

CH-53K

1.0.0

Generated by Doxygen 1.12.0

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 PwmStruct Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 is_running	5
3.1.2.2 pulse_width	5
3.2 TimerStruct Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Field Documentation	6
3.2.2.1 is_running	6
3.2.2.2 time	6
4 File Documentation	7
4.1 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/button_handler.h File Reference	7
4.1.1 Function Documentation	7
4.1.1.1 IsBrightPressed()	7
4.1.1.2 IsDimPressed()	7
4.2 button_handler.h	8
4.3 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/current_handler.h File Reference	8
4.3.1 Enumeration Type Documentation	9
4.3.1.1 CurrentRange_e	9
4.3.2 Function Documentation	9
4.3.2.1 GetCurrent()	9
4.3.2.2 GetCurrentRange()	9
4.4 current_handler.h	10
4.5 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/delay_handler.h File Reference	10
4.5.1 Function Documentation	11
4.5.1.1 BrightnessDelay()	11
4.5.1.2 DelayHit()	11
4.5.1.3 RestartDelayCounter()	11
4.5.1.4 StartDelayCounter()	11
4.6 delay_handler.h	12
4.7 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/fram.h File Reference	12
4.7.1 Enumeration Type Documentation	13
4.7.1.1 CHIP_SELECT_STATE	13

4.7.1.2 OPCODE_COMMANDS	13
4.7.1.3 STATUS_REGISTER	13
4.7.1.4 WRITE_PROTECT_STATE	14
4.7.2 Function Documentation	14
4.7.2.1 framChipSelect()	14
4.7.2.2 framReadMemory()	14
4.7.2.3 framReadSr()	14
4.7.2.4 framTest()	14
4.7.2.5 framWriteDisable()	15
4.7.2.6 framWriteEnable()	15
4.7.2.7 framWriteMemory()	15
4.7.2.8 framWriteProtect()	15
4.7.2.9 framWriteSr()	15
4.8 fram.h	16
4.9 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/logger.h File Reference	17
4.9.1 Function Documentation	17
4.9.1.1 LogNumber()	17
4.9.1.2 LogString()	17
4.9.1.3 ReadLog()	18
4.10 logger.h	19
4.11 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/main.h File Reference	19
4.11.1 Function Documentation	20
4.11.1.1 Error_Handler()	20
4.12 main.h	20
4.13 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/my_printf.h File Reference	21
4.14 my_printf.h	21
4.15 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/pwm_handler.h File Reference	21
4.15.1 Macro Definition Documentation	22
4.15.1.1 BRIGHTNESS_STEPS	22
4.15.1.2 HOLD_BRIGHTNESS_JUMP	22
4.15.2 Function Documentation	22
4.15.2.1 DecreaseBrightness()	22
4.15.2.2 GetBrightness()	22
4.15.2.3 GetPwm()	23
4.15.2.4 IncreaseBrightness()	23
4.15.2.5 InitPwm()	23
4.15.2.6 SetBrightness()	23
4.15.2.7 SetPwm()	24
4.15.2.8 TurnOffPwm()	24
4.16 pwm_handler.h	24

4.17	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/stm32l412xx-bsp.h	
	File Reference	25
4.17.1	Macro Definition Documentation	26
4.17.1.1	AMPMETER_ADC_GPIO_Port	26
4.17.1.2	AMPMETER_ADC_Pin	27
4.17.1.3	BRIGHT_GPIO_Port	27
4.17.1.4	BRIGHT_Pin	27
4.17.1.5	CLK_FREQ_HZ	27
4.17.1.6	DIM_GPIO_Port	27
4.17.1.7	DIM_Pin	27
4.17.1.8	EEPROM_MISO_GPIO_Port	27
4.17.1.9	EEPROM_MISO_Pin	27
4.17.1.10	EEPROM_MOSI_GPIO_Port	27
4.17.1.11	EEPROM_MOSI_Pin	27
4.17.1.12	EEPROM_SCK_GPIO_Port	28
4.17.1.13	EEPROM_SCK_Pin	28
4.17.1.14	NVIC_PRIORITYGROUP_0	28
4.17.1.15	NVIC_PRIORITYGROUP_1	28
4.17.1.16	NVIC_PRIORITYGROUP_2	28
4.17.1.17	NVIC_PRIORITYGROUP_3	28
4.17.1.18	NVIC_PRIORITYGROUP_4	28
4.17.1.19	PWM_OUT_GPIO_Port	28
4.17.1.20	PWM_OUT_Pin	29
4.17.1.21	SPI_NSS_GPIO_Port	29
4.17.1.22	SPI_NSS_Pin	29
4.17.1.23	SPI_WP_GPIO_Port	29
4.17.1.24	SPI_WP_Pin	29
4.17.1.25	THERMISTOR_ADC_GPIO_Port	29
4.17.1.26	THERMISTOR_ADC_Pin	29
4.17.1.27	TIM2_CLK_DEV	29
4.17.1.28	TIM2_CLK_PRESCALER	29
4.17.1.29	VOLTMETER_ADC_GPIO_Port	29
4.17.1.30	VOLTMETER_ADC_Pin	30
4.17.2	Typedef Documentation	30
4.17.2.1	GPIO_PinState	30
4.17.3	Enumeration Type Documentation	30
4.17.3.1	anonymous enum	30
4.17.3.2	anonymous enum	30
4.17.4	Function Documentation	30
4.17.4.1	disableChipSelect()	30
4.17.4.2	DisablePWM1()	30
4.17.4.3	disableWriteProtect()	31

4.17.4.4	enableChipSelect()	31
4.17.4.5	EnablePWM1()	31
4.17.4.6	enableWriteProtect()	31
4.17.4.7	Error_Handler()	31
4.17.4.8	GetCurrentValue()	31
4.17.4.9	GetThermistorValue()	32
4.17.4.10	GetTIM2Cnt()	32
4.17.4.11	GetVoltageValue()	32
4.17.4.12	ReadBrightPin()	32
4.17.4.13	ReadDimPin()	33
4.17.4.14	receiveData()	33
4.17.4.15	RestartTIM2()	33
4.17.4.16	sendUARTChar()	33
4.17.4.17	SetPW11()	33
4.17.4.18	StartPWM11()	34
4.17.4.19	StartTIM2()	34
4.17.4.20	StopPWM11()	34
4.17.4.21	transferData()	34
4.18	stm32l412xx-bsp.h	35
4.19	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/temperature_↵ handler.h File Reference	37
4.19.1	Enumeration Type Documentation	37
4.19.1.1	TemperatureRange_e	37
4.19.2	Function Documentation	38
4.19.2.1	GetTemperature()	38
4.19.2.2	GetTemperatureRange()	38
4.20	temperature_handler.h	38
4.21	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/voltage_handler.h File Reference	39
4.21.1	Enumeration Type Documentation	39
4.21.1.1	VoltageRange_e	39
4.21.2	Function Documentation	40
4.21.2.1	GetVoltage()	40
4.21.2.2	GetVoltageRange()	40
4.22	voltage_handler.h	40
4.23	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/button_handler.c File Reference	41
4.23.1	Function Documentation	41
4.23.1.1	IsBrightPressed()	41
4.23.1.2	IsDimPressed()	41
4.24	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/current_handler.c File Reference	41
4.24.1	Function Documentation	42

4.24.1.1	GetCurrent()	42
4.24.1.2	GetCurrentRange()	42
4.24.2	Variable Documentation	42
4.24.2.1	CurrentErrorThreshold_dA	42
4.24.2.2	CurrentHighThreshold_dA	43
4.24.2.3	dAmpsToRaw	43
4.24.2.4	RawTodAmps	43
4.25	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/delay_handler.c File Reference	43
4.25.1	Function Documentation	43
4.25.1.1	BrightnessDelay()	43
4.25.1.2	DelayHit()	44
4.25.1.3	RestartDelayCounter()	44
4.25.1.4	StartDelayCounter()	44
4.25.2	Variable Documentation	44
4.25.2.1	Tim2ClkKhz	44
4.26	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/fram.c File Reference	44
4.26.1	Macro Definition Documentation	45
4.26.1.1	TADD	45
4.26.1.2	TLEN	45
4.26.2	Function Documentation	45
4.26.2.1	framChipSelect()	45
4.26.2.2	framReadMemory()	45
4.26.2.3	framReadSr()	46
4.26.2.4	framTest()	46
4.26.2.5	framWriteDisable()	46
4.26.2.6	framWriteEnable()	46
4.26.2.7	framWriteMemory()	46
4.26.2.8	framWriteProtect()	47
4.26.2.9	framWriteSr()	47
4.27	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/logger.c File Reference	47
4.27.1	Function Documentation	47
4.27.1.1	LogNumber()	47
4.27.1.2	LogString()	47
4.27.1.3	ReadLog()	48
4.28	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/main.c File Reference	48
4.28.1	Macro Definition Documentation	49
4.28.1.1	AVG_STEP_DIFF_MS	49
4.28.1.2	AVG_STEP_TIME_MS	49
4.28.1.3	LOWER_STEP_TIME_MS	49
4.28.1.4	LOWER_SWEEP_TIME_MS	49
4.28.1.5	TOTAL_SWEEP_TIME_MS	49

4.28.1.6 UPPER_STEP_TIME_MS	49
4.28.1.7 UPPER_SWEEP_TIME_MS	49
4.28.2 Function Documentation	49
4.28.2.1 main()	49
4.28.2.2 SystemClock_Config()	50
4.28.3 Variable Documentation	50
4.28.3.1 HighStepTimeMs	50
4.28.3.2 LowStepTimeMs	50
4.29 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/my_printf.c File Reference	50
4.30 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/pwm_handler.c File Reference	50
4.30.1 Macro Definition Documentation	51
4.30.1.1 PW_PERIOD	51
4.30.2 Function Documentation	51
4.30.2.1 DecreaseBrightness()	51
4.30.2.2 GetBrightness()	52
4.30.2.3 GetPwm()	52
4.30.2.4 IncreaseBrightness()	52
4.30.2.5 InitPwm()	52
4.30.2.6 SetBrightness()	52
4.30.2.7 SetPwm()	53
4.30.2.8 TurnOffPwm()	53
4.30.3 Variable Documentation	53
4.30.3.1 HalfBrightness	53
4.30.3.2 HotPwmRatio	53
4.30.3.3 MaxBrightness	53
4.30.3.4 MaxPw	53
4.30.3.5 MinBrightness	54
4.30.3.6 MinPw	54
4.30.3.7 WarmPwmRatio	54
4.31 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/stm32l412xx-bsp.c File Reference	54
4.31.1 Function Documentation	55
4.31.1.1 disableChipSelect()	55
4.31.1.2 DisablePWM1()	55
4.31.1.3 disableWriteProtect()	55
4.31.1.4 enableChipSelect()	56
4.31.1.5 EnablePWM1()	56
4.31.1.6 enableWriteProtect()	56
4.31.1.7 Error_Handler()	56
4.31.1.8 GetCurrentValue()	56
4.31.1.9 GetThermistorValue()	56

4.31.1.10	GetTIM2Cnt()	57
4.31.1.11	GetVoltageValue()	57
4.31.1.12	ReadBrightPin()	57
4.31.1.13	ReadDimPin()	57
4.31.1.14	receiveData()	58
4.31.1.15	RestartTIM2()	58
4.31.1.16	sendUARTChar()	58
4.31.1.17	SetPW11()	58
4.31.1.18	StartPWM11()	59
4.31.1.19	StartTIM2()	59
4.31.1.20	StopPWM11()	59
4.31.1.21	transferData()	59
4.32	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/temperature_↔ handler.c File Reference	59
4.32.1	Function Documentation	60
4.32.1.1	GetTemperature()	60
4.32.1.2	GetTemperatureRange()	60
4.32.2	Variable Documentation	60
4.32.2.1	CoolingThreshold1_dC	60
4.32.2.2	CoolingThreshold2_dC	61
4.32.2.3	dCelciusToThermistor	61
4.32.2.4	HeatingThreshold1_dC	61
4.32.2.5	HeatingThreshold2_dC	61
4.32.2.6	ThermistorTodCelcius	61
4.33	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/voltage_handler.c File Reference	61
4.33.1	Function Documentation	62
4.33.1.1	GetVoltage()	62
4.33.1.2	GetVoltageRange()	62
4.33.2	Variable Documentation	62
4.33.2.1	dVoltsToRaw	62
4.33.2.2	RawTodVolts	62
4.33.2.3	VoltageErrorHighThreshold_dV	63
4.33.2.4	VoltageErrorLowThreshold_dV	63
4.33.2.5	VoltageHighThreshold_dV	63
4.33.2.6	VoltageLowThreshold_dV	63

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

PwmStruct	5
TimerStruct	6

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/button_handler.h	7
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/current_handler.h	8
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/delay_handler.h	10
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/fram.h	12
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/logger.h	17
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/main.h	19
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/my_printf.h	21
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/pwm_handler.h	21
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/stm32l412xx-bsp.h	25
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/temperature_handler.h	37
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/voltage_handler.h	39
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/button_handler.c	41
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/current_handler.c	41
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/delay_handler.c	43
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/fram.c	44
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/logger.c	47
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/main.c	48
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/my_printf.c	50
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/pwm_handler.c	50
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/stm32l412xx-bsp.c	54
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/temperature_handler.c	59
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/voltage_handler.c	61

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 "stm321412xx-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	<i>Bright</i>	Pin State, pressed or not
-----	---------------	---------------------------

4.1.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (  
    void )
```

Return state of dim button.

Parameters

out	<i>Dim</i>	Pin State, pressed or not
-----	------------	---------------------------

4.2 button_handler.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 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	<i>current</i>	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	<i>Current</i>	current range
-----	----------------	---------------

4.4 current_handler.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 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
00030 typedef enum
00031 {
00032     CurrentNormal = 0,
00033     CurrentHigh = 1,
00034     CurrentError = 2
00035 } CurrentRange_e;
00036
00037 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	<i>brightness</i>	Current brightness level
out	<i>Returns</i>	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	<i>delay_ms</i>	Time in ms to check if timer has hit
out	<i>Returns</i>	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 (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 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

<code>CSS_ASSERT</code>	chip select disabled
<code>CSS_RELEASE</code>	chip select enabled

4.7.1.2 OPCODE_COMMANDS

enum `OPCODE_COMMANDS`

opcode command

Enumerator

<code>OC_WREN</code>	set write enable latch
<code>OC_WRDI</code>	write disable
<code>OC_RDSR</code>	read status register
<code>OC_WRSR</code>	write status register
<code>OC_READ</code>	read memory data
<code>OC_WRITE</code>	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	<i>none</i>	
-----	-------------	--

4.7.2.6 framWriteEnable()

```
void framWriteEnable (
    void )
```

This routine resets the write enable latch.

Parameters

out	<i>none</i>	
-----	-------------	--

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) ----- */
00032  typedef enum {
00033
00034      OC_WREN = 6 ,
00035      OC_WRDI = 4 ,
00036      OC_RDSR = 5 ,
00037      OC_WRSR = 1 ,
00038      OC_READ = 3 ,
00039      OC_WRITE = 2 ,
00040  } OP_CODE_COMMANDS ;
00041
00042  typedef enum {
00043
00044      SR_WEL = 0x2 ,
00045      SR_BP0 = 0x4 ,
00046      SR_BP1 = 0x8 ,
00047      SR_WPEN = 0x80 ,
00048  } STATUS_REGISTER ;
00049
00050  typedef enum {
00051
00052      WPS_PROTECTED = 0 ,
00053      WPS_WRITEABLE = 1 ,
00054  } WRITE_PROTECT_STATE ;
00055
00056  typedef enum {
00057
00058      CSS_ASSERT = 0 ,
00059      CSS_RELEASE = 1 ,
00060  } CHIP_SELECT_STATE ;
00061
00062  /* Prototype Definition */
00063  void framWriteProtect( WRITE_PROTECT_STATE state ) ;
00064
00065  void framChipSelect( CHIP_SELECT_STATE state ) ;
00066
00067  void framReadSr( unsigned char *srP ) ;
00068
00069  void framWriteSr( unsigned char sr ) ;
00070
00071  void framWriteDisable( void ) ;
00072
00073  void framWriteEnable( void ) ;
00074
00075  void framReadMemory ( unsigned short addr, unsigned char *rdBufP, unsigned short len ) ;
00076
00077  void framWriteMemory( unsigned short addr, const unsigned char* const wrBufP, unsigned short len ) ;
00078
00079  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	<i>number</i>	Number to log
in	<i>write_beginning</i>	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	<i>string</i>	Pointer to string to log
in	<i>write_beginning</i>	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	<i>address</i>	Address to read from
in	<i>string</i>	Pointer to return data string
in	<i>bytes</i>	Number of bytes to read

4.10 logger.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 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     #ifndef __cplusplus
00026     extern "C" {
00027     #endif
00028
00029     /* Includes -----*/
00030
00031     /* Private includes -----*/
00032     /* USER CODE BEGIN Includes */
00033     #include "stm32l412xx-bsp.h"
00034     /* USER CODE END Includes */
00035
00036     /* Exported types -----*/
00037     /* USER CODE BEGIN ET */
00038
00039     /* USER CODE END ET */
00040
00041     /* Exported constants -----*/
00042     /* USER CODE BEGIN EC */
00043
00044     /* USER CODE END EC */
00045
00046     /* Exported macro -----*/
00047     /* USER CODE BEGIN EM */
00048
00049     /* USER CODE END EM */
00050
00051     /* Exported functions prototypes -----*/
00052     void Error_Handler(void);
00053
00054     /* USER CODE BEGIN EFP */
00055
00056     /* USER CODE END EFP */
00057
```

```

00058
00059     /* Private defines -----*/
00060
00061     /* USER CODE BEGIN Private defines */
00062
00063     /* USER CODE END Private defines */
00064
00065     #ifndef __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 (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 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	<i>button_held</i>	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	<i>Current</i>	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	<i>Current</i>	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	<i>button_held</i>	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	<i>brightness</i>	Brightness 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 (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 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	<i>Ammeter</i>	raw ADC value
-----	----------------	---------------

4.17.4.9 GetThermistorValue()

```
int16_t GetThermistorValue (  
    void )
```

Returns raw ADC value from thermistor.

Parameters

out	<i>Thermistor</i>	raw ADC value
-----	-------------------	---------------

4.17.4.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (  
    void )
```

Returns value in the Timer 2 counter.

Parameters

out	<i>Value</i>	of Timer 2 counter
-----	--------------	--------------------

4.17.4.11 GetVoltageValue()

```
int16_t GetVoltageValue (  
    void )
```

Returns raw ADC value from voltmeter.

Parameters

out	<i>Voltmeter</i>	raw ADC value
-----	------------------	---------------

4.17.4.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (  
    void )
```

Reads bright pin value.

Parameters

out	<i>Bright</i>	pin state
-----	---------------	-----------

4.17.4.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (
    void )
```

Reads dim pin value.

Parameters

out	<i>Dim</i>	pin state
-----	------------	-----------

4.17.4.14 receiveData()

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

Gets data from SPI lines.

Parameters

in	<i>rxData</i>	Pointer to data buffer
in	<i>bytes</i>	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	<i>c</i>	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	<i>pulse_width</i>	Value out of 255 to set pulse width to
----	--------------------	--

4.17.4.18 StartPWM1()

```
void StartPWM1 (  
    void )
```

Starts PWM Timer 1 Channel 1 output.

4.17.4.19 StartTIM2()

```
void StartTIM2 (  
    void )
```

Starts Timer 2 counter.

4.17.4.20 StopPWM1()

```
void StopPWM1 (  
    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	<i>txData</i>	Pointer to data to send
in	<i>bytes</i>	Number of bytes to send

4.18 stm32l412xx-bsp.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 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
00112     typedef struct
00113     {
00114         uint8_t is_running;
00115         uint32_t pulse_width;
00116     } PwmStruct;
00117
00118     typedef struct
00119     {
00120         uint8_t is_running;
00121         uint32_t time;
00122     } TimerStruct;
00123
00124 #endif /* STM32L412xx */
00125
00126 /* Interrupt Handlers */
00127 #ifndef NVIC_PRIORITYGROUP_0
00128     #define NVIC_PRIORITYGROUP_0 ((uint32_t)0x00000007)
00129     #define NVIC_PRIORITYGROUP_1 ((uint32_t)0x00000006)
00130     #define NVIC_PRIORITYGROUP_2 ((uint32_t)0x00000005)
00131     #define NVIC_PRIORITYGROUP_3 ((uint32_t)0x00000004)
00132     #define NVIC_PRIORITYGROUP_4 ((uint32_t)0x00000003)
00133 #endif
00134
00135 /* Button Defines */
00136 typedef uint8_t GPIO_PinState;
00137 enum { BUTTON_PRESSED = 1, BUTTON_UNPRESSED = 0 };
00138 enum { PIN_SET = 1, PIN_RESET = 0 };
00139
00140 /* Clock frequency Values */
00141 #define CLK_FREQ_HZ (8000000)
00142 #define TIM2_CLK_DEV (1)
00143 #define TIM2_CLK_PRESCALER (8000)
00144
00145 /* Returns button state */
00146 GPIO_PinState ReadDimPin( void );
00147
00148 GPIO_PinState ReadBrightPin( void );
00149 /* PWM Outputs */
00150 void EnablePWM1( void );
00151
00152 void DisablePWM1( void );
00153
00154 void StartPWM1( void );
00155
00156 void StopPWM1( void );
00157
00158 void SetPW1( uint32_t pulse_width );
00159
00160 /* Timers */
00161 void StartTIM2( void );
00162
00163 void RestartTIM2( void );
00164
00165 uint32_t GetTIM2Cnt( void );
00166
00167 int16_t GetThermistorValue( void );
00168
00169 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	<i>temperature</i>	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	<i>Current</i>	temperature range
-----	----------------	-------------------

4.20 temperature_handler.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 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
00032     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

<code>VoltageNormal</code>	Normal Operating Voltage
<code>VoltageLow</code>	Voltage low, but ok
<code>VoltageHigh</code>	Voltage high, but ok
<code>VoltageErrorLow</code>	Voltage too low
<code>VoltageErrorHigh</code>	Voltage too high

4.21.2 Function Documentation

4.21.2.1 GetVoltage()

```
uint16_t GetVoltage (
    void )
```

Get voltage from voltmeter.

Parameters

out	<i>voltage</i>	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	<i>Current</i>	voltage range
-----	----------------	---------------

4.22 voltage_handler.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 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
00030 typedef enum
00031 {
00032     VoltageNormal      = 0,
00033     VoltageLow         = 1,
00034     VoltageHigh        = 2,
00035     VoltageErrorLow     = 3,
00036     VoltageErrorHigh    = 4
00037 } VoltageRange_e;
00038
00039 uint16_t GetVoltage( void );
00040
00041 VoltageRange_e GetVoltageRange( void );
00042
00043 #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	<i>Bright</i>	Pin State, pressed or not
-----	---------------	---------------------------

4.23.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (  
    void )
```

Return state of dim button.

Parameters

out	<i>Dim</i>	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	<i>current</i>	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	<i>Current</i>	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	<i>brightness</i>	Current brightness level
out	<i>Returns</i>	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	<i>delay_ms</i>	Time in ms to check if timer has hit
out	<i>Returns</i>	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	<i>none</i>	
-----	-------------	--

4.26.2.6 framWriteEnable()

```
void framWriteEnable (
    void )
```

This routine resets the write enable latch.

Parameters

out	<i>none</i>	
-----	-------------	--

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	<i>number</i>	Number to log
in	<i>write_beginning</i>	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	<i>string</i>	Pointer to string to log
in	<i>write_beginning</i>	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	<i>address</i>	Address to read from
in	<i>string</i>	Pointer to return data string
in	<i>bytes</i>	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

<i>int</i>	
------------	--

4.28.2.2 SystemClock_Config()

```
void SystemClock_Config (
    void )
```

System Clock Configuration.

Return values

<i>None</i>	
-------------	--

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	<i>button_held</i>	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	<i>Current</i>	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	<i>Current</i>	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	<i>button_held</i>	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	<i>brightness</i>	Brightness 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

<i>None</i>	
-------------	--

4.31.1.8 GetCurrentValue()

```
int16_t GetCurrentValue (
    void )
```

Returns raw ADC value from ammeter.

Parameters

out	<i>Ammeter</i>	raw ADC value
-----	----------------	---------------

4.31.1.9 GetThermistorValue()

```
int16_t GetThermistorValue (
    void )
```

Returns raw ADC value from thermistor.

Parameters

out	<i>Thermistor</i>	raw ADC value
-----	-------------------	---------------

4.31.1.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (  
    void )
```

Returns value in the Timer 2 counter.

Parameters

out	<i>Value</i>	of Timer 2 counter
-----	--------------	--------------------

4.31.1.11 GetVoltageValue()

```
int16_t GetVoltageValue (  
    void )
```

Returns raw ADC value from voltmeter.

Parameters

out	<i>Voltmeter</i>	raw ADC value
-----	------------------	---------------

4.31.1.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (  
    void )
```

Reads bright pin value.

Parameters

out	<i>Bright</i>	pin state
-----	---------------	-----------

4.31.1.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (  
    void )
```

Reads dim pin value.

Parameters

out	<i>Dim</i>	pin state
-----	------------	-----------

4.31.1.14 receiveData()

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

Gets data from SPI lines.

Parameters

in	<i>rxData</i>	Pointer to data buffer
in	<i>bytes</i>	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	<i>c</i>	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	<i>pulse_width</i>	Value out of 255 to set pulse width to
----	--------------------	--

4.31.1.18 StartPWM1()

```
void StartPWM1 (
    void )
```

Starts PWM Timer 1 Channel 1 output.

4.31.1.19 StartTIM2()

```
void StartTIM2 (
    void )
```

Starts Timer 2 counter.

4.31.1.20 StopPWM1()

```
void StopPWM1 (
    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	<i>txData</i>	Pointer to data to send
in	<i>bytes</i>	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	<i>temperature</i>	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	<i>Current</i>	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	<i>voltage</i>	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	<i>Current</i>	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

AMPMETER_ADC_GPIO_Port
 stm32l412xx-bsp.h, [26](#)
AMPMETER_ADC_Pin
 stm32l412xx-bsp.h, [26](#)
AVG_STEP_DIFF_MS
 main.c, [49](#)
AVG_STEP_TIME_MS
 main.c, [49](#)

BRIGHT_GPIO_Port
 stm32l412xx-bsp.h, [27](#)
BRIGHT_Pin
 stm32l412xx-bsp.h, [27](#)
BRIGHTNESS_STEPS
 pwm_handler.h, [22](#)
BrightnessDelay
 delay_handler.c, [43](#)
 delay_handler.h, [11](#)
button_handler.c
 IsBrightPressed, [41](#)
 IsDimPressed, [41](#)
button_handler.h
 IsBrightPressed, [7](#)
 IsDimPressed, [7](#)
BUTTON_PRESSED
 stm32l412xx-bsp.h, [30](#)
BUTTON_UNPRESSED
 stm32l412xx-bsp.h, [30](#)

C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [7](#), [8](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [8](#), [10](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [10](#), [12](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [12](#), [16](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [17](#), [19](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [19](#), [20](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [21](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [21](#), [24](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 bsp.h, [25](#), [35](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [37](#), [38](#)

C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [39](#), [40](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [41](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [41](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [43](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [44](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [47](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [48](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [50](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [50](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 bsp.c, [54](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [59](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [61](#)
CHIP_SELECT_STATE
 fram.h, [13](#)
CLK_FREQ_HZ
 stm32l412xx-bsp.h, [27](#)
CoolingThreshold1_dC
 Controller/Core/Inc/button_handler.h,
 temperature_handler.c, [80](#)
CoolingThreshold2_dC
 Controller/Core/Inc/current_handler.h,
 temperature_handler.c, [80](#)
CSS_ASSERT
 fram.h, [13](#)
CSS_RELEASE
 fram.h, [13](#)
current_handler.c
 Controller/Core/Inc/logger.h,
 CurrentErrorThreshold_dA, [42](#)
 CurrentHighThreshold_dA, [42](#)
 dAmpsToRaw, [43](#)
 GetCurrent, [42](#)
 GetCurrentRange, [42](#)
 RawTodAmps, [43](#)
 Controller/Core/Inc/pwm_handler.h,
 current_handler.h
 CurrentError, [9](#)
 CurrentHigh, [9](#)
 CurrentNormal, [9](#)
 CurrentRange_e, [9](#)
 GetCurrent, [9](#)

- GetCurrentRange, [9](#)
- CurrentError
 - current_handler.h, [9](#)
- CurrentErrorThreshold_dA
 - current_handler.c, [42](#)
- CurrentHigh
 - current_handler.h, [9](#)
- CurrentHighThreshold_dA
 - current_handler.c, [42](#)
- CurrentNormal
 - current_handler.h, [9](#)
- CurrentRange_e
 - current_handler.h, [9](#)
- dAmpsToRaw
 - current_handler.c, [43](#)
- dCelciusToThermistor
 - temperature_handler.c, [61](#)
- DecreaseBrightness
 - pwm_handler.c, [51](#)
 - pwm_handler.h, [22](#)
- delay_handler.c
 - BrightnessDelay, [43](#)
 - DelayHit, [44](#)
 - RestartDelayCounter, [44](#)
 - StartDelayCounter, [44](#)
 - Tim2ClkKhz, [44](#)
- delay_handler.h
 - BrightnessDelay, [11](#)
 - DelayHit, [11](#)
 - RestartDelayCounter, [11](#)
 - StartDelayCounter, [11](#)
- DelayHit
 - delay_handler.c, [44](#)
 - delay_handler.h, [11](#)
- DIM_GPIO_Port
 - stm32l412xx-bsp.h, [27](#)
- DIM_Pin
 - stm32l412xx-bsp.h, [27](#)
- disableChipSelect
 - stm32l412xx-bsp.c, [55](#)
 - stm32l412xx-bsp.h, [30](#)
- DisablePWM1
 - stm32l412xx-bsp.c, [55](#)
 - stm32l412xx-bsp.h, [30](#)
- disableWriteProtect
 - stm32l412xx-bsp.c, [55](#)
 - stm32l412xx-bsp.h, [30](#)
- dVoltsToRaw
 - voltage_handler.c, [62](#)
- EEPROM_MISO_GPIO_Port
 - stm32l412xx-bsp.h, [27](#)
- EEPROM_MISO_Pin
 - stm32l412xx-bsp.h, [27](#)
- EEPROM_MOSI_GPIO_Port
 - stm32l412xx-bsp.h, [27](#)
- EEPROM_MOSI_Pin
 - stm32l412xx-bsp.h, [27](#)
- EEPROM_SCK_GPIO_Port
 - stm32l412xx-bsp.h, [27](#)
- EEPROM_SCK_Pin
 - stm32l412xx-bsp.h, [28](#)
- enableChipSelect
 - stm32l412xx-bsp.c, [55](#)
 - stm32l412xx-bsp.h, [31](#)
- EnablePWM1
 - stm32l412xx-bsp.c, [56](#)
 - stm32l412xx-bsp.h, [31](#)
- enableWriteProtect
 - stm32l412xx-bsp.c, [56](#)
 - stm32l412xx-bsp.h, [31](#)
- Error_Handler
 - main.h, [20](#)
 - stm32l412xx-bsp.c, [56](#)
 - stm32l412xx-bsp.h, [31](#)
- fram.c
 - framChipSelect, [45](#)
 - framReadMemory, [45](#)
 - framReadSr, [45](#)
 - framTest, [46](#)
 - framWriteDisable, [46](#)
 - framWriteEnable, [46](#)
 - framWriteMemory, [46](#)
 - framWriteProtect, [46](#)
 - framWriteSr, [47](#)
 - TADD, [45](#)
 - TLEN, [45](#)
- fram.h
 - CHIP_SELECT_STATE, [13](#)
 - CSS_ASSERT, [13](#)
 - CSS_RELEASE, [13](#)
 - framChipSelect, [14](#)
 - framReadMemory, [14](#)
 - framReadSr, [14](#)
 - framTest, [14](#)
 - framWriteDisable, [15](#)
 - framWriteEnable, [15](#)
 - framWriteMemory, [15](#)
 - framWriteProtect, [15](#)
 - framWriteSr, [15](#)
 - OC_RD SR, [13](#)
 - OC_READ, [13](#)
 - OC_WRDI, [13](#)
 - OC_WREN, [13](#)
 - OC_WRITE, [13](#)
 - OC_WRSR, [13](#)
 - OPCODE_COMMANDS, [13](#)
 - SR_BP0, [14](#)
 - SR_BP1, [14](#)
 - SR_WEL, [14](#)
 - SR_WPEN, [14](#)
 - STATUS_REGISTER, [13](#)
 - WPS_PROTECTED, [14](#)
 - WPS_WRITEABLE, [14](#)
 - WRITE_PROTECT_STATE, [14](#)
- framChipSelect

- fram.c, [45](#)
- fram.h, [14](#)
- framReadMemory
 - fram.c, [45](#)
 - fram.h, [14](#)
- framReadSr
 - fram.c, [45](#)
 - fram.h, [14](#)
- framTest
 - fram.c, [46](#)
 - fram.h, [14](#)
- framWriteDisable
 - fram.c, [46](#)
 - fram.h, [15](#)
- framWriteEnable
 - fram.c, [46](#)
 - fram.h, [15](#)
- framWriteMemory
 - fram.c, [46](#)
 - fram.h, [15](#)
- framWriteProtect
 - fram.c, [46](#)
 - fram.h, [15](#)
- framWriteSr
 - fram.c, [47](#)
 - fram.h, [15](#)
- GetBrightness
 - pwm_handler.c, [52](#)
 - pwm_handler.h, [22](#)
- GetCurrent
 - current_handler.c, [42](#)
 - current_handler.h, [9](#)
- GetCurrentRange
 - current_handler.c, [42](#)
 - current_handler.h, [9](#)
- GetCurrentValue
 - stm32l412xx-bsp.c, [56](#)
 - stm32l412xx-bsp.h, [31](#)
- GetPwm
 - pwm_handler.c, [52](#)
 - pwm_handler.h, [23](#)
- GetTemperature
 - temperature_handler.c, [60](#)
 - temperature_handler.h, [38](#)
- GetTemperatureRange
 - temperature_handler.c, [60](#)
 - temperature_handler.h, [38](#)
- GetThermistorValue
 - stm32l412xx-bsp.c, [56](#)
 - stm32l412xx-bsp.h, [32](#)
- GetTIM2Cnt
 - stm32l412xx-bsp.c, [57](#)
 - stm32l412xx-bsp.h, [32](#)
- GetVoltage
 - voltage_handler.c, [62](#)
 - voltage_handler.h, [40](#)
- GetVoltageRange
 - voltage_handler.c, [62](#)
- voltage_handler.h, [40](#)
- GetVoltageValue
 - stm32l412xx-bsp.c, [57](#)
 - stm32l412xx-bsp.h, [32](#)
- GPIO_PinState
 - stm32l412xx-bsp.h, [30](#)
- HalfBrightness
 - pwm_handler.c, [53](#)
- HeatingThreshold1_dC
 - temperature_handler.c, [61](#)
- HeatingThreshold2_dC
 - temperature_handler.c, [61](#)
- HighStepTimeMs
 - main.c, [50](#)
- HOLD_BRIGHTNESS_JUMP
 - pwm_handler.h, [22](#)
- HotPwmRatio
 - pwm_handler.c, [53](#)
- IncreaseBrightness
 - pwm_handler.c, [52](#)
 - pwm_handler.h, [23](#)
- InitPwm
 - pwm_handler.c, [52](#)
 - pwm_handler.h, [23](#)
- is_running
 - PwmStruct, [5](#)
 - TimerStruct, [6](#)
- IsBrightPressed
 - button_handler.c, [41](#)
 - button_handler.h, [7](#)
- IsDimPressed
 - button_handler.c, [41](#)
 - button_handler.h, [7](#)
- logger.c
 - LogNumber, [47](#)
 - LogString, [47](#)
 - ReadLog, [48](#)
- logger.h
 - LogNumber, [17](#)
 - LogString, [17](#)
 - ReadLog, [17](#)
- LogNumber
 - logger.c, [47](#)
 - logger.h, [17](#)
- LogString
 - logger.c, [47](#)
 - logger.h, [17](#)
- LOWER_STEP_TIME_MS
 - main.c, [49](#)
- LOWER_SWEEP_TIME_MS
 - main.c, [49](#)
- LowStepTimeMs
 - main.c, [50](#)
- main
 - main.c, [49](#)

- main.c
 - AVG_STEP_DIFF_MS, [49](#)
 - AVG_STEP_TIME_MS, [49](#)
 - HighStepTimeMs, [50](#)
 - LOWER_STEP_TIME_MS, [49](#)
 - LOWER_SWEEP_TIME_MS, [49](#)
 - LowStepTimeMs, [50](#)
 - main, [49](#)
 - SystemClock_Config, [50](#)
 - TOTAL_SWEEP_TIME_MS, [49](#)
 - UPPER_STEP_TIME_MS, [49](#)
 - UPPER_SWEEP_TIME_MS, [49](#)
- main.h
 - Error_Handler, [20](#)
- MaxBrightness
 - pwm_handler.c, [53](#)
- MaxPw
 - pwm_handler.c, [53](#)
- MinBrightness
 - pwm_handler.c, [53](#)
- MinPw
 - pwm_handler.c, [54](#)
- NVIC_PRIORITYGROUP_0
 - stm32l412xx-bsp.h, [28](#)
- NVIC_PRIORITYGROUP_1
 - stm32l412xx-bsp.h, [28](#)
- NVIC_PRIORITYGROUP_2
 - stm32l412xx-bsp.h, [28](#)
- NVIC_PRIORITYGROUP_3
 - stm32l412xx-bsp.h, [28](#)
- NVIC_PRIORITYGROUP_4
 - stm32l412xx-bsp.h, [28](#)
- OC_RDSR
 - fram.h, [13](#)
- OC_READ
 - fram.h, [13](#)
- OC_WRDI
 - fram.h, [13](#)
- OC_WREN
 - fram.h, [13](#)
- OC_WRITE
 - fram.h, [13](#)
- OC_WRSR
 - fram.h, [13](#)
- OPCODE_COMMANDS
 - fram.h, [13](#)
- PIN_RESET
 - stm32l412xx-bsp.h, [30](#)
- PIN_SET
 - stm32l412xx-bsp.h, [30](#)
- pulse_width
 - PwmStruct, [5](#)
- PW_PERIOD
 - pwm_handler.c, [51](#)
- pwm_handler.c
 - DecreaseBrightness, [51](#)
 - GetBrightness, [52](#)
 - GetPwm, [52](#)
 - HalfBrightness, [53](#)
 - HotPwmRatio, [53](#)
 - IncreaseBrightness, [52](#)
 - InitPwm, [52](#)
 - MaxBrightness, [53](#)
 - MaxPw, [53](#)
 - MinBrightness, [53](#)
 - MinPw, [54](#)
 - PW_PERIOD, [51](#)
 - SetBrightness, [52](#)
 - SetPwm, [53](#)
 - TurnOffPwm, [53](#)
 - WarmPwmRatio, [54](#)
- pwm_handler.h
 - BRIGHTNESS_STEPS, [22](#)
 - DecreaseBrightness, [22](#)
 - GetBrightness, [22](#)
 - GetPwm, [23](#)
 - HOLD_BRIGHTNESS_JUMP, [22](#)
 - IncreaseBrightness, [23](#)
 - InitPwm, [23](#)
 - SetBrightness, [23](#)
 - SetPwm, [23](#)
 - TurnOffPwm, [24](#)
- PWM_OUT_GPIO_Port
 - stm32l412xx-bsp.h, [28](#)
- PWM_OUT_Pin
 - stm32l412xx-bsp.h, [28](#)
- PwmStruct, [5](#)
 - is_running, [5](#)
 - pulse_width, [5](#)
- RawTodAmps
 - current_handler.c, [43](#)
- RawTodVolts
 - voltage_handler.c, [62](#)
- ReadBrightPin
 - stm32l412xx-bsp.c, [57](#)
 - stm32l412xx-bsp.h, [32](#)
- ReadDimPin
 - stm32l412xx-bsp.c, [57](#)
 - stm32l412xx-bsp.h, [33](#)
- ReadLog
 - logger.c, [48](#)
 - logger.h, [17](#)
- receiveData
 - stm32l412xx-bsp.c, [58](#)
 - stm32l412xx-bsp.h, [33](#)
- RestartDelayCounter
 - delay_handler.c, [44](#)
 - delay_handler.h, [11](#)
- RestartTIM2
 - stm32l412xx-bsp.c, [58](#)
 - stm32l412xx-bsp.h, [33](#)
- sendUARTChar
 - stm32l412xx-bsp.c, [58](#)

- stm32l412xx-bsp.h, [33](#)
- SetBrightness
 - pwm_handler.c, [52](#)
 - pwm_handler.h, [23](#)
- SetPW11
 - stm32l412xx-bsp.c, [58](#)
 - stm32l412xx-bsp.h, [33](#)
- SetPwm
 - pwm_handler.c, [53](#)
 - pwm_handler.h, [23](#)
- SPI_NSS_GPIO_Port
 - stm32l412xx-bsp.h, [29](#)
- SPI_NSS_Pin
 - stm32l412xx-bsp.h, [29](#)
- SPI_WP_GPIO_Port
 - stm32l412xx-bsp.h, [29](#)
- SPI_WP_Pin
 - stm32l412xx-bsp.h, [29](#)
- SR_BP0
 - fram.h, [14](#)
- SR_BP1
 - fram.h, [14](#)
- SR_WEL
 - fram.h, [14](#)
- SR_WPEN
 - fram.h, [14](#)
- StartDelayCounter
 - delay_handler.c, [44](#)
 - delay_handler.h, [11](#)
- StartPWM11
 - stm32l412xx-bsp.c, [58](#)
 - stm32l412xx-bsp.h, [34](#)
- StartTIM2
 - stm32l412xx-bsp.c, [59](#)
 - stm32l412xx-bsp.h, [34](#)
- STATUS_REGISTER
 - fram.h, [13](#)
- stm32l412xx-bsp.c
 - disableChipSelect, [55](#)
 - DisablePWM1, [55](#)
 - disableWriteProtect, [55](#)
 - enableChipSelect, [55](#)
 - EnablePWM1, [56](#)
 - enableWriteProtect, [56](#)
 - Error_Handler, [56](#)
 - GetCurrentValue, [56](#)
 - GetThermistorValue, [56](#)
 - GetTIM2Cnt, [57](#)
 - GetVoltageValue, [57](#)
 - ReadBrightPin, [57](#)
 - ReadDimPin, [57](#)
 - receiveData, [58](#)
 - RestartTIM2, [58](#)
 - sendUARTChar, [58](#)
 - SetPW11, [58](#)
 - StartPWM11, [58](#)
 - StartTIM2, [59](#)
 - StopPWM11, [59](#)
- transferData, [59](#)
- stm32l412xx-bsp.h
 - AMPMETER_ADC_GPIO_Port, [26](#)
 - AMPMETER_ADC_Pin, [26](#)
 - BRIGHT_GPIO_Port, [27](#)
 - BRIGHT_Pin, [27](#)
 - BUTTON_PRESSED, [30](#)
 - BUTTON_UNPRESSED, [30](#)
 - CLK_FREQ_HZ, [27](#)
 - DIM_GPIO_Port, [27](#)
 - DIM_Pin, [27](#)
 - disableChipSelect, [30](#)
 - DisablePWM1, [30](#)
 - disableWriteProtect, [30](#)
 - EEPROM_MISO_GPIO_Port, [27](#)
 - EEPROM_MISO_Pin, [27](#)
 - EEPROM_MOSI_GPIO_Port, [27](#)
 - EEPROM_MOSI_Pin, [27](#)
 - EEPROM_SCK_GPIO_Port, [27](#)
 - EEPROM_SCK_Pin, [28](#)
 - enableChipSelect, [31](#)
 - EnablePWM1, [31](#)
 - enableWriteProtect, [31](#)
 - Error_Handler, [31](#)
 - GetCurrentValue, [31](#)
 - GetThermistorValue, [32](#)
 - GetTIM2Cnt, [32](#)
 - GetVoltageValue, [32](#)
 - GPIO_PinState, [30](#)
 - NVIC_PRIORITYGROUP_0, [28](#)
 - NVIC_PRIORITYGROUP_1, [28](#)
 - NVIC_PRIORITYGROUP_2, [28](#)
 - NVIC_PRIORITYGROUP_3, [28](#)
 - NVIC_PRIORITYGROUP_4, [28](#)
 - PIN_RESET, [30](#)
 - PIN_SET, [30](#)
 - PWM_OUT_GPIO_Port, [28](#)
 - PWM_OUT_Pin, [28](#)
 - ReadBrightPin, [32](#)
 - ReadDimPin, [33](#)
 - receiveData, [33](#)
 - RestartTIM2, [33](#)
 - sendUARTChar, [33](#)
 - SetPW11, [33](#)
 - SPI_NSS_GPIO_Port, [29](#)
 - SPI_NSS_Pin, [29](#)
 - SPI_WP_GPIO_Port, [29](#)
 - SPI_WP_Pin, [29](#)
 - StartPWM11, [34](#)
 - StartTIM2, [34](#)
 - StopPWM11, [34](#)
 - THERMISTOR_ADC_GPIO_Port, [29](#)
 - THERMISTOR_ADC_Pin, [29](#)
 - TIM2_CLK_DEV, [29](#)
 - TIM2_CLK_PRESCALER, [29](#)
 - transferData, [34](#)
 - VOLTMETER_ADC_GPIO_Port, [29](#)
 - VOLTMETER_ADC_Pin, [29](#)

- StopPWM11
 - stm32l412xx-bsp.c, [59](#)
 - stm32l412xx-bsp.h, [34](#)
- SystemClock_Config
 - main.c, [50](#)
- TADD
 - fram.c, [45](#)
- TempCool
 - temperature_handler.h, [37](#)
- temperature_handler.c
 - CoolingThreshold1_dC, [60](#)
 - CoolingThreshold2_dC, [60](#)
 - dCelciusToThermistor, [61](#)
 - GetTemperature, [60](#)
 - GetTemperatureRange, [60](#)
 - HeatingThreshold1_dC, [61](#)
 - HeatingThreshold2_dC, [61](#)
 - ThermistorTodCelcius, [61](#)
- temperature_handler.h
 - GetTemperature, [38](#)
 - GetTemperatureRange, [38](#)
 - TempCool, [37](#)
 - TemperatureRange_e, [37](#)
 - TempHot, [37](#)
 - TempWarm, [37](#)
- TemperatureRange_e
 - temperature_handler.h, [37](#)
- TempHot
 - temperature_handler.h, [37](#)
- TempWarm
 - temperature_handler.h, [37](#)
- THERMISTOR_ADC_GPIO_Port
 - stm32l412xx-bsp.h, [29](#)
- THERMISTOR_ADC_Pin
 - stm32l412xx-bsp.h, [29](#)
- ThermistorTodCelcius
 - temperature_handler.c, [61](#)
- TIM2_CLK_DEV
 - stm32l412xx-bsp.h, [29](#)
- TIM2_CLK_PRESCALER
 - stm32l412xx-bsp.h, [29](#)
- Tim2ClkKhz
 - delay_handler.c, [44](#)
- time
 - TimerStruct, [6](#)
- TimerStruct, [6](#)
 - is_running, [6](#)
 - time, [6](#)
- TLEN
 - fram.c, [45](#)
- TOTAL_SWEEP_TIME_MS
 - main.c, [49](#)
- transferData
 - stm32l412xx-bsp.c, [59](#)
 - stm32l412xx-bsp.h, [34](#)
- TurnOffPwm
 - pwm_handler.c, [53](#)
 - pwm_handler.h, [24](#)
- UPPER_STEP_TIME_MS
 - main.c, [49](#)
- UPPER_SWEEP_TIME_MS
 - main.c, [49](#)
- voltage_handler.c
 - dVoltsToRaw, [62](#)
 - GetVoltage, [62](#)
 - GetVoltageRange, [62](#)
 - RawTodVolts, [62](#)
 - VoltageErrorHighThreshold_dV, [62](#)
 - VoltageErrorLowThreshold_dV, [63](#)
 - VoltageHighThreshold_dV, [63](#)
 - VoltageLowThreshold_dV, [63](#)
- voltage_handler.h
 - GetVoltage, [40](#)
 - GetVoltageRange, [40](#)
 - VoltageErrorHigh, [39](#)
 - VoltageErrorLow, [39](#)
 - VoltageHigh, [39](#)
 - VoltageLow, [39](#)
 - VoltageNormal, [39](#)
 - VoltageRange_e, [39](#)
- VoltageErrorHigh
 - voltage_handler.h, [39](#)
- VoltageErrorHighThreshold_dV
 - voltage_handler.c, [62](#)
- VoltageErrorLow
 - voltage_handler.h, [39](#)
- VoltageErrorLowThreshold_dV
 - voltage_handler.c, [63](#)
- VoltageHigh
 - voltage_handler.h, [39](#)
- VoltageHighThreshold_dV
 - voltage_handler.c, [63](#)
- VoltageLow
 - voltage_handler.h, [39](#)
- VoltageLowThreshold_dV
 - voltage_handler.c, [63](#)
- VoltageNormal
 - voltage_handler.h, [39](#)
- VoltageRange_e
 - voltage_handler.h, [39](#)
- VOLTMETER_ADC_GPIO_Port
 - stm32l412xx-bsp.h, [29](#)
- VOLTMETER_ADC_Pin
 - stm32l412xx-bsp.h, [29](#)
- WarmPwmRatio
 - pwm_handler.c, [54](#)
- WPS_PROTECTED
 - fram.h, [14](#)
- WPS_WRITEABLE
 - fram.h, [14](#)
- WRITE_PROTECT_STATE
 - fram.h, [14](#)