFX_OCTANT1
Monochrome graphical display system, 19
GFX_OCTANT2
Monochrome graphical display system, 19
GFX_OCTANT3
Monochrome graphical display system, 19
GFX_OCTANT4
Monochrome graphical display system, 19
GFX_OCTANT5
Monochrome graphical display system, 19
GFX_OCTANT6
Monochrome graphical display system, 19
GFX_OCTANT7
Monochrome graphical display system, 19
GFX_PIXEL_CLR
Monochrome graphical display system, 21
GFX_PIXEL_SET
Monochrome graphical display system, 21
GFX_PIXEL_XOR
Monochrome graphical display system, 21
GFX_QUADRANT0
Monochrome graphical display system, 20
GFX_QUADRANT1
Monochrome graphical display system, 20
GFX_QUADRANT2
Monochrome graphical display system, 20
GFX_QUADRANT3
Monochrome graphical display system, 20
GFX_RIGHTHALF
Monochrome graphical display system, 20
GFX_SPI_IS_BUSY
gfx_definitions.h, 75
GFX_SPI_WRITE_FUNCTION
gfx_definitions.h, 75
GFX_TOPHALF
Monochrome graphical display system, 20
GFX_WHOLE
Monochrome graphical display system, 21
Generic monochrome graphic primitives, 30
gfx_mono_generic_draw_circle, 31
gfx_mono_generic_draw_filled_circle, 32
gfx_mono_generic_draw_filled_rect, 33
gfx_mono_generic_draw_horizontal_line, 35
gfx_mono_generic_draw_line, 36
gfx_mono_generic_draw_rect, 37
gfx_mono_generic_draw_vertical_line, 38
gfx_mono_generic_put_bitmap, 39
gfx_coord_t
Monochrome graphical display system, 21
gfx_definitions.h
GFX_DATA_CMD_SEL_CLEAR, 74
GFX_DATA_CMD_SEL_SET, 74
GFX_DELAY_FUNCTION, 74
GFX_DISPLAY_RESET_CLEAR, 74
GFX_DISPLAY_RESET_SET, 74
GFX_DISPLAY_SS_N_CLEAR, 74
GFX_DISPLAY_SS_N_SET, 74
GFX_MONO_UG_2832HSWEG04, 74
GFX_SPI_IS_BUSY, 75
GFX_SPI_WRITE_FUNCTION, 75
PRINTF_BLOCKING, 75
gfx_mono.h
PROGMEM_DECLARE, 77
PROGMEM_PTR_T, 77
PROGMEM_READ_BYTE, 77
PROGMEM_STRING_T, 77
PROGMEM_T, 77
gfx_mono_bitmap, 70
data, 71
height, 71
pixmap, 71
progmem, 71
type, 71
width, 72
gfx_mono_bitmap_type
Monochrome graphical display system, 21
gfx_mono_color
Monochrome graphical display system, 21
gfx_mono_color_t

Monochrome graphical display system, 21
gfx_mono_draw_char
GFX Mono Font Library, 41
gfx_mono_draw_circle
Monochrome graphical display system, 16
gfx_mono_draw_filled_circle
Monochrome graphical display system, 16
gfx_mono_draw_filled_rect
Monochrome graphical display system, 17
gfx_mono_draw_horizontal_line
Monochrome graphical display system, 17
gfx_mono_draw_line
Monochrome graphical display system, 17
gfx_mono_draw_pixel
2832HSWEG04 graphic library abstraction, 48
gfx_mono_draw_progmem_string
GFX Mono Font Library, 42
gfx_mono_draw_rect
Monochrome graphical display system, 18
gfx_mono_draw_string
GFX Mono Font Library, 43
gfx_mono_draw_vertical_line
Monochrome graphical display system, 18
gfx_mono_framebuffer_draw_pixel
Framebuffer, 22
gfx_mono_framebuffer_get_byte
Framebuffer, 23
gfx_mono_framebuffer_get_page
Framebuffer, 24
gfx_mono_framebuffer_get_pixel
Framebuffer, 25
gfx_mono_framebuffer_mask_byte
Framebuffer, 26
gfx_mono_framebuffer_put_byte
Framebuffer, 27
gfx_mono_framebuffer_put_page
Framebuffer, 27
gfx_mono_generic_draw_circle
Generic monochrome graphic primitives, 31
gfx_mono_generic_draw_filled_circle
Generic monochrome graphic primitives, 32
gfx_mono_generic_draw_filled_rect
Generic monochrome graphic primitives, 33
gfx_mono_generic_draw_horizontal_line
Generic monochrome graphic primitives, 35
gfx_mono_generic_draw_line
Generic monochrome graphic primitives, 36
gfx_mono_generic_draw_rect
Generic monochrome graphic primitives, 37
gfx_mono_generic_draw_vertical_line
Generic monochrome graphic primitives, 38
gfx_mono_generic_put_bitmap
Generic monochrome graphic primitives, 39
gfx_mono_get_byte
2832HSWEG04 graphic library abstraction, 48
gfx_mono_get_page
2832HSWEG04 graphic library abstraction, 48
gfx_mono_get_pixel
2832HSWEG04 graphic library abstraction, 48
gfx_mono_get_progmem_string_bounding_box
GFX Mono Font Library, 44
gfx_mono_get_string_bounding_box
GFX Mono Font Library, 45
gfx_mono_init
2832HSWEG04 graphic library abstraction, 48
gfx_mono_mask_byte
2832HSWEG04 graphic library abstraction, 49
gfx_mono_put_bitmap
2832HSWEG04 graphic library abstraction, 50
gfx_mono_put_byte
2832HSWEG04 graphic library abstraction, 50
gfx_mono_put_framebuffer
2832HSWEG04 graphic library abstraction, 50
gfx_mono_put_page
2832HSWEG04 graphic library abstraction, 50
gfx_mono_set_framebuffer
Framebuffer, 28
gfx_mono_ssd1306_draw_pixel
2832HSWEG04 graphic library abstraction, 50
gfx_mono_ssd1306_get_byte
2832HSWEG04 graphic library abstraction, 51
gfx_mono_ssd1306_get_page
2832HSWEG04 graphic library abstraction, 52
gfx_mono_ssd1306_get_pixel
2832HSWEG04 graphic library abstraction, 53
gfx_mono_ssd1306_init
2832HSWEG04 graphic library abstraction, 54
gfx_mono_ssd1306_mask_byte
2832HSWEG04 graphic library abstraction, 55
gfx_mono_ssd1306_put_byte
2832HSWEG04 graphic library abstraction, 55
gfx_mono_ssd1306_put_framebuffer
2832HSWEG04 graphic library abstraction, 56
gfx_mono_ssd1306_put_page
2832HSWEG04 graphic library abstraction, 57
gfx_mono_text.c
CONFIG_FONT_PIXELS_PER_BYTE, 83
EXTMEM_BUF_SIZE, 83
gfx_mono_ug_2832hsweg04.c
CONFIG_SSD1306_FRAMEBUFFER, 86
framebuffer, 87
Gfx_sysfont, 68
SYSFONT_DEFINE_GLYPHS, 68
SYSFONT_FIRSTCHAR, 68
SYSFONT_HEIGHT, 68
SYSFONT_LASTCHAR, 68
SYSFONT_LINESPACING, 68
SYSFONT_WIDTH, 68
USE_FONT_BPMONO_10x16, 68
gfx_sysfont.c
SYSFONT_DEFINE_GLYPHS, 93
sysfont, 93

height
font, 69
gfx_mono_bitmap, 71

last_char
font, 70

Monochrome graphical display system, 13
GFX_BOTTOMHALF, 15
GFX_LEFTHALF, 15
GFX_MONO_BITMAP_PROGMEM, 21
GFX_MONO_BITMAP_RAM, 21
GFX_OCTANT0, 18
GFX_OCTANT1, 19
GFX_OCTANT2, 19
GFX_OCTANT3, 19
GFX_OCTANT4, 19
GFX_OCTANT5, 19
GFX_OCTANT6, 19
GFX_OCTANT7, 19
GFX_PIXEL_CLR, 21
GFX_PIXEL_SET, 21
GFX_PIXEL_XOR, 21
GFX_QUADRANT0, 20
GFX_QUADRANT1, 20
GFX_QUADRANT2, 20
GFX_QUADRANT3, 20
GFX_RIGHTHALF, 20
GFX_TOPHALF, 20
GFX_WHOLE, 21
gfx_coord_t, 21
gfx_mono_bitmap_type, 21
gfx_mono_color, 21
gfx_mono_color_t, 21
gfx_mono_draw_circle, 16
gfx_mono_draw_filled_circle, 16
gfx_mono_draw_filled_rect, 17
gfx_mono_draw_horizontal_line, 17
gfx_mono_draw_line, 17
gfx_mono_draw_rect, 18
gfx_mono_draw_vertical_line, 18

PRINTF_BLOCKING
gfx_definitions.h, 75
PROGMEM_DECLARE
gfx_mono.h, 77
PROGMEM_PTR_T
gfx_mono.h, 77
PROGMEM_READ_BYTE
gfx_mono.h, 77
PROGMEM_STRING_T
gfx_mono.h, 77
PROGMEM_T
gfx_mono.h, 77
pixmap
gfx_mono_bitmap, 71
progmem
font, 70
gfx_mono_bitmap, 71

SSD1306 OLED Controller Low-level driver, 59
SSD1306_CMD_ACTIVATE_SCROLL, 60
SSD1306_CMD_COL_ADD_SET_LSB, 60
SSD1306_CMD_COL_ADD_SET_MSB, 60
SSD1306_CMD_CONTINUOUS_SCROLL_V_A↩
ND_H_LEFT, 61
SSD1306_CMD_CONTINUOUS_SCROLL_V_A↩
ND_H_RIGHT, 61
SSD1306_CMD_DEACTIVATE_SCROLL, 61
SSD1306_CMD_ENTIRE_DISPLAY_AND_GD↩
DRAM_ON, 61
SSD1306_CMD_ENTIRE_DISPLAY_ON, 61
SSD1306_CMD_NOP, 61
SSD1306_CMD_SCROLL_H_LEFT, 61
SSD1306_CMD_SCROLL_H_RIGHT, 61
SSD1306_CMD_SET_CHARGE_PUMP_SETTI↩
NG, 61
SSD1306_CMD_SET_COLUMN_ADDRESS, 61
SSD1306_CMD_SET_COM_OUTPUT_SCAN_↩
DOWN, 62
SSD1306_CMD_SET_COM_OUTPUT_SCAN_↩
UP, 62
SSD1306_CMD_SET_COM_PINS, 62
SSD1306_CMD_SET_CONTRAST_CONTROL↩
_FOR_BANK0, 62
SSD1306_CMD_SET_DISPLAY_CLOCK_DIVI↩
DE_RATIO, 62
SSD1306_CMD_SET_DISPLAY_OFFSET, 62
SSD1306_CMD_SET_DISPLAY_OFF, 62
SSD1306_CMD_SET_DISPLAY_ON, 62
SSD1306_CMD_SET_DISPLAY_START_LINE,
62
SSD1306_CMD_SET_INVERSE_DISPLAY, 63
SSD1306_CMD_SET_MEMORY_ADDRESSIN↩
G_MODE, 63
SSD1306_CMD_SET_MULTIPLEX_RATIO, 63
SSD1306_CMD_SET_NORMAL_DISPLAY, 63
SSD1306_CMD_SET_PAGE_ADDRESS, 63
SSD1306_CMD_SET_PAGE_START_ADDRESS,
63
SSD1306_CMD_SET_PRE_CHARGE_PERIOD,
63
SSD1306_CMD_SET_SEGMENT_RE_MAP_C↩
OL0_SEG0, 63
SSD1306_CMD_SET_SEGMENT_RE_MAP_C↩
OL127_SEG0, 63
SSD1306_CMD_SET_VCOMH_DESELECT_LE↩
VEL, 64
SSD1306_CMD_SET_VERTICAL_SCROLL_A↩
REA, 64
ssd1306_init, 64
ssd1306_master, 67
ssd1306_slave, 67
ssd1306_write_command, 65
ssd1306_write_data, 66
SSD1306_CMD_ACTIVATE_SCROLL
SSD1306 OLED Controller Low-level driver, 60
SSD1306_CMD_COL_ADD_SET_LSB
SSD1306 OLED Controller Low-level driver, 60
SSD1306_CMD_COL_ADD_SET_MSB
SSD1306 OLED Controller Low-level driver, 60

SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_↩
H_LEFT
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_CONTINUOUS_SCROLL_V_AND_↩
H_RIGHT
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_DEACTIVATE_SCROLL
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_ENTIRE_DISPLAY_AND_GDDRA↩
M_ON
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_ENTIRE_DISPLAY_ON
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_NOP
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_SCROLL_H_LEFT
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_SCROLL_H_RIGHT
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_SET_CHARGE_PUMP_SETTING
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_SET_COLUMN_ADDRESS
SSD1306 OLED Controller Low-level driver, 61
SSD1306_CMD_SET_COM_OUTPUT_SCAN_DOWN
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_COM_OUTPUT_SCAN_UP
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_COM_PINS
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_CONTRAST_CONTROL_FO↩
R_BANK0
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_DISPLAY_CLOCK_DIVIDE_R↩
ATIO
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_DISPLAY_OFFSET
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_DISPLAY_OFF
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_DISPLAY_ON
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_DISPLAY_START_LINE
SSD1306 OLED Controller Low-level driver, 62
SSD1306_CMD_SET_INVERSE_DISPLAY
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_MEMORY_ADDRESSING_M↩
ODE
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_MULTIPLEX_RATIO
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_NORMAL_DISPLAY
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_PAGE_ADDRESS
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_PAGE_START_ADDRESS
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_PRE_CHARGE_PERIOD
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_SEGMENT_RE_MAP_COL0_↩
SEG0
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_SEGMENT_RE_MAP_CO↩
L127_SEG0
SSD1306 OLED Controller Low-level driver, 63
SSD1306_CMD_SET_VCOMH_DESELECT_LEVEL
SSD1306 OLED Controller Low-level driver, 64
SSD1306_CMD_SET_VERTICAL_SCROLL_AREA
SSD1306 OLED Controller Low-level driver, 64
SYSFONT_DEFINE_GLYPHS
Gfx_sysfont, 68
gfx_sysfont.c, 93
SYSFONT_FIRSTCHAR
Gfx_sysfont, 68
SYSFONT_HEIGHT
Gfx_sysfont, 68
SYSFONT_LASTCHAR
Gfx_sysfont, 68
SYSFONT_LINESPACING
Gfx_sysfont, 68
SYSFONT_WIDTH
Gfx_sysfont, 68
ssd1306_init
SSD1306 OLED Controller Low-level driver, 64
ssd1306_master
SSD1306 OLED Controller Low-level driver, 67
ssd1306_slave
SSD1306 OLED Controller Low-level driver, 67
ssd1306_write_command
SSD1306 OLED Controller Low-level driver, 65
ssd1306_write_data
SSD1306 OLED Controller Low-level driver, 66
sysfont
gfx_sysfont.c, 93

type
font, 70
gfx_mono_bitmap, 71

USE_FONT_BPMONO_10x16
Gfx_sysfont, 68

width
font, 70
gfx_mono_bitmap, 72