CH-53K

1.0.0

Generated by Doxygen 1.12.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 PwmStruct Struct Reference

`#include <stm32l412xx-bsp.h>`

Data Fields

- uint8_t is_running
- uint32_t pulse_width

3.1.1 Detailed Description

Testing PWM Struct

3.1.2 Field Documentation

3.1.2.1 is_running

`uint8_t is_running`

pwm is running

3.1.2.2 pulse_width

`uint32_t pulse_width`

pulse width value

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

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

3.2 TimerStruct Struct Reference

`#include <stm32l412xx-bsp.h>`

Data Fields

- uint8_t is_running
- uint32_t time

3.2.1 Detailed Description

Testing Timer Struct

3.2.2 Field Documentation

3.2.2.1 is_running

`uint8_t is_running`

timer is running

3.2.2.2 time

`uint32_t time`

timer value

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

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

Chapter 4

File Documentation

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

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

Functions

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

4.1.1 Function Documentation

4.1.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
            void )
```

Return state of brighten button.

Parameters

out	*Bright*	Pin State, pressed or not

4.1.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
            void )
```

Return state of dim button.

Parameters

out	*Dim*	Pin State, pressed or not

4.1.1.3 IsTogglePressed()

```
GPIO_PinState IsTogglePressed (
            void )
```

Return state of toggle button.

Parameters

out	*Toogle*	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 IsTogglePressed ( void );
00037
00042 GPIO_PinState IsDimPressed ( void );
00043
00048 GPIO_PinState IsBrightPressed ( void );
00049
00050 #endif /* INC_button_handlerh */
```

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

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

Enumerations

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

Functions

- uint16_t GetCurrent (void)

  Get current from ammeter.
- CurrentRange_e GetCurrentRange (void)

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

4.3.1 Enumeration Type Documentation

4.3.1.1 CurrentRange_e

```
enum CurrentRange_e
```

Current Range Enum

Enumerator

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

4.3.2 Function Documentation

4.3.2.1 GetCurrent()

```
uint16_t GetCurrent (
            void )
```

Get current from ammeter.

Parameters

out	*current*	level in dA

4.3.2.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
            void )
```

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

Parameters

out	*Current*	current range

4.4 current_handler.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002   *
00003   *  @attention
00004   *  Copyright (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
00031 typedef enum
00032 {
00033     CurrentNormal = 0,
00034     CurrentHigh = 1,
00035     CurrentError = 2
00036 } CurrentRange_e;
00037
00043 uint16_t GetCurrent ( void );
00044
00053 CurrentRange_e GetCurrentRange ( void );
00054
00055 #endif /* INC_current_handlerh */
```

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

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

Functions

- void StartDelayCounter (void)

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

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

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

  Returns a delay value for a brightness level.

4.5.1 Function Documentation

4.5.1.1 BrightnessDelay()

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

Returns a delay value for a brightness level.

Parameters

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

4.5.1.2 DelayHit()

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

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

Parameters

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

4.5.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
            void )
```

Restart the delay counter.

4.5.1.4 StartDelayCounter()

```
void StartDelayCounter (
            void )
```

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

4.6 delay_handler.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  *  Copyright (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 );
00040
00046 uint8_t DelayHit ( uint32_t delay_ms );
00047
00053 uint16_t BrightnessDelay ( int8_t brightness );
00054
00055 #endif /* INC_delay_handlerh */
```

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

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

Enumerations

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

Functions

- void framWriteProtect (WRITE_PROTECT_STATE state)
- void framChipSelect (CHIP_SELECT_STATE state)
- void framReadSr (unsigned char *srP)
- void framWriteSr (unsigned char sr)

- void framWriteDisable (void)

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

  This routine resets the write enable latch.
- void framReadMemory (unsigned short addr, unsigned char *rdBufP, unsigned short len)
- void framWriteMemory (unsigned short addr, const unsigned char *const wrBufP, unsigned short len)
- uint8_t framTest (void)

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

4.7.1 Enumeration Type Documentation

4.7.1.1 CHIP_SELECT_STATE

`enum CHIP_SELECT_STATE`

chip select state

Enumerator

CSS_ASSERT	chip select disabled
CSS_RELEASE	chip select enabled

4.7.1.2 OPCODE_COMMANDS

`enum OPCODE_COMMANDS`

opcode command

Enumerator

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

4.7.1.3 STATUS_REGISTER

`enum STATUS_REGISTER`

status register

Enumerator

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

4.7.1.4 WRITE_PROTECT_STATE

```
enum WRITE_PROTECT_STATE
```

write protect state

Enumerator

WPS_PROTECTED	write protected
WPS_WRITEABLE	write enabled

4.7.2 Function Documentation

4.7.2.1 framChipSelect()

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

4.7.2.2 framReadMemory()

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

4.7.2.3 framReadSr()

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

4.7.2.4 framTest()

```
uint8_t framTest (
            void )
```

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

Parameters

out	*1*	= pass, 0 = fail

4.7.2.5 framWriteDisable()

```
void framWriteDisable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.7.2.6 framWriteEnable()

```
void framWriteEnable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.7.2.7 framWriteMemory()

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

4.7.2.8 framWriteProtect()

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

4.7.2.9 framWriteSr()

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

4.8 fram.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  *  Copyright (c) 2022, 2023 Luminator, An LTG Company
00005  *  All rights reserved.
00006  *  Any use without the prior written consent of Luminator,
00007  *  An LTG Company is strictly prohibited.
00008  *
00009  *******************************************************************************
00010  *******************************************************************************
00011  *
00012  *  @file fram.h
00013  *
00014  *  @brief This module contains definitions and structures to support
00015  *          fram.c SPI FRAM operations.
00016  *
00017  *  Revision History:
00018  *    Date         Name            Ver     Remarks
00019  *    -----------  ----------      ---     --------------------------------
00020  *    04/09/2023   Mark Lane       0       Original Version
00021  *    09/10/2024 - Austin Green -  2.0 -   Doxyfile documentation
00022  *
00023  *  Notes:
00024  *
00025  *******************************************************************************/
00026 #ifndef _FRAM_H_
00027 #define _FRAM_H_
00028
00029 #include "stm32l412xx-bsp.h"
00030
00031 /* ------------------------ Local Definition(s) ---------------------------- */
00035 typedef enum
00036 {
00037
00038     OC_WREN = 6,
00039     OC_WRDI = 4,
00040     OC_RDSR = 5,
00041     OC_WRSR = 1,
00042     OC_READ = 3,
00043     OC_WRITE = 2,
00045 } OPCODE_COMMANDS ;
00046
00047
00051 typedef enum
00052 {
00053
00054     SR_WEL = 0x2,
00055     SR_BP0 = 0x4,
00056     SR_BP1 = 0x8,
00057     SR_WPEN = 0x80,
00059 } STATUS_REGISTER ;
00060
00061
00065 typedef enum
00066 {
00067
00068     WPS_PROTECTED = 0,
00069     WPS_WRITEABLE = 1,
00071 } WRITE_PROTECT_STATE ;
00072
00073
00077 typedef enum
00078 {
00079
00080     CSS_ASSERT = 0,
00081     CSS_RELEASE = 1,
00083 } CHIP_SELECT_STATE ;
00084
00085 /* Prototype Definition */
00095 void framWriteProtect ( WRITE_PROTECT_STATE state ) ;
00096
00105 void framChipSelect ( CHIP_SELECT_STATE state ) ;
00106
00115 void framReadSr ( unsigned char* srP ) ;
00116
00125 void framWriteSr ( unsigned char sr ) ;
00126
00133 void framWriteDisable ( void ) ;
00134
00141 void framWriteEnable ( void ) ;
00142
00157 void framReadMemory ( unsigned short addr, unsigned char* rdBufP,
```

```
00158                       unsigned short len ) ;
00159
00174 void framWriteMemory ( unsigned short addr, const unsigned char* const wrBufP,
00175                         unsigned short len ) ;
00176
00187 uint8_t framTest ( void ) ;
00188
00189
00190 #endif
```

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

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

Functions

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

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

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

  Reads the log at a given address and size.

4.9.1 Function Documentation

4.9.1.1 LogNumber()

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

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

Parameters

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

4.9.1.2 LogString()

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

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

Parameters

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

4.9.1.3 ReadLog()

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

Reads the log at a given address and size.

Parameters

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

4.10 logger.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002  *
00003  * @attention
00004  *  Copyright (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

: Header for main.c file. This file contains the common defines of the application.

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

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

4.11.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

4.11.2 Function Documentation

4.11.2.1 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 */
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
00068 #endif
00069
00070 #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, uint8_t isIr)

  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, uint8_t isIr)

  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 (uint8_t isIr)

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

  turn off PWM
- int8_t GetBrightness (uint8_t isIr)

  Return Brightness variable.
- void SetBrightness (int8_t brightness, uint8_t isIr)

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

  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,
            uint8_t isIr)
```

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

Parameters

in	*button_held*	If the button is being held (decrements by 3 if so)
in	*isIr*	Are we controlling IR or Visible LEDs

4.15.2.2 GetBrightness()

```
int8_t GetBrightness (
            uint8_t isIr)
```

Return Brightness variable.

Parameters

in	*isIr*	Are we controlling IR or Visible LEDs
out	*Current*	LED brightness level

4.15.2.3 GetPwm()

```
uint8_t GetPwm (
            uint8_t isIr)
```

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

Parameters

in	*isIr*	Are we controlling IR or Visible LEDs
out	*Current*	PWM value

4.15.2.4 IncreaseBrightness()

```
void IncreaseBrightness (
            uint8_t button_held,
            uint8_t isIr)
```

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

Parameters

in	*button_held*	If the button is being held (increments by 3 if so)
in	*isIr*	Are we controlling IR or Visible LEDs

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,
            uint8_t isIr)
```

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

Parameters

in	*isIr*	Are we controlling IR or Visible LEDs
in	*brightness*	Brightness to set
in	*brightness*	Brightness to set
in	*isIr*	Are we controlling IR or Visible LEDs

4.15.2.7 SetPwm()

```
void SetPwm (
            uint8_t isIr)
```

set PWM based on pwm value

Parameters

in	*is↩ lr*	Are we controlling IR or Visible LEDs

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
00049 void DecreaseBrightness ( uint8_t button_held,
00050                           uint8_t isIr ); // decrease brightness
00051
00060 void IncreaseBrightness ( uint8_t button_held,
00061                           uint8_t isIr ); // increase brightness
00062
00067 void SetPwm ( uint8_t isIr ); // turn on and set PWM
00068
00072 void TurnOffPwm ( void ); // turn of PWM
00073
00079 int8_t GetBrightness ( uint8_t isIr ); // get value of Brightness
00080
00086 void SetBrightness ( int8_t brightness,
00087                      uint8_t isIr ); // set value of Brightness
00088
00094 uint8_t GetPwm ( uint8_t isIr ); // get value of current PWM
00095
00096 #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 SPI_NSS_Pin 0
- #define SPI_NSS_GPIO_Port 0
- #define SPI_WP_Pin 0
- #define SPI_WP_GPIO_Port 0
- #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 VIS_IR_Pin 0
- #define VIS_IR_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 USB_RENUM_N_Pin 0
- #define USB_RENUM_N_GPIO_Port 0
- #define SWDIO_Pin 0
- #define SWDIO_GPIO_Port 0
- #define SWCLK_Pin 0
- #define SWCLK_GPIO_Port 0
- #define BOOT0_Pin 0
- #define BOOT0_GPIO_Port 0
- #define BUTTON_UNPRESSED (GPIO_PIN_RESET)
- #define BUTTON_PRESSED (GPIO_PIN_SET)
- #define CLK_FREQ_HZ (8000000)
- #define TIM2_CLK_DEV (1)
- #define TIM2_CLK_PRESCALER (8000)

Enumerations

- enum GPIO_PinState { GPIO_PIN_RESET = 0U , GPIO_PIN_SET }
- enum { PIN_SET = 1 , PIN_RESET = 0 }

Functions

- GPIO_PinState ReadTogglePin (void)

  Reads toggle pin value.
- 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.

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 BOOT0_GPIO_Port

```
#define BOOT0_GPIO_Port 0
```

4.17.1.4 BOOT0_Pin

```
#define BOOT0_Pin 0
```

4.17.1.5 BRIGHT_GPIO_Port

```
#define BRIGHT_GPIO_Port 0
```

4.17.1.6 BRIGHT_Pin

```
#define BRIGHT_Pin 0
```

4.17.1.7 BUTTON_PRESSED

```
#define BUTTON_PRESSED (GPIO_PIN_SET)
```

4.17.1.8 BUTTON_UNPRESSED

```
#define BUTTON_UNPRESSED (GPIO_PIN_RESET)
```

4.17.1.9 CLK_FREQ_HZ

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

4.17.1.10 DIM_GPIO_Port

```
#define DIM_GPIO_Port 0
```

4.17.1.11 DIM_Pin

```
#define DIM_Pin 0
```

4.17.1.12 EEPROM_MISO_GPIO_Port

```
#define EEPROM_MISO_GPIO_Port 0
```

4.17.1.13 EEPROM_MISO_Pin

```
#define EEPROM_MISO_Pin 0
```

4.17.1.14 EEPROM_MOSI_GPIO_Port

```
#define EEPROM_MOSI_GPIO_Port 0
```

4.17.1.15 EEPROM_MOSI_Pin

```
#define EEPROM_MOSI_Pin 0
```

4.17.1.16 EEPROM_SCK_GPIO_Port

```
#define EEPROM_SCK_GPIO_Port 0
```

4.17.1.17 EEPROM_SCK_Pin

```
#define EEPROM_SCK_Pin 0
```

4.17.1.18 PWM_OUT_GPIO_Port

```
#define PWM_OUT_GPIO_Port 0
```

4.17.1.19 PWM_OUT_Pin

```
#define PWM_OUT_Pin 0
```

4.17.1.20 SPI_NSS_GPIO_Port

```
#define SPI_NSS_GPIO_Port 0
```

4.17.1.21 SPI_NSS_Pin

```
#define SPI_NSS_Pin 0
```

4.17.1.22 SPI_WP_GPIO_Port

```
#define SPI_WP_GPIO_Port 0
```

4.17.1.23 SPI_WP_Pin

```
#define SPI_WP_Pin 0
```

4.17.1.24 SWCLK_GPIO_Port

```
#define SWCLK_GPIO_Port 0
```

4.17.1.25 SWCLK_Pin

```
#define SWCLK_Pin 0
```

4.17.1.26 SWDIO_GPIO_Port

```
#define SWDIO_GPIO_Port 0
```

4.17.1.27 SWDIO_Pin

```
#define SWDIO_Pin 0
```

4.17.1.28 THERMISTOR_ADC_GPIO_Port

```
#define THERMISTOR_ADC_GPIO_Port 0
```

4.17.1.29 THERMISTOR_ADC_Pin

```
#define THERMISTOR_ADC_Pin 0
```

4.17.1.30 TIM2_CLK_DEV

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

4.17.1.31 TIM2_CLK_PRESCALER

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

4.17.1.32 USB_RENUM_N_GPIO_Port

```
#define USB_RENUM_N_GPIO_Port 0
```

4.17.1.33 USB_RENUM_N_Pin

```
#define USB_RENUM_N_Pin 0
```

4.17.1.34 VIS_IR_GPIO_Port

```
#define VIS_IR_GPIO_Port 0
```

4.17.1.35 VIS_IR_Pin

```
#define VIS_IR_Pin 0
```

4.17.1.36 VOLTMETER_ADC_GPIO_Port

```
#define VOLTMETER_ADC_GPIO_Port 0
```

4.17.1.37 VOLTMETER_ADC_Pin

```
#define VOLTMETER_ADC_Pin 0
```

4.17.2 Enumeration Type Documentation

4.17.2.1 anonymous enum

```
anonymous enum
```

Enumerator

PIN_SET	
PIN_RESET	

4.17.2.2 GPIO_PinState

```
enum GPIO_PinState
```

GPIO_PinState for Testing

Enumerator

GPIO_PIN_RESET	
GPIO_PIN_SET	

4.17.3 Function Documentation

4.17.3.1 disableChipSelect()

```
void disableChipSelect (
            void )
```

Disables SPI chip select line (active low)

4.17.3.2 DisablePWM1()

```
void DisablePWM1 (
            void )
```

Disables Timer 1.

4.17.3.3 disableWriteProtect()

```
void disableWriteProtect (
            void )
```

Disables SPI write protect line (active high)

4.17.3.4 enableChipSelect()

```
void enableChipSelect (
            void )
```

Enables SPI chip select line (active low)

4.17.3.5 EnablePWM1()

```
void EnablePWM1 (
            void )
```

Enables Timer 1.

4.17.3.6 enableWriteProtect()

```
void enableWriteProtect (
            void )
```

Enables SPI write protect line (active high)

4.17.3.7 GetCurrentValue()

```
int16_t GetCurrentValue (
            void )
```

Returns raw ADC value from ammeter.

Parameters

out	*Ammeter*	raw ADC value

4.17.3.8 GetThermistorValue()

```
int16_t GetThermistorValue (
            void )
```

Returns raw ADC value from thermistor.

Parameters

out	*Thermistor*	raw ADC value

4.17.3.9 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
            void )
```

Returns value in the Timer 2 counter.

Parameters

out	*Value*	of Timer 2 counter

4.17.3.10 GetVoltageValue()

```
int16_t GetVoltageValue (
            void )
```

Returns raw ADC value from voltmeter.

Parameters

out	*Voltmeter*	raw ADC value

4.17.3.11 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
            void )
```

Reads bright pin value.

Parameters

out	*Bright*	pin state

4.17.3.12 ReadDimPin()

```
GPIO_PinState ReadDimPin (
            void )
```

Reads dim pin value.

Parameters

out	*Dim*	pin state

4.17.3.13 ReadTogglePin()

```
GPIO_PinState ReadTogglePin (
            void )
```

Reads toggle pin value.

Parameters

out	*Toggle*	pin state

4.17.3.14 receiveData()

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

Gets data from SPI lines.

Parameters

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

4.17.3.15 RestartTIM2()

```
void RestartTIM2 (
            void )
```

Resets Timer 2 counter to zero.

4.17.3.16 sendUARTChar()

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

Sends character via UART line.

Parameters

in	*c*	Character to send via UART

4.17.3.17 SetPW11()

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

Sets PWM Timer 1 Channel 1 value.

Parameters

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

4.17.3.18 StartPWM11()

```
void StartPWM11 (
            void )
```

Starts PWM Timer 1 Channel 1 output.

4.17.3.19 StartTIM2()

```
void StartTIM2 (
            void )
```

Starts Timer 2 counter.

4.17.3.20 StopPWM11()

```
void StopPWM11 (
            void )
```

Stops PWM Timer 1 Channel 1 output.

4.17.3.21 transferData()

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

Sends data via SPI lines.

Parameters

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

4.18 stm32l412xx-bsp.h

Go to the documentation of this file.

```
00001 /*******************************************************************************
00002  *
00003  *  @attention
00004  *  Copyright (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_hal.h"
00034
00035 #include "stm32l4xx_ll_adc.h"
00036 #include "stm32l4xx_ll_crs.h"
00037 #include "stm32l4xx_ll_rcc.h"
00038 #include "stm32l4xx_ll_bus.h"
00039 #include "stm32l4xx_ll_system.h"
00040 #include "stm32l4xx_ll_exti.h"
00041 #include "stm32l4xx_ll_cortex.h"
00042 #include "stm32l4xx_ll_utils.h"
00043 #include "stm32l4xx_ll_pwr.h"
00044 #include "stm32l4xx_ll_dma.h"
00045 #include "stm32l4xx_ll_spi.h"
00046 #include "stm32l4xx_ll_tim.h"
00047 #include "stm32l4xx_ll_usart.h"
00048 #include "stm32l4xx_ll_gpio.h"
00049
00050 /* Peripherals */
00051 #include "adc.h"
00052 #include "spi.h"
00053 #include "tim.h"
00054 #ifdef ENABLE_UART_DEBUGGING /* tracing enabled */
00055     /* Peripherals enabled for UART */
00056     #include "usart.h"
00057 #endif /* ENABLE_UART_DEBUGGING */
00058 // #include "usb_device.h"
00059 #include "gpio.h"
00060
00061 #define THERMISTOR_ADC_Pin LL_GPIO_PIN_0
00062 #define THERMISTOR_ADC_GPIO_Port GPIOA
00063 #define BRIGHT_Pin LL_GPIO_PIN_1
00064 #define BRIGHT_GPIO_Port GPIOA
00065 #define DIM_Pin LL_GPIO_PIN_3
00066 #define DIM_GPIO_Port GPIOA
00067 #define VIS_IR_Pin LL_GPIO_PIN_4
00068 #define VIS_IR_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 USB_RENUM_N_Pin LL_GPIO_PIN_10
00082 #define USB_RENUM_N_GPIO_Port GPIOA
00083 #define SWDIO_Pin LL_GPIO_PIN_13
00084 #define SWDIO_GPIO_Port GPIOA
00085 #define SWCLK_Pin LL_GPIO_PIN_14
00086 #define SWCLK_GPIO_Port GPIOA
00087 #define SPI_WP_Pin LL_GPIO_PIN_5
00088 #define SPI_WP_GPIO_Port GPIOB
00089 #define SPI_NSS_Pin LL_GPIO_PIN_6
00090 #define SPI_NSS_GPIO_Port GPIOB
00091 #define BOOT0_Pin LL_GPIO_PIN_3
00092 #define BOOT0_GPIO_Port GPIOH
00093
00094 #else /* STM32L412xx */
00095
00096 /* Below is for debugging purposes */
00097 #define SPI_NSS_Pin               0
00098 #define SPI_NSS_GPIO_Port         0
00099 #define SPI_WP_Pin                0
00100 #define SPI_WP_GPIO_Port          0
00101 #define THERMISTOR_ADC_Pin        0
00102 #define THERMISTOR_ADC_GPIO_Port  0
00103 #define BRIGHT_Pin                0
00104 #define BRIGHT_GPIO_Port          0
00105 #define DIM_Pin                   0
00106 #define DIM_GPIO_Port             0
00107 #define VIS_IR_Pin                0
00108 #define VIS_IR_GPIO_Port          0
00109 #define EEPROM_SCK_Pin            0
00110 #define EEPROM_SCK_GPIO_Port      0
00111 #define EEPROM_MISO_Pin           0
00112 #define EEPROM_MISO_GPIO_Port     0
00113 #define EEPROM_MOSI_Pin           0
00114 #define EEPROM_MOSI_GPIO_Port     0
00115 #define VOLTMETER_ADC_Pin         0
00116 #define VOLTMETER_ADC_GPIO_Port   0
00117 #define AMPMETER_ADC_Pin          0
00118 #define AMPMETER_ADC_GPIO_Port    0
00119 #define PWM_OUT_Pin               0
00120 #define PWM_OUT_GPIO_Port         0
00121 #define USB_RENUM_N_Pin           0
00122 #define USB_RENUM_N_GPIO_Port     0
00123 #define SWDIO_Pin                 0
00124 #define SWDIO_GPIO_Port           0
00125 #define SWCLK_Pin                 0
00126 #define SWCLK_GPIO_Port           0
00127 #define BOOT0_Pin                 0
00128 #define BOOT0_GPIO_Port           0
00129
00133 typedef enum
00134 {
00135     GPIO_PIN_RESET = 0U,
00136     GPIO_PIN_SET
00137 } GPIO_PinState;
00138
00142 typedef struct
00143 {
00144     uint8_t is_running;
00145     uint32_t pulse_width;
00146 } PwmStruct;
00147
00151 typedef struct
00152 {
00153     uint8_t is_running;
00154     uint32_t time;
00155 } TimerStruct;
00156
00157 #endif /* STM32L412xx */
00158
00159 /* Button Defines */
00160 #define BUTTON_UNPRESSED (GPIO_PIN_RESET)
00161 #define BUTTON_PRESSED (GPIO_PIN_SET)
00162 enum { PIN_SET = 1, PIN_RESET = 0};
00163
00164 /* Clock frequency Values */
00165 #define CLK_FREQ_HZ  (8000000)
```

```
00166 #define TIM2_CLK_DEV  (1)
00167 #define TIM2_CLK_PRESCALER  (8000)
00168
00169 /* Returns button state */
00174 GPIO_PinState ReadTogglePin ( void );
00175
00180 GPIO_PinState ReadDimPin ( void );
00181
00186 GPIO_PinState ReadBrightPin ( void );
00187 /* PWM Outputs */
00191 void EnablePWM1 ( void );
00192
00196 void DisablePWM1 ( void );
00197
00201 void StartPWM11 ( void );
00202
00206 void StopPWM11 ( void );
00207
00212 void SetPW11 ( uint32_t pulse_width );
00213
00214 /* Timers */
00218 void StartTIM2 ( void );
00219
00223 void RestartTIM2 ( void );
00224
00229 uint32_t GetTIM2Cnt ( void );
00230
00235 int16_t GetThermistorValue ( void );
00236
00241 int16_t GetCurrentValue ( void );
00242
00247 int16_t GetVoltageValue ( void );
00248
00252 void enableWriteProtect ( void );
00253
00257 void disableWriteProtect ( void );
00258
00262 void enableChipSelect ( void );
00263
00267 void disableChipSelect ( void );
00268
00274 void transferData ( const unsigned char* const txData, const uint32_t bytes );
00275
00281 void receiveData ( unsigned char* rxData, const uint32_t bytes );
00282
00287 void sendUARTChar ( char c );
00288
00289 #endif /* INC_bsph */
```

4.19 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_↩ Controller/Core/Inc/stm32l4xx_hal_conf.h File Reference

HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h.

```
#include "stm32l4xx_hal_rcc.h"
#include "stm32l4xx_hal_gpio.h"
#include "stm32l4xx_hal_dma.h"
#include "stm32l4xx_hal_cortex.h"
#include "stm32l4xx_hal_exti.h"
#include "stm32l4xx_hal_flash.h"
#include "stm32l4xx_hal_pcd.h"
#include "stm32l4xx_hal_pwr.h"
```

Macros

- #define HAL_MODULE_ENABLED

  This is the list of modules to be used in the HAL driver.

- #define HAL_PCD_MODULE_ENABLED
- #define HAL_GPIO_MODULE_ENABLED
- #define HAL_EXTI_MODULE_ENABLED
- #define HAL_DMA_MODULE_ENABLED
- #define HAL_RCC_MODULE_ENABLED
- #define HAL_FLASH_MODULE_ENABLED
- #define HAL_PWR_MODULE_ENABLED
- #define HAL_CORTEX_MODULE_ENABLED
- #define HSE_VALUE ((uint32_t)8000000U)

  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSE_STARTUP_TIMEOUT ((uint32_t)100U)
- #define MSI_VALUE ((uint32_t)48000000U)

  Internal Multiple Speed oscillator (MSI) default value. This value is the default MSI range value after Reset.
- #define HSI_VALUE ((uint32_t)16000000U)

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define HSI48_VALUE ((uint32_t)48000000U)

  Internal High Speed oscillator (HSI48) value for USB FS, SDMMC and RNG. This internal oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special Clock Recovery System (CRS) circuitry. When the CRS is not used, the HSI48 RC oscillator runs on it default frequency which is subject to manufacturing process variations.
- #define LSI_VALUE 32000U

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE 32768U

  External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.
- #define LSE_STARTUP_TIMEOUT 5000U
- #define EXTERNAL_SAI1_CLOCK_VALUE 48000U

  External clock source for SAI1 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.
- #define EXTERNAL_SAI2_CLOCK_VALUE 48000U

  External clock source for SAI2 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.
- #define VDD_VALUE 3300U

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY 0U
- #define USE_RTOS 0U
- #define PREFETCH_ENABLE 0U
- #define INSTRUCTION_CACHE_ENABLE 1U
- #define DATA_CACHE_ENABLE 1U
- #define USE_HAL_ADC_REGISTER_CALLBACKS 0U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define USE_HAL_CAN_REGISTER_CALLBACKS 0U
- #define USE_HAL_COMP_REGISTER_CALLBACKS 0U
- #define USE_HAL_CRYP_REGISTER_CALLBACKS 0U
- #define USE_HAL_DAC_REGISTER_CALLBACKS 0U
- #define USE_HAL_DCMI_REGISTER_CALLBACKS 0U
- #define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U
- #define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U
- #define USE_HAL_DSI_REGISTER_CALLBACKS 0U
- #define USE_HAL_GFXMMU_REGISTER_CALLBACKS 0U
- #define USE_HAL_HASH_REGISTER_CALLBACKS 0U
- #define USE_HAL_HCD_REGISTER_CALLBACKS 0U

- #define USE_HAL_I2C_REGISTER_CALLBACKS 0U
- #define USE_HAL_IRDA_REGISTER_CALLBACKS 0U
- #define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U
- #define USE_HAL_LTDC_REGISTER_CALLBACKS 0U
- #define USE_HAL_MMC_REGISTER_CALLBACKS 0U
- #define USE_HAL_OPAMP_REGISTER_CALLBACKS 0U
- #define USE_HAL_OSPI_REGISTER_CALLBACKS 0U
- #define USE_HAL_PCD_REGISTER_CALLBACKS 0U
- #define USE_HAL_QSPI_REGISTER_CALLBACKS 0U
- #define USE_HAL_RNG_REGISTER_CALLBACKS 0U
- #define USE_HAL_RTC_REGISTER_CALLBACKS 0U
- #define USE_HAL_SAI_REGISTER_CALLBACKS 0U
- #define USE_HAL_SD_REGISTER_CALLBACKS 0U
- #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U
- #define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U
- #define USE_HAL_SPI_REGISTER_CALLBACKS 0U
- #define USE_HAL_SWPMI_REGISTER_CALLBACKS 0U
- #define USE_HAL_TIM_REGISTER_CALLBACKS 0U
- #define USE_HAL_TSC_REGISTER_CALLBACKS 0U
- #define USE_HAL_UART_REGISTER_CALLBACKS 0U
- #define USE_HAL_USART_REGISTER_CALLBACKS 0U
- #define USE_HAL_WWDG_REGISTER_CALLBACKS 0U
- #define USE_SPI_CRC 0U
- #define assert_param(expr)

  Include module's header file.

4.19.1 Detailed Description

HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h.

Author

MCD Application Team

Attention

4.19.2 Macro Definition Documentation

4.19.2.1 assert_param

```
#define assert_param(
              expr)
```

Value:

```
((void)0U)
```

Include module's header file.

4.19.2.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1U
```

4.19.2.3 EXTERNAL_SAI1_CLOCK_VALUE

```
#define EXTERNAL_SAI1_CLOCK_VALUE 48000U
```

External clock source for SAI1 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.

Value of the SAI1 External clock source in Hz

4.19.2.4 EXTERNAL_SAI2_CLOCK_VALUE

```
#define EXTERNAL_SAI2_CLOCK_VALUE 48000U
```

External clock source for SAI2 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.

Value of the SAI2 External clock source in Hz

4.19.2.5 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

4.19.2.6 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

4.19.2.7 HAL_EXTI_MODULE_ENABLED

```
#define HAL_EXTI_MODULE_ENABLED
```

4.19.2.8 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

4.19.2.9 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

4.19.2.10 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

4.19.2.11 HAL_PCD_MODULE_ENABLED

```
#define HAL_PCD_MODULE_ENABLED
```

4.19.2.12 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

4.19.2.13 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

4.19.2.14 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100U)
```

Time out for HSE start up, in ms

4.19.2.15 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000U)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

4.19.2.16 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000U)
```

Internal High Speed oscillator (HSI48) value for USB FS, SDMMC and RNG. This internal oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special Clock Recovery System (CRS) circuitry. When the CRS is not used, the HSI48 RC oscillator runs on it default frequency which is subject to manufacturing process variations.

Value of the Internal High Speed oscillator for USB FS/SDMMC/RNG in Hz. The real value my vary depending on manufacturing process variations.

4.19.2.17 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000U)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

4.19.2.18 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1U
```

4.19.2.19 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

4.19.2.20 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.

$<$ Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External oscillator in Hz

4.19.2.21 LSI_VALUE

```
#define LSI_VALUE 32000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

4.19.2.22 MSI_VALUE

```
#define MSI_VALUE ((uint32_t)48000000U)
```

Internal Multiple Speed oscillator (MSI) default value. This value is the default MSI range value after Reset.

Value of the Internal oscillator in Hz

4.19.2.23 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 0U
```

4.19.2.24 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 0U
```

tick interrupt priority

4.19.2.25 USE_HAL_ADC_REGISTER_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

Set below the peripheral configuration to "1U" to add the support of HAL callback registration/deregistration feature for the HAL driver(s). This allows user application to provide specific callback functions thanks to HAL_PPP_↩ RegisterCallback() rather than overwriting the default weak callback functions (see each stm32l4xx_hal_ppp.h file for possible callback identifiers defined in HAL_PPP_CallbackIDTypeDef for each PPP peripheral).

4.19.2.26 USE_HAL_CAN_REGISTER_CALLBACKS

```
#define USE_HAL_CAN_REGISTER_CALLBACKS 0U
```

4.19.2.27 USE_HAL_COMP_REGISTER_CALLBACKS

```
#define USE_HAL_COMP_REGISTER_CALLBACKS 0U
```

4.19.2.28 USE_HAL_CRYP_REGISTER_CALLBACKS

```
#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U
```

4.19.2.29 USE_HAL_DAC_REGISTER_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U
```

4.19.2.30 USE_HAL_DCMI_REGISTER_CALLBACKS

```
#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U
```

4.19.2.31 USE_HAL_DFSDM_REGISTER_CALLBACKS

```
#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U
```

4.19.2.32 USE_HAL_DMA2D_REGISTER_CALLBACKS

```
#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U
```

4.19.2.33 USE_HAL_DSI_REGISTER_CALLBACKS

```
#define USE_HAL_DSI_REGISTER_CALLBACKS 0U
```

4.19.2.34 USE_HAL_GFXMMU_REGISTER_CALLBACKS

```
#define USE_HAL_GFXMMU_REGISTER_CALLBACKS 0U
```

4.19.2.35 USE_HAL_HASH_REGISTER_CALLBACKS

```
#define USE_HAL_HASH_REGISTER_CALLBACKS 0U
```

4.19.2.36 USE_HAL_HCD_REGISTER_CALLBACKS

```
#define USE_HAL_HCD_REGISTER_CALLBACKS 0U
```

4.19.2.37 USE_HAL_I2C_REGISTER_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U
```

4.19.2.38 USE_HAL_IRDA_REGISTER_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U
```

4.19.2.39 USE_HAL_LPTIM_REGISTER_CALLBACKS

```
#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U
```

4.19.2.40 USE_HAL_LTDC_REGISTER_CALLBACKS

```
#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U
```

4.19.2.41 USE_HAL_MMC_REGISTER_CALLBACKS

```
#define USE_HAL_MMC_REGISTER_CALLBACKS 0U
```

4.19.2.42 USE_HAL_OPAMP_REGISTER_CALLBACKS

```
#define USE_HAL_OPAMP_REGISTER_CALLBACKS 0U
```

4.19.2.43 USE_HAL_OSPI_REGISTER_CALLBACKS

```
#define USE_HAL_OSPI_REGISTER_CALLBACKS 0U
```

4.19.2.44 USE_HAL_PCD_REGISTER_CALLBACKS

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U
```

4.19.2.45 USE_HAL_QSPI_REGISTER_CALLBACKS

```
#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U
```

4.19.2.46 USE_HAL_RNG_REGISTER_CALLBACKS

```
#define USE_HAL_RNG_REGISTER_CALLBACKS 0U
```

4.19.2.47 USE_HAL_RTC_REGISTER_CALLBACKS

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U
```

4.19.2.48 USE_HAL_SAI_REGISTER_CALLBACKS

```
#define USE_HAL_SAI_REGISTER_CALLBACKS 0U
```

4.19.2.49 USE_HAL_SD_REGISTER_CALLBACKS

```
#define USE_HAL_SD_REGISTER_CALLBACKS 0U
```

4.19.2.50 USE_HAL_SMARTCARD_REGISTER_CALLBACKS

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U
```

4.19.2.51 USE_HAL_SMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U
```

4.19.2.52 USE_HAL_SPI_REGISTER_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U
```

4.19.2.53 USE_HAL_SWPMI_REGISTER_CALLBACKS

```
#define USE_HAL_SWPMI_REGISTER_CALLBACKS 0U
```

4.19.2.54 USE_HAL_TIM_REGISTER_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U
```

4.19.2.55 USE_HAL_TSC_REGISTER_CALLBACKS

```
#define USE_HAL_TSC_REGISTER_CALLBACKS 0U
```

4.19.2.56 USE_HAL_UART_REGISTER_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U
```

4.19.2.57 USE_HAL_USART_REGISTER_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U
```

4.19.2.58 USE_HAL_WWDG_REGISTER_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U
```

4.19.2.59 USE_RTOS

```
#define USE_RTOS 0U
```

4.19.2.60 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

4.19.2.61 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

4.20 stm32l4xx_hal_conf.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00021 /* USER CODE END Header */
00022
00023 /* Define to prevent recursive inclusion -------------------------------------*/
00024 #ifndef INC_stm32l4xx_hal_confh
00025 #define INC_stm32l4xx_hal_confh
00026
00027 #ifdef __cplusplus
00028 extern "C" {
00029 #endif
00030
00031 /* Exported types ------------------------------------------------------------*/
00032 /* Exported constants --------------------------------------------------------*/
00033
00034 /* ########################## Module Selection ############################## */
00038 #define HAL_MODULE_ENABLED
00039 /*#define HAL_ADC_MODULE_ENABLED   */
00040 /*#define HAL_CRYP_MODULE_ENABLED   */
00041 /*#define HAL_CAN_MODULE_ENABLED   */
00042 /*#define HAL_COMP_MODULE_ENABLED   */
00043 /*#define HAL_I2C_MODULE_ENABLED   */
00044 /*#define HAL_CRC_MODULE_ENABLED   */
00045 /*#define HAL_CRYP_MODULE_ENABLED   */
00046 /*#define HAL_DAC_MODULE_ENABLED   */
00047 /*#define HAL_DCMI_MODULE_ENABLED   */
00048 /*#define HAL_DMA2D_MODULE_ENABLED   */
00049 /*#define HAL_DFSDM_MODULE_ENABLED   */
00050 /*#define HAL_DSI_MODULE_ENABLED   */
00051 /*#define HAL_FIREWALL_MODULE_ENABLED   */
00052 /*#define HAL_GFXMMU_MODULE_ENABLED   */
00053 /*#define HAL_HCD_MODULE_ENABLED   */
00054 /*#define HAL_HASH_MODULE_ENABLED   */
00055 /*#define HAL_I2S_MODULE_ENABLED   */
00056 /*#define HAL_IRDA_MODULE_ENABLED   */
00057 /*#define HAL_IWDG_MODULE_ENABLED   */
00058 /*#define HAL_LTDC_MODULE_ENABLED   */
00059 /*#define HAL_LCD_MODULE_ENABLED   */
00060 /*#define HAL_LPTIM_MODULE_ENABLED   */
00061 /*#define HAL_MMC_MODULE_ENABLED   */
00062 /*#define HAL_NAND_MODULE_ENABLED   */
00063 /*#define HAL_NOR_MODULE_ENABLED   */
00064 /*#define HAL_OPAMP_MODULE_ENABLED   */
00065 /*#define HAL_OSPI_MODULE_ENABLED   */
00066 /*#define HAL_OSPI_MODULE_ENABLED   */
00067 #define HAL_PCD_MODULE_ENABLED
00068 /*#define HAL_PKA_MODULE_ENABLED   */
00069 /*#define HAL_QSPI_MODULE_ENABLED   */
00070 /*#define HAL_QSPI_MODULE_ENABLED   */
00071 /*#define HAL_RNG_MODULE_ENABLED   */
00072 /*#define HAL_RTC_MODULE_ENABLED   */
00073 /*#define HAL_SAI_MODULE_ENABLED   */
00074 /*#define HAL_SD_MODULE_ENABLED   */
00075 /*#define HAL_SMBUS_MODULE_ENABLED   */
00076 /*#define HAL_SMARTCARD_MODULE_ENABLED   */
00077 /*#define HAL_SPI_MODULE_ENABLED   */
00078 /*#define HAL_SRAM_MODULE_ENABLED   */
00079 /*#define HAL_SWPMI_MODULE_ENABLED   */
00080 /*#define HAL_TIM_MODULE_ENABLED   */
00081 /*#define HAL_TSC_MODULE_ENABLED   */
00082 /*#define HAL_UART_MODULE_ENABLED   */
00083 /*#define HAL_USART_MODULE_ENABLED   */
00084 /*#define HAL_WWDG_MODULE_ENABLED   */
00085 /*#define HAL_EXTI_MODULE_ENABLED   */
00086 /*#define HAL_PSSI_MODULE_ENABLED   */
00087 #define HAL_GPIO_MODULE_ENABLED
00088 #define HAL_EXTI_MODULE_ENABLED
00089 #define HAL_DMA_MODULE_ENABLED
00090 #define HAL_RCC_MODULE_ENABLED
00091 #define HAL_FLASH_MODULE_ENABLED
00092 #define HAL_PWR_MODULE_ENABLED
00093 #define HAL_CORTEX_MODULE_ENABLED
00094
00095 /* ########################## Oscillator Values adaptation ####################*/
00101 #if !defined  (HSE_VALUE)
00102 #define HSE_VALUE    ((uint32_t)8000000U)
00103 #endif /* HSE_VALUE */
00104
00105 #if !defined  (HSE_STARTUP_TIMEOUT)
00106 #define HSE_STARTUP_TIMEOUT    ((uint32_t)100U)
00107 #endif /* HSE_STARTUP_TIMEOUT */
00108
00113 #if !defined  (MSI_VALUE)
```

```
00114 #define MSI_VALUE    ((uint32_t)48000000U)
00115 #endif /* MSI_VALUE */
00121 #if !defined  (HSI_VALUE)
00122 #define HSI_VALUE    ((uint32_t)16000000U)
00123 #endif /* HSI_VALUE */
00124
00132 #if !defined  (HSI48_VALUE)
00133 #define HSI48_VALUE   ((uint32_t)48000000U)
00135 #endif /* HSI48_VALUE */
00136
00140 #if !defined  (LSI_VALUE)
00141 #define LSI_VALUE  32000U
00142 #endif /* LSI_VALUE */
00150 #if !defined  (LSE_VALUE)
00151 #define LSE_VALUE    32768U
00152 #endif /* LSE_VALUE */
00153
00154 #if !defined  (LSE_STARTUP_TIMEOUT)
00155 #define LSE_STARTUP_TIMEOUT    5000U
00156 #endif /* HSE_STARTUP_TIMEOUT */
00157
00163 #if !defined  (EXTERNAL_SAI1_CLOCK_VALUE)
00164 #define EXTERNAL_SAI1_CLOCK_VALUE    48000U
00165 #endif /* EXTERNAL_SAI1_CLOCK_VALUE */
00166
00172 #if !defined  (EXTERNAL_SAI2_CLOCK_VALUE)
00173 #define EXTERNAL_SAI2_CLOCK_VALUE    48000U
00174 #endif /* EXTERNAL_SAI2_CLOCK_VALUE */
00175
00176 /* Tip: To avoid modifying this file each time you need to use different HSE,
00177    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00178
00179 /* ########################### System Configuration ######################### */
00184 #define  VDD_VALUE            3300U
00185 #define  TICK_INT_PRIORITY            0U
00186 #define  USE_RTOS                     0U
00187 #define  PREFETCH_ENABLE              0U
00188 #define  INSTRUCTION_CACHE_ENABLE     1U
00189 #define  DATA_CACHE_ENABLE            1U
00190
00191 /* ########################### Assert Selection ############################## */
00196 /* #define USE_FULL_ASSERT    1U */
00197
00198 /* ################### Register callback feature configuration ############### */
00208 #define USE_HAL_ADC_REGISTER_CALLBACKS        0U
00209 #define USE_HAL_CAN_REGISTER_CALLBACKS        0U
00210 #define USE_HAL_COMP_REGISTER_CALLBACKS       0U
00211 #define USE_HAL_CRYP_REGISTER_CALLBACKS       0U
00212 #define USE_HAL_DAC_REGISTER_CALLBACKS        0U
00213 #define USE_HAL_DCMI_REGISTER_CALLBACKS       0U
00214 #define USE_HAL_DFSDM_REGISTER_CALLBACKS      0U
00215 #define USE_HAL_DMA2D_REGISTER_CALLBACKS      0U
00216 #define USE_HAL_DSI_REGISTER_CALLBACKS        0U
00217 #define USE_HAL_GFXMMU_REGISTER_CALLBACKS     0U
00218 #define USE_HAL_HASH_REGISTER_CALLBACKS       0U
00219 #define USE_HAL_HCD_REGISTER_CALLBACKS        0U
00220 #define USE_HAL_I2C_REGISTER_CALLBACKS        0U
00221 #define USE_HAL_IRDA_REGISTER_CALLBACKS       0U
00222 #define USE_HAL_LPTIM_REGISTER_CALLBACKS      0U
00223 #define USE_HAL_LTDC_REGISTER_CALLBACKS       0U
00224 #define USE_HAL_MMC_REGISTER_CALLBACKS        0U
00225 #define USE_HAL_OPAMP_REGISTER_CALLBACKS      0U
00226 #define USE_HAL_OSPI_REGISTER_CALLBACKS       0U
00227 #define USE_HAL_PCD_REGISTER_CALLBACKS        0U
00228 #define USE_HAL_QSPI_REGISTER_CALLBACKS       0U
00229 #define USE_HAL_RNG_REGISTER_CALLBACKS        0U
00230 #define USE_HAL_RTC_REGISTER_CALLBACKS        0U
00231 #define USE_HAL_SAI_REGISTER_CALLBACKS        0U
00232 #define USE_HAL_SD_REGISTER_CALLBACKS         0U
00233 #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS  0U
00234 #define USE_HAL_SMBUS_REGISTER_CALLBACKS      0U
00235 #define USE_HAL_SPI_REGISTER_CALLBACKS        0U
00236 #define USE_HAL_SWPMI_REGISTER_CALLBACKS      0U
00237 #define USE_HAL_TIM_REGISTER_CALLBACKS        0U
00238 #define USE_HAL_TSC_REGISTER_CALLBACKS        0U
00239 #define USE_HAL_UART_REGISTER_CALLBACKS       0U
00240 #define USE_HAL_USART_REGISTER_CALLBACKS      0U
00241 #define USE_HAL_WWDG_REGISTER_CALLBACKS       0U
00242
00243 /* ################### SPI peripheral configuration ########################## */
00244
00245 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00246  * Activated: CRC code is present inside driver
00247  * Deactivated: CRC code cleaned from driver
00248  */
00249
00250 #define USE_SPI_CRC                   0U
```

```
00251
00252 /* Includes ------------------------------------------------------------------*/
00257 #ifdef HAL_RCC_MODULE_ENABLED
00258 #include "stm32l4xx_hal_rcc.h"
00259 #endif /* HAL_RCC_MODULE_ENABLED */
00260
00261 #ifdef HAL_GPIO_MODULE_ENABLED
00262 #include "stm32l4xx_hal_gpio.h"
00263 #endif /* HAL_GPIO_MODULE_ENABLED */
00264
00265 #ifdef HAL_DMA_MODULE_ENABLED
00266 #include "stm32l4xx_hal_dma.h"
00267 #endif /* HAL_DMA_MODULE_ENABLED */
00268
00269 #ifdef HAL_DFSDM_MODULE_ENABLED
00270 #include "stm32l4xx_hal_dfsdm.h"
00271 #endif /* HAL_DFSDM_MODULE_ENABLED */
00272
00273 #ifdef HAL_CORTEX_MODULE_ENABLED
00274 #include "stm32l4xx_hal_cortex.h"
00275 #endif /* HAL_CORTEX_MODULE_ENABLED */
00276
00277 #ifdef HAL_ADC_MODULE_ENABLED
00278 #include "stm32l4xx_hal_adc.h"
00279 #endif /* HAL_ADC_MODULE_ENABLED */
00280
00281 #ifdef HAL_CAN_MODULE_ENABLED
00282 #include "stm32l4xx_hal_can.h"
00283 #endif /* HAL_CAN_MODULE_ENABLED */
00284
00285 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00286 #include "Legacy/stm32l4xx_hal_can_legacy.h"
00287 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00288
00289 #ifdef HAL_COMP_MODULE_ENABLED
00290 #include "stm32l4xx_hal_comp.h"
00291 #endif /* HAL_COMP_MODULE_ENABLED */
00292
00293 #ifdef HAL_CRC_MODULE_ENABLED
00294 #include "stm32l4xx_hal_crc.h"
00295 #endif /* HAL_CRC_MODULE_ENABLED */
00296
00297 #ifdef HAL_CRYP_MODULE_ENABLED
00298 #include "stm32l4xx_hal_cryp.h"
00299 #endif /* HAL_CRYP_MODULE_ENABLED */
00300
00301 #ifdef HAL_DAC_MODULE_ENABLED
00302 #include "stm32l4xx_hal_dac.h"
00303 #endif /* HAL_DAC_MODULE_ENABLED */
00304
00305 #ifdef HAL_DCMI_MODULE_ENABLED
00306 #include "stm32l4xx_hal_dcmi.h"
00307 #endif /* HAL_DCMI_MODULE_ENABLED */
00308
00309 #ifdef HAL_DMA2D_MODULE_ENABLED
00310 #include "stm32l4xx_hal_dma2d.h"
00311 #endif /* HAL_DMA2D_MODULE_ENABLED */
00312
00313 #ifdef HAL_DSI_MODULE_ENABLED
00314 #include "stm32l4xx_hal_dsi.h"
00315 #endif /* HAL_DSI_MODULE_ENABLED */
00316
00317 #ifdef HAL_EXTI_MODULE_ENABLED
00318 #include "stm32l4xx_hal_exti.h"
00319 #endif /* HAL_EXTI_MODULE_ENABLED */
00320
00321 #ifdef HAL_GFXMMU_MODULE_ENABLED
00322 #include "stm32l4xx_hal_gfxmmu.h"
00323 #endif /* HAL_GFXMMU_MODULE_ENABLED */
00324
00325 #ifdef HAL_FIREWALL_MODULE_ENABLED
00326 #include "stm32l4xx_hal_firewall.h"
00327 #endif /* HAL_FIREWALL_MODULE_ENABLED */
00328
00329 #ifdef HAL_FLASH_MODULE_ENABLED
00330 #include "stm32l4xx_hal_flash.h"
00331 #endif /* HAL_FLASH_MODULE_ENABLED */
00332
00333 #ifdef HAL_HASH_MODULE_ENABLED
00334 #include "stm32l4xx_hal_hash.h"
00335 #endif /* HAL_HASH_MODULE_ENABLED */
00336
00337 #ifdef HAL_HCD_MODULE_ENABLED
00338 #include "stm32l4xx_hal_hcd.h"
00339 #endif /* HAL_HCD_MODULE_ENABLED */
00340
00341 #ifdef HAL_I2C_MODULE_ENABLED
```

```
00342 #include "stm32l4xx_hal_i2c.h"
00343 #endif /* HAL_I2C_MODULE_ENABLED */
00344
00345 #ifdef HAL_IRDA_MODULE_ENABLED
00346 #include "stm32l4xx_hal_irda.h"
00347 #endif /* HAL_IRDA_MODULE_ENABLED */
00348
00349 #ifdef HAL_IWDG_MODULE_ENABLED
00350 #include "stm32l4xx_hal_iwdg.h"
00351 #endif /* HAL_IWDG_MODULE_ENABLED */
00352
00353 #ifdef HAL_LCD_MODULE_ENABLED
00354 #include "stm32l4xx_hal_lcd.h"
00355 #endif /* HAL_LCD_MODULE_ENABLED */
00356
00357 #ifdef HAL_LPTIM_MODULE_ENABLED
00358 #include "stm32l4xx_hal_lptim.h"
00359 #endif /* HAL_LPTIM_MODULE_ENABLED */
00360
00361 #ifdef HAL_LTDC_MODULE_ENABLED
00362 #include "stm32l4xx_hal_ltdc.h"
00363 #endif /* HAL_LTDC_MODULE_ENABLED */
00364
00365 #ifdef HAL_MMC_MODULE_ENABLED
00366 #include "stm32l4xx_hal_mmc.h"
00367 #endif /* HAL_MMC_MODULE_ENABLED */
00368
00369 #ifdef HAL_NAND_MODULE_ENABLED
00370 #include "stm32l4xx_hal_nand.h"
00371 #endif /* HAL_NAND_MODULE_ENABLED */
00372
00373 #ifdef HAL_NOR_MODULE_ENABLED
00374 #include "stm32l4xx_hal_nor.h"
00375 #endif /* HAL_NOR_MODULE_ENABLED */
00376
00377 #ifdef HAL_OPAMP_MODULE_ENABLED
00378 #include "stm32l4xx_hal_opamp.h"
00379 #endif /* HAL_OPAMP_MODULE_ENABLED */
00380
00381 #ifdef HAL_OSPI_MODULE_ENABLED
00382 #include "stm32l4xx_hal_ospi.h"
00383 #endif /* HAL_OSPI_MODULE_ENABLED */
00384
00385 #ifdef HAL_PCD_MODULE_ENABLED
00386 #include "stm32l4xx_hal_pcd.h"
00387 #endif /* HAL_PCD_MODULE_ENABLED */
00388
00389 #ifdef HAL_PKA_MODULE_ENABLED
00390 #include "stm32l4xx_hal_pka.h"
00391 #endif /* HAL_PKA_MODULE_ENABLED */
00392
00393 #ifdef HAL_PSSI_MODULE_ENABLED
00394 #include "stm32l4xx_hal_pssi.h"
00395 #endif /* HAL_PSSI_MODULE_ENABLED */
00396
00397 #ifdef HAL_PWR_MODULE_ENABLED
00398 #include "stm32l4xx_hal_pwr.h"
00399 #endif /* HAL_PWR_MODULE_ENABLED */
00400
00401 #ifdef HAL_QSPI_MODULE_ENABLED
00402 #include "stm32l4xx_hal_qspi.h"
00403 #endif /* HAL_QSPI_MODULE_ENABLED */
00404
00405 #ifdef HAL_RNG_MODULE_ENABLED
00406 #include "stm32l4xx_hal_rng.h"
00407 #endif /* HAL_RNG_MODULE_ENABLED */
00408
00409 #ifdef HAL_RTC_MODULE_ENABLED
00410 #include "stm32l4xx_hal_rtc.h"
00411 #endif /* HAL_RTC_MODULE_ENABLED */
00412
00413 #ifdef HAL_SAI_MODULE_ENABLED
00414 #include "stm32l4xx_hal_sai.h"
00415 #endif /* HAL_SAI_MODULE_ENABLED */
00416
00417 #ifdef HAL_SD_MODULE_ENABLED
00418 #include "stm32l4xx_hal_sd.h"
00419 #endif /* HAL_SD_MODULE_ENABLED */
00420
00421 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00422 #include "stm32l4xx_hal_smartcard.h"
00423 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00424
00425 #ifdef HAL_SMBUS_MODULE_ENABLED
00426 #include "stm32l4xx_hal_smbus.h"
00427 #endif /* HAL_SMBUS_MODULE_ENABLED */
00428
```

```
00429 #ifdef HAL_SPI_MODULE_ENABLED
00430 #include "stm32l4xx_hal_spi.h"
00431 #endif /* HAL_SPI_MODULE_ENABLED */
00432
00433 #ifdef HAL_SRAM_MODULE_ENABLED
00434 #include "stm32l4xx_hal_sram.h"
00435 #endif /* HAL_SRAM_MODULE_ENABLED */
00436
00437 #ifdef HAL_SWPMI_MODULE_ENABLED
00438 #include "stm32l4xx_hal_swpmi.h"
00439 #endif /* HAL_SWPMI_MODULE_ENABLED */
00440
00441 #ifdef HAL_TIM_MODULE_ENABLED
00442 #include "stm32l4xx_hal_tim.h"
00443 #endif /* HAL_TIM_MODULE_ENABLED */
00444
00445 #ifdef HAL_TSC_MODULE_ENABLED
00446 #include "stm32l4xx_hal_tsc.h"
00447 #endif /* HAL_TSC_MODULE_ENABLED */
00448
00449 #ifdef HAL_UART_MODULE_ENABLED
00450 #include "stm32l4xx_hal_uart.h"
00451 #endif /* HAL_UART_MODULE_ENABLED */
00452
00453 #ifdef HAL_USART_MODULE_ENABLED
00454 #include "stm32l4xx_hal_usart.h"
00455 #endif /* HAL_USART_MODULE_ENABLED */
00456
00457 #ifdef HAL_WWDG_MODULE_ENABLED
00458 #include "stm32l4xx_hal_wwdg.h"
00459 #endif /* HAL_WWDG_MODULE_ENABLED */
00460
00461 /* Exported macro ------------------------------------------------------------*/
00462 #ifdef  USE_FULL_ASSERT
00471 #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00472 /* Exported functions ------------------------------------------------------- */
00473 void assert_failed ( uint8_t* file, uint32_t line );
00474 #else
00475 #define assert_param(expr) ((void)0U)
00476 #endif /* USE_FULL_ASSERT */
00477
00478 #ifdef __cplusplus
00479 }
00480
00481 #endif
00482
00483 #endif /* INC_stm32l4xx_hal_confh */
```

4.21 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.

4.21.1 Enumeration Type Documentation

4.21.1.1 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

4.21.2 Function Documentation

4.21.2.1 GetTemperature()

```
int32_t GetTemperature (
            void )
```

Get temperature from thermistor.

Parameters

out	*temperature*	level in dC

4.21.2.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
            void )
```

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

Parameters

out	*Current*	temperature range

4.22 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
00033 typedef enum
00034 {
00035     TempCool = 0,
00036     TempWarm = 1,
00037     TempHot = 2
00038 } TemperatureRange_e;
00039
00045 int32_t GetTemperature ( void );
00046
00056 TemperatureRange_e GetTemperatureRange ( void );
00057
00058 #endif /* INC_temperature_handlerh */
```

4.23 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.

4.23.1 Enumeration Type Documentation

4.23.1.1 VoltageRange_e

```
enum VoltageRange_e
```

Voltage Range Enum

Enumerator

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

4.23.2 Function Documentation

4.23.2.1 GetVoltage()

```
uint16_t GetVoltage (
            void )
```

Get voltage from voltmeter.

Parameters

out	*voltage*	level in dV

4.23.2.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
            void )
```

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

Parameters

out	*Current*	voltage range

4.24 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
00031 typedef enum
00032 {
00033     VoltageNormal = 0,
00034     VoltageLow = 1,
00035     VoltageHigh = 2,
00036     VoltageErrorLow = 3,
00037     VoltageErrorHigh = 4
00038 } VoltageRange_e;
00039
00045 uint16_t GetVoltage ( void );
00046
00057 VoltageRange_e GetVoltageRange ( void );
00058
00059 #endif /* INC_voltage_handlerh */
```

4.25 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 IsTogglePressed (void)

  Return state of toggle button.
- GPIO_PinState IsDimPressed (void)

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

  Return state of brighten button.

4.25.1 Function Documentation

4.25.1.1 IsBrightPressed()

```
GPIO_PinState IsBrightPressed (
            void )
```

Return state of brighten button.

Parameters

out	*Bright*	Pin State, pressed or not

4.25.1.2 IsDimPressed()

```
GPIO_PinState IsDimPressed (
            void )
```

Return state of dim button.

Parameters

out	*Dim*	Pin State, pressed or not

4.25.1.3 IsTogglePressed()

```
GPIO_PinState IsTogglePressed (
            void )
```

Return state of toggle button.

Parameters

out	*Toogle*	Pin State, pressed or not

4.26 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

4.26.1 Function Documentation

4.26.1.1 GetCurrent()

```
uint16_t GetCurrent (
            void )
```

Get current from ammeter.

Parameters

out	*current*	level in dA

4.26.1.2 GetCurrentRange()

```
CurrentRange_e GetCurrentRange (
            void )
```

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

Parameters

out	*Current*	current range

4.26.2 Variable Documentation

4.26.2.1 CurrentErrorThreshold_dA

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

High Current Error Level in dA

4.26.2.2 CurrentHighThreshold_dA

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

High Current Level in dA

4.26.2.3 dAmpsToRaw

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

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

4.26.2.4 RawTodAmps

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

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

4.27 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

4.27.1 Function Documentation

4.27.1.1 BrightnessDelay()

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

Returns a delay value for a brightness level.

Parameters

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

4.27.1.2 DelayHit()

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

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

Parameters

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

4.27.1.3 RestartDelayCounter()

```
void RestartDelayCounter (
            void )
```

Restart the delay counter.

4.27.1.4 StartDelayCounter()

```
void StartDelayCounter (
            void )
```

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

4.27.2 Variable Documentation

4.27.2.1 Tim2ClkKhz

```
const float Tim2ClkKhz
```

Initial value:

```
= ( 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

4.28 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.

4.28.1 Macro Definition Documentation

4.28.1.1 TADD

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

4.28.1.2 TLEN

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

4.28.2 Function Documentation

4.28.2.1 framChipSelect()

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

4.28.2.2 framReadMemory()

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

4.28.2.3 framReadSr()

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

4.28.2.4 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

4.28.2.5 framWriteDisable()

```
void framWriteDisable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.28.2.6 framWriteEnable()

```
void framWriteEnable (
            void )
```

This routine resets the write enable latch.

Parameters

out	*none*	

4.28.2.7 framWriteMemory()

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

4.28.2.8 framWriteProtect()

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

4.28.2.9 framWriteSr()

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

4.29 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.

4.29.1 Function Documentation

4.29.1.1 LogNumber()

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

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

Parameters

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

4.29.1.2 LogString()

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

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

Parameters

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

4.29.1.3 ReadLog()

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

Reads the log at a given address and size.

Parameters

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

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

: Main program body

```
#include "main.h"
#include "usb_device.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.
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

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

4.30.1 Detailed Description

: Main program body

Attention

4.30.2 Macro Definition Documentation

4.30.2.1 AVG_STEP_DIFF_MS

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

4.30.2.2 AVG_STEP_TIME_MS

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

4.30.2.3 LOWER_STEP_TIME_MS

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

4.30.2.4 LOWER_SWEEP_TIME_MS

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

4.30.2.5 TOTAL_SWEEP_TIME_MS

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

4.30.2.6 UPPER_STEP_TIME_MS

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

4.30.2.7 UPPER_SWEEP_TIME_MS

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

4.30.3 Function Documentation

4.30.3.1 Error_Handler()

```
void Error_Handler (
            void )
```

This function is executed in case of error occurrence.

Return values

None	

4.30.3.2 main()

```
int main (
            void )
```

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

Return values

int	

4.30.3.3 SystemClock_Config()

```
void SystemClock_Config (
            void )
```

System Clock Configuration.

Return values

None	

4.30.4 Variable Documentation

4.30.4.1 HighStepTimeMs

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

4.30.4.2 LowStepTimeMs

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

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

4.32 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, uint8_t isIr)

  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, uint8_t isIr)

  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 (uint8_t isIr)

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

  turn off PWM
- int8_t GetBrightness (uint8_t isIr)

  Return Brightness variable.
- void SetBrightness (int8_t brightness, uint8_t isIr)

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

  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
- const float MinPw = (0)
- const float MaxPw = (PW_PERIOD)
- const float WarmPwmRatio = (0.90f)
- const float HotPwmRatio = (0.50f)

4.32.1 Macro Definition Documentation

4.32.1.1 PW_PERIOD

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

4.32.2 Function Documentation

4.32.2.1 DecreaseBrightness()

```
void DecreaseBrightness (
            uint8_t button_held,
            uint8_t isIr)
```

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

Parameters

in	*button_held*	If the button is being held (decrements by 3 if so)
in	*isIr*	Are we controlling IR or Visible LEDs

4.32.2.2 GetBrightness()

```
int8_t GetBrightness (
            uint8_t isIr)
```

Return Brightness variable.

Parameters

in	*isIr*	Are we controlling IR or Visible LEDs
out	*Current*	LED brightness level

4.32.2.3 GetPwm()

```
uint8_t GetPwm (
            uint8_t isIr)
```

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

Parameters

in	*isIr*	Are we controlling IR or Visible LEDs
out	*Current*	PWM value

4.32.2.4 IncreaseBrightness()

```
void IncreaseBrightness (
            uint8_t button_held,
            uint8_t isIr)
```

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

Parameters

in	*button_held*	If the button is being held (increments by 3 if so)
in	*isIr*	Are we controlling IR or Visible LEDs

4.32.2.5 InitPwm()

```
void InitPwm (
            void )
```

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

4.32.2.6 SetBrightness()

```
void SetBrightness (
            int8_t brightness,
            uint8_t isIr)
```

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

Parameters

in	*brightness*	Brightness to set
in	*isIr*	Are we controlling IR or Visible LEDs

4.32.2.7 SetPwm()

```
void SetPwm (
            uint8_t isIr)
```

set PWM based on pwm value

Parameters

in	*is_ir*	Are we controlling IR or Visible LEDs

4.32.2.8 TurnOffPwm()

```
void TurnOffPwm (
            void )
```

turn off PWM

4.32.3 Variable Documentation

4.32.3.1 HalfBrightness

```
const uint8_t HalfBrightness
```

Initial value:

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

Half Brightness Step (24)

4.32.3.2 HotPwmRatio

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

Hot thermal state pwm constant

4.32.3.3 MaxBrightness

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

Max Brightness Step (49)

4.32.3.4 MaxPw

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

Max pulse width value (PW_PERIOD(255))

4.32.3.5 MinBrightness

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

Min Brightness Step (0)

4.32.3.6 MinPw

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

Min pulse width value (0)

4.32.3.7 WarmPwmRatio

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

Warm thermal state pwm constant

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

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

Functions

- GPIO_PinState ReadTogglePin (void)
  Reads toggle pin value.
- 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.

4.33.1 Function Documentation

4.33.1.1 disableChipSelect()

```
void disableChipSelect (
            void )
```

Disables SPI chip select line (active low)

4.33.1.2 DisablePWM1()

```
void DisablePWM1 (
            void )
```

Disables Timer 1.

4.33.1.3 disableWriteProtect()

```
void disableWriteProtect (
            void )
```

Disables SPI write protect line (active high)

4.33.1.4 enableChipSelect()

```
void enableChipSelect (
            void )
```

Enables SPI chip select line (active low)

4.33.1.5 EnablePWM1()

```
void EnablePWM1 (
            void )
```

Enables Timer 1.

4.33.1.6 enableWriteProtect()

```
void enableWriteProtect (
            void )
```

Enables SPI write protect line (active high)

4.33.1.7 GetCurrentValue()

```
int16_t GetCurrentValue (
            void )
```

Returns raw ADC value from ammeter.

Parameters

out	*Ammeter*	raw ADC value

4.33.1.8 GetThermistorValue()

```
int16_t GetThermistorValue (
            void )
```

Returns raw ADC value from thermistor.

Parameters

out	*Thermistor*	raw ADC value

4.33.1.9 GetTIM2Cnt()

```
uint32_t GetTIM2Cnt (
            void )
```

Returns value in the Timer 2 counter.

Parameters

out	*Value*	of Timer 2 counter

4.33.1.10 GetVoltageValue()

```
int16_t GetVoltageValue (
            void )
```

Returns raw ADC value from voltmeter.

Parameters

out	*Voltmeter*	raw ADC value

4.33.1.11 ReadBrightPin()

```
GPIO_PinState ReadBrightPin (
            void )
```

Reads bright pin value.

Parameters

out	*Bright*	pin state

4.33.1.12 ReadDimPin()

```
GPIO_PinState ReadDimPin (
            void )
```

Reads dim pin value.

Parameters

out	*Dim*	pin state

4.33.1.13 ReadTogglePin()

```
GPIO_PinState ReadTogglePin (
            void )
```

Reads toggle pin value.

Parameters

out	*Toggle*	pin state

4.33.1.14 receiveData()

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

Gets data from SPI lines.

Parameters

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

4.33.1.15 RestartTIM2()

```
void RestartTIM2 (
            void )
```

Resets Timer 2 counter to zero.

4.33.1.16 sendUARTChar()

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

Sends character via UART line.

Parameters

in	*c*	Character to send via UART

4.33.1.17 SetPW11()

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

Sets PWM Timer 1 Channel 1 value.

Parameters

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

4.33.1.18 StartPWM11()

```
void StartPWM11 (
            void )
```

Starts PWM Timer 1 Channel 1 output.

4.33.1.19 StartTIM2()

```
void StartTIM2 (
            void )
```

Starts Timer 2 counter.

4.33.1.20 StopPWM11()

```
void StopPWM11 (
            void )
```

Stops PWM Timer 1 Channel 1 output.

4.33.1.21 transferData()

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

Sends data via SPI lines.

Parameters

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

4.34 C:/Users/agreen/Documents/Projects/Aveo/CH-53K_LED_Controller/Core/Src/stm32l4xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

Functions

- void HAL_MspInit (void)

4.34.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

4.34.2 Function Documentation

4.34.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void )
```

Initializes the Global MSP.

4.35 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)

4.35.1 Function Documentation

4.35.1.1 GetTemperature()

```
int32_t GetTemperature (
            void )
```

Get temperature from thermistor.

Parameters

out	*temperature*	level in dC

4.35.1.2 GetTemperatureRange()

```
TemperatureRange_e GetTemperatureRange (
            void )
```

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

Parameters

out	*Current*	temperature range

4.35.2 Variable Documentation

4.35.2.1 CoolingThreshold1_dC

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

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

4.35.2.2 CoolingThreshold2_dC

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

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

4.35.2.3 dCelciusToThermistor

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

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

4.35.2.4 HeatingThreshold1_dC

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

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

4.35.2.5 HeatingThreshold2_dC

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

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

4.35.2.6 ThermistorTodCelcius

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

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

4.36 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

4.36.1 Function Documentation

4.36.1.1 GetVoltage()

```
uint16_t GetVoltage (
            void )
```

Get voltage from voltmeter.

Parameters

out	*voltage*	level in dV

4.36.1.2 GetVoltageRange()

```
VoltageRange_e GetVoltageRange (
            void )
```

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

Parameters

out	*Current*	voltage range

4.36.2 Variable Documentation

4.36.2.1 dVoltsToRaw

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

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

4.36.2.2 RawTodVolts

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

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

4.36.2.3 VoltageErrorHighThreshold_dV

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

High Voltage Error Level in dV

4.36.2.4 VoltageErrorLowThreshold_dV

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

Low Voltage Error Level in dV

4.36.2.5 VoltageHighThreshold_dV

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

High Voltage Level in dV

4.36.2.6 VoltageLowThreshold_dV

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

Low Voltage Level in dV

Index