CrossWorks Shield Library

Version: 3.1

Contents

CrossWorks Shield Library

CrossWorks Shield Library

About the CrossWorks Shield Library

The *CrossWorks Shield Library* presents a standardized API for delivering high-quality example code for a wide range of microcontrollers and evaluation boards. Additional components that integrate with the Shield Library are:

- *CrossWorks Platform Library*: is a standardized API to deliver working examples for a wide range of evaluation boards.
- *CrossWorks Tools Library*: provides tools, such as memory tests, and add-ons for CTL, such as a stream system, read-write locks, and ring buffers.
- *CrossWorks Device Library*: provides drivers for common digital sensors, such as accelerometers, gyroscopes, magnetometers, and so on.
- *CrossWorks Graphics Library*: is a library of simple graphics functions for readily-available LCD controllers.
- *CrossWorks TCP/IP Library*: provides TCP/IP networking for integrated and external network controllers on memory-constrained microcontrollers.
- *CrossWorks Mass Storage Library*: provides a FAT-based file system for mass storage on SD and MMC cards, or any device with a block-based interface.
- *CrossWorks Shield Library*: provides drivers for a range of Arduino-style shields.
- *CrossWorks CoreBASIC Library*: provides a full-featured, network-enabled BASIC interpreter which demonstrates the capabilities of these libraries.

Architecture

The *CrossWorks Shield Library* is one part of the *CrossWorks Target Library*. Many of the low-level functions provided by the target library are built using features of the *CrossWorks Tasking Library* for multi-threaded operation.

Delivery format

The *CrossWorks Shield Library* is delivered in source form.

Feedback

This facility is a work in progress and may undergo rapid change. If you have comments, observations, suggestions, or problems, please feel free to air them on the **CrossWorks Target and Platform API** discussion forum.

License

The following terms apply to the Rowley Associates Shield Library.

Introduction

About the CrossWorks Shield Library

The *CrossWorks Shield Library* is a standard API that runs on a collection of popular microprocessors and evaluation boards. It is a way for Rowley Associates to deliver examples, from simple to complex, for those boards.

In particular, the Shield Library requires the *CrossWorks Tasking Library* for operation. Because the Shield Library, and facilities built on top of it, use interrupts and background processing, we made the decision to use the CrossWorks Tasking Library as a foundation stone for the Platform Library. We have not abstracted the Shield Library to use a generic RTOS as this adds more complexity to the design.

Why use the Shield Library?

Standardizing on the Shield Library provised a certain amount of portability for you applications. Rather than using vendor-supplied libraries that get you running quickly on their silicon, you can invest some time learning the Shield Library and use that knowledge across different architectures. You are, however, committing to use CrossWorks, CTL, and the Shield Library for the long term.

What the Shield Library isn't

The Shield Library it is not a general-purpose API supporting every feature offered by common devices, nor does it cater for all devices within a family. The Shield Library is tested on the microprocessors and evaluation boards that Rowley Associates deliver examples for. Certainly, you can use it with little or no modification on boards that have other processors in the families we support, but you will need to customize the Shield Library implementation yourself.

What the Shield Library runs on

The Shield Library runs on the following microprocessor families:

- LPC1700
- LM3S
- KL05Z
- KL25Z
- STM32F1
- STM32F4

The range of boards and microprocessors that run the Shield Library continues to expand. Please check the CrossWorks web site for the latest information.

<adafruit_tft_touch_shield.h>

Overview

Graphics drivers for the Adafruit TFT Touch Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

https://www.adafruit.com/products/376

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ADAFRUIT_TFT_TOUCH_SHIELD_t	Driver data
Shield	
adafruit_tft_touch_fini	Finalize shield
adafruit_tft_touch_init	Initialize shield

ADAFRUIT_TFT_TOUCH_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  short int current_register;
  short int current_data;
  int current_data_valid;
} ADAFRUIT_TFT_TOUCH_SHIELD_t;
```

Description

ADAFRUIT_TFT_TOUCH_SHIELD_t holds the instance data required to run the touch shield.

Structure

`tft_driver`

The driver for the LCD display.

`current_register`

Private data to accelerate LCD operations.

`current_data`

Private data to accelerate LCD operations.

`current_data_valid`

Private data to accelerate LCD operations.

adafruit_tft_touch_fini

Synopsis

```
CTL_STATUS_t adafruit_tft_touch_fini(ADAFRUIT_TFT_TOUCH_SHIELD_t *self);
```

Description

adafruit_tft_touch_fini finalizes the shield **self** and deselects it.

Return Value

adafruit_tft_touch_fini returns a standard status code.

adafruit_tft_touch_init

Synopsis

```
CTL_STATUS_t adafruit_tft_touch_init(ADAFRUIT_TFT_TOUCH_SHIELD_t *self);
```

Description

adafruit_tft_touch_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

adafruit_tft_touch_init returns a standard status code.

<ciseco_led_matrix_shield.h>

Overview

Graphics drivers for the Ciseco LED Matrix Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://shop.ciseco.co.uk/led-matrix-shield-new-r3-uno-style/

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
CISECO_LED_MATRIX_SHIELD_t	Driver data
Shield	
ciseco_led_matrix_shield_fini	Finalize shield
ciseco_led_matrix_shield_init	Initialize shield
ciseco_led_matrix_shield_set_polarity	Set display polarity
ciseco_led_matrix_shield_set_scan_frequency	Set display scan line frequency

CISECO_LED_MATRIX_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_DRIVER_t core;
  unsigned char frame[];
  int row;
  int frequency;
  int polarity;
  PLATFORM_HOOK_t hook;
} CISECO_LED_MATRIX_SHIELD_t;
```

Description

CISECO_LED_MATRIX_SHIELD_t holds the instance data required to run the touch shield.

Structure

`core`

Core graphics driver.

`frame`

The 8x8 frame buffer.

`row`

Private data that holds the current scan row.

`frequency`

Private data that holds the scan line frequency.

`polarity`

Private data that holds the display polarity (white on black, black on white).

`hook`

Private data that hooks the high frequency timer to refresh the display.

ciseco_led_matrix_shield_fini

Synopsis

```
CTL_STATUS_t ciseco_led_matrix_shield_fini(CISECO_LED_MATRIX_SHIELD_t *self);
```

Description

ciseco_led_matrix_shield_fini finalizes the shield **self** and deselects it.

Return Value

ciseco_led_matrix_shield_fini returns a standard status code.

ciseco_led_matrix_shield_init

Synopsis

```
CTL_STATUS_t ciseco_led_matrix_shield_init(CISECO_LED_MATRIX_SHIELD_t *self);
```

Description

ciseco_led_matrix_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

ciseco_led_matrix_shield_init returns a standard status code.

ciseco_led_matrix_shield_set_polarity

Synopsis

```
CTL_STATUS_t ciseco_led_matrix_shield_set_polarity(CISECO_LED_MATRIX_SHIELD_t *self,
                                                   int polarity);
```

Description

ciseco_led_matrix_shield_set_polarity sets the display polarity of the shield **self** to **polarity**. If **polarity** is zero, non-zero pixels illuminate a LED and provide a white-on-black display, and if **polarity** is non-zero, zero pixels illuminate a LED and provide a black-on-white display.

Return Value

ciseco_led_matrix_shield_set_polarity returns a standard status code.

ciseco_led_matrix_shield_set_scan_frequency

Synopsis

```
CTL_STATUS_t ciseco_led_matrix_shield_set_scan_frequency(CISECO_LED_MATRIX_SHIELD_t *self,
                                                         int hz);
```

Description

ciseco_led_matrix_shield_set_scan_frequency sets the scan line frequency of the shield **self** to **hz** Hertz.

Return Value

ciseco_led_matrix_shield_set_scan_frequency returns a standard status code.

<itead_studio_colors_shield.h>

Overview

Graphics drivers for the Itead Studio Colors Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://imall.iteadstudio.com/development-platform/arduino/shields/im120417002.html

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_COLORS_SHIELD_t	Driver data
Shield	
itead_studio_colors_shield_fini	Finalize shield
itead_studio_colors_shield_init	Initialize shield
itead_studio_colors_shield_set_scan_frequency	Set display scan line frequency

ITEAD_STUDIO_COLORS_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_DRIVER_t core;
  unsigned long frame[][];
  unsigned short row;
  unsigned frequency;
  PLATFORM_HOOK_t hook;
} ITEAD_STUDIO_COLORS_SHIELD_t;
```

Description

ITEAD_STUDIO_COLORS_SHIELD_t holds the instance data required to run the touch shield.

Structure

`core`

Core graphics driver.

`frame`

The 8x8 frame buffer.

`row`

Private data that holds the current scan row.

`frequency`

Private data that holds the scan line frequency.

`hook`

Private data that hooks the high frequency timer to refresh the display.

itead_studio_colors_shield_fini

Synopsis

```
CTL_STATUS_t itead_studio_colors_shield_fini(ITEAD_STUDIO_COLORS_SHIELD_t *self);
```

Description

itead_studio_colors_shield_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_colors_shield_fini returns a standard status code.

itead_studio_colors_shield_init

Synopsis

```
CTL_STATUS_t itead_studio_colors_shield_init(ITEAD_STUDIO_COLORS_SHIELD_t *self);
```

Description

itead_studio_colors_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_colors_shield_init returns a standard status code.

itead_studio_colors_shield_set_scan_frequency

Synopsis

```
CTL_STATUS_t itead_studio_colors_shield_set_scan_frequency(ITEAD_STUDIO_COLORS_SHIELD_t *self,
                                                           int hz);
```

Description

itead_studio_colors_shield_set_scan_frequency sets the scan line frequency of the shield **self** to **hz** Hertz.

Return Value

itead_studio_colors_shield_set_scan_frequency returns a standard status code.

<itead_studio_ibridge_lcd.h>

Overview

Graphics drivers for the Itead IBridge or IBridge Lite with the optional LCD fitted.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://imall.iteadstudio.com/development-platform/arduino/shields/im120417001.html
http://imall.iteadstudio.com/development-platform/arduino/shields/im120417023.html

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_IBRIDGE_LCD_t	Driver data
Shield	
itead_studio_ibridge_lcd_fini	Finalize shield
itead_studio_ibridge_lcd_init	Initialize shield

ITEAD_STUDIO_IBRIDGE_LCD_t

Synopsis

```
typedef struct {
  CTL_SPI_DEVICE_t device;
  PCD8544_DRIVER_t driver;
  PLATFORM_HOOK_t hook;
  SOFTWARE_SPI_BUS_t bus;
} ITEAD_STUDIO_IBRIDGE_LCD_t;
```

Description

ITEAD_STUDIO_IBRIDGE_LCD_t holds the instance data required to run the IBridge display.

Structure

device

The SPI device instance that communicates with the Epson LCD.

driver

The PCD8544 graphics device driver.

hook

Private data that hooks the background timer to refresh the display.

bus

A software SPI bus because the IBridge does not conform to the standard SPI pinning of the Arduino.

itead_studio_ibridge_lcd_fini

Synopsis

```
CTL_STATUS_t itead_studio_ibridge_lcd_fini(ITEAD_STUDIO_IBRIDGE_LCD_t *self);
```

Description

itead_studio_ibridge_lcd_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_ibridge_lcd_fini returns a standard status code.

itead_studio_ibridge_lcd_init

Synopsis

```
CTL_STATUS_t itead_studio_ibridge_lcd_init(ITEAD_STUDIO_IBRIDGE_LCD_t *self);
```

Description

itead_studio_ibridge_lcd_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_ibridge_lcd_init returns a standard status code.

<itead_studio_itdb02_2v2.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-2.2 LCD module.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=149 — shield
http://iteadstudio.com/store/index.php?main_page=product_info&products_id=529 — 2.2" LCD module

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_2v2_t	Driver data
Shield	
itead_studio_itdb02_2v2_fini	Finalize shield
itead_studio_itdb02_2v2_init	Initialize shield

ITEAD_STUDIO_ITDB02_2v2_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  int last_register;
} ITEAD_STUDIO_ITDB02_2v2_t;
```

Description

ITEAD_STUDIO_ITDB02_2v2_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the HX8340B.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`last_register`

Private data for acceleration.

itead_studio_itdb02_2v2_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v2_fini(ITEAD_STUDIO_ITDB02_2v2_t *self);
```

Description

itead_studio_itdb02_2v2_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_2v2_fini returns a standard status code.

itead_studio_itdb02_2v2_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v2_init(ITEAD_STUDIO_ITDB02_2v2_t *self);
```

Description

itead_studio_itdb02_2v2_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_2v2_init returns a standard status code.

<itead_studio_itdb02_2v4d.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-2.4D LCD module.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=149 — shield
http://iteadstudio.com/store/index.php?main_page=product_info&products_id=55 — 2.4" LCD module

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_2v4D_t	Driver data
Shield	
itead_studio_itdb02_2v4d_fini	Finalize shield
itead_studio_itdb02_2v4d_init	Initialize shield

ITEAD_STUDIO_ITDB02_2v4D_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  short int current_register;
} ITEAD_STUDIO_ITDB02_2v4D_t;
```

Description

ITEAD_STUDIO_ITDB02_2v4D_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the ILI9325.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`current_register`

Private data for acceleration.

itead_studio_itdb02_2v4d_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v4d_fini(ITEAD_STUDIO_ITDB02_2v4D_t *self);
```

Description

itead_studio_itdb02_2v4d_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_2v4d_fini returns a standard status code.

itead_studio_itdb02_2v4d_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v4d_init(ITEAD_STUDIO_ITDB02_2v4D_t *self);
```

Description

itead_studio_itdb02_2v4d_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_2v4d_init returns a standard status code.

<itead_studio_itdb02_2v4e.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-2.4E LCD module.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=149 — ITDB02 shield
http://iteadstudio.com/store/index.php?main_page=product_info&products_id=55 — 2.4" LCD module
http://iteadstudio.com/store/index.php?main_page=product_info&products_id=473 — 2.4" LCD module and shield combined

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_2v4E_t	Driver data
Shield	
itead_studio_itdb02_2v4e_fini	Finalize shield
itead_studio_itdb02_2v4e_init	Initialize shield

ITEAD_STUDIO_ITDB02_2v4E_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  unsigned last_register;
} ITEAD_STUDIO_ITDB02_2v4E_t;
```

Description

ITEAD_STUDIO_ITDB02_2v4E_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the S6D1121.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`last_register`

Private data for acceleration.

itead_studio_itdb02_2v4e_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v4e_fini(ITEAD_STUDIO_ITDB02_2v4E_t *self);
```

Description

itead_studio_itdb02_2v4e_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_2v4e_fini returns a standard status code.

itead_studio_itdb02_2v4e_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v4e_init(ITEAD_STUDIO_ITDB02_2v4E_t *self);
```

Description

itead_studio_itdb02_2v4e_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_2v4e_init returns a standard status code.

<itead_studio_itdb02_2v8.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-2.8 LCD module.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=149 — shield
http://iteadstudio.com/store/index.php?main_page=product_info&products_id=530 — 2.8" LCD module

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_2v8_t	Driver data
Shield	
itead_studio_itdb02_2v8_fini	Finalize shield
itead_studio_itdb02_2v8_init	Initialize shield

ITEAD_STUDIO_ITDB02_2v8_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  unsigned short current_register;
} ITEAD_STUDIO_ITDB02_2v8_t;
```

Description

ITEAD_STUDIO_ITDB02_2v8_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the ILI9325.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`current_register`

Private data for acceleration.

itead_studio_itdb02_2v8_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v8_fini(ITEAD_STUDIO_ITDB02_2v8_t *self);
```

Description

itead_studio_itdb02_2v8_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_2v8_fini returns a standard status code.

itead_studio_itdb02_2v8_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_2v8_init(ITEAD_STUDIO_ITDB02_2v8_t *self);
```

Description

itead_studio_itdb02_2v8_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_2v8_init returns a standard status code.

<itead_studio_itdb02_3v2s.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-3.2S LCD module.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=149 — shield
http://iteadstudio.com/store/index.php?main_page=product_info&products_id=54 — 3.2" LCD module

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_3v2S_t	Driver data
Shield	
itead_studio_itdb02_3v2s_fini	Finalize shield
itead_studio_itdb02_3v2s_init	Initialize shield

ITEAD_STUDIO_ITDB02_3v2S_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  unsigned short current_register;
} ITEAD_STUDIO_ITDB02_3v2S_t;
```

Description

ITEAD_STUDIO_ITDB02_3v2S_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the SSD1289.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`current_register`

Private data for acceleration.

itead_studio_itdb02_3v2s_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_3v2s_fini(ITEAD_STUDIO_ITDB02_3v2S_t *self);
```

Description

itead_studio_itdb02_3v2s_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_3v2s_fini returns a standard status code.

itead_studio_itdb02_3v2s_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_3v2s_init(ITEAD_STUDIO_ITDB02_3v2S_t *self);
```

Description

itead_studio_itdb02_3v2s_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_3v2s_init returns a standard status code.

<itead_studio_itdb02_3v2wd.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-3.2WD LCD module.

Setup

Using the ITDB02-3.2WD LCD module requires proper configuration of the ITDB02 shield to ensure success. The LCD only functions in 16-bit mode, so all shorting blocks on the ITDB02 must be moved to the `LCD_16bit` side.

Unfortunately, driving the LCD in 16-bit mode utilizes all the pins on the Arduino footprint, leaving nothing free. A disappointing consequence is that you cannot use the SD card interface, nor can you use the touch screen controller.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

This is the ITDB02 shield; currently only the v2 shield is offered for sale and is not compatible with the 3.2WD and SolderCore.

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=149

This is the display module, ITead Studio part number DIS012:

http://iteadstudio.com/store/index.php?main_page=product_info&products_id=263

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_3v2WD_t	Driver data
Shield	
itead_studio_itdb02_3v2wd_fini	Finalize shield
itead_studio_itdb02_3v2wd_init	Initialize shield

ITEAD_STUDIO_ITDB02_3v2WD_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  int current_register;
} ITEAD_STUDIO_ITDB02_3v2WD_t;
```

Description

ITEAD_STUDIO_ITDB02_3v2WD_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the HX8352A.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`current_register`

Private data for acceleration.

itead_studio_itdb02_3v2wd_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_3v2wd_fini(ITEAD_STUDIO_ITDB02_3v2WD_t *self);
```

Description

itead_studio_itdb02_3v2wd_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_3v2wd_fini returns a standard status code.

itead_studio_itdb02_3v2wd_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_3v2wd_init(ITEAD_STUDIO_ITDB02_3v2WD_t *self);
```

Description

itead_studio_itdb02_3v2wd_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_3v2wd_init returns a standard status code.

<itead_studio_itdb02_4v3.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-4.3 LCD module.

Setup

Using the ITDB02-4.3 is unreliable with the ITDB02 v1.2 shield and is incompatible with the ITDB02 v2 shield, so you need to connect the ITDB02-4.3 module to the SolderCore headers directly.

Unfortunately, driving the LCD in 16-bit mode utilizes all the pins on the SolderCore, leaving nothing free. A disappointing consequence is that you cannot use the SD card interface, nor can you use the touch screen controller.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.itead-europe.com/index.php/display/tft-lcm/itdb02-4-3.html

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_4v3_t	Driver data
Shield	
itead_studio_itdb02_4v3_fini	Finalize shield
itead_studio_itdb02_4v3_init	Initialize shield

ITEAD_STUDIO_ITDB02_4v3_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  short int current_register;
} ITEAD_STUDIO_ITDB02_4v3_t;
```

Description

ITEAD_STUDIO_ITDB02_4v3_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the SSD1963.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`current_register`

Private data for acceleration.

itead_studio_itdb02_4v3_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_4v3_fini(ITEAD_STUDIO_ITDB02_4v3_t *self);
```

Description

itead_studio_itdb02_4v3_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_4v3_fini returns a standard status code.

itead_studio_itdb02_4v3_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_4v3_init(ITEAD_STUDIO_ITDB02_4v3_t *self);
```

Description

itead_studio_itdb02_4v3_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_4v3_init returns a standard status code.

<itead_studio_itdb02_5v0.h>

Overview

Drivers for an ITead Studio IDB02 with an ITDB02-5.0 LCD module.

Setup

Using the ITDB02-5.0 is unreliable with the ITDB02 v1.2 shield and is incompatible with the ITDB02 v2 shield, so you need to connect the ITDB02-5.0 module to the SolderCore headers directly.

Unfortunately, driving the LCD in 16-bit mode utilizes all the pins on the SolderCore, leaving nothing free. A disappointing consequence is that you cannot use the SD card interface, nor can you use the touch screen controller.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.itead-europe.com/index.php/display/tft-lcm/itdb02-5-0.html

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
ITEAD_STUDIO_ITDB02_5v0_t	Driver data
Shield	
itead_studio_itdb02_5v0_fini	Finalize shield
itead_studio_itdb02_5v0_init	Initialize shield

ITEAD_STUDIO_ITDB02_5v0_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  ITDB02_BUS_STATE_t bus_state;
  short int current_register;
} ITEAD_STUDIO_ITDB02_5v0_t;
```

Description

ITEAD_STUDIO_ITDB02_5v0_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the SSD1963.

`bus_state`

The bus driver for the ITDB02 with attached graphic display.

`current_register`

Private data for acceleration.

itead_studio_itdb02_5v0_fini

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_5v0_fini(ITEAD_STUDIO_ITDB02_5v0_t *self);
```

Description

itead_studio_itdb02_5v0_fini finalizes the shield **self** and deselects it.

Return Value

itead_studio_itdb02_5v0_fini returns a standard status code.

itead_studio_itdb02_5v0_init

Synopsis

```
CTL_STATUS_t itead_studio_itdb02_5v0_init(ITEAD_STUDIO_ITDB02_5v0_t *self);
```

Description

itead_studio_itdb02_5v0_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

itead_studio_itdb02_5v0_init returns a standard status code.

<itead_studio_itdb02_driver.h>

Overview

Base 8-bit and 16-bit bus driver for ITDB02 v1 and v2 shields.

API Summary

Instance	
ITDB02_BUS_STATE_t	Current bus state
Configuration	
itead_studio_itdb02_16bit_mode_shield_pinout	Pinout for 16-bit mode
itead_studio_itdb02_8bit_mode_shield_pinout	Pinout for 8-bit mode
Setup	
itead_studio_itdb02_init	Finalize shield
I/O	
itead_studio_itdb02_write_16b_bus	Finalize shield
itead_studio_itdb02_write_16b_to_8b_bus	Finalize shield
itead_studio_itdb02_write_8b_bus	Finalize shield

ITDB02_BUS_STATE_t

Synopsis

```
typedef struct {
  unsigned current_data;
  int invalid;
} ITDB02_BUS_STATE_t;
```

Description

ITDB02_BUS_STATE_t holds the instance data required to run the bus.

Structure

`current_data`

Current data driven to the bus.

`invalid`

Private data that indicates validity of bus contents.

itead_studio_itdb02_16bit_mode_shield_pinout

Synopsis

```
PLATFORM_PIN_CONFIGURATION_t itead_studio_itdb02_16bit_mode_shield_pinout[];
```

Description

itead_studio_itdb02_16bit_mode_shield_pinout defines the pinout for LCDs attached in 16-bit mode.

itead_studio_itdb02_8bit_mode_shield_pinout

Synopsis

```
PLATFORM_PIN_CONFIGURATION_t itead_studio_itdb02_8bit_mode_shield_pinout[];
```

Description

itead_studio_itdb02_8bit_mode_shield_pinout defines the pinout for LCDs attached in 16-bit mode.

itead_studio_itdb02_init

Synopsis

```
void itead_studio_itdb02_init(ITDB02_BUS_STATE_t *self);
```

Description

itead_studio_itdb02_init initializes the bus **self**.

itead_studio_itdb02_write_16b_bus

Synopsis

```
void itead_studio_itdb02_write_16b_bus(ITDB02_BUS_STATE_t *self,
                                       int data);
```

Description

itead_studio_itdb02_write_16b_bus writes 16 bits of **data** to the 16-bit ITDB02 bus **self**.

itead_studio_itdb02_write_16b_to_8b_bus

Synopsis

```
void itead_studio_itdb02_write_16b_to_8b_bus(ITDB02_BUS_STATE_t *self,
                                             int data);
```

Description

itead_studio_itdb02_write_16b_to_8b_bus writes 16 bits of **data** to the 16-bit ITDB02 bus **self**. The bus may be implemented as a single 16-bit parallel bus or as an 8-bit bus that requires two 8-bit write cycles.

itead_studio_itdb02_write_8b_bus

Synopsis

```
void itead_studio_itdb02_write_8b_bus(ITDB02_BUS_STATE_t *self,
                                      int data);
```

Description

itead_studio_itdb02_write_8b_bus writes 8 bit of **data** to the 8-bit ITDB02 bus **self**.

<jimmie_rodgers_lol_shield.h>

Overview

Graphics drivers for the Jimmie Rodgers LoL Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://jimmieprodgers.com/kits/lolshield/

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
JIMMIE_RODGERS_LOL_SHIELD_t	Driver data
Shield	
jimmie_rodgers_lol_shield_fini	Finalize shield
jimmie_rodgers_lol_shield_init	Initialize shield
jimmie_rodgers_lol_shield_set_polarity	Set display polarity
jimmie_rodgers_lol_shield_set_scan_frequency	Set display scan line frequency
jimmie_rodgers_lol_shield_write_column	Set entire column
jimmie_rodgers_lol_shield_write_row	Set entire row

JIMMIE_RODGERS_LOL_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_DRIVER_t core;
  PLATFORM_HOOK_t hook;
  unsigned short frequency;
  unsigned short polarity;
  unsigned short cycle;
  unsigned short bitmap[];
  unsigned short map[];
} JIMMIE_RODGERS_LOL_SHIELD_t;
```

Description

JIMMIE_RODGERS_LOL_SHIELD_t holds the instance data required to run the touch shield.

Structure

core

Core graphics driver.

hook

Private data that hooks the high frequency timer to refresh the display.

frequency

Private data that holds the scan line frequency.

polarity

Private data that holds the display polarity (white on black, black on white).

cycle

Private data that holds the Charlieplexing cycle.

bitmap

The plain 9x16 frame buffer.

map

Private Charlieplexing drive map.

jimmie_rodgers_lol_shield_fini

Synopsis

```
CTL_STATUS_t jimmie_rodgers_lol_shield_fini(JIMMIE_RODGERS_LOL_SHIELD_t *self);
```

Description

jimmie_rodgers_lol_shield_fini finalizes the shield **self** and deselects it.

Return Value

jimmie_rodgers_lol_shield_fini returns a standard status code.

jimmie_rodgers_lol_shield_init

Synopsis

```
CTL_STATUS_t jimmie_rodgers_lol_shield_init(JIMMIE_RODGERS_LOL_SHIELD_t *self);
```

Description

jimmie_rodgers_lol_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

jimmie_rodgers_lol_shield_init returns a standard status code.

jimmie_rodgers_lol_shield_set_polarity

Synopsis

```
CTL_STATUS_t jimmie_rodgers_lol_shield_set_polarity(JIMMIE_RODGERS_LOL_SHIELD_t *self,
                                                    int polarity);
```

Description

jimmie_rodgers_lol_shield_set_polarity sets the display polarity of the shield **self** to **polarity**. If **polarity** is zero, non-zero pixels illuminate a LED and provide a white-on-black display, and if **polarity** is non-zero, zero pixels illuminate a LED and provide a black-on-white display.

Return Value

jimmie_rodgers_lol_shield_set_polarity returns a standard status code.

jimmie_rodgers_lol_shield_set_scan_frequency

Synopsis

```
CTL_STATUS_t jimmie_rodgers_lol_shield_set_scan_frequency(JIMMIE_RODGERS_LOL_SHIELD_t *self,
                                                          int hz);
```

Description

jimmie_rodgers_lol_shield_set_scan_frequency sets the scan line frequency of the shield **self** to **hz** Hertz.

Return Value

jimmie_rodgers_lol_shield_set_scan_frequency returns a standard status code.

jimmie_rodgers_lol_shield_write_column

Synopsis

```
void jimmie_rodgers_lol_shield_write_column(JIMMIE_RODGERS_LOL_SHIELD_t *self,
                                            int col,
                                            int bits);
```

Description

jimmie_rodgers_lol_shield_write_column sets the column **col** to hold the data **bits** where each bit corresponds to the color of a single row.

jimmie_rodgers_lol_shield_write_row

Synopsis

```
void jimmie_rodgers_lol_shield_write_row(JIMMIE_RODGERS_LOL_SHIELD_t *self,
                                         int row,
                                         int bits);
```

Description

jimmie_rodgers_lol_shield_write_row sets the row **row** to hold the data **bits** where each bit corresponds to the color of a single column.

<nuelectronics_3310_lcd_shield.h>

Overview

Drivers for the NuElectronics 3310 LCD Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.nuelectronics.com/estore/index.php?main_page=product_info&products_id=12

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
NUELECTRONICS_3310_LCD_SHIELD_t	Driver data
Shield	
nuelectronics_3310_lcd_shield_fini	Finalize shield
nuelectronics_3310_lcd_shield_init	Initialize shield

NUELECTRONICS_3310_LCD_SHIELD_t

Synopsis

```
typedef struct {
  CTL_SPI_DEVICE_t device;
  PCD8544_DRIVER_t driver;
  PLATFORM_HOOK_t hook;
} NUELECTRONICS_3310_LCD_SHIELD_t;
```

Description

NUELECTRONICS_3310_LCD_SHIELD_t holds the instance data required to run the IBridge display.

Structure

device

The SPI device instance that communicates with the Epson LCD.

driver

The PCD8544 graphics device driver.

hook

Private data that hooks the background timer to refresh the display.

nuelectronics_3310_lcd_shield_fini

Synopsis

```
CTL_STATUS_t nuelectronics_3310_lcd_shield_fini(NUELECTRONICS_3310_LCD_SHIELD_t *self);
```

Description

nuelectronics_3310_lcd_shield_fini finalizes the shield **self** and deselects it.

Return Value

nuelectronics_3310_lcd_shield_fini returns a standard status code.

nuelectronics_3310_lcd_shield_init

Synopsis

```
CTL_STATUS_t nuelectronics_3310_lcd_shield_init(NUELECTRONICS_3310_LCD_SHIELD_t *self);
```

Description

nuelectronics_3310_lcd_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

nuelectronics_3310_lcd_shield_init returns a standard status code.

<seeed_studio_tft_touch_shield.h>

Overview

Drivers for the Seeed Studio TFT Touch Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.seeedstudio.com/depot/28-tft-touch-shield-p-864.html

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
SEEED_STUDIO_TFT_TOUCH_SHIELD_t	Driver data
Shield	
seeed_studio_tft_touch_fini	Finalize shield
seeed_studio_tft_touch_init	Initialize shield

SEEED_STUDIO_TFT_TOUCH_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t tft_driver;
  short int current_register;
  short int current_data;
  unsigned char current_data_valid;
} SEEED_STUDIO_TFT_TOUCH_SHIELD_t;
```

Description

SEEED_STUDIO_TFT_TOUCH_SHIELD_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the ILI9325.

`current_register`

Private data that contains the selected register index.

`current_data`

Private data that contains the data driven to the bus.

`current_data_valid`

Private flag that indicates whether the **current_data** member is valid.

seeed_studio_tft_touch_fini

Synopsis

```
CTL_STATUS_t seeed_studio_tft_touch_fini(SEEED_STUDIO_TFT_TOUCH_SHIELD_t *self);
```

Description

seeed_studio_tft_touch_fini finalizes the shield **self** and deselects it.

Return Value

seeed_studio_tft_touch_fini returns a standard status code.

seeed_studio_tft_touch_init

Synopsis

```
CTL_STATUS_t seeed_studio_tft_touch_init(SEEED_STUDIO_TFT_TOUCH_SHIELD_t *self);
```

Description

seeed_studio_tft_touch_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

seeed_studio_tft_touch_init returns a standard status code.

<soldercore_graphics_shield.h>

Overview

Drivers for the SolderCore LCD and Arcade Shields.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.soldercore.com/products/lcd-shield
http://www.soldercore.com/products/arcade-shield

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
SOLDERCORE_GRAPHICS_SHIELD_t	Driver data
Shield	
soldercore_arcade_shield_init	Initialize Arcade shield
soldercore_graphics_shield_fini	Finalize Arcade or LCD shield
soldercore_graphics_shield_init	Initialize Arcade or LCD shield
soldercore_lcd_shield_init	Initialize LCD shield
I/O	
soldercore_graphics_shield_set_clut_entry	Set CLUT entry
soldercore_graphics_shield_spi_write	Write to graphics shield, respecting FIFO constraints

SOLDERCORE_GRAPHICS_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_DRIVER_t tft_driver;
  CTL_SPI_DEVICE_t device;
  unsigned pipline_known_free;
  unsigned long model;
  int revision;
} SOLDERCORE_GRAPHICS_SHIELD_t;
```

Description

SOLDERCORE_GRAPHICS_SHIELD_t holds the instance data required to run the display.

Structure

`tft_driver`

The graphics driver for the selected shield.

`device`

The SPI device for communicating with the shield.

`pipline_known_free`

Private data that counts the minimum number of empty slots in the graphics pipeline.

`model`

Private data that indicates the type of shield.

`revision`

Private data that indicates the firmware revision of the shield.

soldercore_arcade_shield_init

Synopsis

```
CTL_STATUS_t soldercore_arcade_shield_init(SOLDERCORE_GRAPHICS_SHIELD_t *self);
```

Description

soldercore_arcade_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

soldercore_arcade_shield_init returns a standard status code.

soldercore_graphics_shield_fini

Synopsis

```
CTL_STATUS_t soldercore_graphics_shield_fini(SOLDERCORE_GRAPHICS_SHIELD_t *self);
```

Description

soldercore_graphics_shield_fini finalizes the shield **self** and deselects it.

Return Value

soldercore_graphics_shield_fini returns a standard status code.

soldercore_graphics_shield_init

Synopsis

```
CTL_STATUS_t soldercore_graphics_shield_init(SOLDERCORE_GRAPHICS_SHIELD_t *self);
```

Description

soldercore_graphics_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

soldercore_graphics_shield_init returns a standard status code.

soldercore_graphics_shield_set_clut_entry

Synopsis

```
void soldercore_graphics_shield_set_clut_entry(SOLDERCORE_GRAPHICS_SHIELD_t *self,
                                               int index,
                                               int rgb);
```

Description

soldercore_graphics_shield_set_clut_entry sets the color lookup table entry **index** (0 through 15) to the RGB value **rgb**.

soldercore_graphics_shield_spi_write

Synopsis

```
void soldercore_graphics_shield_spi_write(CTL_SPI_DEVICE_t *self,
                                          const void *data,
                                          size_t len);
```

Description

soldercore_graphics_shield_spi_write write **len** bytes from **data** to the shield **self**. This respects the readiness of the shield to accept more data by using the RDY signal and the pipeline state.

soldercore_lcd_shield_init

Synopsis

```
CTL_STATUS_t soldercore_lcd_shield_init(SOLDERCORE_GRAPHICS_SHIELD_t *self);
```

Description

soldercore_lcd_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

soldercore_lcd_shield_init returns a standard status code.

<soldercore_sensecore.h>

Overview

Driver for a SenseCore shield.

The SenseCore provides four SExI sites which you can populate with *rivets*. Each site is divided into two halves, an SPI half and an I2C half.

The PCA9655 port expander controls the chip selects to each of the SPI sites, the digital I/O signals to each of the sites, and can optionally control the *Tx* and *Rx* signals.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.soldercore.com/

Test hardware

This code has been tested using a SolderCore and a SenseCore.

API Summary

Types	
SENSECORE_BUS_SIGNAL_t	SenseCore I/O expander signals
SENSECORE_DRIVER_t	SenseCore state
Initialization	
sensecore_init	Initialize SenseCore on I2C bus
sensecore_init_ex	Initialize SenseCore driver (extended)
Functions	
sensecore_get_dio_signal	Get the DIO signal for a single SExI site
sensecore_get_rx_signal	Get the Rx signal for a single SExI site
sensecore_get_tx_signal	Get the Tx signal for a single SExI site
sensecore_select_site	Select a single SExI site
sensecore_set_cs_signal	Set the CS signal for a single SExI site
sensecore_set_dio_signal	Set the DIO signal for a single SExI site
sensecore_set_rx_signal	Set the Rx signal for a single SExI site
sensecore_set_tx_signal	Set the Tx signal for a single SExI site

SENSECORE_BUS_SIGNAL_t

Synopsis

```
typedef enum {
  SENSECORE_DIOA,
  SENSECORE_DIOB,
  SENSECORE_DIOC,
  SENSECORE_DIOD,
  SENSECORE_TXA,
  SENSECORE_TXB,
  SENSECORE_TXC,
  SENSECORE_TXD,
  SENSECORE_CSA,
  SENSECORE_CSB,
  SENSECORE_CSC,
  SENSECORE_CSD,
  SENSECORE_RXA,
  SENSECORE_RXB,
  SENSECORE_RXC,
  SENSECORE_RXD
} SENSECORE_BUS_SIGNAL_t;
```

Description

SENSECORE_BUS_SIGNAL_t enumerates the sixteen signals that the I/O expander provides.

SENSECORE_DRIVER_t

Synopsis

```
typedef struct {
  CTL_PARALLEL_BUS_t expander;
} SENSECORE_DRIVER_t;
```

Description

SENSECORE_DRIVER_t contains the state necessary to control the SenseCore through this API. These fields should be considered private but are exposed for full flexibility.

sensecore_get_dio_signal

Synopsis

```
CTL_STATUS_t sensecore_get_dio_signal(SENSECORE_DRIVER_t *self,
                                      int site);
```

Description

sensecore_get_dio_signal reads the DIO signal for the SenseCore **self** on site **site**. The port pin associated with the DIO signal is set to input mode before reading.

Return Value

sensecore_get_dio_signal returns an extended status code: `0` if the DIO signal is low, `1` if the DIO signal is high, and a negative value indicating an error.

Return Value

sensecore_get_dio_signal returns a standard status code.

Thread Safety

sensecore_get_dio_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_get_rx_signal

Synopsis

```
CTL_STATUS_t sensecore_get_rx_signal(SENSECORE_DRIVER_t *self,
                                     int site);
```

Description

sensecore_get_rx_signal reads the Rx signal for the SenseCore **self** on site **site**. The port pin associated with the Rx signal is set to input mode before reading.

Return Value

sensecore_get_rx_signal returns an extended status code: `0` if the Rx signal is low, `1` if the Rx signal is high, and a negative value indicating an error.

Thread Safety

sensecore_get_rx_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_get_tx_signal

Synopsis

```
CTL_STATUS_t sensecore_get_tx_signal(SENSECORE_DRIVER_t *self,
                                     int site);
```

Description

sensecore_get_tx_signal reads the Tx signal for the SenseCore **self** on site **site**. The port pin associated with the Tx signal is set to input mode before reading.

Return Value

sensecore_get_tx_signal returns an extended status code: `0` if the Tx signal is low, `1` if the Tx signal is high, and a negative value indicating an error.

Thread Safety

sensecore_get_tx_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_init

Synopsis

```
CTL_STATUS_t sensecore_init(SENSECORE_DRIVER_t *self,
                            CTL_I2C_BUS_t *bus);
```

Description

sensecore_init initializes the SenseCore on I2C bus **bus** with the on-board PCA9655 port expander configured to the default 8-bit I2C address AD=001.

Return Value

sensecore_init returns a standard status code.

Thread Safety

sensecore_init is thread-safe if a mutex is associated with the I2C bus **bus**.

sensecore_init_ex

Synopsis

```
CTL_STATUS_t sensecore_init_ex(SENSECORE_DRIVER_t *self,
                               CTL_I2C_BUS_t *bus,
                               int addr,
                               CTL_STATUS_t bus_init *self,
                               CTL_I2C_BUS_t *bus,
                               int addr);
```

Description

sensecore_init_ex initializes the SenseCore on I2C bus **bus** using the port expander configured to the 8-bit I2C address **addr** which is initialized by **bus_init**.

Return Value

sensecore_init_ex returns a standard status code.

Thread Safety

sensecore_init_ex is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_select_site

Synopsis

```
CTL_STATUS_t sensecore_select_site(SENSECORE_DRIVER_t *self,
                                   int site);
```

Description

sensecore_select_site selects the single SenseCore site **site** and ensures that all other sites are deselected.

Return Value

sensecore_select_site returns a standard status code.

Thread Safety

sensecore_select_site is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_set_cs_signal

Synopsis

```
CTL_STATUS_t sensecore_set_cs_signal(SENSECORE_DRIVER_t *self,
                                     int site,
                                     int state);
```

Description

sensecore_set_cs_signal sets the SenseCore CS (site-select) signal for site **site** to **state**. If **state** is zero, the CS signal is pulled low and if **state** is non-zero the CS signal is pulled high.

Note that the SenseCore driver does not ensure that a single site is uniquely selected: it is the client's responsibility to ensure proper selection and deselection of each of the sites or to use **sensecore_select_site**.

Return Value

sensecore_set_cs_signal returns a standard status code.

Thread Safety

sensecore_set_cs_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_set_dio_signal

Synopsis

```
CTL_STATUS_t sensecore_set_dio_signal(SENSECORE_DRIVER_t *self,
                                      int site,
                                      int state);
```

Description

sensecore_set_dio_signal sets the DIO signal for the SenseCore **self** on site **site** to **state**. If **state** is zero, the DIO signal is pulled low and if **state** is non-zero the DIO signal is pulled high.

Return Value

sensecore_set_dio_signal returns a standard status code.

Thread Safety

sensecore_set_dio_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_set_rx_signal

Synopsis

```
CTL_STATUS_t sensecore_set_rx_signal(SENSECORE_DRIVER_t *self,
                                     int site,
                                     int state);
```

Description

sensecore_set_rx_signal sets the Rx signal for the SenseCore **self** on site **site** to **state**. If **state** is zero, the Rx signal is pulled low and if **state** is non-zero the Rx signal is pulled high.

Return Value

sensecore_set_rx_signal returns a standard status code.

Thread Safety

sensecore_set_rx_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

sensecore_set_tx_signal

Synopsis

```
CTL_STATUS_t sensecore_set_tx_signal(SENSECORE_DRIVER_t *self,
                                     int site,
                                     int state);
```

Description

sensecore_set_tx_signal sets the Tx signal for the SenseCore **self** on site **site** to **state**. If **state** is zero, the Tx signal is pulled low and if **state** is non-zero the Tx signal is pulled high.

Return Value

sensecore_set_tx_signal returns a standard status code.

Thread Safety

sensecore_set_tx_signal is thread-safe if the **transport** member in **self** is thread-safe. This requires that a mutex is associated with the I2C bus provided when initializing **self**.

<sparkfun_color_lcd_shield.h>

Overview

Drivers for the SparkFun Color LCD Shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.sparkfun.com/products/9363

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
SPARKFUN_COLOR_LCD_SHIELD_t	Driver data
Shield	
sparkfun_color_lcd_shield_fini	Finalize shield
sparkfun_color_lcd_shield_init	Initialize shield

SPARKFUN_COLOR_LCD_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t driver;
  CTL_SPI_DEVICE_t device;
} SPARKFUN_COLOR_LCD_SHIELD_t;
```

Description

SPARKFUN_COLOR_LCD_SHIELD_t holds the instance data required to run the display.

Structure

`driver`

The graphics driver for the display.

`device`

The SPI device used to communicate with the LCD controller.

sparkfun_color_lcd_shield_fini

Synopsis

```
CTL_STATUS_t sparkfun_color_lcd_shield_fini(SPARKFUN_COLOR_LCD_SHIELD_t *self);
```

Description

sparkfun_color_lcd_shield_fini finalizes the shield **self** and deselects it.

Return Value

sparkfun_color_lcd_shield_fini returns a standard status code.

sparkfun_color_lcd_shield_init

Synopsis

```
CTL_STATUS_t sparkfun_color_lcd_shield_init(SPARKFUN_COLOR_LCD_SHIELD_t *self);
```

Description

sparkfun_color_lcd_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

sparkfun_color_lcd_shield_init returns a standard status code.

<watterott_msd_shield.h>

Overview

Drivers for the Watterott MSD shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.watterott.com/en/Arduino-mSD-Shield

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
WATTEROTT_MSD_SHIELD_t	Driver data
Shield	
watterott_msd_shield_fini	Finalize shield
watterott_msd_shield_init	Initialize shield

WATTEROTT_MSD_SHIELD_t

Synopsis

```
typedef struct {
  CTL_SPI_DEVICE_t lcd_device;
  CTL_GFX_CONTROLLER_t lcd_driver;
} WATTEROTT_MSD_SHIELD_t;
```

Description

WATTEROTT_MSD_SHIELD_t holds the instance data required to run the display.

Structure

`lcd_device`

The SPI device that communicates with the MSD shield.

`lcd_driver`

The graphics driver for the HX8347D.

watterott_msd_shield_fini

Synopsis

```
CTL_STATUS_t watterott_msd_shield_fini(WATTEROTT_MSD_SHIELD_t *self);
```

Description

watterott_msd_shield_fini finalizes the shield **self** and deselects it.

Return Value

watterott_msd_shield_fini returns a standard status code.

watterott_msd_shield_init

Synopsis

```
CTL_STATUS_t watterott_msd_shield_init(WATTEROTT_MSD_SHIELD_t *self);
```

Description

watterott_msd_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

watterott_msd_shield_init returns a standard status code.

<watterott_s65_shield.h>

Overview

Drivers for the Watterott S65 shield.

Resources

All correct at time of writing, but as always, manufacturers love to move things around on their website.

Web page

http://www.watterott.com/en/Arduino-S65-Shield

Test hardware

This code has been tested using a SolderCore:

http://soldercore.com/products/soldercore/

API Summary

Instance	
WATTEROTT_S65_SHIELD_t	Driver data
Shield	
watterott_s65_shield_fini	Finalize shield
watterott_s65_shield_init	Initialize shield

WATTEROTT_S65_SHIELD_t

Synopsis

```
typedef struct {
  CTL_GFX_CONTROLLER_t lcd_driver;
  CTL_SPI_DEVICE_t lcd_device;
} WATTEROTT_S65_SHIELD_t;
```

Description

WATTEROTT_S65_SHIELD_t holds the instance data required to run the display.

Structure

`lcd_device`

The SPI device that communicates with the MSD shield.

`lcd_driver`

The graphics driver for the HD66773.

watterott_s65_shield_fini

Synopsis

```
CTL_STATUS_t watterott_s65_shield_fini(WATTEROTT_S65_SHIELD_t *self);
```

Description

watterott_s65_shield_fini finalizes the shield **self** and deselects it.

Return Value

watterott_s65_shield_fini returns a standard status code.

watterott_s65_shield_init

Synopsis

```
CTL_STATUS_t watterott_s65_shield_init(WATTEROTT_S65_SHIELD_t *self);
```

Description

watterott_s65_shield_init initializes the shield instance data **self** and prepares the shield for use. The graphics display is selected for use, but the display is not cleared.

Return Value

watterott_s65_shield_init returns a standard status code.