BASKETBALL LAUNCHER

Generated by Doxygen 1.10.0

Chapter 1

MAIN PAGE

PICTURE UPDATE NEEDED

1.1 MOTIVATION

The motivation for this project is to build a prototype for a full-scale basketball-playing robot. The project will start with a small-scale version to test the concept and gather insights for future improvements. This approach will provide an opportunity to develop customized PCBAs that can control the robot for its intended purposes.

1.2 OBJECTIVE

The objective of the project includes integrating the mechanical design with PCBA electronic component. This project will primarily use 3D-printed parts, along with shafts and bearings to support the robot's housing. The PCBAs will be designed using Fusion 360 software and manufactured by JLCPCB. The programming for the STM32 microcontroller will be done using the C language.

Note: The image below is provided solely for illustrative purposes and does not represent the actual project.

The project is divided into two main assemblies:

- The controller will use an STM32 BlackPill microcontroller with an IMU sensor attached to the user's hand. The controller will send signals via Bluetooth to the launcher, allowing the launcher to receive the data and spin according to the IMU sensor's input.
- The launcher will be the main assembly, receiving data from the controller and moving in 2 degrees of freedom (yaw and pitch angles). It will then shoot the basketball when the controller send launching flag. The launcher will use a customized PCBA centered around the STM32F401CEU6 chip, which will act as the core component to determine movement.

1.3 IDEATION DESIGN

The overall design of the two main assemblies in the project is sketched and listed below for reference. The sketch represents a rough idea of how all components are connected and communicate with each other. The PDF of the sketch is also attached with higher solution can be found in by access through the picture below

1.4 CAD MODELING

The design is inspired by the Peashooter from the Plants vs. Zombies mobile game. In terms of mechanical design, custom machined parts are limited to facilitate rapid prototyping and maintain focus on electronics and programming. Most custom parts will be 3D printed to save time and effort, except for bearings and shafts, which will be purchased from McMaster-Carr as off-the-shelf components. The CAD modeling phase of the project was completed early to allow the main focus to be on electronics and program implementation. The CAD figure below shows the final design of the launcher. The SolidWorks CAD model can access through the CAD sketch below.

1.5 LAUNCHER

The launcher will be the main assembly, receiving data from the controller and moving in 2 degrees of freedom (yaw and pitch angles). It will then shoot the basketball when the controller sends out signals. The launcher will use a customized PCBA centered around the STM32F401CEU6 chip, which will act as the core component to determine movement. The launcher includes a cycloidal gearbox and stepper motors with a planetary gearbox for precise movement control. Additionally, the launcher is equipped with brushless motors and 3 limit switches to ensure accurate positioning. More detail about the launcher Bill of Materials, design, source code can be found by access through the launcher picture below:

1.6 CONTROLLER

The controller will use an STM32 BlackPill microcontroller with an IMU sensor attached to the user's hand. It will send signals via Bluetooth to the launcher, allowing the launcher to receive the data and rotate according to the IMU sensor's input. The data transfer includes the velocity of the angles, and the communication uses a UART connection between the HC-05 Bluetooth module. The controller also features buttons and an indicator LED, all mounted on a 3D-printed handle bracket. More detail about the controller's Bill of Materials, design, source code can be found by access through the launcher picture below:

1.7 FUTURE IMPROVEMENT

Even though, the project is built and run appropriately according to the project's objectives. The following improvements can be considered for future version. Especially, for the upcoming full scale basketball launcher

- Improve the IMU to a 9 DOF sensor that can sense accurate absolute angle
- Improve stronger stepper driver that can allow faster and stronger rotation motion
- Increase moving angle of the launcher to more than 120°

1.8 YOUTUBE REFERENCE

1.9 REPOSITORY REFERENCE

You may find it more useful to read through the website exploring the source code. All code that will be referenced in this portfolio relate to project is accessible through:

NOTE: You may find the Controller's code with a small specification "_C" at the end of the file that stands for Controller code. Files' name and the files' header code need to be renamed without that term in case the projects are recreated. The inconvenience is due to the drawback of uploading codes in 2 projects to one GitHub Repository.

For example:

- **main_C.c** (p. 53) means the main code file for the controller. Needs to be changed to **main.c** (p. 99) before running in any IDE.

1.10 CONTACT INFO

Author: Vinh Vo

Email: vinhvo.career@gmail.com

LinkedIn: https://www.linkedin.com/in/vinhvo98/

Youtube: https://www.youtube.com/channel/UCh_4F4CJVqvAhHmCMTvIb-w

Major: MS Mechanical Engineer at Cal Poly San Luis Obispo

Date: May 15, 2024

Chapter 2

CONTROLLER

2.1 FINAL DESIGN

The controller will use an STM32 BlackPill microcontroller with an IMU sensor attached to the user's hand. It will send signals via Bluetooth to the launcher, allowing the launcher to receive the data and rotate according to the IMU sensor's input. The data transfer includes the velocity of the angles, and the communication uses a UART connection between the HC-05 Bluetooth module. The controller also features buttons and an indicator LED, all mounted on a 3D-printed handle bracket.

2.2 CONTROLLER PCBA

Below are the overall schematic and footprint of the PCBA that is used to integrate the STM32 BlackPill, mpu6050 IMU, HC-05, and LEDs together. The PDF of this schematic is also attached with higher solution can be found in the picture below

2.3 FINITE STATE MACHINE

State	Description	Actions	Transitions
STATE0	INIT - Attempt to connect to IMU	- Attempt to connect - Turn on LED1 if successful	- If unsuccessful, transition to **STATE 1** - If successful, transition to **STATE 2**
STATE1	ERROR- Failed to connecto IMU	- Toggle LED1 if failed to connect to the IMU	- Always transition to **STATE 0**
STATE2	IMU - Read data from the IMU	- Continuosly read the data from the IMU	- Always transition to **STATE 3**
STATE3	BUTTON_LED - Read buttons & update LEDs	- Countinously read the button - Update the LEDs	- Always transition back to **STATE 4**
STATE4	TRANSFER - Send data to UART1 and UART6	- Send data to UART1 (bluetooth) - UART6 (debug)	- Always transition back to **STATE 2**

2.4 BILL OF MATERIAL - 3D PRINTED PARTS

Most of the 3D printed parts are printed using BambuLab XC1 3D printer and BambuLab filaments. All the CAD 3D model files can be found here for reference:

2.5 BILL OF MATERIAL - ELECTRONICS

All of the components for electronics are listed in the attached PDF file here below

Chapter 3

LAUNCHER

3.1 FINAL DESIGN

The launcher will be the main assembly, receiving data from the controller and moving in 2 degrees of freedom (yaw and pitch angles). It will then shoot the basketball when the controller sends out signals. The launcher will use a customized PCBA centered around the STM32F401CEU6 chip, which will act as the core component to determine movement. The launcher includes a cycloidal gearbox and stepper motors with a planetary gearbox for precise movement control. Additionally, the launcher is equipped with brushless motors and 3 limit switches to ensure accurate positioning.

--> UPDATE PICTURE

3.2 LAUNCHER PCBA

This PCB is the main component of the project, serving as the centerpiece of the launcher. It operates the launcher based on commands received from the controller The PDF of this schematic is also attached with higher solution can be found in `PDF SCHEMATIC`. Below are the all of the PCB schematic contains total of 4 pages describe in the table below

Page	Description
1	Power source schematic for power the pcb, drivers, mcu, etc
2	All connection to power and program for STM32F401CEU6 chip
3	All connection to power and program for DRV8825 stepper motor driver
4	Additional pins + test pads and connectors for limit switches + stepper motors

3.3 FINITE STATE MACHINE

State	Description	Actions	Transitions
STATE0	INIT	- Init all objects in the system	- Always transition to **STATE 1**
STATE1	HOME POSITION CALIBRATION	- Launcher finding limit switches and homing	- Always transition to **STATE 2** if done calibration - Otherwise stay in **STATE 1**
STATE2	DECISION HUB	- Jumping to different state according to condition	- Move to **STATE 3** if MOVE == 1 & SHOT == 0 - Move to **STATE 4** if SHOT == 1 & MOVE == 0 - Move to **STATE 5** if hit any switches - Move to **STATE 6** if ESTOP == 1
STATE3	STEPPER RUN	- Runnning stepper motor	- Always transition to **STATE 2**
STATE4	BLDC RUN	- Running BLDC Motor until fast speed	- Always transition to **STATE 2**
STATE5	LIMIT SWITCH	- Return to home position after hitting the switches	- Stay in **STATE 5" until done homing** **- Then transition to** $**$**STATE 2** when done
STATE6	ESTOP	- Turn off everything if we hitting the ESTOP	- Stop the launcher immediately

3.4 BILL OF MATERIAL - CYCLOIDAL GEARBOX

CYCLOIDAL GEARBOX DEMONSTRATION

Components	Description	Image
Bearing	OD = 55 mm + ID = 35 mm + t = 10 mm	
Bearing	OD = 47 mm + ID = 35 mm + t = 7 mm	
Bearing	OD = 37 mm + ID = 25 mm + t = 7 mm	
Bearing	OD = 15 mm + ID = 6 mm + t = 5 mm	
Bolt	M6 + L = 70 mm + Bolt	
Nut	M6 + t = 5 mm + Nut	
Nut	M6 + t = 5 mm + Lock Nut	

3.5 BILL OF MATERIAL - MECHANICAL

Components	Description	Image
NEMA17	NEMA17 stepper motor	
NEMA17 + GEAR BOX	NEMA17 stepper motor + planetary gearbox	
D4215 BLDC	D4215 BLDC 650 Kvmotor	
UBEC ESC	Hobby ESC control for BLDC	
O-RING	Friction O-Ring	
SWITCH	Switch for BLDC	
BASKETBALL	5 inches basketball	
LIMIT SWITCH	Limit switches for safety	
BATTERY	Lipo battery 11.1 V	

3.6 BILL OF MATERIAL - 3D PRINTED PARTS

Most of the 3D printed parts are printed using BambuLab XC1 3D printer and BambuLab filaments. All the CAD 3D model files can be found here for reference:

3.7 BILL OF MATERIAL - ELECTRONICS

All of the components for electronics are listed in the attached PDF file here below

Chapter 4

REPORTS

4.1 PROPOSAL

4.2 BILL OF MATERIAL

4.3 REFERENCE MANUAL

4.4 OVERALL REPORT

Chapter 5

Topic Index

5.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Topic Documentation

8.1 CMSIS

Topics

- **Stm32f4xx_system**

8.1.1 Detailed Description

8.1.2 Stm32f4xx_system

Topics

- **STM32F4xx_System_Private_Includes**
- **STM32F4xx_System_Private_TypesDefinitions**
- **STM32F4xx_System_Private_Defines**
- **STM32F4xx_System_Private_Macros**
- **STM32F4xx_System_Private_Variables**
- **STM32F4xx_System_Private_FunctionPrototypes**
- **STM32F4xx_System_Private_Functions**

8.1.2.1 Detailed Description

8.1.2.2 STM32F4xx_System_Private_Includes

Macros

- #define **HSE_VALUE** ((uint32_t)25000000)
- #define **HSI_VALUE** ((uint32_t)16000000)
- #define **HSE_VALUE** ((uint32_t)25000000)
- #define **HSI_VALUE** ((uint32_t)16000000)

8.1.2.2.1 Detailed Description

8.1.2.2.2 Macro Definition Documentation

8.1.2.2.2.1 HSE_VALUE [1/2]

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

8.1.2.2.2.2 HSE_VALUE [2/2]

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

8.1.2.2.2.3 HSI_VALUE [1/2]

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

Value of the Internal oscillator in Hz

8.1.2.2.2.4 HSI_VALUE [2/2]

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

Value of the Internal oscillator in Hz

8.1.2.3 STM32F4xx_System_Private_TypesDefinitions

8.1.2.4 STM32F4xx_System_Private_Defines

8.1.2.5 STM32F4xx_System_Private_Macros

8.1.2.6 STM32F4xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 16000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}
- uint32_t **SystemCoreClock** = 16000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

8.1.2.6.1 Detailed Description

8.1.2.6.2 Variable Documentation

8.1.2.6.2.1 AHBPrescTable [1/2]

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
```

8.1.2.6.2.2 AHBPrescTable [2/2]

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
```

8.1.2.6.2.3 APBPrescTable [1/2]

```
const uint8_t APBPrescTable[8] = {0, 0, 0, 0, 1, 2, 3, 4}
```

8.1.2.6.2.4 APBPrescTable [2/2]

```
const uint8_t APBPrescTable[8] = {0, 0, 0, 0, 1, 2, 3, 4}
```

8.1.2.6.2.5 SystemCoreClock [1/2]

```
uint32_t SystemCoreClock = 16000000
```

8.1.2.6.2.6 SystemCoreClock [2/2]

```
uint32_t SystemCoreClock = 16000000
```

8.1.2.7 STM32F4xx_System_Private_FunctionPrototypes

8.1.2.8 STM32F4xx_System_Private_Functions

Functions

- void **SystemInit** (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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.

8.1.2.8.1 Detailed Description

8.1.2.8.2 Function Documentation

8.1.2.8.2.1 SystemCoreClockUpdate()

```
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 HSI, SystemCoreClock will contain the **HSI_VALUE(∗)** (p. 20)
- If SYSCLK source is HSE, SystemCoreClock will contain the **HSE_VALUE(∗∗)** (p. 20)
- If SYSCLK source is PLL, SystemCoreClock will contain the **HSE_VALUE(∗∗)** (p. 20) or **HSI_VALUE(∗)** (p. 20) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in **stm32f4xx_hal_conf.h** (p. 79) 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 **stm32f4xx_hal_conf.h** (p. 79) file (its value depends on the application requirements), 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	

8.1.2.8.2.2 SystemInit()

```
void SystemInit (
            void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

Parameters

None	

Return values

None	

Chapter 9

Data Structure Documentation

9.1 D4215X Struct Reference

Structure to define a D4215 motor with timer, channel, and speed.

`#include <d4215.h>`

Data Fields

- TIM_HandleTypeDef * **timer**
- uint32_t **channel**
- int32_t **speed**

9.1.1 Detailed Description

Structure to define a D4215 motor with timer, channel, and speed.

9.1.2 Field Documentation

9.1.2.1 channel

`uint32_t channel`

Timer channel for the motor

9.1.2.2 speed

`int32_t speed`

Speed of the motor

9.1.2.3 timer

```
TIM_HandleTypeDef* timer
```

Timer handle for the motor

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

- Ball_Launcher_Main/Core/Inc/ **d4215.h**

9.2 LEDX Struct Reference

Structure to define an LED with GPIO port and pin.

```
#include <led.h>
```

Data Fields

- GPIO_TypeDef * **GPIOx**
- uint16_t **PIN**

9.2.1 Detailed Description

Structure to define an LED with GPIO port and pin.

9.2.2 Field Documentation

9.2.2.1 GPIOx

```
GPIO_TypeDef* GPIOx
```

GPIO port of the LED

9.2.2.2 PIN

```
uint16_t PIN
```

GPIO pin of the LED

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

- Ball_Launcher_Main/Core/Inc/ **led.h**

9.3 MPU6050 Struct Reference

Struct representing a mpu6050 imu sensor.

`#include <mpu6050.h>`

Data Fields

- I2C_HandleTypeDef * **hi2c**
- HAL_StatusTypeDef **status**
- uint16_t **addr**
- uint16_t **dt**
- int16_t **gX**
- int16_t **gY**
- int16_t **gZ**
- int16_t **gX_offset**
- int16_t **gY_offset**
- int16_t **gZ_offset**

9.3.1 Detailed Description

Struct representing a mpu6050 imu sensor.

9.3.2 Field Documentation

9.3.2.1 addr

`uint16_t addr`

I2C address

9.3.2.2 dt

`uint16_t dt`

Delta time

9.3.2.3 gX

`int16_t gX`

Gyroscope X axis data

9.3.2.4 gX_offset

`int16_t gX_offset`

Gyroscope X axis offset

9.3.2.5 gY

```
int16_t gY
```

Gyroscope Y axis data

9.3.2.6 gY_offset

```
int16_t gY_offset
```

Gyroscope Y axis offset

9.3.2.7 gZ

```
int16_t gZ
```

Gyroscope Z axis data

9.3.2.8 gZ_offset

```
int16_t gZ_offset
```

Gyroscope Z axis offset

9.3.2.9 hi2c

```
I2C_HandleTypeDef* hi2c
```

I2C handle

9.3.2.10 status

```
HAL_StatusTypeDef status
```

HAL status

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

- Ball_Launcher_Controller/Core/Inc/ **mpu6050.h**

9.4 RadioX Struct Reference

Structure to define a radio driver with timer, channels, and pulse width parameters.

```
#include <radio_driver.h>
```

Data Fields

- TIM_HandleTypeDef * **timer**
- uint32_t **channel1**
- uint32_t **channel2**
- uint32_t **sum1**
- uint32_t **sum2**
- uint32_t **counter1**
- uint32_t **counter2**
- double **pulseWidth**

9.4.1 Detailed Description

Structure to define a radio driver with timer, channels, and pulse width parameters.

9.4.2 Field Documentation

9.4.2.1 channel1

```
uint32_t channel1
```

Timer channel 1

9.4.2.2 channel2

```
uint32_t channel2
```

Timer channel 2

9.4.2.3 counter1

```
uint32_t counter1
```

Counter for channel 1

9.4.2.4 counter2

```
uint32_t counter2
```

Counter for channel 2

9.4.2.5 pulseWidth

```
double pulseWidth
```

Pulse width

9.4.2.6 sum1

```
uint32_t sum1
```

Sum of pulse widths for channel 1

9.4.2.7 sum2

```
uint32_t sum2
```

Sum of pulse widths for channel 2

9.4.2.8 timer

```
TIM_HandleTypeDef* timer
```

Timer handle for the radio driver

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

- Ball_Launcher_Main/Core/Inc/ **radio_driver.h**

9.5 StepperX Struct Reference

Structure to define a stepper motor driver with GPIO ports and pins.

```
#include <stepper_driver.h>
```

Data Fields

- GPIO_TypeDef * **GPIOx**
- uint16_t **EN_PIN**
- uint16_t **DIR_PIN**
- uint16_t **STP_PIN**

9.5.1 Detailed Description

Structure to define a stepper motor driver with GPIO ports and pins.

9.5.2 Field Documentation

9.5.2.1 DIR_PIN

```
uint16_t DIR_PIN
```

Direction pin of the stepper motor driver

9.5.2.2 EN_PIN

```
uint16_t EN_PIN
```

Enable pin of the stepper motor driver

9.5.2.3 GPIOx

```
GPIO_TypeDef* GPIOx
```

GPIO port of the stepper motor driver

9.5.2.4 STP_PIN

```
uint16_t STP_PIN
```

Step pin of the stepper motor driver

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

- Ball_Launcher_Main/Core/Inc/ **stepper_driver.h**

9.6 SwitchX Struct Reference

Structure to define a switch with GPIO port, pin, and status.

```
#include <switch.h>
```

Data Fields

- GPIO_TypeDef * **GPIOx**
- uint16_t **PIN**
- uint16_t **status**

9.6.1 Detailed Description

Structure to define a switch with GPIO port, pin, and status.

9.6.2 Field Documentation

9.6.2.1 GPIOx

```
GPIO_TypeDef* GPIOx
```

GPIO port of the switch

9.6.2.2 PIN

```
uint16_t PIN
```

GPIO pin of the switch

9.6.2.3 status

```
uint16_t status
```

Status of the switch

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

- Ball_Launcher_Main/Core/Inc/ **switch.h**

Chapter 10

File Documentation

10.1 Ball_Launcher_Controller/Core/Inc/main_C.h File Reference

Header for **main_C.c** (p. 53) file. This file contains the common defines of the application.

```
#include "stm32f4xx_hal.h"
```

Functions

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

10.1.1 Detailed Description

Header for **main_C.c** (p. 53) file. This file contains the common defines of the application.

Attention

10.1.2 Function Documentation

10.1.2.1 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

10.2 Ball_Launcher_Controller/Core/Inc/mpu6050.h File Reference

Header for **mpu6050.c** (p. 57) file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

Data Structures

- struct **MPU6050**

  Struct representing a mpu6050 imu sensor.

Functions

- uint16_t **mpu6050_init** (**MPU6050** ∗imux, I2C_HandleTypeDef ∗hi2c)

  *Initializes the **MPU6050** (p. 27) sensor.*
- void **mpu6050_calibrate** (**MPU6050** ∗imux)

  *Calibrates the **MPU6050** (p. 27) sensor.*
- void **mpu6050_update** (**MPU6050** ∗imux)

  *Updates the **MPU6050** (p. 27) sensor data.*
- int16_t **mpu6050_get_gX** (**MPU6050** ∗imux)

  Gets the calibrated X-axis gyroscope data.
- int16_t **mpu6050_get_gY** (**MPU6050** ∗imux)

  Gets the calibrated Y-axis gyroscope data.
- int16_t **mpu6050_get_gZ** (**MPU6050** ∗imux)

  Gets the calibrated Z-axis gyroscope data.
- int32_t **mpu6050_get_X** (**MPU6050** ∗imux)

  Gets the X-axis accelerometer data.
- int32_t **mpu6050_get_Y** (**MPU6050** ∗imux)

  Gets the Y-axis accelerometer data.
- int32_t **mpu6050_get_Z** (**MPU6050** ∗imux)

  Gets the Z-axis accelerometer data.

10.2.1 Detailed Description

Header for **mpu6050.c** (p. 57) file. This file contains the common defines of the application.

10.2.2 Function Documentation

10.2.2.1 mpu6050_calibrate()

```
void mpu6050_calibrate (
            MPU6050 * imux )
```

Calibrates the **MPU6050** (p. 27) sensor.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

10.2.2.2 mpu6050_get_gX()

```
int16_t mpu6050_get_gX (
            MPU6050 * imux )
```

Gets the calibrated X-axis gyroscope data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int16_t Calibrated X-axis gyroscope data.

10.2.2.3 mpu6050_get_gY()

```
int16_t mpu6050_get_gY (
            MPU6050 * imux )
```

Gets the calibrated Y-axis gyroscope data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int16_t Calibrated Y-axis gyroscope data.

10.2.2.4 mpu6050_get_gZ()

```
int16_t mpu6050_get_gZ (
            MPU6050 * imux )
```

Gets the calibrated Z-axis gyroscope data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int16_t Calibrated Z-axis gyroscope data.

10.2.2.5 mpu6050_get_X()

```
int32_t mpu6050_get_X (
            MPU6050 * imux )
```

Gets the X-axis accelerometer data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int32_t X-axis accelerometer data.

10.2.2.6 mpu6050_get_Y()

```
int32_t mpu6050_get_Y (
            MPU6050 * imux )
```

Gets the Y-axis accelerometer data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int32_t Y-axis accelerometer data.

10.2.2.7 mpu6050_get_Z()

```
int32_t mpu6050_get_Z (
            MPU6050 * imux )
```

Gets the Z-axis accelerometer data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int32_t Z-axis accelerometer data.

10.2.2.8 mpu6050_init()

```
uint16_t mpu6050_init (
              MPU6050 * imux,
             I2C_HandleTypeDef * hi2c )
```

Initializes the **MPU6050** (p. 27) sensor.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.
hi2c	Pointer to the I2C handle structure.

Returns

uint16_t Returns 1 if initialization is successful, otherwise 0.

10.2.2.9 mpu6050_update()

```
void mpu6050_update (
              MPU6050 * imux )
```

Updates the **MPU6050** (p. 27) sensor data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.
dt	Time interval in milliseconds.

10.3 Ball_Launcher_Controller/Core/Inc/stm32f4xx_hal_conf_C.h File Reference

HAL configuration template file. This file should be copied to the application folder and renamed to **stm32f4xx_↩ hal_conf_C.h** (p. 37).

```
#include "stm32f4xx_hal_rcc.h"
#include "stm32f4xx_hal_gpio.h"
#include "stm32f4xx_hal_exti.h"
#include "stm32f4xx_hal_dma.h"
#include "stm32f4xx_hal_cortex.h"
#include "stm32f4xx_hal_flash.h"
#include "stm32f4xx_hal_i2c.h"
#include "stm32f4xx_hal_pwr.h"
#include "stm32f4xx_hal_uart.h"
```

Macros

- #define **HAL_MODULE_ENABLED**

  This is the list of modules to be used in the HAL driver.
- #define **HAL_I2C_MODULE_ENABLED**
- #define **HAL_UART_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** 25000000U

  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** 100U
- #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 **LSI_VALUE** 32000U

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

  External Low Speed oscillator (LSE) value.
- #define **LSE_STARTUP_TIMEOUT** 5000U
- #define **EXTERNAL_CLOCK_VALUE** 12288000U

  External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.
- #define **VDD_VALUE** 3300U

  This is the HAL system configuration section.
- #define **TICK_INT_PRIORITY** 15U
- #define **USE_RTOS** 0U
- #define **PREFETCH_ENABLE** 1U
- #define **INSTRUCTION_CACHE_ENABLE** 1U
- #define **DATA_CACHE_ENABLE** 1U
- #define **USE_HAL_ADC_REGISTER_CALLBACKS** 0U /∗ ADC register callback disabled ∗/
- #define **USE_HAL_CAN_REGISTER_CALLBACKS** 0U /∗ CAN register callback disabled ∗/
- #define **USE_HAL_CEC_REGISTER_CALLBACKS** 0U /∗ CEC register callback disabled ∗/
- #define **USE_HAL_CRYP_REGISTER_CALLBACKS** 0U /∗ CRYP register callback disabled ∗/
- #define **USE_HAL_DAC_REGISTER_CALLBACKS** 0U /∗ DAC register callback disabled ∗/
- #define **USE_HAL_DCMI_REGISTER_CALLBACKS** 0U /∗ DCMI register callback disabled ∗/
- #define **USE_HAL_DFSDM_REGISTER_CALLBACKS** 0U /∗ DFSDM register callback disabled ∗/
- #define **USE_HAL_DMA2D_REGISTER_CALLBACKS** 0U /∗ DMA2D register callback disabled ∗/
- #define **USE_HAL_DSI_REGISTER_CALLBACKS** 0U /∗ DSI register callback disabled ∗/
- #define **USE_HAL_ETH_REGISTER_CALLBACKS** 0U /∗ ETH register callback disabled ∗/
- #define **USE_HAL_HASH_REGISTER_CALLBACKS** 0U /∗ HASH register callback disabled ∗/
- #define **USE_HAL_HCD_REGISTER_CALLBACKS** 0U /∗ HCD register callback disabled ∗/
- #define **USE_HAL_I2C_REGISTER_CALLBACKS** 0U /∗ I2C register callback disabled ∗/
- #define **USE_HAL_FMPI2C_REGISTER_CALLBACKS** 0U /∗ FMPI2C register callback disabled ∗/
- #define **USE_HAL_FMPSMBUS_REGISTER_CALLBACKS** 0U /∗ FMPSMBUS register callback disabled ∗/
- #define **USE_HAL_I2S_REGISTER_CALLBACKS** 0U /∗ I2S register callback disabled ∗/
- #define **USE_HAL_IRDA_REGISTER_CALLBACKS** 0U /∗ IRDA register callback disabled ∗/
- #define **USE_HAL_LPTIM_REGISTER_CALLBACKS** 0U /∗ LPTIM register callback disabled ∗/

- #define **USE_HAL_LTDC_REGISTER_CALLBACKS** 0U /* LTDC register callback disabled */
- #define **USE_HAL_MMC_REGISTER_CALLBACKS** 0U /* MMC register callback disabled */
- #define **USE_HAL_NAND_REGISTER_CALLBACKS** 0U /* NAND register callback disabled */
- #define **USE_HAL_NOR_REGISTER_CALLBACKS** 0U /* NOR register callback disabled */
- #define **USE_HAL_PCCARD_REGISTER_CALLBACKS** 0U /* PCCARD register callback disabled */
- #define **USE_HAL_PCD_REGISTER_CALLBACKS** 0U /* PCD register callback disabled */
- #define **USE_HAL_QSPI_REGISTER_CALLBACKS** 0U /* QSPI register callback disabled */
- #define **USE_HAL_RNG_REGISTER_CALLBACKS** 0U /* RNG register callback disabled */
- #define **USE_HAL_RTC_REGISTER_CALLBACKS** 0U /* RTC register callback disabled */
- #define **USE_HAL_SAI_REGISTER_CALLBACKS** 0U /* SAI register callback disabled */
- #define **USE_HAL_SD_REGISTER_CALLBACKS** 0U /* SD register callback disabled */
- #define **USE_HAL_SMARTCARD_REGISTER_CALLBACKS** 0U /* SMARTCARD register callback disabled */
- #define **USE_HAL_SDRAM_REGISTER_CALLBACKS** 0U /* SDRAM register callback disabled */
- #define **USE_HAL_SRAM_REGISTER_CALLBACKS** 0U /* SRAM register callback disabled */
- #define **USE_HAL_SPDIFRX_REGISTER_CALLBACKS** 0U /* SPDIFRX register callback disabled */
- #define **USE_HAL_SMBUS_REGISTER_CALLBACKS** 0U /* SMBUS register callback disabled */
- #define **USE_HAL_SPI_REGISTER_CALLBACKS** 0U /* SPI register callback disabled */
- #define **USE_HAL_TIM_REGISTER_CALLBACKS** 0U /* TIM register callback disabled */
- #define **USE_HAL_UART_REGISTER_CALLBACKS** 0U /* UART register callback disabled */
- #define **USE_HAL_USART_REGISTER_CALLBACKS** 0U /* USART register callback disabled */
- #define **USE_HAL_WWDG_REGISTER_CALLBACKS** 0U /* WWDG register callback disabled */
- #define **MAC_ADDR0** 2U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define **MAC_ADDR1** 0U
- #define **MAC_ADDR2** 0U
- #define **MAC_ADDR3** 0U
- #define **MAC_ADDR4** 0U
- #define **MAC_ADDR5** 0U
- #define **ETH_RX_BUF_SIZE** ETH_MAX_PACKET_SIZE /* **buffer** size for receive */
- #define **ETH_TX_BUF_SIZE** ETH_MAX_PACKET_SIZE /* **buffer** size for transmit */
- #define **ETH_RXBUFNB** 4U /* 4 Rx buffers of size **ETH_RX_BUF_SIZE** */
- #define **ETH_TXBUFNB** 4U /* 4 Tx buffers of size **ETH_TX_BUF_SIZE** */
- #define **DP83848_PHY_ADDRESS**
- #define **PHY_RESET_DELAY** 0x000000FFU
- #define **PHY_CONFIG_DELAY** 0x00000FFFU
- #define **PHY_READ_TO** 0x0000FFFFU
- #define **PHY_WRITE_TO** 0x0000FFFFU
- #define **PHY_BCR** ((uint16_t)0x0000U)
- #define **PHY_BSR** ((uint16_t)0x0001U)
- #define **PHY_RESET** ((uint16_t)0x8000U)
- #define **PHY_LOOPBACK** ((uint16_t)0x4000U)
- #define **PHY_FULLDUPLEX_100M** ((uint16_t)0x2100U)
- #define **PHY_HALFDUPLEX_100M** ((uint16_t)0x2000U)
- #define **PHY_FULLDUPLEX_10M** ((uint16_t)0x0100U)
- #define **PHY_HALFDUPLEX_10M** ((uint16_t)0x0000U)
- #define **PHY_AUTONEGOTIATION** ((uint16_t)0x1000U)
- #define **PHY_RESTART_AUTONEGOTIATION** ((uint16_t)0x0200U)
- #define **PHY_POWERDOWN** ((uint16_t)0x0800U)
- #define **PHY_ISOLATE** ((uint16_t)0x0400U)
- #define **PHY_AUTONEGO_COMPLETE** ((uint16_t)0x0020U)
- #define **PHY_LINKED_STATUS** ((uint16_t)0x0004U)
- #define **PHY_JABBER_DETECTION** ((uint16_t)0x0002U)
- #define **PHY_SR** ((uint16_t))

- #define **PHY_SPEED_STATUS** ((uint16_t))
- #define **PHY_DUPLEX_STATUS** ((uint16_t))
- #define **USE_SPI_CRC** 0U
- #define **assert_param**(expr) ((void)0U)

  Include module's header file.

10.3.1 Detailed Description

HAL configuration template file. This file should be copied to the application folder and renamed to **stm32f4xx_↩ hal_conf_C.h** (p. 37).

Author

MCD Application Team

Attention

10.3.2 Macro Definition Documentation

10.3.2.1 assert_param

```
#define assert_param(
             expr ) ((void)0U)
```

Include module's header file.

10.3.2.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1U
```

10.3.2.3 DP83848_PHY_ADDRESS

```
#define DP83848_PHY_ADDRESS
```

10.3.2.4 ETH_RX_BUF_SIZE

```
#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /*  buffer size for receive */
```

10.3.2.5 ETH_RXBUFNB

```
#define ETH_RXBUFNB 4U /* 4 Rx buffers of size  ETH_RX_BUF_SIZE */
```

10.3.2.6 ETH_TX_BUF_SIZE

```
#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /*  buffer size for transmit */
```

10.3.2.7 ETH_TXBUFNB

```
#define ETH_TXBUFNB 4U /* 4 Tx buffers of size  ETH_TX_BUF_SIZE */
```

10.3.2.8 EXTERNAL_CLOCK_VALUE

```
#define EXTERNAL_CLOCK_VALUE 12288000U
```

External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.

Value of the External audio frequency in Hz

10.3.2.9 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

10.3.2.10 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

10.3.2.11 HAL_EXTI_MODULE_ENABLED

```
#define HAL_EXTI_MODULE_ENABLED
```

10.3.2.12 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

10.3.2.13 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

10.3.2.14 HAL_I2C_MODULE_ENABLED

```
#define HAL_I2C_MODULE_ENABLED
```

10.3.2.15 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

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

10.3.2.16 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

10.3.2.17 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

10.3.2.18 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

10.3.2.19 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT 100U
```

Time out for HSE start up, in ms

10.3.2.20 HSE_VALUE

```
#define HSE_VALUE 25000000U
```

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

10.3.2.21 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

10.3.2.22 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1U
```

10.3.2.23 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

10.3.2.24 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value.

< 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 Low Speed oscillator in Hz

10.3.2.25 LSI_VALUE

```
#define LSI_VALUE 32000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

10.3.2.26 MAC_ADDR0

```
#define MAC_ADDR0 2U
```

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

10.3.2.27 MAC_ADDR1

```
#define MAC_ADDR1 0U
```

10.3.2.28 MAC_ADDR2

```
#define MAC_ADDR2 0U
```

10.3.2.29 MAC_ADDR3

```
#define MAC_ADDR3 0U
```

10.3.2.30 MAC_ADDR4

```
#define MAC_ADDR4 0U
```

10.3.2.31 MAC_ADDR5

```
#define MAC_ADDR5 0U
```

10.3.2.32 PHY_AUTONEGO_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
```

Auto-Negotiation process completed

10.3.2.33 PHY_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
```

Enable auto-negotiation function

10.3.2.34 PHY_BCR

```
#define PHY_BCR ((uint16_t)0x0000U)
```

Transceiver Basic Control Register

10.3.2.35 PHY_BSR

```
#define PHY_BSR ((uint16_t)0x0001U)
```

Transceiver Basic Status Register

10.3.2.36 PHY_CONFIG_DELAY

```
#define PHY_CONFIG_DELAY 0x00000FFFU
```

10.3.2.37 PHY_DUPLEX_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t))
```

PHY Duplex mask

10.3.2.38 PHY_FULLDUPLEX_100M

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
```

Set the full-duplex mode at 100 Mb/s

10.3.2.39 PHY_FULLDUPLEX_10M

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
```

Set the full-duplex mode at 10 Mb/s

10.3.2.40 PHY_HALFDUPLEX_100M

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
```

Set the half-duplex mode at 100 Mb/s

10.3.2.41 PHY_HALFDUPLEX_10M

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
```

Set the half-duplex mode at 10 Mb/s

10.3.2.42 PHY_ISOLATE

```
#define PHY_ISOLATE ((uint16_t)0x0400U)
```

Isolate PHY from MII

10.3.2.43 PHY_JABBER_DETECTION

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
```

Jabber condition detected

10.3.2.44 PHY_LINKED_STATUS

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004U)
```

Valid link established

10.3.2.45 PHY_LOOPBACK

```
#define PHY_LOOPBACK ((uint16_t)0x4000U)
```

Select loop-back mode

10.3.2.46 PHY_POWERDOWN

```
#define PHY_POWERDOWN ((uint16_t)0x0800U)
```

Select the power down mode

10.3.2.47 PHY_READ_TO

```
#define PHY_READ_TO 0x0000FFFFU
```

10.3.2.48 PHY_RESET

```
#define PHY_RESET ((uint16_t)0x8000U)
```

PHY Reset

10.3.2.49 PHY_RESET_DELAY

```
#define PHY_RESET_DELAY 0x000000FFU
```

10.3.2.50 PHY_RESTART_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
```

Restart auto-negotiation function

10.3.2.51 PHY_SPEED_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t))
```

PHY Speed mask

10.3.2.52 PHY_SR

```
#define PHY_SR ((uint16_t))
```

PHY status register Offset

10.3.2.53 PHY_WRITE_TO

```
#define PHY_WRITE_TO 0x0000FFFFU
```

10.3.2.54 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1U
```

10.3.2.55 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority

10.3.2.56 USE_HAL_ADC_REGISTER_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */
```

10.3.2.57 USE_HAL_CAN_REGISTER_CALLBACKS

```
#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */
```

10.3.2.58 USE_HAL_CEC_REGISTER_CALLBACKS

```
#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */
```

10.3.2.59 USE_HAL_CRYP_REGISTER_CALLBACKS

```
#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */
```

10.3.2.60 USE_HAL_DAC_REGISTER_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */
```

10.3.2.61 USE_HAL_DCMI_REGISTER_CALLBACKS

```
#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */
```

10.3.2.62 USE_HAL_DFSDM_REGISTER_CALLBACKS

```
#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */
```

10.3.2.63 USE_HAL_DMA2D_REGISTER_CALLBACKS

```
#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */
```

10.3.2.64 USE_HAL_DSI_REGISTER_CALLBACKS

```
#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */
```

10.3.2.65 USE_HAL_ETH_REGISTER_CALLBACKS

```
#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */
```

10.3.2.66 USE_HAL_FMPI2C_REGISTER_CALLBACKS

```
#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */
```

10.3.2.67 USE_HAL_FMPSMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */
```

10.3.2.68 USE_HAL_HASH_REGISTER_CALLBACKS

```
#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */
```

10.3.2.69 USE_HAL_HCD_REGISTER_CALLBACKS

```
#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */
```

10.3.2.70 USE_HAL_I2C_REGISTER_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */
```

10.3.2.71 USE_HAL_I2S_REGISTER_CALLBACKS

```
#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */
```

10.3.2.72 USE_HAL_IRDA_REGISTER_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */
```

10.3.2.73 USE_HAL_LPTIM_REGISTER_CALLBACKS

```
#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */
```

10.3.2.74 USE_HAL_LTDC_REGISTER_CALLBACKS

```
#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */
```

10.3.2.75 USE_HAL_MMC_REGISTER_CALLBACKS

```
#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */
```

10.3.2.76 USE_HAL_NAND_REGISTER_CALLBACKS

```
#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */
```

10.3.2.77 USE_HAL_NOR_REGISTER_CALLBACKS

```
#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */
```

10.3.2.78 USE_HAL_PCCARD_REGISTER_CALLBACKS

```
#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */
```

10.3.2.79 USE_HAL_PCD_REGISTER_CALLBACKS

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */
```

10.3.2.80 USE_HAL_QSPI_REGISTER_CALLBACKS

```
#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */
```

10.3.2.81 USE_HAL_RNG_REGISTER_CALLBACKS

```
#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */
```

10.3.2.82 USE_HAL_RTC_REGISTER_CALLBACKS

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */
```

10.3.2.83 USE_HAL_SAI_REGISTER_CALLBACKS

```
#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */
```

10.3.2.84 USE_HAL_SD_REGISTER_CALLBACKS

```
#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */
```

10.3.2.85 USE_HAL_SDRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */
```

10.3.2.86 USE_HAL_SMARTCARD_REGISTER_CALLBACKS

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
```

10.3.2.87 USE_HAL_SMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */
```

10.3.2.88 USE_HAL_SPDIFRX_REGISTER_CALLBACKS

```
#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */
```

10.3.2.89 USE_HAL_SPI_REGISTER_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */
```

10.3.2.90 USE_HAL_SRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */
```

10.3.2.91 USE_HAL_TIM_REGISTER_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */
```

10.3.2.92 USE_HAL_UART_REGISTER_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */
```

10.3.2.93 USE_HAL_USART_REGISTER_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */
```

10.3.2.94 USE_HAL_WWDG_REGISTER_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */
```

10.3.2.95 USE_RTOS

```
#define USE_RTOS 0U
```

10.3.2.96 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

10.3.2.97 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

10.4 Ball_Launcher_Controller/Core/Inc/stm32f4xx_it_C.h File Reference

This file contains the headers of the interrupt handlers.

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 Pre-fetch 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.

10.4.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

10.4.2 Function Documentation

10.4.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

10.4.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

10.4.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

10.4.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

10.4.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

10.4.2.6 PendSV_Handler()

```
void PendSV_Handler (
             void  )
```

This function handles Pendable request for system service.

10.4.2.7 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

10.4.2.8 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

10.4.2.9 UsageFault_Handler()

```
void UsageFault_Handler (
             void  )
```

This function handles Undefined instruction or illegal state.

10.5 Ball_Launcher_Controller/Core/Src/main_C.c File Reference

Main program body.

```
#include "main.h"
#include "mpu6050.h"
#include <stdio.h>
#include <string.h>
```

Functions

- void **SystemClock_Config** (void)

  System Clock Configuration.
- int **main** (void)

  The application entry point.
- void **Error_Handler** (void)

  This function is executed in case of error occurrence.

Variables

- I2C_HandleTypeDef **hi2c1**
- UART_HandleTypeDef **huart1**
- UART_HandleTypeDef **huart2**
- int16_t **gX** = 0
- int16_t **gY** = 0
- int16_t **gZ** = 0
- uint16_t **stat** = 0
- uint16_t **shot** = 0
- uint16_t **move** = 0
- uint16_t **STATE** = 0
- uint16_t **STATE_0_INIT** = 0
- uint16_t **STATE_1_ERROR** = 1
- uint16_t **STATE_2_IMU** = 2
- uint16_t **STATE_3_BUTTON_LED** = 3
- uint16_t **STATE_4_TRANSFER** = 4
- char **buffer** [100]

10.5.1 Detailed Description

Main program body.

Attention

10.5.2 Function Documentation

10.5.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

10.5.2.2 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

10.5.2.3 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

10.5.3 Variable Documentation

10.5.3.1 buffer

```
char buffer[100]
```

10.5.3.2 gX

```
int16_t gX = 0
```

10.5.3.3 gY

```
int16_t gY = 0
```

10.5.3.4 gZ

```
int16_t gZ = 0
```

10.5.3.5 hi2c1

```
I2C_HandleTypeDef hi2c1
```

10.5.3.6 huart1

```
UART_HandleTypeDef huart1
```

10.5.3.7 huart2

```
UART_HandleTypeDef huart2
```

10.5.3.8 move

```
uint16_t move = 0
```

10.5.3.9 shot

```
uint16_t shot = 0
```

10.5.3.10 stat

```
uint16_t stat = 0
```

10.5.3.11 STATE

```
uint16_t STATE = 0
```

10.5.3.12 STATE_0_INIT

```
uint16_t STATE_0_INIT = 0
```

10.5.3.13 STATE_1_ERROR

```
uint16_t STATE_1_ERROR = 1
```

10.5.3.14 STATE_2_IMU

```
uint16_t STATE_2_IMU = 2
```

10.5.3.15 STATE_3_BUTTON_LED

```
uint16_t STATE_3_BUTTON_LED = 3
```

10.5.3.16 STATE_4_TRANSFER

```
uint16_t STATE_4_TRANSFER = 4
```

10.6 Ball_Launcher_Controller/Core/Src/mpu6050.c File Reference

Implementation of mpu6050 sensor functions.

```
#include "mpu6050.h"
```

Functions

- uint16_t **mpu6050_init** (**MPU6050** ∗imux, I2C_HandleTypeDef ∗hi2c)
  *Initializes the **MPU6050** (p. 27) sensor.*
- void **mpu6050_calibrate** (**MPU6050** ∗imux)
  *Calibrates the **MPU6050** (p. 27) sensor.*
- void **mpu6050_update** (**MPU6050** ∗imux)
  *Updates the **MPU6050** (p. 27) sensor data.*
- int16_t **mpu6050_get_gX** (**MPU6050** ∗imux)
  Gets the calibrated X-axis gyroscope data.
- int16_t **mpu6050_get_gY** (**MPU6050** ∗imux)
  Gets the calibrated Y-axis gyroscope data.
- int16_t **mpu6050_get_gZ** (**MPU6050** ∗imux)
  Gets the calibrated Z-axis gyroscope data.

10.6.1 Detailed Description

Implementation of mpu6050 sensor functions.

Created on: May 3, 2024 Author: vvinh

10.6.2 Function Documentation

10.6.2.1 mpu6050_calibrate()

```
void mpu6050_calibrate (
            MPU6050 * imux )
```

Calibrates the **MPU6050** (p. 27) sensor.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

10.6.2.2 mpu6050_get_gX()

```
int16_t mpu6050_get_gX (
            MPU6050 * imux )
```

Gets the calibrated X-axis gyroscope data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int16_t Calibrated X-axis gyroscope data.

10.6.2.3 mpu6050_get_gY()

```
int16_t mpu6050_get_gY (
            MPU6050 * imux )
```

Gets the calibrated Y-axis gyroscope data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int16_t Calibrated Y-axis gyroscope data.

10.6.2.4 mpu6050_get_gZ()

```
int16_t mpu6050_get_gZ (
            MPU6050 * imux )
```

Gets the calibrated Z-axis gyroscope data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.

Returns

int16_t Calibrated Z-axis gyroscope data.

10.6.2.5 mpu6050_init()

```
uint16_t mpu6050_init (
```

```
                MPU6050 * imux,
               I2C_HandleTypeDef * hi2c )
```

Initializes the **MPU6050** (p. 27) sensor.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.
hi2c	Pointer to the I2C handle structure.

Returns

uint16_t Returns 1 if initialization is successful, otherwise 0.

10.6.2.6 mpu6050_update()

```
void mpu6050_update (
                MPU6050 * imux )
```

Updates the **MPU6050** (p. 27) sensor data.

Parameters

imux	Pointer to the **MPU6050** (p. 27) structure.
dt	Time interval in milliseconds.

10.7 Ball_Launcher_Controller/Core/Src/stm32f4xx_hal_msp_C.c File Reference

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

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

Functions

- void **HAL_MspInit** (void)
- void **HAL_I2C_MspInit** (I2C_HandleTypeDef *hi2c)

  I2C MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_I2C_MspDeInit** (I2C_HandleTypeDef *hi2c)

  I2C MSP De-Initialization This function freeze the hardware resources used in this example.
- void **HAL_UART_MspInit** (UART_HandleTypeDef *huart)

  UART MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_UART_MspDeInit** (UART_HandleTypeDef *huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.

10.7.1 Detailed Description

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

Attention

10.7.2 Function Documentation

10.7.2.1 HAL_I2C_MspDeInit()

```
void HAL_I2C_MspDeInit (
            I2C_HandleTypeDef * hi2c )
```

I2C MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hi2c	I2C handle pointer

Return values

None	

I2C1 GPIO Configuration PB6 ------> I2C1_SCL PB7 ------> I2C1_SDA

10.7.2.2 HAL_I2C_MspInit()

```
void HAL_I2C_MspInit (
            I2C_HandleTypeDef * hi2c )
```

I2C MSP Initialization This function configures the hardware resources used in this example.

Parameters

hi2c	I2C handle pointer

Return values

None	

I2C1 GPIO Configuration PB6 ------> I2C1_SCL PB7 ------> I2C1_SDA

10.7.2.3 HAL_MspInit()

```
void HAL_MspInit (
             void  )
```

Initializes the Global MSP.

10.7.2.4 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
             UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART1 GPIO Configuration PA9 ------> USART1_TX PA10 ------> USART1_RX

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

10.7.2.5 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
             UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART1 GPIO Configuration PA9 ------> USART1_TX PA10 ------> USART1_RX

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

10.8 Ball_Launcher_Controller/Core/Src/stm32f4xx_it_C.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_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 Pre-fetch 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.

10.8.1 Detailed Description

Interrupt Service Routines.

Attention

10.8.2 Function Documentation

10.8.2.1 BusFault_Handler()

```
void BusFault_Handler (
            void  )
```

This function handles Pre-fetch fault, memory access fault.

10.8.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

10.8.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

10.8.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

10.8.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

10.8.2.6 PendSV_Handler()

```
void PendSV_Handler (
             void  )
```

This function handles Pendable request for system service.

10.8.2.7 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

10.8.2.8 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

10.8.2.9 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

10.9 Ball_Launcher_Controller/Core/Src/syscalls_C.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Functions

- int **__io_putchar** (int ch) **__attribute__**((weak))
- int **__io_getchar** (void)
- void **initialise_monitor_handles** ()
- int **_getpid** (void)
- int **_kill** (int pid, int sig)
- void **_exit** (int status)
- **__attribute__** ((weak))
- int **_close** (int file)
- int **_fstat** (int file, struct **stat** ∗st)
- int **_isatty** (int file)
- int **_lseek** (int file, int ptr, int dir)
- int **_open** (char ∗path, int flags,...)
- int **_wait** (int ∗status)
- int **_unlink** (char ∗name)
- int **_times** (struct tms ∗buf)
- int **_stat** (char ∗file, struct **stat** ∗st)
- int **_link** (char ∗old, char ∗new)
- int **_fork** (void)
- int **_execve** (char ∗name, char ∗∗argv, char ∗∗env)

Variables

- char ∗∗ **environ** = __env

10.9.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

10.9.2 Function Documentation

10.9.2.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

10.9.2.2 __io_getchar()

```
int __io_getchar (
             void  )  [extern]
```

10.9.2.3 __io_putchar()

```
int __io_putchar (
             int ch )  [extern]
```

10.9.2.4 _close()

```
int _close (
             int file )
```

10.9.2.5 _execve()

```
int _execve (
             char * name,
             char ** argv,
             char ** env )
```

10.9.2.6 _exit()

```
void _exit (
            int status )
```

10.9.2.7 _fork()

```
int _fork (
            void  )
```

10.9.2.8 _fstat()

```
int _fstat (
            int file,
            struct  stat * st )
```

10.9.2.9 _getpid()

```
int _getpid (
            void  )
```

10.9.2.10 _isatty()

```
int _isatty (
            int file )
```

10.9.2.11 _kill()

```
int _kill (
            int pid,
            int sig )
```

10.9.2.12 _link()

```
int _link (
            char * old,
            char * new )
```

10.9.2.13 _lseek()

```
int _lseek (
            int file,
            int ptr,
            int dir )
```

10.9.2.14 _open()

```
int _open (
             char * path,
             int flags,
              ...  )
```

10.9.2.15 _stat()

```
int _stat (
             char * file,
             struct  stat * st )
```

10.9.2.16 _times()

```
int _times (
             struct tms * buf )
```

10.9.2.17 _unlink()

```
int _unlink (
             char * name )
```

10.9.2.18 _wait()

```
int _wait (
             int * status )
```

10.9.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

10.9.3 Variable Documentation

10.9.3.1 environ

```
char** environ = __env
```

10.10 Ball_Launcher_Controller/Core/Src/sysmem_C.c File Reference

STM32CubeIDE System Memory calls file.

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

Functions

- void * **_sbrk** (ptrdiff_t incr)

  _sbrk() *(p. 68) allocates memory to the newlib heap and is used by malloc and others from the C library*

10.10.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

10.10.2 Function Documentation

10.10.2.1 _sbrk()

```
void * _sbrk (
            ptrdiff_t incr )
```

_sbrk() (p. 68) allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ##########################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ##########################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

10.11 Ball_Launcher_Controller/Core/Src/system_stm32f4xx_C.c File Reference

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

```
#include "stm32f4xx.h"
```

Macros

- #define **HSE_VALUE** ((uint32_t)25000000)
- #define **HSI_VALUE** ((uint32_t)16000000)

Functions

- void **SystemInit** (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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** = 16000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

10.11.1 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()** (p. 22): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f4xx.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()** (p. 22): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

10.12 Ball_Launcher_Main/Core/Inc/d4215.h File Reference

Header for **d4215.c** (p. 96) file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

Data Structures

- struct **D4215X**
  Structure to define a D4215 motor with timer, channel, and speed.

Functions

- void **D4215_init** (**D4215X** ∗bldcx, TIM_HandleTypeDef ∗timer, uint32_t channel)
  Initializes the D4215 motor with the specified timer and channel.
- void **D4215_set** (**D4215X** ∗bldcx, int32_t speed)
  Sets the speed of the D4215 motor.

10.12.1 Detailed Description

Header for **d4215.c** (p. 96) file. This file contains the common defines of the application.

Created on: May 1, 2024

Author: vvinh

10.12.2 Function Documentation

10.12.2.1 D4215_init()

```
void D4215_init (
              D4215X * bldcx,
             TIM_HandleTypeDef * timer,
             uint32_t channel )
```

Initializes the D4215 motor with the specified timer and channel.

Parameters

bldcx	Pointer to the **D4215X** (p. 25) structure
timer	Timer handle
channel	Timer channel

< Start PWM for the specified timer and channel

10.12.2.2 D4215_set()

```
void D4215_set (
              D4215X * bldcx,
             int32_t spd )
```

Sets the speed of the D4215 motor.

Parameters

bldcx	Pointer to the **D4215X** (p. 25) structure
speed	Speed of the motor (range: 0 to 100)
bldcx	Pointer to the **D4215X** (p. 25) structure
spd	Speed of the motor (range: 5 to 10)

< Calculate the duty cycle based on the input speed

< Set the duty cycle for the PWM

10.13 Ball_Launcher_Main/Core/Inc/led.h File Reference

Header for **led.c** (p. 97) file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

Data Structures

- struct **LEDX**

  Structure to define an LED with GPIO port and pin.

Functions

- void **LED_init** (**LEDX** *LEDx, GPIO_TypeDef *GPIO, uint16_t PIN)

  Initializes the LED with the specified GPIO port and pin.
- void **LED_on** (**LEDX** *LEDx)

  Turns on the LED.
- void **LED_off** (**LEDX** *LEDx)

  Turns off the LED.
- void **LED_toggle** (**LEDX** *LEDx)

  Toggles the LED state.

10.13.1 Detailed Description

Header for **led.c** (p. 97) file. This file contains the common defines of the application.

Created on: May 17, 2024

Author: vvinh

10.13.2 Function Documentation

10.13.2.1 LED_init()

```
void LED_init (
              LEDX * LEDx,
             GPIO_TypeDef * GPIO,
             uint16_t PIN )
```

Initializes the LED with the specified GPIO port and pin.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure
GPIO	GPIO port
PIN	GPIO pin

< Assign the GPIO port

< Assign the GPIO pin

10.13.2.2 LED_off()

```
void LED_off (
              LEDX * LEDx )
```

Turns off the LED.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure

< Set the pin low to turn off the LED

10.13.2.3 LED_on()

```
void LED_on (
              LEDX * LEDx )
```

Turns on the LED.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure

< Set the pin high to turn on the LED

10.13.2.4 LED_toggle()

```
void LED_toggle (
              LEDX * LEDx )
```

Toggles the LED state.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure

< Toggle the pin to change the LED state

10.14 Ball_Launcher_Main/Core/Inc/main.h File Reference

Header for **main.c** (p. 99) file. This file contains the common defines of the application.

```
#include "stm32f4xx_hal.h"
```

Functions

- void **HAL_TIM_MspPostInit** (TIM_HandleTypeDef *htim)
- void **Error_Handler** (void)

  This function is executed in case of error occurrence.

10.14.1 Detailed Description

Header for **main.c** (p. 99) file. This file contains the common defines of the application.

Attention

10.14.2 Function Documentation

10.14.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

10.14.2.2 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * htim )
```

TIM2 GPIO Configuration PA1 ------> TIM2_CH2 PA2 ------> TIM2_CH3

10.15 Ball_Launcher_Main/Core/Inc/radio_driver.h File Reference

Header for **radio_driver.c** (p. 106) file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

Data Structures

- struct **RadioX**
  Structure to define a radio driver with timer, channels, and pulse width parameters.

Functions

- void **Radio_init** (**RadioX** ∗radio, TIM_HandleTypeDef ∗timer, uint32_t channel1, uint32_t channel2)
  Initializes the radio driver with the specified timer and channels.
- void **capturePulseWidth** (**RadioX** ∗radio)
  Captures the pulse width for the radio driver.
- void **update** (**RadioX** ∗radio, int pw1, int pw2)
  Updates the radio driver with pulse width values.
- double **Radio_getPulseWidth** (**RadioX** ∗radio)
  Gets the pulse width of the radio driver.

10.15.1 Detailed Description

Header for **radio_driver.c** (p. 106) file. This file contains the common defines of the application.

Created on: May 2, 2024

Author: vvinh

10.15.2 Function Documentation

10.15.2.1 capturePulseWidth()

```
void capturePulseWidth (
             RadioX * radio )
```

Captures the pulse width for the radio driver.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure

< Read captured value for channel 1

< Read captured value for channel 2

< Calculate the pulse width

10.15.2.2 Radio_getPulseWidth()

```
double Radio_getPulseWidth (
              RadioX * radio )
```

Gets the pulse width of the radio driver.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure

Returns

double Pulse width value

Parameters

radio	Pointer to the **RadioX** (p. 28) structure

Returns

double Pulse width value in milliseconds

10.15.2.3 Radio_init()

```
void Radio_init (
              RadioX * radio,
             TIM_HandleTypeDef * timer,
             uint32_t channel1,
             uint32_t channel2 )
```

Initializes the radio driver with the specified timer and channels.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure
timer	Timer handle
channel1	Timer channel 1
channel2	Timer channel 2

< Start input capture interrupt for channel 1

< Start input capture interrupt for channel 2

10.15.2.4 update()

```
void update (
              RadioX * radio,
             int pw1,
             int pw2 )
```

Updates the radio driver with pulse width values.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure
pw1	Pulse width value for channel 1
pw2	Pulse width value for channel 2

10.16 Ball_Launcher_Main/Core/Inc/stepper_driver.h File Reference

Header for **stepper_driver.c** (p. 108) file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

Data Structures

- struct **StepperX**

  Structure to define a stepper motor driver with GPIO ports and pins.

Functions

- void **Stepper_init** (**StepperX** *stepperx, GPIO_TypeDef *GPIO, uint16_t EN_PIN, uint16_t DIR_PIN, uint16_t STP_PIN)

  Initializes the stepper motor driver with the specified GPIO port and pins.
- void **Stepper_enable** (**StepperX** *stepperx)

  Enables the stepper motor driver.
- void **Stepper_disable** (**StepperX** *stepperx)

  Disables the stepper motor driver.
- void **Stepper_setspeed** (**StepperX** *stepperx, uint16_t speed, uint8_t dir)

  Sets the speed and direction of the stepper motor.
- void **SysTick_Init** (void)

  Initializes the system tick for delay functions.
- void **Delay_us** (uint32_t us)

  Delays the program execution for a specified number of microseconds.

10.16.1 Detailed Description

Header for **stepper_driver.c** (p. 108) file. This file contains the common defines of the application.

Created on: May 17, 2024

Author: vvinh

10.16.2 Function Documentation

10.16.2.1 Delay_us()

```
void Delay_us (
            uint32_t us )
```

Delays the program execution for a specified number of microseconds.

Parameters

us	Number of microseconds to delay

10.16.2.2 Stepper_disable()

```
void Stepper_disable (
              StepperX * stepperx )
```

Disables the stepper motor driver.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure

10.16.2.3 Stepper_enable()

```
void Stepper_enable (
              StepperX * stepperx )
```

Enables the stepper motor driver.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure

10.16.2.4 Stepper_init()

```
void Stepper_init (
```

```
 StepperX * stepperx,
GPIO_TypeDef * GPIO,
uint16_t EN_PIN,
uint16_t DIR_PIN,
uint16_t STP_PIN )
```

Initializes the stepper motor driver with the specified GPIO port and pins.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure
GPIO	GPIO port
EN_PIN	Enable pin
DIR_PIN	Direction pin
STP_PIN	Step pin

10.16.2.5 Stepper_setspeed()

```
void Stepper_setspeed (
             StepperX * stepperx,
            uint16_t speed,
            uint8_t dir )
```

Sets the speed and direction of the stepper motor.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure
speed	Speed of the stepper motor
dir	Direction of the stepper motor (1 for one direction, 0 for the opposite)
stepperx	Pointer to the **StepperX** (p. 30) structure
speed	Speed of the stepper motor (delay between steps)
dir	Direction of the stepper motor (1 for one direction, 0 for the opposite)

< Set the direction pin

< Set the step pin high

< Delay to control the speed

< Set the step pin low

< Delay to control the speed

10.16.2.6 SysTick_Init()

```
void SysTick_Init (
            void  )
```

Initializes the system tick for delay functions.

Initializes the system tick for delay functions.

10.17 Ball_Launcher_Main/Core/Inc/stm32f4xx_hal_conf.h File Reference

```
#include "stm32f4xx_hal_rcc.h"
#include "stm32f4xx_hal_gpio.h"
#include "stm32f4xx_hal_exti.h"
#include "stm32f4xx_hal_dma.h"
#include "stm32f4xx_hal_cortex.h"
#include "stm32f4xx_hal_flash.h"
#include "stm32f4xx_hal_pwr.h"
#include "stm32f4xx_hal_tim.h"
#include "stm32f4xx_hal_uart.h"
```

Macros

- #define **HAL_MODULE_ENABLED**

  This is the list of modules to be used in the HAL driver.
- #define **HAL_TIM_MODULE_ENABLED**
- #define **HAL_UART_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** 25000000U

  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** 100U
- #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 **LSI_VALUE** 32000U

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

  External Low Speed oscillator (LSE) value.
- #define **LSE_STARTUP_TIMEOUT** 5000U
- #define **EXTERNAL_CLOCK_VALUE** 12288000U

  External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.
- #define **VDD_VALUE** 3300U

  This is the HAL system configuration section.
- #define **TICK_INT_PRIORITY** 15U
- #define **USE_RTOS** 0U
- #define **PREFETCH_ENABLE** 1U
- #define **INSTRUCTION_CACHE_ENABLE** 1U
- #define **DATA_CACHE_ENABLE** 1U
- #define **USE_HAL_ADC_REGISTER_CALLBACKS** 0U /∗ ADC register callback disabled ∗/
- #define **USE_HAL_CAN_REGISTER_CALLBACKS** 0U /∗ CAN register callback disabled ∗/
- #define **USE_HAL_CEC_REGISTER_CALLBACKS** 0U /∗ CEC register callback disabled ∗/
- #define **USE_HAL_CRYP_REGISTER_CALLBACKS** 0U /∗ CRYP register callback disabled ∗/

- #define **USE_HAL_DAC_REGISTER_CALLBACKS** 0U /* DAC register callback disabled */
- #define **USE_HAL_DCMI_REGISTER_CALLBACKS** 0U /* DCMI register callback disabled */
- #define **USE_HAL_DFSDM_REGISTER_CALLBACKS** 0U /* DFSDM register callback disabled */
- #define **USE_HAL_DMA2D_REGISTER_CALLBACKS** 0U /* DMA2D register callback disabled */
- #define **USE_HAL_DSI_REGISTER_CALLBACKS** 0U /* DSI register callback disabled */
- #define **USE_HAL_ETH_REGISTER_CALLBACKS** 0U /* ETH register callback disabled */
- #define **USE_HAL_HASH_REGISTER_CALLBACKS** 0U /* HASH register callback disabled */
- #define **USE_HAL_HCD_REGISTER_CALLBACKS** 0U /* HCD register callback disabled */
- #define **USE_HAL_I2C_REGISTER_CALLBACKS** 0U /* I2C register callback disabled */
- #define **USE_HAL_FMPI2C_REGISTER_CALLBACKS** 0U /* FMPI2C register callback disabled */
- #define **USE_HAL_FMPSMBUS_REGISTER_CALLBACKS** 0U /* FMPSMBUS register callback disabled */
- #define **USE_HAL_I2S_REGISTER_CALLBACKS** 0U /* I2S register callback disabled */
- #define **USE_HAL_IRDA_REGISTER_CALLBACKS** 0U /* IRDA register callback disabled */
- #define **USE_HAL_LPTIM_REGISTER_CALLBACKS** 0U /* LPTIM register callback disabled */
- #define **USE_HAL_LTDC_REGISTER_CALLBACKS** 0U /* LTDC register callback disabled */
- #define **USE_HAL_MMC_REGISTER_CALLBACKS** 0U /* MMC register callback disabled */
- #define **USE_HAL_NAND_REGISTER_CALLBACKS** 0U /* NAND register callback disabled */
- #define **USE_HAL_NOR_REGISTER_CALLBACKS** 0U /* NOR register callback disabled */
- #define **USE_HAL_PCCARD_REGISTER_CALLBACKS** 0U /* PCCARD register callback disabled */
- #define **USE_HAL_PCD_REGISTER_CALLBACKS** 0U /* PCD register callback disabled */
- #define **USE_HAL_QSPI_REGISTER_CALLBACKS** 0U /* QSPI register callback disabled */
- #define **USE_HAL_RNG_REGISTER_CALLBACKS** 0U /* RNG register callback disabled */
- #define **USE_HAL_RTC_REGISTER_CALLBACKS** 0U /* RTC register callback disabled */
- #define **USE_HAL_SAI_REGISTER_CALLBACKS** 0U /* SAI register callback disabled */
- #define **USE_HAL_SD_REGISTER_CALLBACKS** 0U /* SD register callback disabled */
- #define **USE_HAL_SMARTCARD_REGISTER_CALLBACKS** 0U /* SMARTCARD register callback disabled */
- #define **USE_HAL_SDRAM_REGISTER_CALLBACKS** 0U /* SDRAM register callback disabled */
- #define **USE_HAL_SRAM_REGISTER_CALLBACKS** 0U /* SRAM register callback disabled */
- #define **USE_HAL_SPDIFRX_REGISTER_CALLBACKS** 0U /* SPDIFRX register callback disabled */
- #define **USE_HAL_SMBUS_REGISTER_CALLBACKS** 0U /* SMBUS register callback disabled */
- #define **USE_HAL_SPI_REGISTER_CALLBACKS** 0U /* SPI register callback disabled */
- #define **USE_HAL_TIM_REGISTER_CALLBACKS** 0U /* TIM register callback disabled */
- #define **USE_HAL_UART_REGISTER_CALLBACKS** 0U /* UART register callback disabled */
- #define **USE_HAL_USART_REGISTER_CALLBACKS** 0U /* USART register callback disabled */
- #define **USE_HAL_WWDG_REGISTER_CALLBACKS** 0U /* WWDG register callback disabled */
- #define **MAC_ADDR0** 2U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define **MAC_ADDR1** 0U
- #define **MAC_ADDR2** 0U
- #define **MAC_ADDR3** 0U
- #define **MAC_ADDR4** 0U
- #define **MAC_ADDR5** 0U
- #define **ETH_RX_BUF_SIZE** ETH_MAX_PACKET_SIZE /* **buffer** size for receive */
- #define **ETH_TX_BUF_SIZE** ETH_MAX_PACKET_SIZE /* **buffer** size for transmit */
- #define **ETH_RXBUFNB** 4U /* 4 Rx buffers of size **ETH_RX_BUF_SIZE** */
- #define **ETH_TXBUFNB** 4U /* 4 Tx buffers of size **ETH_TX_BUF_SIZE** */
- #define **DP83848_PHY_ADDRESS**
- #define **PHY_RESET_DELAY** 0x000000FFU
- #define **PHY_CONFIG_DELAY** 0x00000FFFU
- #define **PHY_READ_TO** 0x0000FFFFU
- #define **PHY_WRITE_TO** 0x0000FFFFU
- #define **PHY_BCR** ((uint16_t)0x0000U)

- #define **PHY_BSR** ((uint16_t)0x0001U)
- #define **PHY_RESET** ((uint16_t)0x8000U)
- #define **PHY_LOOPBACK** ((uint16_t)0x4000U)
- #define **PHY_FULLDUPLEX_100M** ((uint16_t)0x2100U)
- #define **PHY_HALFDUPLEX_100M** ((uint16_t)0x2000U)
- #define **PHY_FULLDUPLEX_10M** ((uint16_t)0x0100U)
- #define **PHY_HALFDUPLEX_10M** ((uint16_t)0x0000U)
- #define **PHY_AUTONEGOTIATION** ((uint16_t)0x1000U)
- #define **PHY_RESTART_AUTONEGOTIATION** ((uint16_t)0x0200U)
- #define **PHY_POWERDOWN** ((uint16_t)0x0800U)
- #define **PHY_ISOLATE** ((uint16_t)0x0400U)
- #define **PHY_AUTONEGO_COMPLETE** ((uint16_t)0x0020U)
- #define **PHY_LINKED_STATUS** ((uint16_t)0x0004U)
- #define **PHY_JABBER_DETECTION** ((uint16_t)0x0002U)
- #define **PHY_SR** ((uint16_t))
- #define **PHY_SPEED_STATUS** ((uint16_t))
- #define **PHY_DUPLEX_STATUS** ((uint16_t))
- #define **USE_SPI_CRC** 0U
- #define **assert_param**(expr) ((void)0U)

  Include module's header file.

10.17.1 Macro Definition Documentation

10.17.1.1 assert_param

```
#define assert_param(
             expr ) ((void)0U)
```

Include module's header file.

10.17.1.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1U
```

10.17.1.3 DP83848_PHY_ADDRESS

```
#define DP83848_PHY_ADDRESS
```

10.17.1.4 ETH_RX_BUF_SIZE

```
#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /*  buffer size for receive */
```

10.17.1.5 ETH_RXBUFNB

```
#define ETH_RXBUFNB 4U /* 4 Rx buffers of size  ETH_RX_BUF_SIZE */
```

10.17.1.6 ETH_TX_BUF_SIZE

```
#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /*  buffer size for transmit */
```

10.17.1.7 ETH_TXBUFNB

```
#define ETH_TXBUFNB 4U /* 4 Tx buffers of size  ETH_TX_BUF_SIZE */
```

10.17.1.8 EXTERNAL_CLOCK_VALUE

```
#define EXTERNAL_CLOCK_VALUE 12288000U
```

External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.

Value of the External audio frequency in Hz

10.17.1.9 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

10.17.1.10 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

10.17.1.11 HAL_EXTI_MODULE_ENABLED

```
#define HAL_EXTI_MODULE_ENABLED
```

10.17.1.12 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

10.17.1.13 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

10.17.1.14 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

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

10.17.1.15 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

10.17.1.16 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

10.17.1.17 HAL_TIM_MODULE_ENABLED

```
#define HAL_TIM_MODULE_ENABLED
```

10.17.1.18 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

10.17.1.19 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT 100U
```

Time out for HSE start up, in ms

10.17.1.20 HSE_VALUE

```
#define HSE_VALUE 25000000U
```

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

10.17.1.21 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

10.17.1.22 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1U
```

10.17.1.23 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

10.17.1.24 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value.

< 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 Low Speed oscillator in Hz

10.17.1.25 LSI_VALUE

```
#define LSI_VALUE 32000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

10.17.1.26 MAC_ADDR0

```
#define MAC_ADDR0 2U
```

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

10.17.1.27 MAC_ADDR1

```
#define MAC_ADDR1 0U
```

10.17.1.28 MAC_ADDR2

```
#define MAC_ADDR2 0U
```

10.17.1.29 MAC_ADDR3

```
#define MAC_ADDR3 0U
```

10.17.1.30 MAC_ADDR4

```
#define MAC_ADDR4 0U
```

10.17.1.31 MAC_ADDR5

```
#define MAC_ADDR5 0U
```

10.17.1.32 PHY_AUTONEGO_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
```

Auto-Negotiation process completed

10.17.1.33 PHY_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
```

Enable auto-negotiation function

10.17.1.34 PHY_BCR

```
#define PHY_BCR ((uint16_t)0x0000U)
```

Transceiver Basic Control Register

10.17.1.35 PHY_BSR

```
#define PHY_BSR ((uint16_t)0x0001U)
```

Transceiver Basic Status Register

10.17.1.36 PHY_CONFIG_DELAY

```
#define PHY_CONFIG_DELAY 0x00000FFFU
```

10.17.1.37 PHY_DUPLEX_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t))
```

PHY Duplex mask

10.17.1.38 PHY_FULLDUPLEX_100M

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
```

Set the full-duplex mode at 100 Mb/s

10.17.1.39 PHY_FULLDUPLEX_10M

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
```

Set the full-duplex mode at 10 Mb/s

10.17.1.40 PHY_HALFDUPLEX_100M

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
```

Set the half-duplex mode at 100 Mb/s

10.17.1.41 PHY_HALFDUPLEX_10M

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
```

Set the half-duplex mode at 10 Mb/s

10.17.1.42 PHY_ISOLATE

```
#define PHY_ISOLATE ((uint16_t)0x0400U)
```

Isolate PHY from MII

10.17.1.43 PHY_JABBER_DETECTION

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
```

Jabber condition detected

10.17.1.44 PHY_LINKED_STATUS

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004U)
```

Valid link established

10.17.1.45 PHY_LOOPBACK

```
#define PHY_LOOPBACK ((uint16_t)0x4000U)
```

Select loop-back mode

10.17.1.46 PHY_POWERDOWN

```
#define PHY_POWERDOWN ((uint16_t)0x0800U)
```

Select the power down mode

10.17.1.47 PHY_READ_TO

```
#define PHY_READ_TO 0x0000FFFFU
```

10.17.1.48 PHY_RESET

```
#define PHY_RESET ((uint16_t)0x8000U)
```

PHY Reset

10.17.1.49 PHY_RESET_DELAY

```
#define PHY_RESET_DELAY 0x000000FFU
```

10.17.1.50 PHY_RESTART_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
```

Restart auto-negotiation function

10.17.1.51 PHY_SPEED_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t))
```

PHY Speed mask

10.17.1.52 PHY_SR

```
#define PHY_SR ((uint16_t))
```

PHY status register Offset

10.17.1.53 PHY_WRITE_TO

```
#define PHY_WRITE_TO 0x0000FFFFU
```

10.17.1.54 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1U
```

10.17.1.55 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority

10.17.1.56 USE_HAL_ADC_REGISTER_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */
```

10.17.1.57 USE_HAL_CAN_REGISTER_CALLBACKS

```
#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */
```

10.17.1.58 USE_HAL_CEC_REGISTER_CALLBACKS

```
#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */
```

10.17.1.59 USE_HAL_CRYP_REGISTER_CALLBACKS

```
#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */
```

10.17.1.60 USE_HAL_DAC_REGISTER_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */
```

10.17.1.61 USE_HAL_DCMI_REGISTER_CALLBACKS

```
#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */
```

10.17.1.62 USE_HAL_DFSDM_REGISTER_CALLBACKS

```
#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */
```

10.17.1.63 USE_HAL_DMA2D_REGISTER_CALLBACKS

```
#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */
```

10.17.1.64 USE_HAL_DSI_REGISTER_CALLBACKS

```
#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */
```

10.17.1.65 USE_HAL_ETH_REGISTER_CALLBACKS

```
#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */
```

10.17.1.66 USE_HAL_FMPI2C_REGISTER_CALLBACKS

```
#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */
```

10.17.1.67 USE_HAL_FMPSMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */
```

10.17.1.68 USE_HAL_HASH_REGISTER_CALLBACKS

```
#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */
```

10.17.1.69 USE_HAL_HCD_REGISTER_CALLBACKS

```
#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */
```

10.17.1.70 USE_HAL_I2C_REGISTER_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */
```

10.17.1.71 USE_HAL_I2S_REGISTER_CALLBACKS

```
#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */
```

10.17.1.72 USE_HAL_IRDA_REGISTER_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */
```

10.17.1.73 USE_HAL_LPTIM_REGISTER_CALLBACKS

```
#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */
```

10.17.1.74 USE_HAL_LTDC_REGISTER_CALLBACKS

```
#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */
```

10.17.1.75 USE_HAL_MMC_REGISTER_CALLBACKS

```
#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */
```

10.17.1.76 USE_HAL_NAND_REGISTER_CALLBACKS

```
#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */
```

10.17.1.77 USE_HAL_NOR_REGISTER_CALLBACKS

```
#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */
```

10.17.1.78 USE_HAL_PCCARD_REGISTER_CALLBACKS

```
#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */
```

10.17.1.79 USE_HAL_PCD_REGISTER_CALLBACKS

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */
```

10.17.1.80 USE_HAL_QSPI_REGISTER_CALLBACKS

```
#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */
```

10.17.1.81 USE_HAL_RNG_REGISTER_CALLBACKS

```
#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */
```

10.17.1.82 USE_HAL_RTC_REGISTER_CALLBACKS

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */
```

10.17.1.83 USE_HAL_SAI_REGISTER_CALLBACKS

```
#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */
```

10.17.1.84 USE_HAL_SD_REGISTER_CALLBACKS

```
#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */
```

10.17.1.85 USE_HAL_SDRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */
```

10.17.1.86 USE_HAL_SMARTCARD_REGISTER_CALLBACKS

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
```

10.17.1.87 USE_HAL_SMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */
```

10.17.1.88 USE_HAL_SPDIFRX_REGISTER_CALLBACKS

```
#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */
```

10.17.1.89 USE_HAL_SPI_REGISTER_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */
```

10.17.1.90 USE_HAL_SRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */
```

10.17.1.91 USE_HAL_TIM_REGISTER_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */
```

10.17.1.92 USE_HAL_UART_REGISTER_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */
```

10.17.1.93 USE_HAL_USART_REGISTER_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */
```

10.17.1.94 USE_HAL_WWDG_REGISTER_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */
```

10.17.1.95 USE_RTOS

```
#define USE_RTOS 0U
```

10.17.1.96 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

10.17.1.97 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

10.18 Ball_Launcher_Main/Core/Inc/stm32f4xx_it.h File Reference

This file contains the headers of the interrupt handlers.

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 Pre-fetch 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 **TIM4_IRQHandler** (void)
  This function handles TIM4 global interrupt.
- void **USART1_IRQHandler** (void)
  This function handles USART1 global interrupt.

10.18.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

10.18.2 Function Documentation

10.18.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

10.18.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

10.18.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

10.18.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

10.18.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

10.18.2.6 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

10.18.2.7 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

10.18.2.8 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

10.18.2.9 TIM4_IRQHandler()

```
void TIM4_IRQHandler (
            void  )
```

This function handles TIM4 global interrupt.

10.18.2.10 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

10.18.2.11 USART1_IRQHandler()

```
void USART1_IRQHandler (
            void  )
```

This function handles USART1 global interrupt.

10.19 Ball_Launcher_Main/Core/Inc/switch.h File Reference

Header for **switch.c** (p. 117) file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

Data Structures

- struct **SwitchX**

  Structure to define a switch with GPIO port, pin, and status.

Functions

- void **Switch_init** (**SwitchX** ∗switchx, GPIO_TypeDef ∗GPIO, uint16_t PIN)

  Initializes the switch with the specified GPIO port and pin.
- uint8_t **Switch_getStatus** (**SwitchX** ∗switchx)

  Gets the status of the switch.

10.19.1 Detailed Description

Header for **switch.c** (p. 117) file. This file contains the common defines of the application.

Created on: May 17, 2024

Author: vvinh

10.19.2 Function Documentation

10.19.2.1 Switch_getStatus()

```
uint8_t Switch_getStatus (
              SwitchX * switchx )
```

Gets the status of the switch.

Parameters

switchx	Pointer to the **SwitchX** (p. 31) structure

Returns

uint8_t Status of the switch (1 if pressed, 0 if not pressed)

10.19.2.2 Switch_init()

```
void Switch_init (
              SwitchX * switchx,
             GPIO_TypeDef * GPIO,
             uint16_t PIN )
```

Initializes the switch with the specified GPIO port and pin.

Parameters

switchx	Pointer to the **SwitchX** (p. 31) structure
GPIO	GPIO port
PIN	GPIO pin

< Assign the GPIO port

< Assign the GPIO pin

< Initialize the status to 0

10.20 Ball_Launcher_Main/Core/Src/d4215.c File Reference

Source file for D4215 motor operations. This file contains the implementation of the functions for initializing and controlling a D4215 motor.

```
#include "d4215.h"
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <stdint.h>
```

Functions

- void **D4215_init** (**D4215X** *bldcx, TIM_HandleTypeDef *timer, uint32_t channel)
  Initializes the D4215 motor with the specified timer and channel.
- void **D4215_set** (**D4215X** *bldcx, int32_t spd)
  Sets the speed of the D4215 motor.

10.20.1 Detailed Description

Source file for D4215 motor operations. This file contains the implementation of the functions for initializing and controlling a D4215 motor.

Created on: May 1, 2024

Author: vvinh

10.20.2 Function Documentation

10.20.2.1 D4215_init()

```
void D4215_init (
              D4215X * bldcx,
             TIM_HandleTypeDef * timer,
             uint32_t channel )
```

Initializes the D4215 motor with the specified timer and channel.

Parameters

bldcx	Pointer to the **D4215X** (p. 25) structure
timer	Timer handle
channel	Timer channel

< Start PWM for the specified timer and channel

10.20.2.2 D4215_set()

```
void D4215_set (
              D4215X * bldcx,
             int32_t spd )
```

Sets the speed of the D4215 motor.

Parameters

bldcx	Pointer to the **D4215X** (p. 25) structure
spd	Speed of the motor (range: 5 to 10)

< Calculate the duty cycle based on the input speed

< Set the duty cycle for the PWM

10.21 Ball_Launcher_Main/Core/Src/led.c File Reference

Source file for LED operations. This file contains the implementation of the functions for initializing and controlling an LED.

```
#include "led.h"
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <stdint.h>
```

Functions

- void **LED_init** (**LEDX** *LEDx, GPIO_TypeDef *GPIO, uint16_t PIN)

  Initializes the LED with the specified GPIO port and pin.
- void **LED_on** (**LEDX** *LEDx)

  Turns on the LED.
- void **LED_off** (**LEDX** *LEDx)

  Turns off the LED.
- void **LED_toggle** (**LEDX** *LEDx)

  Toggles the LED state.

10.21.1 Detailed Description

Source file for LED operations. This file contains the implementation of the functions for initializing and controlling an LED.

Created on: May 17, 2024

Author: vvinh

10.21.2 Function Documentation

10.21.2.1 LED_init()

```
void LED_init (
              LEDX * LEDx,
             GPIO_TypeDef * GPIO,
             uint16_t PIN )
```

Initializes the LED with the specified GPIO port and pin.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure
GPIO	GPIO port
PIN	GPIO pin

< Assign the GPIO port

< Assign the GPIO pin

10.21.2.2 LED_off()

```
void LED_off (
              LEDX * LEDx )
```

Turns off the LED.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure

< Set the pin low to turn off the LED

10.21.2.3 LED_on()

```
void LED_on (
              LEDX * LEDx )
```

Turns on the LED.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure

< Set the pin high to turn on the LED

10.21.2.4 LED_toggle()

```
void LED_toggle (
              LEDX * LEDx )
```

Toggles the LED state.

Parameters

LEDx	Pointer to the **LEDX** (p. 26) structure

< Toggle the pin to change the LED state

10.22 Ball_Launcher_Main/Core/Src/main.c File Reference

Main program body.

```
#include "main.h"
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include "stm32f4xx_hal.h"
#include "stepper_driver.h"
#include "switch.h"
#include "d4215.h"
#include "led.h"
```

Functions

- void **SystemClock_Config** (void)

  System Clock Configuration.
- void **HAL_UART_RxCpltCallback** (UART_HandleTypeDef *huart)
- void **dataProcess** (char * **data**)
- uint16_t **speedCalculate** (uint16_t first, uint16_t second, uint16_t third)
- uint16_t **map** (uint16_t x, uint16_t in_min, uint16_t in_max, uint16_t out_min, uint16_t out_max)
- int **main** (void)

  The application entry point.
- void **HAL_TIM_IC_CaptureCallback** (TIM_HandleTypeDef *htim)
- void **Error_Handler** (void)

  This function is executed in case of error occurrence.

Variables

- TIM_HandleTypeDef **htim2**
- TIM_HandleTypeDef **htim4**
- UART_HandleTypeDef **huart1**
- UART_HandleTypeDef **huart6**
- char **charIn**
- char **data** [20]
- uint32_t **ti** = 0
- uint32_t **tf** = 0
- uint32_t **dt** = 0
- uint16_t **yaw** = 0
- uint16_t **pitch** = 0
- uint16_t **angleTarget** = 0
- uint16_t **rise** = 0
- uint16_t **fall** = 0
- uint16_t **total** = 0
- uint8_t **yawDirection** = 0
- uint8_t **pitchDirection** = 0
- uint8_t **yawDirectionSW** = 0
- uint8_t **pitchDirectionSW** = 0
- uint8_t **idx** = 0
- uint8_t **MOVE** = 0
- uint8_t **SHOT** = 0
- uint8_t **ESTOP** = 0
- uint8_t **CAL** = 0
- uint8_t **HOME** = 0
- uint8_t **SW1** = 0
- uint8_t **SW2** = 0
- uint8_t **SW3** = 0
- uint8_t **SW** = 0
- uint8_t **STATE** = 0
- uint8_t **STATE_0_INIT** = 0
- uint8_t **STATE_1_HUB** = 1
- uint8_t **STATE_2_ESTOP** = 2
- uint8_t **STATE_3_STEPPER** = 3
- uint8_t **STATE_4_BLDC** = 4

10.22.1 Detailed Description

Main program body.

Attention

10.22.2 Function Documentation

10.22.2.1 dataProcess()

```
void dataProcess (
            char * data )
```

10.22.2.2 Error_Handler()

```
void Error_Handler (
            void  )
```

This function is executed in case of error occurrence.

Return values

None	

10.22.2.3 HAL_TIM_IC_CaptureCallback()

```
void HAL_TIM_IC_CaptureCallback (
            TIM_HandleTypeDef * htim )
```

10.22.2.4 HAL_UART_RxCpltCallback()

```
void HAL_UART_RxCpltCallback (
            UART_HandleTypeDef * huart )
```

10.22.2.5 main()

```
int main (
            void  )
```

The application entry point.

Return values

int	

10.22.2.6 map()

```
uint16_t map (
            uint16_t x,
            uint16_t in_min,
            uint16_t in_max,
```

```
            uint16_t out_min,
            uint16_t out_max )
```

10.22.2.7 speedCalculate()

```
uint16_t speedCalculate (
            uint16_t first,
            uint16_t second,
            uint16_t third )
```

10.22.2.8 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

10.22.3 Variable Documentation

10.22.3.1 angleTarget

```
uint16_t angleTarget = 0
```

10.22.3.2 CAL

```
uint8_t CAL = 0
```

10.22.3.3 charIn

```
char charIn
```

10.22.3.4 data

```
char data[20]
```

10.22.3.5 dt

```
uint32_t dt = 0
```

10.22.3.6 ESTOP

```
uint8_t ESTOP = 0
```

10.22.3.7 fall

```
uint16_t fall = 0
```

10.22.3.8 HOME

```
uint8_t HOME = 0
```

10.22.3.9 htim2

```
TIM_HandleTypeDef htim2
```

10.22.3.10 htim4

```
TIM_HandleTypeDef htim4
```

10.22.3.11 huart1

```
UART_HandleTypeDef huart1
```

10.22.3.12 huart6

```
UART_HandleTypeDef huart6
```

10.22.3.13 idx

```
uint8_t idx = 0
```

10.22.3.14 MOVE

```
uint8_t MOVE = 0
```

10.22.3.15 pitch

```
uint16_t pitch = 0
```

10.22.3.16 pitchDirection

```
uint8_t pitchDirection = 0
```

10.22.3.17 pitchDirectionSW

```
uint8_t pitchDirectionSW = 0
```

10.22.3.18 rise

```
uint16_t rise = 0
```

10.22.3.19 SHOT

```
uint8_t SHOT = 0
```

10.22.3.20 STATE

```
uint8_t STATE = 0
```

10.22.3.21 STATE_0_INIT

```
uint8_t STATE_0_INIT = 0
```

10.22.3.22 STATE_1_HUB

```
uint8_t STATE_1_HUB = 1
```

10.22.3.23 STATE_2_ESTOP

```
uint8_t STATE_2_ESTOP = 2
```

10.22.3.24 STATE_3_STEPPER

```
uint8_t STATE_3_STEPPER = 3
```

10.22.3.25 STATE_4_BLDC

```
uint8_t STATE_4_BLDC = 4
```

10.22.3.26 SW

```
uint8_t SW = 0
```

10.22.3.27 SW1

```
uint8_t SW1 = 0
```

10.22.3.28 SW2

```
uint8_t SW2 = 0
```

10.22.3.29 SW3

```
uint8_t SW3 = 0
```

10.22.3.30 tf

```
uint32_t tf = 0
```

10.22.3.31 ti

```
uint32_t ti = 0
```

10.22.3.32 total

```
uint16_t total = 0
```

10.22.3.33 yaw

```
uint16_t yaw = 0
```

10.22.3.34 yawDirection

```
uint8_t yawDirection = 0
```

10.22.3.35 yawDirectionSW

```
uint8_t yawDirectionSW = 0
```

10.23 Ball_Launcher_Main/Core/Src/radio_driver.c File Reference

Source file for radio driver operations. This file contains the implementation of the functions for initializing and handling a radio driver.

```
#include "radio_driver.h"
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <stdint.h>
```

Functions

- void **Radio_init** (**RadioX** ∗radio, TIM_HandleTypeDef ∗timer, uint32_t channel1, uint32_t channel2)

  Initializes the radio driver with the specified timer and channels.
- void **capturePulseWidth** (**RadioX** ∗radio)

  Captures the pulse width for the radio driver.
- void **update** (**RadioX** ∗radio, int pw1, int pw2)

  Updates the radio driver with pulse width values.
- double **Radio_getPulseWidth** (**RadioX** ∗radio)

  Gets the pulse width of the radio driver.

10.23.1 Detailed Description

Source file for radio driver operations. This file contains the implementation of the functions for initializing and handling a radio driver.

Created on: May 2, 2024

Author: vvinh

10.23.2 Function Documentation

10.23.2.1 capturePulseWidth()

```
void capturePulseWidth (
            RadioX * radio )
```

Captures the pulse width for the radio driver.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure

< Read captured value for channel 1

< Read captured value for channel 2

< Calculate the pulse width

10.23.2.2 Radio_getPulseWidth()

```
double Radio_getPulseWidth (
              RadioX * radio )
```

Gets the pulse width of the radio driver.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure

Returns

double Pulse width value in milliseconds

10.23.2.3 Radio_init()

```
void Radio_init (
              RadioX * radio,
             TIM_HandleTypeDef * timer,
             uint32_t channel1,
             uint32_t channel2 )
```

Initializes the radio driver with the specified timer and channels.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure
timer	Timer handle
channel1	Timer channel 1
channel2	Timer channel 2

< Start input capture interrupt for channel 1

< Start input capture interrupt for channel 2

10.23.2.4 update()

```
void update (
              RadioX * radio,
             int pw1,
             int pw2 )
```

Updates the radio driver with pulse width values.

Parameters

radio	Pointer to the **RadioX** (p. 28) structure
pw1	Pulse width value for channel 1
pw2	Pulse width value for channel 2

10.24 Ball_Launcher_Main/Core/Src/stepper_driver.c File Reference

Source file for stepper motor driver operations. This file contains the implementation of the functions for initializing and controlling a stepper motor.

```
#include "stepper_driver.h"
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <stdint.h>
```

Functions

- void **Stepper_init** (**StepperX** ∗stepperx, GPIO_TypeDef ∗GPIO, uint16_t EN_PIN, uint16_t DIR_PIN, uint16_t STP_PIN)

  Initializes the stepper motor driver with the specified GPIO port and pins.
- void **Stepper_enable** (**StepperX** ∗stepperx)

  Enables the stepper motor driver.
- void **Stepper_disable** (**StepperX** ∗stepperx)

  Disables the stepper motor driver.
- void **Stepper_setspeed** (**StepperX** ∗stepperx, uint16_t speed, uint8_t dir)

  Sets the speed and direction of the stepper motor.
- void **SysTick_Init** (void)

  Initializes the system tick for microsecond delays.
- void **Delay_us** (uint32_t us)

  Delays the program execution for a specified number of microseconds.

10.24.1 Detailed Description

Source file for stepper motor driver operations. This file contains the implementation of the functions for initializing and controlling a stepper motor.

Created on: May 17, 2024

Author: vvinh

10.24.2 Function Documentation

10.24.2.1 Delay_us()

```
void Delay_us (
            uint32_t us )
```

Delays the program execution for a specified number of microseconds.

Parameters

us	Number of microseconds to delay

10.24.2.2 Stepper_disable()

```
void Stepper_disable (
              StepperX * stepperx )
```

Disables the stepper motor driver.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure

10.24.2.3 Stepper_enable()

```
void Stepper_enable (
              StepperX * stepperx )
```

Enables the stepper motor driver.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure

10.24.2.4 Stepper_init()

```
void Stepper_init (
              StepperX * stepperx,
             GPIO_TypeDef * GPIO,
             uint16_t EN_PIN,
             uint16_t DIR_PIN,
             uint16_t STP_PIN )
```

Initializes the stepper motor driver with the specified GPIO port and pins.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure
GPIO	GPIO port
EN_PIN	Enable pin
DIR_PIN	Direction pin
STP_PIN	Step pin

10.24.2.5 Stepper_setspeed()

```
void Stepper_setspeed (
              StepperX * stepperx,
             uint16_t speed,
             uint8_t dir )
```

Sets the speed and direction of the stepper motor.

Parameters

stepperx	Pointer to the **StepperX** (p. 30) structure
speed	Speed of the stepper motor (delay between steps)
dir	Direction of the stepper motor (1 for one direction, 0 for the opposite)

< Set the direction pin

< Set the step pin high

< Delay to control the speed

< Set the step pin low

< Delay to control the speed

10.24.2.6 SysTick_Init()

```
void SysTick_Init (
             void  )
```

Initializes the system tick for microsecond delays.

Initializes the system tick for delay functions.

10.25 Ball_Launcher_Main/Core/Src/stm32f4xx_hal_msp.c File Reference

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

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

Functions

- void **HAL_TIM_MspPostInit** (TIM_HandleTypeDef ∗htim)
- void **HAL_MspInit** (void)
- void **HAL_TIM_PWM_MspInit** (TIM_HandleTypeDef ∗htim_pwm)

  TIM_PWM MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_TIM_IC_MspInit** (TIM_HandleTypeDef ∗htim_ic)

  TIM_IC MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_TIM_PWM_MspDeInit** (TIM_HandleTypeDef ∗htim_pwm)

  TIM_PWM MSP De-Initialization This function freeze the hardware resources used in this example.
- void **HAL_TIM_IC_MspDeInit** (TIM_HandleTypeDef ∗htim_ic)

  TIM_IC MSP De-Initialization This function freeze the hardware resources used in this example.
- void **HAL_UART_MspInit** (UART_HandleTypeDef ∗huart)

  UART MSP Initialization This function configures the hardware resources used in this example.
- void **HAL_UART_MspDeInit** (UART_HandleTypeDef ∗huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.

10.25.1 Detailed Description

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

Attention

10.25.2 Function Documentation

10.25.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void  )
```

Initializes the Global MSP.

10.25.2.2 HAL_TIM_IC_MspDeInit()

```
void HAL_TIM_IC_MspDeInit (
            TIM_HandleTypeDef * htim_ic )
```

TIM_IC MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

htim_ic	TIM_IC handle pointer

Return values

None	

TIM4 GPIO Configuration PB6 ------> TIM4_CH1

10.25.2.3 HAL_TIM_IC_MspInit()

```
void HAL_TIM_IC_MspInit (
            TIM_HandleTypeDef * htim_ic )
```

TIM_IC MSP Initialization This function configures the hardware resources used in this example.

Parameters

htim_ic	TIM_IC handle pointer

Return values

None	

TIM4 GPIO Configuration PB6 ------> TIM4_CH1

10.25.2.4 HAL_TIM_MspPostInit()

```
void HAL_TIM_MspPostInit (
            TIM_HandleTypeDef * htim )
```

TIM2 GPIO Configuration PA1 ------> TIM2_CH2 PA2 ------> TIM2_CH3

10.25.2.5 HAL_TIM_PWM_MspDeInit()

```
void HAL_TIM_PWM_MspDeInit (
            TIM_HandleTypeDef * htim_pwm )
```

TIM_PWM MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

htim_pwm	TIM_PWM handle pointer

Return values

None	

10.25.2.6 HAL_TIM_PWM_MspInit()

```
void HAL_TIM_PWM_MspInit (
            TIM_HandleTypeDef * htim_pwm )
```

TIM_PWM MSP Initialization This function configures the hardware resources used in this example.

Parameters

htim_pwm	TIM_PWM handle pointer

Return values

None	

10.25.2.7 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART1 GPIO Configuration PA9 ------> USART1_TX PA10 ------> USART1_RX

USART6 GPIO Configuration PA11 ------> USART6_TX PA12 ------> USART6_RX

10.25.2.8 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART1 GPIO Configuration PA9 ------> USART1_TX PA10 ------> USART1_RX

USART6 GPIO Configuration PA11 ------> USART6_TX PA12 ------> USART6_RX

10.26 Ball_Launcher_Main/Core/Src/stm32f4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_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 Pre-fetch 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 **TIM4_IRQHandler** (void)

  This function handles TIM4 global interrupt.
- void **USART1_IRQHandler** (void)

  This function handles USART1 global interrupt.

Variables

- TIM_HandleTypeDef **htim4**
- UART_HandleTypeDef **huart1**

10.26.1 Detailed Description

Interrupt Service Routines.

Attention

10.26.2 Function Documentation

10.26.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

10.26.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

10.26.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

10.26.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

10.26.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

10.26.2.6 PendSV_Handler()

```
void PendSV_Handler (
             void  )
```

This function handles Pendable request for system service.

10.26.2.7 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

10.26.2.8 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

10.26.2.9 TIM4_IRQHandler()

```
void TIM4_IRQHandler (
             void  )
```

This function handles TIM4 global interrupt.

10.26.2.10 UsageFault_Handler()

```
void UsageFault_Handler (
             void  )
```

This function handles Undefined instruction or illegal state.

10.26.2.11 USART1_IRQHandler()

```
void USART1_IRQHandler (
             void  )
```

This function handles USART1 global interrupt.

10.26.3 Variable Documentation

10.26.3.1 htim4

```
TIM_HandleTypeDef htim4  [extern]
```

10.26.3.2 huart1

```
UART_HandleTypeDef huart1  [extern]
```

10.27 Ball_Launcher_Main/Core/Src/switch.c File Reference

Source file for switch operations. This file contains the implementation of the functions for initializing and getting the status of a switch.

```
#include "switch.h"
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <stdint.h>
```

Functions

- void **Switch_init** (**SwitchX** ∗switchx, GPIO_TypeDef ∗GPIO, uint16_t PIN)

  Initializes the switch with the specified GPIO port and pin.
- uint8_t **Switch_getStatus** (**SwitchX** ∗switchx)

  Gets the status of the switch.

10.27.1 Detailed Description

Source file for switch operations. This file contains the implementation of the functions for initializing and getting the status of a switch.

Created on: May 17, 2024

Author: vvinh

10.27.2 Function Documentation

10.27.2.1 Switch_getStatus()

```
uint8_t Switch_getStatus (
              SwitchX * switchx )
```

Gets the status of the switch.

Parameters

switchx	Pointer to the **SwitchX** (p. 31) structure

Returns

uint8_t Status of the switch (1 if pressed, 0 if not pressed)

10.27.2.2 Switch_init()

```
void Switch_init (
              SwitchX * switchx,
             GPIO_TypeDef * GPIO,
             uint16_t PIN )
```

Initializes the switch with the specified GPIO port and pin.

Parameters

switchx	Pointer to the **SwitchX** (p. 31) structure
GPIO	GPIO port
PIN	GPIO pin

< Assign the GPIO port

< Assign the GPIO pin

< Initialize the status to 0

10.28 Ball_Launcher_Main/Core/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Functions

- int **__io_putchar** (int ch) **__attribute__**((weak))
- int **__io_getchar** (void)
- void **initialise_monitor_handles** ()
- int **_getpid** (void)
- int **_kill** (int pid, int sig)
- void **_exit** (int status)
- **__attribute__** ((weak))
- int **_close** (int file)
- int **_fstat** (int file, struct **stat** *st)

- int **_isatty** (int file)
- int **_lseek** (int file, int ptr, int dir)
- int **_open** (char *path, int flags,...)
- int **_wait** (int *status)
- int **_unlink** (char *name)
- int **_times** (struct tms *buf)
- int **_stat** (char *file, struct **stat** *st)
- int **_link** (char *old, char *new)
- int **_fork** (void)
- int **_execve** (char *name, char **argv, char **env)

Variables

- char ** **environ** = __env

10.28.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

10.28.2 Function Documentation

10.28.2.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

10.28.2.2 __io_getchar()

```
int __io_getchar (
             void  )  [extern]
```

10.28.2.3 __io_putchar()

```
int __io_putchar (
            int ch )  [extern]
```

10.28.2.4 _close()

```
int _close (
            int file )
```

10.28.2.5 _execve()

```
int _execve (
            char * name,
            char ** argv,
            char ** env )
```

10.28.2.6 _exit()

```
void _exit (
            int status )
```

10.28.2.7 _fork()

```
int _fork (
            void  )
```

10.28.2.8 _fstat()

```
int _fstat (
            int file,
            struct  stat * st )
```

10.28.2.9 _getpid()

```
int _getpid (
            void  )
```

10.28.2.10 _isatty()

```
int _isatty (
            int file )
```

10.28.2.11 _kill()

```
int _kill (
              int pid,
              int sig )
```

10.28.2.12 _link()

```
int _link (
              char * old,
              char * new )
```

10.28.2.13 _lseek()

```
int _lseek (
              int file,
              int ptr,
              int dir )
```

10.28.2.14 _open()

```
int _open (
              char * path,
              int flags,
               ...  )
```

10.28.2.15 _stat()

```
int _stat (
              char * file,
              struct  stat * st )
```

10.28.2.16 _times()

```
int _times (
              struct tms * buf )
```

10.28.2.17 _unlink()

```
int _unlink (
              char * name )
```

10.28.2.18 _wait()

```
int _wait (
              int * status )
```

10.28.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

10.28.3 Variable Documentation

10.28.3.1 environ

```
char** environ = __env
```

10.29 Ball_Launcher_Main/Core/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

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

Functions

- void * **_sbrk** (ptrdiff_t incr)

  _sbrk() *(*p. *123) allocates memory to the newlib heap and is used by malloc and others from the C library*

10.29.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

10.29.2 Function Documentation

10.29.2.1 _sbrk()

```
void * _sbrk (
             ptrdiff_t incr )
```

_sbrk() (p. 123) allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ##########################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ##########################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

Pointer to allocated memory

10.30 Ball_Launcher_Main/Core/Src/system_stm32f4xx.c File Reference

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

```
#include "stm32f4xx.h"
```

Macros

- #define **HSE_VALUE** ((uint32_t)25000000)
- #define **HSI_VALUE** ((uint32_t)16000000)

Functions

- void **SystemInit** (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.
- 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** = 16000000
- const uint8_t **AHBPrescTable** [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8] = {0, 0, 0, 0, 1, 2, 3, 4}

10.30.1 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()** (p. 22): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f4xx.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()** (p. 22): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

10.31 pages/controller.c File Reference

Controller Component Documentation.

10.31.1 Detailed Description

Controller Component Documentation.

10.32 pages/launcher.c File Reference

Launcher Component Documentation.

10.32.1 Detailed Description

Launcher Component Documentation.

10.33 pages/mainpage.c File Reference

Main Page Documentation for Project.

10.33.1 Detailed Description

Main Page Documentation for Project.

10.34 pages/report.c File Reference

Launcher Documentation.

10.34.1 Detailed Description

Launcher Documentation.

Index