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 Detailed Description

Testing PWM Struct

3.1.2 Field Documentation

3.1.2.1 is_running

`uint8_t is_running`

pwm is running

3.1.2.2 pulse_width

`uint32_t pulse_width`

pulse width value

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 Detailed Description

Testing Timer Struct

3.2.2 Field Documentation

3.2.2.1 is_running

`uint8_t is_running`

timer is running

3.2.2.2 time

`uint32_t time`

timer value

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)
  Return state of dim button.
- GPIO_PinState IsBrightPressed (void)
  Return state of brighten button.

4.1.1 Function Documentation

4.1.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
            void )
```

Return state of brighten button.

Parameters

out	*Bright*	Pin State, pressed or not

4.1.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
            void )
```

Return state of dim button.

Parameters

out	*Dim*	Pin State, pressed or not

4.2 button_handler.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002  *
00003  * @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *****************************************************************************
00010  *****************************************************************************
00011  *
00012  * @file button_handler.h
00013  *
00014  * @brief Returns the button state of the three board buttons
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00019  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00020  *
00021  * Notes: Depends on the board support package bsp for GPIO_PinState
00022  *
00023  *****************************************************************************/
00024
00025 #ifndef INC_button_handlerh
00026     #define INC_button_handlerh
00027
00028     #include <stdint.h>
00029
00030     #include "stm32l412xx-bsp.h"
00031
00036     GPIO_PinState IsDimPressed    ( void );
00037
00042     GPIO_PinState IsBrightPressed ( void );
00043
00044 #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)

  Get current from ammeter.
- CurrentRange_e GetCurrentRange (void)

  Get range that the current falls into Possible ranges are Normal - Normal Operating Current High - Current high, but ok Error - Current too high.

4.3.1 Enumeration Type Documentation

4.3.1.1 CurrentRange_e

```
enum CurrentRange_e
```

Current Range Enum

Enumerator

CurrentNormal	Normal Operating Current
CurrentHigh	Current high, but ok
CurrentError	Current too high

4.3.2 Function Documentation

4.3.2.1 GetCurrent()

```
uint16_t GetCurrent (
            void )
```

Get current from ammeter.

Parameters

out	*current*	level in dA

4.3.2.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
            void )
```

Get range that the current falls into Possible ranges are Normal - Normal Operating Current High - Current high, but ok Error - Current too high.

Parameters

out	*Current*	current range

4.4 current_handler.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  * @file current_handler.h
00013  *
00014  * @brief Handles getting current and reporting values.
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 08/05/2024 - Austin Green -  1.0 -  Initial Document
00019  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00020  *
00021  * Notes:
00022  *
00023  *******************************************************************************/
00024
00025 #ifndef INC_current_handlerh
00026     #define INC_current_handlerh
00027
00028     #include <stdint.h>
00029
00031     typedef enum
00032     {
00033         CurrentNormal   = 0,
00034         CurrentHigh     = 1,
00035         CurrentError    = 2
00036     } CurrentRange_e;
00037
00043     uint16_t GetCurrent( void );
00044
00053     CurrentRange_e GetCurrentRange( void );
00054
00055 #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)

  Starts the delay counter, only needs to be called once on init.
- void RestartDelayCounter (void)

  Restart the delay counter.
- uint8_t DelayHit (uint32_t delay_ms)

  Checks if the delay (ms) was hit based on timer value.
- uint16_t BrightnessDelay (int8_t brightness)

  Returns a delay value for a brightness level.

4.5.1 Function Documentation

4.5.1.1 BrightnessDelay()

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

Returns a delay value for a brightness level.

Parameters

in	*brightness*	Current brightness level
out	*Returns*	delay to satisfy specs at current brightness level

4.5.1.2 DelayHit()

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

Checks if the delay (ms) was hit based on timer value.

Parameters

in	*delay_ms*	Time in ms to check if timer has hit
out	*Returns*	1 if delay has been hit

4.5.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
            void )
```

Restart the delay counter.

4.5.1.4 StartDelayCounter()

```
void StartDelayCounter (
            void )
```

Starts the delay counter, only needs to be called once on init.

4.6 delay_handler.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  * @file delay_handler.h
00013  *
00014  * @brief Handles system counters and delays
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00019  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00020  *
00021  * Notes:
00022  *
00023  *******************************************************************************/
00024
00025 /* Define to prevent recursive inclusion -------------------------------------*/
00026 #ifndef INC_delay_handlerh
00027     #define INC_delay_handlerh
00028
00029     #include <stdint.h>
00030
00034   void    StartDelayCounter(void);     // start the counter
00035
00039   void    RestartDelayCounter(void);
00045   uint8_t DelayHit(uint32_t delay_ms);
00046
00052   uint16_t BrightnessDelay(int8_t brightness);
00053
00054 #endif /* INC_delay_handlerh */
```

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

```
#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)

This routine resets the write enable latch.

- void framWriteEnable (void)

  This routine resets the write enable latch.
- 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)

  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.

4.7.1 Enumeration Type Documentation

4.7.1.1 CHIP_SELECT_STATE

`enum CHIP_SELECT_STATE`

chip select state

Enumerator

CSS_ASSERT	chip select disabled
CSS_RELEASE	chip select enabled

4.7.1.2 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

4.7.1.3 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

4.7.1.4 WRITE_PROTECT_STATE

```
enum WRITE_PROTECT_STATE
```

write protect state

Enumerator

WPS_PROTECTED	write protected
WPS_WRITEABLE	write enabled

4.7.2 Function Documentation

4.7.2.1 framChipSelect()

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

4.7.2.2 framReadMemory()

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

4.7.2.3 framReadSr()

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

4.7.2.4 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	*1*	= pass, 0 = fail

4.7.2.5 framWriteDisable()

```
void framWriteDisable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.7.2.6 framWriteEnable()

```
void framWriteEnable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.7.2.7 framWriteMemory()

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

4.7.2.8 framWriteProtect()

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

4.7.2.9 framWriteSr()

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

4.8 fram.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002  *
00003  *  @attention
00004  *  Copyright (c) 2022, 2023 Luminator, An LTG Company
00005  *  All rights reserved.
00006  *  Any use without the prior written consent of Luminator,
00007  *  An LTG Company is strictly prohibited.
00008  *
00009  *****************************************************************************
00010  *****************************************************************************
00011  *
00012  *  @file fram.h
00013  *
00014  *  @brief This module contains definitions and structures to support
00015  *          fram.c SPI FRAM operations.
00016  *
00017  *  Revision History:
00018  *    Date         Name            Ver     Remarks
00019  *    -----------  ----------      ---     --------------------------------
00020  *    04/09/2023   Mark Lane       0       Original Version
00021  *    09/10/2024 - Austin Green -  2.0 -   Doxyfile documentation
00022  *
00023  *  Notes:
00024  *
00025  *****************************************************************************/
00026 #ifndef _FRAM_H_
00027 #define _FRAM_H_
00028
00029 #include "stm32l412xx-bsp.h"
00030
00031 /* ------------------------ Local Definition(s) ---------------------------- */
00035 typedef enum {
00036
00037   OC_WREN  = 6 ,
00038   OC_WRDI  = 4 ,
00039   OC_RDSR  = 5 ,
00040   OC_WRSR  = 1 ,
00041   OC_READ  = 3 ,
00042   OC_WRITE = 2 ,
00044 } OPCODE_COMMANDS ;
00045
00046
00050 typedef enum {
00051
00052   SR_WEL  = 0x2  ,
00053   SR_BP0  = 0x4  ,
00054   SR_BP1  = 0x8  ,
00055   SR_WPEN = 0x80 ,
00057 } STATUS_REGISTER ;
00058
00059
00063 typedef enum {
00064
00065   WPS_PROTECTED = 0 ,
00066   WPS_WRITEABLE = 1 ,
00068 } WRITE_PROTECT_STATE ;
00069
00070
00074 typedef enum {
00075
00076   CSS_ASSERT  = 0 ,
00077   CSS_RELEASE = 1 ,
00079 } CHIP_SELECT_STATE ;
00080
00081 /* Prototype Definition */
00091 void framWriteProtect( WRITE_PROTECT_STATE state  ) ;
00092
00101 void framChipSelect( CHIP_SELECT_STATE state  ) ;
00102
00111 void framReadSr( unsigned char *srP ) ;
00112
00121 void framWriteSr( unsigned char sr ) ;
00122
00129 void framWriteDisable( void ) ;
00130
00137 void framWriteEnable( void )  ;
00138
00153 void framReadMemory ( unsigned short addr, unsigned char *rdBufP, unsigned short len ) ;
00154
00169 void framWriteMemory( unsigned short addr, const unsigned char* const wrBufP, unsigned short len ) ;
00170
00181 uint8_t framTest( void ) ;
```

```
00182
00183
00184 #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)

  Log a string to tail_pointer, use write_beginning flag to write the beginning.
- void LogNumber (const int32_t number, uint8_t write_beginning)

  Logs a number by converting the number to a string and using the LogString function.
- void ReadLog (const uint32_t address, char *string, const uint32_t bytes)

  Reads the log at a given address and size.

4.9.1 Function Documentation

4.9.1.1 LogNumber()

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

Logs a number by converting the number to a string and using the LogString function.

Parameters

in	*number*	Number to log
in	*write_beginning*	Write log at the beginning of log area

4.9.1.2 LogString()

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

Log a string to tail_pointer, use write_beginning flag to write the beginning.

Parameters

in	*string*	Pointer to string to log
in	*write_beginning*	Write log at the beginning of log area

4.9.1.3 ReadLog()

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

Reads the log at a given address and size.

Parameters

in	*address*	Address to read from
in	*string*	Pointer to return data string
in	*bytes*	Number of bytes to read

4.10 logger.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  * @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  * @file logger.h
00013  *
00014  * @brief Handles logging and reading of data to memory
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00019  * 08/05/2024 - Austin Green -  1.1 -  Added Log Number
00020  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00021  *
00022  * Notes:
00023  *
00024  *******************************************************************************/
00025
00026 #ifndef INC_loggerh
00027   #define INC_loggerh
00028
00029   #include <stdint.h>
00030
00031
00037   void LogString( const char* const string, uint8_t write_beginning );
00038
00044   void LogNumber( const int32_t number, uint8_t write_beginning );
00045
00052   void ReadLog(  const uint32_t address, char* string, const uint32_t bytes );
00053
00054 #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.

```
00001 /* USER CODE BEGIN Header */
00002 /****
00003     ******************************************************************************
00004     * @file           : main.h
00005     * @brief          : Header for main.c file.
00006     *                   This file contains the common defines of the application.
00007     ******************************************************************************
00008     * @attention
00009     *
00010     * Copyright (c) 2024 STMicroelectronics.
00011     * All rights reserved.
00012     *
00013     * This software is licensed under terms that can be found in the LICENSE file
00014     * in the root directory of this software component.
00015     * If no LICENSE file comes with this software, it is provided AS-IS.
00016     *
00017     ******************************************************************************
00018     */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -------------------------------------*/
00022 #ifndef INC_mainh
00023     #define INC_mainh
00024
00025     #ifdef __cplusplus
00026         extern "C" {
00027     #endif
00028
00029     /* Includes ------------------------------------------------------------------*/
00030
00031     /* Private includes ----------------------------------------------------------*/
00032     /* USER CODE BEGIN Includes */
00033     #include "stm321412xx-bsp.h"
00034
00035     /* USER CODE END Includes */
00036
00037     /* Exported types ------------------------------------------------------------*/
00038     /* USER CODE BEGIN ET */
00039
00040     /* USER CODE END ET */
00041
00042     /* Exported constants --------------------------------------------------------*/
00043     /* USER CODE BEGIN EC */
00044
00045     /* USER CODE END EC */
00046
00047     /* Exported macro ------------------------------------------------------------*/
00048     /* USER CODE BEGIN EM */
00049
00050     /* USER CODE END EM */
00051
00052     /* Exported functions prototypes ---------------------------------------------*/
00053     void Error_Handler(void);
00054
00055     /* USER CODE BEGIN EFP */
00056
00057     /* USER CODE END EFP */
```

```
00058
00059     /* Private defines -----------------------------------------------------------*/
00060
00061     /* USER CODE BEGIN Private defines */
00062
00063     /* USER CODE END Private defines */
00064
00065     #ifdef __cplusplus
00066         }
00067     #endif
00068
00069 #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.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  * @file my_printf.h
00013  *
00014  * @brief Prints characters to a terminal for debugging purposes
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00019  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00020  *
00021  * Notes:
00022  *
00023  *******************************************************************************/
00024
00025 // Software tracing with printf()
00026 #ifndef INC_my_printfh
00027     #define INC_my_printfh
00028
00029     #ifdef ENABLE_UART_DEBUGGING /* tracing enabled */
00030         #include <stdio.h>
00031     #endif /* ENABLE_UART_DEBUGGING */
00032
00033 #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)

  Init PwmArray var Set brightness to half value, enable pwm, and turn off.
- void DecreaseBrightness (uint8_t button_held)

  Decrease brightness by 1 (for button press) or 3 (for button hold) This functions can be made to increase brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.
- void IncreaseBrightness (uint8_t button_held)

  Increase brightness by 1 (for button press) or 3 (for button hold) This functions can be made to decrease brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.
- void SetPwm (void)

  set PWM based on pwm value
- void TurnOffPwm (void)

  turn off PWM
- int8_t GetBrightness (void)

  Return ledBrightness variable.
- void SetBrightness (int8_t brightness)

  Set ledBrightness variable, guards to ensure we don't go over max or min.
- uint8_t GetPwm (void)

  Get the PWM value based on the brightness and the temperature range.

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)
```

Decrease brightness by 1 (for button press) or 3 (for button hold) This functions can be made to increase brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.

Parameters

in	*button_held*	If the button is being held (decrements by 3 if so)

4.15.2.2 GetBrightness()

```
int8_t GetBrightness (
            void )
```

Return ledBrightness variable.

Parameters

out	*Current*	LED brightness level

4.15.2.3 GetPwm()

```
uint8_t GetPwm (
            void )
```

Get the PWM value based on the brightness and the temperature range.

Parameters

out	*Current*	PWM value

4.15.2.4 IncreaseBrightness()

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

Increase brightness by 1 (for button press) or 3 (for button hold) This functions can be made to decrease brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.

Parameters

in	*button_held*	If the button is being held (increments by 3 if so)

4.15.2.5 InitPwm()

```
void InitPwm (
            void )
```

Init PwmArray var Set brightness to half value, enable pwm, and turn off.

4.15.2.6 SetBrightness()

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

Set ledBrightness variable, guards to ensure we don't go over max or min.

Parameters

in	*brightness*	Brightess to set

4.15.2.7 SetPwm()

```
void SetPwm (
             void )
```

set PWM based on pwm value

4.15.2.8 TurnOffPwm()

```
void TurnOffPwm (
             void )
```

turn off PWM

4.16 pwm_handler.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *****************************************************************************
00010  *****************************************************************************
00011  *
00012  * @file pwm_handler.h
00013  *
00014  * @brief Handles the PWM output of the lights. Output is determined by a
00015  *          Brightness variable that is controlled by this file.
00016  *
00017  * Revision History:
00018  * Date       - Name         -  Ver -  Remarks
00019  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00020  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00021  *
00022  * Notes:
00023  *
00024  *****************************************************************************/
00025
00026 #ifndef INC_pwm_handlerh
00027     #define INC_pwm_handlerh
00028
00029     #include <stdint.h>
00030
00031     /* Brightness Steps */
00032     #define BRIGHTNESS_STEPS        (50)
00033     #define HOLD_BRIGHTNESS_JUMP    (3)
00034
00039   void    InitPwm(void);                              // Init Pwm var
00040
00048   void    DecreaseBrightness( uint8_t button_held );  // decrease brightness
00049
00057   void    IncreaseBrightness( uint8_t button_held );  // increase brightness
00058
00062   void    SetPwm( void );                             // turn on and set PWM
00063
00067   void    TurnOffPwm( void );                         // turn of PWM
00068
00073   int8_t  GetBrightness( void );                      // get value of Brightness
00074
00079   void    SetBrightness( int8_t brightness );         // set value of Brightness
00080
00085   uint8_t GetPwm( void );                             // get value of current PWM
00086
00087 #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)

  Reads dim pin value.
- GPIO_PinState ReadBrightPin (void)

  Reads bright pin value.
- void EnablePWM1 (void)

  Enables Timer 1.
- void DisablePWM1 (void)

  Disables Timer 1.
- void StartPWM11 (void)

  Starts PWM Timer 1 Channel 1 output.
- void StopPWM11 (void)

  Stops PWM Timer 1 Channel 1 output.
- void SetPW11 (uint32_t pulse_width)

  Sets PWM Timer 1 Channel 1 value.
- void StartTIM2 (void)

  Starts Timer 2 counter.
- void RestartTIM2 (void)

  Resets Timer 2 counter to zero.
- uint32_t GetTIM2Cnt (void)

  Returns value in the Timer 2 counter.
- int16_t GetThermistorValue (void)

  Returns raw ADC value from thermistor.
- int16_t GetCurrentValue (void)

  Returns raw ADC value from ammeter.
- int16_t GetVoltageValue (void)

  Returns raw ADC value from voltmeter.
- void enableWriteProtect (void)

  Enables SPI write protect line (active high)
- void disableWriteProtect (void)

  Disables SPI write protect line (active high)
- void enableChipSelect (void)

  Enables SPI chip select line (active low)
- void disableChipSelect (void)

  Disables SPI chip select line (active low)
- void transferData (const unsigned char ∗const txData, const uint32_t bytes)

  Sends data via SPI lines.
- void receiveData (unsigned char ∗rxData, const uint32_t bytes)

  Gets data from SPI lines.
- void sendUARTChar (char c)

  Sends character via UART line.
- 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 )
```

Disables SPI chip select line (active low)

4.17.4.2 DisablePWM1()

```
void DisablePWM1 (
            void )
```

Disables Timer 1.

4.17.4.3 disableWriteProtect()

```
void disableWriteProtect (
            void )
```

Disables SPI write protect line (active high)

4.17.4.4 enableChipSelect()

```
void enableChipSelect (
            void )
```

Enables SPI chip select line (active low)

4.17.4.5 EnablePWM1()

```
void EnablePWM1 (
            void )
```

Enables Timer 1.

4.17.4.6 enableWriteProtect()

```
void enableWriteProtect (
            void )
```

Enables SPI write protect line (active high)

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 )
```

Returns raw ADC value from ammeter.

Parameters

out	*Ammeter*	raw ADC value

4.17.4.9 GetThermistorValue()

```
int16_t GetThermistorValue (
            void )
```

Returns raw ADC value from thermistor.

Parameters

out	*Thermistor*	raw ADC value

4.17.4.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
            void )
```

Returns value in the Timer 2 counter.

Parameters

out	*Value*	of Timer 2 counter

4.17.4.11 GetVoltageValue()

```
int16_t GetVoltageValue (
            void )
```

Returns raw ADC value from voltmeter.

Parameters

out	*Voltmeter*	raw ADC value

4.17.4.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
            void )
```

Reads bright pin value.

Parameters

out	*Bright*	pin state

4.17.4.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (
            void )
```

Reads dim pin value.

Parameters

out	*Dim*	pin state

4.17.4.14 receiveData()

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

Gets data from SPI lines.

Parameters

in	*rxData*	Pointer to data buffer
in	*bytes*	Number of bytes to receive

4.17.4.15 RestartTIM2()

```
void RestartTIM2 (
            void )
```

Resets Timer 2 counter to zero.

4.17.4.16 sendUARTChar()

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

Sends character via UART line.

Parameters

in	*c*	Character to send via UART

4.17.4.17 SetPW11()

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

Sets PWM Timer 1 Channel 1 value.

Parameters

in	*pulse_width*	Value out of 255 to set pulse width to

4.17.4.18 StartPWM11()

```
void StartPWM11 (
            void )
```

Starts PWM Timer 1 Channel 1 output.

4.17.4.19 StartTIM2()

```
void StartTIM2 (
            void )
```

Starts Timer 2 counter.

4.17.4.20 StopPWM11()

```
void StopPWM11 (
            void )
```

Stops PWM Timer 1 Channel 1 output.

4.17.4.21 transferData()

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

Sends data via SPI lines.

Parameters

in	*txData*	Pointer to data to send
in	*bytes*	Number of bytes to send

4.18 stm32l412xx-bsp.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  * @file stm32l412xx-bsp.h
00013  *
00014  * @brief Board Support Package for STM32L412xx
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00019  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00020  *
00021  * Notes: This uses the Low Level ST API to access the board pins
00022  *
00023  *******************************************************************************/
00024
00025 #ifndef INC_bsph
00026   #define INC_bsph
00027
00028   #include <stdint.h>
00029
00030   /* Private defines -----------------------------------------------------------*/
00031   #ifdef STM32L412xx
00032
00033     #include "stm32l4xx_ll_adc.h"
00034     #include "stm32l4xx_ll_crs.h"
00035     #include "stm32l4xx_ll_rcc.h"
00036     #include "stm32l4xx_ll_bus.h"
00037     #include "stm32l4xx_ll_system.h"
00038     #include "stm32l4xx_ll_exti.h"
00039     #include "stm32l4xx_ll_cortex.h"
00040     #include "stm32l4xx_ll_utils.h"
00041     #include "stm32l4xx_ll_pwr.h"
00042     #include "stm32l4xx_ll_dma.h"
00043     #include "stm32l4xx_ll_spi.h"
00044     #include "stm32l4xx_ll_tim.h"
00045     #include "stm32l4xx_ll_usart.h"
00046     #include "stm32l4xx_ll_gpio.h"
00047
00048     #if defined(USE_FULL_ASSERT)
00049         #include "stm32_assert.h"
00050     #endif /* USE_FULL_ASSERT */
00051
00052     /* Peripherals */
00053     #include "adc.h"
00054     #include "gpio.h"
00055     #include "spi.h"
00056     #include "tim.h"
00057
00058     #ifdef ENABLE_UART_DEBUGGING /* tracing enabled */
00059         /* Peripherals enabled for UART */
00060         #include "usart.h"
00061     #endif /* ENABLE_UART_DEBUGGING */
00062
00063     #define THERMISTOR_ADC_Pin LL_GPIO_PIN_0
00064     #define THERMISTOR_ADC_GPIO_Port GPIOA
00065     #define BRIGHT_Pin LL_GPIO_PIN_1
00066     #define BRIGHT_GPIO_Port GPIOA
00067     #define DIM_Pin LL_GPIO_PIN_3
00068     #define DIM_GPIO_Port GPIOA
00069     #define EEPROM_SCK_Pin LL_GPIO_PIN_5
00070     #define EEPROM_SCK_GPIO_Port GPIOA
00071     #define EEPROM_MISO_Pin LL_GPIO_PIN_6
00072     #define EEPROM_MISO_GPIO_Port GPIOA
00073     #define EEPROM_MOSI_Pin LL_GPIO_PIN_7
00074     #define EEPROM_MOSI_GPIO_Port GPIOA
00075     #define VOLTMETER_ADC_Pin LL_GPIO_PIN_0
00076     #define VOLTMETER_ADC_GPIO_Port GPIOB
00077     #define AMPMETER_ADC_Pin LL_GPIO_PIN_1
00078     #define AMPMETER_ADC_GPIO_Port GPIOB
00079     #define PWM_OUT_Pin LL_GPIO_PIN_8
00080     #define PWM_OUT_GPIO_Port GPIOA
00081     #define SPI_WP_Pin LL_GPIO_PIN_10
00082     #define SPI_WP_GPIO_Port GPIOA
```

```
00083     #define SPI_NSS_Pin LL_GPIO_PIN_11
00084     #define SPI_NSS_GPIO_Port GPIOA
00085
00086   #else /* STM32L412xx */
00087
00088     /* Below is for debugging purposes */
00089     #define THERMISTOR_ADC_Pin          0
00090     #define THERMISTOR_ADC_GPIO_Port    0
00091     #define BRIGHT_Pin                  0
00092     #define BRIGHT_GPIO_Port            0
00093     #define DIM_Pin                     0
00094     #define DIM_GPIO_Port               0
00095     #define EEPROM_SCK_Pin              0
00096     #define EEPROM_SCK_GPIO_Port        0
00097     #define EEPROM_MISO_Pin             0
00098     #define EEPROM_MISO_GPIO_Port       0
00099     #define EEPROM_MOSI_Pin             0
00100     #define EEPROM_MOSI_GPIO_Port       0
00101     #define VOLTMETER_ADC_Pin           0
00102     #define VOLTMETER_ADC_GPIO_Port     0
00103     #define AMPMETER_ADC_Pin            0
00104     #define AMPMETER_ADC_GPIO_Port      0
00105     #define PWM_OUT_Pin                 0
00106     #define PWM_OUT_GPIO_Port           0
00107     #define SPI_WP_Pin                  0
00108     #define SPI_WP_GPIO_Port            0
00109     #define SPI_NSS_Pin                 0
00110     #define SPI_NSS_GPIO_Port           0
00111
00115     typedef struct
00116     {
00117       uint8_t is_running;
00118       uint32_t pulse_width;
00119     } PwmStruct;
00120
00124     typedef struct
00125     {
00126       uint8_t is_running;
00127       uint32_t time;
00128     } TimerStruct;
00129
00130   #endif /* STM32L412xx */
00131
00132   /* Interrupt Handlers */
00133   #ifndef NVIC_PRIORITYGROUP_0
00134     #define NVIC_PRIORITYGROUP_0         ((uint32_t)0x00000007)
00136     #define NVIC_PRIORITYGROUP_1         ((uint32_t)0x00000006)
00138     #define NVIC_PRIORITYGROUP_2         ((uint32_t)0x00000005)
00140     #define NVIC_PRIORITYGROUP_3         ((uint32_t)0x00000004)
00142     #define NVIC_PRIORITYGROUP_4         ((uint32_t)0x00000003)
00144   #endif
00145
00146   /* Button Defines */
00147   typedef uint8_t GPIO_PinState;
00148   enum { BUTTON_PRESSED = 1, BUTTON_UNPRESSED = 0};
00149   enum { PIN_SET = 1, PIN_RESET = 0};
00150
00151   /* Clock frequency Values */
00152   #define CLK_FREQ_HZ  (8000000)
00153   #define TIM2_CLK_DEV  (1)
00154   #define TIM2_CLK_PRESCALER  (8000)
00155
00156   /* Returns button state */
00161   GPIO_PinState ReadDimPin( void );
00162
00167   GPIO_PinState ReadBrightPin( void );
00168   /* PWM Outputs */
00172   void EnablePWM1( void );
00173
00177   void DisablePWM1( void );
00178
00182   void StartPWM11( void );
00183
00187   void StopPWM11( void );
00188
00193   void SetPW11( uint32_t pulse_width );
00194
00195   /* Timers */
00199   void StartTIM2( void );
00200
00204   void RestartTIM2( void );
00205
00210   uint32_t GetTIM2Cnt( void );
00211
00216   int16_t GetThermistorValue( void );
00217
00222   int16_t GetCurrentValue( void );
```

```
00223
00228   int16_t GetVoltageValue( void );
00229
00233   void enableWriteProtect( void );
00234
00238   void disableWriteProtect( void );
00239
00243   void enableChipSelect( void );
00244
00248   void disableChipSelect( void );
00249
00255   void transferData( const unsigned char* const txData, const uint32_t bytes );
00256
00262   void receiveData( unsigned char* rxData, const uint32_t bytes );
00263
00268   void sendUARTChar(char c);
00269
00274   void Error_Handler(void);
00275
00276
00277 #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)

  Get temperature from thermistor.
- TemperatureRange_e GetTemperatureRange (void)

  Get range that the temperature falls into. There is an increased threshold to fall back into the previous state. Possible ranges are Cool - Normal Operating Temperature Warm - Temperature is elevated, decrease brightness Hot - Temperature is hot, lower brightness significantly.

4.19.1 Enumeration Type Documentation

4.19.1.1 TemperatureRange_e

```
enum TemperatureRange_e
```

Temperature Range Enum

Enumerator

TempCool	Normal Operating Temperature
TempWarm	Temperature is elevated, decrease brightness
TempHot	Temperature is hot, lower brightness significantly

4.19.2 Function Documentation

4.19.2.1 GetTemperature()

```
int32_t GetTemperature (
            void )
```

Get temperature from thermistor.

Parameters

out	*temperature*	level in dC

4.19.2.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
            void )
```

Get range that the temperature falls into. There is an increased threshold to fall back into the previous state. Possible ranges are Cool - Normal Operating Temperature Warm - Temperature is elevated, decrease brightness Hot - Temperature is hot, lower brightness significantly.

Parameters

out	*Current*	temperature range

4.20 temperature_handler.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *****************************************************************************
00010  *****************************************************************************
00011  *
00012  * @file temperature_handler.h
00013  *
00014  * @brief Handles getting this temperature and transitioning between
00015  *          temperature states.
00016  *
00017  * Revision History:
00018  * Date       - Name         -  Ver -  Remarks
00019  * 07/31/2024 - Austin Green -  1.0 -  Initial Document
00020  * 08/05/2024 - Austin Green -  1.1 -  Refactor to not use floats
00021  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00022  *
00023  * Notes:
00024  *
00025  *****************************************************************************/
00026
00027 #ifndef INC_temperature_handlerh
00028     #define INC_temperature_handlerh
00029
00030     #include <stdint.h>
00031
00033     typedef enum
```

```
00034     {
00035         TempCool    = 0,
00036         TempWarm    = 1,
00037         TempHot     = 2
00038     } TemperatureRange_e;
00039
00045     int32_t GetTemperature( void );
00046
00056     TemperatureRange_e GetTemperatureRange( void );
00057
00058 #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)

  Get voltage from voltmeter.
- VoltageRange_e GetVoltageRange (void)

  Get range that the voltage falls into Possible ranges are Normal - Normal Operating Voltage Low - Voltage low, but ok High - Voltage high, but ok ErrorLow - Voltage too low ErrorHigh - Voltage too high.

4.21.1 Enumeration Type Documentation

4.21.1.1 VoltageRange_e

```
enum VoltageRange_e
```

Voltage Range Enum

Enumerator

VoltageNormal	Normal Operating Voltage
VoltageLow	Voltage low, but ok
VoltageHigh	Voltage high, but ok
VoltageErrorLow	Voltage too low
VoltageErrorHigh	Voltage too high

4.21.2 Function Documentation

4.21.2.1 GetVoltage()

```
uint16_t GetVoltage (
            void )
```

Get voltage from voltmeter.

Parameters

out	*voltage*	level in dV

4.21.2.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
            void )
```

Get range that the voltage falls into Possible ranges are Normal - Normal Operating Voltage Low - Voltage low, but ok High - Voltage high, but ok ErrorLow - Voltage too low ErrorHigh - Voltage too high.

Parameters

out	*Current*	voltage range

4.22 voltage_handler.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  * Copyright © 2007 Luminator Mark IV
00005  * All rights reserved.
00006  * Any use without the prior written consent of Luminator Mark IV
00007  * is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  * @file voltage_handler.h
00013  *
00014  * @brief Handles getting voltage and reporting values.
00015  *
00016  * Revision History:
00017  * Date       - Name         -  Ver -  Remarks
00018  * 08/05/2024 - Austin Green -  1.0 -  Initial Document
00019  * 09/10/2024 - Austin Green -  2.0 -  Doxyfile documentation
00020  *
00021  * Notes:
00022  *
00023  *******************************************************************************/
00024
00025 #ifndef INC_voltage_handlerh
00026     #define INC_voltage_handlerh
00027
00028     #include <stdint.h>
00029
00031     typedef enum
00032     {
00033         VoltageNormal       = 0,
00034         VoltageLow          = 1,
00035         VoltageHigh         = 2,
00036         VoltageErrorLow     = 3,
00037         VoltageErrorHigh    = 4
00038     } VoltageRange_e;
00039
00045     uint16_t GetVoltage( void );
00046
00057     VoltageRange_e GetVoltageRange( void );
00058
00059 #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)

  Return state of dim button.
- GPIO_PinState IsBrightPressed (void)

  Return state of brighten button.

4.23.1 Function Documentation

4.23.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
            void )
```

Return state of brighten button.

Parameters

out	*Bright*	Pin State, pressed or not

4.23.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
            void )
```

Return state of dim button.

Parameters

out	*Dim*	Pin State, pressed or not

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)

  Get current from ammeter.
- CurrentRange_e GetCurrentRange (void)

  Get range that the current falls into Possible ranges are Normal - Normal Operating Current High - Current high, but ok Error - Current too high.

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 )
```

Get current from ammeter.

Parameters

out	*current*	level in dA

4.24.1.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
            void )
```

Get range that the current falls into Possible ranges are Normal - Normal Operating Current High - Current high, but ok Error - Current too high.

Parameters

out	*Current*	current range

4.24.2 Variable Documentation

4.24.2.1 CurrentErrorThreshold_dA

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

High Current Error Level in dA

4.24.2.2 CurrentHighThreshold_dA

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

High Current Level in dA

4.24.2.3 dAmpsToRaw

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

DeciAmps (A∗0.1) to raw value out of ammeter

4.24.2.4 RawTodAmps

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

Raw value out of ammeter to deciAmps (A∗0.1)

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)
  Starts the delay counter, only needs to be called once on init.
- void RestartDelayCounter (void)
  Restart the delay counter.
- uint8_t DelayHit (uint32_t delay_ms)
  Checks if the delay (ms) was hit based on timer value.
- uint16_t BrightnessDelay (int8_t brightness)
  Returns a delay value for a brightness level.

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)
```

Returns a delay value for a brightness level.

Parameters

in	*brightness*	Current brightness level
out	*Returns*	delay to satisfy specs at current brightness level

4.25.1.2 DelayHit()

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

Checks if the delay (ms) was hit based on timer value.

Parameters

in	*delay_ms*	Time in ms to check if timer has hit
out	*Returns*	1 if delay has been hit

4.25.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
            void )
```

Restart the delay counter.

4.25.1.4 StartDelayCounter()

```
void StartDelayCounter (
            void )
```

Starts the delay counter, only needs to be called once on init.

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)
```

Timer clock used to check the delay values in stm32l412xx-bsp.h

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

```
#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)

  This routine resets the write enable latch.
- void framWriteEnable (void)

  This routine resets the write enable latch.
- 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)

  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.

4.26.1 Macro Definition Documentation

4.26.1.1 TADD

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

4.26.1.2 TLEN

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

4.26.2 Function Documentation

4.26.2.1 framChipSelect()

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

4.26.2.2 framReadMemory()

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

4.26.2.3 framReadSr()

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

4.26.2.4 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	*1*	= pass, 0 = fail

4.26.2.5 framWriteDisable()

```
void framWriteDisable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.26.2.6 framWriteEnable()

```
void framWriteEnable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.26.2.7 framWriteMemory()

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

4.26.2.8 framWriteProtect()

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

4.26.2.9 framWriteSr()

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

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)

  Log a string to tail_pointer, use write_beginning flag to write the beginning.
- void LogNumber (const int32_t number, uint8_t write_beginning)

  Logs a number by converting the number to a string and using the LogString function.
- void ReadLog (const uint32_t address, char *string, const uint32_t bytes)

  Reads the log at a given address and size.

4.27.1 Function Documentation

4.27.1.1 LogNumber()

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

Logs a number by converting the number to a string and using the LogString function.

Parameters

in	*number*	Number to log
in	*write_beginning*	Write log at the beginning of log area

4.27.1.2 LogString()

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

Log a string to tail_pointer, use write_beginning flag to write the beginning.

Parameters

in	*string*	Pointer to string to log
in	*write_beginning*	Write log at the beginning of log area

4.27.1.3 ReadLog()

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

Reads the log at a given address and size.

Parameters

in	*address*	Address to read from
in	*string*	Pointer to return data string
in	*bytes*	Number of bytes to read

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)

  Init PwmArray var Set brightness to half value, enable pwm, and turn off.
- void DecreaseBrightness (uint8_t button_held)

  Decrease brightness by 1 (for button press) or 3 (for button hold) This functions can be made to increase brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.
- void IncreaseBrightness (uint8_t button_held)

  Increase brightness by 1 (for button press) or 3 (for button hold) This functions can be made to decrease brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.
- void SetPwm (void)

  set PWM based on pwm value
- void TurnOffPwm (void)

  turn off PWM
- int8_t GetBrightness (void)

  Return ledBrightness variable.
- void SetBrightness (int8_t brightness)

  Set ledBrightness variable, guards to ensure we don't go over max or min.
- uint8_t GetPwm (void)

  Get the PWM value based on the brightness and the temperature range.

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)
```

Decrease brightness by 1 (for button press) or 3 (for button hold) This functions can be made to increase brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.

Parameters

in	*button_held*	If the button is being held (decrements by 3 if so)

4.30.2.2 GetBrightness()

```
int8_t GetBrightness (
            void )
```

Return ledBrightness variable.

Parameters

out	*Current*	LED brightness level

4.30.2.3 GetPwm()

```
uint8_t GetPwm (
            void )
```

Get the PWM value based on the brightness and the temperature range.

Parameters

out	*Current*	PWM value

4.30.2.4 IncreaseBrightness()

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

Increase brightness by 1 (for button press) or 3 (for button hold) This functions can be made to decrease brightness value with the REVERSE_BRIGHTNESS flag Afterwards, set pwm output.

Parameters

in	*button_held*	If the button is being held (increments by 3 if so)

4.30.2.5 InitPwm()

```
void InitPwm (
            void )
```

Init PwmArray var Set brightness to half value, enable pwm, and turn off.

4.30.2.6 SetBrightness()

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

Set ledBrightness variable, guards to ensure we don't go over max or min.

Parameters

in	*brightness*	Brightess to set

4.30.2.7 SetPwm()

```
void SetPwm (
            void )
```

set PWM based on pwm value

4.30.2.8 TurnOffPwm()

```
void TurnOffPwm (
            void )
```

turn off PWM

4.30.3 Variable Documentation

4.30.3.1 HalfBrightness

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

Half Brightness Step (24)

4.30.3.2 HotPwmRatio

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

Hot thermal state pwm constant

4.30.3.3 MaxBrightness

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

Max Brightness Step (49)

4.30.3.4 MaxPw

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

Max pulse width value (PW_PERIOD(255))

4.30.3.5 MinBrightness

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

Min Brightness Step (0)

4.30.3.6 MinPw

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

Min pulse width value (0)

4.30.3.7 WarmPwmRatio

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

Warm thermal state pwm constant

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)
  Reads dim pin value.
- GPIO_PinState ReadBrightPin (void)
  Reads bright pin value.
- void EnablePWM1 (void)
  Enables Timer 1.
- void DisablePWM1 (void)
  Disables Timer 1.
- void StartPWM11 (void)
  Starts PWM Timer 1 Channel 1 output.
- void StopPWM11 (void)
  Stops PWM Timer 1 Channel 1 output.
- void SetPW11 (uint32_t pulse_width)
  Sets PWM Timer 1 Channel 1 value.
- void StartTIM2 (void)
  Starts Timer 2 counter.
- void RestartTIM2 (void)
  Resets Timer 2 counter to zero.
- uint32_t GetTIM2Cnt (void)
  Returns value in the Timer 2 counter.
- int16_t GetThermistorValue (void)

Returns raw ADC value from thermistor.

- int16_t GetCurrentValue (void)

  Returns raw ADC value from ammeter.
- int16_t GetVoltageValue (void)

  Returns raw ADC value from voltmeter.
- void enableWriteProtect (void)

  Enables SPI write protect line (active high)
- void disableWriteProtect (void)

  Disables SPI write protect line (active high)
- void enableChipSelect (void)

  Enables SPI chip select line (active low)
- void disableChipSelect (void)

  Disables SPI chip select line (active low)
- void transferData (const unsigned char ∗const txData, const uint32_t bytes)

  Sends data via SPI lines.
- void receiveData (unsigned char ∗rxData, const uint32_t bytes)

  Gets data from SPI lines.
- void sendUARTChar (char c)

  Sends character via UART line.
- 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 )
```

Disables SPI chip select line (active low)

4.31.1.2 DisablePWM1()

```
void DisablePWM1 (
            void )
```

Disables Timer 1.

4.31.1.3 disableWriteProtect()

```
void disableWriteProtect (
            void )
```

Disables SPI write protect line (active high)

4.31.1.4 enableChipSelect()

```
void enableChipSelect (
            void )
```

Enables SPI chip select line (active low)

4.31.1.5 EnablePWM1()

```
void EnablePWM1 (
            void )
```

Enables Timer 1.

4.31.1.6 enableWriteProtect()

```
void enableWriteProtect (
            void )
```

Enables SPI write protect line (active high)

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 )
```

Returns raw ADC value from ammeter.

Parameters

out	*Ammeter*	raw ADC value

4.31.1.9 GetThermistorValue()

```
int16_t GetThermistorValue (
            void )
```

Returns raw ADC value from thermistor.

Parameters

out	*Thermistor*	raw ADC value

4.31.1.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
            void )
```

Returns value in the Timer 2 counter.

Parameters

out	*Value*	of Timer 2 counter

4.31.1.11 GetVoltageValue()

```
int16_t GetVoltageValue (
            void )
```

Returns raw ADC value from voltmeter.

Parameters

out	*Voltmeter*	raw ADC value

4.31.1.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
            void )
```

Reads bright pin value.

Parameters

out	*Bright*	pin state

4.31.1.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (
            void )
```

Reads dim pin value.

Parameters

out	*Dim*	pin state

4.31.1.14 receiveData()

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

Gets data from SPI lines.

Parameters

in	*rxData*	Pointer to data buffer
in	*bytes*	Number of bytes to receive

4.31.1.15 RestartTIM2()

```
void RestartTIM2 (
            void )
```

Resets Timer 2 counter to zero.

4.31.1.16 sendUARTChar()

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

Sends character via UART line.

Parameters

in	*c*	Character to send via UART

4.31.1.17 SetPW11()

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

Sets PWM Timer 1 Channel 1 value.

Parameters

in	*pulse_width*	Value out of 255 to set pulse width to

4.31.1.18 StartPWM11()

```
void StartPWM11 (
            void )
```

Starts PWM Timer 1 Channel 1 output.

4.31.1.19 StartTIM2()

```
void StartTIM2 (
            void )
```

Starts Timer 2 counter.

4.31.1.20 StopPWM11()

```
void StopPWM11 (
            void )
```

Stops PWM Timer 1 Channel 1 output.

4.31.1.21 transferData()

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

Sends data via SPI lines.

Parameters

in	*txData*	Pointer to data to send
in	*bytes*	Number of bytes to send

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)

  Get temperature from thermistor.
- TemperatureRange_e GetTemperatureRange (void)

  Get range that the temperature falls into. There is an increased threshold to fall back into the previous state. Possible ranges are Cool - Normal Operating Temperature Warm - Temperature is elevated, decrease brightness Hot - Temperature is hot, lower brightness significantly.

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 )
```

Get temperature from thermistor.

Parameters

out	*temperature*	level in dC

4.32.1.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
            void )
```

Get range that the temperature falls into. There is an increased threshold to fall back into the previous state. Possible ranges are Cool - Normal Operating Temperature Warm - Temperature is elevated, decrease brightness Hot - Temperature is hot, lower brightness significantly.

Parameters

out	*Current*	temperature range

4.32.2 Variable Documentation

4.32.2.1 CoolingThreshold1_dC

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

Cooling Threshold 1 (cooling down from Warm to Cool) in dC

4.32.2.2 CoolingThreshold2_dC

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

Cooling Threshold 2 (cooling down from Hot to Warm) in dC

4.32.2.3 dCelciusToThermistor

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

DeciCelcius (V∗0.1) to raw value out of thermistor

4.32.2.4 HeatingThreshold1_dC

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

Heating Threshold 1 (heating up from Cool to Warm) in dC

4.32.2.5 HeatingThreshold2_dC

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

Heating Threshold 2 (heating up from Warm to Hot) in dC

4.32.2.6 ThermistorTodCelcius

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

Raw value out of thermistor to deciCelcius (C∗0.1)

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)

  Get voltage from voltmeter.
- VoltageRange_e GetVoltageRange (void)

  Get range that the voltage falls into Possible ranges are Normal - Normal Operating Voltage Low - Voltage low, but ok High - Voltage high, but ok ErrorLow - Voltage too low ErrorHigh - Voltage too high.

Variables

- const uint16_t RawTodVolts = (1)
- const uint16_t dVoltsToRaw = (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 )
```

Get voltage from voltmeter.

Parameters

out	*voltage*	level in dV

4.33.1.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
            void )
```

Get range that the voltage falls into Possible ranges are Normal - Normal Operating Voltage Low - Voltage low, but ok High - Voltage high, but ok ErrorLow - Voltage too low ErrorHigh - Voltage too high.

Parameters

out	*Current*	voltage range

4.33.2 Variable Documentation

4.33.2.1 dVoltsToRaw

```
const uint16_t dVoltsToRaw = (1)
```

DeciCelcius (C∗0.1) to raw value out of voltmeter

4.33.2.2 RawTodVolts

```
const uint16_t RawTodVolts = (1)
```

Raw value out of voltmeter to deciVolts (V∗0.1)

4.33.2.3 VoltageErrorHighThreshold_dV

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

High Voltage Error Level in dV

4.33.2.4 VoltageErrorLowThreshold_dV

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

Low Voltage Error Level in dV

4.33.2.5 VoltageHighThreshold_dV

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

High Voltage Level in dV

4.33.2.6 VoltageLowThreshold_dV

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

Low Voltage Level in dV

Index