My Project

AUTHOR
Version
Tue Jan 28 2020

Table of Contents

Table of contents

Module Index

Modules

Here is a list of all modules:

File Index

File List

Here is a list of all documented files with brief descriptions:

Module Documentation

CMSIS

Modules

- **Stm32l4xx_system**

Detailed Description

Stm32l4xx_system

Modules

- **STM32L4xx_System_Private_Includes**
- **STM32L4xx_System_Private_TypesDefinitions**
- **STM32L4xx_System_Private_Defines**
- **STM32L4xx_System_Private_Macros**
- **STM32L4xx_System_Private_Variables**
- **STM32L4xx_System_Private_FunctionPrototypes**
- **STM32L4xx_System_Private_Functions**

Detailed Description

STM32L4xx_System_Private_Includes

Macros

- #define **HSE_VALUE** 8000000U
- #define **MSI_VALUE** 4000000U
- #define **HSI_VALUE** 16000000U

Detailed Description

Macro Definition Documentation

#define HSE_VALUE 8000000U

Value of the External oscillator in Hz

#define HSI_VALUE 16000000U

Value of the Internal oscillator in Hz

#define MSI_VALUE   4000000U

Value of the Internal oscillator in Hz

STM32L4xx_System_Private_TypesDefinitions

STM32L4xx_System_Private_Defines

Macros

- #define **VECT_TAB_OFFSET**   0x00

Detailed Description

Macro Definition Documentation

#define VECT_TAB_OFFSET   0x00

< Uncomment the following line if you need to relocate your vector Table in Internal SRAM. Vector Table base offset field. This value must be a multiple of 0x200.

STM32L4xx_System_Private_Macros

STM32L4xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 4000000U
- const uint8_t **AHBPrescTable** [16] = {0U, 0U, 0U, 0U, 0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U, 6U, 7U, 8U, 9U}
- const uint8_t **APBPrescTable** [8] = {0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U}
- const uint32_t **MSIRangeTable** [12]

Detailed Description

Variable Documentation

const uint32_t MSIRangeTable[12]

```
Initial value:= {100000U,  200000U,  400000U,  800000U,  1000000U,  2000000U,
                             4000000U, 8000000U, 16000000U, 24000000U, 32000000U,
48000000U}
```

STM32L4xx_System_Private_FunctionPrototypes

STM32L4xx_System_Private_Functions

Functions

- void **SystemInit** (void)
  Setup the microcontroller system.

- void **SystemCoreClockUpdate** (void)
  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Detailed Description

Function Documentation

void SystemCoreClockUpdate (void)

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is MSI, SystemCoreClock will contain the **MSI_VALUE(*)**
- If SYSCLK source is HSI, SystemCoreClock will contain the **HSI_VALUE(**)**
- If SYSCLK source is HSE, SystemCoreClock will contain the **HSE_VALUE(***)**
- If SYSCLK source is PLL, SystemCoreClock will contain the **HSE_VALUE(***)** or **HSI_VALUE(*)** or **MSI_VALUE(*)** multiplied/divided by the PLL factors.

(*) MSI_VALUE is a constant defined in stm32l4xx_hal.h file (default value 4 MHz) but the real value may vary depending on the variations in voltage and temperature.

(**) HSI_VALUE is a constant defined in stm32l4xx_hal.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.

(***) HSE_VALUE is a constant defined in stm32l4xx_hal.h file (default value 8 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

void SystemInit (void)

Setup the microcontroller system.

Parameters

None	

Return values

None	

File Documentation

Src/apds_9930.c File Reference

```
#include "apds_9930.h"
```

Functions

- int **APDS_init** (I2C_HandleTypeDef *i2c)
- uint16_t **ALS_readCh0** (I2C_HandleTypeDef *i2c)
- uint16_t **ALS_readCh1** (I2C_HandleTypeDef *i2c)
- float **scale_results** (uint16_t result0, uint16_t result1)
- int **APDS_enable** (I2C_HandleTypeDef *i2c)
- int **APDS_disable** (I2C_HandleTypeDef *i2c)
- int **set_ATIME** (I2C_HandleTypeDef *i2c, uint8_t time)
- int **set_PTIME** (I2C_HandleTypeDef *i2c)
- int **set_WTIME** (I2C_HandleTypeDef *i2c, uint8_t time)
- int **set_ALS_thresholds** (I2C_HandleTypeDef *i2c, uint8_t low1, uint8_t low2, uint8_t high1, uint8_t high2)
- int **set_PROX_thresholds** (I2C_HandleTypeDef *i2c, uint8_t low1, uint8_t low2, uint8_t high1, uint8_t high2)
- int **set_PERS** (I2C_HandleTypeDef *i2c, uint8_t PPERS, uint8_t APERS)
- int **set_AGL** (I2C_HandleTypeDef *i2c, int enable)
- int **set_WLONG** (I2C_HandleTypeDef *i2c, int enable)
- int **set_PDL** (I2C_HandleTypeDef *i2c, int enable)
- int **set_PPULSE** (I2C_HandleTypeDef *i2c, uint8_t pulse)
- int **set_Control** (I2C_HandleTypeDef *i2c, uint8_t PDRIVE, uint8_t PDIODE, uint8_t PGAIN, uint8_t AGAIN)
- int **set_POFFSET** (I2C_HandleTypeDef *i2c, uint8_t offset)
- uint8_t **read_status** (I2C_HandleTypeDef *i2c)

Detailed Description

apds_9930.c

Created on: Jan 2, 2020

Author

Iga

Function Documentation

uint16_t ALS_readCh0 (I2C_HandleTypeDef * *i2c*)

Reads the value in register for channel Ch0 - measurement result

Parameters

i2c	pointer to I2C handle

Returns

measured value

uint16_t ALS_readCh1 (I2C_HandleTypeDef * *i2c*)

Reads the value in register for channel Ch01- measurement result

Parameters

i2c	pointer to I2C handle

Returns

measured value

int APDS_disable (I2C_HandleTypeDef * *i2c*)

Disables the APDS sensor

Parameters

i2c	pointer to I2C handle

Returns

ifSuccessful

int APDS_enable (I2C_HandleTypeDef * *i2c*)

Enables the APDS sensor in the ambient light mode

Parameters

i2c	pointer to I2C handle

Returns

ifSuccessful

int APDS_init (I2C_HandleTypeDef * *i2c*)

Initializes APDS light sensor with default values

Parameters

i2c	pointer to I2C handle

Returns

ifSuccessful

float scale_results (uint16_t *result0*, uint16_t *result1*)

Function scales measured values to the range <0;1>, calculates the average and the illuminance to set

Parameters

result0	from channel 0
result1	from channel 1

Returns

illuminance to set

int set_ATIME (I2C_HandleTypeDef * *i2c*, uint8_t *time*)

Sets the integration time for ambient light in APDS

Parameters

i2c	pointer to I2C handle

Returns

ifSuccessful

int set_PTIME (I2C_HandleTypeDef * *i2c*)

Sets the integration time for proximity measurement in APDS

Parameters

i2c	pointer to I2C handle

Returns

ifSuccessful

int set_WTIME (I2C_HandleTypeDef * *i2c*, uint8_t *time*)

Sets the wait time in APDS

Parameters

i2c	pointer to I2C handle
time	to set

Returns

ifSuccessful

Src/hcsr04.c File Reference

```
#include "hcsr04.h"
```

Functions

- void **init_hcsr** ()
- int **get_init_signal_state** ()
- void **start_init_signal** ()
- void **init_singal_step** ()
- void **distance** ()

Variables

- GPIO_PinState **last**

Detailed Description

hcsr04.c

Created on: Jan 8, 2020

Author

Iga

Function Documentation

void distance ()

Measures the distance between an object and the sensor writes it in counter2 and after 5 measurements without an object in range stops the lamp

int get_init_signal_state ()

Getter for the value of the start signal

Returns

start state

void init_hcsr ()

Initializes the sensor

void init_singal_step ()

Generates the init signal for the sensor

void start_init_signal ()

Starts the init signal for the sensor

Src/main.c File Reference

: Main program body

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

Functions

- void **SystemClock_Config** (void)
  System Clock Configuration.

- int **main** (void)
  The application entry point.

- void **calculate_illuminance** (int mode)
- void **Error_Handler** (void)
  This function is executed in case of error occurrence.

Variables

- I2C_HandleTypeDef **hi2c1**
- SPI_HandleTypeDef **hspi1**
- TIM_HandleTypeDef **htim2**
- TIM_HandleTypeDef **htim3**
- UART_HandleTypeDef **huart2**

Detailed Description

: Main program body

Attention

Function Documentation

void Error_Handler (void)

This function is executed in case of error occurrence.

Return values

None	

int main (void)

The application entry point.

Return values

int	

void SystemClock_Config (void)

System Clock Configuration.

Return values

None	

Configure LSE Drive Capability

Initializes the CPU, AHB and APB busses clocks

Initializes the CPU, AHB and APB busses clocks

Configure the main internal regulator output voltage

Enable MSI Auto calibration

Src/RGB_LED.c File Reference

```
#include "RGB_LED.h"
```

Functions

- uint8_t **getR** ()
- uint8_t **getG** ()
- uint8_t **getB** ()
- void **initializeLEDs** (SPI_HandleTypeDef *hspiInit, float HInit, float SInit, float VInit)
- void **startRainbow** ()
- void **startColorPicker** (int destR, int destG, int destB, float time)
- void **startStrobotron** (float frequency)
- void **startIdle** ()
- void **stopIdle** ()
- void **toggleLEDsActive** ()
- void **setIlluminance** (float illuminance)
- void **setColorBit** (uint32_t color)
- void **HSVtoRGB** (float H, float S, float V)
- void **makeBufferFromHSV** (float H, float S, float V)
- void **makeBufferFromRGB** (uint8_t red, uint8_t green, uint8_t blue)
- void **sendBuffer** ()
- void **makeRainbowStep** ()
- void **makeColorPickerStep** ()
- void **makeStrobotronStep** ()
- void **makeIdleStep** ()
- bool **toggleLamp** ()
- void **setLampActivation** (bool state)
- int **makeLampStep** ()

Detailed Description

RGB_LED.c

Created on: 1 gru 2019

Author

Marcin

Function Documentation

uint8_t getB ()

Getter for value of blue color, which is displayed on the LEDs

Returns

blue color's value

uint8_t getG ()

Getter for value of green color, which is displayed on the LEDs.

Returns

green color's value

uint8_t getR ()

Getter for value of red color, which is displayed on the LEDs

Returns

red color's value

void HSVtoRGB (float *H*, float *S*, float *V*)

Function converts color from HSV format to RGB format and sets the result to global variables

Parameters

H	hue of the color in HSV format
S	saturation of the color in HSV format
V	value of the color in HSV format

void initializeLEDs (SPI_HandleTypeDef * *hspiInit*, float *HInit*, float *SInit*, float *VInit*)

Function sets needed variables and display initial color given in HSV format.

void makeBufferFromHSV (float *H*, float *S*, float *V*)

Function sets the SPI buffer with color given in HSV format

Parameters

H	hue of the color in HSV format
S	saturation of the color in HSV format
V	value of the color in HSV format

void makeBufferFromRGB (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Function sets the SPI buffer with color given in RGB format

Parameters

red	chosen value for red color
green	chosen value for green color
blue	chosen value for blue color

void makeColorPickerStep ()

Function makes one step in the color picker mode - checks, if the color was reached and if not, sets color closer to the goal value on the LEDs.

void makeIdleStep ()

Function makes one step in the idle mode - switches LEDs off, if they are still on.

int makeLampStep ()

Function makes one step of the RGB LED programme - checks, if conditions are achieved and if yes, sets color to the RGB LEDs.

void makeRainbowStep ()

Function makes one step in the rainbow mode - changes H and V values and sends new color to the driver

void makeStrobotronStep ()

Function makes one step in the strobotron mode - checks, if the time has passed and if yes, sets random color on the LEDs.

void sendBuffer ()

Function sends buffer to the LEDs' driver using SPI interface

void setColorBit (uint32_t *color*)

Function sets one bit (ZERO or ONE value) of the color in RGB format in the SPI buffer

Parameters

color	value of the color that is being set to the buffer

void setIlluminance (float *illuminance*)

Function sets illuminance in strobotron and rainbow mode to change lamp's brightness

Parameters

illuminance	value of illuminance between 0 and 1 used to change lamp's brightness

void setLampActivation (bool *state*)

Function sets lamp's actvation mode - false value stops lamp's work.

Parameters

state	new activation mode

void startColorPicker (int *destR*, int *destG*, int *destB*, float *time*)

Function starts the color picker mode, which changes displayed color to chosen one in given time.

Parameters

destR	chosen value for red color
destG	chosen value for green color
destB	chosen value for blue color
time	chosen time, in which LEDs will change their color

void startIdle ()

Function starts idle mode.

void startRainbow ()

Function starts the rainbow mode, which goes through every color on repeat

void startStrobotron (float *frequency*)

Function starts the strobotron mode, which changes displayed color to random one on repeat with given frequency.

Parameters

frequency	chosen frequency in which colors will change

void stopIdle ()

Function stops idle mode.

bool toggleLamp ()

Function changes lamp's activation mode - false value stops lamp's work.

Returns

new lamp activation mode

void toggleLEDsActive ()

Function stops currently displayed mode.

Src/stm32l4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32l4xx_it.h"
```

Functions

- void **NMI_Handler** (void)
  This function handles Non maskable interrupt.

- void **HardFault_Handler** (void)
  This function handles Hard fault interrupt.

- void **MemManage_Handler** (void)
  This function handles Memory management fault.

- void **BusFault_Handler** (void)
  This function handles Prefetch fault, memory access fault.

- void **UsageFault_Handler** (void)
  This function handles Undefined instruction or illegal state.

- void **SVC_Handler** (void)
  This function handles System service call via SWI instruction.

- void **DebugMon_Handler** (void)
  This function handles Debug monitor.

- void **PendSV_Handler** (void)
  This function handles Pendable request for system service.

- void **SysTick_Handler** (void)
  This function handles System tick timer.

- void **TIM2_IRQHandler** (void)
  This function handles TIM2 global interrupt.

- void **TIM3_IRQHandler** (void)
  This function handles TIM3 global interrupt.

- void **USART2_IRQHandler** (void)
  This function handles USART2 global interrupt.

- void **EXTI15_10_IRQHandler** (void)
  This function handles EXTI line[15:10] interrupts.

Variables

- int **prevMode** = IDLE
- int **mode** = IDLE
- int **APDScounter**
- int **cnt**
- TIM_HandleTypeDef **htim2**
- TIM_HandleTypeDef **htim3**
- UART_HandleTypeDef **huart2**

Detailed Description

Interrupt Service Routines.

Attention

© Copyright (c) 2019 STMicroelectronics. All rights reserved.

Src/system_stm32l4xx.c File Reference

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.
`#include "stm32l4xx.h"`

Macros

- #define **HSE_VALUE**   8000000U
- #define **MSI_VALUE**   4000000U
- #define **HSI_VALUE**   16000000U
- #define **VECT_TAB_OFFSET**   0x00

Functions

- void **SystemInit** (void)
  Setup the microcontroller system.

- void **SystemCoreClockUpdate** (void)
  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 4000000U
- const uint8_t **AHBPrescTable** [16] = {0U, 0U, 0U, 0U, 0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U, 6U, 7U, 8U, 9U}
- const uint8_t **APBPrescTable** [8] = {0U, 0U, 0U, 0U, 1U, 2U, 3U, 4U}
- const uint32_t **MSIRangeTable** [12]

Detailed Description

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

Author

MCD Application Team This file provides two functions and one global variable to be called from user application:

- **SystemInit()**: This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32l4xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- **SystemCoreClockUpdate()**: Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

After each device reset the MSI (4 MHz) is used as system clock source. Then **SystemInit()** function is called, in "startup_stm32l4xx.s" file, to configure the system clock before to branch to main program.

This file configures the system clock as follows:

System Clock source	**MSI**
SYSCLK(Hz)	**4000000**
HCLK(Hz)	**4000000**
AHB Prescaler	**1**
APB1 Prescaler	**1**
APB2 Prescaler	**1**
PLL_M	**1**
PLL_N	**8**
PLL_P	**7**
PLL_Q	**2**
PLL_R	**2**
PLLSAI1_P	**NA**
PLLSAI1_Q	**NA**
PLLSAI1_R	**NA**
PLLSAI2_P	**NA**

PLLSAI2_Q | NA

PLLSAI2_R | NA

Require 48MHz for USB OTG FS, | Disabled

SDIO and RNG clock |

===

Attention

© Copyright (c) 2017 STMicroelectronics. All rights reserved.

Index

INDEX