

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 Field Documentation	5
3.1.1.1 is_running	5
3.1.1.2 pulse_width	5
3.2 TimerStruct Struct Reference	5
3.2.1 Field Documentation	6
3.2.1.1 is_running	6
3.2.1.2 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	8
4.3.1.1 CurrentRange_e	8
4.3.2 Function Documentation	9
4.3.2.1 GetCurrent()	9
4.3.2.2 GetCurrentRange()	9
4.4 current_handler.h	9
4.5 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/delay_handler.h File Reference	9
4.5.1 Function Documentation	10
4.5.1.1 BrightnessDelay()	10
4.5.1.2 DelayHit()	10
4.5.1.3 RestartDelayCounter()	10
4.5.1.4 StartDelayCounter()	10
4.6 delay_handler.h	10
4.7 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/fram.h File Reference	11
4.7.1 Detailed Description	11
4.7.2 Enumeration Type Documentation	11
4.7.2.1 CHIP_SELECT_STATE	11
4.7.2.2 OPCODE_COMMANDS	12

4.7.2.3 STATUS_REGISTER	12
4.7.2.4 WRITE_PROTECT_STATE	12
4.7.3 Function Documentation	13
4.7.3.1 framChipSelect()	13
4.7.3.2 framReadMemory()	13
4.7.3.3 framReadSr()	13
4.7.3.4 framTest()	13
4.7.3.5 framWriteDisable()	13
4.7.3.6 framWriteEnable()	14
4.7.3.7 framWriteMemory()	14
4.7.3.8 framWriteProtect()	14
4.7.3.9 framWriteSr()	14
4.8 fram.h	15
4.9 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/logger.h File Reference	15
4.9.1 Function Documentation	16
4.9.1.1 LogNumber()	16
4.9.1.2 LogString()	16
4.9.1.3 ReadLog()	16
4.10 logger.h	16
4.11 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/main.h File Reference	16
4.11.1 Function Documentation	17
4.11.1.1 Error_Handler()	17
4.12 main.h	17
4.13 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/my_printf.h File Reference	18
4.14 my_printf.h	18
4.15 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/pwm_handler.h File Reference	18
4.15.1 Macro Definition Documentation	19
4.15.1.1 BRIGHTNESS_STEPS	19
4.15.1.2 HOLD_BRIGHTNESS_JUMP	19
4.15.2 Function Documentation	19
4.15.2.1 DecreaseBrightness()	19
4.15.2.2 GetBrightness()	19
4.15.2.3 GetPwm()	19
4.15.2.4 IncreaseBrightness()	19
4.15.2.5 InitPwm()	19
4.15.2.6 SetBrightness()	19
4.15.2.7 SetPwm()	19
4.15.2.8 TurnOffPwm()	19
4.16 pwm_handler.h	20
4.17 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/stm32l412xx-bsp.h File Reference	20

4.17.1 Macro Definition Documentation	22
4.17.1.1 AMPMETER_ADC_GPIO_Port	22
4.17.1.2 AMPMETER_ADC_Pin	22
4.17.1.3 BRIGHT_GPIO_Port	22
4.17.1.4 BRIGHT_Pin	22
4.17.1.5 CLK_FREQ_HZ	22
4.17.1.6 DIM_GPIO_Port	22
4.17.1.7 DIM_Pin	22
4.17.1.8 EEPROM_MISO_GPIO_Port	22
4.17.1.9 EEPROM_MISO_Pin	22
4.17.1.10 EEPROM_MOSI_GPIO_Port	22
4.17.1.11 EEPROM_MOSI_Pin	22
4.17.1.12 EEPROM_SCK_GPIO_Port	22
4.17.1.13 EEPROM_SCK_Pin	22
4.17.1.14 NVIC_PRIORITYGROUP_0	22
4.17.1.15 NVIC_PRIORITYGROUP_1	23
4.17.1.16 NVIC_PRIORITYGROUP_2	23
4.17.1.17 NVIC_PRIORITYGROUP_3	23
4.17.1.18 NVIC_PRIORITYGROUP_4	23
4.17.1.19 PWM_OUT_GPIO_Port	23
4.17.1.20 PWM_OUT_Pin	23
4.17.1.21 SPI_NSS_GPIO_Port	23
4.17.1.22 SPI_NSS_Pin	23
4.17.1.23 SPI_WP_GPIO_Port	23
4.17.1.24 SPI_WP_Pin	23
4.17.1.25 THERMISTOR_ADC_GPIO_Port	23
4.17.1.26 THERMISTOR_ADC_Pin	23
4.17.1.27 TIM2_CLK_DEV	23
4.17.1.28 TIM2_CLK_PRESCALER	23
4.17.1.29 VOLTMETER_ADC_GPIO_Port	24
4.17.1.30 VOLTMETER_ADC_Pin	24
4.17.2 Typedef Documentation	24
4.17.2.1 GPIO_PinState	24
4.17.3 Enumeration Type Documentation	24
4.17.3.1 anonymous enum	24
4.17.3.2 anonymous enum	24
4.17.4 Function Documentation	24
4.17.4.1 disableChipSelect()	24
4.17.4.2 DisablePWM1()	24
4.17.4.3 disableWriteProtect()	24
4.17.4.4 enableChipSelect()	24
4.17.4.5 EnablePWM1()	25

4.17.4.6 enableWriteProtect()	25
4.17.4.7 Error_Handler()	25
4.17.4.8 GetCurrentValue()	25
4.17.4.9 GetThermistorValue()	25
4.17.4.10 GetTIM2Cnt()	25
4.17.4.11 GetVoltageValue()	25
4.17.4.12 ReadBrightPin()	25
4.17.4.13 ReadDimPin()	25
4.17.4.14 receiveData()	25
4.17.4.15 RestartTIM2()	25
4.17.4.16 sendUARTChar()	26
4.17.4.17 SetPW11()	26
4.17.4.18 StartPWM11()	26
4.17.4.19 StartTIM2()	26
4.17.4.20 StopPWM11()	26
4.17.4.21 transferData()	26
4.18 stm32l412xx-bsp.h	26
4.19 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/temperature_↔ handler.h File Reference	28
4.19.1 Enumeration Type Documentation	28
4.19.1.1 TemperatureRange_e	28
4.19.2 Function Documentation	29
4.19.2.1 GetTemperature()	29
4.19.2.2 GetTemperatureRange()	29
4.20 temperature_handler.h	29
4.21 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/voltage_handler.h File Reference	29
4.21.1 Enumeration Type Documentation	30
4.21.1.1 VoltageRange_e	30
4.21.2 Function Documentation	30
4.21.2.1 GetVoltage()	30
4.21.2.2 GetVoltageRange()	30
4.22 voltage_handler.h	30
4.23 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/button_handler.c File Reference	31
4.23.1 Function Documentation	31
4.23.1.1 IsBrightPressed()	31
4.23.1.2 IsDimPressed()	31
4.24 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/current_handler.c File Reference	31
4.24.1 Function Documentation	32
4.24.1.1 GetCurrent()	32
4.24.1.2 GetCurrentRange()	32

4.24.2 Variable Documentation	32
4.24.2.1 CurrentErrorThreshold_dA	32
4.24.2.2 CurrentHighThreshold_dA	32
4.24.2.3 dAmpsToRaw	32
4.24.2.4 RawTodAmps	32
4.25 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/delay_handler.c File Reference	32
4.25.1 Function Documentation	32
4.25.1.1 BrightnessDelay()	32
4.25.1.2 DelayHit()	32
4.25.1.3 RestartDelayCounter()	33
4.25.1.4 StartDelayCounter()	33
4.25.2 Variable Documentation	33
4.25.2.1 Tim2ClkKhz	33
4.26 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/fram.c File Reference	33
4.26.1 Detailed Description	33
4.26.2 Macro Definition Documentation	34
4.26.2.1 TADD	34
4.26.2.2 TLEN	34
4.26.3 Function Documentation	34
4.26.3.1 framChipSelect()	34
4.26.3.2 framReadMemory()	34
4.26.3.3 framReadSr()	34
4.26.3.4 framTest()	34
4.26.3.5 framWriteDisable()	35
4.26.3.6 framWriteEnable()	35
4.26.3.7 framWriteMemory()	35
4.26.3.8 framWriteProtect()	35
4.26.3.9 framWriteSr()	36
4.27 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/logger.c File Reference	37
4.27.1 Function Documentation	37
4.27.1.1 LogNumber()	37
4.27.1.2 LogString()	37
4.27.1.3 ReadLog()	37
4.28 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/main.c File Reference	37
4.28.1 Macro Definition Documentation	38
4.28.1.1 AVG_STEP_DIFF_MS	38
4.28.1.2 AVG_STEP_TIME_MS	38
4.28.1.3 LOWER_STEP_TIME_MS	38
4.28.1.4 LOWER_SWEEP_TIME_MS	38
4.28.1.5 TOTAL_SWEEP_TIME_MS	38
4.28.1.6 UPPER_STEP_TIME_MS	38

4.28.1.7 UPPER_SWEEP_TIME_MS	38
4.28.2 Function Documentation	38
4.28.2.1 main()	38
4.28.2.2 SystemClock_Config()	39
4.28.3 Variable Documentation	39
4.28.3.1 HighStepTimeMs	39
4.28.3.2 LowStepTimeMs	39
4.29 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/my_printf.c File Reference	39
4.30 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/pwm_handler.c File Reference	39
4.30.1 Macro Definition Documentation	40
4.30.1.1 PW_PERIOD	40
4.30.2 Function Documentation	40
4.30.2.1 DecreaseBrightness()	40
4.30.2.2 GetBrightness()	40
4.30.2.3 GetPwm()	40
4.30.2.4 IncreaseBrightness()	40
4.30.2.5 InitPwm()	40
4.30.2.6 SetBrightness()	40
4.30.2.7 SetPwm()	40
4.30.2.8 TurnOffPwm()	40
4.30.3 Variable Documentation	41
4.30.3.1 HalfBrightness	41
4.30.3.2 HotPwmRatio	41
4.30.3.3 MaxBrightness	41
4.30.3.4 MaxPw	41
4.30.3.5 MinBrightness	41
4.30.3.6 MinPw	41
4.30.3.7 WarmPwmRatio	41
4.31 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/stm32l412xx-bsp.c File Reference	41
4.31.1 Function Documentation	42
4.31.1.1 disableChipSelect()	42
4.31.1.2 DisablePWM1()	42
4.31.1.3 disableWriteProtect()	42
4.31.1.4 enableChipSelect()	42
4.31.1.5 EnablePWM1()	42
4.31.1.6 enableWriteProtect()	42
4.31.1.7 Error_Handler()	42
4.31.1.8 GetCurrentValue()	42
4.31.1.9 GetThermistorValue()	42
4.31.1.10 GetTIM2Cnt()	42

4.31.1.11	GetVoltageValue()	43
4.31.1.12	ReadBrightPin()	43
4.31.1.13	ReadDimPin()	43
4.31.1.14	receiveData()	43
4.31.1.15	RestartTIM2()	43
4.31.1.16	sendUARTChar()	43
4.31.1.17	SetPW11()	43
4.31.1.18	StartPWM11()	43
4.31.1.19	StartTIM2()	43
4.31.1.20	StopPWM11()	43
4.31.1.21	transferData()	43
4.32	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/temperature_↵ handler.c File Reference	44
4.32.1	Function Documentation	44
4.32.1.1	GetTemperature()	44
4.32.1.2	GetTemperatureRange()	44
4.32.2	Variable Documentation	44
4.32.2.1	CoolingThreshold1_dC	44
4.32.2.2	CoolingThreshold2_dC	44
4.32.2.3	dCelciusToThermistor	44
4.32.2.4	HeatingThreshold1_dC	44
4.32.2.5	HeatingThreshold2_dC	44
4.32.2.6	ThermistorTodCelcius	44
4.33	C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/voltage_handler.c File Reference	45
4.33.1	Function Documentation	45
4.33.1.1	GetVoltage()	45
4.33.1.2	GetVoltageRange()	45
4.33.2	Variable Documentation	45
4.33.2.1	dColtsToRaw	45
4.33.2.2	RawTodColts	45
4.33.2.3	VoltageErrorHighThreshold_dV	45
4.33.2.4	VoltageErrorLowThreshold_dV	45
4.33.2.5	VoltageHighThreshold_dV	45
4.33.2.6	VoltageLowThreshold_dV	45

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

PwmStruct	5
TimerStruct	5

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	9
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ fram.h This module contains definitions and structures to support SPI FRAM operations	11
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ logger.h	15
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ main.h	16
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ my_printf.h	18
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ pwm_handler.h	18
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ stm32l412xx-bsp.h	20
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ temperature_handler.h	28
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/ voltage_handler.h	29
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ button_handler.c	31
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ current_handler.c	31
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ delay_handler.c	32
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ fram.c This module contains routines to access the SPI FRAM. The FRAM used in the device is 32Kx8 with an operating frequency up to 20 MHz. It supports SPI Mode 0 & 3. The hardware supports the external Write Protect pin. The nature of this FRAM supports unlimited read/writes and data retention beyond the human life span	33
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ logger.c	37
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ main.c	37
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ my_printf.c	39
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ pwm_handler.c	39
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ stm32l412xx-bsp.c	41
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ temperature_handler.c	44
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/ voltage_handler.c	45

Chapter 3

Data Structure Documentation

3.1 PwmStruct Struct Reference

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

Data Fields

- uint8_t [is_running](#)
- uint32_t [pulse_width](#)

3.1.1 Field Documentation

3.1.1.1 is_running

```
uint8_t is_running
```

3.1.1.2 pulse_width

```
uint32_t pulse_width
```

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

- C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Inc/[stm32l412xx-bsp.h](#)

3.2 TimerStruct Struct Reference

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

Data Fields

- `uint8_t` [is_running](#)
- `uint32_t` [time](#)

3.2.1 Field Documentation

3.2.1.1 is_running

`uint8_t is_running`

3.2.1.2 time

`uint32_t time`

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

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

Chapter 4

File Documentation

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

```
#include <stdint.h>
#include "stm321412xx-bsp.h"
```

Functions

- [GPIO_PinState IsDimPressed](#) (void)
- [GPIO_PinState IsBrightPressed](#) (void)

4.1.1 Function Documentation

4.1.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
    void )
```

4.1.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
    void )
```

4.2 button_handler.h

[Go to the documentation of this file.](#)

```

00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: button_handler.h
00010 //
00011 // Description: Returns the button state of the three board buttons
00012 //
00013 // Revision History:
00014 // Date      - Name      - Ver - Remarks
00015 // 07/31/2024 - Austin Green - 1.0 - Initial Document
00016 //
00017 // Notes: Depends on the board support package bsp for GPIO_PinState
00018 //
00019 // *****
00020
00021 #ifndef INC_button_handlerh
00022     #define INC_button_handlerh
00023
00024     #include <stdint.h>
00025
00026     #include "stm32l412xx-bsp.h"
00027
00028     /* Return state of buttons */
00029     GPIO_PinState IsDimPressed ( void );
00030     GPIO_PinState IsBrightPressed ( void );
00031
00032 #endif /* INC_button_handlerh */

```

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

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

Enumerations

- enum [CurrentRange_e](#) { [CurrentNormal](#) = 0 , [CurrentHigh](#) = 1 , [CurrentError](#) = 2 }

Functions

- uint16_t [GetCurrent](#) (void)
- [CurrentRange_e](#) [GetCurrentRange](#) (void)

4.3.1 Enumeration Type Documentation

4.3.1.1 CurrentRange_e

```
enum CurrentRange\_e
```

Enumerator

CurrentNormal	
CurrentHigh	
CurrentError	

4.3.2 Function Documentation

4.3.2.1 GetCurrent()

```
uint16_t GetCurrent (
    void )
```

4.3.2.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
    void )
```

4.4 current_handler.h

[Go to the documentation of this file.](#)

```
00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: current_handler.h
00010 //
00011 // Description: Handles getting current and reporting values.
00012 //
00013 // Revision History:
00014 // Date - Name - Ver - Remarks
00015 // 08/05/2024 - Austin Green - 1.0 - Initial Document
00016 //
00017 // Notes:
00018 //
00019 // *****
00020
00021 #ifndef INC_current_handlerh
00022     #define INC_current_handlerh
00023
00024     #include <stdint.h>
00025
00026     /* Current Range Enum */
00027     typedef enum
00028     {
00029         CurrentNormal = 0,
00030         CurrentHigh = 1,
00031         CurrentError = 2
00032     } CurrentRange_e;
00033
00034     /* Get Current */
00035     uint16_t GetCurrent( void ); // in dA
00036     CurrentRange_e GetCurrentRange( void );
00037
00038 #endif /* INC_current_handlerh */
```

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

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

Functions

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

4.5.1 Function Documentation

4.5.1.1 BrightnessDelay()

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

4.5.1.2 DelayHit()

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

4.5.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
    void )
```

4.5.1.4 StartDelayCounter()

```
void StartDelayCounter (
    void )
```

4.6 delay_handler.h

[Go to the documentation of this file.](#)

```
00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: delay_handler.h
00010 //
00011 // Description: Handles system counters and delays
00012 //
00013 // Revision History:
00014 // Date           - Name           - Ver - Remarks
00015 // 07/31/2024 - Austin Green - 1.0 - Initial Document
00016 //
00017 // Notes:
00018 //
00019 // *****
00020
00021 /* Define to prevent recursive inclusion -----*/
00022 #ifndef INC_delay_handlerh
00023     #define INC_delay_handlerh
00024
00025     #include <stdint.h>
00026
00027     /* Check if delay has been hit */
00028     void StartDelayCounter(void); // start the counter
00029     void RestartDelayCounter(void); // restart the counter
00030     uint8_t DelayHit(uint32_t delay_ms); // is delay in ms hit
00031     /* Get ms delay for given brightness */
00032     uint16_t BrightnessDelay(int8_t brightness);
00033
00034 #endif /* INC_delay_handlerh */
```

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

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

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

Enumerations

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

Functions

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

4.7.1 Detailed Description

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

Attention

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

Revision History: Date Name Ver Remarks

04/09/2023 Mark Lane 0 Original Version

Notes:

4.7.2 Enumeration Type Documentation

4.7.2.1 CHIP_SELECT_STATE

```
enum CHIP\_SELECT\_STATE
```

chip select state

Enumerator

CSS_ASSERT	chip select disabled
CSS_RELEASE	chip select enabled

4.7.2.2 OPCODE_COMMANDS

enum [OPCODE_COMMANDS](#)

opcode command

Enumerator

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

4.7.2.3 STATUS_REGISTER

enum [STATUS_REGISTER](#)

status register

Enumerator

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

4.7.2.4 WRITE_PROTECT_STATE

enum [WRITE_PROTECT_STATE](#)

write protect state

Enumerator

WPS_PROTECTED	write protected
WPS_WRITEABLE	write enabled

4.7.3 Function Documentation

4.7.3.1 framChipSelect()

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

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

Parameters

in	<i>state</i>	assert = 0, release = 1
out	<i>none</i>	

4.7.3.2 framReadMemory()

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

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

Parameters

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

4.7.3.3 framReadSr()

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

This routine reads the FRAM status register

Parameters

in	<i>destination</i>	pointer for FRAM status register
out	<i>none</i>	

4.7.3.4 framTest()

```
uint8_t framTest (
    void )
```

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

Parameters

out	<i>pass</i>	= 1, fail = 0
-----	-------------	---------------

4.7.3.5 framWriteDisable()

```
void framWriteDisable (
    void )
```

This routine resets the write enable latch

Parameters

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

4.7.3.6 framWriteEnable()

```
void framWriteEnable (
    void )
```

This routine sets the write enable latch

Parameters

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

4.7.3.7 framWriteMemory()

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

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

Parameters

in	<i>addr</i>	FRAM memory address to write data
in	<i>wrBufP</i>	pointer to data to write
in	<i>len</i>	number of byte(s) to write
out	<i>none</i>	

4.7.3.8 framWriteProtect()

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

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

Parameters

in	<i>state</i>	disable = 0, enable = 1
out	<i>none</i>	

4.7.3.9 framWriteSr()

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

This routine writes the FRAM status register

Parameters

in	<i>data</i>	value written to FRAM status register
out	<i>none</i>	

4.8 fram.h

[Go to the documentation of this file.](#)

```

00001
00030 #ifndef _FRAM_H_
00031 #define _FRAM_H_
00032
00033 #include "stm32l412xx-bsp.h"
00034
00035 /* Definition */
00039 typedef enum {
00040
00041     OC_WREN = 6 ,
00042     OC_WRDI = 4 ,
00043     OC_RDSR = 5 ,
00044     OC_WRSR = 1 ,
00045     OC_READ = 3 ,
00046     OC_WRITE = 2 ,
00048 } OPCODE_COMMANDS ;
00049
00050
00054 typedef enum {
00055
00056     SR_WEL = 0x2 ,
00057     SR_BP0 = 0x4 ,
00058     SR_BP1 = 0x8 ,
00059     SR_WPEN = 0x80 ,
00061 } STATUS_REGISTER ;
00062
00063
00067 typedef enum {
00068
00069     WPS_PROTECTED = 0 ,
00070     WPS_WRITEABLE = 1 ,
00072 } WRITE_PROTECT_STATE ;
00073
00074
00078 typedef enum {
00079
00080     CSS_ASSERT = 0 ,
00081     CSS_RELEASE = 1 ,
00083 } CHIP_SELECT_STATE ;
00084
00085
00086 /* Prototype Definition */
00087
00094 void framWriteProtect( WRITE_PROTECT_STATE state ) ;
00095
00102 void framChipSelect( CHIP_SELECT_STATE state ) ;
00103
00104
00111 void framReadSr( unsigned char *srP ) ;
00112
00119 void framWriteSr( unsigned char sr ) ;
00120
00121
00126 void framWriteDisable( void ) ;
00127
00132 void framWriteEnable( void ) ;
00133
00134
00144 void framReadMemory ( unsigned short addr, unsigned char *rdBufP, unsigned short len ) ;
00145
00155 void framWriteMemory( unsigned short addr, const unsigned char* const wrBufP, unsigned short len ) ;
00156
00157
00164 uint8_t framTest( void ) ;
00165
00166
00167 #endif

```

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

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

Functions

- void [LogString](#) (const char *const string, uint8_t write_beginning)

- void [LogNumber](#) (const int32_t number, uint8_t write_beginning)
- void [ReadLog](#) (const uint32_t address, char *string, const uint32_t bytes)

4.9.1 Function Documentation

4.9.1.1 LogNumber()

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

4.9.1.2 LogString()

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

4.9.1.3 ReadLog()

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

4.10 logger.h

[Go to the documentation of this file.](#)

```
00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: logger.h
00010 //
00011 // Description: Handles logging and reading of data to memory
00012 //
00013 // Revision History:
00014 // Date - Name - Ver - Remarks
00015 // 07/31/2024 - Austin Green - 1.0 - Initial Document
00016 // 08/05/2024 - Austin Green - 1.1 - Added Log Number
00017 //
00018 // Notes:
00019 //
00020 // *****
00021
00022 #ifndef INC_loggerh
00023     #define INC_loggerh
00024
00025     #include <stdint.h>
00026
00027     /* Store and Read Logs */
00028     void LogString( const char* const string, uint8_t write_beginning );
00029     void LogNumber( const int32_t number, uint8_t write_beginning );
00030     void ReadLog( const uint32_t address, char* string, const uint32_t bytes );
00031
00032 #endif /* INC_loggerh */
```

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

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

Functions

- void [Error_Handler](#) (void)

This function is executed in case of error occurrence.

4.11.1 Function Documentation

4.11.1.1 Error_Handler()

```
void Error_Handler (
    void )
```

This function is executed in case of error occurrence.

Return values

None	
------	--

4.12 main.h

[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00002 /****
00003      *****
00004      * @file           : main.h
00005      * @brief          : Header for main.c file.
00006      *                  : This file contains the common defines of the application.
00007      *****
00008      * @attention
00009      *
00010      * Copyright (c) 2024 STMicroelectronics.
00011      * All rights reserved.
00012      *
00013      * This software is licensed under terms that can be found in the LICENSE file
00014      * in the root directory of this software component.
00015      * If no LICENSE file comes with this software, it is provided AS-IS.
00016      *
00017      *****
00018      */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -----*/
00022 #ifndef INC_mainh
00023     #define INC_mainh
00024
00025     #ifdef __cplusplus
00026     extern "C" {
00027     #endif
00028
00029     /* Includes -----*/
00030
00031     /* Private includes -----*/
00032     /* USER CODE BEGIN Includes */
00033     #include "stm32l412xx-bsp.h"
00034
00035     /* USER CODE END Includes */
00036
00037     /* Exported types -----*/
00038     /* USER CODE BEGIN ET */
00039
00040     /* USER CODE END ET */
00041
00042     /* Exported constants -----*/
00043     /* USER CODE BEGIN EC */
00044
00045     /* USER CODE END EC */
00046
00047     /* Exported macro -----*/
00048     /* USER CODE BEGIN EM */
00049
00050     /* USER CODE END EM */
00051
00052     /* Exported functions prototypes -----*/
00053     void Error\_Handler(void);
00054
00055     /* USER CODE BEGIN EFP */
00056
```

```

00057      /* USER CODE END EFP */
00058
00059      /* Private defines -----*/
00060
00061      /* USER CODE BEGIN Private defines */
00062
00063      /* USER CODE END Private defines */
00064
00065      #ifdef __cplusplus
00066      }
00067      #endif
00068
00069 #endif /* INC_mainh */

```

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

4.14 my_printf.h

[Go to the documentation of this file.](#)

```

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

```

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

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

Macros

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

Functions

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

- void [SetBrightness](#) (int8_t brightness)
- uint8_t [GetPwm](#) (void)

4.15.1 Macro Definition Documentation

4.15.1.1 BRIGHTNESS_STEPS

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

4.15.1.2 HOLD_BRIGHTNESS_JUMP

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

4.15.2 Function Documentation

4.15.2.1 DecreaseBrightness()

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

4.15.2.2 GetBrightness()

```
int8_t GetBrightness (  
    void )
```

4.15.2.3 GetPwm()

```
uint8_t GetPwm (  
    void )
```

4.15.2.4 IncreaseBrightness()

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

4.15.2.5 InitPwm()

```
void InitPwm (  
    void )
```

4.15.2.6 SetBrightness()

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

4.15.2.7 SetPwm()

```
void SetPwm (  
    void )
```

4.15.2.8 TurnOffPwm()

```
void TurnOffPwm (  
    void )
```

4.16 pwm_handler.h

[Go to the documentation of this file.](#)

```

00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: pwm_handler.h
00010 //
00011 // Description: Handles the PWM output of the lights. Output is
00012 //              determined by a Brightness variable that is
00013 //              controlled by this file.
00014 //
00015 // Revision History:
00016 // Date      - Name      - Ver - Remarks
00017 // 07/31/2024 - Austin Green - 1.0 - Initial Document
00018 //
00019 // Notes:
00020 //
00021 // *****
00022
00023 #ifndef INC_pwm_handlerh
00024     #define INC_pwm_handlerh
00025
00026     #include <stdint.h>
00027
00028     /* Brightness Steps */
00029     #define BRIGHTNESS_STEPS      (50)
00030     #define HOLD_BRIGHTNESS_JUMP  (3)
00031
00032     /* Initialize system */
00033     void InitPwm(void); // Init Pwm var
00034     /* Decrease/Increase brightness levels */
00035     void DecreaseBrightness( uint8_t button_held ); // decrease brightness
00036     void IncreaseBrightness( uint8_t button_held ); // increase brightness
00037     /* Set PWM value based on internal brightness */
00038     void SetPwm( void ); // turn on and set PWM
00039     void TurnOffPwm( void ); // turn of PWM
00040     /* Get or set internal values */
00041     int8_t GetBrightness( void ); // get value of Brightness
00042     void SetBrightness( int8_t brightness ); // set value of Brightness
00043     uint8_t GetPwm( void ); // get value of current PWM
00044
00045 #endif /* INC_pwm_handlerh */

```

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

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

Data Structures

- struct [PwmStruct](#)
- struct [TimerStruct](#)

Macros

- #define [THERMISTOR_ADC_Pin](#) 0
- #define [THERMISTOR_ADC_GPIO_Port](#) 0
- #define [BRIGHT_Pin](#) 0
- #define [BRIGHT_GPIO_Port](#) 0
- #define [DIM_Pin](#) 0
- #define [DIM_GPIO_Port](#) 0
- #define [EEPROM_SCK_Pin](#) 0
- #define [EEPROM_SCK_GPIO_Port](#) 0
- #define [EEPROM_MISO_Pin](#) 0
- #define [EEPROM_MISO_GPIO_Port](#) 0

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

Typedefs

- typedef uint8_t [GPIO_PinState](#)

Enumerations

- enum { [BUTTON_PRESSED](#) = 1 , [BUTTON_UNPRESSED](#) = 0 }
- enum { [PIN_SET](#) = 1 , [PIN_RESET](#) = 0 }

Functions

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

This function is executed in case of error occurrence.

4.17.1 Macro Definition Documentation

4.17.1.1 AMPMETER_ADC_GPIO_Port

```
#define AMPMETER_ADC_GPIO_Port 0
```

4.17.1.2 AMPMETER_ADC_Pin

```
#define AMPMETER_ADC_Pin 0
```

4.17.1.3 BRIGHT_GPIO_Port

```
#define BRIGHT_GPIO_Port 0
```

4.17.1.4 BRIGHT_Pin

```
#define BRIGHT_Pin 0
```

4.17.1.5 CLK_FREQ_HZ

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

4.17.1.6 DIM_GPIO_Port

```
#define DIM_GPIO_Port 0
```

4.17.1.7 DIM_Pin

```
#define DIM_Pin 0
```

4.17.1.8 EEPROM_MISO_GPIO_Port

```
#define EEPROM_MISO_GPIO_Port 0
```

4.17.1.9 EEPROM_MISO_Pin

```
#define EEPROM_MISO_Pin 0
```

4.17.1.10 EEPROM_MOSI_GPIO_Port

```
#define EEPROM_MOSI_GPIO_Port 0
```

4.17.1.11 EEPROM_MOSI_Pin

```
#define EEPROM_MOSI_Pin 0
```

4.17.1.12 EEPROM_SCK_GPIO_Port

```
#define EEPROM_SCK_GPIO_Port 0
```

4.17.1.13 EEPROM_SCK_Pin

```
#define EEPROM_SCK_Pin 0
```

4.17.1.14 NVIC_PRIORITYGROUP_0

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

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

4.17.1.15 NVIC_PRIORITYGROUP_1

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

4.17.1.16 NVIC_PRIORITYGROUP_2

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

4.17.1.17 NVIC_PRIORITYGROUP_3

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

4.17.1.18 NVIC_PRIORITYGROUP_4

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

4.17.1.19 PWM_OUT_GPIO_Port

```
#define PWM_OUT_GPIO_Port 0
```

4.17.1.20 PWM_OUT_Pin

```
#define PWM_OUT_Pin 0
```

4.17.1.21 SPI_NSS_GPIO_Port

```
#define SPI_NSS_GPIO_Port 0
```

4.17.1.22 SPI_NSS_Pin

```
#define SPI_NSS_Pin 0
```

4.17.1.23 SPI_WP_GPIO_Port

```
#define SPI_WP_GPIO_Port 0
```

4.17.1.24 SPI_WP_Pin

```
#define SPI_WP_Pin 0
```

4.17.1.25 THERMISTOR_ADC_GPIO_Port

```
#define THERMISTOR_ADC_GPIO_Port 0
```

4.17.1.26 THERMISTOR_ADC_Pin

```
#define THERMISTOR_ADC_Pin 0
```

4.17.1.27 TIM2_CLK_DEV

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

4.17.1.28 TIM2_CLK_PRESCALER

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

4.17.1.29 VOLTMETER_ADC_GPIO_Port

```
#define VOLTMETER_ADC_GPIO_Port 0
```

4.17.1.30 VOLTMETER_ADC_Pin

```
#define VOLTMETER_ADC_Pin 0
```

4.17.2 Typedef Documentation

4.17.2.1 GPIO_PinState

```
typedef uint8_t GPIO_PinState
```

4.17.3 Enumeration Type Documentation

4.17.3.1 anonymous enum

anonymous enum

Enumerator

BUTTON_PRESSED	
BUTTON_UNPRESSED	

4.17.3.2 anonymous enum

anonymous enum

Enumerator

PIN_SET	
PIN_RESET	

4.17.4 Function Documentation

4.17.4.1 disableChipSelect()

```
void disableChipSelect (  
    void )
```

4.17.4.2 DisablePWM1()

```
void DisablePWM1 (  
    void )
```

4.17.4.3 disableWriteProtect()

```
void disableWriteProtect (  
    void )
```

4.17.4.4 enableChipSelect()

```
void enableChipSelect (  
    void )
```

4.17.4.5 EnablePWM1()

```
void EnablePWM1 (
    void )
```

4.17.4.6 enableWriteProtect()

```
void enableWriteProtect (
    void )
```

4.17.4.7 Error_Handler()

```
void Error_Handler (
    void )
```

This function is executed in case of error occurrence.

Return values

None	
------	--

4.17.4.8 GetCurrentValue()

```
int16_t GetCurrentValue (
    void )
```

4.17.4.9 GetThermistorValue()

```
int16_t GetThermistorValue (
    void )
```

4.17.4.10 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
    void )
```

4.17.4.11 GetVoltageValue()

```
int16_t GetVoltageValue (
    void )
```

4.17.4.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
    void )
```

4.17.4.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (
    void )
```

4.17.4.14 receiveData()

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

4.17.4.15 RestartTIM2()

```
void RestartTIM2 (
    void )
```

4.17.4.16 sendUARTChar()

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

4.17.4.17 SetPW11()

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

4.17.4.18 StartPWM11()

```
void StartPWM11 (
    void )
```

4.17.4.19 StartTIM2()

```
void StartTIM2 (
    void )
```

4.17.4.20 StopPWM11()

```
void StopPWM11 (
    void )
```

4.17.4.21 transferData()

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

4.18 stm32l412xx-bsp.h

[Go to the documentation of this file.](#)

```
00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: stm32l412xx-bsp.h
00010 //
00011 // Description: Board Support Package for STM32L412xx
00012 //
00013 // Revision History:
00014 // Date      - Name      - Ver - Remarks
00015 // 07/31/2024 - Austin Green - 1.0 - Initial Document
00016 //
00017 // Notes: This uses the Low Level ST API to access the board pins
00018 //
00019 // *****
00020
00021 #ifndef INC_bsph
00022     #define INC_bsph
00023
00024     #include <stdint.h>
00025
00026     /* Private defines -----*/
00027     #ifdef STM32L412xx
00028
00029         #include "stm32l4xx_ll_adc.h"
00030         #include "stm32l4xx_ll_crs.h"
00031         #include "stm32l4xx_ll_rcc.h"
00032         #include "stm32l4xx_ll_bus.h"
00033         #include "stm32l4xx_ll_system.h"
00034         #include "stm32l4xx_ll_exti.h"
00035         #include "stm32l4xx_ll_cortex.h"
00036         #include "stm32l4xx_ll_utils.h"
00037         #include "stm32l4xx_ll_pwr.h"
```

```

00038     #include "stm32l4xx_ll_dma.h"
00039     #include "stm32l4xx_ll_spi.h"
00040     #include "stm32l4xx_ll_tim.h"
00041     #include "stm32l4xx_ll_usart.h"
00042     #include "stm32l4xx_ll_gpio.h"
00043
00044     #if defined(USE_FULL_ASSERT)
00045         #include "stm32_assert.h"
00046     #endif /* USE_FULL_ASSERT */
00047
00048     /* Peripherals */
00049     #include "adc.h"
00050     #include "gpio.h"
00051     #include "spi.h"
00052     #include "tim.h"
00053
00054     #ifdef ENABLE_UART_DEBUGGING /* tracing enabled */
00055         /* Peripherals enabled for UART */
00056         #include "usart.h"
00057     #endif /* ENABLE_UART_DEBUGGING */
00058
00059     #define THERMISTOR_ADC_Pin LL_GPIO_PIN_0
00060     #define THERMISTOR_ADC_GPIO_Port GPIOA
00061     #define BRIGHT_Pin LL_GPIO_PIN_1
00062     #define BRIGHT_GPIO_Port GPIOA
00063     #define DIM_Pin LL_GPIO_PIN_3
00064     #define DIM_GPIO_Port GPIOA
00065     #define EEPROM_SCK_Pin LL_GPIO_PIN_5
00066     #define EEPROM_SCK_GPIO_Port GPIOA
00067     #define EEPROM_MISO_Pin LL_GPIO_PIN_6
00068     #define EEPROM_MISO_GPIO_Port GPIOA
00069     #define EEPROM_MOSI_Pin LL_GPIO_PIN_7
00070     #define EEPROM_MOSI_GPIO_Port GPIOA
00071     #define VOLTMETER_ADC_Pin LL_GPIO_PIN_0
00072     #define VOLTMETER_ADC_GPIO_Port GPIOB
00073     #define AMPMETER_ADC_Pin LL_GPIO_PIN_1
00074     #define AMPMETER_ADC_GPIO_Port GPIOB
00075     #define PWM_OUT_Pin LL_GPIO_PIN_8
00076     #define PWM_OUT_GPIO_Port GPIOA
00077     #define SPI_WP_Pin LL_GPIO_PIN_10
00078     #define SPI_WP_GPIO_Port GPIOA
00079     #define SPI_NSS_Pin LL_GPIO_PIN_11
00080     #define SPI_NSS_GPIO_Port GPIOA
00081
00082     #else /* STM32L412xx */
00083
00084         /* Below is for debugging purposes */
00085         #define THERMISTOR_ADC_Pin 0
00086         #define THERMISTOR_ADC_GPIO_Port 0
00087         #define BRIGHT_Pin 0
00088         #define BRIGHT_GPIO_Port 0
00089         #define DIM_Pin 0
00090         #define DIM_GPIO_Port 0
00091         #define EEPROM_SCK_Pin 0
00092         #define EEPROM_SCK_GPIO_Port 0
00093         #define EEPROM_MISO_Pin 0
00094         #define EEPROM_MISO_GPIO_Port 0
00095         #define EEPROM_MOSI_Pin 0
00096         #define EEPROM_MOSI_GPIO_Port 0
00097         #define VOLTMETER_ADC_Pin 0
00098         #define VOLTMETER_ADC_GPIO_Port 0
00099         #define AMPMETER_ADC_Pin 0
00100         #define AMPMETER_ADC_GPIO_Port 0
00101         #define PWM_OUT_Pin 0
00102         #define PWM_OUT_GPIO_Port 0
00103         #define SPI_WP_Pin 0
00104         #define SPI_WP_GPIO_Port 0
00105         #define SPI_NSS_Pin 0
00106         #define SPI_NSS_GPIO_Port 0
00107
00108         typedef struct
00109         {
00110             uint8_t is_running;
00111             uint32_t pulse_width;
00112         } PwmStruct;
00113
00114         typedef struct
00115         {
00116             uint8_t is_running;
00117             uint32_t time;
00118         } TimerStruct;
00119
00120     #endif /* STM32L412xx */
00121
00122     /* Interrupt Handlers */
00123     #ifndef NVIC_PRIORITYGROUP_0
00124         #define NVIC_PRIORITYGROUP_0 ((uint32_t)0x00000007)

```

```

00126         #define NVIC_PRIORITYGROUP_1          ((uint32_t)0x00000006)
00128         #define NVIC_PRIORITYGROUP_2          ((uint32_t)0x00000005)
00130         #define NVIC_PRIORITYGROUP_3          ((uint32_t)0x00000004)
00132         #define NVIC_PRIORITYGROUP_4          ((uint32_t)0x00000003)
00134     #endif
00135
00136     /* Button Defines */
00137     typedef uint8_t GPIO_PinState;
00138     enum { BUTTON_PRESSED = 1, BUTTON_UNPRESSED = 0 };
00139     enum { PIN_SET = 1, PIN_RESET = 0 };
00140
00141     /* Clock frequency Values */
00142     #define CLK_FREQ_HZ      (8000000)
00143     #define TIM2_CLK_DEV      (1)
00144     #define TIM2_CLK_PRESCALER      (8000)
00145
00146     /* Returns button state */
00147     GPIO_PinState ReadDimPin( void );
00148     GPIO_PinState ReadBrightPin( void );
00149     /* PWM Outputs */
00150     void EnablePWM1( void );
00151     void DisablePWM1( void );
00152     void StartPWM1( void );
00153     void StopPWM1( void );
00154     void SetPW1( uint32_t pulse_width );
00155     /* Timers */
00156     void StartTIM2( void );
00157     void RestartTIM2( void );
00158     uint32_t GetTIM2Cnt( void );
00159     /* Thermistor */
00160     int16_t GetThermistorValue( void );
00161     /* Current Sensing */
00162     int16_t GetCurrentValue( void );
00163     /* Voltage Sensing */
00164     int16_t GetVoltageValue( void );
00165     /* Logging */
00166     void enableWriteProtect( void );
00167     void disableWriteProtect( void );
00168     void enableChipSelect( void );
00169     void disableChipSelect( void );
00170     void transferData( const unsigned char* const txData, const uint32_t bytes );
00171     void receiveData( unsigned char* rxData, const uint32_t bytes );
00172     /* UART Output */
00173     void sendUARTChar(char c);
00174     /* Errors */
00175     void Error_Handler(void);
00176
00177
00178 #endif /* INC_bsph */

```

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

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

Enumerations

- enum [TemperatureRange_e](#) { [TempCool](#) = 0 , [TempWarm](#) = 1 , [TempHot](#) = 2 }

Functions

- int32_t [GetTemperature](#) (void)
- [TemperatureRange_e](#) [GetTemperatureRange](#) (void)

4.19.1 Enumeration Type Documentation

4.19.1.1 TemperatureRange_e

```
enum TemperatureRange\_e
```

Enumerator

TempCool	
--------------------------	--

Enumerator

TempWarm	
TempHot	

4.19.2 Function Documentation

4.19.2.1 GetTemperature()

```
int32_t GetTemperature (
    void )
```

4.19.2.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
    void )
```

4.20 temperature_handler.h

[Go to the documentation of this file.](#)

```
00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: temperature_handler.h
00010 //
00011 // Description: Handles getting this temperature and transitioning
00012 //              between temperature states.
00013 //
00014 // Revision History:
00015 // Date      - Name      - Ver - Remarks
00016 // 07/31/2024 - Austin Green - 1.0 - Initial Document
00017 // 08/05/2024 - Austin Green - 1.1 - Refactor to not use floats
00018 //
00019 // Notes:
00020 //
00021 // *****
00022
00023 #ifndef INC_temperature_handlerh
00024     #define INC_temperature_handlerh
00025
00026     #include <stdint.h>
00027
00028     /* Temperature Range Enum */
00029     typedef enum
00030     {
00031         TempCool    = 0,
00032         TempWarm    = 1,
00033         TempHot     = 2
00034     } TemperatureRange_e;
00035
00036     /* Get Temperature */
00037     int32_t GetTemperature( void ); // in dC
00038     TemperatureRange_e GetTemperatureRange( void );
00039
00040 #endif /* INC_temperature_handlerh */
```

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

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

Enumerations

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

Functions

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

4.21.1 Enumeration Type Documentation

4.21.1.1 VoltageRange_e

enum `VoltageRange_e`

Enumerator

<code>VoltageNormal</code>	
<code>VoltageLow</code>	
<code>VoltageHigh</code>	
<code>VoltageErrorLow</code>	
<code>VoltageErrorHigh</code>	

4.21.2 Function Documentation

4.21.2.1 GetVoltage()

```
uint16_t GetVoltage (
    void )
```

4.21.2.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
    void )
```

4.22 voltage_handler.h

[Go to the documentation of this file.](#)

```
00001 // *****
00002 // Copyright © 2007 Luminator Mark IV
00003 // All rights reserved.
00004 // Any use without the prior written consent of Luminator Mark IV
00005 // is strictly prohibited.
00006 // *****
00007 // *****
00008 //
00009 // Filename: voltage_handler.h
00010 //
00011 // Description: Handles getting voltage and reporting values.
00012 //
00013 // Revision History:
00014 // Date - Name - Ver - Remarks
00015 // 08/05/2024 - Austin Green - 1.0 - Initial Document
00016 //
00017 // Notes:
00018 //
00019 // *****
00020
00021 #ifndef INC_voltage_handlerh
00022     #define INC_voltage_handlerh
00023
00024     #include <stdint.h>
00025
00026     /* Voltage Range Enum */
```



```

00027     typedef enum
00028     {
00029         VoltageNormal      = 0,
00030         VoltageLow         = 1,
00031         VoltageHigh        = 2,
00032         VoltageErrorLow    = 3,
00033         VoltageErrorHigh   = 4,
00034     } VoltageRange_e;
00035
00036     /* Get Voltage */
00037     uint16_t GetVoltage( void ); // in dV
00038     VoltageRange_e GetVoltageRange( void );
00039
00040 #endif /* INC_voltage_handlerh */

```

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

```

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

```

Functions

- [GPIO_PinState IsDimPressed](#) (void)
- [GPIO_PinState IsBrightPressed](#) (void)

4.23.1 Function Documentation

4.23.1.1 IsBrightPressed()

```

GPIO_PinState IsBrightPressed (
    void )

```

4.23.1.2 IsDimPressed()

```

GPIO_PinState IsDimPressed (
    void )

```

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

```

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

```

Functions

- [uint16_t GetCurrent](#) (void)
- [CurrentRange_e GetCurrentRange](#) (void)

Variables

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

4.24.1 Function Documentation

4.24.1.1 GetCurrent()

```
uint16_t GetCurrent (
    void )
```

4.24.1.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
    void )
```

4.24.2 Variable Documentation

4.24.2.1 CurrentErrorThreshold_dA

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

4.24.2.2 CurrentHighThreshold_dA

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

4.24.2.3 dAmpsToRaw

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

4.24.2.4 RawTodAmps

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

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

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

Functions

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

Variables

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

4.25.1 Function Documentation

4.25.1.1 BrightnessDelay()

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

4.25.1.2 DelayHit()

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

4.25.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
    void )
```

4.25.1.4 StartDelayCounter()

```
void StartDelayCounter (
    void )
```

4.25.2 Variable Documentation

4.25.2.1 Tim2ClkKhz

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

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

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

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

Macros

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

Functions

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

4.26.1 Detailed Description

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

Attention

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

Revision History: Date Name Ver Remarks

04/09/2023 Mark Lane 0 Original Version 06/10/2024 Austin Green 1 Add Logging 08/12/2024 Austin Green 1.2

Remove Mirror References

Notes:

4.26.2 Macro Definition Documentation

4.26.2.1 TADD

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

4.26.2.2 TLEN

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

4.26.3 Function Documentation

4.26.3.1 framChipSelect()

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

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

Parameters

in	<i>state</i>	assert = 0, release = 1
out	<i>none</i>	

4.26.3.2 framReadMemory()

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

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

Parameters

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

4.26.3.3 framReadSr()

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

This routine reads the FRAM status register

Parameters

in	<i>destination</i>	pointer for FRAM status register
out	<i>none</i>	

4.26.3.4 framTest()

```
uint8_t framTest (
    void )
```

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

Parameters

out	<i>pass</i>	= 1, fail = 0
-----	-------------	---------------

4.26.3.5 framWriteDisable()

```
void framWriteDisable (
    void )
```

This routine resets the write enable latch

Parameters

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

4.26.3.6 framWriteEnable()

```
void framWriteEnable (
    void )
```

This routine sets the write enable latch

Parameters

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

4.26.3.7 framWriteMemory()

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

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

Parameters

in	<i>addr</i>	FRAM memory address to write data
in	<i>wrBufP</i>	pointer to data to write
in	<i>len</i>	number of byte(s) to write
out	<i>none</i>	

4.26.3.8 framWriteProtect()

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

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

Parameters

in	<i>state</i>	disable = 0, enable = 1
out	<i>none</i>	

4.26.3.9 framWriteSr()

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

This routine writes the FRAM status register

Parameters

in	<i>data</i>	value written to FRAM status register
out	<i>none</i>	

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

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

Functions

- void [LogString](#) (const char *const string, uint8_t write_beginning)
- void [LogNumber](#) (const int32_t number, uint8_t write_beginning)
- void [ReadLog](#) (const uint32_t address, char *string, const uint32_t bytes)

4.27.1 Function Documentation

4.27.1.1 LogNumber()

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

4.27.1.2 LogString()

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

4.27.1.3 ReadLog()

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

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

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

Macros

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

Functions

- void `SystemClock_Config` (void)
System Clock Configuration.
- int `main` (void)
The application entry point. Initialize variables and go into bare metal loop. Polls buttons and sensors.

Variables

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

4.28.1 Macro Definition Documentation

4.28.1.1 AVG_STEP_DIFF_MS

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

4.28.1.2 AVG_STEP_TIME_MS

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

4.28.1.3 LOWER_STEP_TIME_MS

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

4.28.1.4 LOWER_SWEEP_TIME_MS

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

4.28.1.5 TOTAL_SWEEP_TIME_MS

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

4.28.1.6 UPPER_STEP_TIME_MS

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

4.28.1.7 UPPER_SWEEP_TIME_MS

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

4.28.2 Function Documentation

4.28.2.1 main()

```
int main (
    void )
```

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

Return values

<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 "stm321412xx-bsp.h"  
#include "temperature_handler.h"  
#include "my_printf.h"
```

Macros

- #define `PW_PERIOD` (255)

Functions

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

Variables

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

4.30.1 Macro Definition Documentation

4.30.1.1 PW_PERIOD

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

4.30.2 Function Documentation

4.30.2.1 DecreaseBrightness()

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

4.30.2.2 GetBrightness()

```
int8_t GetBrightness (  
    void )
```

4.30.2.3 GetPwm()

```
uint8_t GetPwm (  
    void )
```

4.30.2.4 IncreaseBrightness()

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

4.30.2.5 InitPwm()

```
void InitPwm (  
    void )
```

4.30.2.6 SetBrightness()

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

4.30.2.7 SetPwm()

```
void SetPwm (  
    void )
```

4.30.2.8 TurnOffPwm()

```
void TurnOffPwm (  
    void )
```

4.30.3 Variable Documentation

4.30.3.1 HalfBrightness

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

4.30.3.2 HotPwmRatio

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

4.30.3.3 MaxBrightness

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

4.30.3.4 MaxPw

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

4.30.3.5 MinBrightness

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

4.30.3.6 MinPw

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

4.30.3.7 WarmPwmRatio

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

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

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

Functions

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

- void `Error_Handler` (void)

This function is executed in case of error occurrence.

4.31.1 Function Documentation

4.31.1.1 `disableChipSelect()`

```
void disableChipSelect (
    void )
```

4.31.1.2 `DisablePWM1()`

```
void DisablePWM1 (
    void )
```

4.31.1.3 `disableWriteProtect()`

```
void disableWriteProtect (
    void )
```

4.31.1.4 `enableChipSelect()`

```
void enableChipSelect (
    void )
```

4.31.1.5 `EnablePWM1()`

```
void EnablePWM1 (
    void )
```

4.31.1.6 `enableWriteProtect()`

```
void enableWriteProtect (
    void )
```

4.31.1.7 `Error_Handler()`

```
void Error_Handler (
    void )
```

This function is executed in case of error occurrence.

Return values

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

4.31.1.8 `GetCurrentValue()`

```
int16_t GetCurrentValue (
    void )
```

4.31.1.9 `GetThermistorValue()`

```
int16_t GetThermistorValue (
    void )
```

4.31.1.10 `GetTIM2Cnt()`

```
uint32_t GetTIM2Cnt (
    void )
```

4.31.1.11 GetVoltageValue()

```
int16_t GetVoltageValue (  
    void )
```

4.31.1.12 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (  
    void )
```

4.31.1.13 ReadDimPin()

```
GPIO_PinState ReadDimPin (  
    void )
```

4.31.1.14 receiveData()

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

4.31.1.15 RestartTIM2()

```
void RestartTIM2 (  
    void )
```

4.31.1.16 sendUARTChar()

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

4.31.1.17 SetPW11()

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

4.31.1.18 StartPWM11()

```
void StartPWM11 (  
    void )
```

4.31.1.19 StartTIM2()

```
void StartTIM2 (  
    void )
```

4.31.1.20 StopPWM11()

```
void StopPWM11 (  
    void )
```

4.31.1.21 transferData()

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

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

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

Functions

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

Variables

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

4.32.1 Function Documentation

4.32.1.1 GetTemperature()

```
int32_t GetTemperature (
    void )
```

4.32.1.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
    void )
```

4.32.2 Variable Documentation

4.32.2.1 CoolingThreshold1_dC

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

4.32.2.2 CoolingThreshold2_dC

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

4.32.2.3 dCelciusToThermistor

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

4.32.2.4 HeatingThreshold1_dC

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

4.32.2.5 HeatingThreshold2_dC

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

4.32.2.6 ThermistorTodCelcius

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

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

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

Functions

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

Variables

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

4.33.1 Function Documentation

4.33.1.1 `GetVoltage()`

```
uint16_t GetVoltage (
    void )
```

4.33.1.2 `GetVoltageRange()`

```
VoltageRange_e GetVoltageRange (
    void )
```

4.33.2 Variable Documentation

4.33.2.1 `dColtsToRaw`

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

4.33.2.2 `RawTodColts`

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

4.33.2.3 `VoltageErrorHighThreshold_dV`

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

4.33.2.4 `VoltageErrorLowThreshold_dV`

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

4.33.2.5 `VoltageHighThreshold_dV`

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

4.33.2.6 `VoltageLowThreshold_dV`

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


Index

AMPMETER_ADC_GPIO_Port
 stm32l412xx-bsp.h, [22](#)
AMPMETER_ADC_Pin
 stm32l412xx-bsp.h, [22](#)
AVG_STEP_DIFF_MS
 main.c, [38](#)
AVG_STEP_TIME_MS
 main.c, [38](#)

BRIGHT_GPIO_Port
 stm32l412xx-bsp.h, [22](#)
BRIGHT_Pin
 stm32l412xx-bsp.h, [22](#)
BRIGHTNESS_STEPS
 pwm_handler.h, [19](#)
BrightnessDelay
 delay_handler.c, [32](#)
 delay_handler.h, [10](#)
button_handler.c
 IsBrightPressed, [31](#)
 IsDimPressed, [31](#)
button_handler.h
 IsBrightPressed, [7](#)
 IsDimPressed, [7](#)
BUTTON_PRESSED
 stm32l412xx-bsp.h, [24](#)
BUTTON_UNPRESSED
 stm32l412xx-bsp.h, [24](#)

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](#), [9](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [9](#), [10](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [11](#), [15](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [15](#), [16](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [16](#), [17](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [18](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [18](#), [20](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 bsp.h, [20](#), [26](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [28](#), [29](#)

C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [29](#), [30](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [31](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [31](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [32](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [33](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [37](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [37](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [39](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [39](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 bsp.c, [41](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [44](#)
C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core
 [45](#)
CHIP_SELECT_STATE
 fram.h, [11](#)
CLK_FREQ_HZ
 stm32l412xx-bsp.h, [22](#)
CoolingThreshold1_dC
 Controller/Core/Inc/button_handler.h,
 temperature_handler.c, [44](#)
CoolingThreshold2_dC
 Controller/Core/Inc/current_handler.h,
 temperature_handler.c, [44](#)
CSS_ASSERT
 Controller/Core/Inc/delay_handler.h,
 fram.h, [12](#)
CSS_RELEASE
 Controller/Core/Inc/fram.h,
 fram.h, [12](#)
current_handler.c
 Controller/Core/Inc/logger.h,
 CurrentErrorThreshold_dA, [32](#)
 CurrentHighThreshold_dA, [32](#)
 dAmpsToRaw, [32](#)
 GetCurrent, [32](#)
 GetCurrentRange, [32](#)
 RawTodAmps, [32](#)
 Controller/Core/Inc/pwm_handler.h,
 current_handler.h
 CurrentError, [8](#)
 CurrentHigh, [8](#)
 CurrentNormal, [8](#)
 CurrentRange_e, [8](#)
 GetCurrent, [9](#)

- GetCurrentRange, [9](#)
- CurrentError
 - current_handler.h, [8](#)
- CurrentErrorThreshold_dA
 - current_handler.c, [32](#)
- CurrentHigh
 - current_handler.h, [8](#)
- CurrentHighThreshold_dA
 - current_handler.c, [32](#)
- CurrentNormal
 - current_handler.h, [8](#)
- CurrentRange_e
 - current_handler.h, [8](#)
- dAmpsToRaw
 - current_handler.c, [32](#)
- dCelciusToThermistor
 - temperature_handler.c, [44](#)
- dColtsToRaw
 - voltage_handler.c, [45](#)
- DecreaseBrightness
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- delay_handler.c
 - BrightnessDelay, [32](#)
 - DelayHit, [32](#)
 - RestartDelayCounter, [32](#)
 - StartDelayCounter, [33](#)
 - Tim2ClkKhz, [33](#)
- delay_handler.h
 - BrightnessDelay, [10](#)
 - DelayHit, [10](#)
 - RestartDelayCounter, [10](#)
 - StartDelayCounter, [10](#)
- DelayHit
 - delay_handler.c, [32](#)
 - delay_handler.h, [10](#)
- DIM_GPIO_Port
 - stm32l412xx-bsp.h, [22](#)
- DIM_Pin
 - stm32l412xx-bsp.h, [22](#)
- disableChipSelect
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [24](#)
- DisablePWM1
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [24](#)
- disableWriteProtect
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [24](#)
- EEPROM_MISO_GPIO_Port
 - stm32l412xx-bsp.h, [22](#)
- EEPROM_MISO_Pin
 - stm32l412xx-bsp.h, [22](#)
- EEPROM_MOSI_GPIO_Port
 - stm32l412xx-bsp.h, [22](#)
- EEPROM_MOSI_Pin
 - stm32l412xx-bsp.h, [22](#)
- EEPROM_SCK_GPIO_Port
 - stm32l412xx-bsp.h, [22](#)
- EEPROM_SCK_Pin
 - stm32l412xx-bsp.h, [22](#)
- enableChipSelect
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [24](#)
- EnablePWM1
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [24](#)
- enableWriteProtect
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [25](#)
- Error_Handler
 - main.h, [17](#)
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [25](#)
- fram.c
 - framChipSelect, [34](#)
 - framReadMemory, [34](#)
 - framReadSr, [34](#)
 - framTest, [34](#)
 - framWriteDisable, [35](#)
 - framWriteEnable, [35](#)
 - framWriteMemory, [35](#)
 - framWriteProtect, [35](#)
 - framWriteSr, [35](#)
 - TADD, [34](#)
 - TLEN, [34](#)
- fram.h
 - CHIP_SELECT_STATE, [11](#)
 - CSS_ASSERT, [12](#)
 - CSS_RELEASE, [12](#)
 - framChipSelect, [13](#)
 - framReadMemory, [13](#)
 - framReadSr, [13](#)
 - framTest, [13](#)
 - framWriteDisable, [13](#)
 - framWriteEnable, [14](#)
 - framWriteMemory, [14](#)
 - framWriteProtect, [14](#)
 - framWriteSr, [14](#)
 - OC_RD SR, [12](#)
 - OC_READ, [12](#)
 - OC_WRDI, [12](#)
 - OC_WREN, [12](#)
 - OC_WRITE, [12](#)
 - OC_WRSR, [12](#)
 - OPCODE_COMMANDS, [12](#)
 - SR_BP0, [12](#)
 - SR_BP1, [12](#)
 - SR_WEL, [12](#)
 - SR_WPEN, [12](#)
 - STATUS_REGISTER, [12](#)
 - WPS_PROTECTED, [12](#)
 - WPS_WRITEABLE, [12](#)
 - WRITE_PROTECT_STATE, [12](#)
- framChipSelect

- fram.c, [34](#)
- fram.h, [13](#)
- framReadMemory
 - fram.c, [34](#)
 - fram.h, [13](#)
- framReadSr
 - fram.c, [34](#)
 - fram.h, [13](#)
- framTest
 - fram.c, [34](#)
 - fram.h, [13](#)
- framWriteDisable
 - fram.c, [35](#)
 - fram.h, [13](#)
- framWriteEnable
 - fram.c, [35](#)
 - fram.h, [14](#)
- framWriteMemory
 - fram.c, [35](#)
 - fram.h, [14](#)
- framWriteProtect
 - fram.c, [35](#)
 - fram.h, [14](#)
- framWriteSr
 - fram.c, [35](#)
 - fram.h, [14](#)
- GetBrightness
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- GetCurrent
 - current_handler.c, [32](#)
 - current_handler.h, [9](#)
- GetCurrentRange
 - current_handler.c, [32](#)
 - current_handler.h, [9](#)
- GetCurrentValue
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [25](#)
- GetPwm
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- GetTemperature
 - temperature_handler.c, [44](#)
 - temperature_handler.h, [29](#)
- GetTemperatureRange
 - temperature_handler.c, [44](#)
 - temperature_handler.h, [29](#)
- GetThermistorValue
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [25](#)
- GetTIM2Cnt
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [25](#)
- GetVoltage
 - voltage_handler.c, [45](#)
 - voltage_handler.h, [30](#)
- GetVoltageRange
 - voltage_handler.c, [45](#)
- voltage_handler.h, [30](#)
- GetVoltageValue
 - stm32l412xx-bsp.c, [42](#)
 - stm32l412xx-bsp.h, [25](#)
- GPIO_PinState
 - stm32l412xx-bsp.h, [24](#)
- HalfBrightness
 - pwm_handler.c, [41](#)
- HeatingThreshold1_dC
 - temperature_handler.c, [44](#)
- HeatingThreshold2_dC
 - temperature_handler.c, [44](#)
- HighStepTimeMs
 - main.c, [39](#)
- HOLD_BRIGHTNESS_JUMP
 - pwm_handler.h, [19](#)
- HotPwmRatio
 - pwm_handler.c, [41](#)
- IncreaseBrightness
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- InitPwm
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- is_running
 - PwmStruct, [5](#)
 - TimerStruct, [6](#)
- IsBrightPressed
 - button_handler.c, [31](#)
 - button_handler.h, [7](#)
- IsDimPressed
 - button_handler.c, [31](#)
 - button_handler.h, [7](#)
- logger.c
 - LogNumber, [37](#)
 - LogString, [37](#)
 - ReadLog, [37](#)
- logger.h
 - LogNumber, [16](#)
 - LogString, [16](#)
 - ReadLog, [16](#)
- LogNumber
 - logger.c, [37](#)
 - logger.h, [16](#)
- LogString
 - logger.c, [37](#)
 - logger.h, [16](#)
- LOWER_STEP_TIME_MS
 - main.c, [38](#)
- LOWER_SWEEP_TIME_MS
 - main.c, [38](#)
- LowStepTimeMs
 - main.c, [39](#)
- main
 - main.c, [38](#)

- main.c
 - AVG_STEP_DIFF_MS, 38
 - AVG_STEP_TIME_MS, 38
 - HighStepTimeMs, 39
 - LOWER_STEP_TIME_MS, 38
 - LOWER_SWEEP_TIME_MS, 38
 - LowStepTimeMs, 39
 - main, 38
 - SystemClock_Config, 39
 - TOTAL_SWEEP_TIME_MS, 38
 - UPPER_STEP_TIME_MS, 38
 - UPPER_SWEEP_TIME_MS, 38
- main.h
 - Error_Handler, 17
- MaxBrightness
 - pwm_handler.c, 41
- MaxPw
 - pwm_handler.c, 41
- MinBrightness
 - pwm_handler.c, 41
- MinPw
 - pwm_handler.c, 41
- NVIC_PRIORITYGROUP_0
 - stm32l412xx-bsp.h, 22
- NVIC_PRIORITYGROUP_1
 - stm32l412xx-bsp.h, 22
- NVIC_PRIORITYGROUP_2
 - stm32l412xx-bsp.h, 23
- NVIC_PRIORITYGROUP_3
 - stm32l412xx-bsp.h, 23
- NVIC_PRIORITYGROUP_4
 - stm32l412xx-bsp.h, 23
- OC_RDSR
 - fram.h, 12
- OC_READ
 - fram.h, 12
- OC_WRDI
 - fram.h, 12
- OC_WREN
 - fram.h, 12
- OC_WRITE
 - fram.h, 12
- OC_WRSR
 - fram.h, 12
- OPCODE_COMMANDS
 - fram.h, 12
- PIN_RESET
 - stm32l412xx-bsp.h, 24
- PIN_SET
 - stm32l412xx-bsp.h, 24
- pulse_width
 - PwmStruct, 5
- PW_PERIOD
 - pwm_handler.c, 40
- pwm_handler.c
 - DecreaseBrightness, 40
 - GetBrightness, 40
 - GetPwm, 40
 - HalfBrightness, 41
 - HotPwmRatio, 41
 - IncreaseBrightness, 40
 - InitPwm, 40
 - MaxBrightness, 41
 - MaxPw, 41
 - MinBrightness, 41
 - MinPw, 41
 - PW_PERIOD, 40
 - SetBrightness, 40
 - SetPwm, 40
 - TurnOffPwm, 40
 - WarmPwmRatio, 41
- pwm_handler.h
 - BRIGHTNESS_STEPS, 19
 - DecreaseBrightness, 19
 - GetBrightness, 19
 - GetPwm, 19
 - HOLD_BRIGHTNESS_JUMP, 19
 - IncreaseBrightness, 19
 - InitPwm, 19
 - SetBrightness, 19
 - SetPwm, 19
 - TurnOffPwm, 19
- PWM_OUT_GPIO_Port
 - stm32l412xx-bsp.h, 23
- PWM_OUT_Pin
 - stm32l412xx-bsp.h, 23
- PwmStruct, 5
 - is_running, 5
 - pulse_width, 5
- RawTodAmps
 - current_handler.c, 32
- RawTodColts
 - voltage_handler.c, 45
- ReadBrightPin
 - stm32l412xx-bsp.c, 43
 - stm32l412xx-bsp.h, 25
- ReadDimPin
 - stm32l412xx-bsp.c, 43
 - stm32l412xx-bsp.h, 25
- ReadLog
 - logger.c, 37
 - logger.h, 16
- receiveData
 - stm32l412xx-bsp.c, 43
 - stm32l412xx-bsp.h, 25
- RestartDelayCounter
 - delay_handler.c, 32
 - delay_handler.h, 10
- RestartTIM2
 - stm32l412xx-bsp.c, 43
 - stm32l412xx-bsp.h, 25
- sendUARTChar
 - stm32l412xx-bsp.c, 43

- stm32l412xx-bsp.h, [25](#)
- SetBrightness
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- SetPW11
 - stm32l412xx-bsp.c, [43](#)
 - stm32l412xx-bsp.h, [26](#)
- SetPwm
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- SPI_NSS_GPIO_Port
 - stm32l412xx-bsp.h, [23](#)
- SPI_NSS_Pin
 - stm32l412xx-bsp.h, [23](#)
- SPI_WP_GPIO_Port
 - stm32l412xx-bsp.h, [23](#)
- SPI_WP_Pin
 - stm32l412xx-bsp.h, [23](#)
- SR_BP0
 - fram.h, [12](#)
- SR_BP1
 - fram.h, [12](#)
- SR_WEL
 - fram.h, [12](#)
- SR_WPEN
 - fram.h, [12](#)
- StartDelayCounter
 - delay_handler.c, [33](#)
 - delay_handler.h, [10](#)
- StartPWM11
 - stm32l412xx-bsp.c, [43](#)
 - stm32l412xx-bsp.h, [26](#)
- StartTIM2
 - stm32l412xx-bsp.c, [43](#)
 - stm32l412xx-bsp.h, [26](#)
- STATUS_REGISTER
 - fram.h, [12](#)
- stm32l412xx-bsp.c
 - disableChipSelect, [42](#)
 - DisablePWM1, [42](#)
 - disableWriteProtect, [42](#)
 - enableChipSelect, [42](#)
 - EnablePWM1, [42](#)
 - enableWriteProtect, [42](#)
 - Error_Handler, [42](#)
 - GetCurrentValue, [42](#)
 - GetThermistorValue, [42](#)
 - GetTIM2Cnt, [42](#)
 - GetVoltageValue, [42](#)
 - ReadBrightPin, [43](#)
 - ReadDimPin, [43](#)
 - receiveData, [43](#)
 - RestartTIM2, [43](#)
 - sendUARTChar, [43](#)
 - SetPW11, [43](#)
 - StartPWM11, [43](#)
 - StartTIM2, [43](#)
 - StopPWM11, [43](#)
- transferData, [43](#)
- stm32l412xx-bsp.h
 - AMPMETER_ADC_GPIO_Port, [22](#)
 - AMPMETER_ADC_Pin, [22](#)
 - BRIGHT_GPIO_Port, [22](#)
 - BRIGHT_Pin, [22](#)
 - BUTTON_PRESSED, [24](#)
 - BUTTON_UNPRESSED, [24](#)
 - CLK_FREQ_HZ, [22](#)
 - DIM_GPIO_Port, [22](#)
 - DIM_Pin, [22](#)
 - disableChipSelect, [24](#)
 - DisablePWM1, [24](#)
 - disableWriteProtect, [24](#)
 - EEPROM_MISO_GPIO_Port, [22](#)
 - EEPROM_MISO_Pin, [22](#)
 - EEPROM_MOSI_GPIO_Port, [22](#)
 - EEPROM_MOSI_Pin, [22](#)
 - EEPROM_SCK_GPIO_Port, [22](#)
 - EEPROM_SCK_Pin, [22](#)
 - enableChipSelect, [24](#)
 - EnablePWM1, [24](#)
 - enableWriteProtect, [25](#)
 - Error_Handler, [25](#)
 - GetCurrentValue, [25](#)
 - GetThermistorValue, [25](#)
 - GetTIM2Cnt, [25](#)
 - GetVoltageValue, [25](#)
 - GPIO_PinState, [24](#)
 - NVIC_PRIORITYGROUP_0, [22](#)
 - NVIC_PRIORITYGROUP_1, [22](#)
 - NVIC_PRIORITYGROUP_2, [23](#)
 - NVIC_PRIORITYGROUP_3, [23](#)
 - NVIC_PRIORITYGROUP_4, [23](#)
 - PIN_RESET, [24](#)
 - PIN_SET, [24](#)
 - PWM_OUT_GPIO_Port, [23](#)
 - PWM_OUT_Pin, [23](#)
 - ReadBrightPin, [25](#)
 - ReadDimPin, [25](#)
 - receiveData, [25](#)
 - RestartTIM2, [25](#)
 - sendUARTChar, [25](#)
 - SetPW11, [26](#)
 - SPI_NSS_GPIO_Port, [23](#)
 - SPI_NSS_Pin, [23](#)
 - SPI_WP_GPIO_Port, [23](#)
 - SPI_WP_Pin, [23](#)
 - StartPWM11, [26](#)
 - StartTIM2, [26](#)
 - StopPWM11, [26](#)
 - THERMISTOR_ADC_GPIO_Port, [23](#)
 - THERMISTOR_ADC_Pin, [23](#)
 - TIM2_CLK_DEV, [23](#)
 - TIM2_CLK_PRESCALER, [23](#)
 - transferData, [26](#)
 - VOLTMETER_ADC_GPIO_Port, [23](#)
 - VOLTMETER_ADC_Pin, [24](#)

- StopPWM11
 - stm32l412xx-bsp.c, [43](#)
 - stm32l412xx-bsp.h, [26](#)
- SystemClock_Config
 - main.c, [39](#)
- TADD
 - fram.c, [34](#)
- TempCool
 - temperature_handler.h, [28](#)
- temperature_handler.c
 - CoolingThreshold1_dC, [44](#)
 - CoolingThreshold2_dC, [44](#)
 - dCelciusToThermistor, [44](#)
 - GetTemperature, [44](#)
 - GetTemperatureRange, [44](#)
 - HeatingThreshold1_dC, [44](#)
 - HeatingThreshold2_dC, [44](#)
 - ThermistorTodCelcius, [44](#)
- temperature_handler.h
 - GetTemperature, [29](#)
 - GetTemperatureRange, [29](#)
 - TempCool, [28](#)
 - TemperatureRange_e, [28](#)
 - TempHot, [29](#)
 - TempWarm, [29](#)
- TemperatureRange_e
 - temperature_handler.h, [28](#)
- TempHot
 - temperature_handler.h, [29](#)
- TempWarm
 - temperature_handler.h, [29](#)
- THERMISTOR_ADC_GPIO_Port
 - stm32l412xx-bsp.h, [23](#)
- THERMISTOR_ADC_Pin
 - stm32l412xx-bsp.h, [23](#)
- ThermistorTodCelcius
 - temperature_handler.c, [44](#)
- TIM2_CLK_DEV
 - stm32l412xx-bsp.h, [23](#)
- TIM2_CLK_PRESCALER
 - stm32l412xx-bsp.h, [23](#)
- Tim2ClkKhz
 - delay_handler.c, [33](#)
- time
 - TimerStruct, [6](#)
- TimerStruct, [5](#)
 - is_running, [6](#)
 - time, [6](#)
- TLEN
 - fram.c, [34](#)
- TOTAL_SWEEP_TIME_MS
 - main.c, [38](#)
- transferData
 - stm32l412xx-bsp.c, [43](#)
 - stm32l412xx-bsp.h, [26](#)
- TurnOffPwm
 - pwm_handler.c, [40](#)
 - pwm_handler.h, [19](#)
- UPPER_STEP_TIME_MS
 - main.c, [38](#)
- UPPER_SWEEP_TIME_MS
 - main.c, [38](#)
- voltage_handler.c
 - dColtsToRaw, [45](#)
 - GetVoltage, [45](#)
 - GetVoltageRange, [45](#)
 - RawTodColts, [45](#)
 - VoltageErrorHighThreshold_dV, [45](#)
 - VoltageErrorLowThreshold_dV, [45](#)
 - VoltageHighThreshold_dV, [45](#)
 - VoltageLowThreshold_dV, [45](#)
- voltage_handler.h
 - GetVoltage, [30](#)
 - GetVoltageRange, [30](#)
 - VoltageErrorHigh, [30](#)
 - VoltageErrorLow, [30](#)
 - VoltageHigh, [30](#)
 - VoltageLow, [30](#)
 - VoltageNormal, [30](#)
 - VoltageRange_e, [30](#)
- VoltageErrorHigh
 - voltage_handler.h, [30](#)
- VoltageErrorHighThreshold_dV
 - voltage_handler.c, [45](#)
- VoltageErrorLow
 - voltage_handler.h, [30](#)
- VoltageErrorLowThreshold_dV
 - voltage_handler.c, [45](#)
- VoltageHigh
 - voltage_handler.h, [30](#)
- VoltageHighThreshold_dV
 - voltage_handler.c, [45](#)
- VoltageLow
 - voltage_handler.h, [30](#)
- VoltageLowThreshold_dV
 - voltage_handler.c, [45](#)
- VoltageNormal
 - voltage_handler.h, [30](#)
- VoltageRange_e
 - voltage_handler.h, [30](#)
- VOLTMETER_ADC_GPIO_Port
 - stm32l412xx-bsp.h, [23](#)
- VOLTMETER_ADC_Pin
 - stm32l412xx-bsp.h, [24](#)
- WarmPwmRatio
 - pwm_handler.c, [41](#)
- WPS_PROTECTED
 - fram.h, [12](#)
- WPS_WRITEABLE
 - fram.h, [12](#)
- WRITE_PROTECT_STATE
 - fram.h, [12](#)