ECEN 5013 Final Project

Generated by Doxygen 1.8.12

Contents

Chapter 1

ECEN5013_final_project

Final project for ECEN 5013 Embedded Software Essentials Designing a camera module interface for the STM32↩ F429

This project integrates the STM32F429I-DISC1 board with an OV7670/OV5642 camera module and an ESP8266 wifi module.

For full functional documentation, explore the doc folder. You will find documentation created by Doxygen and a final report on the system.

PIN MAP

```
/**********************************
 * DCMI      | Pin    | OVxxxx Pin
 * ---------------------------------
 * VSYNC     | PB7    | 5
 * HSYNC     | PA4    | 6
 * PIXCLK    | PA6    | 7
 * D7        | PB9    | 9
 * D6        | PB8    | 10
 * D5        | PD3    | 11
 * D4        | PC11   | 12
 * D3        | PC9    | 13
 * D2        | PC8    | 14
 * D1        | PC7    | 15
 * D0        | PC6    | 16
 **********************************/

/**********************************
 * I2C       | Pin    | OVxxxx Pin
 * ---------------------------------
 * SCL       | PB10   | 3
 * SDA       | PB11   | 4
 **********************************/

/**********************************
 * MCO1      | Pin    | OVxxxx Pin
 * ---------------------------------
 * MCO1      | PA8    | 8
 **********************************/

/**********************************
 * UART 2 Host        | Pin
 * ---------------------------------
 * Tx                 | D5
 * Rx                 | D6
 * ---------------------------------
 * UART cable connections
 * Black GND to GND
```

```
 * White Rx to D5 Tx
 * Green Tx to D6 Rx
 * Red NC
 ********************************/

/*********************************
 * UART 1 ESP8266   | Pin
 * -------------------------------
 * Tx               | A9
 * -------------------------------
 * Connect to GPIO13 (D7) on the
 * ESP8266 board
 ********************************/
```

Chapter 2

Bug List

File cam.h
No known bugs.

File cmd.h
No known bugs.

File err.h
No known bugs.

File esp8266.h
No known bugs.

File log.h
No known bugs.

File mod.h
No known bugs.

File ov5642.h
No known bugs.

File ov5642_regs.h
No known bugs.

File ov7670.h
No known bugs.

File ov7670_regs.h
No known bugs.

File prof.h
No known bugs.

File sdram.h
No known bugs.

File template.h
No known bugs.

File test.h
No known bugs.

File wifi.h
No known bugs.

Chapter 3

File Index

3.1 File List

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

Chapter 4

File Documentation

4.1 inc/project/cam.h File Reference

Function prototypes for the camera controller.

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

Functions

- cam_status_t cam_Init ()
  Initialize the camera.
- cam_status_t cam_Configure ()
  Configure the camera module.
- cam_status_t cam_Capture ()
  Capture an image.
- cam_status_t cam_Transfer ()
  Transfer an image to the debug console.

4.1.1 Detailed Description

Function prototypes for the camera controller.

This contains the prototypes, macros, constants, and global variables for the camera controller.

Author

Ben Heberlein

Bug No known bugs.

4.1.2 Function Documentation

4.1.2.1 cam_Capture()

```
cam_status_t cam_Capture ( )
```

Capture an image.

This function should capture and store an image in the SDRAM.

Returns

a status of type cam_status_t

4.1.2.2 cam_Configure()

```
cam_status_t cam_Configure ( )
```

Configure the camera module.

This function should initialize the camera to take images and store them in SDRAM.

Returns

a status of type cam_status_t

4.1.2.3 cam_Init()

```
cam_status_t cam_Init ( )
```

Initialize the camera.

This function initializes the camera. It should be called before attempting to configure or take an image.

Returns

a status of type cam_status_t

4.1.2.4 cam_Transfer()

```
cam_status_t cam_Transfer ( )
```

Transfer an image to the debug console.

This function transfers an image from SDRAM to the debug host interface. The logger must be enabled for this function to work.

Returns

a status of type cam_status_t

4.2 inc/project/cmd.h File Reference

Function prototypes for the command module.

```
#include <stdint.h>
#include "log.h"
#include "err.h"
#include "mod.h"
```

Macros

- #define **CMD_QUEUE_CAP** 256
- #define **CMD_BAUDRATE** 115200

Typedefs

- typedef uint8_t **gen_func_t**
- typedef enum log_func_e **log_func_t**
- typedef enum cmd_func_e **cmd_func_t**
- typedef enum stdlib_func_e **stdlib_func_t**
- typedef enum sdram_func_e **sdram_func_t**
- typedef enum ov5642_func_e **ov5642_func_t**
- typedef enum ov7670_func_e **ov7670_func_t**
- typedef enum prof_func_e **prof_func_t**
- typedef enum test_func_e **test_func_t**
- typedef enum cam_func_e **cam_func_t**
- typedef enum cmd_queue_state_e **cmd_queue_state_t**

Enumerations

- enum **log_func_e** { **LOG_FUNC_INIT** }
- enum **cmd_func_e** { **CMD_FUNC_INIT** }
- enum **stdlib_func_e** { **STDLIB_FUNC_DUMMY** }
- enum **sdram_func_e** { **SDRAM_FUNC_INIT** }
- enum **ov5642_func_e** { **ov5642_FUNC_DUMMY** }
- enum **ov7670_func_e** { **OV7670_FUNC_DUMMY** }
- enum **prof_func_e** { **PROF_FUNC_INIT** }
- enum **test_func_e** { **TEST_FUNC_DUMMY** }
- enum **cam_func_e** { **CAM_FUNC_INIT**, **CAM_FUNC_CONFIG**, **CAM_FUNC_CAPTURE**, **CAM_FUNC_↩ TRANSFER** }
- enum **cmd_queue_state_e** { **CMD_QUEUE_EMPTY**, **CMD_QUEUE_FULL**, **CMD_QUEUE_PARTIAL**, **C↩ MD_QUEUE_INVALID** }

Functions

- struct __**attribute**__ ((packed)) cmd_cmd_s
- cmd_status_t cmd_queueInit ()

  Initialize the queue.
- cmd_status_t cmd_CmdDeallocate (cmd_cmd_t ∗cmd)

  Deallocate command memory at the command pointer.
- cmd_status_t cmd_CmdAllocate (cmd_cmd_t ∗∗cmd, uint16_t dataLen)

  Allocate command memory at the command pointer.
- cmd_status_t cmd_QueueGetStatus ()

  Get the status of the queue.
- cmd_status_t cmd_QueuePut (cmd_cmd_t ∗cmd)

  Put a command in the queue.
- cmd_status_t cmd_QueueGet (cmd_cmd_t ∗∗cmd)

  Get a command from the queue.
- cmd_status_t cmd_Init ()

  Initialize the command module.
- cmd_status_t cmd_Loop ()

  The main command loop.

Variables

- **cmd_cmd_t**
- **cmd_queue_t**

4.2.1 Detailed Description

Function prototypes for the command module.

This contains the prototypes, macros, constants, and global variables for the command module.

Author

Ben Heberlein

Bug No known bugs.

4.2.2 Function Documentation

4.2.2.1 cmd_CmdAllocate()

```
cmd_status_t cmd_CmdAllocate (
            cmd_cmd_t ** cmd,
            uint16_t dataLen )
```

Allocate command memory at the command pointer.

Note: When the command is finally executed, the memory will be freed.

This function will allocate memory for the cmd_cmd_t structure and the cmd_data field, both of which will be freed when the command is executed. You can free the command memory prematurely by calling cmd_CmdDeallocate() but don't do this if you you are planning to put the command into the queue!

Returns

a status value of the type cmd_status_t

4.2.2.2 cmd_CmdDeallocate()

```
cmd_status_t cmd_CmdDeallocate (
            cmd_cmd_t * cmd )
```

Deallocate command memory at the command pointer.

This function will deallocate memory for the cmd_cmd_t structure and the cmd_data field.

Parameters

cmd	a pointer to the command to free

Returns

a status value of the type cmd_status_t

4.2.2.3 cmd_Init()

```
cmd_status_t cmd_Init ( )
```

Initialize the command module.

This function initializes the main command queue, and if the compiler directive __CMD is set, initalizes UART2 for Rx mode to receive commands from the debug interface. It should be noted that the command queue and loop will still function without the __CMD directive. The directive only turns off the UART module and the connection to the debug interface.

Returns

a status value of the type cmd_status_t

4.2.2.4 cmd_Loop()

```
cmd_status_t cmd_Loop ( )
```

The main command loop.

This function checks the main command queue for pending commands and executes them if there are any. It also calls cmd_Deallocate to free the memory associated with commands.

Returns

a status value of the type cmd_status_t

4.2.2.5 cmd_QueueGet()

```
cmd_status_t cmd_QueueGet (
            cmd_cmd_t ** cmd )
```

Get a command from the queue.

This function gets a command from the queue, returning QUEUE_INFO_OK or an error if the queue is empty or in an invalid state. The command is put in the pointer location cmd (passed by reference);

Parameters

cmd	the location to put the command

Returns

a status value of the type cmd_status_t

4.2.2.6 cmd_QueueGetStatus()

```
cmd_status_t cmd_QueueGetStatus ( )
```

Get the status of the queue.

This will return a status code that will tell the caller if the cmd_queue is empty, full, partial, or in an invalid state.

Parameters

cmd	the address of a pointer to a cmd
dataLen	data length to allocate

Returns

a status value of the type cmd_status_t

4.2.2.7 cmd_queueInit()

```
cmd_status_t cmd_queueInit ( )
```

Initialize the queue.

Initializes the cmd_queue

Returns

a status value of the type cmd_status_t

4.2.2.8 cmd_QueuePut()

```
cmd_status_t cmd_QueuePut (
             cmd_cmd_t * cmd )
```

Put a command in the queue.

This function puts a command in the queue if the queue is not full.

NOTE: For the command to get executed correctly, the caller must allocate space on the heap for both the command and the command data! If the command passed in is not allocated, the command loop will try to execute on garbage memory!

Example call: cmd_cmd_t ∗cmd = (cmd_cmd_t ∗) malloc(sizeof(cmd_cmd_t)); cmd->cmd_module = LOG; cmd->cmd_function = LOG_FUNC_LOG; cmd->cmd_dataLen = 2; cmd->cmd_data = (uint8_t ∗) malloc(cmd->cmd↩ _dataLen); cmd->cmd_data[0] = 1; cmd->cmd_data[1] = 0;

or using cmd_Allocate(): cmd_cmd_t cmd; // Address of pointer is important cmd_status_t ret = cmd_Allocate(&cmd, 2); cmd->module = LOG; cmd->function = LOG_FUNC_LOG; cmd->cmd_data[0] = 1; cmd->cmd_data[0] = 0;

This would log a message to the debug console CMD: CMD_INFO_OK

Parameters

a	command to put into the queue

Returns

a status value of the type cmd_status_t

4.3 inc/project/err.h File Reference

Definitions for errors, warnings, and status.

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

Macros

- #define **INFO** 0
- #define **WARN** 20
- #define **ERR** 40
- #define **END** 60

Typedefs

- typedef uint8_t **gen_status_t**
- typedef enum log_status_e **log_status_t**
- typedef enum cmd_status_e **cmd_status_t**
- typedef enum stdlib_status_e **stdlib_status_e**
- typedef enum sdram_status_e **sdram_status_t**
- typedef enum ov5642_status_e **ov5642_status_t**
- typedef enum ov7670_status_e **ov7670_status_t**
- typedef enum prof_status_e **prof_status_t**
- typedef enum test_status_e **test_status_t**
- typedef enum cam_status_e **cam_status_t**
- typedef enum esp8266_status_e **esp8266_status_t**
- typedef enum wifi_status_e **wifi_status_t**

Enumerations

- enum **log_status_e** {
  LOG_INFO_OK = INFO, **LOG_INFO_UNKNOWN** = WARN-1, **LOG_WARN_ALINIT** = WARN, **LOG_WA**↩
  RN_IGNORED = WARN+1,
  LOG_WARN_UNKNOWN = ERR-1, **LOG_ERR_DATASIZE** = ERR, **LOG_ERR_MSGSIZE** = ERR+1, **LO**↩
  G_ERR_LOGOFF = ERR+2,
  LOG_ERR_UNKNOWN = END-1 }

- enum **cmd_status_e** {
  CMD_INFO_OK = INFO, **CMD_INFO_INTERRUPT** = INFO+1, **CMD_INFO_QUEUEEMPTY** = INFO+2, **C↩
  MD_INFO_QUEUEFULL** = INFO+3,
  CMD_INFO_QUEUEPARTIAL = INFO+4, **CMD_INFO_UNKNOWN** = WARN-1, **CMD_WARN_FREE** = W↩
  ARN, **CMD_WARN_ALINIT** = WARN+1,
  CMD_WARN_UNKNOWN = ERR-1, **CMD_ERR_QUEUEEMPTY** = ERR, **CMD_ERR_QUEUEFULL** = ER↩
  R+1, **CMD_ERR_QUEUEINVALID** = ERR+2,
  CMD_ERR_MALLOC = ERR+3, **CMD_ERR_NULLPTR** = ERR+4, **CMD_ERR_DATA** = ERR+5, **CMD_E↩
  RR_NOFUNC** = ERR+6,
  CMD_ERR_NOMOD = ERR+7, **CMD_ERR_UNKNOWN** = END-1 }
- enum **stdlib_status_e** { **STDLIB_INFO_OK** = INFO, **STDLIB_INFO_UNKNOWN** = WARN-1, **STDLIB_W↩
  ARN_UNKNOWN** = ERR-1, **STDLIB_ERR_UNKNOWN** = END-1 }
- enum **sdram_status_e** {
  SDRAM_INFO_OK = INFO, **SDRAM_INFO_UNKNOWN** = WARN-1, **SDRAM_WARN_ALINIT** = WARN, **S↩
  DRAM_WARN_UNKNOWN** = ERR-1,
  SDRAM_ERR_UNKNOWN = END-1 }
- enum **ov5642_status_e** {
  OV5642_INFO_OK = INFO, **OV5642_INFO_IMAGE** = INFO+1, **OV5642_INFO_UNKNOWN** = WARN-1, **O↩
  V5642_WARN_ALINIT** = WARN,
  OV5642_WARN_UNKNOWN = ERR-1, **OV5642_ERR_I2CSTART** = ERR, **OV5642_ERR_I2CREAD** = E↩
  RR+1, **OV5642_ERR_I2CWRITE** = ERR+2,
  OV5642_ERR_I2CTIMEOUT = ERR+3, **OV5642_ERR_UNKNOWN** = END-1 }
- enum **ov7670_status_e** {
  OV7670_INFO_OK = INFO, **OV7670_INFO_IMAGE** = INFO+1, **OV7670_INFO_UNKNOWN** = WARN-1, **O↩
  V7670_WARN_ALINIT** = WARN,
  OV7670_WARN_UNKNOWN = ERR-1, **OV7670_ERR_I2CSTART** = ERR, **OV7670_ERR_I2CREAD** = E↩
  RR+1, **OV7670_ERR_I2CWRITE** = ERR+2,
  OV7670_ERR_I2CTIMEOUT = ERR+3, **OV7670_ERR_UNKNOWN** = END-1 }
- enum **prof_status_e** {
  PROF_INFO_OK = INFO, **PROF_INFO_RESULTS** = INFO+1, **PROF_INFO_UNKNOWN** = WARN-1, **PR↩
  OF_WARN_ALINIT** = WARN,
  PROF_WARN_UNKNOWN = ERR-1, **PROF_ERR_UNKNOWN** = END-1 }
- enum **test_status_e** {
  TEST_INFO_OK = INFO, **TEST_INFO_PASSED** = INFO+1, **TEST_INFO_CONFIRM** = INFO+2, **TEST_I↩
  NFO_UNKNOWN** = WARN-1,
  TEST_WARN_FAILED = WARN, **TEST_WARN_UNKNOWN** = ERR-1, **TEST_ERR_UNKNOWN** = END-1 }
- enum **cam_status_e** {
  CAM_INFO_OK = INFO, **CAM_INFO_IMAGE** = INFO+1, **CAM_INFO_UNKNOWN** = WARN-1, **CAM_WA↩
  RN_ALINIT** = WARN,
  CAM_WARN_ALCONF = WARN+1, **CAM_WARN_UNKNOWN** = ERR-1, **CAM_ERR_INIT** = ERR, **CAM↩
  _ERR_CONFIG** = ERR+1,
  CAM_ERR_CAPTURE = ERR+2, **CAM_ERR_TRANSFER** = ERR+3, **CAM_ERR_UNKNOWN** = END-1 }
- enum **esp8266_status_e** { **ESP8266_INFO_OK** = INFO, **ESP8266_INFO_UNKNOWN** = WARN-1, **ES↩
  P8266_WARN_UNKNOWN** = ERR-1, **ESP8266_ERR_UNKNOWN** = END-1 }
- enum **wifi_status_e** {
  WIFI_INFO_OK = INFO, **WIFI_INFO_UNKNOWN** = WARN-1, **WIFI_WARN_ALINIT** = WARN, **WIFI_WA↩
  RN_UNKNOWN** = ERR-1,
  WIFI_ERR_DATASIZE = ERR, **WIFI_ERR_MSGSIZE** = ERR+1, **WIFI_ERR_SEND** = ERR+2, **WIFI_ERR↩
  _INIT** = ERR+3,
  WIFI_ERR_UNKNOWN = END-1 }

4.3.1 Detailed Description

Definitions for errors, warnings, and status.

This contains the definitions for errors, warnings, and status for every module. INFO enum values are reserved as 0-19, WARN enum values are reserved as 20-39, and ERR enum values are reserved as 40-59.

This file also handles the standard peripheral library assertion failure by logging the failure to the debug interface.

Author

Ben Heberlein

Bug No known bugs.

4.4 inc/project/esp8266.h File Reference

Function declarateions for the esp8266 module.

```
#include "err.h"
#include "wifi.h"
#include <stdint.h>
```

Macros

- #define **ESP8266_BAUDRATE** 9600

Functions

- esp8266_status_t esp8266_Send (wifi_packet_t *wifi_packet)

  Send data to the ESP8622 over UART.
- esp8266_status_t esp8266_Init ()

  Initialize the ESP8266 driver.

4.4.1 Detailed Description

Function declarateions for the esp8266 module.

This code is for the stm32f429 side. The actual NodeMCU code running on the esp8266 can be found in the periph folder in the root directory.

Author

Ben Heberlein

Bug No known bugs.

4.4.2 Function Documentation

4.4.2.1 esp8266_Init()

```
esp8266_status_t esp8266_Init ( )
```

Initialize the ESP8266 driver.

This function initializes UART1 to communicate with the ESP8622 wifi module.

Returns

a return code of type esp8266_status_t

4.4.2.2 esp8266_Send()

```
esp8266_status_t esp8266_Send (
            wifi_packet_t * wifi_packet )
```

Send data to the ESP8622 over UART.

ESP866 should be initialized before calling this function. This function sends a single wifi packet over UART.

Parameters

wifi_packet	the packet to send

Returns

a return code of type esp8266_status_t

4.5 inc/project/log.h File Reference

Logging functions for the debug serial terminal connection.

Macros

- #define **log_Log**(...) LOG_ERR_LOGOFF

4.5.1 Detailed Description

Logging functions for the debug serial terminal connection.

This file contains logging functions, macros, and initialization code for the serial logger. This code can be toggled on or off with the __LOG preprocessor macro.

Author

Ben Heberlein

Bug No known bugs.

4.6 inc/project/mod.h File Reference

Module definitions.

Typedefs

- typedef enum mod_e mod_t

  Enumeration of module types.

Enumerations

- enum mod_e {
  LOG, **CMD**, **STDLIB**, **SDRAM**,
  OV5642, **OV7670**, **PROF**, **TEST**,
  CAM, **ESP8266**, **WIFI** }

  Enumeration of module types.

4.6.1 Detailed Description

Module definitions.

This file defines a module enumeration for use with the logger and the command modules.

Author

Ben Heberlein

Bug No known bugs.

4.7 inc/project/ov5642.h File Reference

Function prototypes for the ov5642 camera.

```
#include "err.h"
#include "ov5642_regs.h"
#include <stdint.h>
```

Macros

- #define **OV5642_DCMI_BASEADDR** ((uint32_t)0x50050000)
- #define **OV5642_DCMI_OFFSETDR** 0x28
- #define **OV5642_DCMI_PERIPHADDR** (OV5642_DCMI_BASEADDR | OV5642_DCMI_OFFSETDR)
- #define **OV5642_IMAGE_BUFSIZE** 65535
- #define **OV5642_I2C2_SPEED** 100000
- #define **OV5642_I2C2_ACK** 1
- #define **OV5642_I2C2_NACK** 0
- #define **OV5642_I2C2_TIMEOUT** 0x4000
- #define **OV5642_I2C2_READADDR** 0x79
- #define **OV5642_I2C2_WRITEADDR** 0x78

Functions

- ov5642_status_t ov5642_clockInit ()
  Initialize the clock with RCC.
- ov5642_status_t ov5642_dmaInit ()
  Initialize the DMA controller.
- ov5642_status_t ov5642_dcmiInit ()
  Initialize the DCMI module.
- ov5642_status_t ov5642_i2cInit ()
  Initialize I2C.
- ov5642_status_t ov5642_i2cStart (uint8_t address, uint8_t direction)
  Start I2C transmission.
- ov5642_status_t ov5642_i2cStop ()
  Stop I2C transmission.
- ov5642_status_t ov5642_i2cRead (uint8_t *data, uint8_t ack)

Read a byte from the I2C bus.

- ov5642_status_t ov5642_i2cWrite (uint8_t data)

  Write a byte to the I2C bus.
- ov5642_status_t ov5642_regWrite (uint16_t reg, uint8_t value)

  Write a register in the OV5642.
- ov5642_status_t ov5642_regRead (uint16_t reg, uint8_t *value)

  Read a register in the OV5642.
- ov5642_status_t ov5642_regWriteArray (const ov5642_reg_t *reg)

  Write an array of register values in the OV5642.
- void DCMI_IRQHandler ()

  Frame complete interrupt handler.
- ov5642_status_t ov5642_Init ()

  Initialize the ov5642 camera.
- ov5642_status_t ov5642_Configure ()

  Transfer a configuration to the camera over I2C.
- ov5642_status_t ov5642_Capture ()

  Capture an image and put it into SDRAM.
- ov5642_status_t ov5642_Transfer ()

  Transfer an image from SDRAM to the host.

4.7.1 Detailed Description

Function prototypes for the ov5642 camera.

This contains the driver function prototypes, macros, constants, and global variables.

Author

Ben Heberlein

Bug No known bugs.

4.7.2 Function Documentation

4.7.2.1 DCMI_IRQHandler()

```
void DCMI_IRQHandler ( )
```

Frame complete interrupt handler.

This file implements an interrupt handler for the frame complete interrupt in the DCMI controller. Currently this handler does nothing.

4.7.2.2 ov5642_Capture()

```
ov5642_status_t ov5642_Capture ( )
```

Capture an image and put it into SDRAM.

This function commands the ov5642 module to take an image, and transfers the image to SDRAM using DMA and DCMI functionality.

Returns

a status code of the type ov5642_status_t

4.7.2.3 ov5642_clockInit()

```
ov5642_status_t ov5642_clockInit ( )
```

Initialize the clock with RCC.

This function initializes the clock used by the ov5642 camera module.

Returns

a status code of the type ov5642_status_t

4.7.2.4 ov5642_Configure()

```
ov5642_status_t ov5642_Configure ( )
```

Transfer a configuration to the camera over I2C.

This function transfers a configuration from the definitions in ov5642_regs.h to the camera module over I2C.

Returns

a status code of the type ov5642_status_t

4.7.2.5 ov5642_dcmiInit()

```
ov5642_status_t ov5642_dcmiInit ( )
```

Initialize the DCMI module.

This function initializes and configures DCMI. The following pin mapping is used.

DCMI | Pin

VSYNC | PB7 HSYNC | PA4 PIXCLK | PA6 D7 | PB9 D6 | PB8 D5 | PD3 D4 | PC11 D3 | PC9 D2 | PC8 D1 | PC7

D0 | PC6

DCMI is configured for snapshot mode.

Returns

a status code of the type ov5642_status_t

4.7.2.6 ov5642_dmaInit()

```
ov5642_status_t ov5642_dmaInit ( )
```

Initialize the DMA controller.

This function initializes and configures DMA for the ov5642 camera module. DMA is configured with DMA2, Stream 1, Channel 1. See page 310 of the STM32F4 family reference manual RM0090.

Configured for circular mode, with maximum stream NDT transfer size. Configured to transfer into SDRAM.

Returns

a status code of the type ov5642_status_t

4.7.2.7 ov5642_i2cInit()

```
ov5642_status_t ov5642_i2cInit ( )
```

Initialize I2C.

This function initializes I2C to set and control ov5642 camera configuration.

Returns

a status code of the type ov5642_status_t

4.7.2.8 ov5642_i2cRead()

```
ov5642_status_t ov5642_i2cRead (
            uint8_t * data,
            uint8_t ack )
```

Read a byte from the I2C bus.

This function reads a single byte and sends and ACK signal back to the sender. The byte is put in the memory location of data

Parameters

data	the byte read

Returns

a status code of the type ov5642_status_t

4.7.2.9 ov5642_i2cStart()

```
ov5642_status_t ov5642_i2cStart (
```

```
            uint8_t address,
            uint8_t direction )
```

Start I2C transmission.

This function starts a transmission on 12C2 to the address given with the specified direction. Use either I2C_↩ Direction_Transmitter or I2C_Driection_Receiver for the direction parameter.

Parameters

address	address of the slave
direction	transfer direction

Returns

a status code of the type ov5642_status_t

4.7.2.10 ov5642_i2cStop()

```
ov5642_status_t ov5642_i2cStop ( )
```

Stop I2C transmission.

This function stops an I2C2 transaction by sending the STOP condition.

Returns

a status code of the type ov5642_status_t

4.7.2.11 ov5642_i2cWrite()

```
ov5642_status_t ov5642_i2cWrite (
            uint8_t data )
```

Write a byte to the I2C bus.

This function writes a single byte and sends and waits for the byte to be transmitted. If ack is true, request another byte. If ack is false, end the do not request another byte (ending transmission).

Parameters

data	the byte to send
ack	if ack is true, request another byte

Returns

a status code of the type ov5642_status_t

4.7.2.12 ov5642_Init()

```
ov5642_status_t ov5642_Init ( )
```

Initialize the ov5642 camera.

This function fully initializes the ov5642 camera. This should be called before attempting to configure or take an image.

Returns

a status code of the type ov5642_status_t

4.7.2.13 ov5642_regRead()

```
ov5642_status_t ov5642_regRead (
            uint16_t reg,
            uint8_t * value )
```

Read a register in the OV5642.

This function reads a register in the OV5642 using the I2C interface. The I2C interface must be configured before using this function.

Parameters

reg	the register to read
value	a pointer to the value that is read

Returns

a status code of the type ov5642_status_t

4.7.2.14 ov5642_regWrite()

```
ov5642_status_t ov5642_regWrite (
            uint16_t reg,
            uint8_t value )
```

Write a register in the OV5642.

This function writes a register in the OV5642 using the I2C interface. The I2C interface must be configured before using this function.

Parameters

reg	the register to write
value	the value to write to the register

Returns

a status code of the type ov5642_status_t

4.7.2.15 ov5642_regWriteArray()

```
ov5642_status_t ov5642_regWriteArray (
            const ov5642_reg_t * reg )
```

Write an array of register values in the OV5642.

This function writes the registers in the OV5642 using the I2C interface. The I2C interface must be configured before using this function. This function writes the registers based on address-value mappings in the ov5642_regs.h file.

Parameters

reg	the register mapping to write.

Returns

a status code of the type ov5642_status_t

4.7.2.16 ov5642_Transfer()

```
ov5642_status_t ov5642_Transfer ( )
```

Transfer an image from SDRAM to the host.

This function uses the logger to transfer a full image from SDRAM to the host computer. This function will not work if the logger is disabled (directive __LOG needs to be on).

Returns

a status code of the type ov5642_status_t

4.8 inc/project/ov5642_regs.h File Reference

Register configuration definitions for the ov5642 camera module.

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

Macros

- #define **OV5642_CHIPID_HIGH** 0x300a
- #define **OV5642_CHIPID_LOW** 0x300b

Functions

- struct **__attribute__** ((packed)) ov5642_reg_s

Variables

- **ov5642_reg_t**

4.8.1 Detailed Description

Register configuration definitions for the ov5642 camera module.

This register map was taken from the ArduCAM github at https://github.com/ArduCAM/Arduino.

This code was released under the MIT license.

Author

ArduCam
Ben Heberlein

Bug No known bugs.

4.9 inc/project/ov7670.h File Reference

Function prototypes for the ov7670 camera.

```
#include "err.h"
#include "ov7670_regs.h"
#include <stdint.h>
```

Macros

- #define **OV7670_DCMI_BASEADDR** ((uint32_t)0x50050000)
- #define **OV7670_DCMI_OFFSETDR** 0x28
- #define **OV7670_DCMI_PERIPHADDR** (OV7670_DCMI_BASEADDR | OV7670_DCMI_OFFSETDR)
- #define **OV7670_IMAGE_BUFSIZE** 320∗240∗2
- #define **OV7670_I2C2_SPEED** 100000
- #define **OV7670_I2C2_ACK** 1
- #define **OV7670_I2C2_NACK** 0
- #define **OV7670_I2C2_TIMEOUT** 0x4000
- #define **OV7670_I2C2_READADDR** 0x43
- #define **OV7670_I2C2_WRITEADDR** 0x42

Functions

- ov7670_status_t ov7670_clockInit ()

  Initialize the clock with RCC.
- ov7670_status_t ov7670_dmaInit ()

  Initialize the DMA controller.
- ov7670_status_t ov7670_dcmiInit ()

  Initialize the DCMI module.
- ov7670_status_t ov7670_i2cInit ()

  Initialize I2C.
- ov7670_status_t ov7670_i2cStart (uint8_t address, uint8_t direction)

  Start I2C transmission.
- ov7670_status_t ov7670_i2cStop ()

  Stop I2C transmission.
- ov7670_status_t ov7670_i2cRead (uint8_t *data, uint8_t ack)

  Read a byte from the I2C bus.
- ov7670_status_t ov7670_i2cWrite (uint8_t data)

  Write a byte to the I2C bus.
- ov7670_status_t ov7670_regWrite (uint8_t reg, uint8_t value)

  Write a register in the OV7670.
- ov7670_status_t ov7670_regRead (uint8_t reg, uint8_t *value)

  Read a register in the OV7670.
- ov7670_status_t ov7670_regWriteArray (const ov7670_reg_t *reg)

  Write an array of register values in the OV7670.
- void DCMI_IRQHandler ()

  Frame complete interrupt handler.
- ov7670_status_t ov7670_Init ()

  Initialize the ov7670 camera.
- ov7670_status_t ov7670_Configure ()

  Transfer a configuration to the camera over I2C.
- ov7670_status_t ov7670_Capture ()

  Capture an image and put it into SDRAM.
- ov7670_status_t ov7670_Transfer ()

  Transfer an image from SDRAM to the host.

4.9.1 Detailed Description

Function prototypes for the ov7670 camera.

This contains the driver function prototypes, macros, constants, and global variables.

Author

Ben Heberlein

Bug No known bugs.

4.9.2 Function Documentation

4.9.2.1 DCMI_IRQHandler()

```
void DCMI_IRQHandler ( )
```

Frame complete interrupt handler.

This file implements an interrupt handler for the frame complete interrupt in the DCMI controller. Currently this handler does nothing.

4.9.2.2 ov7670_Capture()

```
ov7670_status_t ov7670_Capture ( )
```

Capture an image and put it into SDRAM.

This function commands the ov7670 module to take an image, and transfers the image to SDRAM using DMA and DCMI functionality.

Returns

a status code of the type ov7670_status_t

4.9.2.3 ov7670_clockInit()

```
ov7670_status_t ov7670_clockInit ( )
```

Initialize the clock with RCC.

This function initializes the clock used by the ov7670 camera module.

Returns

a status code of the type ov7670_status_t

4.9.2.4 ov7670_Configure()

```
ov7670_status_t ov7670_Configure ( )
```

Transfer a configuration to the camera over I2C.

This function transfers a configuration from the definitions in ov7670_regs.h to the camera module over I2C.

Returns

a status code of the type ov7670_status_t

4.9.2.5 ov7670_dcmiInit()

```
ov7670_status_t ov7670_dcmiInit ( )
```

Initialize the DCMI module.

This function initializes and configures DCMI. The following pin mapping is used.

DCMI | Pin

VSYNC | PB7 HSYNC | PA4 PIXCLK | PA6 D7 | PB9 D6 | PB8 D5 | PD3 D4 | PC11 D3 | PC9 D2 | PC8 D1 | PC7

D0 | PC6

DCMI is configured for snapshot mode.

Returns

a status code of the type ov7670_status_t

4.9.2.6 ov7670_dmaInit()

```
ov7670_status_t ov7670_dmaInit ( )
```

Initialize the DMA controller.

This function initializes and configures DMA for the ov7670 camera module. DMA is configured with DMA2, Stream 1, Channel 1. See page 310 of the STM32F4 family reference manual RM0090.

Configured for circular mode, with maximum stream NDT transfer size. Configured to transfer into SDRAM.

Returns

a status code of the type ov7670_status_t

4.9.2.7 ov7670_i2cInit()

```
ov7670_status_t ov7670_i2cInit ( )
```

Initialize I2C.

This function initializes I2C to set and control ov7670 camera configuration.

Returns

a status code of the type ov7670_status_t

4.9.2.8 ov7670_i2cRead()

```
ov7670_status_t ov7670_i2cRead (
            uint8_t * data,
            uint8_t ack )
```

Read a byte from the I2C bus.

This function reads a single byte and sends and ACK signal back to the sender. The byte is put in the memory location of data

Parameters

data	the byte read

Returns

a status code of the type ov7670_status_t

4.9.2.9 ov7670_i2cStart()

```
ov7670_status_t ov7670_i2cStart (
            uint8_t address,
            uint8_t direction )
```

Start I2C transmission.

This function starts a transmission on 12C2 to the address given with the specified direction. Use either I2C_↩ Direction_Transmitter or I2C_Driection_Receiver for the direction parameter.

Parameters

address	address of the slave
direction	transfer direction

Returns

a status code of the type ov7670_status_t

4.9.2.10 ov7670_i2cStop()

```
ov7670_status_t ov7670_i2cStop ( )
```

Stop I2C transmission.

This function stops an I2C2 transaction by sending the STOP condition.

Returns

a status code of the type ov7670_status_t

4.9.2.11 ov7670_i2cWrite()

```
ov7670_status_t ov7670_i2cWrite (
            uint8_t data )
```

Write a byte to the I2C bus.

This function writes a single byte and sends and waits for the byte to be transmitted. If ack is true, request another byte. If ack is false, end the do not request another byte (ending transmission).

Parameters

data	the byte to send
ack	if ack is true, request another byte

Returns

a status code of the type ov7670_status_t

4.9.2.12 ov7670_Init()

```
ov7670_status_t ov7670_Init ( )
```

Initialize the ov7670 camera.

This function fully initializes the ov7670 camera. This should be called before attempting to configure or take an image.

Returns

a status code of the type ov7670_status_t

4.9.2.13 ov7670_regRead()

```
ov7670_status_t ov7670_regRead (
            uint8_t reg,
            uint8_t * value )
```

Read a register in the OV7670.

This function reads a register in the OV7670 using the I2C interface. The I2C interface must be configured before using this function.

Parameters

reg	the register to read
value	a pointer to the value that is read

Returns

a status code of the type ov7670_status_t

4.9.2.14 ov7670_regWrite()

```
ov7670_status_t ov7670_regWrite (
            uint8_t reg,
            uint8_t value )
```

Write a register in the OV7670.

This function writes a register in the OV7670 using the I2C interface. The I2C interface must be configured before using this function.

Parameters

reg	the register to write
value	the value to write to the register

Returns

a status code of the type ov7670_status_t

4.9.2.15 ov7670_regWriteArray()

```
ov7670_status_t ov7670_regWriteArray (
            const ov7670_reg_t * reg )
```

Write an array of register values in the OV7670.

This function writes the registers in the OV7670 using the I2C interface. The I2C interface must be configured before using this function. This function writes the registers based on address-value mappings in the ov7670_regs.h file.

Parameters

reg	the register mapping to write.

Returns

a status code of the type ov7670_status_t

4.9.2.16 ov7670_Transfer()

```
ov7670_status_t ov7670_Transfer ( )
```

Transfer an image from SDRAM to the host.

This function uses the logger to transfer a full image from SDRAM to the host computer. This function will not work if the logger is disabled (directive __LOG needs to be on).

Returns

a status code of the type ov7670_status_t

4.10 inc/project/ov7670_regs.h File Reference

Register configuration definitions for the ov7670 camera module.

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

Macros

- #define **REG_GAIN** 0x00 /* Gain lower 8 bits (rest in vref) */
- #define **REG_BLUE** 0x01 /* blue gain */
- #define **REG_RED** 0x02 /* red gain */
- #define **REG_VREF** 0x03 /* Pieces of GAIN, VSTART, VSTOP */
- #define **REG_COM1** 0x04 /* Control 1 */
- #define **COM1_CCIR656** 0x40 /* CCIR656 enable */
- #define **REG_BAVE** 0x05 /* U/B Average level */
- #define **REG_GbAVE** 0x06 /* Y/Gb Average level */
- #define **REG_AECHH** 0x07 /* AEC MS 5 bits */
- #define **REG_RAVE** 0x08 /* V/R Average level */
- #define **REG_COM2** 0x09 /* Control 2 */
- #define **COM2_SSLEEP** 0x10 /* Soft sleep mode */
- #define **REG_PID** 0x0a /* Product ID MSB */
- #define **REG_VER** 0x0b /* Product ID LSB */
- #define **REG_COM3** 0x0c /* Control 3 */
- #define **COM3_SWAP** 0x40 /* Byte swap */
- #define **COM3_SCALEEN** 0x08 /* Enable scaling */
- #define **COM3_DCWEN** 0x04 /* Enable downsamp/crop/window */
- #define **REG_COM4** 0x0d /* Control 4 */
- #define **REG_COM5** 0x0e /* All "reserved" */
- #define **REG_COM6** 0x0f /* Control 6 */
- #define **REG_AECH** 0x10 /* More bits of AEC value */
- #define **REG_CLKRC** 0x11 /* Clocl control */
- #define **CLK_EXT** 0x40 /* Use external clock directly */
- #define **CLK_SCALE** 0x3f /* Mask for internal clock scale */
- #define **REG_COM7** 0x12 /* Control 7 */
- #define **COM7_RESET** 0x80 /* Register reset */
- #define **COM7_FMT_MASK** 0x38
- #define **COM7_FMT_VGA** 0x00
- #define **COM7_FMT_CIF** 0x20 /* CIF format */
- #define **COM7_FMT_QVGA** 0x10 /* QVGA format */
- #define **COM7_FMT_QCIF** 0x08 /* QCIF format */
- #define **COM7_RGB** 0x04 /* bits 0 and 2 - RGB format */
- #define **COM7_YUV** 0x00 /* YUV */
- #define **COM7_BAYER** 0x01 /* Bayer format */
- #define **COM7_PBAYER** 0x05 /* "Processed bayer" */
- #define **REG_COM8** 0x13 /* Control 8 */
- #define **COM8_FASTAEC** 0x80 /* Enable fast AGC/AEC */
- #define **COM8_AECSTEP** 0x40 /* Unlimited AEC step size */
- #define **COM8_BFILT** 0x20 /* Band filter enable */
- #define **COM8_AGC** 0x04 /* Auto gain enable */
- #define **COM8_AWB** 0x02 /* White balance enable */
- #define **COM8_AEC** 0x01 /* Auto exposure enable */
- #define **REG_COM9** 0x14 /* Control 9 - gain ceiling */
- #define **REG_COM10** 0x15 /* Control 10 */
- #define **COM10_HSYNC** 0x40 /* HSYNC instead of HREF */
- #define **COM10_PCLK_HB** 0x20 /* Suppress PCLK on horiz blank */
- #define **COM10_HREF_REV** 0x08 /* Reverse HREF */
- #define **COM10_VS_LEAD** 0x04 /* VSYNC on clock leading edge */
- #define **COM10_VS_NEG** 0x02 /* VSYNC negative */
- #define **COM10_HS_NEG** 0x01 /* HSYNC negative */
- #define **REG_HSTART** 0x17 /* Horiz start high bits */
- #define **REG_HSTOP** 0x18 /* Horiz stop high bits */

- #define **REG_VSTART** 0x19 /* Vert start high bits */
- #define **REG_VSTOP** 0x1a /* Vert stop high bits */
- #define **REG_PSHFT** 0x1b /* Pixel delay after HREF */
- #define **REG_MIDH** 0x1c /* Manuf. ID high */
- #define **REG_MIDL** 0x1d /* Manuf. ID low */
- #define **REG_MVFP** 0x1e /* Mirror / vflip */
- #define **MVFP_MIRROR** 0x20 /* Mirror image */
- #define **MVFP_FLIP** 0x10 /* Vertical flip */
- #define **REG_AEW** 0x24 /* AGC upper limit */
- #define **REG_AEB** 0x25 /* AGC lower limit */
- #define **REG_VPT** 0x26 /* AGC/AEC fast mode op region */
- #define **REG_HSYST** 0x30 /* HSYNC rising edge delay */
- #define **REG_HSYEN** 0x31 /* HSYNC falling edge delay */
- #define **REG_HREF** 0x32 /* HREF pieces */
- #define **REG_TSLB** 0x3a /* lots of stuff */
- #define **TSLB_YLAST** 0x04 /* UYVY or VYUY - see com13 */
- #define **REG_COM11** 0x3b /* Control 11 */
- #define **COM11_NIGHT** 0x80 /* NIght mode enable */
- #define **COM11_NMFR** 0x60 /* Two bit NM frame rate */
- #define **COM11_HZAUTO** 0x10 /* Auto detect 50/60 Hz */
- #define **COM11_50HZ** 0x08 /* Manual 50Hz select */
- #define **COM11_EXP** 0x02
- #define **REG_COM12** 0x3c /* Control 12 */
- #define **COM12_HREF** 0x80 /* HREF always */
- #define **REG_COM13** 0x3d /* Control 13 */
- #define **COM13_GAMMA** 0x80 /* Gamma enable */
- #define **COM13_UVSAT** 0x40 /* UV saturation auto adjustment */
- #define **COM13_UVSWAP** 0x01 /* V before U - w/TSLB */
- #define **REG_COM14** 0x3e /* Control 14 */
- #define **COM14_DCWEN** 0x10 /* DCW/PCLK-scale enable */
- #define **REG_EDGE** 0x3f /* Edge enhancement factor */
- #define **REG_COM15** 0x40 /* Control 15 */
- #define **COM15_R10F0** 0x00 /* Data range 10 to F0 */
- #define **COM15_R01FE** 0x80 /* 01 to FE */
- #define **COM15_R00FF** 0xc0 /* 00 to FF */
- #define **COM15_RGB565** 0x10 /* RGB565 output */
- #define **COM15_RGB555** 0x30 /* RGB555 output */
- #define **REG_COM16** 0x41 /* Control 16 */
- #define **COM16_AWBGAIN** 0x08 /* AWB gain enable */
- #define **REG_COM17** 0x42 /* Control 17 */
- #define **COM17_AECWIN** 0xc0 /* AEC window - must match COM4 */
- #define **COM17_CBAR** 0x08 /* DSP Color bar */
- #define **REG_CMATRIX_BASE** 0x4f
- #define **CMATRIX_LEN** 6
- #define **REG_CMATRIX_SIGN** 0x58
- #define **REG_BRIGHT** 0x55 /* Brightness */
- #define **REG_CONTRAS** 0x56 /* Contrast control */
- #define **REG_GFIX** 0x69 /* Fix gain control */
- #define **REG_DBLV** 0x6b /* PLL control an debugging */
- #define **DBLV_BYPASS** 0x00 /* Bypass PLL */
- #define **DBLV_X4** 0x01 /* clock x4 */
- #define **DBLV_X6** 0x10 /* clock x6 */
- #define **DBLV_X8** 0x11 /* clock x8 */
- #define **REG_REG76** 0x76 /* OV's name */
- #define **R76_BLKPCOR** 0x80 /* Black pixel correction enable */

- #define **R76_WHTPCOR** 0x40 /∗ White pixel correction enable ∗/
- #define **REG_RGB444** 0x8c /∗ RGB 444 control ∗/
- #define **R444_ENABLE** 0x02 /∗ Turn on RGB444, overrides 5x5 ∗/
- #define **R444_RGBX** 0x01 /∗ Empty nibble at end ∗/
- #define **REG_HAECC1** 0x9f /∗ Hist AEC/AGC control 1 ∗/
- #define **REG_HAECC2** 0xa0 /∗ Hist AEC/AGC control 2 ∗/
- #define **REG_BD50MAX** 0xa5 /∗ 50hz banding step limit ∗/
- #define **REG_HAECC3** 0xa6 /∗ Hist AEC/AGC control 3 ∗/
- #define **REG_HAECC4** 0xa7 /∗ Hist AEC/AGC control 4 ∗/
- #define **REG_HAECC5** 0xa8 /∗ Hist AEC/AGC control 5 ∗/
- #define **REG_HAECC6** 0xa9 /∗ Hist AEC/AGC control 6 ∗/
- #define **REG_HAECC7** 0xaa /∗ Hist AEC/AGC control 7 ∗/
- #define **REG_BD60MAX** 0xab /∗ 60hz banding step limit ∗/

Functions

- struct **__attribute__** ((packed)) ov7670_reg_s

Variables

- **ov7670_reg_t**

4.10.1 Detailed Description

Register configuration definitions for the ov7670 camera module.

Taken from Linux kernel's drivers/media/i2c/ov7670.c

Copyright 2006 One Laptop Per Child Association, Inc. Written by Jonathan Corbet with substantial inspiration from Mark McClelland's ovcamchip code.

Copyright 2006-7 Jonathan Corbet corbet@lwn.net

This file may be distributed under the terms of the GNU General Public License, version 2.

Author

Jonathan Corbet
Ben Heberlein

Bug No known bugs.

4.11 inc/project/prof.h File Reference

Function prototypes for the profiler.

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

Macros

- #define prof_Profile(x, msg) x;
  The main profile function.

4.11.1 Detailed Description

Function prototypes for the profiler.

This contains the prototypes and macros for the profiler module.

Author

Ben Heberlein

Bug No known bugs.

4.11.2 Macro Definition Documentation

4.11.2.1 prof_Profile

```
#define prof_Profile(
              x,
              msg ) x;
```

The main profile function.

This macro implements a function that will profile the piece of code it surrounds. If the compiler directive __PROF is not set, the macro defaults to just executing the piece of code.

The macro just wraps the code segment under test with the prof_start() and prof_stop() function defined earlier.

Example usage: For a single function or statement prof_Profile(xxx_functionToProfile(), "Benchmark identifier\0");

Several statements or functions prof_Profile(xxx_functionToProfile1(); xxx_functionToProfile2(); uint32_t someVariable = 100; // ...etc , "Benchmark identifier\0");

For finer control of error handling, consider calling the start and stop functions directly.

Parameters

x	statements to execute
msg	benchmark message ('\0' terminated)

Returns

status code PROF_INFO_OK in all cases.

4.12 inc/project/sdram.h File Reference

Function prototypes for the SDRAM functions.

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

Macros

- #define **SDRAM_BASEADDR** 0xD0100000
- #define **SDRAM_IMAGEADDR** SDRAM_BASEADDR

Functions

- sdram_status_t sdram_Init ()
  Initialize the SDRAM interface.
- sdram_status_t sdram_write (uint32_t ∗buf, uint32_t addr, uint32_t size)
  Writes to the SDRAM.
- sdram_status_t sdram_read (uint32_t ∗buf, uint32_t addr, uint32_t size)
  Reads from the SDRAM.

4.12.1 Detailed Description

Function prototypes for the SDRAM functions.

This contains the prototypes for the SDRAM functions based on compiler directives.

If __STM32F429I_DISCOVERY is defined, we use the implementation of the SDRAM interface from the discovery board libraries.

Author

Ben Heberlein

Bug No known bugs.

4.12.2 Function Documentation

4.12.2.1 sdram_Init()

```
sdram_status_t sdram_Init ( )
```

Initialize the SDRAM interface.

This will initialize the SDRAM interface either using the discovery board library or a custom implementations

Returns

a status code of the type sdram_status_t

4.12.2.2 sdram_read()

```
sdram_status_t sdram_read (
            uint32_t * buf,
            uint32_t addr,
            uint32_t size )
```

Reads from the SDRAM.

This will read fram the sdram at a specified address with the supplied buffer, address, and transfer size.

Parameters

buf	the buffer to write to
addr	the address to read from
size	size in bytes to read

Returns

a status code of the type sdram_status_t

4.12.2.3 sdram_write()

```
sdram_status_t sdram_write (
            uint32_t * buf,
            uint32_t addr,
            uint32_t size )
```

Writes to the SDRAM.

This will write to the sdram at a specified address using the supplied buffer, address, and transfer size.

Parameters

buf	the buffer to read from
addr	the address to write to
size	size in bytes to write

Returns

a status code of the type sdram_status_t

4.13 inc/project/template.h File Reference

Function prototypes for the template file.

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

Functions

- uint8_t template_privateFunction (uint8_t ch)

  Short description of the function.
- uint8_t template_PublicFunction (uint8_t ch)

  Short description of the function.

4.13.1 Detailed Description

Function prototypes for the template file.

This contains the prototypes, macros, constants, and global variables for the template file.

Author

Ben Heberlein

Bug No known bugs.

4.13.2 Function Documentation

4.13.2.1 template_privateFunction()

```
uint8_t template_privateFunction (
            uint8_t ch )
```

Short description of the function.

Long description of the function, and certain important use cases associated with the function.

Parameters

ch	the input byte

Returns

The return byte

4.13.2.2 template_PublicFunction()

```
uint8_t template_PublicFunction (
            uint8_t ch )
```

Short description of the function.

Long description of the function, and certain important use cases associated with the function.

Parameters

ch	the input byte

Returns

The return byte

4.14 inc/project/wifi.h File Reference

Function prototypes for the wifi driver.

```
#include "err.h"
#include "mod.h"
#include <stdint.h>
```

Macros

- #define **WIFI_MAXMSGSIZE** 255
- #define **WIFI_MAXDATASIZE** 16777216

Functions

- struct **__attribute__** ((packed)) wifi_packet_s
- wifi_status_t wifi_Send (mod_t mod, gen_status_t status, char *msg, uint32_t len, uint8_t *data)

  Send data over wifi.
- wifi_status_t wifi_Init ()

  Wifi initialization.

Variables

- **wifi_packet_t**

4.14.1 Detailed Description

Function prototypes for the wifi driver.

This contains the prototypes, macros, constants, and global variables for the wifi driver. Compiler directives choose the wifi module.

Author

Ben Heberlein

Bug No known bugs.

4.14.2 Function Documentation

4.14.2.1 wifi_Init()

```
wifi_status_t wifi_Init ( )
```

Wifi initialization.

This function initializes the wifi module.

Returns

a status code of type wifi_status_t

4.14.2.2 wifi_Send()

```
wifi_status_t wifi_Send (
              mod_t mod,
              gen_status_t status,
              char * msg,
              uint32_t len,
              uint8_t * data )
```

Send data over wifi.

This function sends data over the wifi module.

Parameters

module	of the type mod_t
status	code to send
msg	message as a pointer to uint8_t
len	optional data length paramter
data	optional pointer to data bufffer

Returns

a status code of type wifi_status_t

4.15 inc/test/test.h File Reference

Declarations for the test framework.

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

Functions

- test_status_t test_log ()

  logging functions
- char * **test_log_Init** ()
- char * **test_log_Log** ()
- test_status_t test_prof ()

  profiler functions
- char * **test_prof_Init** ()
- char * **test_prof_Profile** ()

4.15.1 Detailed Description

Declarations for the test framework.

Defines several macros that handle unit testing. Test files should be in the src/test folder, with names corresponding to the module under test. For example, for unit tests of logger functions, there would be a file test_log.c in src/test.

Inside the module test file, there should be tests corresponding to each function under test. For example, if there were functions in log.c called log_someFunction1(), log_someFunction2(), log_someFunction3(), we would have associated test function in test_log.c test_log_someFunction1(), test_log_someFunction2(), test_log_some↩ Function3(), ...etc.

These functions report output with the logger. Make sure both __TEST and __LOG are defined in the build system before using these functions.

This test framework is based off of MinUnit - a minimal unit testing framework for C, found at http://www.↩ jera.com/techinfo/jtns/jtn002.html MinUnit is released for any purpose by the author.

Author

Ben Heberlein

Bug No known bugs.

Index