CH-53K

1.0.0

Generated by Doxygen 1.12.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 PwmStruct Struct Reference

```
#include <stm32l412xx-bsp.h>
```

Data Fields

- uint8_t is_running
- uint32_t pulse_width

3.1.1 Field Documentation

3.1.1.1 is_running

```
uint8_t is_running
```

3.1.1.2 pulse_width

```
uint32_t pulse_width
```

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

- C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/stm32l412xx-bsp.h

3.2 TimerStruct Struct Reference

```
#include <stm32l412xx-bsp.h>
```

Data Fields

- uint8_t is_running
- uint32_t time

3.2.1 Field Documentation

3.2.1.1 is_running

```
uint8_t is_running
```

3.2.1.2 time

```
uint32_t time
```

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

- C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/stm32l412xx-bsp.h

Chapter 4

File Documentation

4.1 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/button_handler.h File Reference

```
#include <stdint.h>
#include "stm32l412xx-bsp.h"
```

Functions

- GPIO_PinState IsDimPressed (void)
- GPIO_PinState IsBrightPressed (void)

4.1.1 Function Documentation

4.1.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
            void )
```

4.1.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
            void )
```

4.2 button_handler.h

Go to the documentation of this file.

```
// ****************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ****************************************************************************
// ****************************************************************************
//
// Filename: button_handler.h
//
// Description: Returns the button state of the three board buttons
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes: Depends on the board support package bsp for GPIO_PinState
//
// ****************************************************************************

#ifndef INC_button_handlerh
    #define INC_button_handlerh

    #include <stdint.h>

    #include "stm321412xx-bsp.h"

    /* Return state of buttons */
    GPIO_PinState IsDimPressed    ( void );
    GPIO_PinState IsBrightPressed ( void );

#endif /* INC_button_handlerh */
```

4.3 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/current_handler.h File Reference

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

Enumerations

- enum CurrentRange_e { CurrentNormal = 0 , CurrentHigh = 1 , CurrentError = 2 }

Functions

- uint16_t GetCurrent (void)
- CurrentRange_e GetCurrentRange (void)

4.3.1 Enumeration Type Documentation

4.3.1.1 CurrentRange_e

```
enum CurrentRange_e
```

Enumerator

CurrentNormal	
CurrentHigh	
CurrentError	

4.3.2 Function Documentation

4.3.2.1 GetCurrent()

```
uint16_t GetCurrent (
             void )
```

4.3.2.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
             void )
```

4.4 current_handler.h

Go to the documentation of this file.

```
// ***************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ***************************************************************************
// ***************************************************************************
//
// Filename: current_handler.h
//
// Description: Handles getting current and reporting values.
//
// Revision History:
// Date        - Name          -  Ver -  Remarks
// 08/05/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes:
//
// ***************************************************************************

#ifndef INC_current_handlerh
    #define INC_current_handlerh

    #include <stdint.h>

    /* Current Range Enum */
    typedef enum
    {
        CurrentNormal   = 0,
        CurrentHigh     = 1,
        CurrentError    = 2
    } CurrentRange_e;

    /* Get Current */
    uint16_t GetCurrent( void ); // in dA
    CurrentRange_e GetCurrentRange( void );

#endif /* INC_current_handlerh */
```

4.5 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/delay_handler.h File Reference

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

Functions

- void StartDelayCounter (void)
- void RestartDelayCounter (void)
- uint8_t DelayHit (uint32_t delay_ms)
- uint16_t BrightnessDelay (int8_t brightness)

4.5.1 Function Documentation

4.5.1.1 BrightnessDelay()

```
uint16_t BrightnessDelay (
            int8_t brightness)
```

4.5.1.2 DelayHit()

```
uint8_t DelayHit (
            uint32_t delay_ms)
```

4.5.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
            void )
```

4.5.1.4 StartDelayCounter()

```
void StartDelayCounter (
            void )
```

4.6 delay_handler.h

Go to the documentation of this file.

```
// ****************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ****************************************************************************
// ****************************************************************************
//
// Filename: delay_handler.h
//
// Description: Handles system counters and delays
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes:
//
// ****************************************************************************

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef INC_delay_handlerh
    #define INC_delay_handlerh

    #include <stdint.h>

    /* Check if delay has been hit */
    void    StartDelayCounter(void);     // start the counter
    void    RestartDelayCounter(void);   // restart the counter
    uint8_t DelayHit(uint32_t delay_ms); // is delay in ms hit
    /* Get ms delay for given brightness */
    uint16_t BrightnessDelay(int8_t brightness);

#endif /* INC_delay_handlerh */
```

4.7 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/fram.h File Reference

This module contains definitions and structures to support SPI FRAM operations.

```
#include "stm32l412xx-bsp.h"
```

Enumerations

- enum OPCODE_COMMANDS {
  OC_WREN = 6 , OC_WRDI = 4 , OC_RDSR = 5 , OC_WRSR = 1 ,
  OC_READ = 3 , OC_WRITE = 2 }
- enum STATUS_REGISTER { SR_WEL = 0x2 , SR_BP0 = 0x4 , SR_BP1 = 0x8 , SR_WPEN = 0x80 }
- enum WRITE_PROTECT_STATE { WPS_PROTECTED = 0 , WPS_WRITEABLE = 1 }
- enum CHIP_SELECT_STATE { CSS_ASSERT = 0 , CSS_RELEASE = 1 }

Functions

- void framWriteProtect (WRITE_PROTECT_STATE state)
- void framChipSelect (CHIP_SELECT_STATE state)
- void framReadSr (unsigned char ∗srP)
- void framWriteSr (unsigned char sr)
- void framWriteDisable (void)
- void framWriteEnable (void)
- void framReadMemory (unsigned short addr, unsigned char ∗rdBufP, unsigned short len)
- void framWriteMemory (unsigned short addr, const unsigned char ∗const wrBufP, unsigned short len)
- uint8_t framTest (void)

4.7.1 Detailed Description

This module contains definitions and structures to support SPI FRAM operations.

Attention

Copyright (c) 2022, 2023 Luminator, An LTG Company All rights reserved. Any use without the prior written consent of Luminator, An LTG Company is strictly prohibited.

Revision History: Date Name Ver Remarks
04/09/2023 Mark Lane 0 Original Version
Notes:

4.7.2 Enumeration Type Documentation

4.7.2.1 CHIP_SELECT_STATE

```
enum CHIP_SELECT_STATE
```

chip select state

Enumerator

CSS_ASSERT	chip select disabled
CSS_RELEASE	chip select enabled

4.7.2.2 OPCODE_COMMANDS

`enum OPCODE_COMMANDS`

opcode command

Enumerator

OC_WREN	set write enable latch
OC_WRDI	write disable
OC_RDSR	read status register
OC_WRSR	write status register
OC_READ	read memory data
OC_WRITE	write memory data

4.7.2.3 STATUS_REGISTER

`enum STATUS_REGISTER`

status register

Enumerator

SR_WEL	write-enable latch
SR_BP0	block protect bit 0
SR_BP1	block protect bit 1
SR_WPEN	enable write protect pin

4.7.2.4 WRITE_PROTECT_STATE

`enum WRITE_PROTECT_STATE`

write protect state

Enumerator

WPS_PROTECTED	write protected
WPS_WRITEABLE	write enabled

4.7.3 Function Documentation

4.7.3.1 framChipSelect()

```
void framChipSelect (
            CHIP_SELECT_STATE state)
```

This routine sets the chip select pin to the correct "state".

Parameters

in	*state*	assert = 0, release = 1
out	*none*	

4.7.3.2 framReadMemory()

```
void framReadMemory (
            unsigned short addr,
            unsigned char * rdBufP,
            unsigned short len)
```

This routine reads FRAM memory starting at "addr" for "len" byte(s). The FRAM read data is stored at the pointer referenced by "rdBufP".

Parameters

in	*addr*	FRAM memory address to read data
in	*rdBufP*	destination pointer to store read data
in	*len*	number of byte(s) to read
out	*none*	

4.7.3.3 framReadSr()

```
void framReadSr (
            unsigned char * srP)
```

This routine reads the FRAM status register

Parameters

in	*destination*	pointer for FRAM status register
out	*none*	

4.7.3.4 framTest()

```
uint8_t framTest (
            void )
```

This routine is a test function for FRAM access. It writes "TLEN" bytes of an incrementing pattern into FRAM at address "TADD". It reads the same length into a buffer and verifies the pattern.

Parameters

out	*pass*	= 1, fail = 0

4.7.3.5 framWriteDisable()

```
void framWriteDisable (
            void )
```

This routine resets the write enable latch

Parameters

out	*none*	

4.7.3.6 framWriteEnable()

```
void framWriteEnable (
             void )
```

This routine sets the write enable latch

Parameters

out	*none*	

4.7.3.7 framWriteMemory()

```
void framWriteMemory (
             unsigned short addr,
             const unsigned char *const wrBufP,
             unsigned short len)
```

This routine writes FRAM memory starting at "addr" for "len" byte(s). The data referenced by the pointer "wrBufP" is written into FRAM memory.

Parameters

in	*addr*	FRAM memory address to write data
in	*wrBufP*	pointer to data to write
in	*len*	number of byte(s) to write
out	*none*	

4.7.3.8 framWriteProtect()

```
void framWriteProtect (
             WRITE_PROTECT_STATE state)
```

This routine sets the hardware write protect pin to the correct "state".

Parameters

in	*state*	disable = 0, enable = 1
out	*none*	

4.7.3.9 framWriteSr()

```
void framWriteSr (
             unsigned char sr)
```

This routine writes the FRAM status register

Parameters

in	*data*	value written to FRAM status register
out	*none*	

4.8 fram.h

Go to the documentation of this file.

```
#ifndef _FRAM_H_
#define _FRAM_H_

#include "stm321412xx-bsp.h"

/* Definition */
typedef enum {

  OC_WREN  = 6 ,
  OC_WRDI  = 4 ,
  OC_RDSR  = 5 ,
  OC_WRSR  = 1 ,
  OC_READ  = 3 ,
  OC_WRITE = 2 ,
} OPCODE_COMMANDS ;


typedef enum {

  SR_WEL  = 0x2  ,
  SR_BP0  = 0x4  ,
  SR_BP1  = 0x8  ,
  SR_WPEN = 0x80 ,
} STATUS_REGISTER ;


typedef enum {

  WPS_PROTECTED = 0 ,
  WPS_WRITEABLE = 1 ,
} WRITE_PROTECT_STATE ;


typedef enum {

  CSS_ASSERT  = 0 ,
  CSS_RELEASE = 1 ,
} CHIP_SELECT_STATE ;


/* Prototype Definition */

void framWriteProtect( WRITE_PROTECT_STATE state  ) ;

void framChipSelect( CHIP_SELECT_STATE state  ) ;


void framReadSr( unsigned char *srP ) ;

void framWriteSr( unsigned char sr ) ;


void framWriteDisable( void ) ;

void framWriteEnable( void )  ;


void framReadMemory ( unsigned short addr, unsigned char *rdBufP, unsigned short len ) ;

void framWriteMemory( unsigned short addr, const unsigned char* const wrBufP, unsigned short len ) ;


uint8_t framTest( void ) ;


#endif
```

4.9 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/logger.h File Reference

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

Functions

- void LogString (const char ∗const string, uint8_t write_beginning)

- void LogNumber (const int32_t number, uint8_t write_beginning)
- void ReadLog (const uint32_t address, char ∗string, const uint32_t bytes)

4.9.1 Function Documentation

4.9.1.1 LogNumber()

```
void LogNumber (
            const int32_t number,
            uint8_t write_beginning)
```

4.9.1.2 LogString()

```
void LogString (
            const char *const string,
            uint8_t write_beginning)
```

4.9.1.3 ReadLog()

```
void ReadLog (
            const uint32_t address,
            char * string,
            const uint32_t bytes)
```

4.10 logger.h

Go to the documentation of this file.

```
// ****************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ****************************************************************************
// ****************************************************************************
//
// Filename: logger.h
//
// Description: Handles logging and reading of data to memory
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
// 08/05/2024 - Austin Green -  1.1 -  Added Log Number
//
// Notes:
//
// ****************************************************************************

#ifndef INC_loggerh
    #define INC_loggerh

    #include <stdint.h>

    /* Store and Read Logs */
    void LogString( const char* const string, uint8_t write_beginning );
    void LogNumber( const int32_t number, uint8_t write_beginning );
    void ReadLog(  const uint32_t address, char* string, const uint32_t bytes );

#endif /* INC_loggerh */
```

4.11 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/main.h File Reference

```
#include "stm32l412xx-bsp.h"
```

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.11.1 Function Documentation

4.11.1.1 Error_Handler()

```
void Error_Handler (
               void )
```

This function is executed in case of error occurrence.

Return values

None	

4.12 main.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/****
    ******************************************************************************
    * @file           : main.h
    * @brief          : Header for main.c file.
    *                   This file contains the common defines of the application.
    ******************************************************************************
    * @attention
    *
    * Copyright (c) 2024 STMicroelectronics.
    * All rights reserved.
    *
    * This software is licensed under terms that can be found in the LICENSE file
    * in the root directory of this software component.
    * If no LICENSE file comes with this software, it is provided AS-IS.
    *
    ******************************************************************************
    */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef INC_mainh
    #define INC_mainh

    #ifdef __cplusplus
        extern "C" {
    #endif

    /* Includes ------------------------------------------------------------------*/

    /* Private includes ----------------------------------------------------------*/
    /* USER CODE BEGIN Includes */
    #include "stm32l412xx-bsp.h"

    /* USER CODE END Includes */

    /* Exported types ------------------------------------------------------------*/
    /* USER CODE BEGIN ET */

    /* USER CODE END ET */

    /* Exported constants --------------------------------------------------------*/
    /* USER CODE BEGIN EC */

    /* USER CODE END EC */

    /* Exported macro ------------------------------------------------------------*/
    /* USER CODE BEGIN EM */

    /* USER CODE END EM */

    /* Exported functions prototypes ---------------------------------------------*/
    void Error_Handler(void);

    /* USER CODE BEGIN EFP */

    /* USER CODE END EFP */

    /* Private defines -----------------------------------------------------------*/

    /* USER CODE BEGIN Private defines */

    /* USER CODE END Private defines */

    #ifdef __cplusplus
        }
    #endif

#endif /* INC_mainh */
```

4.13 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/my_printf.h File Reference

4.14 my_printf.h

Go to the documentation of this file.

```
// ***************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ***************************************************************************
// ***************************************************************************
//
// Filename: my_printf.h
//
// Description: Prints characters to a terminal for debugging purposes
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes:
//
// ***************************************************************************

// Software tracing with printf()
#ifndef INC_my_printfh
    #define INC_my_printfh

    #ifdef ENABLE_UART_DEBUGGING /* tracing enabled */
        #include <stdio.h>
    #endif /* ENABLE_UART_DEBUGGING */

#endif /* INC_my_printfh */
```

4.15 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/pwm_handler.h File Reference

`#include <stdint.h>`

Macros

- #define BRIGHTNESS_STEPS (50)
- #define HOLD_BRIGHTNESS_JUMP (3)

Functions

- void InitPwm (void)
- void DecreaseBrightness (uint8_t button_held)
- void IncreaseBrightness (uint8_t button_held)
- void SetPwm (void)
- void TurnOffPwm (void)
- int8_t GetBrightness (void)

- void SetBrightness (int8_t brightness)
- uint8_t GetPwm (void)

4.15.1 Macro Definition Documentation

4.15.1.1 BRIGHTNESS_STEPS

```
#define BRIGHTNESS_STEPS (50)
```

4.15.1.2 HOLD_BRIGHTNESS_JUMP

```
#define HOLD_BRIGHTNESS_JUMP (3)
```

4.15.2 Function Documentation

4.15.2.1 DecreaseBrightness()

```
void DecreaseBrightness (
             uint8_t button_held)
```

4.15.2.2 GetBrightness()

```
int8_t GetBrightness (
             void )
```

4.15.2.3 GetPwm()

```
uint8_t GetPwm (
             void )
```

4.15.2.4 IncreaseBrightness()

```
void IncreaseBrightness (
             uint8_t button_held)
```

4.15.2.5 InitPwm()

```
void InitPwm (
             void )
```

4.15.2.6 SetBrightness()

```
void SetBrightness (
             int8_t brightness)
```

4.15.2.7 SetPwm()

```
void SetPwm (
             void )
```

4.15.2.8 TurnOffPwm()

```
void TurnOffPwm (
             void )
```

4.16 pwm_handler.h

Go to the documentation of this file.

```
// ****************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ****************************************************************************
// ****************************************************************************
//
// Filename: pwm_handler.h
//
// Description: Handles the PWM output of the lights. Output is
//                  determined by a Brightness variable that is
//                  controlled by this file.
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes:
//
// ****************************************************************************

#ifndef INC_pwm_handlerh
    #define INC_pwm_handlerh

    #include <stdint.h>

    /* Brightness Steps */
    #define BRIGHTNESS_STEPS        (50)
    #define HOLD_BRIGHTNESS_JUMP    (3)

    /* Initialize system */
    void    InitPwm(void);                               // Init Pwm var
    /* Decrease/Increase brightness levels */
    void    DecreaseBrightness( uint8_t button_held );  // decrease brightness
    void    IncreaseBrightness( uint8_t button_held );  // increase brightness
    /* Set PWM value based on internal brightness */
    void    SetPwm( void );                              // turn on and set PWM
    void    TurnOffPwm( void );                          // turn of PWM
    /* Get or set internal values */
    int8_t  GetBrightness( void );                       // get value of Brightness
    void    SetBrightness( int8_t brightness );          // set value of Brightness
    uint8_t GetPwm( void );                              // get value of current PWM

#endif /* INC_pwm_handlerh */
```

4.17 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/stm32l412xx-bsp.h File Reference

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

Data Structures

- struct PwmStruct
- struct TimerStruct

Macros

- #define THERMISTOR_ADC_Pin 0
- #define THERMISTOR_ADC_GPIO_Port 0
- #define BRIGHT_Pin 0
- #define BRIGHT_GPIO_Port 0
- #define DIM_Pin 0
- #define DIM_GPIO_Port 0
- #define EEPROM_SCK_Pin 0
- #define EEPROM_SCK_GPIO_Port 0
- #define EEPROM_MISO_Pin 0
- #define EEPROM_MISO_GPIO_Port 0

- #define EEPROM_MOSI_Pin 0
- #define EEPROM_MOSI_GPIO_Port 0
- #define VOLTMETER_ADC_Pin 0
- #define VOLTMETER_ADC_GPIO_Port 0
- #define AMPMETER_ADC_Pin 0
- #define AMPMETER_ADC_GPIO_Port 0
- #define PWM_OUT_Pin 0
- #define PWM_OUT_GPIO_Port 0
- #define SPI_WP_Pin 0
- #define SPI_WP_GPIO_Port 0
- #define SPI_NSS_Pin 0
- #define SPI_NSS_GPIO_Port 0
- #define NVIC_PRIORITYGROUP_0 ((uint32_t)0x00000007)
- #define NVIC_PRIORITYGROUP_1 ((uint32_t)0x00000006)
- #define NVIC_PRIORITYGROUP_2 ((uint32_t)0x00000005)
- #define NVIC_PRIORITYGROUP_3 ((uint32_t)0x00000004)
- #define NVIC_PRIORITYGROUP_4 ((uint32_t)0x00000003)
- #define CLK_FREQ_HZ (8000000)
- #define TIM2_CLK_DEV (1)
- #define TIM2_CLK_PRESCALER (8000)

Typedefs

- typedef uint8_t GPIO_PinState

Enumerations

- enum { BUTTON_PRESSED = 1 , BUTTON_UNPRESSED = 0 }
- enum { PIN_SET = 1 , PIN_RESET = 0 }

Functions

- GPIO_PinState ReadDimPin (void)
- GPIO_PinState ReadBrightPin (void)
- void EnablePWM1 (void)
- void DisablePWM1 (void)
- void StartPWM11 (void)
- void StopPWM11 (void)
- void SetPW11 (uint32_t pulse_width)
- void StartTIM2 (void)
- void RestartTIM2 (void)
- uint32_t GetTIM2Cnt (void)
- int16_t GetThermistorValue (void)
- int16_t GetCurrentValue (void)
- int16_t GetVoltageValue (void)
- void enableWriteProtect (void)
- void disableWriteProtect (void)
- void enableChipSelect (void)
- void disableChipSelect (void)
- void transferData (const unsigned char ∗const txData, const uint32_t bytes)
- void receiveData (unsigned char ∗rxData, const uint32_t bytes)
- void sendUARTChar (char c)
- void Error_Handler (void)

    This function is executed in case of error occurrence.

4.17.1 Macro Definition Documentation

4.17.1.1 AMPMETER_ADC_GPIO_Port

```
#define AMPMETER_ADC_GPIO_Port 0
```

4.17.1.2 AMPMETER_ADC_Pin

```
#define AMPMETER_ADC_Pin 0
```

4.17.1.3 BRIGHT_GPIO_Port

```
#define BRIGHT_GPIO_Port 0
```

4.17.1.4 BRIGHT_Pin

```
#define BRIGHT_Pin 0
```

4.17.1.5 CLK_FREQ_HZ

```
#define CLK_FREQ_HZ (8000000)
```

4.17.1.6 DIM_GPIO_Port

```
#define DIM_GPIO_Port 0
```

4.17.1.7 DIM_Pin

```
#define DIM_Pin 0
```

4.17.1.8 EEPROM_MISO_GPIO_Port

```
#define EEPROM_MISO_GPIO_Port 0
```

4.17.1.9 EEPROM_MISO_Pin

```
#define EEPROM_MISO_Pin 0
```

4.17.1.10 EEPROM_MOSI_GPIO_Port

```
#define EEPROM_MOSI_GPIO_Port 0
```

4.17.1.11 EEPROM_MOSI_Pin

```
#define EEPROM_MOSI_Pin 0
```

4.17.1.12 EEPROM_SCK_GPIO_Port

```
#define EEPROM_SCK_GPIO_Port 0
```

4.17.1.13 EEPROM_SCK_Pin

```
#define EEPROM_SCK_Pin 0
```

4.17.1.14 NVIC_PRIORITYGROUP_0

```
#define NVIC_PRIORITYGROUP_0 ((uint32_t)0x00000007)
```

0 bit for pre-emption priority, 4 bits for subpriority

4.17.1.15 NVIC_PRIORITYGROUP_1

```
#define NVIC_PRIORITYGROUP_1 ((uint32_t)0x00000006)
```
1 bit for pre-emption priority, 3 bits for subpriority

4.17.1.16 NVIC_PRIORITYGROUP_2

```
#define NVIC_PRIORITYGROUP_2 ((uint32_t)0x00000005)
```
2 bits for pre-emption priority, 2 bits for subpriority

4.17.1.17 NVIC_PRIORITYGROUP_3

```
#define NVIC_PRIORITYGROUP_3 ((uint32_t)0x00000004)
```
3 bits for pre-emption priority, 1 bit for subpriority

4.17.1.18 NVIC_PRIORITYGROUP_4

```
#define NVIC_PRIORITYGROUP_4 ((uint32_t)0x00000003)
```
4 bits for pre-emption priority, 0 bit for subpriority

4.17.1.19 PWM_OUT_GPIO_Port

```
#define PWM_OUT_GPIO_Port 0
```

4.17.1.20 PWM_OUT_Pin

```
#define PWM_OUT_Pin 0
```

4.17.1.21 SPI_NSS_GPIO_Port

```
#define SPI_NSS_GPIO_Port 0
```

4.17.1.22 SPI_NSS_Pin

```
#define SPI_NSS_Pin 0
```

4.17.1.23 SPI_WP_GPIO_Port

```
#define SPI_WP_GPIO_Port 0
```

4.17.1.24 SPI_WP_Pin

```
#define SPI_WP_Pin 0
```

4.17.1.25 THERMISTOR_ADC_GPIO_Port

```
#define THERMISTOR_ADC_GPIO_Port 0
```

4.17.1.26 THERMISTOR_ADC_Pin

```
#define THERMISTOR_ADC_Pin 0
```

4.17.1.27 TIM2_CLK_DEV

```
#define TIM2_CLK_DEV (1)
```

4.17.1.28 TIM2_CLK_PRESCALER

```
#define TIM2_CLK_PRESCALER (8000)
```

4.17.1.29 VOLTMETER_ADC_GPIO_Port

```
#define VOLTMETER_ADC_GPIO_Port 0
```

4.17.1.30 VOLTMETER_ADC_Pin

```
#define VOLTMETER_ADC_Pin 0
```

4.17.2 Typedef Documentation

4.17.2.1 GPIO_PinState

```
typedef uint8_t GPIO_PinState
```

4.17.3 Enumeration Type Documentation

4.17.3.1 anonymous enum

```
anonymous enum
```

Enumerator

BUTTON_PRESSED	
BUTTON_UNPRESSED	

4.17.3.2 anonymous enum

```
anonymous enum
```

Enumerator

PIN_SET	
PIN_RESET	

4.17.4 Function Documentation

4.17.4.1 disableChipSelect()

```
void disableChipSelect (
             void )
```

4.17.4.2 DisablePWM1()

```
void DisablePWM1 (
             void )
```

4.17.4.3 disableWriteProtect()

```
void disableWriteProtect (
             void )
```

4.17.4.4 enableChipSelect()

```
void enableChipSelect (
             void )
```

4.17.4.5 EnablePWM1()

```
void EnablePWM1 (
             void )
```

4.17.4.6 enableWriteProtect()

```
void enableWriteProtect (
             void )
```

4.17.4.7 Error_Handler()

```
void Error_Handler (
             void )
```

This function is executed in case of error occurrence.

Return values

None	

4.17.4.8 GetCurrentValue()

```
int16_t GetCurrentValue (
             void )
```

4.17.4.9 GetThermistorValue()

```
int16_t GetThermistorValue (
             void )
```

4.17.4.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
             void )
```

4.17.4.11 GetVoltageValue()

```
int16_t GetVoltageValue (
             void )
```

4.17.4.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
             void )
```

4.17.4.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (
             void )
```

4.17.4.14 receiveData()

```
void receiveData (
             unsigned char * rxData,
             const uint32_t bytes)
```

4.17.4.15 RestartTIM2()

```
void RestartTIM2 (
             void )
```

4.17.4.16 sendUARTChar()

```
void sendUARTChar (
            char c)
```

4.17.4.17 SetPW11()

```
void SetPW11 (
            uint32_t pulse_width)
```

4.17.4.18 StartPWM11()

```
void StartPWM11 (
            void )
```

4.17.4.19 StartTIM2()

```
void StartTIM2 (
            void )
```

4.17.4.20 StopPWM11()

```
void StopPWM11 (
            void )
```

4.17.4.21 transferData()

```
void transferData (
            const unsigned char *const txData,
            const uint32_t bytes)
```

4.18 stm32l412xx-bsp.h

Go to the documentation of this file.

```
// ****************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ****************************************************************************
// ****************************************************************************
//
// Filename: stm32l412xx-bsp.h
//
// Description: Board Support Package for STM32L412xx
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes: This uses the Low Level ST API to access the board pins
//
// ****************************************************************************

#ifndef INC_bsph
    #define INC_bsph

    #include <stdint.h>

    /* Private defines -----------------------------------------------------------*/
    #ifdef STM32L412xx

        #include "stm32l4xx_ll_adc.h"
        #include "stm32l4xx_ll_crs.h"
        #include "stm32l4xx_ll_rcc.h"
        #include "stm32l4xx_ll_bus.h"
        #include "stm32l4xx_ll_system.h"
        #include "stm32l4xx_ll_exti.h"
        #include "stm32l4xx_ll_cortex.h"
        #include "stm32l4xx_ll_utils.h"
        #include "stm32l4xx_ll_pwr.h"
        #include "stm32l4xx_ll_dma.h"
        #include "stm32l4xx_ll_spi.h"
        #include "stm32l4xx_ll_tim.h"
        #include "stm32l4xx_ll_usart.h"
        #include "stm32l4xx_ll_gpio.h"

        #if defined(USE_FULL_ASSERT)
            #include "stm32_assert.h"
        #endif /* USE_FULL_ASSERT */

        /* Peripherals */
        #include "adc.h"
        #include "gpio.h"
        #include "spi.h"
        #include "tim.h"

        #ifdef ENABLE_UART_DEBUGGING /* tracing enabled */
            /* Peripherals enabled for UART */
            #include "usart.h"
        #endif /* ENABLE_UART_DEBUGGING */

        #define THERMISTOR_ADC_Pin LL_GPIO_PIN_0
        #define THERMISTOR_ADC_GPIO_Port GPIOA
        #define BRIGHT_Pin LL_GPIO_PIN_1
        #define BRIGHT_GPIO_Port GPIOA
        #define DIM_Pin LL_GPIO_PIN_3
        #define DIM_GPIO_Port GPIOA
        #define EEPROM_SCK_Pin LL_GPIO_PIN_5
        #define EEPROM_SCK_GPIO_Port GPIOA
        #define EEPROM_MISO_Pin LL_GPIO_PIN_6
        #define EEPROM_MISO_GPIO_Port GPIOA
        #define EEPROM_MOSI_Pin LL_GPIO_PIN_7
        #define EEPROM_MOSI_GPIO_Port GPIOA
        #define VOLTMETER_ADC_Pin LL_GPIO_PIN_0
        #define VOLTMETER_ADC_GPIO_Port GPIOB
        #define AMPMETER_ADC_Pin LL_GPIO_PIN_1
        #define AMPMETER_ADC_GPIO_Port GPIOB
        #define PWM_OUT_Pin LL_GPIO_PIN_8
        #define PWM_OUT_GPIO_Port GPIOA
        #define SPI_WP_Pin LL_GPIO_PIN_10
        #define SPI_WP_GPIO_Port GPIOA
        #define SPI_NSS_Pin LL_GPIO_PIN_11
        #define SPI_NSS_GPIO_Port GPIOA

    #else /* STM32L412xx */

        /* Below is for debugging purposes */
        #define THERMISTOR_ADC_Pin          0
        #define THERMISTOR_ADC_GPIO_Port    0
        #define BRIGHT_Pin                  0
        #define BRIGHT_GPIO_Port            0
        #define DIM_Pin                     0
        #define DIM_GPIO_Port               0
        #define EEPROM_SCK_Pin              0
        #define EEPROM_SCK_GPIO_Port        0
        #define EEPROM_MISO_Pin             0
        #define EEPROM_MISO_GPIO_Port       0
        #define EEPROM_MOSI_Pin             0
        #define EEPROM_MOSI_GPIO_Port       0
        #define VOLTMETER_ADC_Pin           0
        #define VOLTMETER_ADC_GPIO_Port     0
        #define AMPMETER_ADC_Pin            0
        #define AMPMETER_ADC_GPIO_Port      0
        #define PWM_OUT_Pin                 0
        #define PWM_OUT_GPIO_Port           0
        #define SPI_WP_Pin                  0
        #define SPI_WP_GPIO_Port            0
        #define SPI_NSS_Pin                 0
        #define SPI_NSS_GPIO_Port           0

        typedef struct
        {
            uint8_t is_running;
            uint32_t pulse_width;
        } PwmStruct;

        typedef struct
        {
            uint8_t is_running;
            uint32_t time;
        } TimerStruct;

    #endif /* STM32L412xx */

    /* Interrupt Handlers */
    #ifndef NVIC_PRIORITYGROUP_0
        #define NVIC_PRIORITYGROUP_0         ((uint32_t)0x00000007)
        #define NVIC_PRIORITYGROUP_1         ((uint32_t)0x00000006)
        #define NVIC_PRIORITYGROUP_2         ((uint32_t)0x00000005)
        #define NVIC_PRIORITYGROUP_3         ((uint32_t)0x00000004)
        #define NVIC_PRIORITYGROUP_4         ((uint32_t)0x00000003)
    #endif

    /* Button Defines */
    typedef uint8_t GPIO_PinState;
    enum { BUTTON_PRESSED = 1, BUTTON_UNPRESSED = 0};
    enum { PIN_SET = 1, PIN_RESET = 0};

    /* Clock frequency Values */
    #define CLK_FREQ_HZ  (8000000)
    #define TIM2_CLK_DEV  (1)
    #define TIM2_CLK_PRESCALER  (8000)

    /* Returns button state */
    GPIO_PinState ReadDimPin( void );
    GPIO_PinState ReadBrightPin( void );
    /* PWM Outputs */
    void EnablePWM1( void );
    void DisablePWM1( void );
    void StartPWM11( void );
    void StopPWM11( void );
    void SetPW11( uint32_t pulse_width );
    /* Timers */
    void StartTIM2( void );
    void RestartTIM2( void );
    uint32_t GetTIM2Cnt( void );
    /* Thermistor */
    int16_t GetThermistorValue( void );
    /* Current Sensing */
    int16_t GetCurrentValue( void );
    /* Voltage Sensing */
    int16_t GetVoltageValue( void );
    /* Logging */
    void enableWriteProtect( void );
    void disableWriteProtect( void );
    void enableChipSelect( void );
    void disableChipSelect( void );
    void transferData( const unsigned char* const txData, const uint32_t bytes );
    void receiveData( unsigned char* rxData, const uint32_t bytes );
    /* UART Output */
    void sendUARTChar(char c);
    /* Errors */
    void Error_Handler(void);


#endif /* INC_bsph */
```

4.19 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/temperature_handler.h File Reference

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

Enumerations

- enum TemperatureRange_e { TempCool = 0 , TempWarm = 1 , TempHot = 2 }

Functions

- int32_t GetTemperature (void)
- TemperatureRange_e GetTemperatureRange (void)

4.19.1 Enumeration Type Documentation

4.19.1.1 TemperatureRange_e

```
enum TemperatureRange_e
```

Enumerator

TempCool	

Enumerator

TempWarm	
TempHot	

4.19.2 Function Documentation

4.19.2.1 GetTemperature()

```
int32_t GetTemperature (
            void )
```

4.19.2.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
            void )
```

4.20 temperature_handler.h

Go to the documentation of this file.

```
// ***************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ***************************************************************************
// ***************************************************************************
//
// Filename: temperature_handler.h
//
// Description: Handles getting this temperature and transitioning
//                between temperature states.
//
// Revision History:
// Date        - Name         -  Ver -  Remarks
// 07/31/2024 - Austin Green -  1.0 -  Initial Document
// 08/05/2024 - Austin Green -  1.1 -  Refactor to not use floats
//
// Notes:
//
// ***************************************************************************

#ifndef INC_temperature_handlerh
    #define INC_temperature_handlerh

    #include <stdint.h>

    /* Temperature Range Enum */
    typedef enum
    {
        TempCool     = 0,
        TempWarm     = 1,
        TempHot      = 2
    } TemperatureRange_e;

    /* Get Temperature */
    int32_t GetTemperature( void ); // in dC
    TemperatureRange_e GetTemperatureRange( void );

#endif /* INC_temperature_handlerh */
```

4.21 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/voltage_handler.h File Reference

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

Enumerations

- enum VoltageRange_e {
  VoltageNormal = 0 , VoltageLow = 1 , VoltageHigh = 2 , VoltageErrorLow = 3 ,
  VoltageErrorHigh = 4 }

Functions

- uint16_t GetVoltage (void)
- VoltageRange_e GetVoltageRange (void)

4.21.1 Enumeration Type Documentation

4.21.1.1 VoltageRange_e

```
enum VoltageRange_e
```

Enumerator

VoltageNormal	
VoltageLow	
VoltageHigh	
VoltageErrorLow	
VoltageErrorHigh	

4.21.2 Function Documentation

4.21.2.1 GetVoltage()

```
uint16_t GetVoltage (
             void )
```

4.21.2.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
             void )
```

4.22 voltage_handler.h

Go to the documentation of this file.

```
// ****************************************************************************
// Copyright © 2007 Luminator Mark IV
// All rights reserved.
// Any use without the prior written consent of Luminator Mark IV
// is strictly prohibited.
// ****************************************************************************
// ****************************************************************************
//
// Filename: voltage_handler.h
//
// Description: Handles getting voltage and reporting values.
//
// Revision History:
// Date       - Name         -  Ver -  Remarks
// 08/05/2024 - Austin Green -  1.0 -  Initial Document
//
// Notes:
//
// ****************************************************************************

#ifndef INC_voltage_handlerh
    #define INC_voltage_handlerh

    #include <stdint.h>

    /* Voltage Range Enum */
    typedef enum
    {
        VoltageNormal       = 0,
        VoltageLow          = 1,
        VoltageHigh         = 2,
        VoltageErrorLow     = 3,
        VoltageErrorHigh    = 4
    } VoltageRange_e;

    /* Get Voltage */
    uint16_t GetVoltage( void ); // in dV
    VoltageRange_e GetVoltageRange( void );

#endif /* INC_voltage_handlerh */
```

4.23 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/button_handler.c File Reference

```
#include "button_handler.h"
#include "stm32l412xx-bsp.h"
```

Functions

- GPIO_PinState IsDimPressed (void)
- GPIO_PinState IsBrightPressed (void)

4.23.1 Function Documentation

4.23.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
            void )
```

4.23.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
            void )
```

4.24 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/current_handler.c File Reference

```
#include <stdio.h>
#include "current_handler.h"
#include "stm32l412xx-bsp.h"
#include "logger.h"
```

Functions

- uint16_t GetCurrent (void)
- CurrentRange_e GetCurrentRange (void)

Variables

- const uint16_t RawTodAmps = (1)
- const uint16_t dAmpsToRaw = (1)
- const uint16_t CurrentHighThreshold_dA = 35u
- const uint16_t CurrentErrorThreshold_dA = 40u

4.24.1 Function Documentation

4.24.1.1 GetCurrent()

```
uint16_t GetCurrent (
            void )
```

4.24.1.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
            void )
```

4.24.2 Variable Documentation

4.24.2.1 CurrentErrorThreshold_dA

```
const uint16_t CurrentErrorThreshold_dA = 40u
```

4.24.2.2 CurrentHighThreshold_dA

```
const uint16_t CurrentHighThreshold_dA = 35u
```

4.24.2.3 dAmpsToRaw

```
const uint16_t dAmpsToRaw = (1)
```

4.24.2.4 RawTodAmps

```
const uint16_t RawTodAmps = (1)
```

4.25 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/delay_handler.c File Reference

```
#include "delay_handler.h"
#include "stm32l412xx-bsp.h"
```

Functions

- void StartDelayCounter (void)
- void RestartDelayCounter (void)
- uint8_t DelayHit (uint32_t delay_ms)
- uint16_t BrightnessDelay (int8_t brightness)

Variables

- const float Tim2ClkKhz = (CLK_FREQ_HZ / (float)TIM2_CLK_DEV / (float)TIM2_CLK_PRESCALER / 1000.0f)

4.25.1 Function Documentation

4.25.1.1 BrightnessDelay()

```
uint16_t BrightnessDelay (
            int8_t brightness)
```

4.25.1.2 DelayHit()

```
uint8_t DelayHit (
            uint32_t delay_ms)
```

4.25.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
            void )
```

4.25.1.4 StartDelayCounter()

```
void StartDelayCounter (
            void )
```

4.25.2 Variable Documentation

4.25.2.1 Tim2ClkKhz

```
const float Tim2ClkKhz = (CLK_FREQ_HZ / (float)TIM2_CLK_DEV / (float)TIM2_CLK_PRESCALER /
1000.0f)
```

4.26 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/fram.c File Reference

This module contains routines to access the SPI FRAM. The FRAM used in the device is 32Kx8 with an operating frequency up to 20 MHz. It supports SPI Mode 0 & 3. The hardware supports the external Write Protect pin. The nature of this FRAM supports unlimited read/writes and data retention beyond the human life span.

```
#include <string.h>
#include "fram.h"
#include "stm32l412xx-bsp.h"
#include "spi.h"
```

Macros

- #define TLEN (16)
- #define TADD (0x200)

Functions

- void framWriteProtect (WRITE_PROTECT_STATE state)
- void framChipSelect (CHIP_SELECT_STATE state)
- void framReadSr (unsigned char ∗srP)
- void framWriteSr (unsigned char sr)
- void framWriteDisable (void)
- void framWriteEnable (void)
- void framReadMemory (unsigned short addr, unsigned char ∗rdBufP, unsigned short len)
- void framWriteMemory (unsigned short addr, const unsigned char ∗const wrBufP, unsigned short len)
- uint8_t framTest (void)

4.26.1 Detailed Description

This module contains routines to access the SPI FRAM. The FRAM used in the device is 32Kx8 with an operating frequency up to 20 MHz. It supports SPI Mode 0 & 3. The hardware supports the external Write Protect pin. The nature of this FRAM supports unlimited read/writes and data retention beyond the human life span.

Attention

Revision History: Date Name Ver Remarks
04/09/2023 Mark Lane 0 Original Version 06/10/2024 Austin Green 1 Add Logging 08/12/2024 Austin Green 1.2 Remove Mirror References
Notes:

4.26.2 Macro Definition Documentation

4.26.2.1 TADD

```
#define TADD (0x200)
```

4.26.2.2 TLEN

```
#define TLEN (16)
```

4.26.3 Function Documentation

4.26.3.1 framChipSelect()

```
void framChipSelect (
            CHIP_SELECT_STATE state)
```

This routine sets the chip select pin to the correct "state".

Parameters

in	*state*	assert = 0, release = 1
out	*none*	

4.26.3.2 framReadMemory()

```
void framReadMemory (
            unsigned short addr,
            unsigned char * rdBufP,
            unsigned short len)
```

This routine reads FRAM memory starting at "addr" for "len" byte(s). The FRAM read data is stored at the pointer referenced by "rdBufP".

Parameters

in	*addr*	FRAM memory address to read data
in	*rdBufP*	destination pointer to store read data
in	*len*	number of byte(s) to read
out	*none*	

4.26.3.3 framReadSr()

```
void framReadSr (
            unsigned char * srP)
```

This routine reads the FRAM status register

Parameters

in	*destination*	pointer for FRAM status register
out	*none*	

4.26.3.4 framTest()

```
uint8_t framTest (
            void )
```

This routine is a test function for FRAM access. It writes "TLEN" bytes of an incrementing pattern into FRAM at address "TADD". It reads the same length into a buffer and verifies the pattern.

Parameters

out	*pass*	= 1, fail = 0

4.26.3.5 framWriteDisable()

```
void framWriteDisable (
             void )
```

This routine resets the write enable latch

Parameters

out	*none*	

4.26.3.6 framWriteEnable()

```
void framWriteEnable (
             void )
```

This routine sets the write enable latch

Parameters

out	*none*	

4.26.3.7 framWriteMemory()

```
void framWriteMemory (
             unsigned short addr,
             const unsigned char *const wrBufP,
             unsigned short len)
```

This routine writes FRAM memory starting at "addr" for "len" byte(s). The data referenced by the pointer "wrBufP" is written into FRAM memory.

Parameters

in	*addr*	FRAM memory address to write data
in	*wrBufP*	pointer to data to write
in	*len*	number of byte(s) to write
out	*none*	

4.26.3.8 framWriteProtect()

```
void framWriteProtect (
             WRITE_PROTECT_STATE state)
```

This routine sets the hardware write protect pin to the correct "state".

Parameters

in	*state*	disable = 0, enable = 1
out	*none*	

4.26.3.9 framWriteSr()

```
void framWriteSr (
            unsigned char sr)
```

This routine writes the FRAM status register

Parameters

in	*data*	value written to FRAM status register
out	*none*	

4.27 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/logger.c File Reference

```
#include <string.h>
#include <stdio.h>
#include "logger.h"
#include "fram.h"
#include "stm32l412xx-bsp.h"
```

Functions

- void LogString (const char ∗const string, uint8_t write_beginning)
- void LogNumber (const int32_t number, uint8_t write_beginning)
- void ReadLog (const uint32_t address, char ∗string, const uint32_t bytes)

4.27.1 Function Documentation

4.27.1.1 LogNumber()

```
void LogNumber (
            const int32_t number,
            uint8_t write_beginning)
```

4.27.1.2 LogString()

```
void LogString (
            const char *const string,
            uint8_t write_beginning)
```

4.27.1.3 ReadLog()

```
void ReadLog (
            const uint32_t address,
            char * string,
            const uint32_t bytes)
```

4.28 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/main.c File Reference

```
#include "main.h"
#include "stm32l412xx-bsp.h"
#include "pwm_handler.h"
#include "delay_handler.h"
#include "button_handler.h"
#include "current_handler.h"
#include "voltage_handler.h"
#include "my_printf.h"
```

Macros

- #define LOWER_SWEEP_TIME_MS (3375)
- #define UPPER_SWEEP_TIME_MS (4000)
- #define TOTAL_SWEEP_TIME_MS (7375)
- #define LOWER_STEP_TIME_MS (LOWER_SWEEP_TIME_MS / (BRIGHTNESS_STEPS / 2.0f))
- #define UPPER_STEP_TIME_MS (UPPER_SWEEP_TIME_MS / (BRIGHTNESS_STEPS / 2.0f))
- #define AVG_STEP_TIME_MS ((UPPER_STEP_TIME_MS + LOWER_STEP_TIME_MS) / 2.0f)
- #define AVG_STEP_DIFF_MS (AVG_STEP_TIME_MS - LOWER_STEP_TIME_MS)

Functions

- void SystemClock_Config (void)

  System Clock Configuration.
- int main (void)

  The application entry point. Initialize variables and go into bare metal loop. Polls buttons and sensors.

Variables

- const float LowStepTimeMs = (LOWER_STEP_TIME_MS - AVG_STEP_DIFF_MS)
- const float HighStepTimeMs = (UPPER_STEP_TIME_MS + AVG_STEP_DIFF_MS)

4.28.1 Macro Definition Documentation

4.28.1.1 AVG_STEP_DIFF_MS

```
#define AVG_STEP_DIFF_MS (AVG_STEP_TIME_MS - LOWER_STEP_TIME_MS)
```

4.28.1.2 AVG_STEP_TIME_MS

```
#define AVG_STEP_TIME_MS ((UPPER_STEP_TIME_MS + LOWER_STEP_TIME_MS) / 2.0f)
```

4.28.1.3 LOWER_STEP_TIME_MS

```
#define LOWER_STEP_TIME_MS (LOWER_SWEEP_TIME_MS / (BRIGHTNESS_STEPS / 2.0f))
```

4.28.1.4 LOWER_SWEEP_TIME_MS

```
#define LOWER_SWEEP_TIME_MS (3375)
```

4.28.1.5 TOTAL_SWEEP_TIME_MS

```
#define TOTAL_SWEEP_TIME_MS (7375)
```

4.28.1.6 UPPER_STEP_TIME_MS

```
#define UPPER_STEP_TIME_MS (UPPER_SWEEP_TIME_MS / (BRIGHTNESS_STEPS / 2.0f))
```

4.28.1.7 UPPER_SWEEP_TIME_MS

```
#define UPPER_SWEEP_TIME_MS (4000)
```

4.28.2 Function Documentation

4.28.2.1 main()

```
int main (
            void )
```

The application entry point. Initialize variables and go into bare metal loop. Polls buttons and sensors.

Return values

int	

4.28.2.2 SystemClock_Config()

```
void SystemClock_Config (
            void )
```

System Clock Configuration.

Return values

None	

4.28.3 Variable Documentation

4.28.3.1 HighStepTimeMs

```
const float HighStepTimeMs = (UPPER_STEP_TIME_MS + AVG_STEP_DIFF_MS)
```

4.28.3.2 LowStepTimeMs

```
const float LowStepTimeMs = (LOWER_STEP_TIME_MS - AVG_STEP_DIFF_MS)
```

4.29 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/my_printf.c File Reference

4.30 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/pwm_handler.c File Reference

```
#include "pwm_handler.h"
#include "stm32l412xx-bsp.h"
#include "temperature_handler.h"
#include "my_printf.h"
```

Macros

- #define PW_PERIOD (255)

Functions

- void InitPwm (void)
- void DecreaseBrightness (uint8_t button_held)
- void IncreaseBrightness (uint8_t button_held)
- void SetPwm (void)
- void TurnOffPwm (void)
- int8_t GetBrightness (void)
- void SetBrightness (int8_t brightness)
- uint8_t GetPwm (void)

Variables

- const uint8_t MaxBrightness = (BRIGHTNESS_STEPS - 1)
- const uint8_t MinBrightness = (0)
- const uint8_t HalfBrightness = ((uint8_t)((BRIGHTNESS_STEPS - 1) / 2.0f))
- const float MinPw = (0)
- const float MaxPw = (PW_PERIOD)
- const float WarmPwmRatio = (0.90f)
- const float HotPwmRatio = (0.50f)

4.30.1 Macro Definition Documentation

4.30.1.1 PW_PERIOD

```
#define PW_PERIOD (255)
```

4.30.2 Function Documentation

4.30.2.1 DecreaseBrightness()

```
void DecreaseBrightness (
             uint8_t button_held)
```

4.30.2.2 GetBrightness()

```
int8_t GetBrightness (
             void )
```

4.30.2.3 GetPwm()

```
uint8_t GetPwm (
             void )
```

4.30.2.4 IncreaseBrightness()

```
void IncreaseBrightness (
             uint8_t button_held)
```

4.30.2.5 InitPwm()

```
void InitPwm (
             void )
```

4.30.2.6 SetBrightness()

```
void SetBrightness (
             int8_t brightness)
```

4.30.2.7 SetPwm()

```
void SetPwm (
             void )
```

4.30.2.8 TurnOffPwm()

```
void TurnOffPwm (
             void )
```

4.30.3 Variable Documentation

4.30.3.1 HalfBrightness

```
const uint8_t HalfBrightness = ((uint8_t)((BRIGHTNESS_STEPS - 1) / 2.0f))
```

4.30.3.2 HotPwmRatio

```
const float HotPwmRatio = (0.50f)
```

4.30.3.3 MaxBrightness

```
const uint8_t MaxBrightness = (BRIGHTNESS_STEPS - 1)
```

4.30.3.4 MaxPw

```
const float MaxPw = (PW_PERIOD)
```

4.30.3.5 MinBrightness

```
const uint8_t MinBrightness = (0)
```

4.30.3.6 MinPw

```
const float MinPw = (0)
```

4.30.3.7 WarmPwmRatio

```
const float WarmPwmRatio = (0.90f)
```

4.31 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/stm32l412xx-bsp.c File Reference

```
#include "stm32l412xx-bsp.h"
```

Functions

- GPIO_PinState ReadDimPin (void)
- GPIO_PinState ReadBrightPin (void)
- void EnablePWM1 (void)
- void DisablePWM1 (void)
- void StartPWM11 (void)
- void StopPWM11 (void)
- void SetPW11 (uint32_t pulse_width)
- void StartTIM2 (void)
- void RestartTIM2 (void)
- uint32_t GetTIM2Cnt (void)
- int16_t GetThermistorValue (void)
- int16_t GetCurrentValue (void)
- int16_t GetVoltageValue (void)
- void enableWriteProtect (void)
- void disableWriteProtect (void)
- void enableChipSelect (void)
- void disableChipSelect (void)
- void transferData (const unsigned char ∗const txData, const uint32_t bytes)
- void receiveData (unsigned char ∗rxData, const uint32_t bytes)
- void sendUARTChar (char c)

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.31.1 Function Documentation

4.31.1.1 disableChipSelect()

```
void disableChipSelect (
             void )
```

4.31.1.2 DisablePWM1()

```
void DisablePWM1 (
             void )
```

4.31.1.3 disableWriteProtect()

```
void disableWriteProtect (
             void )
```

4.31.1.4 enableChipSelect()

```
void enableChipSelect (
             void )
```

4.31.1.5 EnablePWM1()

```
void EnablePWM1 (
             void )
```

4.31.1.6 enableWriteProtect()

```
void enableWriteProtect (
             void )
```

4.31.1.7 Error_Handler()

```
void Error_Handler (
             void )
```

This function is executed in case of error occurrence.

Return values

None	

4.31.1.8 GetCurrentValue()

```
int16_t GetCurrentValue (
             void )
```

4.31.1.9 GetThermistorValue()

```
int16_t GetThermistorValue (
             void )
```

4.31.1.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
             void )
```

4.31.1.11 GetVoltageValue()

```
int16_t GetVoltageValue (
             void )
```

4.31.1.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
             void )
```

4.31.1.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (
             void )
```

4.31.1.14 receiveData()

```
void receiveData (
             unsigned char * rxData,
             const uint32_t bytes)
```

4.31.1.15 RestartTIM2()

```
void RestartTIM2 (
             void )
```

4.31.1.16 sendUARTChar()

```
void sendUARTChar (
             char c)
```

4.31.1.17 SetPW11()

```
void SetPW11 (
             uint32_t pulse_width)
```

4.31.1.18 StartPWM11()

```
void StartPWM11 (
             void )
```

4.31.1.19 StartTIM2()

```
void StartTIM2 (
             void )
```

4.31.1.20 StopPWM11()

```
void StopPWM11 (
             void )
```

4.31.1.21 transferData()

```
void transferData (
             const unsigned char *const txData,
             const uint32_t bytes)
```

4.32 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/temperature_handler.c File Reference

```
#include <string.h>
#include "temperature_handler.h"
#include "stm32l412xx-bsp.h"
#include "logger.h"
```

Functions

- int32_t GetTemperature (void)
- TemperatureRange_e GetTemperatureRange (void)

Variables

- const int32_t ThermistorTodCelcius = (1)
- const int32_t dCelciusToThermistor = (1)
- const int32_t HeatingThreshold1_dC = (1000)
- const int32_t HeatingThreshold2_dC = (1200)
- const int32_t CoolingThreshold1_dC = (800)
- const int32_t CoolingThreshold2_dC = (1000)

4.32.1 Function Documentation

4.32.1.1 GetTemperature()

```
int32_t GetTemperature (
            void )
```

4.32.1.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
            void )
```

4.32.2 Variable Documentation

4.32.2.1 CoolingThreshold1_dC

```
const int32_t CoolingThreshold1_dC = (800)
```

4.32.2.2 CoolingThreshold2_dC

```
const int32_t CoolingThreshold2_dC = (1000)
```

4.32.2.3 dCelciusToThermistor

```
const int32_t dCelciusToThermistor = (1)
```

4.32.2.4 HeatingThreshold1_dC

```
const int32_t HeatingThreshold1_dC = (1000)
```

4.32.2.5 HeatingThreshold2_dC

```
const int32_t HeatingThreshold2_dC = (1200)
```

4.32.2.6 ThermistorTodCelcius

```
const int32_t ThermistorTodCelcius = (1)
```

4.33 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Src/voltage_handler.c File Reference

```
#include <stdio.h>
#include "voltage_handler.h"
#include "stm32l412xx-bsp.h"
#include "logger.h"
```

Functions

- uint16_t GetVoltage (void)
- VoltageRange_e GetVoltageRange (void)

Variables

- const uint16_t RawTodColts = (1)
- const uint16_t dColtsToRaw = (1)
- const uint16_t VoltageErrorLowThreshold_dV = 240u
- const uint16_t VoltageLowThreshold_dV = 260u
- const uint16_t VoltageHighThreshold_dV = 300u
- const uint16_t VoltageErrorHighThreshold_dV = 320u

4.33.1 Function Documentation

4.33.1.1 GetVoltage()

```
uint16_t GetVoltage (
            void )
```

4.33.1.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
            void )
```

4.33.2 Variable Documentation

4.33.2.1 dColtsToRaw

```
const uint16_t dColtsToRaw = (1)
```

4.33.2.2 RawTodColts

```
const uint16_t RawTodColts = (1)
```

4.33.2.3 VoltageErrorHighThreshold_dV

```
const uint16_t VoltageErrorHighThreshold_dV = 320u
```

4.33.2.4 VoltageErrorLowThreshold_dV

```
const uint16_t VoltageErrorLowThreshold_dV = 240u
```

4.33.2.5 VoltageHighThreshold_dV

```
const uint16_t VoltageHighThreshold_dV = 300u
```

4.33.2.6 VoltageLowThreshold_dV

```
const uint16_t VoltageLowThreshold_dV = 260u
```

Index