ST7565 PSoC Component

1.0

Generated by Doxygen 1.8.10

Thu Dec 10 2015 13:03:53

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 Source/ST7565-Component.cylib/ST7565_Serial_v1_0/API/drawing.c File Reference

ST7565 Drawing methods.

```
#include "`$INSTANCE_NAME`_drawing.h"
#include "`$INSTANCE_NAME`_font.h"
```

Functions

- void $INSTANCE_NAME _draw_pixel (uint8_t x, uint8_t y, uint8_t color)
  Draw a specific pixel.
- void $INSTANCE_NAME _draw_circle (uint8_t x0, uint8_t y0, uint8_t r, uint8_t color)
  Draw a circle.
- void $INSTANCE_NAME _draw_line (uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1, uint8_t color)
  Draw a line.
- void $INSTANCE_NAME _draw_line_hoz (uint8_t x, uint8_t y, uint8_t w, uint8_t color)
  Draw a horizontal line. Faster than drawing a normal line.
- void $INSTANCE_NAME _draw_line_ver (uint8_t x, uint8_t y, uint8_t h, uint8_t color)
  Draw a vertical line. Faster than drawing a normal line.
- void $INSTANCE_NAME _draw_rect (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t color)
  Draw a rectangle.
- void $INSTANCE_NAME _draw_character (uint8_t x, uint8_t y, uint8_t color, uint8_t size, uint8_t c)
  Draw a character.
- void $INSTANCE_NAME _fill_rect (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t color)
  Fill a rectangle.
- void $INSTANCE_NAME _draw_fillScreen (uint8_t color)
  Fill the whole screen.
- void $INSTANCE_NAME _set_cursor (uint8_t x, uint8_t y)
  Sets the cursor to a specific position. This is used by write∗().
- void $INSTANCE_NAME _set_textColor (uint8_t color)
  Sets the text color.
- void $INSTANCE_NAME _set_textSize (uint8_t size)
  Sets the text size.
- void $INSTANCE_NAME _write_char (uint8_t c)
  Writes a char to the cursor position.
- void $INSTANCE_NAME _write_string (char ∗c)
  Writes a string to the cursor position.

Variables

- uint8_t $INSTANCE_NAME **_buffer** [128 ∗64/8]
- uint8_t $INSTANCE_NAME **_temp**
- uint8_t $INSTANCE_NAME **_cursor_x** = 0
- uint8_t $INSTANCE_NAME **_cursor_y** = 0
- uint8_t $INSTANCE_NAME **_txtcolor** = 1
- uint8_t $INSTANCE_NAME **_txtsize** = 1

2.1.1 Detailed Description

ST7565 Drawing methods.

Author

Thanasis Georgiou

2.1.2 Function Documentation

2.1.2.1 void $INSTANCE_NAME _draw_character (uint8_t *x,* uint8_t *y,* uint8_t *color,* uint8_t *size,* uint8_t *c*)

Draw a character.

Parameters

x	Where to draw the character's first pixel (top-left) (X).
y	Where to draw the character's first pixel (top-left) (X).
color	The character's color (0 clears, 1 sets).
size	How large should the character be. 1 = 8 pixels height.
c	The character to draw.

2.1.2.2 void $INSTANCE_NAME _draw_circle (uint8_t *x0,* uint8_t *y0,* uint8_t *r,* uint8_t *color*)

Draw a circle.

Parameters

x0	The X position of the circle's center.
y0	The Y position of the circle's center.
r	The circle's radius.
color	The color to use (0 clears, 1 sets).

2.1.2.3 void $INSTANCE_NAME _draw_fillScreen (uint8_t *color*)

Fill the whole screen.

Parameters

color	The color to fill (0 clears, 1 sets).

2.1.2.4 void $INSTANCE_NAME _draw_line (uint8_t *x0,* uint8_t *y0,* uint8_t *x1,* uint8_t *y1,* uint8_t *color*)

Draw a line.

Parameters

x0	X position of the line's first point.
y0	Y position of the line's first point.
x1	X position of the line's second point.
y1	Y position of the line's second point.
color	The color to use (0 clears, 1 sets).

2.1.2.5 void $INSTANCE_NAME _draw_line_hoz (uint8_t *x,* uint8_t *y,* uint8_t *w,* uint8_t *color*)

Draw a horizontal line. Faster than drawing a normal line.

Parameters

x	The X of the line's left point.
y	The Y of the line's left point.
w	The line's width.
color	The line's color (0 clears, 1 sets).

2.1.2.6 void $INSTANCE_NAME _draw_line_ver (uint8_t *x,* uint8_t *y,* uint8_t *h,* uint8_t *color*)

Draw a vertical line. Faster than drawing a normal line.

Parameters

x	The X of the line's top point.
y	The Y of the line's top point.
h	The line's height.
color	The line's color (0 clears, 1 sets).

2.1.2.7 void $INSTANCE_NAME _draw_pixel (uint8_t *x,* uint8_t *y,* uint8_t *color*)

Draw a specific pixel.

Parameters

x	The X position of the pixel.
y	The Y position of the pixel.
color	0 To clear the pixel, 1 to turn it on.

2.1.2.8 void $INSTANCE_NAME _draw_rect (uint8_t *x,* uint8_t *y,* uint8_t *w,* uint8_t *h,* uint8_t *color*)

Draw a rectangle.

Parameters

x	The X of the rectangle's top left point.
y	The Y of the rectangle's top left point.
h	The rectangle's height.
w	The rectangle's width.
color	The line's color (0 clears, 1 sets).

2.1.2.9 void $INSTANCE_NAME _fill_rect (uint8_t *x,* uint8_t *y,* uint8_t *w,* uint8_t *h,* uint8_t *color*)

Fill a rectangle.

Parameters

x	The X of the rectangle's top left point.
y	The Y of the rectangle's top left point.
h	The rectangle's height.
w	The rectangle's width.
color	The line's color (0 clears, 1 sets).

2.1.2.10 void $INSTANCE_NAME _set_cursor (uint8_t *x*, uint8_t *y*)

Sets the cursor to a specific position. This is used by *write*∗().

Parameters

x	The X position of the cursor (pixels).
y	The Y position of the cursor (pixels).

2.1.2.11 void $INSTANCE_NAME _set_textColor (uint8_t *color*)

Sets the text color.

Parameters

color	The text' color (0 white, 1 black).

2.1.2.12 void $INSTANCE_NAME _set_textSize (uint8_t *size*)

Sets the text size.

Parameters

size	The new text size (1 = 8 pixels height).

2.1.2.13 void $INSTANCE_NAME _write_char (uint8_t *c*)

Writes a char to the cursor position.

Parameters

c	The char to write.

2.1.2.14 void $INSTANCE_NAME _write_string (char ∗ *c*)

Writes a string to the cursor position.

Parameters

c	The string to write (terminated by \0).

2.2 Source/ST7565-Component.cylib/ST7565_Serial_v1_0/API/drawing.h File Reference

ST7565 Drawing methods.

```
#include "cytypes.h"
```

Macros

- #define SCREEN_WIDTH 128
- #define **SCREEN_HEIGHT** 64

Functions

- void $INSTANCE_NAME _draw_pixel (uint8_t x, uint8_t y, uint8_t color)

  Draw a specific pixel.
- void $INSTANCE_NAME _draw_circle (uint8_t x0, uint8_t y0, uint8_t r, uint8_t color)

  Draw a circle.
- void $INSTANCE_NAME _draw_line (uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1, uint8_t color)

  Draw a line.
- void $INSTANCE_NAME _draw_line_hoz (uint8_t x, uint8_t y, uint8_t w, uint8_t color)

  Draw a horizontal line. Faster than drawing a normal line.
- void $INSTANCE_NAME _draw_line_ver (uint8_t x, uint8_t y, uint8_t h, uint8_t color)

  Draw a vertical line. Faster than drawing a normal line.
- void $INSTANCE_NAME _draw_rect (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t color)

  Draw a rectangle.
- void $INSTANCE_NAME _draw_character (uint8_t x, uint8_t y, uint8_t color, uint8_t size, uint8_t c)

  Draw a character.
- void $INSTANCE_NAME _fill_rect (uint8_t x, uint8_t y, uint8_t w, uint8_t h, uint8_t color)

  Fill a rectangle.
- void $INSTANCE_NAME _draw_fillScreen (uint8_t color)

  Fill the whole screen.
- void $INSTANCE_NAME _set_cursor (uint8_t x, uint8_t y)

  Sets the cursor to a specific position. This is used by write∗().
- void $INSTANCE_NAME _set_textColor (uint8_t color)

  Sets the text color.
- void $INSTANCE_NAME _set_textSize (uint8_t size)

  Sets the text size.
- void $INSTANCE_NAME _write_char (uint8_t c)

  Writes a char to the cursor position.
- void $INSTANCE_NAME _write_string (char ∗c)

  Writes a string to the cursor position.

2.2.1 Detailed Description

ST7565 Drawing methods.

Author

Thanasis Georgiou

2.2.2 Macro Definition Documentation

2.2.2.1 #define SCREEN_WIDTH 128

The screen's width

2.2.3 Function Documentation

2.2.3.1 void $INSTANCE_NAME _draw_character (uint8_t *x,* uint8_t *y,* uint8_t *color,* uint8_t *size,* uint8_t *c*)

Draw a character.

Parameters

x	Where to draw the character's first pixel (top-left) (X).
y	Where to draw the character's first pixel (top-left) (X).
color	The character's color (0 clears, 1 sets).
size	How large should the character be. 1 = 8 pixels height.
c	The character to draw.

2.2.3.2 void $INSTANCE_NAME _draw_circle (uint8_t *x0,* uint8_t *y0,* uint8_t *r,* uint8_t *color*)

Draw a circle.

Parameters

x0	The X position of the circle's center.
y0	The Y position of the circle's center.
r	The circle's radius.
color	The color to use (0 clears, 1 sets).

2.2.3.3 void $INSTANCE_NAME _draw_fillScreen (uint8_t *color*)

Fill the whole screen.

Parameters

color	The color to fill (0 clears, 1 sets).

2.2.3.4 void $INSTANCE_NAME _draw_line (uint8_t *x0,* uint8_t *y0,* uint8_t *x1,* uint8_t *y1,* uint8_t *color*)

Draw a line.

Parameters

x0	X position of the line's first point.
y0	Y position of the line's first point.
x1	X position of the line's second point.
y1	Y position of the line's second point.
color	The color to use (0 clears, 1 sets).

2.2.3.5 void $INSTANCE_NAME _draw_line_hoz (uint8_t *x,* uint8_t *y,* uint8_t *w,* uint8_t *color*)

Draw a horizontal line. Faster than drawing a normal line.

Parameters

x	The X of the line's left point.
y	The Y of the line's left point.

w	The line's width.
color	The line's color (0 clears, 1 sets).

2.2.3.6 void $INSTANCE_NAME _draw_line_ver (uint8_t *x,* uint8_t *y,* uint8_t *h,* uint8_t *color*)

Draw a vertical line. Faster than drawing a normal line.

Parameters

x	The X of the line's top point.
y	The Y of the line's top point.
h	The line's height.
color	The line's color (0 clears, 1 sets).

2.2.3.7 void $INSTANCE_NAME _draw_pixel (uint8_t *x,* uint8_t *y,* uint8_t *color*)

Draw a specific pixel.

Parameters

x	The X position of the pixel.
y	The Y position of the pixel.
color	0 To clear the pixel, 1 to turn it on.

2.2.3.8 void $INSTANCE_NAME _draw_rect (uint8_t *x,* uint8_t *y,* uint8_t *w,* uint8_t *h,* uint8_t *color*)

Draw a rectangle.

Parameters

x	The X of the rectangle's top left point.
y	The Y of the rectangle's top left point.
h	The rectangle's height.
w	The rectangle's width.
color	The line's color (0 clears, 1 sets).

2.2.3.9 void $INSTANCE_NAME _fill_rect (uint8_t *x,* uint8_t *y,* uint8_t *w,* uint8_t *h,* uint8_t *color*)

Fill a rectangle.

Parameters

x	The X of the rectangle's top left point.
y	The Y of the rectangle's top left point.
h	The rectangle's height.
w	The rectangle's width.
color	The line's color (0 clears, 1 sets).

2.2.3.10 void $INSTANCE_NAME _set_cursor (uint8_t *x,* uint8_t *y*)

Sets the cursor to a specific position. This is used by *write*∗().

Parameters

x	The X position of the cursor (pixels).
y	The Y position of the cursor (pixels).

2.2.3.11 void $INSTANCE_NAME _set_textColor (uint8_t *color*)

Sets the text color.

Parameters

color	The text' color (0 white, 1 black).

2.2.3.12 void $INSTANCE_NAME _set_textSize (uint8_t *size*)

Sets the text size.

Parameters

size	The new text size (1 = 8 pixels height).

2.2.3.13 void $INSTANCE_NAME _write_char (uint8_t *c*)

Writes a char to the cursor position.

Parameters

c	The char to write.

2.2.3.14 void $INSTANCE_NAME _write_string (char * *c*)

Writes a string to the cursor position.

Parameters

c	The string to write (terminated by \0).

2.3 Source/ST7565-Component.cylib/ST7565_Serial_v1_0/API/font.h File Reference

Standard ASCII font from the amazing AdafruitGFXLibrary.

2.3.1 Detailed Description

Standard ASCII font from the amazing AdafruitGFXLibrary.

Author

Adafruit Industries

2.4 Source/ST7565-Component.cylib/ST7565_Serial_v1_0/API/st7565.c File Reference

ST7565 Control.

```
#include "`$INSTANCE_NAME`_st7565.h"
```

Functions

- **CY_ISR** (‘$INSTANCE_NAME‘_InterruptHandler)
- void $INSTANCE_NAME _reset ()

  Resets the LCD.
- void $INSTANCE_NAME _init ()

  Initialize the LCD.
- void $INSTANCE_NAME _set_contrast (uint8_t value)

  Sets the display contrast.
- void $INSTANCE_NAME _refresh ()

  Instructs that a refresh takes place.
- void $INSTANCE_NAME _refresh_loop ()

  Refresh the display if required. PLACE IN MAIN LOOP.
- int8 $INSTANCE_NAME _is_refreshing ()

  Checks if the display is currently refreshing.
- uint8_t ∗$INSTANCE_NAME _get_buffer ()

  Returns the display buffer.

Variables

- int $INSTANCE_NAME **_pagemap** [ ] = { 3, 2, 1, 0, 7, 6, 5, 4 }
- uint8_t $INSTANCE_NAME **_buffer** [ ]
- int8_t $INSTANCE_NAME **_txFinished** = 0
- int8_t $INSTANCE_NAME **_isRefreshing** = 0
- int8_t $INSTANCE_NAME **_currentPage** = -1
- int8_t $INSTANCE_NAME **_refreshStage** = 0
- uint8_t $INSTANCE_NAME **_selectPage** [ ] = {0, CMD_SET_COLUMN_LOWER | (0x0 & 0xf), CMD_SET↩ _COLUMN_UPPER | ((0x0 >> 4) & 0xf), CMD_RMW}

2.4.1 Detailed Description

ST7565 Control.

Author

Thanasis Georgiou

2.4.2 Function Documentation

2.4.2.1 uint8_t∗ $INSTANCE_NAME _get_buffer ( )

Returns the display buffer.

/returns The pointer to the display buffer. The length is 1024 bytes.

2.4.2.2 void $INSTANCE_NAME _init ( )

Initialize the LCD.

Initializes the LCD. In detail:

- Initializes the hardware SPI port and enables interrupts.
- Resets the LCD to a known state.
- Sets the LCD Bias level, selects the ADC and the SHL.
- Turns on the power circuits of the controller.
- Sets the operating voltage and the contrast.
- Empty the memory buffer.
- Turns the display on.

For this display, a 1024 byte buffer is allocated in the PSoC's SRAM. There is no need to do double buffering since a refresh is only carried out when instructed by _refresh().

2.4.2.3 int8 $INSTANCE_NAME _is_refreshing ( )

Checks if the display is currently refreshing.

/returns 1 if the display is refreshing, 0 if not.

2.4.2.4 void $INSTANCE_NAME _refresh ( )

Instructs that a refresh takes place.

Sets the "refresh" flag so the display can start refreshing on the next main loop. Assuming each SPI transfer finishes before the main loop ends it will take 8 + 8 main loops to refresh the whole display.

2.4.2.5 void $INSTANCE_NAME _refresh_loop ( )

Refresh the display if required. PLACE IN MAIN LOOP.

This will refresh the display depanding on if it's required and the last refresh state of the display. It is REQUIRED to place this inside the main loop of your application, differently the display will never refresh.

2.4.2.6 void $INSTANCE_NAME _reset ( )

Resets the LCD.

Resets the display

2.4.2.7 void $INSTANCE_NAME _set_contrast (uint8_t *value*)

Sets the display contrast.

Sets the display contrast by sending two bytes.

Parameters

value	The new contrast value (0-63). Values outside the range are clipped to 63.

2.5 Source/ST7565-Component.cylib/ST7565_Serial_v1_0/API/st7565.h File Reference

ST7565 Control.

```
#include "cytypes.h"
#include "cyfitter.h"
#include <stdint.h>
#include "`$INSTANCE_NAME`_drawing.h"
```

Macros

- #define **HIGH** 1
- #define **LOW** 0
- #define CMD_DISPLAY_OFF 0xAE
- #define CMD_DISPLAY_ON 0xAF
- #define CMD_SET_DISP_START_LINE 0x40
- #define CMD_SET_PAGE 0xB0
- #define CMD_SET_COLUMN_UPPER 0x10
- #define CMD_SET_COLUMN_LOWER 0x00
- #define CMD_SET_ADC_NORMAL 0xA0
- #define CMD_SET_ADC_REVERSE 0xA1
- #define CMD_SET_DISP_NORMAL 0xA6
- #define CMD_SET_DISP_REVERSE 0xA7
- #define **CMD_SET_ALLPTS_NORMAL** 0xA4
- #define **CMD_SET_ALLPTS_ON** 0xA5
- #define **CMD_SET_BIAS_9** 0xA2
- #define **CMD_SET_BIAS_7** 0xA3
- #define **CMD_RMW** 0xE0
- #define **CMD_RMW_CLEAR** 0xEE
- #define **CMD_INTERNAL_RESET** 0xE2
- #define **CMD_SET_COM_NORMAL** 0xC0
- #define **CMD_SET_COM_REVERSE** 0xC8
- #define **CMD_SET_POWER_CONTROL** 0x28
- #define **CMD_SET_RESISTOR_RATIO** 0x20
- #define **CMD_SET_CONTRAST** 0x81
- #define **CMD_SET_STATIC_OFF** 0xAC
- #define **CMD_SET_STATIC_ON** 0xAD
- #define **CMD_SET_STATIC_REG** 0x0
- #define **CMD_SET_BOOSTER_FIRST** 0xF8
- #define **CMD_SET_BOOSTER_234** 0
- #define **CMD_SET_BOOSTER_5** 1
- #define **CMD_SET_BOOSTER_6** 3
- #define **CMD_NOP** 0xE3
- #define **CMD_TEST** 0xF0
- #define **MSB_POSITION** 0x80u
- #define **SHIFT_BY_1** 0x01u
- #define **INSTANCE_NAME** `_unselect() `$INSTANCE_NAME`_Pin_CS_Write(HIGH);
- #define **INSTANCE_NAME** `_select() `$INSTANCE_NAME`_Pin_CS_Write(LOW);
- #define **INSTANCE_NAME** `_mode_control() `$INSTANCE_NAME`_Pin_A0_Write(LOW);
- #define **INSTANCE_NAME** `_mode_data() `$INSTANCE_NAME`_Pin_A0_Write(HIGH);

Functions

- void $INSTANCE_NAME _reset ()

  Resets the LCD.
- void $INSTANCE_NAME _init ()

  Initialize the LCD.
- void $INSTANCE_NAME _set_contrast (uint8_t value)

  Sets the display contrast.
- void $INSTANCE_NAME _refresh ()

  Instructs that a refresh takes place.
- void $INSTANCE_NAME _refresh_loop ()

  Refresh the display if required. PLACE IN MAIN LOOP.
- int8 $INSTANCE_NAME _is_refreshing ()

  Checks if the display is currently refreshing.
- uint8_t *$INSTANCE_NAME _get_buffer ()

  Returns the display buffer.

2.5.1 Detailed Description

ST7565 Control.

Author

Thanasis Georgiou

2.5.2 Macro Definition Documentation

2.5.2.1 #define CMD_DISPLAY_OFF 0xAE

Turns the display off.

2.5.2.2 #define CMD_DISPLAY_ON 0xAF

Turns the display on.

2.5.2.3 #define CMD_SET_ADC_NORMAL 0xA0

Set ADC to normal mode.

2.5.2.4 #define CMD_SET_ADC_REVERSE 0xA1

Set ADC to reverse mode.

2.5.2.5 #define CMD_SET_COLUMN_LOWER 0x00

Set lower column.

2.5.2.6 #define CMD_SET_COLUMN_UPPER 0x10

Set upper column.

2.5.2.7 #define CMD_SET_DISP_NORMAL 0xA6

Set the display to normal mode.

2.5.2.8 #define CMD_SET_DISP_REVERSE 0xA7

Set the display to reverse mode (useful for flashing).

2.5.2.9 #define CMD_SET_DISP_START_LINE 0x40

Sets the start line.

2.5.2.10 #define CMD_SET_PAGE 0xB0

Selects a page.

2.5.3 Function Documentation

2.5.3.1 uint8_t* $INSTANCE_NAME _get_buffer ( )

Returns the display buffer.

/returns The pointer to the display buffer. The length is 1024 bytes.

2.5.3.2 void $INSTANCE_NAME _init ( )

Initialize the LCD.

Initializes the LCD. In detail:

- Initializes the hardware SPI port and enables interrupts.
- Resets the LCD to a known state.
- Sets the LCD Bias level, selects the ADC and the SHL.
- Turns on the power circuits of the controller.
- Sets the operating voltage and the contrast.
- Empty the memory buffer.
- Turns the display on.

For this display, a 1024 byte buffer is allocated in the PSoC's SRAM. There is no need to do double buffering since a refresh is only carried out when instructed by _refresh().

2.5.3.3 int8 $INSTANCE_NAME _is_refreshing ( )

Checks if the display is currently refreshing.

/returns 1 if the display is refreshing, 0 if not.

2.5.3.4 void $INSTANCE_NAME _refresh ( )

Instructs that a refresh takes place.

Sets the "refresh" flag so the display can start refreshing on the next main loop. Assuming each SPI transfer finishes before the main loop ends it will take 8 + 8 main loops to refresh the whole display.

2.5.3.5 void $INSTANCE_NAME _refresh_loop ( )

Refresh the display if required. PLACE IN MAIN LOOP.

This will refresh the display depanding on if it's required and the last refresh state of the display. It is REQUIRED to place this inside the main loop of your application, differently the display will never refresh.

2.5.3.6 void $INSTANCE_NAME _reset ( )

Resets the LCD.

Resets the LCD by pulling the RST pin LOW for 2uS. This call is takes about 4uS.

Resets the display

2.5.3.7 void $INSTANCE_NAME _set_contrast (uint8_t *value*)

Sets the display contrast.

Sets the display contrast by sending two bytes.

Parameters

value	The new contrast value (0-63). Values outside the range are clipped to 63.

Index