RobotROS

Generated by Doxygen 1.9.8

Chapter 1

Robot ROS

1.1 Principal function

function main : init function and start kernel.
function microros_task : Create the publishers and the subscribers and exploit them.
function task_Motor_Left : Control the left motor.
function task_Motor_Right : Control the right motor.
function task_VL53 : Get the VL53 measure and put it in the queue.
function task_Grove_LCD : Get the information from rhe queue and print it on the LCD.
function task_Supervision : The brain's robot decide of the action depending of data receive from microROS.

1.2 Secondary function

function createPublisher : use to create a default publisher.
function createSubscriber : use to create a default subscriber.
function CHECKMRRET : Test if a microRos function success else print errror message.
function SubscriberCallbackFunction : callback call when message is receive.

1.3 Test function

function test_uart2 : Test printf and scanf.
function test_vl53 : Test VL53 sensors.
function test_motor : Test correcteur.

1.4 Config define

1.4.1 Config exo

1.4.2 Config param

Author

titouan melon

Attention

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 AMessage Struct Reference

Data Fields

- char command
- int data

4.1.1 Detailed Description

Use to send data to lcd's task

Definition at line 110 of file main.c.

4.1.2 Field Documentation

4.1.2.1 command

```
char command
```

Represent the direction of the robot

Definition at line 112 of file main.c.

4.1.2.2 data

```
int data
```

Represent the mode of the robot

Definition at line 113 of file main.c.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Src/main.c

4.2 MicroRosPubMsg Struct Reference

Data Fields

- char dir
- int mode
- int speed

4.2.1 Detailed Description

Use to send information from the task decision to microRos task

Definition at line 119 of file main.c.

4.2.2 Field Documentation

4.2.2.1 dir

```
char dir
```

Represent the direction of the robot

Definition at line 121 of file main.c.

4.2.2.2 mode

```
int mode
```

Represent the mode of the robot

Definition at line 122 of file main.c.

4.2.2.3 speed

```
int speed
```

Represent the speed of the robot

Definition at line 123 of file main.c.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Src/main.c

4.3 MicroRosSubMsg Struct Reference

Data Fields

- int dir
- int x
- int y
- int mode
- int speed

4.3.1 Detailed Description

Use to send information get by microRos to decision task

Definition at line 129 of file main.c.

4.3.2 Field Documentation

4.3.2.1 dir

```
int dir
```

Represent the direction send by the IHM

Definition at line 131 of file main.c.

4.3.2.2 mode

```
int mode
```

Represent the mode send by the IHM

Definition at line 134 of file main.c.

4.3.2.3 speed

```
int speed
```

Represent the speed send by the IHM

Definition at line 135 of file main.c.

4.3.2.4 x

```
int x
```

Represent the x position send by the camera

Definition at line 132 of file main.c.

4.3.2.5 y

```
int y
```

Represent the y position send by the camera

Definition at line 133 of file main.c.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/main.c

4.4 SequenceStepEnables Struct Reference

Data Fields

- uint8_t tcc
- uint8_t msrc
- uint8_t dss
- uint8_t pre_range
- uint8_t final_range

4.4.1 Detailed Description

Definition at line 291 of file VL53L0X.h.

4.4.2 Field Documentation

4.4.2.1 dss

```
uint8_t dss
```

Definition at line 292 of file VL53L0X.h.

4.4.2.2 final_range

```
uint8_t final_range
```

Definition at line 292 of file VL53L0X.h.

4.4.2.3 msrc

```
uint8_t msrc
```

Definition at line 292 of file VL53L0X.h.

4.4.2.4 pre_range

```
uint8_t pre_range
```

Definition at line 292 of file VL53L0X.h.

4.4.2.5 tcc

```
uint8_t tcc
```

Definition at line 292 of file VL53L0X.h.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/VL53L0X.h

4.5 SequenceStepTimeouts Struct Reference

Data Fields

- uint16_t pre_range_vcsel_period_pclks
- uint16_t final_range_vcsel_period_pclks
- uint16_t msrc_dss_tcc_mclks
- uint16_t pre_range_mclks
- uint16_t final_range_mclks
- uint32_t msrc_dss_tcc_us
- uint32_t pre_range_us
- uint32_t final_range_us

4.5.1 Detailed Description

Definition at line 295 of file VL53L0X.h.

4.5.2 Field Documentation

4.5.2.1 final_range_mclks

```
uint16_t final_range_mclks
```

Definition at line 298 of file VL53L0X.h.

4.5.2.2 final_range_us

```
uint32_t final_range_us
```

Definition at line 299 of file VL53L0X.h.

4.5.2.3 final_range_vcsel_period_pclks

```
uint16_t final_range_vcsel_period_pclks
```

Definition at line 296 of file VL53L0X.h.

4.5.2.4 msrc_dss_tcc_mclks

```
uint16_t msrc_dss_tcc_mclks
```

Definition at line 298 of file VL53L0X.h.

4.5.2.5 msrc_dss_tcc_us

```
uint32_t msrc_dss_tcc_us
```

Definition at line 299 of file VL53L0X.h.

4.5.2.6 pre_range_mclks

```
uint16_t pre_range_mclks
```

Definition at line 298 of file VL53L0X.h.

4.5.2.7 pre_range_us

```
uint32_t pre_range_us
```

Definition at line 299 of file VL53L0X.h.

4.5.2.8 pre_range_vcsel_period_pclks

```
uint16_t pre_range_vcsel_period_pclks
```

Definition at line 296 of file VL53L0X.h.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Inc/VL53L0X.h

4.6 statInfo_t Struct Reference

Data Fields

- uint16_t rawDistance
- uint16_t signalCnt
- uint16_t ambientCnt
- uint16_t spadCnt
- uint8_t rangeStatus

4.6.1 Detailed Description

Definition at line 193 of file VL53L0X.h.

4.6.2 Field Documentation

4.6.2.1 ambientCnt

```
uint16_t ambientCnt
```

Definition at line 196 of file VL53L0X.h.

4.6.2.2 rangeStatus

```
uint8_t rangeStatus
```

Definition at line 198 of file VL53L0X.h.

4.6.2.3 rawDistance

```
uint16_t rawDistance
```

Definition at line 194 of file VL53L0X.h.

4.6.2.4 signalCnt

```
uint16_t signalCnt
```

Definition at line 195 of file VL53L0X.h.

4.6.2.5 spadCnt

```
uint16_t spadCnt
```

Definition at line 197 of file VL53L0X.h.

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

- C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base↩_robot/Core/Inc/VL53L0X.h

Chapter 5

File Documentation

5.1 captDistIR.h

```
00001 /*
00002  * IRMeasure.h
00003  */
00004
00005 #ifndef INC_CAPTDISTIR_H_
00006 #define INC_CAPTDISTIR_H_
00007
00008 #include "main.h"
00009
00010
00011 void captDistIR_Init(void);
00012 int captDistIR_Get(int*);
00013
00014
00015 #endif /* INC_CAPTDISTIR_H_ */
```

5.2 config.h

```
00001 /*
00002  * config.h
00003  */
00004
00005 #ifndef INC_CONFIG_H_
00006 #define INC_CONFIG_H_
00007
00008 //=======================================================
00009 // USART : CHOIX DE LA LIAISON SERIE
00010 // USART2 : USART_STLINK (cable)
00011 // USART6 : USART_ZIGBEE (sans fil)
00012 //=======================================================
00013 #define USE_USART_STLINK 1          // A Commenter pour utiliser stlink dans term_printf !! faire un
      clean
00014 //#define   USE_USART_ZIGBEE 1
00015
00016 #define NB_CAR_TO_RECEIVE       1       // nombre de caractères à recevoir pour déclencher une
      interruption
00017
00018 #define USART2_BAUDRATE         115200
00019 #define USART6_BAUDRATE         9600
00020 //=======================================================
00021 //                  LIAISON I2C
00022 //=======================================================
00023 #define I2C1_CLOCKSPEED         100000
00024 #define I2C2_CLOCKSPEED         100000
00025
00026 // CAPTEUR I2C DISTANCE ULTRASON SRF02
00027 #define CAPT_US_LEFT_ADDRESS    0xE0
00028 #define CAPT_US_RIGHT_ADDRESS   0xE2
00029
00030 // IMU MPU9250
00031 #define    MPU9250_ADDRESS      0x68
00032 #define    AK8963_ADDRESS       0x0C
00033
00034 // ECRAN LCD
```

```
#define SCREEN_LCD_ADDRESS      (0x30«1)

// SAMPLING PERIOD in ms
#define SAMPLING_PERIOD_ms      5

#endif /* INC_CONFIG_H_ */

/*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*/
```

5.3 drv_gpio.h

```
/*
 * drv_gpio.h
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */

#ifndef INC_DRV_GPIO_H_
#define INC_DRV_GPIO_H_

void MX_GPIO_Init(void);

#endif /* INC_DRV_GPIO_H_ */
```

5.4 drv_i2c.h

```
/*
 * drv_i2c.h
 *
 *  Created on: Mar 13, 2023
 *      Author: kerhoas
 */

#ifndef INC_DRV_I2C_H_
#define INC_DRV_I2C_H_

void MX_I2C1_Init(void);

// Transmit n_data bytes to i2c slave
int i2c1_WriteBuffer(uint16_t addrSlave, uint8_t *data, int n_data);

// Receive n_data bytes from i2c slave
int i2c1_ReadBuffer(uint16_t addrSlave, uint8_t *data, int n_data);

// Receive n_data bytes - located at regAddr - from i2c slave
int i2c1_ReadRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data);

// Write n_data bytes - have to be written at regAddr - to i2c slave
int i2c1_WriteRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data);

// Write 1 byte at regAddr Slave - Interrupt Method
void i2c1_WriteRegByte_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t data);

// Read 1 byte from regAddr Slave - Interrupt Method
void i2c1_ReadRegBuffer_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t* datas, int len);

// Write 1 byte to regAddr (16 bits) Slave
int i2c1_WriteReg16Byte(uint16_t addrSlave, uint16_t  regAddr,  uint8_t data);

// Write 16 bits word to regAddr (16 bits) Slave
int i2c1_WriteReg16Word16(uint16_t addrSlave, uint16_t  regAddr,  uint16_t data);

// Write 32 bits word to regAddr (16 bits) Slave
int i2c1_WriteReg16Word32(uint16_t addrSlave, uint16_t  regAddr,  uint32_t data);

// Read 1 byte from regAddr (16 bits) Slave
int i2c1_ReadReg16Byte(uint16_t addrSlave, uint16_t  regAddr, uint8_t *data);

// Read 16 bits word from regAddr (16 bits) Slave
int i2c1_ReadReg16Word16(uint16_t addrSlave, uint16_t  regAddr, uint16_t *data);

// Read 32 bits word from regAddr (16 bits) Slave
int i2c1_ReadReg16Word32(uint16_t addrSlave, uint16_t  regAddr, uint32_t *data);

// Read n_data bytes from regAddr (16 bits) Slave
int i2c1_ReadReg16Buffer(uint16_t addrSlave, uint16_t  regAddr,  uint8_t *data, int n_data);

#endif /* INC_DRV_I2C_H_ */
```

5.5 drv_uart.h

```
00001 /*
00002  * drv_uart.h
00003  *
00004  *  Created on: Mar 13, 2023
00005  *      Author: kerhoas
00006  */
00007
00008 #ifndef INC_DRV_UART_H_
00009 #define INC_DRV_UART_H_
00010
00011 void MX_USART1_UART_Init(void);
00012 void MX_USART2_UART_Init(void);
00013 void MX_DMA_Init(void);
00014
00015
00016
00017
00018 #endif /* INC_DRV_UART_H_ */
```

5.6 FreeRTOSConfig.h

```
00001 /* USER CODE BEGIN Header */
00002 /*
00003  * FreeRTOS Kernel V10.3.1
00004  * Portion Copyright (C) 2017 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
00005  * Portion Copyright (C) 2019 StMicroelectronics, Inc.  All Rights Reserved.
00006  *
00007  * Permission is hereby granted, free of charge, to any person obtaining a copy of
00008  * this software and associated documentation files (the "Software"), to deal in
00009  * the Software without restriction, including without limitation the rights to
00010  * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
00011  * the Software, and to permit persons to whom the Software is furnished to do so,
00012  * subject to the following conditions:
00013  *
00014  * The above copyright notice and this permission notice shall be included in all
00015  * copies or substantial portions of the Software.
00016  *
00017  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00018  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
00019  * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
00020  * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
00021  * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
00022  * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00023  *
00024  * http://www.FreeRTOS.org
00025  * http://aws.amazon.com/freertos
00026  *
00027  * 1 tab == 4 spaces!
00028  */
00029 /* USER CODE END Header */
00030
00031 #ifndef FREERTOS_CONFIG_H
00032 #define FREERTOS_CONFIG_H
00033
00034 /*-----------------------------------------------------------
00035  * Application specific definitions.
00036  *
00037  * These definitions should be adjusted for your particular hardware and
00038  * application requirements.
00039  *
00040  * These parameters and more are described within the 'configuration' section of the
00041  * FreeRTOS API documentation available on the FreeRTOS.org web site.
00042  *
00043  * See http://www.freertos.org/a00110.html
00044  *----------------------------------------------------------*/
00045
00046 /* USER CODE BEGIN Includes */
00047 /* Section where include file can be added */
00048 /* USER CODE END Includes */
00049
00050 /* Ensure definitions are only used by the compiler, and not by the assembler. */
00051 #if defined(__ICCARM__) || defined(__CC_ARM) || defined(__GNUC__)
00052   #include <stdint.h>
00053   extern uint32_t SystemCoreClock;
00054 #endif
00055 #ifndef CMSIS_device_header
00056 #define CMSIS_device_header "stm32f4xx.h"
00057 #endif /* CMSIS_device_header */
00058
00059 #define configENABLE_FPU                        0
00060 #define configENABLE_MPU                        0
```

```
00061
00062 #define configUSE_PREEMPTION                    1
00063 #define configSUPPORT_STATIC_ALLOCATION         1
00064 #define configSUPPORT_DYNAMIC_ALLOCATION        1
00065 #define configUSE_IDLE_HOOK                     0
00066 #define configUSE_TICK_HOOK                     0
00067 #define configCPU_CLOCK_HZ                      ( SystemCoreClock )
00068 #define configTICK_RATE_HZ                      ((TickType_t)1000)
00069 #define configMAX_PRIORITIES                    ( 56 )
00070 #define configMINIMAL_STACK_SIZE                ((uint16_t)128)
00071 #define configTOTAL_HEAP_SIZE                   ((size_t)16384)//((size_t)15360)
00072 #define configMAX_TASK_NAME_LEN                 ( 16 )
00073 #define configUSE_TRACE_FACILITY                1
00074 #define configUSE_16_BIT_TICKS                  0
00075 #define configUSE_MUTEXES                       1
00076 #define configQUEUE_REGISTRY_SIZE               8
00077 #define configUSE_RECURSIVE_MUTEXES             1
00078 #define configUSE_COUNTING_SEMAPHORES           1
00079 #define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
00080 /* USER CODE BEGIN MESSAGE_BUFFER_LENGTH_TYPE */
00081 /* Defaults to size_t for backward compatibility, but can be changed
00082    if lengths will always be less than the number of bytes in a size_t. */
00083 #define configMESSAGE_BUFFER_LENGTH_TYPE        size_t
00084 /* USER CODE END MESSAGE_BUFFER_LENGTH_TYPE */
00085
00086 /* Co-routine definitions. */
00087 #define configUSE_CO_ROUTINES                   0
00088 #define configMAX_CO_ROUTINE_PRIORITIES         ( 2 )
00089
00090 /* Software timer definitions. */
00091 #define configUSE_TIMERS                        1
00092 #define configTIMER_TASK_PRIORITY               ( 2 )
00093 #define configTIMER_QUEUE_LENGTH                10
00094 #define configTIMER_TASK_STACK_DEPTH            256
00095
00096 /* The following flag must be enabled only when using newlib */
00097 #define configUSE_NEWLIB_REENTRANT          1
00098
00099 /* CMSIS-RTOS V2 flags */
00100 #define configUSE_OS2_THREAD_SUSPEND_RESUME  1
00101 #define configUSE_OS2_THREAD_ENUMERATE       1
00102 #define configUSE_OS2_EVENTFLAGS_FROM_ISR    1
00103 #define configUSE_OS2_THREAD_FLAGS           1
00104 #define configUSE_OS2_TIMER                  1
00105 #define configUSE_OS2_MUTEX                  1
00106
00107 /* Set the following definitions to 1 to include the API function, or zero
00108 to exclude the API function. */
00109 #define INCLUDE_vTaskPrioritySet             1
00110 #define INCLUDE_uxTaskPriorityGet            1
00111 #define INCLUDE_vTaskDelete                  1
00112 #define INCLUDE_vTaskCleanUpResources        0
00113 #define INCLUDE_vTaskSuspend                 1
00114 #define INCLUDE_vTaskDelayUntil              1
00115 #define INCLUDE_vTaskDelay                   1
00116 #define INCLUDE_xTaskGetSchedulerState       1
00117 #define INCLUDE_xTimerPendFunctionCall       1
00118 #define INCLUDE_xQueueGetMutexHolder         1
00119 #define INCLUDE_uxTaskGetStackHighWaterMark  1
00120 #define INCLUDE_xTaskGetCurrentTaskHandle    1
00121 #define INCLUDE_eTaskGetState                1
00122
00123 /*
00124  * The CMSIS-RTOS V2 FreeRTOS wrapper is dependent on the heap implementation used
00125  * by the application thus the correct define need to be enabled below
00126  */
00127 #define USE_FreeRTOS_HEAP_4
00128
00129 /* Cortex-M specific definitions. */
00130 #ifdef __NVIC_PRIO_BITS
00131  /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
00132  #define configPRIO_BITS         __NVIC_PRIO_BITS
00133 #else
00134  #define configPRIO_BITS         4
00135 #endif
00136
00137 /* The lowest interrupt priority that can be used in a call to a "set priority"
00138 function. */
00139 #define configLIBRARY_LOWEST_INTERRUPT_PRIORITY   15
00140
00141 /* The highest interrupt priority that can be used by any interrupt service
00142 routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
00143 INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
00144 PRIORITY THAN THIS! (higher priorities are lower numeric values. */
00145 #define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5
00146
00147 /* Interrupt priorities used by the kernel port layer itself.  These are generic
```

```
00148 to all Cortex-M ports, and do not rely on any particular library functions. */
00149 #define configKERNEL_INTERRUPT_PRIORITY         ( configLIBRARY_LOWEST_INTERRUPT_PRIORITY « (8 -
      configPRIO_BITS) )
00150 /* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
00151 See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
00152 #define configMAX_SYSCALL_INTERRUPT_PRIORITY    ( configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY « (8 -
      configPRIO_BITS) )
00153
00154 /* Normal assert() semantics without relying on the provision of an assert.h
00155 header file. */
00156 /* USER CODE BEGIN 1 */
00157 #define configASSERT( x ) if ((x) == 0) {taskDISABLE_INTERRUPTS(); for( ;; );}
00158 /* USER CODE END 1 */
00159
00160 /* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
00161 standard names. */
00162 #define vPortSVCHandler    SVC_Handler
00163 #define xPortPendSVHandler PendSV_Handler
00164
00165 /* IMPORTANT: After 10.3.1 update, Systick_Handler comes from NVIC (if SYS timebase = systick),
      otherwise from cmsis_os2.c */
00166
00167 #define USE_CUSTOM_SYSTICK_HANDLER_IMPLEMENTATION 0
00168
00169 /* USER CODE BEGIN Defines */
00170 /* Section where parameter definitions can be added (for instance, to override default ones in
      FreeRTOS.h) */
00171 /* USER CODE END Defines */
00172
00173 #endif /* FREERTOS_CONFIG_H */
```

5.7 groveLCD.h

```
00001 /*
00002  * groveLCD.h
00003  *
00004  *  Created on: Oct 16, 2019
00005  *      Author: kerhoas
00006  */
00007 #ifndef INC_GROVELCD_H_
00008 #define INC_GROVELCD_H_
00009
00010 #include "main.h"
00011
00012 // Device I2C Arress
00013 #define LCD_ADDRESS     (0x7c)
00014 #define RGB_ADDRESS     (0xc4)
00015
00016
00017 // color define
00018 #define WHITE           0
00019 #define RED             1
00020 #define GREEN           2
00021 #define BLUE            3
00022
00023 #define REG_RED         0x04        // pwm2
00024 #define REG_GREEN       0x03        // pwm1
00025 #define REG_BLUE        0x02        // pwm0
00026
00027 #define REG_MODE1       0x00
00028 #define REG_MODE2       0x01
00029 #define REG_OUTPUT      0x08
00030
00031 // commands
00032 #define LCD_CLEARDISPLAY 0x01
00033 #define LCD_RETURNHOME 0x02
00034 #define LCD_ENTRYMODESET 0x04
00035 #define LCD_DISPLAYCONTROL 0x08
00036 #define LCD_CURSORSHIFT 0x10
00037 #define LCD_FUNCTIONSET 0x20
00038 #define LCD_SETCGRAMADDR 0x40
00039 #define LCD_SETDDRAMADDR 0x80
00040
00041 // flags for display entry mode
00042 #define LCD_ENTRYRIGHT 0x00
00043 #define LCD_ENTRYLEFT 0x02
00044 #define LCD_ENTRYSHIFTINCREMENT 0x01
00045 #define LCD_ENTRYSHIFTDECREMENT 0x00
00046
00047 // flags for display on/off control
00048 #define LCD_DISPLAYON 0x04
00049 #define LCD_DISPLAYOFF 0x00
00050 #define LCD_CURSORON 0x02
```

```
00051 #define LCD_CURSOROFF 0x00
00052 #define LCD_BLINKON 0x01
00053 #define LCD_BLINKOFF 0x00
00054
00055 // flags for display/cursor shift
00056 #define LCD_DISPLAYMOVE 0x08
00057 #define LCD_CURSORMOVE 0x00
00058 #define LCD_MOVERIGHT 0x04
00059 #define LCD_MOVELEFT 0x00
00060
00061 // flags for function set
00062 #define LCD_8BITMODE 0x10
00063 #define LCD_4BITMODE 0x00
00064 #define LCD_2LINE 0x08
00065 #define LCD_1LINE 0x00
00066 #define LCD_5x10DOTS 0x04
00067 #define LCD_5x8DOTS 0x00
00068
00069 void groveLCD_test();
00070 void groveLCD_begin(uint8_t cols, uint8_t lines, uint8_t dotsize);
00071 void groveLCD_setColorAll();
00072 void groveLCD_setColorWhite();
00073 void groveLCD_clear();
00074 void groveLCD_home();
00075 void groveLCD_setCursor(uint8_t col, uint8_t row);
00076 void groveLCD_noDisplay();
00077 void groveLCD_display() ;
00078 void groveLCD_noCursor();
00079 void groveLCD_cursor() ;
00080 void groveLCD_noBlink();
00081 void groveLCD_blink();
00082 void groveLCD_scrollDisplayLeft(void);
00083 void groveLCD_scrollDisplayRight(void);
00084 void groveLCD_leftToRight(void);
00085 void groveLCD_rightToLeft(void);
00086 void groveLCD_autoscroll(void);
00087 void groveLCD_noAutoscroll(void);
00088 void groveLCD_createChar(uint8_t location, uint8_t charmap[]);
00089 void groveLCD_blinkLED(void);
00090 void groveLCD_noBlinkLED(void);
00091 void groveLCD_command(uint8_t value);
00092 int groveLCD_write(uint8_t value);
00093 void groveLCD_setReg(unsigned char addr, unsigned char dta);
00094 void groveLCD_setRGB(unsigned char r, unsigned char g, unsigned char b);
00095 void groveLCD_setColor(unsigned char color);
00096 void groveLCD_putString(char* s);
00097 void groveLCD_term_printf(const char* fmt, ...);
00098
00099
00100
00101 #endif /* INC_GROVELCD_H_ */
```

5.8 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/main.h File Reference

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

```
#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <rcl/rcl.h>
#include <rcl/error_handling.h>
#include <rclc/rclc.h>
#include <rclc/executor.h>
#include <uxr/client/transport.h>
#include <rmw_microxrcedds_c/config.h>
#include <rmw_microros/rmw_microros.h>
#include <std_msgs/msg/int32.h>
#include <std_msgs/msg/string.h>
#include <std_msgs/msg/header.h>
#include "FreeRTOS.h"
```

```
#include "task.h"
#include "queue.h"
#include "semphr.h"
#include "systemclock.h"
#include "drv_uart.h"
#include "drv_gpio.h"
#include "drv_i2c.h"
#include "cmsis_os.h"
#include "microROS.h"
#include "retarget.h"
```

Macros

- #define B1_Pin GPIO_PIN_13
- #define B1_GPIO_Port GPIOC
- #define USART_TX_Pin GPIO_PIN_2
- #define USART_TX_GPIO_Port GPIOA
- #define USART_RX_Pin GPIO_PIN_3
- #define USART_RX_GPIO_Port GPIOA
- #define LD2_Pin GPIO_PIN_5
- #define LD2_GPIO_Port GPIOA
- #define TMS_Pin GPIO_PIN_13
- #define TMS_GPIO_Port GPIOA
- #define TCK_Pin GPIO_PIN_14
- #define TCK_GPIO_Port GPIOA
- #define SWO_Pin GPIO_PIN_3
- #define SWO_GPIO_Port GPIOB

Functions

- void Error_Handler (void)
- void CHECKMRRET (rcl_ret_t ret, char ∗msg)
- void SubscriberCallbackFunction (const void ∗msgin)
- void microros_task (void ∗argument)
- void task_Motor_Left (void ∗pvParameters)
- void task_Motor_Right (void ∗pvParameters)
- void task_VL53 (void ∗pvParameters)
- void task_Grove_LCD (void ∗pvParameters)
- void task_Supervision (void ∗pvParameters)
- int main (void)

Test function

- void test_uart2 (void ∗pvParameters)
- void test_vl53 (void ∗pvParameters)
- void test_motor (void ∗pvParameters)

5.8.1 Detailed Description

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

Definition in file main.h.

5.8.2 Macro Definition Documentation

5.8.2.1 B1_GPIO_Port

```
#define B1_GPIO_Port GPIOC
```

Definition at line 217 of file main.h.

5.8.2.2 B1_Pin

```
#define B1_Pin GPIO_PIN_13
```

Definition at line 216 of file main.h.

5.8.2.3 LD2_GPIO_Port

```
#define LD2_GPIO_Port GPIOA
```

Definition at line 223 of file main.h.

5.8.2.4 LD2_Pin

```
#define LD2_Pin GPIO_PIN_5
```

Definition at line 222 of file main.h.

5.8.2.5 SWO_GPIO_Port

```
#define SWO_GPIO_Port GPIOB
```

Definition at line 229 of file main.h.

5.8.2.6 SWO_Pin

```
#define SWO_Pin GPIO_PIN_3
```

Definition at line 228 of file main.h.

5.8.2.7 TCK_GPIO_Port

```
#define TCK_GPIO_Port GPIOA
```

Definition at line 227 of file main.h.

5.8.2.8 TCK_Pin

```
#define TCK_Pin GPIO_PIN_14
```

Definition at line 226 of file main.h.

5.8.2.9 TMS_GPIO_Port

```
#define TMS_GPIO_Port GPIOA
```

Definition at line 225 of file main.h.

5.8.2.10 TMS_Pin

```
#define TMS_Pin GPIO_PIN_13
```

Definition at line 224 of file main.h.

5.8.2.11 USART_RX_GPIO_Port

```
#define USART_RX_GPIO_Port GPIOA
```

Definition at line 221 of file main.h.

5.8.2.12 USART_RX_Pin

```
#define USART_RX_Pin GPIO_PIN_3
```

Definition at line 220 of file main.h.

5.8.2.13 USART_TX_GPIO_Port

```
#define USART_TX_GPIO_Port GPIOA
```

Definition at line 219 of file main.h.

5.8.2.14 USART_TX_Pin

```
#define USART_TX_Pin GPIO_PIN_2
```

Definition at line 218 of file main.h.

5.8.3 Function Documentation

5.8.3.1 CHECKMRRET()

```
void CHECKMRRET (
            rcl_ret_t ret,
            char * msg )
```

check if microRos function success else print msg in console

Parameters

ret	return value of microRos function
msg	message to print if fail

Definition at line 153 of file main.c.

5.8.3.2 Error_Handler()

```
void Error_Handler (
             void  )
```

Definition at line 941 of file main.c.

5.8.3.3 main()

```
int main (
             void  )
```

Init all GPIO and drivers, start the task, init semaphore and queue and launch the kernel

- Config EXSTARTUP
  - Launch microRos, supervision, left motor, right motor and lcd task
- Config EXTEST_UART2
  - Launch test_uart2 task
- Config EXCORRECTOR
  - Launch test_motor task
- Config EXTESTCORRECTOR
  - Launch supervision, left motor and right motor task
- Config EXTEST_VL53
  - Launch test_vl53 task
- Config EXTEST_MICROROS
  - Launch microRos task
- Config EXFINAL
  - Launch microRos, supervision, left motor, right motor, vl53 and lcd task

Definition at line 775 of file main.c.

5.8.3.4 microros_task()

```
void microros_task (
            void * argument )
```

- All config
  - Create the node *STM32_node*
  - Set the Domain id of microRos
- Config EXSTARTUP :
  - Create a publisher and send a message on it
- Config EXTEST_MICROROS :
  - Create a publisher, a subscriber and an executor
  - Init the executor and add the subscriber to it
  - Run the executor and send the receive message on the publisher
- Config EXFINAL :
  - Create 3 publishers, 5 subscriber and an executor
  - Init the executor and add the 5 subscribers to it
  - run the executor and if they are no elements waiting to be read by the task decision put the receive information in the queue If decison task send data then publish data to microRos

Parameters

argument	

Definition at line 166 of file main.c.

5.8.3.5 SubscriberCallbackFunction()

```
void SubscriberCallbackFunction (
            const void * msgin )
```

callback call by microros when a message is receive here use as debug and just print the receive msg

Parameters

message	receive

Definition at line 155 of file main.c.

5.8.3.6 task_Grove_LCD()

```
void task_Grove_LCD (
            void * pvParameters )
```

Task use to write information on LCD depending of the data in the LCD queue

- Config EXSTARTUP :
  - Print 'TEST' LCD on screen
- Config EXFINAL :
  - Print different messages depending of the actual mode

Parameters

argument	

5.8.3.7 task_Motor_Left()

```
void task_Motor_Left (
            void * pvParameters )
```

Task use to control the left motor of the robot

Parameters

argument	

Definition at line 365 of file main.c.

5.8.3.8 task_Motor_Right()

```
void task_Motor_Right (
            void * pvParameters )
```

Task use to control the right motor of the robot

Parameters

argument	

Definition at line 391 of file main.c.

5.8.3.9 task_Supervision()

```
void task_Supervision (
            void * pvParameters )
```

Brain of the robot. get information for MicroRos and VL53 task, then send speed to left and right motor, lcd and microRos task

- Config EXSTARTUP :
  - Make the robot drive forward until an obstacle are found

- Config EXTESTCORRECTOR :
  - Make the robot drive forward at speed set by config
- Config EXFINAL :
  - Make robot switch beetween 3 behaviour depending of the mode
  - Obstacle : drive and avoid obstacles
  - Manual : drive in direction set in ihm
  - Camera : follow an object

Parameters

argument	

Definition at line 476 of file main.c.

5.8.3.10 task_VL53()

```
void task_VL53 (
            void * pvParameters )
```

task that get the value of the VL53 sensor and put it on the VL53 queue

Parameters

argument	

5.8.3.11 test_motor()

```
void test_motor (
            void * pvParameters )
```

Use to set the duty cycle and register the motor speed at each Te

Definition at line 895 of file main.c.

5.8.3.12 test_uart2()

```
void test_uart2 (
            void * pvParameters )
```

Use to test printf and scanf function

Definition at line 872 of file main.c.

5.8.3.13 test_vl53()

```
void test_vl53 (
            void * pvParameters )
```

Use to test the VL53 sensor

Definition at line 884 of file main.c.

5.9 main.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00045 /* USER CODE END Header */
00046
00047 /* Define to prevent recursive inclusion -------------------------------------*/
00048 #ifndef __MAIN_H
00049 #define __MAIN_H
00050
00051 #ifdef __cplusplus
00052 extern "C" {
00053 #endif
00054
00055 /* Includes ------------------------------------------------------------------*/
00056 #include "stm32f4xx_hal.h"
00057
00058 #include <stdio.h>
00059 #include <rcl/rcl.h>
00060 #include <rcl/error_handling.h>
00061 #include <rclc/rclc.h>
00062 #include <rclc/executor.h>
00063 #include <uxr/client/transport.h>
00064 #include <rmw_microxrcedds_c/config.h>
00065 #include <rmw_microros/rmw_microros.h>
00066
00067 #include <std_msgs/msg/int32.h>
00068 #include <std_msgs/msg/string.h>
00069 #include <std_msgs/msg/header.h>
00070
00071 #include "FreeRTOS.h"
00072 #include "task.h"
00073 #include "queue.h"
00074 #include "semphr.h"
00075
00076 #include "systemclock.h"
00077 #include "drv_uart.h"
00078 #include "drv_gpio.h"
00079 #include "drv_i2c.h"
00080 #include "cmsis_os.h"
00081
00082 #include "microROS.h" //Custom microRos utils
00083 #include "retarget.h" //To redirect printf and scanf on UART2
00084
00085 /* Private includes ----------------------------------------------------------*/
00086 /* USER CODE BEGIN Includes */
00087
00088 /* USER CODE END Includes */
00089
00090 /* Exported types ------------------------------------------------------------*/
00091 /* USER CODE BEGIN ET */
00092
00093 /* USER CODE END ET */
00094
00095 /* Exported constants --------------------------------------------------------*/
00096 /* USER CODE BEGIN EC */
00097
00098 /* USER CODE END EC */
00099
00100 /* Exported macro ------------------------------------------------------------*/
00101 /* USER CODE BEGIN EM */
00102
00103 /* USER CODE END EM */
00104
00105 /* Exported functions prototypes ---------------------------------------------*/
00106 void Error_Handler(void);
00107
00108 /* USER CODE BEGIN EFP */
```

```
00113 void CHECKMRRET(rcl_ret_t ret, char* msg);
00114
00119 void SubscriberCallbackFunction(const void *msgin);
00120
00139 void microros_task(void *argument);
00140
00145 void task_Motor_Left(void *pvParameters);
00146
00151 void task_Motor_Right(void *pvParameters);
00152
00157 void task_VL53(void *pvParameters);
00158
00167 void task_Grove_LCD(void *pvParameters);
00168
00183 void task_Supervision(void *pvParameters);
00184
00187 void test_uart2(void *pvParameters);
00189 void test_vl53(void *pvParameters);
00191 void test_motor(void *pvParameters);
00211 int main(void);
00212
00213 /* USER CODE END EFP */
00214
00215 /* Private defines -----------------------------------------------------------*/
00216 #define B1_Pin GPIO_PIN_13
00217 #define B1_GPIO_Port GPIOC
00218 #define USART_TX_Pin GPIO_PIN_2
00219 #define USART_TX_GPIO_Port GPIOA
00220 #define USART_RX_Pin GPIO_PIN_3
00221 #define USART_RX_GPIO_Port GPIOA
00222 #define LD2_Pin GPIO_PIN_5
00223 #define LD2_GPIO_Port GPIOA
00224 #define TMS_Pin GPIO_PIN_13
00225 #define TMS_GPIO_Port GPIOA
00226 #define TCK_Pin GPIO_PIN_14
00227 #define TCK_GPIO_Port GPIOA
00228 #define SWO_Pin GPIO_PIN_3
00229 #define SWO_GPIO_Port GPIOB
00230 /* USER CODE BEGIN Private defines */
00231
00232 /* USER CODE END Private defines */
00233
00234 #ifdef __cplusplus
00235 }
00236 #endif
00237
00238 #endif /* __MAIN_H */
```

5.10 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/microROS.h File Reference

: Contain microROS topic and default creator for subscriber and publisher

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

Macros

- #define ARRAY_LEN 100
- #define CAPTEUR_DIR_TOPIC "capteur/dir"
- #define ETAT_MODE_TOPIC "etat/mode"
- #define ETAT_SPEED_TOPIC "etat/speed"
- #define CAMERA_X_TOPIC "camera/X"
- #define CAMERA_Y_TOPIC "camera/Y"
- #define TELECOMMANDE_DIR_TOPIC "direction"
- #define CONFIG_MODE_TOPIC "mode"
- #define CONFIG_SPEED_TOPIC "speed"

Functions

- void createPublisher (rcl_publisher_t *publisher, const rcl_node_t *node, const rosidl_message_type_support_t *type_support, const char *topic_name, std_msgs__msg__Int32 *msg)
- void createSubscriber (rcl_subscription_t *subscription, rcl_node_t *node, const rosidl_message_type_support_t *type_support, const char *topic_name, std_msgs__msg__Int32 *msg)

5.10.1 Detailed Description

: Contain microROS topic and default creator for subscriber and publisher

Definition in file microROS.h.

5.10.2 Macro Definition Documentation

5.10.2.1 ARRAY_LEN

```
#define ARRAY_LEN 100
```

Length of string messages

Definition at line 10 of file microROS.h.

5.10.2.2 CAMERA_X_TOPIC

```
#define CAMERA_X_TOPIC "camera/X"
```

Topic name of x camera subscriber

Definition at line 14 of file microROS.h.

5.10.2.3 CAMERA_Y_TOPIC

```
#define CAMERA_Y_TOPIC "camera/Y"
```

Topic name of y camera subscriber

Definition at line 15 of file microROS.h.

5.10.2.4 CAPTEUR_DIR_TOPIC

```
#define CAPTEUR_DIR_TOPIC "capteur/dir"
```

Topic name of direction publisher

Definition at line 11 of file microROS.h.

5.10.2.5 CONFIG_MODE_TOPIC

```
#define CONFIG_MODE_TOPIC "mode"
```

Definition at line 17 of file microROS.h.

5.10.2.6 CONFIG_SPEED_TOPIC

```
#define CONFIG_SPEED_TOPIC "speed"
```

Definition at line 18 of file microROS.h.

5.10.2.7 ETAT_MODE_TOPIC

```
#define ETAT_MODE_TOPIC "etat/mode"
```

Topic name of mode publisher

Definition at line 12 of file microROS.h.

5.10.2.8 ETAT_SPEED_TOPIC

```
#define ETAT_SPEED_TOPIC "etat/speed"
```

Topic name of speed publisher

Definition at line 13 of file microROS.h.

5.10.2.9 TELECOMMANDE_DIR_TOPIC

```
#define TELECOMMANDE_DIR_TOPIC "direction"
```

Definition at line 16 of file microROS.h.

5.10.3 Function Documentation

5.10.3.1 createPublisher()

```
void createPublisher (
            rcl_publisher_t * publisher,
            const rcl_node_t * node,
            const rosidl_message_type_support_t * type_support,
            const char * topic_name,
            std_msgs__msg__Int32 * msg )
```

Create a publisher with default options

Parameters

publisher	microRos structure that represent a publisher
node	microRos structure that represent a node
type_support	microRos structure that represent the type of message
topic_name	The name of the topic
msg	microRos structure that represent the message

Definition at line 5 of file microROS.c.

5.10.3.2 createSubscriber()

```
void createSubscriber (
            rcl_subscription_t * subscription,
            rcl_node_t * node,
            const rosidl_message_type_support_t * type_support,
            const char * topic_name,
            std_msgs__msg__Int32 * msg )
```

Create a subscriber with default options

Parameters

subscription	microRos structure that represent a subscriber
node	microRos structure that represent a node
type_support	microRos structure that represent the type of message
topic_name	The name of the topic
msg	microRos structure that represent the message

Definition at line 24 of file microROS.c.

5.11 microROS.h

Go to the documentation of this file.

```
00001
00006 #ifndef DEF_MICROROS
00007 #define DEF_MICROROS
00008
00009     #include "main.h"
00010     #define ARRAY_LEN 100
00011     #define CAPTEUR_DIR_TOPIC "capteur/dir"
00012     #define ETAT_MODE_TOPIC "etat/mode"
00013     #define ETAT_SPEED_TOPIC "etat/speed"
00014     #define CAMERA_X_TOPIC "camera/X"
00015     #define CAMERA_Y_TOPIC "camera/Y"
00016     #define TELECOMMANDE_DIR_TOPIC "direction"  //"telecommande/dir"
00017     #define CONFIG_MODE_TOPIC "mode"  //"config/mode"
00018     #define CONFIG_SPEED_TOPIC "speed"  //"config/speed"
00019
00028     void createPublisher(rcl_publisher_t* publisher,
00029               const rcl_node_t* node,
00030               const rosidl_message_type_support_t* type_support,
00031               const char* topic_name,
00032               std_msgs__msg__Int32* msg);
00033
00042     void createSubscriber(rcl_subscription_t* subscription,
00043               rcl_node_t* node,
```

```
00044               const rosidl_message_type_support_t* type_support,
00045               const char* topic_name,
00046               std_msgs__msg__Int32* msg);
00047
00048 #endif //DEF_MICROROS
```

5.12 motorCommand.h

```
00001 /*
00002  * MotorCommand.h
00003  */
00004
00005 #ifndef INC_MOTORCOMMAND_H_
00006 #define INC_MOTORCOMMAND_H_
00007
00008 #include "main.h"
00009
00010
00011 void motorCommand_Init(void);
00012 void motorLeft_SetDuty(int);
00013 void motorRight_SetDuty(int);
00014
00015
00016
00017 #endif /* INC_MOTORCOMMAND_H_ */
```

5.13 quadEncoder.h

```
00001 /*
00002  * QuadEncoder.h
00003  */
00004
00005 #ifndef INC_QUADENCODER_H_
00006 #define INC_QUADENCODER_H_
00007
00008 #include "main.h"
00009
00010 void quadEncoder_Init(void);
00011 int16_t quadEncoder_GetPos16L(void);
00012 int16_t quadEncoder_GetPos16R(void);
00013 int32_t quadEncoder_GetPos32L(void);
00014 int32_t quadEncoder_GetPos32R(void);
00015 int16_t quadEncoder_GetSpeedL(void);
00016 int16_t quadEncoder_GetSpeedR(void);
00017 void quadEncoder_CallbackIndexL(void);
00018 void quadEncoder_CallbackIndexR(void);
00019 void quadEncoder_PosCalcL(int*);
00020 void quadEncoder_PosCalcR(int*);
00021
00022 #endif /* INC_QUADENCODER_H_ */
```

5.14 retarget.h

```
00001 /*
00002  * retarget.h
00003  *
00004  *  Created on: Oct 10, 2023
00005  *      Author: rospc
00006  */
00007
00008 #ifndef INC_RETARGET_H_
00009 #define INC_RETARGET_H_
00010
00011 // All credit to Carmine Noviello for this code
00012 //
      https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/include/retarget/retarget.h
00013
00014 #ifndef __RETARGET_H__
00015 #define __RETARGET_H__
00016
00017 #include "stm32f4xx_hal.h"
00018 #include <sys/stat.h>
00019
00020 void RetargetInit(UART_HandleTypeDef *huart);
00021 int _isatty(int fd);
```

```
00022 int _write(int fd, char* ptr, int len);
00023 int _close(int fd);
00024 int _lseek(int fd, int ptr, int dir);
00025 int _read(int fd, char* ptr, int len);
00026 int _fstat(int fd, struct stat* st);
00027 int _getpid(void);
00028 int _kill(int pid, int sig);
00029
00030 #endif //#ifndef _RETARGET_H__
00031
00032 #endif /* INC_RETARGET_H_ */
```

5.15 stm32f4xx_hal_conf.h

```
00001 /* USER CODE BEGIN Header */
00021 /* USER CODE END Header */
00022
00023 /* Define to prevent recursive inclusion -------------------------------------*/
00024 #ifndef __STM32F4xx_HAL_CONF_H
00025 #define __STM32F4xx_HAL_CONF_H
00026
00027 #ifdef __cplusplus
00028  extern "C" {
00029 #endif
00030
00031 /* Exported types ------------------------------------------------------------*/
00032 /* Exported constants --------------------------------------------------------*/
00033
00034 /* ########################## Module Selection ############################## */
00038 #define HAL_MODULE_ENABLED
00039
00040  #define HAL_ADC_MODULE_ENABLED
00041 /* #define HAL_CRYP_MODULE_ENABLED   */
00042 /* #define HAL_CAN_MODULE_ENABLED   */
00043 /* #define HAL_CRC_MODULE_ENABLED   */
00044 /* #define HAL_CAN_LEGACY_MODULE_ENABLED   */
00045 /* #define HAL_CRYP_MODULE_ENABLED   */
00046 /* #define HAL_DAC_MODULE_ENABLED   */
00047 /* #define HAL_DCMI_MODULE_ENABLED   */
00048 /* #define HAL_DMA2D_MODULE_ENABLED   */
00049 /* #define HAL_ETH_MODULE_ENABLED   */
00050 /* #define HAL_NAND_MODULE_ENABLED   */
00051 /* #define HAL_NOR_MODULE_ENABLED   */
00052 /* #define HAL_PCCARD_MODULE_ENABLED   */
00053 /* #define HAL_SRAM_MODULE_ENABLED   */
00054 /* #define HAL_SDRAM_MODULE_ENABLED   */
00055 /* #define HAL_HASH_MODULE_ENABLED   */
00056 #define HAL_I2C_MODULE_ENABLED
00057 /* #define HAL_I2S_MODULE_ENABLED   */
00058 /* #define HAL_IWDG_MODULE_ENABLED   */
00059 /* #define HAL_LTDC_MODULE_ENABLED   */
00060 /* #define HAL_RNG_MODULE_ENABLED   */
00061 /* #define HAL_RTC_MODULE_ENABLED   */
00062 /* #define HAL_SAI_MODULE_ENABLED   */
00063 /* #define HAL_SD_MODULE_ENABLED   */
00064 /* #define HAL_MMC_MODULE_ENABLED   */
00065 /* #define HAL_SPI_MODULE_ENABLED   */
00066 #define HAL_TIM_MODULE_ENABLED
00067 #define HAL_UART_MODULE_ENABLED
00068 /* #define HAL_USART_MODULE_ENABLED   */
00069 /* #define HAL_IRDA_MODULE_ENABLED   */
00070 /* #define HAL_SMARTCARD_MODULE_ENABLED   */
00071 /* #define HAL_SMBUS_MODULE_ENABLED   */
00072 /* #define HAL_WWDG_MODULE_ENABLED   */
00073 /* #define HAL_PCD_MODULE_ENABLED   */
00074 /* #define HAL_HCD_MODULE_ENABLED   */
00075 /* #define HAL_DSI_MODULE_ENABLED   */
00076 /* #define HAL_QSPI_MODULE_ENABLED   */
00077 /* #define HAL_QSPI_MODULE_ENABLED   */
00078 /* #define HAL_CEC_MODULE_ENABLED   */
00079 /* #define HAL_FMPI2C_MODULE_ENABLED   */
00080 /* #define HAL_FMPSMBUS_MODULE_ENABLED   */
00081 /* #define HAL_SPDIFRX_MODULE_ENABLED   */
00082 /* #define HAL_DFSDM_MODULE_ENABLED   */
00083 /* #define HAL_LPTIM_MODULE_ENABLED   */
00084 #define HAL_GPIO_MODULE_ENABLED
00085 #define HAL_EXTI_MODULE_ENABLED
00086 #define HAL_DMA_MODULE_ENABLED
00087 #define HAL_RCC_MODULE_ENABLED
00088 #define HAL_FLASH_MODULE_ENABLED
00089 #define HAL_PWR_MODULE_ENABLED
00090 #define HAL_CORTEX_MODULE_ENABLED
00091
```

```
00092 /* ########################## HSE/HSI Values adaptation ##################### */
00098 #if !defined  (HSE_VALUE)
00099   #define HSE_VALUE    8000000U
00100 #endif /* HSE_VALUE */
00101
00102 #if !defined  (HSE_STARTUP_TIMEOUT)
00103   #define HSE_STARTUP_TIMEOUT    100U
00104 #endif /* HSE_STARTUP_TIMEOUT */
00105
00111 #if !defined  (HSI_VALUE)
00112   #define HSI_VALUE    ((uint32_t)16000000U)
00113 #endif /* HSI_VALUE */
00114
00118 #if !defined  (LSI_VALUE)
00119  #define LSI_VALUE  32000U
00120 #endif /* LSI_VALUE */
00126 #if !defined  (LSE_VALUE)
00127  #define LSE_VALUE  32768U
00128 #endif /* LSE_VALUE */
00129
00130 #if !defined  (LSE_STARTUP_TIMEOUT)
00131   #define LSE_STARTUP_TIMEOUT    5000U
00132 #endif /* LSE_STARTUP_TIMEOUT */
00133
00139 #if !defined  (EXTERNAL_CLOCK_VALUE)
00140   #define EXTERNAL_CLOCK_VALUE    12288000U
00141 #endif /* EXTERNAL_CLOCK_VALUE */
00142
00143 /* Tip: To avoid modifying this file each time you need to use different HSE,
00144    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00145
00146 /* ########################### System Configuration ######################### */
00150 #define  VDD_VALUE            3300U
00151 #define  TICK_INT_PRIORITY            15U
00152 #define  USE_RTOS                     0U
00153 #define  PREFETCH_ENABLE              1U
00154 #define  INSTRUCTION_CACHE_ENABLE     1U
00155 #define  DATA_CACHE_ENABLE            1U
00156
00157 #define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
00158 #define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
00159 #define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
00160 #define  USE_HAL_CRYP_REGISTER_CALLBACKS        0U /* CRYP register callback disabled      */
00161 #define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
00162 #define  USE_HAL_DCMI_REGISTER_CALLBACKS        0U /* DCMI register callback disabled      */
00163 #define  USE_HAL_DFSDM_REGISTER_CALLBACKS       0U /* DFSDM register callback disabled     */
00164 #define  USE_HAL_DMA2D_REGISTER_CALLBACKS       0U /* DMA2D register callback disabled     */
00165 #define  USE_HAL_DSI_REGISTER_CALLBACKS         0U /* DSI register callback disabled       */
00166 #define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback disabled       */
00167 #define  USE_HAL_HASH_REGISTER_CALLBACKS        0U /* HASH register callback disabled      */
00168 #define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback disabled       */
00169 #define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
00170 #define  USE_HAL_FMPI2C_REGISTER_CALLBACKS      0U /* FMPI2C register callback disabled    */
00171 #define  USE_HAL_FMPSMBUS_REGISTER_CALLBACKS    0U /* FMPSMBUS register callback disabled  */
00172 #define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
00173 #define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
00174 #define  USE_HAL_LPTIM_REGISTER_CALLBACKS       0U /* LPTIM register callback disabled     */
00175 #define  USE_HAL_LTDC_REGISTER_CALLBACKS        0U /* LTDC register callback disabled      */
00176 #define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback disabled       */
00177 #define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback disabled      */
00178 #define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback disabled       */
00179 #define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback disabled    */
00180 #define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
00181 #define  USE_HAL_QSPI_REGISTER_CALLBACKS        0U /* QSPI register callback disabled      */
00182 #define  USE_HAL_RNG_REGISTER_CALLBACKS         0U /* RNG register callback disabled       */
00183 #define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
00184 #define  USE_HAL_SAI_REGISTER_CALLBACKS         0U /* SAI register callback disabled       */
00185 #define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback disabled        */
00186 #define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
00187 #define  USE_HAL_SDRAM_REGISTER_CALLBACKS       0U /* SDRAM register callback disabled     */
00188 #define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback disabled      */
00189 #define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /* SPDIFRX register callback disabled   */
00190 #define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
00191 #define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
00192 #define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
00193 #define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
00194 #define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
00195 #define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */
00196
00197 /* ########################## Assert Selection ############################## */
00202 /* #define USE_FULL_ASSERT    1U */
00203
00204 /* ################### Ethernet peripheral configuration ##################### */
00205
00206 /* Section 1 : Ethernet peripheral configuration */
00207
00208 /* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
```

```
00209 #define MAC_ADDR0   2U
00210 #define MAC_ADDR1   0U
00211 #define MAC_ADDR2   0U
00212 #define MAC_ADDR3   0U
00213 #define MAC_ADDR4   0U
00214 #define MAC_ADDR5   0U
00215
00216 /* Definition of the Ethernet driver buffers size and count */
00217 #define ETH_RX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for receive               */
00218 #define ETH_TX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for transmit              */
00219 #define ETH_RXBUFNB                    4U       /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
00220 #define ETH_TXBUFNB                    4U       /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */
00221
00222 /* Section 2: PHY configuration section */
00223
00224 /* DP83848_PHY_ADDRESS Address*/
00225 #define DP83848_PHY_ADDRESS           0x01U
00226 /* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
00227 #define PHY_RESET_DELAY                0x000000FFU
00228 /* PHY Configuration delay */
00229 #define PHY_CONFIG_DELAY               0x00000FFFU
00230
00231 #define PHY_READ_TO                    0x0000FFFFU
00232 #define PHY_WRITE_TO                   0x0000FFFFU
00233
00234 /* Section 3: Common PHY Registers */
00235
00236 #define PHY_BCR                        ((uint16_t)0x0000U)
00237 #define PHY_BSR                        ((uint16_t)0x0001U)
00239 #define PHY_RESET                      ((uint16_t)0x8000U)
00240 #define PHY_LOOPBACK                   ((uint16_t)0x4000U)
00241 #define PHY_FULLDUPLEX_100M            ((uint16_t)0x2100U)
00242 #define PHY_HALFDUPLEX_100M            ((uint16_t)0x2000U)
00243 #define PHY_FULLDUPLEX_10M             ((uint16_t)0x0100U)
00244 #define PHY_HALFDUPLEX_10M             ((uint16_t)0x0000U)
00245 #define PHY_AUTONEGOTIATION            ((uint16_t)0x1000U)
00246 #define PHY_RESTART_AUTONEGOTIATION    ((uint16_t)0x0200U)
00247 #define PHY_POWERDOWN                  ((uint16_t)0x0800U)
00248 #define PHY_ISOLATE                    ((uint16_t)0x0400U)
00250 #define PHY_AUTONEGO_COMPLETE          ((uint16_t)0x0020U)
00251 #define PHY_LINKED_STATUS              ((uint16_t)0x0004U)
00252 #define PHY_JABBER_DETECTION           ((uint16_t)0x0002U)
00254 /* Section 4: Extended PHY Registers */
00255 #define PHY_SR                         ((uint16_t)0x10U)
00257 #define PHY_SPEED_STATUS               ((uint16_t)0x0002U)
00258 #define PHY_DUPLEX_STATUS              ((uint16_t)0x0004U)
00260 /* ################## SPI peripheral configuration ########################### */
00261
00262 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00263 * Activated: CRC code is present inside driver
00264 * Deactivated: CRC code cleaned from driver
00265 */
00266
00267 #define USE_SPI_CRC                    0U
00268
00269 /* Includes ------------------------------------------------------------------*/
00274 #ifdef HAL_RCC_MODULE_ENABLED
00275   #include "stm32f4xx_hal_rcc.h"
00276 #endif /* HAL_RCC_MODULE_ENABLED */
00277
00278 #ifdef HAL_GPIO_MODULE_ENABLED
00279   #include "stm32f4xx_hal_gpio.h"
00280 #endif /* HAL_GPIO_MODULE_ENABLED */
00281
00282 #ifdef HAL_EXTI_MODULE_ENABLED
00283   #include "stm32f4xx_hal_exti.h"
00284 #endif /* HAL_EXTI_MODULE_ENABLED */
00285
00286 #ifdef HAL_DMA_MODULE_ENABLED
00287   #include "stm32f4xx_hal_dma.h"
00288 #endif /* HAL_DMA_MODULE_ENABLED */
00289
00290 #ifdef HAL_CORTEX_MODULE_ENABLED
00291   #include "stm32f4xx_hal_cortex.h"
00292 #endif /* HAL_CORTEX_MODULE_ENABLED */
00293
00294 #ifdef HAL_ADC_MODULE_ENABLED
00295   #include "stm32f4xx_hal_adc.h"
00296 #endif /* HAL_ADC_MODULE_ENABLED */
00297
00298 #ifdef HAL_CAN_MODULE_ENABLED
00299   #include "stm32f4xx_hal_can.h"
00300 #endif /* HAL_CAN_MODULE_ENABLED */
00301
00302 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00303   #include "stm32f4xx_hal_can_legacy.h"
00304 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
```

```
00305
00306 #ifdef HAL_CRC_MODULE_ENABLED
00307   #include "stm32f4xx_hal_crc.h"
00308 #endif /* HAL_CRC_MODULE_ENABLED */
00309
00310 #ifdef HAL_CRYP_MODULE_ENABLED
00311   #include "stm32f4xx_hal_cryp.h"
00312 #endif /* HAL_CRYP_MODULE_ENABLED */
00313
00314 #ifdef HAL_DMA2D_MODULE_ENABLED
00315   #include "stm32f4xx_hal_dma2d.h"
00316 #endif /* HAL_DMA2D_MODULE_ENABLED */
00317
00318 #ifdef HAL_DAC_MODULE_ENABLED
00319   #include "stm32f4xx_hal_dac.h"
00320 #endif /* HAL_DAC_MODULE_ENABLED */
00321
00322 #ifdef HAL_DCMI_MODULE_ENABLED
00323   #include "stm32f4xx_hal_dcmi.h"
00324 #endif /* HAL_DCMI_MODULE_ENABLED */
00325
00326 #ifdef HAL_ETH_MODULE_ENABLED
00327   #include "stm32f4xx_hal_eth.h"
00328 #endif /* HAL_ETH_MODULE_ENABLED */
00329
00330 #ifdef HAL_FLASH_MODULE_ENABLED
00331   #include "stm32f4xx_hal_flash.h"
00332 #endif /* HAL_FLASH_MODULE_ENABLED */
00333
00334 #ifdef HAL_SRAM_MODULE_ENABLED
00335   #include "stm32f4xx_hal_sram.h"
00336 #endif /* HAL_SRAM_MODULE_ENABLED */
00337
00338 #ifdef HAL_NOR_MODULE_ENABLED
00339   #include "stm32f4xx_hal_nor.h"
00340 #endif /* HAL_NOR_MODULE_ENABLED */
00341
00342 #ifdef HAL_NAND_MODULE_ENABLED
00343   #include "stm32f4xx_hal_nand.h"
00344 #endif /* HAL_NAND_MODULE_ENABLED */
00345
00346 #ifdef HAL_PCCARD_MODULE_ENABLED
00347   #include "stm32f4xx_hal_pccard.h"
00348 #endif /* HAL_PCCARD_MODULE_ENABLED */
00349
00350 #ifdef HAL_SDRAM_MODULE_ENABLED
00351   #include "stm32f4xx_hal_sdram.h"
00352 #endif /* HAL_SDRAM_MODULE_ENABLED */
00353
00354 #ifdef HAL_HASH_MODULE_ENABLED
00355  #include "stm32f4xx_hal_hash.h"
00356 #endif /* HAL_HASH_MODULE_ENABLED */
00357
00358 #ifdef HAL_I2C_MODULE_ENABLED
00359  #include "stm32f4xx_hal_i2c.h"
00360 #endif /* HAL_I2C_MODULE_ENABLED */
00361
00362 #ifdef HAL_SMBUS_MODULE_ENABLED
00363  #include "stm32f4xx_hal_smbus.h"
00364 #endif /* HAL_SMBUS_MODULE_ENABLED */
00365
00366 #ifdef HAL_I2S_MODULE_ENABLED
00367  #include "stm32f4xx_hal_i2s.h"
00368 #endif /* HAL_I2S_MODULE_ENABLED */
00369
00370 #ifdef HAL_IWDG_MODULE_ENABLED
00371  #include "stm32f4xx_hal_iwdg.h"
00372 #endif /* HAL_IWDG_MODULE_ENABLED */
00373
00374 #ifdef HAL_LTDC_MODULE_ENABLED
00375  #include "stm32f4xx_hal_ltdc.h"
00376 #endif /* HAL_LTDC_MODULE_ENABLED */
00377
00378 #ifdef HAL_PWR_MODULE_ENABLED
00379  #include "stm32f4xx_hal_pwr.h"
00380 #endif /* HAL_PWR_MODULE_ENABLED */
00381
00382 #ifdef HAL_RNG_MODULE_ENABLED
00383  #include "stm32f4xx_hal_rng.h"
00384 #endif /* HAL_RNG_MODULE_ENABLED */
00385
00386 #ifdef HAL_RTC_MODULE_ENABLED
00387  #include "stm32f4xx_hal_rtc.h"
00388 #endif /* HAL_RTC_MODULE_ENABLED */
00389
00390 #ifdef HAL_SAI_MODULE_ENABLED
00391  #include "stm32f4xx_hal_sai.h"
```

```
00392 #endif /* HAL_SAI_MODULE_ENABLED */
00393
00394 #ifdef HAL_SD_MODULE_ENABLED
00395  #include "stm32f4xx_hal_sd.h"
00396 #endif /* HAL_SD_MODULE_ENABLED */
00397
00398 #ifdef HAL_SPI_MODULE_ENABLED
00399  #include "stm32f4xx_hal_spi.h"
00400 #endif /* HAL_SPI_MODULE_ENABLED */
00401
00402 #ifdef HAL_TIM_MODULE_ENABLED
00403  #include "stm32f4xx_hal_tim.h"
00404 #endif /* HAL_TIM_MODULE_ENABLED */
00405
00406 #ifdef HAL_UART_MODULE_ENABLED
00407  #include "stm32f4xx_hal_uart.h"
00408 #endif /* HAL_UART_MODULE_ENABLED */
00409
00410 #ifdef HAL_USART_MODULE_ENABLED
00411  #include "stm32f4xx_hal_usart.h"
00412 #endif /* HAL_USART_MODULE_ENABLED */
00413
00414 #ifdef HAL_IRDA_MODULE_ENABLED
00415  #include "stm32f4xx_hal_irda.h"
00416 #endif /* HAL_IRDA_MODULE_ENABLED */
00417
00418 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00419  #include "stm32f4xx_hal_smartcard.h"
00420 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00421
00422 #ifdef HAL_WWDG_MODULE_ENABLED
00423  #include "stm32f4xx_hal_wwdg.h"
00424 #endif /* HAL_WWDG_MODULE_ENABLED */
00425
00426 #ifdef HAL_PCD_MODULE_ENABLED
00427  #include "stm32f4xx_hal_pcd.h"
00428 #endif /* HAL_PCD_MODULE_ENABLED */
00429
00430 #ifdef HAL_HCD_MODULE_ENABLED
00431  #include "stm32f4xx_hal_hcd.h"
00432 #endif /* HAL_HCD_MODULE_ENABLED */
00433
00434 #ifdef HAL_DSI_MODULE_ENABLED
00435  #include "stm32f4xx_hal_dsi.h"
00436 #endif /* HAL_DSI_MODULE_ENABLED */
00437
00438 #ifdef HAL_QSPI_MODULE_ENABLED
00439  #include "stm32f4xx_hal_qspi.h"
00440 #endif /* HAL_QSPI_MODULE_ENABLED */
00441
00442 #ifdef HAL_CEC_MODULE_ENABLED
00443  #include "stm32f4xx_hal_cec.h"
00444 #endif /* HAL_CEC_MODULE_ENABLED */
00445
00446 #ifdef HAL_FMPI2C_MODULE_ENABLED
00447  #include "stm32f4xx_hal_fmpi2c.h"
00448 #endif /* HAL_FMPI2C_MODULE_ENABLED */
00449
00450 #ifdef HAL_FMPSMBUS_MODULE_ENABLED
00451  #include "stm32f4xx_hal_fmpsmbus.h"
00452 #endif /* HAL_FMPSMBUS_MODULE_ENABLED */
00453
00454 #ifdef HAL_SPDIFRX_MODULE_ENABLED
00455  #include "stm32f4xx_hal_spdifrx.h"
00456 #endif /* HAL_SPDIFRX_MODULE_ENABLED */
00457
00458 #ifdef HAL_DFSDM_MODULE_ENABLED
00459  #include "stm32f4xx_hal_dfsdm.h"
00460 #endif /* HAL_DFSDM_MODULE_ENABLED */
00461
00462 #ifdef HAL_LPTIM_MODULE_ENABLED
00463  #include "stm32f4xx_hal_lptim.h"
00464 #endif /* HAL_LPTIM_MODULE_ENABLED */
00465
00466 #ifdef HAL_MMC_MODULE_ENABLED
00467  #include "stm32f4xx_hal_mmc.h"
00468 #endif /* HAL_MMC_MODULE_ENABLED */
00469
00470 /* Exported macro ------------------------------------------------------------*/
00471 #ifdef  USE_FULL_ASSERT
00480   #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00481 /* Exported functions ------------------------------------------------------- */
00482   void assert_failed(uint8_t* file, uint32_t line);
00483 #else
00484   #define assert_param(expr) ((void)0U)
00485 #endif /* USE_FULL_ASSERT */
00486
```

```
00487 #ifdef __cplusplus
00488 }
00489 #endif
00490
00491 #endif /* __STM32F4xx_HAL_CONF_H */
```

5.16 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/stm32f4xx↩ _it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void NMI_Handler (void)
- void HardFault_Handler (void)
- void MemManage_Handler (void)
- void BusFault_Handler (void)
- void UsageFault_Handler (void)

  This function handles Undefined instruction or illegal state.
- void DebugMon_Handler (void)
- void DMA1_Stream5_IRQHandler (void)
- void DMA1_Stream6_IRQHandler (void)
- void **TIM1_UP_TIM10_IRQHandler** (void)
- void USART1_IRQHandler (void)
- void USART2_IRQHandler (void)
- void DMA2_Stream2_IRQHandler (void)
- void DMA2_Stream7_IRQHandler (void)

5.16.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

Definition in file stm32f4xx_it.h.

5.16.2 Function Documentation

5.16.2.1 BusFault_Handler()

```
void BusFault_Handler (
            void  )
```

Definition at line 38 of file stm32f4xx_it.c.

5.16.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

Definition at line 55 of file stm32f4xx_it.c.

5.16.2.3 DMA1_Stream5_IRQHandler()

```
void DMA1_Stream5_IRQHandler (
             void  )
```

Definition at line 66 of file stm32f4xx_it.c.

5.16.2.4 DMA1_Stream6_IRQHandler()

```
void DMA1_Stream6_IRQHandler (
             void  )
```

Definition at line 72 of file stm32f4xx_it.c.

5.16.2.5 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
             void  )
```

Definition at line 97 of file stm32f4xx_it.c.

5.16.2.6 DMA2_Stream7_IRQHandler()

```
void DMA2_Stream7_IRQHandler (
             void  )
```

Definition at line 102 of file stm32f4xx_it.c.

5.16.2.7 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

Definition at line 22 of file stm32f4xx_it.c.

5.16.2.8 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

Definition at line 30 of file stm32f4xx_it.c.

5.16.2.9 NMI_Handler()

```
void NMI_Handler (
             void  )
```

Definition at line 15 of file stm32f4xx_it.c.

5.16.2.10 UsageFault_Handler()

```
void UsageFault_Handler (
             void  )
```

This function handles Undefined instruction or illegal state.

Definition at line 48 of file stm32f4xx_it.c.

5.16.2.11 USART1_IRQHandler()

```
void USART1_IRQHandler (
             void  )
```

Definition at line 87 of file stm32f4xx_it.c.

5.16.2.12 USART2_IRQHandler()

```
void USART2_IRQHandler (
             void  )
```

Definition at line 92 of file stm32f4xx_it.c.

5.17 stm32f4xx_it.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32F4xx_IT_H
00022 #define __STM32F4xx_IT_H
00023
00024 #ifdef __cplusplus
00025  extern "C" {
00026 #endif
00027
00028 /* Private includes ----------------------------------------------------------*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types ------------------------------------------------------------*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants --------------------------------------------------------*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
```

```
00043 /* Exported macro ------------------------------------------------------------*/
00044 /* USER CODE BEGIN EM */
00045
00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes ---------------------------------------------*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void MemManage_Handler(void);
00052 void BusFault_Handler(void);
00053 void UsageFault_Handler(void);
00054 void DebugMon_Handler(void);
00055 void DMA1_Stream5_IRQHandler(void);
00056 void DMA1_Stream6_IRQHandler(void);
00057 void TIM1_UP_TIM10_IRQHandler(void);
00058 void USART1_IRQHandler(void);
00059 void USART2_IRQHandler(void);
00060 void DMA2_Stream2_IRQHandler(void);
00061 void DMA2_Stream7_IRQHandler(void);
00062 /* USER CODE BEGIN EFP */
00063
00064 /* USER CODE END EFP */
00065
00066 #ifdef __cplusplus
00067 }
00068 #endif
00069
00070 #endif /* __STM32F4xx_IT_H */
```

5.18 systemclock.h

```
00001 /*
00002  * systemclock.h
00003  *
00004  *  Created on: Mar 13, 2023
00005  *      Author: kerhoas
00006  */
00007
00008 #ifndef INC_SYSTEMCLOCK_H_
00009 #define INC_SYSTEMCLOCK_H_
00010
00011 void SystemClock_Config(void);
00012
00013 #endif /* INC_SYSTEMCLOCK_H_ */
```

5.19 util.h

```
00001 /*
00002  * utils.h
00003  */
00004
00005 #ifndef INC_UTIL_H_
00006 #define INC_UTIL_H_
00007
00008 #include "main.h"
00009
00010 void num2str(char *s, unsigned int number, unsigned int base, unsigned int size, int sp);
00011 unsigned int str2num(char *s, unsigned base);
00012 void reverse(char *str, int len);
00013 int intToStr(int x, char str[], int d);
00014 void float2str( char *res, float n, int afterpoint);
00015 double myPow(double x, int n) ;
00016 void flush_ch(char* ch, int ch_size);
00017 int size_ch(char* ch, int ch_size_max);
00018
00019
00020 #endif /* INC_UTIL_H_ */
```

5.20 VL53L0X.h

```
00001 //#define bool  uint8_t
00002 #define true  1
00003 #define false 0
00004
00005 #define SYSRANGE_START 0x00
```

```
00006 #define SYSTEM_THRESH_HIGH 0x0C
00007 #define SYSTEM_THRESH_LOW 0x0E
00008 #define SYSTEM_SEQUENCE_CONFIG 0x01
00009 #define SYSTEM_RANGE_CONFIG 0x09
00010 #define SYSTEM_INTERMEASUREMENT_PERIOD 0x04
00011
00012
00013 #define SYSTEM_INTERRUPT_GPIO_CONFIG 0x0A
00014
00015 //GPIO Config
00016 #define GPIO_HV_MUX_ACTIVE_HIGH 0x84
00017 #define SYSTEM_INTERRUPT_CLEAR 0x0B
00018 #define I2C_MODE 0x88
00019
00020 // Result registers
00021 #define RESULT_INTERRUPT_STATUS 0x13
00022 #define RESULT_RANGE_STATUS 0x14
00023 #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
00024 #define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
00025 #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
00026 #define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
00027 #define RESULT_PEAK_SIGNAL_RATE_REF 0xB6
00028
00029 //Algo Register
00030 #define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28
00031
00032 //Check limit register
00033 #define MSRC_CONFIG_CONTROL 0x60
00034 #define PRE_RANGE_CONFIG_MIN_SNR 0x27
00035 #define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
00036 #define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
00037 #define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
00038 #define FINAL_RANGE_CONFIG_MIN_SNR 0x67
00039 #define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
00040 #define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
00041 #define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44
00042
00043 // PRE RANGE registers
00044 #define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
00045 #define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
00046 #define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
00047 #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
00048 #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52
00049
00050 //Internal tuning registers
00051 #define INTERNAL_TUNING_1 0x91
00052 #define INTERNAL_TUNING_2 0xFF
00053
00054
00055 //Other registers
00056 #define SYSTEM_HISTOGRAM_BIN 0x81
00057 #define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
00058 #define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
00059 #define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
00060 #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
00061 #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
00062 #define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
00063 #define MSRC_CONFIG_TIMEOUT_MACROP 0x46
00064 #define GLOBAL_CONFIG_SPAD_ENABLES_REF0 0x0B0
00065 #define GLOBAL_CONFIG_SPAD_ENABLES_REF1 0x0B1
00066 #define GLOBAL_CONFIG_SPAD_ENABLES_REF2 0x0B2
00067 #define GLOBAL_CONFIG_SPAD_ENABLES_REF3 0x0B3
00068 #define GLOBAL_CONFIG_SPAD_ENABLES_REF4 0x0B4
00069 #define GLOBAL_CONFIG_SPAD_ENABLES_REF5 0x0B5
00070 #define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
00071 #define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
00072 #define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
00073 #define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
00074 #define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
00075 #define ALGO_PHASECAL_LIM 0x30
00076 #define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30
00077
00078
00079 #define SYSTEM_THRESH_HIGH                          0x0C
00080 #define SYSTEM_THRESH_LOW                          0x0E
00081
00082 #define  SYSTEM_SEQUENCE_CONFIG                      0x01
00083 #define  SYSTEM_RANGE_CONFIG                         0x09
00084 #define  SYSTEM_INTERMEASUREMENT_PERIOD              0x04
00085
00086 #define  SYSTEM_INTERRUPT_CONFIG_GPIO                0x0A
00087
00088 #define  GPIO_HV_MUX_ACTIVE_HIGH                     0x84
00089
00090 #define  SYSTEM_INTERRUPT_CLEAR                      0x0B
00091
00092 #define RESULT_INTERRUPT_STATUS                     0x13
```

```
00093 #define  RESULT_RANGE_STATUS                            0x14
00094
00095 #define  RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN          0xBC
00096 #define  RESULT_CORE_RANGING_TOTAL_EVENTS_RTN           0xC0
00097 #define  RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF          0xD0
00098 #define  RESULT_CORE_RANGING_TOTAL_EVENTS_REF           0xD4
00099 #define  RESULT_PEAK_SIGNAL_RATE_REF                    0xB6
00100
00101 #define  ALGO_PART_TO_PART_RANGE_OFFSET_MM              0x28
00102
00103 //#define  I2C_SLAVE_DEVICE_ADDRESS                      0x8A ################
00104 #define  I2C_SLAVE_DEVICE_ADDRESS                       0x53
00105
00106 #define  MSRC_CONFIG_CONTROL                            0x60
00107
00108 #define  PRE_RANGE_CONFIG_MIN_SNR                       0x27
00109 #define  PRE_RANGE_CONFIG_VALID_PHASE_LOW               0x56
00110 #define  PRE_RANGE_CONFIG_VALID_PHASE_HIGH              0x57
00111 #define  PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT             0x64
00112
00113 #define  FINAL_RANGE_CONFIG_MIN_SNR                     0x67
00114 #define  FINAL_RANGE_CONFIG_VALID_PHASE_LOW             0x47
00115 #define  FINAL_RANGE_CONFIG_VALID_PHASE_HIGH            0x48
00116 #define  FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT    0x44
00117
00118 #define  PRE_RANGE_CONFIG_SIGMA_THRESH_HI               0x61
00119 #define  PRE_RANGE_CONFIG_SIGMA_THRESH_LO               0x62
00120
00121 #define  PRE_RANGE_CONFIG_VCSEL_PERIOD                  0x50
00122 #define  PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI            0x51
00123 #define  PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO             0x52
00124
00125 #define  SYSTEM_HISTOGRAM_BIN                           0x81
00126 #define  HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT          0x33
00127 #define  HISTOGRAM_CONFIG_READOUT_CTRL                  0x55
00128
00129 #define  FINAL_RANGE_CONFIG_VCSEL_PERIOD                0x70
00130 #define  FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI           0x71
00131 #define  FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO           0x72
00132 #define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS         0x20
00133
00134 #define MSRC_CONFIG_TIMEOUT_MACROP                     0x46
00135
00136 #define  SOFT_RESET_GO2_SOFT_RESET_N                    0xBF
00137 #define  IDENTIFICATION_MODEL_ID                        0xC0
00138 #define  IDENTIFICATION_REVISION_ID                    0xC2
00139
00140 #define  OSC_CALIBRATE_VAL                              0xF8
00141
00142 #define  GLOBAL_CONFIG_VCSEL_WIDTH                      0x32
00143 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_0               0xB0
00144 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_1               0xB1
00145 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_2               0xB2
00146 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_3              0xB3
00147 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_4               0xB4
00148 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_5               0xB5
00149
00150 #define  GLOBAL_CONFIG_REF_EN_START_SELECT              0xB6
00151 #define  DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD            0x4E
00152 #define  DYNAMIC_SPAD_REF_EN_START_OFFSET               0x4F
00153 #define  POWER_MANAGEMENT_GO1_POWER_FORCE               0x80
00154
00155 #define  VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV              0x89
00156
00157 #define  ALGO_PHASECAL_LIM                              0x30
00158 #define  ALGO_PHASECAL_CONFIG_TIMEOUT                   0x30
00159
00160 //------------------------------------------------------------
00161 // Defines
00162 //------------------------------------------------------------
00163 // I use a 8-bit number for the address, LSB must be 0 so that I can
00164 // OR over the last bit correctly based on reads and writes
00165 #define ADDRESS_DEFAULT 0b01010010
00166
00167 // Record the current time to check an upcoming timeout against
00168 #define startTimeout() (g_timeoutStartMs = millis())
00169
00170 // Check if timeout is enabled (set to nonzero value) and has expired
00171 #define checkTimeoutExpired() (g_ioTimeout > 0 && ((uint16_t)millis() - g_timeoutStartMs) >
      g_ioTimeout)
00172
00173 // Decode VCSEL (vertical cavity surface emitting laser) pulse period in PCLKs
00174 // from register value
00175 // based on VL53L0X_decode_vcsel_period()
00176 #define decodeVcselPeriod(reg_val)      (((reg_val) + 1) « 1)
00177
00178 // Encode VCSEL pulse period register value from period in PCLKs
```

```
00179 // based on VL53L0X_encode_vcsel_period()
00180 #define encodeVcselPeriod(period_pclks) (((period_pclks) » 1) - 1)
00181
00182 // Calculate macro period in *nanoseconds* from VCSEL period in PCLKs
00183 // based on VL53L0X_calc_macro_period_ps()
00184 // PLL_period_ps = 1655; macro_period_vclks = 2304
00185 #define calcMacroPeriod(vcsel_period_pclks) ((((uint32_t)2304 * (vcsel_period_pclks) * 1655) + 500) /
      1000)
00186
00187 // register addresses from API vl53l0x_device.h (ordered as listed there)
00188
00189
00190 typedef enum { VcselPeriodPreRange, VcselPeriodFinalRange }vcselPeriodType;
00191
00192 // Additional info for one measurement
00193 typedef struct{
00194   uint16_t rawDistance; //uncorrected distance  [mm],   uint16_t
00195   uint16_t signalCnt;   //Signal  Counting Rate [mcps], uint16_t, fixpoint9.7
00196   uint16_t ambientCnt;  //Ambient Counting Rate [mcps], uint16_t, fixpoint9.7
00197   uint16_t spadCnt;     //Effective SPAD return count,  uint16_t, fixpoint8.8
00198   uint8_t  rangeStatus; //Ranging status (0-15)
00199 } statInfo_t;
00200
00201
00202 //------------------------------------------------------------
00203 // API Functions
00204 //------------------------------------------------------------
00205 // configures chip i2c and lib for `new_addr` (8 bit, LSB=0)
00206 void setAddress(uint8_t new_addr);
00207 // Returns the current I2C address.
00208 uint8_t getAddress(void);
00209
00210 // Iniitializes and configures the sensor.
00211 // If the optional argument io_2v8 is 1, the sensor is configured for 2V8 mode (2.8 V I/O);
00212 // if 0, the sensor is left in 1V8 mode. Returns 1 if the initialization completed successfully.
00213 uint8_t initVL53L0X();
00214
00215 // Sets the return signal rate limit to the given value in units of MCPS (mega counts per second).
00216 // This is the minimum amplitude of the signal reflected from the target and received by the sensor
00217 //  necessary for it to report a valid reading. Setting a lower limit increases the potential range
00218 // of the sensor but also increases the likelihood of getting an inaccurate reading because of
00219 //  reflections from objects other than the intended target. This limit is initialized to 0.25 MCPS
00220 //  by default. The return value is a boolean indicating whether the requested limit was valid.
00221 uint8_t setSignalRateLimit(float limit_Mcps);
00222
00223 // Returns the current return signal rate limit in MCPS.
00224 float getSignalRateLimit(void);
00225
00226 // Set the measurement timing budget in microseconds, which is the time allowed
00227 // for one measurement; the ST API and this library take care of splitting the
00228 // timing budget among the sub-steps in the ranging sequence. A longer timing
00229 // budget allows for more accurate measurements. Increasing the budget by a
00230 // factor of N decreases the range measurement standard deviation by a factor of
00231 // sqrt(N). Defaults to about 33 milliseconds; the minimum is 20 ms.
00232 // based on VL53L0X_set_measurement_timing_budget_micro_seconds()
00233 uint8_t setMeasurementTimingBudget(uint32_t budget_us);
00234
00235 // Returns the current measurement timing budget in microseconds.
00236 uint32_t getMeasurementTimingBudget(void);
00237
00238 // Sets the VCSEL (vertical cavity surface emitting laser) pulse period for the given period type
00239 // (VcselPeriodPreRange or VcselPeriodFinalRange) to the given value (in PCLKs).
00240 // Longer periods increase the potential range of the sensor. Valid values are (even numbers only):
00241 // Pre: 12 to 18 (initialized to 14 by default)
00242 // Final: 8 to 14 (initialized to 10 by default)
00243 // The return value is a boolean indicating whether the requested period was valid.
00244 uint8_t setVcselPulsePeriod(vcselPeriodType type, uint8_t period_pclks);
00245
00246 // Returns the current VCSEL pulse period for the given period type.
00247 uint8_t getVcselPulsePeriod(vcselPeriodType type);
00248
00249 // Starts continuous ranging measurements. If the argument period_ms is 0,
00250 // continuous back-to-back mode is used (the sensor takes measurements as often as possible);
00251 // if it is nonzero, continuous timed mode is used, with the specified inter-measurement period
00252 // in milliseconds determining how often the sensor takes a measurement.
00253 void startContinuous(uint32_t period_ms);
00254
00255 // Stops continuous mode.
00256 void stopContinuous(void);
00257
00258 // Returns a range reading in millimeters when continuous mode is active.
00259 // Additional measurement data will be copied into `extraStats` if it is non-zero.
00260 uint16_t readRangeContinuousMillimeters(/* statInfo_t *extraStats*/ );
00261
00262 // Performs a single-shot ranging measurement and returns the reading in millimeters.
00263 // Additional measurement data will be copied into `extraStats` if it is non-zero.
00264 uint16_t readRangeSingleMillimeters( /*statInfo_t *extraStats */);
```

```
00265
00266 // Sets a timeout period in milliseconds after which read operations will abort
00267 // if the sensor is not ready. A value of 0 disables the timeout.
00268 void setTimeout(uint16_t timeout);
00269
00270 // Returns the current timeout period setting.
00271 uint16_t getTimeout(void);
00272
00273 // Indicates whether a read timeout has occurred since the last call to timeoutOccurred().
00274 bool timeoutOccurred(void);
00275
00276 //------------------------------------------------------------
00277 // I2C communication Functions
00278 //------------------------------------------------------------
00279 void writeReg(uint8_t reg, uint8_t value);        // Write an 8-bit register
00280 void writeReg16Bit(uint8_t reg, uint16_t value);  // Write a 16-bit register
00281 void writeReg32Bit(uint8_t reg, uint32_t value);  // Write a 32-bit register
00282 uint8_t readReg(uint8_t reg);                     // Read an 8-bit register
00283 uint16_t readReg16Bit(uint8_t reg);               // Read a 16-bit register
00284 uint32_t readReg32Bit(uint8_t reg);               // Read a 32-bit register
00285 // Write `count' number of bytes from `src' to the sensor, starting at `reg'
00286 void writeMulti(uint8_t reg, uint8_t const *src, uint8_t count);
00287
00288 // TCC: Target CentreCheck
00289 // MSRC: Minimum Signal Rate Check
00290 // DSS: Dynamic Spad Selection
00291 typedef struct {
00292   uint8_t tcc, msrc, dss, pre_range, final_range;
00293 }SequenceStepEnables;
00294
00295 typedef struct {
00296   uint16_t pre_range_vcsel_period_pclks, final_range_vcsel_period_pclks;
00297
00298   uint16_t msrc_dss_tcc_mclks, pre_range_mclks, final_range_mclks;
00299   uint32_t msrc_dss_tcc_us,    pre_range_us,    final_range_us;
00300 }SequenceStepTimeouts;
00301
```

5.21 captDistIR.c

```
00001 /*
00002  * IRMeasure.c
00003  */
00004
00005
00006 #include "captDistIR.h"
00007
00008 ADC_HandleTypeDef    adcHandle;
00009 ADC_HandleTypeDef    adcHandle_12;
00010 ADC_HandleTypeDef    adcHandle_13;
00011 ADC_ChannelConfTypeDef    sConfig;
00012
00013 //================================================================
00014 //          ADC INIT FOR IR SENSOR SHARP GP2D12
00015 //================================================================
00016
00017 void  captDistIR_Init(void)
00018 {
00019     adcHandle.Instance      = ADC1;
00020
00021     adcHandle.Init.ClockPrescaler = ADC_CLOCKPRESCALER_PCLK_DIV2;
00022     adcHandle.Init.DataAlign = ADC_DATAALIGN_RIGHT;
00023     adcHandle.Init.Resolution = ADC_RESOLUTION12b;
00024     // Don't do continuous conversions - do them on demand
00025     adcHandle.Init.ContinuousConvMode    = DISABLE;  // Continuous mode disabled to have only 1
      conversion at each conversion trig
00026     // Disable the scan conversion so we do one at a time */
00027     adcHandle.Init.ScanConvMode = DISABLE;
00028     //Say how many channels would be used by the sequencer
00029     adcHandle.Init.NbrOfConversion = 2;
00030     adcHandle.Init.DiscontinuousConvMode = DISABLE;  // Parameter discarded because sequencer is
      disabled
00031     adcHandle.Init.NbrOfDiscConversion = 2;
00032     adcHandle.Init.ExternalTrigConvEdge =  ADC_EXTERNALTRIGCONVEDGE_NONE ;
00033     //Start conversion by software, not an external trigger
00034     adcHandle.Init.ExternalTrigConv = 0;
00035     adcHandle.Init.DMAContinuousRequests = DISABLE;
00036     adcHandle.Init.EOCSelection = DISABLE;
00037
00038     HAL_ADC_Init(&adcHandle);
00039 }
00040
00041 //================================================================
```

```
00042 //          IR GET (POLL METHOD)
00043 //=================================================================
00044
00045 int  captDistIR_Get(int* tab)
00046 {
00047     sConfig.Channel      = ADC_CHANNEL_4;
00048     sConfig.Rank         = 1;
00049     sConfig.SamplingTime = ADC_SAMPLETIME_56CYCLES;
00050     HAL_ADC_ConfigChannel(&adcHandle, &sConfig);
00051
00052     HAL_ADC_Start(&adcHandle);                  //Start the conversion
00053     HAL_ADC_PollForConversion(&adcHandle,10);   //Processing the conversion
00054     tab[0]=HAL_ADC_GetValue(&adcHandle);        //Return the converted data
00055
00056     sConfig.Channel      = ADC_CHANNEL_8;
00057     sConfig.Rank         = 1;
00058     sConfig.SamplingTime = ADC_SAMPLETIME_56CYCLES;
00059     HAL_ADC_ConfigChannel(&adcHandle, &sConfig);
00060
00061     HAL_ADC_Start(&adcHandle);                      //Start the conversion
00062     HAL_ADC_PollForConversion(&adcHandle,10);       //Processing the conversion
00063     tab[1]=HAL_ADC_GetValue(&adcHandle);            //Return the converted data
00064
00065     return 0;
00066 }
00067 //=================================================================
```

5.22 dma_transport.c

```
00001 #include <uxr/client/transport.h>
00002
00003 #include <rmw_microxrcedds_c/config.h>
00004
00005 #include "main.h"
00006 #include "cmsis_os.h"
00007
00008 #include <unistd.h>
00009 #include <stdio.h>
00010 #include <string.h>
00011 #include <stdbool.h>
00012
00013 #ifdef RMW_UXRCE_TRANSPORT_CUSTOM
00014
00015 // --- micro-ROS Transports ---
00016 #define UART_DMA_BUFFER_SIZE 2048
00017
00018 static uint8_t dma_buffer[UART_DMA_BUFFER_SIZE];
00019 static size_t dma_head = 0, dma_tail = 0;
00020
00021 bool cubemx_transport_open(struct uxrCustomTransport * transport){
00022     UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;
00023     HAL_UART_Receive_DMA(uart, dma_buffer, UART_DMA_BUFFER_SIZE);
00024     return true;
00025 }
00026
00027 bool cubemx_transport_close(struct uxrCustomTransport * transport){
00028     UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;
00029     HAL_UART_DMAStop(uart);
00030     return true;
00031 }
00032
00033 size_t cubemx_transport_write(struct uxrCustomTransport* transport, uint8_t * buf, size_t len, uint8_t
      * err){
00034     UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;
00035
00036     HAL_StatusTypeDef ret;
00037     if (uart->gState == HAL_UART_STATE_READY){
00038         ret = HAL_UART_Transmit_DMA(uart, buf, len);
00039         while (ret == HAL_OK && uart->gState != HAL_UART_STATE_READY){
00040             osDelay(1);
00041         }
00042
00043         return (ret == HAL_OK) ? len : 0;
00044     }else{
00045         return 0;
00046     }
00047 }
00048
00049 size_t cubemx_transport_read(struct uxrCustomTransport* transport, uint8_t* buf, size_t len, int
      timeout, uint8_t* err){
00050     UART_HandleTypeDef * uart = (UART_HandleTypeDef*) transport->args;
00051
00052     int ms_used = 0;
```

```
00053     do
00054     {
00055         __disable_irq();
00056         dma_tail = UART_DMA_BUFFER_SIZE - __HAL_DMA_GET_COUNTER(uart->hdmarx);
00057         __enable_irq();
00058         ms_used++;
00059         osDelay(portTICK_RATE_MS);
00060     } while (dma_head == dma_tail && ms_used < timeout);
00061
00062     size_t wrote = 0;
00063     while ((dma_head != dma_tail) && (wrote < len)){
00064         buf[wrote] = dma_buffer[dma_head];
00065         dma_head = (dma_head + 1) % UART_DMA_BUFFER_SIZE;
00066         wrote++;
00067     }
00068
00069     return wrote;
00070 }
00071
00072 #endif //RMW_UXRCE_TRANSPORT_CUSTOM
```

5.23 drv_gpio.c

```
00001 #include "main.h"
00002 #include "drv_gpio.h"
00003
00004 void MX_GPIO_Init(void)
00005 {
00006   GPIO_InitTypeDef GPIO_InitStruct = {0};
00007
00008   /* GPIO Ports Clock Enable */
00009   __HAL_RCC_GPIOC_CLK_ENABLE();
00010   __HAL_RCC_GPIOH_CLK_ENABLE();
00011   __HAL_RCC_GPIOA_CLK_ENABLE();
00012   __HAL_RCC_GPIOB_CLK_ENABLE();
00013
00014   /*Configure GPIO pin Output Level */
00015   HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
00016
00017   /*Configure GPIO pin : B1_Pin */
00018   GPIO_InitStruct.Pin = B1_Pin;
00019   GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
00020   GPIO_InitStruct.Pull = GPIO_NOPULL;
00021   HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
00022
00023   /*Configure GPIO pin : LD2_Pin */
00024   GPIO_InitStruct.Pin = LD2_Pin;
00025   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00026   GPIO_InitStruct.Pull = GPIO_NOPULL;
00027   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
00028   HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
00029
00030 }
00031
00032 extern void quadEncoder_CallbackIndexL(void);
00033 extern void quadEncoder_CallbackIndexR(void);
00034
00035 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
00036 {
00037     switch(GPIO_Pin)
00038     {
00039     case GPIO_PIN_0 :
00040         quadEncoder_CallbackIndexR();
00041                     break;
00042
00043     case GPIO_PIN_1 :
00044
00045                     break;
00046
00047     case GPIO_PIN_3:
00048                     break;
00049
00050     case GPIO_PIN_10:
00051         quadEncoder_CallbackIndexL();
00052                     break;
00053
00054     case GPIO_PIN_13 :      // USER BUTTON
00055                     break;
00056
00057     default :       break;
00058
00059
00060     }
00061 }
```

5.24 drv_i2c.c

```
00001 #include "main.h"
00002 #include <string.h>
00003 #include "drv_i2c.h"
00004
00005
00006 I2C_HandleTypeDef hi2c1;
00007
00008 void MX_I2C1_Init(void)
00009 {
00010
00011   hi2c1.Instance = I2C1;
00012   hi2c1.Init.ClockSpeed = 100000;
00013   hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
00014   hi2c1.Init.OwnAddress1 = 0;
00015   hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
00016   hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
00017   hi2c1.Init.OwnAddress2 = 0;
00018   hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
00019   hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
00020   if (HAL_I2C_Init(&hi2c1) != HAL_OK)
00021   {
00022     Error_Handler();
00023   }
00024
00025 }
00026
00027
00028
00029
00030
00031 //=========================================================================
00032 // Transmit n_data bytes to i2c slave
00033 //=========================================================================
00034 int i2c1_WriteBuffer(uint16_t addrSlave, uint8_t *data, int n_data)
00035 {
00036     int status;
00037     status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, data, n_data , 100);
00038     return status;
00039 }
00040 //=========================================================================
00041 // Receive n_data bytes from i2c slave
00042 //=========================================================================
00043 int i2c1_ReadBuffer(uint16_t addrSlave, uint8_t *data, int n_data)
00044 {
00045     int status;
00046     status = HAL_I2C_Master_Receive(&hi2c1, addrSlave, data, n_data , 100);
00047     return status;
00048 }
00049 //=========================================================================
00050 // Receive n_data bytes - located at regAddr - from i2c slave
00051 //=========================================================================
00052 int i2c1_ReadRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data)
00053 {
00054     int status;
00055     uint8_t RegAddr;
00056     RegAddr=regAddr;
00057     do{
00058         status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, &RegAddr, 1, 100);
00059         if( status )
00060             break;
00061         status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, data, n_data, n_data*100);
00062     }while(0);
00063     return status;
00064 }
00065
00066 //=========================================================================
00067 // Write n_data bytes - have to be written at regAddr - to i2c slave
00068 //=========================================================================
00069 int i2c1_WriteRegBuffer(uint16_t addrSlave, uint8_t  regAddr,  uint8_t *data, int n_data)
00070 {
00071     int status;
00072     uint8_t RegAddr[0x10];
00073     RegAddr[0]=regAddr;
00074     memcpy(RegAddr+1, data, n_data);
00075    status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, RegAddr, n_data+1, 100);
00076      return status;
00077 }
00078
00079 //=========================================================================
00080 // Write 1 byte at regAddr Slave - Interrupt Method
00081 //=========================================================================
00082 void i2c1_WriteRegByte_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t data)
00083 {
00084
00085 uint8_t datas_to_send[2];
```

```
00086
00087 datas_to_send[0]=regAddr;
00088 datas_to_send[1]=data;
00089
00090     while(HAL_I2C_Master_Transmit_IT(&hi2c1, addrSlave, datas_to_send, 2)!= HAL_OK){}
00091     while (HAL_I2C_GetState(&hi2c1) != HAL_I2C_STATE_READY){}
00092 }
00093 //=========================================================================
00094 // Read 1 byte from regAddr Slave - Interrupt Method
00095 //=========================================================================
00096 void i2c1_ReadRegBuffer_IT(uint16_t addrSlave, uint8_t  regAddr,  uint8_t* datas, int len)
00097 {
00098     while(HAL_I2C_Master_Transmit_IT(&hi2c1, addrSlave, &regAddr, 1)!= HAL_OK){}
00099     while (HAL_I2C_GetState(&hi2c1) != HAL_I2C_STATE_READY){}
00100
00101     while(HAL_I2C_Master_Receive_IT(&hi2c1, addrSlave, datas, len)!= HAL_OK){}
00102     while( HAL_I2C_GetState(&hi2c1) != HAL_I2C_STATE_READY ){}
00103 }
00104
00105 //=========================================================================
00106 // Write 1 byte to regAddr (16 bits) Slave
00107 //=========================================================================
00108 int i2c1_WriteReg16Byte(uint16_t addrSlave, uint16_t  regAddr,  uint8_t data)
00109 {
00110     int status;
00111     uint8_t buffer[3];
00112     buffer[0]=regAddr»8;
00113     buffer[1]=regAddr&0xFF;
00114     buffer[2]=data;
00115
00116     status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 3 , 100);
00117     return status;
00118
00119 }
00120 //=========================================================================
00121 // Write 16 bits word to regAddr (16 bits) Slave
00122 //=========================================================================
00123 int i2c1_WriteReg16Word16(uint16_t addrSlave, uint16_t  regAddr,  uint16_t data)
00124 {
00125     int status;
00126     uint8_t buffer[4];
00127     buffer[0]=regAddr»8;
00128     buffer[1]=regAddr&0xFF;
00129     buffer[2]=data»8;
00130     buffer[3]=data&0xFF;
00131
00132     status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 4 , 100);
00133     return status;
00134 }
00135 //=========================================================================
00136 // Write 32 bits word to regAddr (16 bits) Slave
00137 //=========================================================================
00138 int i2c1_WriteReg16Word32(uint16_t addrSlave, uint16_t  regAddr,  uint32_t data)
00139 {
00140     int status;
00141     uint8_t buffer[4];
00142     buffer[0]=regAddr»8;
00143     buffer[1]=regAddr&0xFF;
00144     buffer[2]=data»24;
00145     buffer[3]=(data»16)&0xFF;
00146     buffer[4]=(data»8)&0xFF;;
00147     buffer[5]=data&0xFF;
00148
00149     status = HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 6 , 100);
00150     return status;
00151 }
00152 //=========================================================================
00153 // Read 1 byte from regAddr (16 bits) Slave
00154 //=========================================================================
00155 int i2c1_ReadReg16Byte(uint16_t addrSlave, uint16_t  regAddr, uint8_t *data)
00156 {
00157     int status;
00158     uint8_t buffer[2];
00159
00160     buffer[0]=regAddr»8;
00161     buffer[1]=regAddr&0xFF;
00162
00163     status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
00164      if(!status ) {
00165         status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, buffer, 1, 100);
00166         if( !status ){
00167             *data=buffer[0];
00168         }
00169      }
00170
00171     return status;
00172 }
```

```
00173 //=========================================================================
00174 // Read 16 bits word from regAddr (16 bits) Slave
00175 //=========================================================================
00176 int i2c1_ReadReg16Word16(uint16_t addrSlave, uint16_t  regAddr, uint16_t *data)
00177 {
00178     int status;
00179     uint8_t buffer[2];
00180 
00181     buffer[0]=regAddr»8;
00182     buffer[1]=regAddr&0xFF;
00183 
00184     status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
00185      if(!status ) {
00186         status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, buffer, 2, 100);
00187         if( !status ){
00188               //VL6180x register are Big endian if cpu is be direct read direct into *data is possible
00189              *data=((uint16_t)buffer[0]«8)|(uint16_t)buffer[1];
00190         }
00191      }
00192 
00193     return status;
00194 }
00195 //=========================================================================
00196 // Read 32 bits word from regAddr (16 bits) Slave
00197 //=========================================================================
00198 int i2c1_ReadReg16Word32(uint16_t addrSlave, uint16_t  regAddr, uint32_t *data)
00199 {
00200     int status;
00201     uint8_t buffer[4];
00202 
00203     buffer[0]=regAddr»8;
00204     buffer[1]=regAddr&0xFF;
00205 
00206     status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
00207      if(!status ) {
00208         status =HAL_I2C_Master_Receive(&hi2c1, addrSlave, buffer, 4, 100);
00209         if( !status ){
00210               //VL6180x register are Big endian if cpu is be direct read direct into *data is possible
00211 
      *data=((uint32_t)buffer[0]«24)|((uint32_t)buffer[1]«16)|((uint32_t)buffer[2]«8)|((uint32_t)buffer[3]);
00212         }
00213      }
00214 
00215     return status;
00216 }
00217 
00218 //=========================================================================
00219 // Read n_data bytes from regAddr (16 bits) Slave
00220 //=========================================================================
00221 int i2c1_ReadReg16Buffer(uint16_t addrSlave, uint16_t  regAddr,  uint8_t *data, int n_data)
00222 {
00223     int status;
00224     uint8_t buffer[2];
00225 
00226     buffer[0]=regAddr»8;
00227     buffer[1]=regAddr&0xFF;
00228 
00229     status=HAL_I2C_Master_Transmit(&hi2c1, addrSlave, buffer, 2, 100);
00230     if( !status ){
00231         status=HAL_I2C_Master_Receive(&hi2c1, addrSlave, data, n_data, n_data*100);
00232     }
00233 
00234 
00235 
00236     return status;
00237 }
00238 
00239 
00240 
00241 
00242 
```

5.25 drv_uart.c

```
00001 #include "main.h"
00002 #include "drv_uart.h"
00003 
00004 UART_HandleTypeDef huart1;
00005 UART_HandleTypeDef huart2;
00006 DMA_HandleTypeDef hdma_usart1_rx;
00007 DMA_HandleTypeDef hdma_usart1_tx;
00008 DMA_HandleTypeDef hdma_usart2_rx;
00009 DMA_HandleTypeDef hdma_usart2_tx;
```

```
00010
00011
00012 void MX_USART1_UART_Init(void)
00013 {
00014   huart1.Instance = USART1;
00015   huart1.Init.BaudRate = 115200;
00016   huart1.Init.WordLength = UART_WORDLENGTH_8B;
00017   huart1.Init.StopBits = UART_STOPBITS_1;
00018   huart1.Init.Parity = UART_PARITY_NONE;
00019   huart1.Init.Mode = UART_MODE_TX_RX;
00020   huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
00021   huart1.Init.OverSampling = UART_OVERSAMPLING_16;
00022   if (HAL_UART_Init(&huart1) != HAL_OK)
00023   {
00024     Error_Handler();
00025   }
00026 }
00027
00028
00029 void MX_USART2_UART_Init(void)
00030 {
00031   huart2.Instance = USART2;
00032   huart2.Init.BaudRate = 115200;
00033   huart2.Init.WordLength = UART_WORDLENGTH_8B;
00034   huart2.Init.StopBits = UART_STOPBITS_1;
00035   huart2.Init.Parity = UART_PARITY_NONE;
00036   huart2.Init.Mode = UART_MODE_TX_RX;
00037   huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
00038   huart2.Init.OverSampling = UART_OVERSAMPLING_16;
00039   if (HAL_UART_Init(&huart2) != HAL_OK)
00040   {
00041     Error_Handler();
00042   }
00043 }
00044
00048 void MX_DMA_Init(void)
00049 {
00050
00051   /* DMA controller clock enable */
00052   __HAL_RCC_DMA2_CLK_ENABLE();
00053   __HAL_RCC_DMA1_CLK_ENABLE();
00054
00055   /* DMA interrupt init */
00056   /* DMA1_Stream5_IRQn interrupt configuration */
00057   HAL_NVIC_SetPriority(DMA1_Stream5_IRQn, 5, 0);
00058   HAL_NVIC_EnableIRQ(DMA1_Stream5_IRQn);
00059   /* DMA1_Stream6_IRQn interrupt configuration */
00060   HAL_NVIC_SetPriority(DMA1_Stream6_IRQn, 5, 0);
00061   HAL_NVIC_EnableIRQ(DMA1_Stream6_IRQn);
00062   /* DMA2_Stream2_IRQn interrupt configuration */
00063   HAL_NVIC_SetPriority(DMA2_Stream2_IRQn, 5, 0);
00064   HAL_NVIC_EnableIRQ(DMA2_Stream2_IRQn);
00065   /* DMA2_Stream7_IRQn interrupt configuration */
00066   HAL_NVIC_SetPriority(DMA2_Stream7_IRQn, 5, 0);
00067   HAL_NVIC_EnableIRQ(DMA2_Stream7_IRQn);
00068
00069 }
```

5.26 freertos.c

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Includes ------------------------------------------------------------------*/
00021 #include "FreeRTOS.h"
00022 #include "task.h"
00023 #include "main.h"
00024
00025 /* Private includes ----------------------------------------------------------*/
00026 /* USER CODE BEGIN Includes */
00027
00028 /* USER CODE END Includes */
00029
00030 /* Private typedef -----------------------------------------------------------*/
00031 /* USER CODE BEGIN PTD */
00032
00033 /* USER CODE END PTD */
00034
00035 /* Private define ------------------------------------------------------------*/
00036 /* USER CODE BEGIN PD */
00037
00038 /* USER CODE END PD */
00039
```

```
00040 /* Private macro -------------------------------------------------------------*/
00041 /* USER CODE BEGIN PM */
00042
00043 /* USER CODE END PM */
00044
00045 /* Private variables ---------------------------------------------------------*/
00046 /* USER CODE BEGIN Variables */
00047
00048 /* USER CODE END Variables */
00049
00050 /* Private function prototypes -----------------------------------------------*/
00051 /* USER CODE BEGIN FunctionPrototypes */
00052
00053 /* USER CODE END FunctionPrototypes */
00054
00055 /* Private application code --------------------------------------------------*/
00056 /* USER CODE BEGIN Application */
00057
00058 /* USER CODE END Application */
00059
```

5.27 groveLCD.c

```
00001 /*
00002  * groveLCD.c
00003  *
00004  *  Created on: Jan 8, 2020
00005  *      Author: kerhoas
00006  */
00007
00008 #include "groveLCD.h"
00009 #include <math.h>
00010 #include "util.h"
00011
00012   uint8_t _displayfunction;
00013   uint8_t _displaycontrol;
00014   uint8_t _displaymode;
00015   uint8_t _initialized;
00016   uint8_t _numlines,_currline;
00017
00018 //=====================================================================
00019 void groveLCD_test()
00020 {
00021     uint8_t tab[2];
00022     tab[1] = 100;
00023   i2c1_WriteRegBuffer(RGB_ADDRESS, REG_RED, tab, 1);
00024
00025 }
00026 //=====================================================================
00027 void i2c_send_byte(unsigned char dta)
00028 {
00029     i2c1_WriteBuffer(LCD_ADDRESS, &dta, 1);
00030 }
00031 //=====================================================================
00032 void i2c_send_byteS(unsigned char *dta, unsigned char len)
00033 {
00034     i2c1_WriteBuffer(LCD_ADDRESS, dta, len);
00035 }
00036 //=====================================================================
00037 void groveLCD_begin(uint8_t cols, uint8_t lines, uint8_t dotsize)
00038 {
00039     if (lines > 1) {
00040         _displayfunction |= LCD_2LINE;
00041     }
00042     _numlines = lines;
00043     _currline = 0;
00044
00045     // for some 1 line displays you can select a 10 pixel high font
00046     if ((dotsize != 0) && (lines == 1)) {
00047         _displayfunction |= LCD_5x10DOTS;
00048     }
00049
00050     // SEE PAGE 45/46 FOR INITIALIZATION SPECIFICATION!
00051     // according to datasheet, we need at least 40ms after power rises above 2.7V
00052     // before sending commands. Arduino can turn on way befer 4.5V so we'll wait 50
00053     HAL_Delay(50);
00054
00055
00056     // this is according to the hitachi HD44780 datasheet
00057     // page 45 figure 23
00058
00059     // Send function set command sequence
00060     groveLCD_command(LCD_FUNCTIONSET | _displayfunction);
```

```
00061     HAL_Delay(5);  // wait more than 4.1ms
00062
00063     // second try
00064     groveLCD_command(LCD_FUNCTIONSET | _displayfunction);
00065     HAL_Delay(5);
00066
00067     // third go
00068     groveLCD_command(LCD_FUNCTIONSET | _displayfunction);
00069
00070
00071     // finally, set # lines, font size, etc.
00072     groveLCD_command(LCD_FUNCTIONSET | _displayfunction);
00073
00074     // turn the display on with no cursor or blinking default
00075     _displaycontrol = LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKOFF;
00076     groveLCD_display();
00077
00078     // clear it off
00079     groveLCD_clear();
00080
00081     // Initialize to default text direction (for romance languages)
00082     _displaymode = LCD_ENTRYLEFT | LCD_ENTRYSHIFTDECREMENT;
00083     // set the entry mode
00084     groveLCD_command(LCD_ENTRYMODESET | _displaymode);
00085
00086
00087     // backlight init
00088     groveLCD_setReg(REG_MODE1, 0);
00089     // set LEDs controllable by both PWM and GRPPWM registers
00090     groveLCD_setReg(REG_OUTPUT, 0xFF);
00091     // set MODE2 values
00092     // 0010 0000 -> 0x20  (DMBLNK to 1, ie blinky mode)
00093     groveLCD_setReg(REG_MODE2, 0x20);
00094
00095     groveLCD_setColorWhite();
00096
00097 }
00098 //===========================================================
00099 void groveLCD_setColorAll(){groveLCD_setRGB(0, 0, 0);}
00100 void groveLCD_setColorWhite(){groveLCD_setRGB(255, 255, 255);}
00101 //===========================================================
00102
00103 /********** high level commands, for the user! */
00104 void groveLCD_clear()
00105 {
00106     groveLCD_command(LCD_CLEARDISPLAY);        // clear display, set cursor position to zero
00107     HAL_Delay(2000);          // this command takes a long time!
00108 }
00109 //===========================================================
00110 void groveLCD_home()
00111 {
00112     groveLCD_command(LCD_RETURNHOME);        // set cursor position to zero
00113     HAL_Delay(2000);        // this command takes a long time!
00114 }
00115 //===========================================================
00116 void groveLCD_setCursor(uint8_t col, uint8_t row)
00117 {
00118     col = (row == 0 ? col|0x80 : col|0xc0);
00119     unsigned char dta[2] = {0x80, col};
00120     i2c_send_byteS(dta, 2);
00121 }
00122 //===========================================================
00123 // Turn the display on/off (quickly)
00124 void groveLCD_noDisplay()
00125 {
00126     _displaycontrol &= ~LCD_DISPLAYON;
00127     groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
00128 }
00129 //===========================================================
00130 void groveLCD_display() {
00131     _displaycontrol |= LCD_DISPLAYON;
00132     groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
00133 }
00134 //===========================================================
00135 // Turns the underline cursor on/off
00136 void groveLCD_noCursor()
00137 {
00138     _displaycontrol &= ~LCD_CURSORON;
00139     groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
00140 }
00141 //===========================================================
00142 void groveLCD_cursor() {
00143     _displaycontrol |= LCD_CURSORON;
00144     groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
00145 }
00146 //===========================================================
00147 // Turn on and off the blinking cursor
```

```
00148 void groveLCD_noBlink()
00149 {
00150     _displaycontrol &= ~LCD_BLINKON;
00151     groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
00152 }
00153 //========================================================================
00154 void groveLCD_blink()
00155 {
00156     _displaycontrol |= LCD_BLINKON;
00157     groveLCD_command(LCD_DISPLAYCONTROL | _displaycontrol);
00158 }
00159 //========================================================================
00160 // These commands scroll the display without changing the RAM
00161 void groveLCD_scrollDisplayLeft(void)
00162 {
00163     groveLCD_command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVELEFT);
00164 }
00165 //========================================================================
00166 void groveLCD_scrollDisplayRight(void)
00167 {
00168     groveLCD_command(LCD_CURSORSHIFT | LCD_DISPLAYMOVE | LCD_MOVERIGHT);
00169 }
00170 //========================================================================
00171 // This is for text that flows Left to Right
00172 void groveLCD_leftToRight(void)
00173 {
00174     _displaymode |= LCD_ENTRYLEFT;
00175     groveLCD_command(LCD_ENTRYMODESET | _displaymode);
00176 }
00177 //========================================================================
00178 // This is for text that flows Right to Left
00179 void groveLCD_rightToLeft(void)
00180 {
00181     _displaymode &= ~LCD_ENTRYLEFT;
00182     groveLCD_command(LCD_ENTRYMODESET | _displaymode);
00183 }
00184 //========================================================================
00185 // This will 'right justify' text from the cursor
00186 void groveLCD_autoscroll(void)
00187 {
00188     _displaymode |= LCD_ENTRYSHIFTINCREMENT;
00189     groveLCD_command(LCD_ENTRYMODESET | _displaymode);
00190 }
00191 //========================================================================
00192 // This will 'left justify' text from the cursor
00193 void groveLCD_noAutoscroll(void)
00194 {
00195     _displaymode &= ~LCD_ENTRYSHIFTINCREMENT;
00196     groveLCD_command(LCD_ENTRYMODESET | _displaymode);
00197 }
00198 //========================================================================
00199 // Allows us to fill the first 8 CGRAM locations
00200 // with custom characters
00201 void groveLCD_createChar(uint8_t location, uint8_t charmap[])
00202 {
00203     location &= 0x7; // we only have 8 locations 0-7
00204     groveLCD_command(LCD_SETCGRAMADDR | (location << 3));
00205
00206     unsigned char dta[9];
00207     dta[0] = 0x40;
00208     for(int i=0; i<8; i++)
00209     {
00210         dta[i+1] = charmap[i];
00211     }
00212     i2c_send_byteS(dta, 9);
00213 }
00214 //========================================================================
00215 // Control the backlight LED blinking
00216 void groveLCD_blinkLED(void)
00217 {
00218     // blink period in seconds = (<reg 7> + 1) / 24
00219     // on/off ratio = <reg 6> / 256
00220     groveLCD_setReg(0x07, 0x17);  // blink every second
00221     groveLCD_setReg(0x06, 0x7f);  // half on, half off
00222 }
00223 //========================================================================
00224 void groveLCD_noBlinkLED(void)
00225 {
00226     groveLCD_setReg(0x07, 0x00);
00227     groveLCD_setReg(0x06, 0xff);
00228 }
00229 //========================================================================
00230 /********** mid level commands, for sending data/cmds */
00231
00232 // send command
00233 void groveLCD_command(uint8_t value)
00234 {
```

```
00235     unsigned char dta[2] = {0x80, value};
00236     i2c_send_byteS(dta, 2);
00237 }
00238 //=================================================================
00239 // send data
00240 int groveLCD_write(uint8_t value)
00241 {
00242     unsigned char dta[2] = {0x40, value};
00243     i2c_send_byteS(dta, 2);
00244     return 1; // assume sucess
00245 }
00246 //=================================================================
00247 void groveLCD_putString(char* s)
00248 {
00249     while(*s != '\0')
00250     {
00251         groveLCD_write(*s);
00252         s++;
00253     }
00254 }
00255 //=================================================================
00256 void groveLCD_setReg(unsigned char addr, unsigned char dta)
00257 {
00258     i2c1_WriteRegBuffer(RGB_ADDRESS, addr,  &dta, 1);
00259 }
00260 //=================================================================
00261 void groveLCD_setRGB(unsigned char r, unsigned char g, unsigned char b)
00262 {
00263     groveLCD_setReg(REG_RED, r);
00264     groveLCD_setReg(REG_GREEN, g);
00265     groveLCD_setReg(REG_BLUE, b);
00266 }
00267 //=================================================================
00268 const unsigned char color_define[4][3] =
00269 {
00270     {255, 255, 255},            // white
00271     {255, 0, 0},                // red
00272     {0, 255, 0},                // green
00273     {0, 0, 255},                // blue
00274 };
00275 //=================================================================
00276 void groveLCD_setColor(unsigned char color)
00277 {
00278     if(color > 3)return ;
00279     groveLCD_setRGB(color_define[color][0], color_define[color][1], color_define[color][2]);
00280 }
00281 //=================================================================
00282 void groveLCD_term_printf(const char* fmt, ...)
00283 {
00284     __gnuc_va_list         ap;
00285     char          *p;
00286     char           ch;
00287     unsigned long  ul;
00288     unsigned long long ull;
00289     unsigned long  size;
00290     unsigned int   sp;
00291     char           s[60];
00292     int first=0;
00293
00294     va_start(ap, fmt);
00295
00296     while (*fmt != '\0') {
00297         if (*fmt =='%') {
00298             size=0; sp=1;
00299             if (*++fmt=='0') {fmt++; sp=0;} // parse %04d --> sp=0
00300             ch=*fmt;
00301             if ((ch>'0') && (ch<='9')) {    // parse %4d --> size=4
00302                 char tmp[10];
00303                 int i=0;
00304                 while ((ch>='0') && (ch<='9')) {
00305                     tmp[i++]=ch;
00306                     ch=*++fmt;
00307                 }
00308                 tmp[i]='\0';
00309                 size=str2num(tmp,10);
00310             }
00311             switch (ch) {
00312                 case '%':
00313                     groveLCD_write('%');
00314                     break;
00315                 case 'c':
00316                     ch = va_arg(ap, int);
00317                     groveLCD_write(ch);
00318                     break;
00319                 case 's':
00320                     p = va_arg(ap, char *);
00321                     groveLCD_putString(p);
```

```
00322                   break;
00323               case 'd':
00324                   ul = va_arg(ap, long);
00325                   if ((long)ul < 0) {
00326                       groveLCD_write('-');
00327                       ul = -(long)ul;
00328                       //size--;
00329                   }
00330                   num2str(s, ul, 10, size, sp);
00331                   groveLCD_putString(s);
00332                   break;
00333               case 'u':
00334                   ul = va_arg(ap, unsigned int);
00335                   num2str(s, ul, 10, size, sp);
00336                   groveLCD_putString(s);
00337                   break;
00338               case 'o':
00339                   ul = va_arg(ap, unsigned int);
00340                   num2str(s, ul, 8, size, sp);
00341                   groveLCD_putString(s);
00342                   break;
00343               case 'p':
00344                   groveLCD_write('0');
00345                   groveLCD_write('x');
00346                   ul = va_arg(ap, unsigned int);
00347                   num2str(s, ul, 16, size, sp);
00348                   groveLCD_putString(s);
00349                   break;
00350               case 'x':
00351                   ul = va_arg(ap, unsigned int);
00352                   num2str(s, ul, 16, size, sp);
00353                   groveLCD_putString(s);
00354                   break;
00355               case 'f':
00356                   if(first==0){ ull = va_arg(ap, long long unsigned int); first = 1;}
00357                   ull = va_arg(ap, long long unsigned int);
00358                   int sign = ( ull & 0x80000000 ) >> 31;
00359                   int m = (ull & 0x000FFFFF) ; // should be 0x007FFFFF
00360                   float mf = (float)m ;
00361                   mf = mf / pow(2.0,20.0);
00362                   mf = mf + 1.0;
00363                   int e = ( ull & 0x78000000 ) >> 23 ; // should be int e = ( ul & 0x7F800000 ) >> 23;
00364                   e = e | (( ull & 0x000F00000 ) >> 20);
00365                   e = e - 127;
00366                   float f = mf*myPow(2.0,e);
00367                   if(sign==1){ groveLCD_write('-'); }
00368                   float2str((char*)s, f, 5);
00369                   groveLCD_putString((char*)s);
00370                   break;
00371
00372               default:
00373                   groveLCD_write(*fmt);
00374           }
00375       } else groveLCD_write(*fmt);
00376       fmt++;
00377   }
00378   va_end(ap);
00379 }
00380 //=================================================================
00381
00382
00383
00384
```

5.28 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/main.c File Reference

file that contain the main code

```
#include "main.h"
#include "motorCommand.h"
#include "quadEncoder.h"
#include "captDistIR.h"
#include "VL53L0X.h"
#include "groveLCD.h"
```

Data Structures

- struct AMessage
- struct MicroRosPubMsg
- struct MicroRosSubMsg

Macros

- #define SAMPLING_PERIOD_ms 5

config exo

- #define EXSTARTUP 0
- #define EXTEST_UART2 1
- #define EXCORRECTOR 2
- #define EXTESTCORRECTOR 3
- #define EXTEST_VL53 4
- #define EXTEST_MICROROS 5
- #define EXFINAL 6
- #define SYNCHRO_EX EXFINAL

config robot

- #define ROS_DOMAIN_ID 0
- #define LCD 0
- #define VL53 0
- #define MICROROS 1
- #define DEBUG_PRINTF 0
- #define DEBUG_MOTOR 0

config correcteur

- #define Te SAMPLING_PERIOD_ms
- #define LKp 0.001
- #define LKi (5.0/(0.1∗40.0))
- #define RKp 0.001
- #define RKi (5.0/(0.1∗40.0))

config default speed for each mode

- #define CMD 1000
- #define VITESSE_KART CMD/2
- #define VITESSE_OBS CMD
- #define VITESSE_CAM CMD/3

config camera settings

- #define CAMERA_X_MIN 0
- #define CAMERA_X_MAX 640
- #define CAMERA_Y_MIN 0
- #define CAMERA_Y_MAX 480

config default behaviour

- #define DEFAULT_MODE MODE_ZIG
- #define DEFAULT_SPEED LOW
- #define DEFAULT_DIR STOP

config test value

- #define NB 200
- #define TEST_CORRECTOR_DUTY 150
- #define TEST_CORRECTOR_SPEEDL -100
- #define TEST_CORRECTOR_SPEEDR -100
- #define TEST_LEFT_MOTOR 1

Enumerations

- enum { **MODE_OBS** , **MODE_ZIG** , **MODE_CAM** , **LAST_MODE** }
- enum {
  STOP_VIT , **LOW** , **FAST** , **SONIC** ,
  LAST_SPEED }
- enum {
  AVANT , **GAUCHE** , **RECULE** , **DROITE** ,
  STOP , **AVANT_GAUCHE** , **AVANT_DROITE** , **RECULE_GAUCHE** ,
  RECULE_DROITE , **LAST_DIR** }

Functions

- void SystemClock_Config (void)
- bool **cubemx_transport_open** (struct uxrCustomTransport ∗transport)
- bool **cubemx_transport_close** (struct uxrCustomTransport ∗transport)
- size_t **cubemx_transport_write** (struct uxrCustomTransport ∗transport, const uint8_t ∗buf, size_t len, uint8_t ∗err)
- size_t **cubemx_transport_read** (struct uxrCustomTransport ∗transport, uint8_t ∗buf, size_t len, int timeout, uint8_t ∗err)
- void ∗ microros_allocate (size_t size, void ∗state)
- void microros_deallocate (void ∗pointer, void ∗state)
- void ∗ microros_reallocate (void ∗pointer, size_t size, void ∗state)
- void ∗ microros_zero_allocate (size_t number_of_elements, size_t size_of_element, void ∗state)
- void CHECKMRRET (rcl_ret_t ret, char ∗msg)
- void SubscriberCallbackFunction (const void ∗msgin)
- void microros_task (void ∗argument)
- void task_Motor_Left (void ∗pvParameters)
- void task_Motor_Right (void ∗pvParameters)
- void task_Supervision (void ∗pvParameters)
- int main (void)
- void test_uart2 (void ∗pvParameters)
- void test_vl53 (void ∗pvParameters)
- void test_motor (void ∗pvParameters)
- void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef ∗htim)

  Period elapsed callback in non blocking mode.
- void Error_Handler (void)

Variables

- UART_HandleTypeDef huart1
- UART_HandleTypeDef huart2
- DMA_HandleTypeDef hdma_usart1_rx
- DMA_HandleTypeDef hdma_usart1_tx
- DMA_HandleTypeDef hdma_usart2_rx
- DMA_HandleTypeDef hdma_usart2_tx
- I2C_HandleTypeDef hi2c1
- osThreadId_t defaultTaskHandle
- const osThreadAttr_t defaultTask_attributes
- int16_t tab_speed [NB]

semaphore

- xSemaphoreHandle xSem_Supervision = NULL

queueHandle

- xQueueHandle q_mot_L = NULL
- xQueueHandle q_mot_R = NULL
- xQueueHandle qhMR_sub = NULL
- xQueueHandle qhMR_pub = NULL
- xQueueHandle qhLCD = NULL
- xQueueHandle qhVl53 = NULL

5.28.1 Detailed Description

file that contain the main code

Definition in file main.c.

5.28.2 Macro Definition Documentation

5.28.2.1 CAMERA_X_MAX

```
#define CAMERA_X_MAX 640
```

Define maximal x position return by camera

Definition at line 72 of file main.c.

5.28.2.2 CAMERA_X_MIN

```
#define CAMERA_X_MIN 0
```

Define minimal x position return by camera

Definition at line 71 of file main.c.

5.28.2.3 CAMERA_Y_MAX

```
#define CAMERA_Y_MAX 480
```

Define maximal y position return by camera

Definition at line 74 of file main.c.

5.28.2.4 CAMERA_Y_MIN

```
#define CAMERA_Y_MIN 0
```

Define minimal y position return by camera

Definition at line 73 of file main.c.

5.28.2.5 CMD

```
#define CMD 1000
```

Can be use as default speed

Definition at line 65 of file main.c.

5.28.2.6 DEBUG_MOTOR

```
#define DEBUG_MOTOR 0
```

Activate motor debug print

Definition at line 55 of file main.c.

5.28.2.7 DEBUG_PRINTF

```
#define DEBUG_PRINTF 0
```

Activate debug print

Definition at line 54 of file main.c.

5.28.2.8 DEFAULT_DIR

```
#define DEFAULT_DIR STOP
```

Default direction at startup

Definition at line 81 of file main.c.

5.28.2.9 DEFAULT_MODE

```
#define DEFAULT_MODE MODE_ZIG
```

Default mode at startup

Definition at line 79 of file main.c.

5.28.2.10 DEFAULT_SPEED

```
#define DEFAULT_SPEED LOW
```

Default speed at startup

Definition at line 80 of file main.c.

5.28.2.11 EXCORRECTOR

```
#define EXCORRECTOR 2
```

Code to calibrate your correcteur

Definition at line 41 of file main.c.

5.28.2.12 EXFINAL

```
#define EXFINAL 6
```

Final code

Definition at line 45 of file main.c.

5.28.2.13 EXSTARTUP

```
#define EXSTARTUP 0
```

startup code

Definition at line 39 of file main.c.

5.28.2.14 EXTEST_MICROROS

```
#define EXTEST_MICROROS 5
```

Test Micro ROS subscriber and publisher

Definition at line 44 of file main.c.

5.28.2.15 EXTEST_UART2

```
#define EXTEST_UART2 1
```

Test printf and scanf function

Definition at line 40 of file main.c.

5.28.2.16 EXTEST_VL53

```
#define EXTEST_VL53 4
```

Test VL530X sensor

Definition at line 43 of file main.c.

5.28.2.17 EXTESTCORRECTOR

```
#define EXTESTCORRECTOR 3
```

Code to test your correcteur

Definition at line 42 of file main.c.

5.28.2.18 LCD

```
#define LCD 0
```

Activate LCD task

Definition at line 51 of file main.c.

5.28.2.19 LKi

```
#define LKi (5.0/(0.1*40.0))
```

Ki factor for the left motor

Definition at line 60 of file main.c.

5.28.2.20 LKp

```
#define LKp 0.001
```

Kp factor for the left motor

Definition at line 59 of file main.c.

5.28.2.21 MICROROS

```
#define MICROROS 1
```

Activate MicroROS task

Definition at line 53 of file main.c.

5.28.2.22 NB

```
#define NB 200
```

Number of samples in correcteur calibration task

Definition at line 84 of file main.c.

5.28.2.23 RKi

```
#define RKi (5.0/(0.1*40.0))
```

Kp factor for the right motor

Definition at line 62 of file main.c.

5.28.2.24 RKp

```
#define RKp 0.001
```

Kp factor for the right motor

Definition at line 61 of file main.c.

5.28.2.25 ROS_DOMAIN_ID

```
#define ROS_DOMAIN_ID 0
```

Define ROS domain id

Definition at line 50 of file main.c.

5.28.2.26 SAMPLING_PERIOD_ms

```
#define SAMPLING_PERIOD_ms 5
```

Define the delay beetween two execution of the same task

Definition at line 37 of file main.c.

5.28.2.27 SYNCHRO_EX

```
#define SYNCHRO_EX EXFINAL
```

Define wich config are executed

Definition at line 47 of file main.c.

5.28.2.28 Te

```
#define Te SAMPLING_PERIOD_ms
```

Definition at line 58 of file main.c.

5.28.2.29 TEST_CORRECTOR_DUTY

```
#define TEST_CORRECTOR_DUTY 150
```

Duty cycle to apply to calibrate the correcteur

Definition at line 85 of file main.c.

5.28.2.30 TEST_CORRECTOR_SPEEDL

```
#define TEST_CORRECTOR_SPEEDL -100
```

Speed to test the left correcteur

Definition at line 86 of file main.c.

5.28.2.31 TEST_CORRECTOR_SPEEDR

```
#define TEST_CORRECTOR_SPEEDR -100
```

Speed to test the right correcteur

Definition at line 87 of file main.c.

5.28.2.32 TEST_LEFT_MOTOR

```
#define TEST_LEFT_MOTOR 1
```

Calibrate left motor correcteur or not

Definition at line 88 of file main.c.

5.28.2.33 VITESSE_CAM

```
#define VITESSE_CAM CMD/3
```

Default speed for camera mode

Definition at line 68 of file main.c.

5.28.2.34 VITESSE_KART

```
#define VITESSE_KART CMD/2
```

Default speed for manual mode

Definition at line 66 of file main.c.

5.28.2.35 VITESSE_OBS

```
#define VITESSE_OBS CMD
```

Default speed for obstacle mode

Definition at line 67 of file main.c.

5.28.2.36 VL53

```
#define VL53 0
```

Activate VL530X task

Definition at line 52 of file main.c.

5.28.3 Enumeration Type Documentation

5.28.3.1 anonymous enum

```
anonymous enum
```

enumerate mode of robot

Definition at line 31 of file main.c.

5.28.3.2 anonymous enum

```
anonymous enum
```

enumerate speed

Definition at line 33 of file main.c.

5.28.3.3 anonymous enum

```
anonymous enum
```

enumerate direction

Definition at line 35 of file main.c.

5.28.4 Function Documentation

5.28.4.1 CHECKMRRET()

```
void CHECKMRRET (
            rcl_ret_t ret,
            char * msg )
```

check if microRos function success else print msg in console

Parameters

ret	return value of microRos function
msg	message to print if fail

Definition at line 153 of file main.c.

5.28.4.2 Error_Handler()

```
void Error_Handler (
             void  )
```

Definition at line 941 of file main.c.

5.28.4.3 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
             TIM_HandleTypeDef * htim )
```

Period elapsed callback in non blocking mode.

Note

This function is called when TIM1 interrupt took place, inside HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment a global variable "uwTick" used as application time base.

Parameters

htim	: TIM handle

Return values

None	

Definition at line 933 of file main.c.

5.28.4.4 main()

```
int main (
             void  )
```

Init all GPIO and drivers, start the task, init semaphore and queue and launch the kernel

- Config EXSTARTUP
  - Launch microRos, supervision, left motor, right motor and lcd task
- Config EXTEST_UART2

  - Launch test_uart2 task
- Config EXCORRECTOR
  - Launch test_motor task
- Config EXTESTCORRECTOR
  - Launch supervision, left motor and right motor task
- Config EXTEST_VL53
  - Launch test_vl53 task
- Config EXTEST_MICROROS
  - Launch microRos task
- Config EXFINAL
  - Launch microRos, supervision, left motor, right motor, vl53 and lcd task

Definition at line 775 of file main.c.

5.28.4.5 microros_allocate()

```
void * microros_allocate (
            size_t size,
            void * state )
```

Definition at line 14 of file microros_allocators.c.

5.28.4.6 microros_deallocate()

```
void microros_deallocate (
            void * pointer,
            void * state )
```

Definition at line 22 of file microros_allocators.c.

5.28.4.7 microros_reallocate()

```
void * microros_reallocate (
            void * pointer,
            size_t size,
            void * state )
```

Definition at line 31 of file microros_allocators.c.

5.28.4.8 microros_task()

```
void microros_task (
            void * argument )
```

- All config
  - Create the node *STM32_node*
  - Set the Domain id of microRos
- Config EXSTARTUP :
  - Create a publisher and send a message on it
- Config EXTEST_MICROROS :
  - Create a publisher, a subscriber and an executor
  - Init the executor and add the subscriber to it
  - Run the executor and send the receive message on the publisher
- Config EXFINAL :
  - Create 3 publishers, 5 subscriber and an executor
  - Init the executor and add the 5 subscribers to it
  - run the executor and if they are no elements waiting to be read by the task decision put the receive information in the queue If decison task send data then publish data to microRos

Parameters

argument	

Definition at line 166 of file main.c.

5.28.4.9 microros_zero_allocate()

```
void * microros_zero_allocate (
            size_t number_of_elements,
            size_t size_of_element,
            void * state )
```

Definition at line 44 of file microros_allocators.c.

5.28.4.10 SubscriberCallbackFunction()

```
void SubscriberCallbackFunction (
            const void * msgin )
```

callback call by microros when a message is receive here use as debug and just print the receive msg

Parameters

message	receive

Definition at line 155 of file main.c.

5.28.4.11 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

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

Definition at line 11 of file systemclock.c.

5.28.4.12 task_Motor_Left()

```
void task_Motor_Left (
            void * pvParameters )
```

Task use to control the left motor of the robot

Parameters

argument	

Definition at line 365 of file main.c.

5.28.4.13 task_Motor_Right()

```
void task_Motor_Right (
            void * pvParameters )
```

Task use to control the right motor of the robot

Parameters

argument	

Definition at line 391 of file main.c.

5.28.4.14 task_Supervision()

```
void task_Supervision (
            void * pvParameters )
```

Brain of the robot. get information for MicroRos and VL53 task, then send speed to left and right motor, lcd and microRos task

- Config EXSTARTUP :
  - Make the robot drive forward until an obstacle are found
- Config EXTESTCORRECTOR :
  - Make the robot drive forward at speed set by config
- Config EXFINAL :
  - Make robot switch beetween 3 behaviour depending of the mode
  - Obstacle : drive and avoid obstacles
  - Manual : drive in direction set in ihm
  - Camera : follow an object

Parameters

argument	

Definition at line 476 of file main.c.

5.28.4.15 test_motor()

```
void test_motor (
            void * pvParameters )
```

Use to set the duty cycle and register the motor speed at each Te

Definition at line 895 of file main.c.

5.28.4.16 test_uart2()

```
void test_uart2 (
            void * pvParameters )
```

Use to test printf and scanf function

Definition at line 872 of file main.c.

5.28.4.17 test_vl53()

```
void test_vl53 (
            void * pvParameters )
```

Use to test the VL53 sensor

Definition at line 884 of file main.c.

5.28.5 Variable Documentation

5.28.5.1 defaultTask_attributes

```
const osThreadAttr_t defaultTask_attributes
```

Initial value:

```
= {
  .name = "defaultTask",
  .stack_size = 3000 * 4,
  .priority = (osPriority_t) osPriorityNormal,
}
```

Definition at line 24 of file main.c.

5.28.5.2 defaultTaskHandle

```
osThreadId_t defaultTaskHandle
```

Definition at line 23 of file main.c.

5.28.5.3 hdma_usart1_rx

```
DMA_HandleTypeDef hdma_usart1_rx  [extern]
```

Definition at line 6 of file drv_uart.c.

5.28.5.4 hdma_usart1_tx

```
DMA_HandleTypeDef hdma_usart1_tx  [extern]
```

Definition at line 7 of file drv_uart.c.

5.28.5.5 hdma_usart2_rx

```
DMA_HandleTypeDef hdma_usart2_rx  [extern]
```

Definition at line 8 of file drv_uart.c.

5.28.5.6 hdma_usart2_tx

```
DMA_HandleTypeDef hdma_usart2_tx  [extern]
```

Definition at line 9 of file drv_uart.c.

5.28.5.7 hi2c1

```
I2C_HandleTypeDef hi2c1  [extern]
```

Definition at line 6 of file drv_i2c.c.

5.28.5.8 huart1

```
UART_HandleTypeDef huart1  [extern]
```

Definition at line 4 of file drv_uart.c.

5.28.5.9 huart2

```
UART_HandleTypeDef huart2  [extern]
```

Definition at line 5 of file drv_uart.c.

5.28.5.10 q_mot_L

```
xQueueHandle q_mot_L = NULL
```

Queue to communicate with left motor task

Definition at line 96 of file main.c.

5.28.5.11 q_mot_R

```
xQueueHandle q_mot_R = NULL
```

Queue to communicate with right motor task

Definition at line 97 of file main.c.

5.28.5.12 qhLCD

```
xQueueHandle qhLCD = NULL
```

Queue to communicate with LCD task

Definition at line 100 of file main.c.

5.28.5.13 qhMR_pub

```
xQueueHandle qhMR_pub = NULL
```

Queue to communicate with microRos task

Definition at line 99 of file main.c.

5.28.5.14 qhMR_sub

```
xQueueHandle qhMR_sub = NULL
```

Queue to get information from microRos task

Definition at line 98 of file main.c.

5.28.5.15 qhVl53

```
xQueueHandle qhVl53 = NULL
```

Queue to communicate with VL53 task

Definition at line 101 of file main.c.

5.28.5.16 tab_speed

```
int16_t tab_speed[NB]
```

use to store speed of motor during calibration of the correcteur

Definition at line 104 of file main.c.

5.28.5.17 xSem_Supervision

```
xSemaphoreHandle xSem_Supervision = NULL
```

Semaphore use in decision task

Definition at line 93 of file main.c.

5.29 main.c

Go to the documentation of this file.

```
00001
00005 #include "main.h"
00006
00007 #include "motorCommand.h"
00008 #include "quadEncoder.h"
00009 #include "captDistIR.h"
00010 #include "VL53L0X.h"
00011 #include "groveLCD.h"
00012
00013 extern UART_HandleTypeDef huart1;
00014 extern UART_HandleTypeDef huart2;
00015 extern DMA_HandleTypeDef hdma_usart1_rx;
00016 extern DMA_HandleTypeDef hdma_usart1_tx;
00017 extern DMA_HandleTypeDef hdma_usart2_rx;
00018 extern DMA_HandleTypeDef hdma_usart2_tx;
00019
00020 extern I2C_HandleTypeDef hi2c1;
00021
00022 /* Definitions for defaultTask */
00023 osThreadId_t defaultTaskHandle;
00024 const osThreadAttr_t defaultTask_attributes = {
00025   .name = "defaultTask",
00026   .stack_size = 3000 * 4,
```

```
00027   .priority = (osPriority_t) osPriorityNormal,
00028 };
00029
00031 enum {MODE_OBS, MODE_ZIG, MODE_CAM, LAST_MODE};
00033 enum {STOP_VIT, LOW, FAST, SONIC, LAST_SPEED};
00035 enum {AVANT, GAUCHE, RECULE, DROITE, STOP, AVANT_GAUCHE, AVANT_DROITE, RECULE_GAUCHE, RECULE_DROITE,
      LAST_DIR};
00036
00037 #define SAMPLING_PERIOD_ms 5
00039 #define EXSTARTUP 0
00040 #define EXTEST_UART2 1
00041 #define EXCORRECTOR 2
00042 #define EXTESTCORRECTOR 3
00043 #define EXTEST_VL53 4
00044 #define EXTEST_MICROROS 5
00045 #define EXFINAL 6
00047 #define SYNCHRO_EX EXFINAL
00050 #define ROS_DOMAIN_ID 0
00051 #define LCD 0
00052 #define VL53 0
00053 #define MICROROS 1
00054 #define DEBUG_PRINTF 0
00055 #define DEBUG_MOTOR 0
00058 #define Te SAMPLING_PERIOD_ms
00059 #define LKp 0.001
00060 #define LKi (5.0/(0.1*40.0))
00061 #define RKp 0.001
00062 #define RKi (5.0/(0.1*40.0))
00065 #define CMD 1000
00066 #define VITESSE_KART CMD/2
00067 #define VITESSE_OBS CMD
00068 #define VITESSE_CAM CMD/3
00071 #define CAMERA_X_MIN 0
00072 #define CAMERA_X_MAX 640
00073 #define CAMERA_Y_MIN 0
00074 #define CAMERA_Y_MAX 480
00075 //#define CAMERA_X_TIER (CAMERA_X_MAX-CAMERA_X_MIN)/3 /**< */
00076 //#define CAMERA_Y_TIER (CAMERA_Y_MAX-CAMERA_Y_MIN)/3 /**< */
00079 #define DEFAULT_MODE MODE_ZIG
00080 #define DEFAULT_SPEED LOW
00081 #define DEFAULT_DIR STOP
00084 #define NB 200
00085 #define TEST_CORRECTOR_DUTY 150
00086 #define TEST_CORRECTOR_SPEEDL -100
00087 #define TEST_CORRECTOR_SPEEDR -100
00088 #define TEST_LEFT_MOTOR 1
00091 // Déclaration des objets synchronisants !! Ne pas oublier de les créer
00093 xSemaphoreHandle xSem_Supervision = NULL;
00096 xQueueHandle q_mot_L = NULL;
00097 xQueueHandle q_mot_R = NULL;
00098 xQueueHandle qhMR_sub = NULL;
00099 xQueueHandle qhMR_pub = NULL;
00100 xQueueHandle qhLCD = NULL;
00101 xQueueHandle qhVl53 = NULL;
00104 int16_t tab_speed[NB];
00110 typedef struct
00111 {
00112     char command;
00113     int data;
00114 } AMessage;
00115
00119 typedef struct
00120 {
00121     char dir;
00122     int mode;
00123     int speed;
00124 } MicroRosPubMsg;
00125
00129 typedef struct
00130 {
00131     int dir;
00132     int x;
00133     int y;
00134     int mode;
00135     int speed;
00136 } MicroRosSubMsg;
00139 //Robot function
00140 void SystemClock_Config(void);
00141
00142 //Micro-Ros function
00143 bool cubemx_transport_open(struct uxrCustomTransport * transport);
00144 bool cubemx_transport_close(struct uxrCustomTransport * transport);
00145 size_t cubemx_transport_write(struct uxrCustomTransport* transport, const uint8_t * buf, size_t len,
      uint8_t * err);
00146 size_t cubemx_transport_read(struct uxrCustomTransport* transport, uint8_t* buf, size_t len, int
      timeout, uint8_t* err);
00147
```

```
00148 void * microros_allocate(size_t size, void * state);
00149 void microros_deallocate(void * pointer, void * state);
00150 void * microros_reallocate(void * pointer, size_t size, void * state);
00151 void * microros_zero_allocate(size_t number_of_elements, size_t size_of_element, void * state);
00152
00153 void CHECKMRRET(rcl_ret_t ret, char* msg){if (ret != RCL_RET_OK){ printf("Error : %d\r\nMsg : %s\r\n",
      (int)ret, msg); }}
00154
00155 void SubscriberCallbackFunction(const void *msgin){
00156 #if SYNCHRO_EX == EXTEST_MICROROS
00157     std_msgs__msg__String * msg = (std_msgs__msg__String * )msgin;
00158     printf("\r\nMessage recue : %s\r\n", msg->data->data);
00159 #elif SYNCHRO_EX == EXFINAL
00160     std_msgs__msg__Int32 * msg = (std_msgs__msg__Int32 * )msgin;
00161     printf("\r\nMessage recue : %ld\r\n", msg->data);
00162 #endif //SYNCHRO_EX
00163 }
00164
00165 // https://github.com/lFatality/stm32_micro_ros_setup
00166 void microros_task(void *argument)
00167 {
00168     // micro-ROS app variable
00169     rclc_support_t support; //Contain information about
00170     rcl_allocator_t allocator;
00171     rcl_node_t node; //microRos structure wich represent a node ROS
00172     rcl_node_options_t node_opt; //microRos structure wich represent option of a node ROS
00173     rclc_executor_t executor; //The executor is use to receive message
00174
00175     // micro-ROS configuration
00176     rmw_uros_set_custom_transport(
00177         true,
00178         (void *) &huart1,
00179         cubemx_transport_open,
00180         cubemx_transport_close,
00181         cubemx_transport_write,
00182         cubemx_transport_read);
00183
00184     rcl_allocator_t freeRTOS_allocator = rcutils_get_zero_initialized_allocator();
00185     freeRTOS_allocator.allocate = microros_allocate;
00186     freeRTOS_allocator.deallocate = microros_deallocate;
00187     freeRTOS_allocator.reallocate = microros_reallocate;
00188     freeRTOS_allocator.zero_allocate =  microros_zero_allocate;
00189
00190     if (!rcutils_set_default_allocator(&freeRTOS_allocator)) {
00191         printf("Error on default allocators (line %d)\r\n", __LINE__);
00192     }
00193
00194     allocator = rcl_get_default_allocator();
00195     //create init_options
00196     CHECKMRRET(rclc_support_init(&support, 0, NULL, &allocator), "error on init support");
00197     // create node
00198     //CHECKMRRET(rclc_node_init_default(&node, "STM32_node", "", &support), "error on init node");
00199     node_opt = rcl_node_get_default_options();
00200     node_opt.domain_id = ROS_DOMAIN_ID;
00201     CHECKMRRET(rclc_node_init_with_options(&node, "STM32_node", "", &support, &node_opt), "error on
      init node");
00202
00203
00204 #if SYNCHRO_EX == EXSTARTUP
00205     static int counter = 0;
00206     rcl_ret_t ret;
00207     rcl_publisher_t publisher;
00208     std_msgs__msg__String msg;
00209
00210     CHECKMRRET(rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs,
      msg, String), "cubemx_publisher"),
00211             "Error when create publisher");
00212
00213     for (;;)
00214     {
00215         sprintf(msg.data.data, "Hello from micro-ROS #%d", counter++);
00216         msg.data.size = strlen(msg.data.data);
00217         ret = rcl_publish(&publisher, &msg, NULL);
00218         if (ret != RCL_RET_OK)
00219             printf("Error publishing (line %d)\r\n", __LINE__);
00220         vTaskDelay(SAMPLING_PERIOD_ms);
00221     }
00222 #elif SYNCHRO_EX == EXTEST_MICROROS
00223     //micro-ros topic variable
00224     rcl_ret_t ret;
00225     rcl_publisher_t publisher;
00226     rcl_subscription_t subscriber;
00227     std_msgs__msg__String msg;
00228
00229     /* Default publisher */
00230     CHECKMRRET(rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs,
      msg, String), "cubemx_publisher"),
```

```
00231             "Error when create publisher");
00232         /* ---------------------- */
00233     /* Default subscriber */
00234     subscriber = rcl_get_zero_initialized_subscription();
00235     CHECKMRRET(rclc_subscription_init_default(&subscriber, &node,
      ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, String),"cubemx_subscriber"),
00236             "Error when create subscriber");
00237     /* ---------------------- */
00238     /* Init string msg */
00239     msg.data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
00240     msg.data.size = 0;
00241     msg.data.capacity = ARRAY_LEN;
00242     /* ---------------- */
00243     // Init executor and add subscriber to it
00244     CHECKMRRET(rclc_executor_init(&executor, &support.context, 1, &allocator), "Error on init
      executor");
00245     CHECKMRRET(rclc_executor_add_subscription(&executor, &subscriber, &msg,
      &SubscriberCallbackFunction, ON_NEW_DATA), "error add subscriber");
00246
00247     for (;;)
00248     {
00249         ret = rclc_executor_spin_some(&executor, 100*1000*1000);
00250         vTaskDelay(SAMPLING_PERIOD_ms);
00251     }
00252 #elif SYNCHRO_EX == EXFINAL
00253     //Init the queue mesage
00254     MicroRosPubMsg MsgToPub = {'N', 0, 0};
00255     MicroRosSubMsg SubToMsg = {DEFAULT_DIR, 0, 0, DEFAULT_MODE, DEFAULT_SPEED};
00256     /* PUBLISHER */
00257     //Use to publish the direction of robot in sensor mode
00258     rcl_publisher_t capteur_dir_pub;
00259     char* capteur_dir_topic = CAPTEUR_DIR_TOPIC;
00260     std_msgs__msg__Int32 capteur_dir_msg;
00261     //Use to publish the actual mode of the robot
00262     rcl_publisher_t etat_mode_pub;
00263     char* etat_mode_topic = ETAT_MODE_TOPIC;
00264     std_msgs__msg__Int32 etat_mode_msg;
00265     //Use to publish the actual speed of the robot
00266     rcl_publisher_t etat_speed_pub;
00267     char* etat_speed_topic = ETAT_SPEED_TOPIC;
00268     std_msgs__msg__Int32 etat_speed_msg;
00269     /* SUBSCRIBER */
00270     //Use to receive the x position of object see by the camera
00271     rcl_subscription_t camera_x_sub;
00272     char* camera_x_topic = CAMERA_X_TOPIC;
00273     std_msgs__msg__Int32 camera_x_msg;
00274     //Use to receive the y position of object see by the camera
00275     rcl_subscription_t camera_y_sub;
00276     char* camera_y_topic = CAMERA_Y_TOPIC;
00277     std_msgs__msg__Int32 camera_y_msg;
00278     //Use to receive the remote control in remote mode
00279     rcl_subscription_t telecommande_dir_sub;
00280     char* telecommande_dir_topic = TELECOMMANDE_DIR_TOPIC;
00281     std_msgs__msg__Int32 telecommande_dir_msg;
00282     //Use to receive the mode config
00283     rcl_subscription_t config_mode_sub;
00284     char* config_mode_topic = CONFIG_MODE_TOPIC;
00285     std_msgs__msg__Int32 config_mode_msg;
00286     //Use to receive the speed config
00287     rcl_subscription_t config_speed_sub;
00288     char* config_speed_topic = CONFIG_SPEED_TOPIC;
00289     std_msgs__msg__Int32 config_speed_msg;
00290
00291     // create publisher
00292     createPublisher(&capteur_dir_pub, &node,
00293         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00294         capteur_dir_topic, &capteur_dir_msg);
00295
00296     createPublisher(&etat_mode_pub, &node,
00297         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00298         etat_mode_topic, &etat_mode_msg);
00299
00300     createPublisher(&etat_speed_pub, &node,
00301         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00302         etat_speed_topic, &etat_speed_msg);
00303
00304     //create subscriber
00305     createSubscriber(&camera_x_sub, &node,
00306         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00307         camera_x_topic, &camera_x_msg);
00308
00309     createSubscriber(&camera_y_sub, &node,
00310             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00311             camera_y_topic, &camera_y_msg);
00312
00313     createSubscriber(&telecommande_dir_sub, &node,
00314             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
```

```
00315             telecommande_dir_topic, &telecommande_dir_msg);
00316
00317     createSubscriber(&config_mode_sub, &node,
00318             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00319             config_mode_topic, &config_mode_msg);
00320
00321     createSubscriber(&config_speed_sub, &node,
00322             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00323             config_speed_topic, &config_speed_msg);
00324
00325     //Init the executor
00326     CHECKMRRET(rclc_executor_init(&executor, &support.context, 5, &allocator), "Error on init
    executor");
00327     /*Add subscriber to executor to let it check if message is receive on this
00328     topic and store the data on the message structure after call the callback*/
00329     CHECKMRRET(rclc_executor_add_subscription(&executor, &camera_x_sub, &camera_x_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add camera_x_sub");
00330     CHECKMRRET(rclc_executor_add_subscription(&executor, &camera_y_sub, &camera_y_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add camera_y_sub");
00331     CHECKMRRET(rclc_executor_add_subscription(&executor, &telecommande_dir_sub, &telecommande_dir_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add telecommande_dir_sub");
00332     CHECKMRRET(rclc_executor_add_subscription(&executor, &config_mode_sub, &config_mode_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add config_mode_sub");
00333     CHECKMRRET(rclc_executor_add_subscription(&executor, &config_speed_sub, &config_speed_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add config_speed_sub");
00334
00335     for(;;)
00336     {
00337         if (!uxQueueMessagesWaiting(qhMR_sub)) //If no message in 'output' queue
00338             xQueueSend(qhMR_sub, ( void * ) &SubToMsg, portMAX_DELAY);
00339         rclc_executor_spin_some(&executor, 1*1000*1000); //Execute executor
00340         SubToMsg.dir = telecommande_dir_msg.data;
00341         SubToMsg.x = camera_x_msg.data;
00342         SubToMsg.y = camera_y_msg.data;
00343         SubToMsg.mode = config_mode_msg.data;
00344         SubToMsg.speed = config_speed_msg.data;
00345
00346         if (uxQueueMessagesWaiting(qhMR_pub)) //If no message in 'input' queue
00347         {
00348             xQueueReceive(qhMR_pub, &MsgToPub, portMAX_DELAY); //Receive data
00349             capteur_dir_msg.data = (int)MsgToPub.dir;
00350             etat_mode_msg.data = MsgToPub.mode;
00351             etat_speed_msg.data = MsgToPub.speed;
00352             //Publish data
00353             CHECKMRRET(rcl_publish(&capteur_dir_pub, &capteur_dir_msg, NULL), "erreur publish
    capteur_dir_pub");
00354             CHECKMRRET(rcl_publish(&etat_mode_pub, &etat_mode_msg, NULL), "erreur publish
    etat_mode_pub");
00355             CHECKMRRET(rcl_publish(&etat_speed_pub, &etat_speed_msg, NULL), "erreur publish
    etat_speed_pub");
00356             #if DEBUG_PRINTF
00357             printf("\r\nReceive from decision :\r\nDirection : %d\r\nMode : %d\r\nSpeed : %d\r\n",
    capteur_dir_msg.data, etat_mode_msg.data, etat_speed_msg.data);
00358             #endif //DEBUG_PRINTF
00359         }
00360         vTaskDelay(SAMPLING_PERIOD_ms);
00361     }
00362 #endif //SYNCHRO_EX
00363 }
00364
00365 void task_Motor_Left(void *pvParameters)
00366 {
00367     int16_t consigne = 0; //Store the desirate speed
00368
00369     float ui = 0.0; //Integral term of the correcteur
00370     float up = 0.0; //Proportionnal term of the correcteur
00371     int err = 0; //Error term of the correcteur
00372     int speed = 0; //Actual speed of motor
00373
00374     for (;;)
00375     {
00376         xQueueReceive(q_mot_L, &consigne, portMAX_DELAY); //receive wanted speed
00377
00378         speed = quadEncoder_GetSpeedL(); //Get actual speed
00379         //Calculate term of correcteur
00380         err=consigne-speed;
00381         up=LKp*(float)err;
00382         ui=ui+LKp*LKi*(float)err;
00383
00384         motorLeft_SetDuty(100+(int)(up+ui)); //Set duty cycle of the motor
00385
00386         xSemaphoreGive(xSem_Supervision); //Give semaphore to liberate the decision task
00387         vTaskDelay(SAMPLING_PERIOD_ms);
00388     }
00389 }
00390
00391 void task_Motor_Right(void *pvParameters)
```

```
00392 {
00393     int16_t consigne = 0; //Store the desirate speed
00394 
00395     float ui = 0.0; //Integral term of the correcteur
00396     float up = 0.0; //Proportionnal term of the correcteur
00397     int err = 0; //Error term of the correcteur
00398     int speed = 0; //Actual speed of motor
00399 
00400     for (;;)
00401     {
00402         xQueueReceive(q_mot_R, &consigne, portMAX_DELAY); //receive wanted speed
00403 
00404         speed = quadEncoder_GetSpeedR(); //Get actual speed
00405         //Calculate term of correcteur
00406         err=consigne-speed;
00407         up=RKp*(float)err;
00408         ui=ui+RKp*RKi*(float)err;
00409 
00410         motorRight_SetDuty(100+(int)(up+ui)); //Set duty cycle of the motor
00411 
00412         xSemaphoreGive(xSem_Supervision);//Give semaphore to liberate the decision task
00413         vTaskDelay(SAMPLING_PERIOD_ms);
00414     }
00415 }
00416 
00417 #if VL53
00418 void task_VL53(void *pvParameters)
00419 {
00420     static uint16_t dist;
00421     static const int SEUIL = 20;
00422     int obs = 0;
00423 
00424     for(;;)
00425     {
00426         dist = readRangeSingleMillimeters()/10;
00427         if (dist < SEUIL && dist != 0)
00428             obs = 1;
00429         else
00430             obs = 0;
00431 
00432         if (!uxQueueMessagesWaiting(qhVl53))
00433             xQueueSend(qhVl53, (void *)&obs, portMAX_DELAY);
00434 
00435         vTaskDelay(SAMPLING_PERIOD_ms);
00436     }
00437 }
00438 #endif //VL53
00439 
00440 
00441 #if LCD
00442 void task_Grove_LCD(void *pvParameters)
00443 {
00444 #if SYNCHRO_EX == EXSTARTUP
00445     for (;;)
00446     {
00447          groveLCD_setCursor(0,0);
00448          groveLCD_term_printf("TEST LCD");
00449          vTaskDelay(100);
00450     }
00451 #elif SYNCHRO_EX == EXFINAL
00452     AMessage pxRxedMessage;
00453 
00454     for(;;)
00455     {
00456         if (uxQueueMessagesWaiting(qhLCD))
00457         {
00458             xQueueReceive(qhLCD, &pxRxedMessage, portMAX_DELAY);
00459             int mode = pxRxedMessage.data;
00460             char direction=pxRxedMessage.command;
00461             groveLCD_setCursor(0,0);
00462             if (mode == MODE_OBS)
00463                 groveLCD_term_printf("M:Obstacle  D:%c", direction);
00464             else if (mode == MODE_ZIG)
00465                 groveLCD_term_printf("M:Manuel        ");
00466             else if (mode == MODE_CAM)
00467                 groveLCD_term_printf("M:Camera        ");
00468         }
00469 
00470         vTaskDelay(SAMPLING_PERIOD_ms);
00471     }
00472 #endif //SYNCHRO_EX
00473 }
00474 #endif //LCD
00475 
00476 void task_Supervision(void *pvParameters)
00477 {
00478 #if SYNCHRO_EX == EXSTARTUP
```

```
00479     int16_t consigne_G=0;
00480     int16_t consigne_D=0;
00481
00482     int tab_mes_ir[2];
00483     uint16_t mes_vl53=0;
00484
00485     vTaskDelay(100);
00486     for (;;)
00487     {
00488         captDistIR_Get(tab_mes_ir);
00489         //mes_vl53 = readRangeSingleMillimeters()/10;
00490
00491         if((tab_mes_ir[0]>2000)||(tab_mes_ir[1]>2000))
00492         { // !! obstacle
00493             consigne_G=0;
00494             consigne_D=0;
00495         }
00496         else
00497         {
00498             consigne_G=1000;
00499             consigne_D=1000;
00500         }
00501
00502         xQueueSend( q_mot_L, ( void * ) &consigne_G,  portMAX_DELAY );
00503         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00504
00505         xQueueSend( q_mot_R, ( void * ) &consigne_D,  portMAX_DELAY );
00506         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00507
00508         vTaskDelay(SAMPLING_PERIOD_ms);
00509     }
00510 #elif SYNCHRO_EX == EXTESTCORRECTOR
00511     int16_t speedLeft = TEST_CORRECTOR_SPEEDL;
00512     int16_t speedRight = TEST_CORRECTOR_SPEEDR;
00513
00514     for (;;)
00515     {
00516         xQueueSend(q_mot_L, (void *)&speedLeft, portMAX_DELAY);
00517         xSemaphoreTake(xSem_Supervision, portMAX_DELAY);
00518
00519         xQueueSend(q_mot_R, (void *)&speedRight, portMAX_DELAY);
00520         xSemaphoreTake(xSem_Supervision, portMAX_DELAY);
00521
00522         vTaskDelay(SAMPLING_PERIOD_ms);
00523     }
00524 #elif SYNCHRO_EX == EXFINAL
00525     int16_t speedLeft;
00526     int16_t speedRight;
00527
00528     int table[2];
00529     #if VL53
00530     int vl53 = 0;
00531     #endif //VL53
00532
00533     static int obs = 0;
00534     static char dir = 'f';
00535     static int direction = DEFAULT_DIR;
00536     static int speed = DEFAULT_SPEED;
00537     static int mode = DEFAULT_MODE;
00538     static int x = 0;
00539     static int y = 0;
00540
00541 #if LCD
00542      AMessage pxMessage;
00543 #endif
00544
00545 #if MICROROS
00546     MicroRosSubMsg SubToMsg;
00547     MicroRosPubMsg MsgToPub;
00548 #endif //MICROROS
00549
00550     for (;;)
00551     {
00552         #if MICROROS
00553         if (uxQueueMessagesWaiting(qhMR_sub))
00554         {
00555             xQueueReceive(qhMR_sub, &SubToMsg, portMAX_DELAY);
00556             if (SubToMsg.mode >= 0 && SubToMsg.mode < LAST_MODE)
00557                 mode = SubToMsg.mode;
00558             if (SubToMsg.dir >= 0 && SubToMsg.dir < LAST_DIR)
00559                 direction = SubToMsg.dir;
00560             if (SubToMsg.speed > 0 && SubToMsg.speed < LAST_SPEED)
00561                 speed = SubToMsg.speed;
00562             x = SubToMsg.x;
00563             y = SubToMsg.y;
00564             #if DEBUG_PRINTF
00565             printf("%cc%c[2J%c[0;0HVariable to make decision : \n\rDirection : %d\r\nMode: %d\r\nSpeed
```

```
      : %d\r\nX: %d\r\nY : %d\r\n", 0x1b, 0x1b, 0x1b, direction, mode, speed, x, y);
00566             #endif //DEBUG_PRINTF
00567         }
00568         #endif //MICROROS
00569
00570         if (mode == MODE_ZIG)
00571         {
00572             dir = 'N';
00573             obs = 0;
00574             switch(direction)
00575             {
00576                 case STOP:
00577                     speedLeft = 0;
00578                     speedRight = 0;
00579                     break;
00580                 case AVANT:
00581                     speedLeft = VITESSE_KART*speed;
00582                     speedRight = VITESSE_KART*speed;
00583                     break;
00584                 case RECULE:
00585                     speedLeft = -VITESSE_KART*speed;
00586                     speedRight = -VITESSE_KART*speed;
00587                     break;
00588                 case DROITE:
00589                     speedLeft = VITESSE_KART*speed;
00590                     speedRight = -VITESSE_KART*speed;
00591                     break;
00592                 case GAUCHE:
00593                     speedLeft = -VITESSE_KART*speed;
00594                     speedRight = VITESSE_KART*speed;
00595                     break;
00596                 case AVANT_GAUCHE:
00597                     speedLeft = (VITESSE_KART/2)*speed;
00598                     speedRight = VITESSE_KART*speed;
00599                     break;
00600                 case AVANT_DROITE:
00601                     speedLeft = VITESSE_KART*speed;
00602                     speedRight = (VITESSE_KART/2)*speed;
00603                     break;
00604                 case RECULE_GAUCHE:
00605                     speedLeft = -VITESSE_KART*speed;
00606                     speedRight = (-VITESSE_KART/2)*speed;
00607                     break;
00608                 case RECULE_DROITE:
00609                     speedLeft = (-VITESSE_KART/2)*speed;
00610                     speedRight = -VITESSE_KART*speed;
00611                     break;
00612                 default:
00613                     speedLeft = 0;
00614                     speedRight = 0;
00615                     break;
00616             }
00617         }
00618         else if (mode == MODE_OBS)
00619         {
00620             captDistIR_Get(table);
00621             #if VL53
00622             if (uxQueueMessagesWaiting(qhVl53))
00623                 xQueueReceive(qhVl53, &vl53, portMAX_DELAY);
00624             else
00625                 vl53 = 0;
00626
00627             if (vl53 == 1) //Il y a un obstacle
00628             {
00629                 speedLeft = 0;
00630                 speedRight = 0;
00631                 dir = 'S';
00632                 obs = 1;
00633             }
00634             else
00635             #endif //VL53
00636             if (table[0] > 1000 || table[1] > 1000)
00637             {
00638                 if (obs > 10)
00639                 {
00640                     speedLeft = VITESSE_OBS;
00641                     speedRight = -VITESSE_OBS/2;
00642                     dir = 'G';
00643                 }
00644                 else
00645                 {
00646                     speedLeft = 0;
00647                     speedRight = 0;
00648
00649                     if (table[0] > table[1] && table[0] > 1000)
00650                     {
00651                         dir = 'G';
```

```
00652                         speedLeft = VITESSE_OBS/2;
00653                         speedRight = -VITESSE_OBS/2;
00654                         if (obs%2 == 0)
00655                             obs++;
00656                     }
00657                     else if (table[0] < table[1] && table[1] > 1000)
00658                     {
00659                         dir = 'D';
00660                         speedLeft = -VITESSE_OBS/2;
00661                         speedRight = VITESSE_OBS/2;
00662                         if (obs%2 == 1)
00663                             obs++;
00664                     }
00665                 }
00666             }
00667             else
00668             {
00669                 speedLeft = VITESSE_OBS;
00670                 speedRight = VITESSE_OBS;
00671                 dir = 'F';
00672                 obs = 0;
00673             }
00674         }
00675         else if (mode == MODE_CAM)
00676         {
00677             dir = 'N';
00678             obs = 0;
00679 
00680             if(x < 0 || y < 0){
00681                 speedLeft = 0;
00682                 speedRight = 0;
00683             }
00684             else {
00685                 speedLeft = VITESSE_CAM - ((CAMERA_X_MAX/2 - x))/3; // (int) (((float)
    ((x-CAMERA_X_MAX/2)/CAMERA_X_MAX))*500);
00686                 speedRight = VITESSE_CAM + ((CAMERA_X_MAX/2 - x))/3; // (int) (((float)
    (x/CAMERA_X_MAX))*500);
00687             }
00688 
00689 
00690 
00691             /*if (x > CAMERA_X_MIN+CAMERA_X_TIER && x < CAMERA_X_MAX-CAMERA_X_TIER && y > CAMERA_Y_MIN
    && y <CAMERA_Y_MIN+CAMERA_Y_TIER) //AVANT
00692             {
00693                 speedLeft = VITESSE_CAM;
00694                 speedRight = VITESSE_CAM;
00695             }
00696             else if (x > CAMERA_X_MAX-CAMERA_X_TIER && x < CAMERA_X_MAX && y > CAMERA_Y_MIN && y <
    CAMERA_Y_MIN+CAMERA_Y_TIER) //AVANT_DROITE:
00697             {
00698                 speedLeft = VITESSE_CAM;
00699                 speedRight = VITESSE_CAM/2;
00700             }
00701             else if (x > CAMERA_X_MIN && x < CAMERA_X_MIN+CAMERA_X_TIER && y > CAMERA_Y_MIN && y <
    CAMERA_Y_MIN+CAMERA_Y_TIER) //AVANT_GAUCHE:
00702             {
00703                 speedLeft = VITESSE_CAM/2;
00704                 speedRight = VITESSE_CAM;
00705             }
00706             else if (x > CAMERA_X_MIN+CAMERA_X_TIER && x < CAMERA_X_MAX-CAMERA_X_TIER && y >
    CAMERA_Y_MIN+CAMERA_Y_TIER && y <CAMERA_Y_MAX-CAMERA_Y_TIER) //STOP
00707             {
00708                 speedLeft = 0;
00709                 speedRight = 0;
00710             }
00711             else if (x > CAMERA_X_MAX-CAMERA_X_TIER && x < CAMERA_X_MAX && y >
    CAMERA_Y_MIN+CAMERA_Y_TIER && y <CAMERA_Y_MAX-CAMERA_Y_TIER) //DROITE
00712             {
00713                 speedLeft = VITESSE_CAM;
00714                 speedRight = -VITESSE_CAM;
00715             }
00716             else if (x > CAMERA_X_MIN && x < CAMERA_X_MIN+CAMERA_X_TIER && y >
    CAMERA_Y_MIN+CAMERA_Y_TIER && y <CAMERA_Y_MAX-CAMERA_Y_TIER) //GAUCHE
00717             {
00718                 speedLeft = -VITESSE_CAM;
00719                 speedRight = VITESSE_CAM;
00720             }
00721             else if (x > CAMERA_X_MIN+CAMERA_X_TIER && x < CAMERA_X_MAX-CAMERA_X_TIER && y >
    CAMERA_Y_MAX-CAMERA_Y_TIER && y < CAMERA_Y_MAX) //RECULE:
00722             {
00723                 speedLeft = -VITESSE_CAM;
00724                 speedRight = -VITESSE_CAM;
00725             }
00726             else if (x > CAMERA_X_MAX-CAMERA_X_TIER && x < CAMERA_X_MAX && y >
    CAMERA_Y_MAX-CAMERA_Y_TIER && y < CAMERA_Y_MAX) //RECULE_DROITE:
00727             {
00728                 speedLeft = -VITESSE_CAM/2;
```

```
00729                 speedRight = -VITESSE_CAM;
00730             }
00731             else if (x > CAMERA_X_MIN && x < CAMERA_X_MIN+CAMERA_X_TIER && y >
      CAMERA_Y_MAX-CAMERA_Y_TIER && y < CAMERA_Y_MAX) //RECULE_GAUCHE:
00732             {
00733                 speedLeft = -VITESSE_CAM;
00734                 speedRight = -VITESSE_CAM/2;
00735             }
00736             else
00737             {
00738                 speedLeft = 0;
00739                 speedRight = 0;
00740             }*/
00741         }
00742 
00743         #if DEBUG_MOTOR
00744         printf("Motor L : %d || R : %d\r\n", speedLeft, speedRight);
00745         #endif
00746 
00747         xQueueSend( q_mot_L, ( void * ) &speedLeft,  portMAX_DELAY );
00748         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00749 
00750         xQueueSend( q_mot_R, ( void * ) &speedRight,  portMAX_DELAY );
00751         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00752 
00753     #if MICROROS
00754         MsgToPub.dir = dir;
00755         MsgToPub.mode = speedLeft; //mode;
00756         MsgToPub.speed = speedRight; //speed;
00757         if (!uxQueueMessagesWaiting(qhMR_pub))
00758             xQueueSend(qhMR_pub, ( void * ) &MsgToPub, portMAX_DELAY);
00759     #endif //MICROROS
00760 
00761     #if LCD
00762         if (!uxQueueMessagesWaiting(qhLCD))
00763         {
00764             pxMessage.data=mode;
00765             pxMessage.command=dir;
00766             xQueueSend( qhLCD, ( void * ) &pxMessage, portMAX_DELAY);
00767         }
00768     #endif //LCD
00769 
00770         vTaskDelay(SAMPLING_PERIOD_ms);
00771     }
00772 #endif //SYNCHRO_EX
00773 }
00774 
00775 int main(void)
00776 {
00777   HAL_Init();
00778   SystemClock_Config();
00779   MX_GPIO_Init();
00780   MX_DMA_Init();
00781   MX_USART2_UART_Init();
00782   MX_I2C1_Init();
00783   MX_USART1_UART_Init();
00784 
00785   RetargetInit(&huart2); //make printf and scanf work with uart2
00786   printf("%cc%c[2J%c[0;0HTitouan//Jeremy//Louanne\r\n", 0x1b, 0x1b, 0x1b);
00787 
00788   motorCommand_Init();
00789   quadEncoder_Init();
00790   captDistIR_Init();
00791 
00792   HAL_Delay(500);
00793 
00794 #if VL53
00795   initVL53L0X();
00796   for (int i=0 ; i<20 ; i++)
00797   {
00798       printf("%d\r\n", readRangeSingleMillimeters()/10);
00799   }
00800   HAL_Delay(500);
00801 #endif //VL53
00802 
00803   // Test Ecran LCD
00804 #if LCD
00805   groveLCD_begin(16,2,0); // !! cette fonction prend du temps
00806   HAL_Delay(100);
00807   groveLCD_setCursor(0,0);
00808   groveLCD_setColor(1);
00809   groveLCD_term_printf("Titouan//Jeremy//Louanne");
00810   HAL_Delay(1000);
00811 #endif //LCD
00812 
00813   osKernelInitialize();
00814   //defaultTaskHandle = osThreadNew(microros_task, NULL, &defaultTask_attributes);
```

```
00815
00816 #if SYNCHRO_EX == EXSTARTUP
00817     #if MICROROS
00818     xTaskCreate( microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, 24,
      NULL);
00819     #endif //MICROROS
00820     xTaskCreate( task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */,
      NULL, 27, NULL);
00821     xTaskCreate( task_Motor_Left, ( const portCHAR * ) "task Mot L", 128 /* stack size */, NULL, 25,
      NULL);
00822     xTaskCreate( task_Motor_Right, ( const portCHAR * ) "task Mot R", 128 /* stack size */, NULL, 26,
      NULL);
00823     #if LCD
00824     xTaskCreate( task_Grove_LCD, ( const portCHAR * ) "task Mot R", 128 /* stack size */, NULL, 23,
      NULL);
00825     #endif
00826 #elif SYNCHRO_EX == EXTEST_UART2
00827     xTaskCreate(test_uart2, ( const portCHAR * ) "task print uart 2", 128 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00828 #elif SYNCHRO_EX == EXCORRECTOR
00829     xTaskCreate(test_motor, ( const portCHAR * ) "task test motor", 128 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00830 #elif SYNCHRO_EX == EXTESTCORRECTOR
00831     xTaskCreate(task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL,
      27, NULL);
00832     xTaskCreate(task_Motor_Left, ( const portCHAR * ) "task Motor Left", 128 /* stack size */, NULL,
      25, NULL);
00833     xTaskCreate(task_Motor_Right, ( const portCHAR * ) "task Motor Right", 128 /* stack size */, NULL,
      26, NULL);
00834 #elif SYNCHRO_EX == EXTEST_VL53
00835     xTaskCreate(test_vl53, ( const portCHAR * ) "test_vl53", 128 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00836 #elif SYNCHRO_EX == EXTEST_MICROROS
00837     xTaskCreate(microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00838 #elif SYNCHRO_EX == EXFINAL
00839     #if MICROROS
00840     xTaskCreate(microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, 24,
      NULL);
00841     #endif //MICROROS
00842     xTaskCreate(task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL,
      27, NULL);
00843     xTaskCreate(task_Motor_Left, ( const portCHAR * ) "task Motor Left", 128 /* stack size */, NULL,
      25, NULL);
00844     xTaskCreate(task_Motor_Right, ( const portCHAR * ) "task Motor Right", 128 /* stack size */, NULL,
      26, NULL);
00845
00846     #if VL53
00847     xTaskCreate(task_VL53, ( const portCHAR * ) "task VL53", 128 /* stack size */, NULL, 23, NULL);
00848     #endif //VL53
00849
00850     #if LCD
00851     xTaskCreate(task_Grove_LCD, ( const portCHAR * ) "task LCD", 128 /* stack size */, NULL, 23,
      NULL);
00852     #endif //LCD
00853 #endif //SYNCHRO_EX
00854
00855     vSemaphoreCreateBinary(xSem_Supervision);
00856
00857     q_mot_L = xQueueCreate(1, sizeof(int16_t));
00858     q_mot_R = xQueueCreate(1, sizeof(int16_t));
00859     qhVl53 = xQueueCreate(1, sizeof(int));
00860
00861     qhMR_sub = xQueueCreate(1, sizeof(MicroRosSubMsg));
00862     qhMR_pub = xQueueCreate(1, sizeof(MicroRosPubMsg));
00863     qhLCD = xQueueCreate(1, sizeof(AMessage));
00864
00865   osKernelStart();
00866   while(1)
00867   {
00868
00869   }
00870 }
00871
00872 void test_uart2(void *pvParameters)
00873 {
00874     char buf[100] = "";
00875     for(;;)
00876     {
00877         printf("Veuillez saisir votre nom :\r\n");
00878         scanf("%s", buf);
00879         printf("bonjour et bienvenue %s\r\n", buf);
00880         vTaskDelay(SAMPLING_PERIOD_ms);
00881     }
00882 }
00883
00884 void test_vl53(void *pvParameters)
```

```
00885 {
00886     uint16_t val;
00887 
00888     for(;;)
00889     {
00890         val = readRangeSingleMillimeters()/10;
00891         printf("Distance capteur : %d\r\n", val);
00892     }
00893 }
00894 
00895 void test_motor(void *pvParameters)
00896 {
00897     int16_t  consigne = TEST_CORRECTOR_DUTY;
00898     if (consigne < 0 || consigne > 200)
00899         consigne = 150;
00900     int speed = 0;
00901     int i = 0;
00902 
00903     for (;;)
00904     {
00905         #if TEST_LEFT_MOTOR
00906         motorLeft_SetDuty(consigne);
00907         speed = quadEncoder_GetSpeedL();
00908         #else
00909         motorRight_SetDuty(consigne);
00910         speed = quadEncoder_GetSpeedR();
00911         #endif
00912 
00913         if(i<NB)
00914         {
00915             tab_speed[i]=speed;
00916             i++;
00917         }
00918         else
00919             printf("sampling end");
00920         vTaskDelay(SAMPLING_PERIOD_ms);
00921     }
00922 }
00923 
00924 //=============================================================================
00933 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
00934 {
00935   if (htim->Instance == TIM4)
00936   {
00937     HAL_IncTick();
00938   }
00939 }
00940 //=============================================================================
00941 void Error_Handler(void)
00942 {
00943   __disable_irq();
00944   while (1)
00945   {}
00946 }
00947 //=============================================================================
00948 #ifdef  USE_FULL_ASSERT
00956 void assert_failed(uint8_t *file, uint32_t line)
00957 {
00958   /* USER CODE BEGIN 6 */
00959   /* User can add his own implementation to report the file name and line number,
00960      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
00961   /* USER CODE END 6 */
00962 }
00963 #endif /* USE_FULL_ASSERT */
```

5.30 microROS.c

```
00001 #include "main.h"
00002 
00003 #define STRING 0
00004 
00005 void createPublisher(rcl_publisher_t* publisher,
00006     const rcl_node_t* node,
00007     const rosidl_message_type_support_t* type_support,
00008     const char* topic_name,
00009     std_msgs__msg__Int32* msg)
00010 {
00011     rcl_ret_t ret = rclc_publisher_init_default(publisher, node, type_support, topic_name);
00012     printf("Publisher %s is created with result %d\r\n", topic_name, (int)ret);
00013 
00014 #if STRING == 1
00015     (*msg).data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
00016     (*msg).data.size = 0;
```

```
00017     (*msg).data.capacity = ARRAY_LEN;
00018 #else
00019     (*msg).data = 0;
00020 #endif
00021
00022 }
00023
00024 void createSubscriber(rcl_subscription_t* subscription,
00025     rcl_node_t* node,
00026     const rosidl_message_type_support_t* type_support,
00027     const char* topic_name,
00028     std_msgs__msg__Int32* msg)
00029 {
00030     *subscription = rcl_get_zero_initialized_subscription();
00031
00032     rcl_ret_t ret = rclc_subscription_init_default(subscription, node,
00033         type_support, topic_name);
00034     printf("Subscription %s is created with result %d\r\n", topic_name, (int)ret);
00035
00036 #if STRING == 1
00037     (*msg).data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
00038     (*msg).data.size = 0;
00039     (*msg).data.capacity = ARRAY_LEN;
00040 #else
00041     (*msg).data = 0;
00042 #endif
00043 }
```

5.31 microros_allocators.c

```
00001
00002 #include <unistd.h>
00003 #include "cmsis_os.h"
00004
00005 int absoluteUsedMemory = 0;
00006 int usedMemory = 0;
00007
00008 void *pvPortMallocMicroROS( size_t xWantedSize );
00009 void vPortFreeMicroROS( void *pv );
00010 void *pvPortReallocMicroROS( void *pv, size_t xWantedSize );
00011 size_t getBlockSize( void *pv );
00012 void *pvPortCallocMicroROS( size_t num, size_t xWantedSize );
00013
00014 void * microros_allocate(size_t size, void * state){
00015   (void) state;
00016   // printf("-- Alloc %d (prev: %d B)\n",size, xPortGetFreeHeapSize());
00017   absoluteUsedMemory += size;
00018   usedMemory += size;
00019   return pvPortMallocMicroROS(size);
00020 }
00021
00022 void microros_deallocate(void * pointer, void * state){
00023   (void) state;
00024   // printf("-- Free %d (prev: %d B)\n",getBlockSize(pointer), xPortGetFreeHeapSize());
00025   if (NULL != pointer){
00026     usedMemory -= getBlockSize(pointer);
00027     vPortFreeMicroROS(pointer);
00028   }
00029 }
00030
00031 void * microros_reallocate(void * pointer, size_t size, void * state){
00032   (void) state;
00033   // printf("-- Realloc %d -> %d (prev: %d B)\n",getBlockSize(pointer),size, xPortGetFreeHeapSize());
00034   absoluteUsedMemory += size;
00035   usedMemory += size;
00036   if (NULL == pointer){
00037     return pvPortMallocMicroROS(size);
00038   } else {
00039     usedMemory -= getBlockSize(pointer);
00040     return pvPortReallocMicroROS(pointer,size);
00041   }
00042 }
00043
00044 void * microros_zero_allocate(size_t number_of_elements, size_t size_of_element, void * state){
00045   (void) state;
00046   // printf("-- Calloc %d x %d = %d -> (prev: %d B)\n",number_of_elements,size_of_element,
      number_of_elements*size_of_element, xPortGetFreeHeapSize());
00047   absoluteUsedMemory += number_of_elements*size_of_element;
00048   usedMemory += number_of_elements*size_of_element;
00049   return pvPortCallocMicroROS(number_of_elements,size_of_element);
00050 }
```

5.32 microros_time.c

```
00001 #include <unistd.h>
00002 #include <time.h>
00003 #include "cmsis_os.h"
00004
00005 #define MICROSECONDS_PER_SECOND     ( 1000000LL )
00006 #define NANOSECONDS_PER_SECOND      ( 1000000000LL )
00007 #define NANOSECONDS_PER_TICK        ( NANOSECONDS_PER_SECOND / configTICK_RATE_HZ )
00009 void UTILS_NanosecondsToTimespec( int64_t llSource,
00010                                   struct timespec * const pxDestination )
00011 {
00012     long lCarrySec = 0;
00013
00014     /* Convert to timespec. */
00015     pxDestination->tv_sec = ( time_t ) ( llSource / NANOSECONDS_PER_SECOND );
00016     pxDestination->tv_nsec = ( long ) ( llSource % NANOSECONDS_PER_SECOND );
00017
00018     /* Subtract from tv_sec if tv_nsec < 0. */
00019     if( pxDestination->tv_nsec < 0L )
00020     {
00021         /* Compute the number of seconds to carry. */
00022         lCarrySec = ( pxDestination->tv_nsec / ( long ) NANOSECONDS_PER_SECOND ) + 1L;
00023
00024         pxDestination->tv_sec -= ( time_t ) ( lCarrySec );
00025         pxDestination->tv_nsec += lCarrySec * ( long ) NANOSECONDS_PER_SECOND;
00026     }
00027 }
00028
00029 int clock_gettime( int clock_id,
00030                    struct timespec * tp )
00031 {
00032     TimeOut_t xCurrentTime = { 0 };
00033
00034     /* Intermediate variable used to convert TimeOut_t to struct timespec.
00035      * Also used to detect overflow issues. It must be unsigned because the
00036      * behavior of signed integer overflow is undefined. */
00037     uint64_t ullTickCount = 0ULL;
00038
00039     /* Silence warnings about unused parameters. */
00040     ( void ) clock_id;
00041
00042     /* Get the current tick count and overflow count. vTaskSetTimeOutState()
00043      * is used to get these values because they are both static in tasks.c. */
00044     vTaskSetTimeOutState( &xCurrentTime );
00045
00046     /* Adjust the tick count for the number of times a TickType_t has overflowed.
00047      * portMAX_DELAY should be the maximum value of a TickType_t. */
00048     ullTickCount = ( uint64_t ) ( xCurrentTime.xOverflowCount ) << ( sizeof( TickType_t ) * 8 );
00049
00050     /* Add the current tick count. */
00051     ullTickCount += xCurrentTime.xTimeOnEntering;
00052
00053     /* Convert ullTickCount to timespec. */
00054     UTILS_NanosecondsToTimespec( ( int64_t ) ullTickCount * NANOSECONDS_PER_TICK, tp );
00055
00056     return 0;
00057 }
```

5.33 motorCommand.c

```
00001 /*
00002  * MotorCommand.c
00003  */
00004
00005 #include "motorCommand.h"
00006
00007 static TIM_HandleTypeDef    TimHandle;
00008 static TIM_OC_InitTypeDef   sConfigOC;
00009
00010 //================================================================
00011 //          PWM INIT
00012 // TIMER 3 (PWM)  : CH1 et CH2
00013 // ENABLE : Sortie Logique (GPIO) PA7
00014 //================================================================
00015
00016 void motorCommand_Init(void)
00017 {
00018     unsigned int uwPrescalerValue = 0;
00019
00020     /* Compute the prescaler value to have TIM4 counter clock equal to 10MHz */
00021       uwPrescalerValue = (unsigned int) ((SystemCoreClock / 10000000) - 1);
00022       TimHandle.Instance = TIM3;
```

```
00023       TimHandle.Init.Period = 200 - 1; // 100MHz/200=50kHz
00024       TimHandle.Init.Prescaler = uwPrescalerValue;
00025       TimHandle.Init.ClockDivision = 0;
00026       TimHandle.Init.CounterMode = TIM_COUNTERMODE_UP;
00027
00028       HAL_TIM_Base_Init(&TimHandle);
00029
00030       sConfigOC.OCMode = TIM_OCMODE_PWM1;
00031       sConfigOC.Pulse = 0x5;// Specifies the pulse value to be loaded into the Capture Compare
      Register. This parameter can be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF */
00032
00033       sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
00034       sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
00035
00036      HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfigOC, TIM_CHANNEL_1);
00037      HAL_TIM_PWM_ConfigChannel(&TimHandle, &sConfigOC, TIM_CHANNEL_2);
00038
00039      // CHANGEMENT DU RAPPORT CYCLIQUE
00040      __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_1, 100);  // 100 : moteurs au repos
00041      __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_2, 100);
00042
00043       HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_1); // MOTOR RIGHT
00044       HAL_TIM_PWM_Start(&TimHandle, TIM_CHANNEL_2); // MOTOR LEFT
00045
00046       // ENABLE MOTEUR (SI INVERSEUR)
00047       //HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 0);
00048       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 0);
00049 }
00050
00051 //=================================================================
00052 //          SET DUTY CYCLE LEFT
00053 //=================================================================
00054 void motorLeft_SetDuty(int duty)
00055 {
00056     __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_1, duty);
00057 }
00058 //=================================================================
00059 //          SET DUTY CYCLE RIGHT
00060 //=================================================================
00061 void motorRight_SetDuty(int duty)
00062 {
00063     __HAL_TIM_SetCompare(&TimHandle, TIM_CHANNEL_2, duty);
00064 }
00065 //=================================================================
00066
00067
```

5.34 quadEncoder.c

```
00001 /*
00002  * QuadEncoder.c
00003  */
00004 #include "quadEncoder.h"
00005
00006 #define SAMPLING_PERIOD_ms      5
00007 #define TE_ms   SAMPLING_PERIOD_ms
00008 #define USE_QUAD_ENCODER_1250_CPR 1
00009
00010 #if USE_QUAD_ENCODER_1250_CPR
00011 #define COUNT_PER_ROUND 1250
00012 #define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
00013 #define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*6555)/1000)
00014 #define HALF_MAX_COUNT (MAX_COUNT>>1)
00015 #define COEFF   6555
00016 #endif
00017
00018 #if USE_QUAD_ENCODER_1000_CPR
00019 #define COUNT_PER_ROUND 1000
00020 #define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
00021 #define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*8192)/1000)
00022 #define HALF_MAX_COUNT (MAX_COUNT>>1)
00023 #define COEFF 8192
00024 #endif
00025
00026 #if USE_QUAD_ENCODER_500_CPR
00027 #define COUNT_PER_ROUND 500
00028 #define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
00029 #define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*16392)/1000)
00030 #define HALF_MAX_COUNT (MAX_COUNT>>1)
00031 #define COEFF 16392
00032 #endif
00033
00034 #if USE_QUAD_ENCODER_250_CPR
```

```
00035 #define COUNT_PER_ROUND 250
00036 #define MAX_CNT_PER_REV (COUNT_PER_ROUND * 4 - 1)
00037 #define MAX_COUNT (int)(((unsigned long)MAX_CNT_PER_REV*32768)/1000)
00038 #define HALF_MAX_COUNT (MAX_COUNT>>1)
00039 #define COEFF 32768
00040 #endif
00041
00042
00043
00044 TIM_HandleTypeDef    TimEncoderHandleLeft;
00045 TIM_HandleTypeDef    TimEncoderHandleRight;
00046
00047 /*****************************************************************************
00048  * TIMER 1, CHANNEL 1 et 2  --> RIGHT
00049  * TIMER 2, CHANNEL 1 et 2  --> LEFT
00050 ***************************************************************************** */
00051 int indexL=0;
00052 static int indexR=0;
00053
00054 //=============================================================
00055 //      TIMER INIT
00056 //=============================================================
00057
00058 void quadEncoder_Init(void)
00059 {
00060     TIM_Encoder_InitTypeDef sConfig;
00061     //-------------------------------------------------
00062     // TIMER 1
00063     //-------------------------------------------------
00064     TimEncoderHandleLeft.Instance = TIM1;
00065     TimEncoderHandleLeft.Init.Prescaler = 0;
00066     TimEncoderHandleLeft.Init.CounterMode = TIM_COUNTERMODE_UP;
00067     TimEncoderHandleLeft.Init.Period = COUNT_PER_ROUND*4;
00068     TimEncoderHandleLeft.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
00069
00070     sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
00071     sConfig.IC1Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
00072     sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
00073     sConfig.IC1Prescaler = TIM_ICPSC_DIV4;
00074     sConfig.IC1Filter = 0x0F;
00075     sConfig.IC2Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
00076     sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;//TIM_ICSELECTION_DIRECTTI;
      //TIM_TI1SELECTION_XORCOMBINATION
00077     sConfig.IC2Prescaler = TIM_ICPSC_DIV4;
00078     sConfig.IC2Filter = 0x0F;
00079
00080     HAL_TIM_Encoder_Init(&TimEncoderHandleLeft, &sConfig);
00081
00082     __HAL_TIM_SetCounter(&TimEncoderHandleLeft, 0);
00083
00084     HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_1);
00085     HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_2);
00086
00087     //-------------------------------------------------
00088     // TIMER 2
00089     //-------------------------------------------------
00090     TimEncoderHandleRight.Instance = TIM2;
00091     TimEncoderHandleRight.Init.Prescaler = 0;
00092     TimEncoderHandleRight.Init.CounterMode = TIM_COUNTERMODE_UP;
00093     TimEncoderHandleRight.Init.Period = COUNT_PER_ROUND*4;
00094     TimEncoderHandleRight.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
00095
00096     sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
00097     sConfig.IC1Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
00098     sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
00099     sConfig.IC1Prescaler = TIM_ICPSC_DIV4;
00100     sConfig.IC1Filter = 0x0F;
00101     sConfig.IC2Polarity = TIM_INPUTCHANNELPOLARITY_RISING;
00102     sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;//TIM_ICSELECTION_DIRECTTI;
      //TIM_TI1SELECTION_XORCOMBINATION
00103     sConfig.IC2Prescaler = TIM_ICPSC_DIV4;
00104     sConfig.IC2Filter = 0x0F;
00105
00106     HAL_TIM_Encoder_Init(&TimEncoderHandleRight, &sConfig);
00107
00108     __HAL_TIM_SetCounter(&TimEncoderHandleRight, 0);
00109
00110     HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_1);
00111     HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_2);
00112 }
00113
00114 //=============================================================
00115 //      POSITION LEFT CALC
00116 //=============================================================
00117
00118 void quadEncoder_PosCalcL(int* AngPos)
00119 {
```

```
00120
00121 int POSCNTcopy = 0;
00122 POSCNTcopy = (int)TIM1->CNT;
00123 AngPos[1] = AngPos[0];
00124 AngPos[0] = (unsigned int)(((unsigned long)POSCNTcopy * COEFF)/1000); // 0 <= POSCNT <= 4999 to 0 <=
      AngPos <= 32767
00125 }
00126
00127 //=============================================================
00128 //      POSITION RIGHT CALC
00129 //=============================================================
00130
00131 void quadEncoder_PosCalcR(int* AngPos)
00132 {
00133
00134 int POSCNTcopy = 0;
00135 POSCNTcopy = (int)TIM2->CNT;
00136 AngPos[1] = AngPos[0];
00137 AngPos[0] = (unsigned int)(((unsigned long)POSCNTcopy * COEFF)/1000); // 0 <= POSCNT <= 4999 to 0 <=
      AngPos <= 32767
00138 }
00139
00140 //=============================================================
00141 //      POSITION LEFT 16 BITS
00142 //=============================================================
00143
00144 int16_t quadEncoder_GetPos16L(void)
00145 {
00146     uint16_t PosL = 0;
00147     PosL=TIM1->CNT;
00148     return (int16_t)PosL;
00149
00150     }
00151
00152 //=============================================================
00153 //      POSITION RIGHT 16 BITS
00154 //=============================================================
00155
00156 int16_t quadEncoder_GetPos16R(void)
00157 {
00158     uint16_t PosR = 0;
00159     PosR=TIM2->CNT;
00160     return (int16_t)PosR;
00161 }
00162 //=============================================================
00163 //      POSITION LEFT 32 BITS   (pos 16 bits + nombre de tours)
00164 //=============================================================
00165
00166 int32_t quadEncoder_GetPos32L(void)
00167 {
00168     int32_t  PosL = 0;
00169     PosL=indexL*4*COUNT_PER_ROUND + (int32_t) quadEncoder_GetPos16L();
00170     return PosL;
00171 }
00172
00173 //=============================================================
00174 //      POSITION RIGHT 32 BITS  (pos 16 bits + nombre de tours)
00175 //=============================================================
00176
00177 int32_t quadEncoder_GetPos32R(void)
00178 {
00179     int32_t PosR = 0;
00180     PosR=indexR*4*COUNT_PER_ROUND + (int32_t) quadEncoder_GetPos16R();
00181     return PosR;
00182 }
00183
00184 //=============================================================
00185 //      SPEED LEFT
00186 //--> must be called every Te
00187 //=============================================================
00188
00189 int16_t quadEncoder_GetSpeedL(void)
00190 {
00191     static int AngPos[2] = {0,0};
00192     static int16_t SpeedL=0;
00193
00194     quadEncoder_PosCalcL(AngPos);
00195     SpeedL = AngPos[0] - AngPos[1];
00196     if (SpeedL >= 0)
00197     {
00198         if (SpeedL >= HALF_MAX_COUNT)
00199             {
00200             SpeedL = SpeedL - MAX_COUNT;
00201             }
00202     }
00203     else
00204     {
```

```
00205         if (SpeedL < -HALF_MAX_COUNT)
00206             {
00207             SpeedL = SpeedL + MAX_COUNT;
00208             }
00209     }
00210 
00211     //**********************************
00212     // CONVERT RPM
00213     // 1 tour = 32767
00214     // Nbre de Tours pendant Te: DELTA_pos/32767
00215     // Nbre de Tours pendant 1s (Te en ms) : (DELTA_pos/32767)*(1000/Te)
00216     // Nbre de Tours par minute : : (DELTA_pos/32767)*((60*1000)/Te)
00217 
00218     SpeedL=(SpeedL*60*1000)/(32767*TE_ms);
00219     return SpeedL;
00220 }
00221 
00222 //=================================================================
00223 //      SPEED RIGHT
00224 //-->  must be called every Te
00225 //=================================================================
00226 
00227 int16_t quadEncoder_GetSpeedR(void)
00228 {
00229     static int AngPos[2] = {0,0};
00230     static int16_t SpeedR=0;
00231 
00232 
00233     quadEncoder_PosCalcR(AngPos);
00234     SpeedR = AngPos[0] - AngPos[1];
00235     if (SpeedR >= 0)
00236     {
00237         if (SpeedR >= HALF_MAX_COUNT)
00238             {
00239             SpeedR = SpeedR - MAX_COUNT;
00240             }
00241     }
00242     else
00243     {
00244         if (SpeedR < -HALF_MAX_COUNT)
00245             {
00246             SpeedR = SpeedR + MAX_COUNT;
00247             }
00248     }
00249     //**********************************
00250     // CONVERT RPM
00251     // 1 tour = 32767
00252     // Nbre de Tours pendant Te: DELTA_pos/32767
00253     // Nbre de Tours pendant 1s (Te en ms) : (DELTA_pos/32767)*(1000/Te)
00254     // Nbre de Tours par minute : : (DELTA_pos/32767)*((60*1000)/Te)
00255 
00256     SpeedR=(SpeedR*60*1000)/(32767*TE_ms);
00257     return SpeedR;
00258 }
00259 
00260 //=================================================================
00261 //      MAJ index Left
00262 //=================================================================
00263 
00264 void quadEncoder_CallbackIndexL()
00265 {
00266                 if (__HAL_TIM_DIRECTION_STATUS(&TimEncoderHandleLeft)==1)
00267                 {
00268                     indexL--;
00269                 }
00270                 else
00271                 {
00272                     indexL++;
00273                 }
00274 
00275 
00276                 __HAL_TIM_SetCounter(&TimEncoderHandleLeft, 0);     // RAZ Counter
00277                 HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_1);
00278                 HAL_TIM_Encoder_Start(&TimEncoderHandleLeft,TIM_CHANNEL_2);
00279 
00280 }
00281 //=================================================================
00282 //      MAJ index Right
00283 //=================================================================
00284 
00285 void quadEncoder_CallbackIndexR()
00286 {
00287                 if (__HAL_TIM_DIRECTION_STATUS(&TimEncoderHandleRight)==1)
00288                 {
00289                     indexR--;
00290 
00291                 }
```

```
00292                 else
00293                 {
00294                     indexR++;
00295                 }
00296 
00297                 __HAL_TIM_SetCounter(&TimEncoderHandleRight, 0);            // RAZ Counter
00298                 HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_1);
00299                 HAL_TIM_Encoder_Start(&TimEncoderHandleRight,TIM_CHANNEL_2);
00300 
00301 }
00302 //==============================================================
00303 
00304 
00305 
```

5.35 retarget.c

```
00001 // All credit to Carmine Noviello for this code
00002 //
      https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/src/retarget/retarget.c
00003 
00004 #include <_ansi.h>
00005 #include <_syslist.h>
00006 #include <errno.h>
00007 #include <sys/time.h>
00008 #include <sys/times.h>
00009 #include <limits.h>
00010 #include <signal.h>
00011 #include <../Inc/retarget.h>
00012 #include <stdint.h>
00013 #include <stdio.h>
00014 
00015 #if !defined(OS_USE_SEMIHOSTING)
00016 
00017 #define STDIN_FILENO  0
00018 #define STDOUT_FILENO 1
00019 #define STDERR_FILENO 2
00020 
00021 UART_HandleTypeDef *gHuart;
00022 
00023 void RetargetInit(UART_HandleTypeDef *huart) {
00024   gHuart = huart;
00025 
00026   /* Disable I/O buffering for STDOUT stream, so that
00027    * chars are sent out as soon as they are printed. */
00028   setvbuf(stdout, NULL, _IONBF, 0);
00029 }
00030 
00031 int _isatty(int fd) {
00032   if (fd >= STDIN_FILENO && fd <= STDERR_FILENO)
00033     return 1;
00034 
00035   errno = EBADF;
00036   return 0;
00037 }
00038 
00039 int _write(int fd, char* ptr, int len) {
00040   HAL_StatusTypeDef hstatus;
00041 
00042   if (fd == STDOUT_FILENO || fd == STDERR_FILENO) {
00043     hstatus = HAL_UART_Transmit(gHuart, (uint8_t *) ptr, len, HAL_MAX_DELAY);
00044     if (hstatus == HAL_OK)
00045       return len;
00046     else
00047       return EIO;
00048   }
00049   errno = EBADF;
00050   return -1;
00051 }
00052 
00053 int _close(int fd) {
00054   if (fd >= STDIN_FILENO && fd <= STDERR_FILENO)
00055     return 0;
00056 
00057   errno = EBADF;
00058   return -1;
00059 }
00060 
00061 int _lseek(int fd, int ptr, int dir) {
00062   (void) fd;
00063   (void) ptr;
00064   (void) dir;
00065 
```

```
00066   errno = EBADF;
00067   return -1;
00068 }
00069
00070 int _read(int fd, char* ptr, int len) {
00071   HAL_StatusTypeDef hstatus;
00072
00073   if (fd == STDIN_FILENO) {
00074     hstatus = HAL_UART_Receive(gHuart, (uint8_t *) ptr, 1, HAL_MAX_DELAY);
00075     if (hstatus == HAL_OK)
00076       return 1;
00077     else
00078       return EIO;
00079   }
00080   errno = EBADF;
00081   return -1;
00082 }
00083
00084 int _fstat(int fd, struct stat* st) {
00085   if (fd >= STDIN_FILENO && fd <= STDERR_FILENO) {
00086     st->st_mode = S_IFCHR;
00087     return 0;
00088   }
00089
00090   errno = EBADF;
00091   return 0;
00092 }
00093
00094 int _getpid(void)
00095 {
00096     return 1;
00097 }
00098
00099 int _kill(int pid, int sig)
00100 {
00101     errno = EINVAL;
00102     return -1;
00103 }
00104
00105 #endif //#if !defined(OS_USE_SEMIHOSTING)
```

5.36 stm32f4xx_hal_msp.c

```
00001 #include "main.h"
00002
00003 #define USART2_IRQ_PRIO 9
00004 #define USART6_IRQ_PRIO 10
00005 //#define EXTI1_IRQ_PRIO    7
00006 #define EXTI0_IRQ_PRIO  6
00007 #define EXTI15_10_IRQ_PRIO  7
00008 #define I2C1_ER_IRQ_PRIO    2
00009 #define I2C1_EV_IRQ_PRIO    11
00010 #define TIM5_IRQ_PRIO       12
00011
00012 extern DMA_HandleTypeDef hdma_usart1_rx;
00013 extern DMA_HandleTypeDef hdma_usart1_tx;
00014 extern DMA_HandleTypeDef hdma_usart2_rx;
00015 extern DMA_HandleTypeDef hdma_usart2_tx;
00016
00017 void HAL_PWM_Timer3_MspInit(void);
00018 void HAL_Encoder_Timer1_MspInit(void);
00019 void HAL_Encoder_Timer2_MspInit(void);
00020 void HAL_adcir_MspInit(void);
00021
00022 void HAL_MspInit(void)
00023 {
00024   __HAL_RCC_SYSCFG_CLK_ENABLE();
00025   __HAL_RCC_PWR_CLK_ENABLE();
00026
00027   __HAL_RCC_GPIOC_CLK_ENABLE();
00028   __HAL_RCC_GPIOH_CLK_ENABLE();
00029   __HAL_RCC_GPIOA_CLK_ENABLE();
00030   __HAL_RCC_GPIOB_CLK_ENABLE();
00031
00032   /* System interrupt init*/
00033   /* PendSV_IRQn interrupt configuration */
00034   HAL_NVIC_SetPriority(PendSV_IRQn, 15, 0);
00035
00036   HAL_PWM_Timer3_MspInit();
00037   HAL_Encoder_Timer1_MspInit();
00038   HAL_Encoder_Timer2_MspInit();
00039   HAL_adcir_MspInit();
00040 }
```

```
00041
00042 /*************************************************************
00043             ENCODER - TIMER1
00044 PWM1/1  --> PA8     -- Encodeur Voie A
00045 PWM1/2  --> PA9     -- Encodeur Voie B
00046 EXTI1   --> PB10        -- Index encodeur
00047 *************************************************************/
00048 void HAL_Encoder_Timer1_MspInit(void)
00049 {
00050       GPIO_InitTypeDef  GPIO_InitStruct;
00051
00052       __TIM1_CLK_ENABLE();
00053
00054       GPIO_InitStruct.Pin = GPIO_PIN_8 | GPIO_PIN_9;
00055       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP; // hal_gpio.h
00056       GPIO_InitStruct.Pull = GPIO_PULLUP;
00057       GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
00058       GPIO_InitStruct.Alternate =  GPIO_AF1_TIM1 ; // hal_gpio_ex.h
00059
00060       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00061
00062       GPIO_InitStruct.Pin = GPIO_PIN_10;
00063       GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
00064       GPIO_InitStruct.Pull = GPIO_NOPULL;
00065
00066       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00067
00068       /* Enable and set EXTI Line0 Interrupt to the lowest priority */
00069       HAL_NVIC_SetPriority(EXTI15_10_IRQn, EXTI15_10_IRQ_PRIO, 0);
00070       HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
00071 }
00072 /*************************************************************
00073               ENCODER - TIMER2
00074 PWM2/1  --> PA0     -- Encodeur Voie A
00075 PWM2/2  --> PA1     -- Encodeur Voie B
00076 EXTI    --> PC0     -- Index Moteur
00077 *************************************************************/
00078 void HAL_Encoder_Timer2_MspInit(void)
00079 {
00080       GPIO_InitTypeDef  GPIO_InitStruct;
00081
00082       __TIM2_CLK_ENABLE();
00083
00084       GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1;
00085       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP; // hal_gpio.h
00086       GPIO_InitStruct.Pull = GPIO_PULLUP;
00087       GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
00088       GPIO_InitStruct.Alternate =  GPIO_AF1_TIM2 ; // hal_gpio_ex.h
00089
00090       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00091
00092       GPIO_InitStruct.Pin = GPIO_PIN_0;
00093       GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
00094       GPIO_InitStruct.Pull = GPIO_NOPULL;
00095
00096       HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
00097
00098       /* Enable and set EXTI Line0 Interrupt to the lowest priority */
00099       HAL_NVIC_SetPriority(EXTI0_IRQn, EXTI0_IRQ_PRIO, 0);
00100       HAL_NVIC_EnableIRQ(EXTI0_IRQn);
00101 }
00102
00103 /*************************************************************
00104 //          PWM - TIMER4 COMMANDE MOTEURS
00105 PA6 --> PWM3/1
00106 PC7 --> PWM3/2
00107 PB3 --> ENABLE MOTEUR (actif état Bas)
00108 *************************************************************/
00109 void HAL_PWM_Timer3_MspInit(void)
00110 {
00111       GPIO_InitTypeDef  GPIO_InitStruct;
00112
00113       __TIM3_CLK_ENABLE();
00114
00115       GPIO_InitStruct.Pin = GPIO_PIN_6;
00116       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00117       GPIO_InitStruct.Pull = GPIO_PULLUP;
00118       GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
00119       GPIO_InitStruct.Alternate =  GPIO_AF2_TIM3 ; // hal_gpio_ex.h
00120
00121       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00122
00123       GPIO_InitStruct.Pin = GPIO_PIN_7;
00124       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00125       GPIO_InitStruct.Pull = GPIO_PULLUP;
00126       GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
00127       GPIO_InitStruct.Alternate =  GPIO_AF2_TIM3 ; // hal_gpio_ex.h
```

```
00128
00129       HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
00130
00131       // ENABLE MOTEUR : SORTIE LOGIQUE PB3
00132       GPIO_InitStruct.Pin = GPIO_PIN_7;
00133       GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00134       GPIO_InitStruct.Pull = GPIO_NOPULL;
00135
00136       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00137       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);
00138
00139       GPIO_InitStruct.Pin = GPIO_PIN_3;
00140       GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00141       GPIO_InitStruct.Pull = GPIO_PULLUP;
00142       GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00143
00144       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00145       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 1);
00146
00147
00148
00149
00150 }
00151
00152 /*************************************************************************
00153             ADC
00154 ADC1_4  --> PA4
00155 ADC1_8  --> PB0
00156 http://stm32f4-discovery.com/2014/04/library-06-ad-converter-on-stm32f4xx/
00157 *************************************************************************/
00158 void HAL_adcir_MspInit(void)
00159 {
00160       GPIO_InitTypeDef  GPIO_InitStruct;
00161       /* Peripheral clock enable */
00162       __ADC1_CLK_ENABLE();
00163
00164       GPIO_InitStruct.Pin = GPIO_PIN_4 ;
00165       GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
00166       GPIO_InitStruct.Pull = GPIO_NOPULL;
00167
00168       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00169
00170
00171       GPIO_InitStruct.Pin = GPIO_PIN_0 ;
00172       GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
00173       GPIO_InitStruct.Pull = GPIO_NOPULL;
00174
00175       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00176
00177 }
00178
00179
00180 /*************************************************************************
00181 * @brief I2C MSP Initialization
00182 * This function configures the hardware resources used in this example
00183 * @param hi2c: I2C handle pointer
00184 * @retval None
00185 *************************************************************************/
00186 void HAL_I2C_MspInit(I2C_HandleTypeDef* hi2c)
00187 {
00188   GPIO_InitTypeDef GPIO_InitStruct = {0};
00189   if(hi2c->Instance==I2C1)
00190   {
00191     __HAL_RCC_GPIOB_CLK_ENABLE();
00196     GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
00197     GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
00198     GPIO_InitStruct.Pull = GPIO_NOPULL;
00199     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00200     GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
00201     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00202
00203     /* Peripheral clock enable */
00204     __HAL_RCC_I2C1_CLK_ENABLE();
00205
00206       HAL_NVIC_SetPriority(I2C1_ER_IRQn, I2C1_ER_IRQ_PRIO, 0);
00207       HAL_NVIC_EnableIRQ(I2C1_ER_IRQn);
00208       HAL_NVIC_SetPriority(I2C1_EV_IRQn, I2C1_EV_IRQ_PRIO, 0);
00209       HAL_NVIC_EnableIRQ(I2C1_EV_IRQn);
00210
00211
00212
00213   }
00214
00215 }
00216
00217 /*************************************************************************
00218 * @brief I2C MSP De-Initialization
```

```
00219 * This function freeze the hardware resources used in this example
00220 * @param hi2c: I2C handle pointer
00221 * @retval None
00222 *******************************************************************/
00223 void HAL_I2C_MspDeInit(I2C_HandleTypeDef* hi2c)
00224 {
00225   if(hi2c->Instance==I2C1)
00226   {
00227   /* USER CODE BEGIN I2C1_MspDeInit 0 */
00228
00229   /* USER CODE END I2C1_MspDeInit 0 */
00230     /* Peripheral clock disable */
00231     __HAL_RCC_I2C1_CLK_DISABLE();
00232
00237     HAL_GPIO_DeInit(GPIOB, GPIO_PIN_6);
00238
00239     HAL_GPIO_DeInit(GPIOB, GPIO_PIN_7);
00240
00241   /* USER CODE BEGIN I2C1_MspDeInit 1 */
00242
00243   /* USER CODE END I2C1_MspDeInit 1 */
00244   }
00245
00246 }
00247
00248 /*******************************************************************
00249 * @brief UART MSP Initialization
00250 * This function configures the hardware resources used in this example
00251 * @param huart: UART handle pointer
00252 * @retval None
00253 *******************************************************************/
00254 void HAL_UART_MspInit(UART_HandleTypeDef* huart)
00255 {
00256   GPIO_InitTypeDef GPIO_InitStruct = {0};
00257   if(huart->Instance==USART1)
00258   {
00259   /* USER CODE BEGIN USART1_MspInit 0 */
00260
00261   /* USER CODE END USART1_MspInit 0 */
00262     /* Peripheral clock enable */
00263     __HAL_RCC_USART1_CLK_ENABLE();
00264
00265     __HAL_RCC_GPIOA_CLK_ENABLE();
00270     GPIO_InitStruct.Pin = GPIO_PIN_10;
00271     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00272     GPIO_InitStruct.Pull = GPIO_NOPULL;
00273     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00274     GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
00275     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00276
00277     GPIO_InitStruct.Pin = GPIO_PIN_6;
00278     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00279     GPIO_InitStruct.Pull = GPIO_NOPULL;
00280     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00281     GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
00282     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
00283
00284     /* USART1 DMA Init */
00285     /* USART1_RX Init */
00286     hdma_usart1_rx.Instance = DMA2_Stream2;
00287     hdma_usart1_rx.Init.Channel = DMA_CHANNEL_4;
00288     hdma_usart1_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
00289     hdma_usart1_rx.Init.PeriphInc = DMA_PINC_DISABLE;
00290     hdma_usart1_rx.Init.MemInc = DMA_MINC_ENABLE;
00291     hdma_usart1_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00292     hdma_usart1_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00293     hdma_usart1_rx.Init.Mode = DMA_CIRCULAR;
00294     hdma_usart1_rx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00295     hdma_usart1_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00296     if (HAL_DMA_Init(&hdma_usart1_rx) != HAL_OK)
00297     {
00298       Error_Handler();
00299     }
00300
00301     __HAL_LINKDMA(huart,hdmarx,hdma_usart1_rx);
00302
00303     /* USART1_TX Init */
00304     hdma_usart1_tx.Instance = DMA2_Stream7;
00305     hdma_usart1_tx.Init.Channel = DMA_CHANNEL_4;
00306     hdma_usart1_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
00307     hdma_usart1_tx.Init.PeriphInc = DMA_PINC_DISABLE;
00308     hdma_usart1_tx.Init.MemInc = DMA_MINC_ENABLE;
00309     hdma_usart1_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00310     hdma_usart1_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00311     hdma_usart1_tx.Init.Mode = DMA_NORMAL;
00312     hdma_usart1_tx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00313     hdma_usart1_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
```

```
00314     if (HAL_DMA_Init(&hdma_usart1_tx) != HAL_OK)
00315     {
00316       Error_Handler();
00317     }
00318
00319     __HAL_LINKDMA(huart,hdmatx,hdma_usart1_tx);
00320
00321     /* USART1 interrupt Init */
00322     HAL_NVIC_SetPriority(USART1_IRQn, 5, 0);
00323     HAL_NVIC_EnableIRQ(USART1_IRQn);
00324   /* USER CODE BEGIN USART1_MspInit 1 */
00325
00326   /* USER CODE END USART1_MspInit 1 */
00327   }
00328   else if(huart->Instance==USART2)
00329   {
00330   /* USER CODE BEGIN USART2_MspInit 0 */
00331
00332   /* USER CODE END USART2_MspInit 0 */
00333     /* Peripheral clock enable */
00334     __HAL_RCC_USART2_CLK_ENABLE();
00335
00336     __HAL_RCC_GPIOA_CLK_ENABLE();
00341     GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
00342     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00343     GPIO_InitStruct.Pull = GPIO_NOPULL;
00344     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00345     GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
00346     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00347
00348     /* USART2 DMA Init */
00349     /* USART2_RX Init */
00350     hdma_usart2_rx.Instance = DMA1_Stream5;
00351     hdma_usart2_rx.Init.Channel = DMA_CHANNEL_4;
00352     hdma_usart2_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
00353     hdma_usart2_rx.Init.PeriphInc = DMA_PINC_DISABLE;
00354     hdma_usart2_rx.Init.MemInc = DMA_MINC_ENABLE;
00355     hdma_usart2_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00356     hdma_usart2_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00357     hdma_usart2_rx.Init.Mode = DMA_CIRCULAR;
00358     hdma_usart2_rx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00359     hdma_usart2_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00360     if (HAL_DMA_Init(&hdma_usart2_rx) != HAL_OK)
00361     {
00362       Error_Handler();
00363     }
00364
00365     __HAL_LINKDMA(huart,hdmarx,hdma_usart2_rx);
00366
00367     /* USART2_TX Init */
00368     hdma_usart2_tx.Instance = DMA1_Stream6;
00369     hdma_usart2_tx.Init.Channel = DMA_CHANNEL_4;
00370     hdma_usart2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
00371     hdma_usart2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
00372     hdma_usart2_tx.Init.MemInc = DMA_MINC_ENABLE;
00373     hdma_usart2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00374     hdma_usart2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00375     hdma_usart2_tx.Init.Mode = DMA_NORMAL;
00376     hdma_usart2_tx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00377     hdma_usart2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00378     if (HAL_DMA_Init(&hdma_usart2_tx) != HAL_OK)
00379     {
00380       Error_Handler();
00381     }
00382
00383     __HAL_LINKDMA(huart,hdmatx,hdma_usart2_tx);
00384
00385     /* USART2 interrupt Init */
00386     HAL_NVIC_SetPriority(USART2_IRQn, 5, 0);
00387     HAL_NVIC_EnableIRQ(USART2_IRQn);
00388   /* USER CODE BEGIN USART2_MspInit 1 */
00389
00390   /* USER CODE END USART2_MspInit 1 */
00391   }
00392
00393 }
00394
00395 /**********************************************************************
00396 * @brief UART MSP De-Initialization
00397 * This function freeze the hardware resources used in this example
00398 * @param huart: UART handle pointer
00399 * @retval None
00400 **********************************************************************/
00401 void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
00402 {
00403   if(huart->Instance==USART1)
00404   {
```

```
00405   /* USER CODE BEGIN USART1_MspDeInit 0 */
00406
00407   /* USER CODE END USART1_MspDeInit 0 */
00408     /* Peripheral clock disable */
00409     __HAL_RCC_USART1_CLK_DISABLE();
00410
00415     HAL_GPIO_DeInit(GPIOA, GPIO_PIN_9|GPIO_PIN_10);
00416
00417     /* USART1 DMA DeInit */
00418     HAL_DMA_DeInit(huart->hdmarx);
00419     HAL_DMA_DeInit(huart->hdmatx);
00420
00421     /* USART1 interrupt DeInit */
00422     HAL_NVIC_DisableIRQ(USART1_IRQn);
00423   /* USER CODE BEGIN USART1_MspDeInit 1 */
00424
00425   /* USER CODE END USART1_MspDeInit 1 */
00426   }
00427   else if(huart->Instance==USART2)
00428   {
00429   /* USER CODE BEGIN USART2_MspDeInit 0 */
00430
00431   /* USER CODE END USART2_MspDeInit 0 */
00432     /* Peripheral clock disable */
00433     __HAL_RCC_USART2_CLK_DISABLE();
00434
00439     HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
00440
00441     /* USART2 DMA DeInit */
00442     HAL_DMA_DeInit(huart->hdmarx);
00443     HAL_DMA_DeInit(huart->hdmatx);
00444
00445     /* USART2 interrupt DeInit */
00446     HAL_NVIC_DisableIRQ(USART2_IRQn);
00447   /* USER CODE BEGIN USART2_MspDeInit 1 */
00448
00449   /* USER CODE END USART2_MspDeInit 1 */
00450   }
00451
00452 }
00453
00454 /* USER CODE BEGIN 1 */
00455
00456 /* USER CODE END 1 */
```

5.37 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/stm32f4xx↩ _hal_timebase_tim.c File Reference

HAL time base based on the hardware TIM.

```
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal_tim.h"
```

Functions

- HAL_StatusTypeDef HAL_InitTick (uint32_t TickPriority)

  This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.
- void HAL_SuspendTick (void)

  Suspend Tick increment.
- void HAL_ResumeTick (void)

  Resume Tick increment.

Variables

- TIM_HandleTypeDef htim4

5.37.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

Definition in file stm32f4xx_hal_timebase_tim.c.

5.37.2 Function Documentation

5.37.2.1 HAL_InitTick()

```
HAL_StatusTypeDef HAL_InitTick (
            uint32_t TickPriority )
```

This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.

Note

This function is called automatically at the beginning of program after reset by HAL_Init() or at any time when clock is configured, by HAL_RCC_ClockConfig().

Parameters

TickPriority	Tick interrupt priority.

Return values

HAL	status

Definition at line 41 of file stm32f4xx_hal_timebase_tim.c.

5.37.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
            void  )
```

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM1 update interrupt.

Parameters

None	

Return values

None	

Definition at line 122 of file stm32f4xx_hal_timebase_tim.c.

5.37.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
             void  )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM1 update interrupt.

Parameters

None	

Return values

None	

Definition at line 110 of file stm32f4xx_hal_timebase_tim.c.

5.37.3 Variable Documentation

5.37.3.1 htim4

```
TIM_HandleTypeDef htim4
```

Definition at line 28 of file stm32f4xx_hal_timebase_tim.c.

5.38 stm32f4xx_hal_timebase_tim.c

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Includes ------------------------------------------------------------------*/
00021 #include "stm32f4xx_hal.h"
```

```
00022 #include "stm32f4xx_hal_tim.h"
00023
00024 /* Private typedef -----------------------------------------------------------*/
00025 /* Private define ------------------------------------------------------------*/
00026 /* Private macro -------------------------------------------------------------*/
00027 /* Private variables ---------------------------------------------------------*/
00028 TIM_HandleTypeDef        htim4;
00029 /* Private function prototypes -----------------------------------------------*/
00030 /* Private functions ---------------------------------------------------------*/
00031
00041 HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
00042 {
00043   RCC_ClkInitTypeDef    clkconfig;
00044   uint32_t              uwTimclock = 0U;
00045
00046   uint32_t              uwPrescalerValue = 0U;
00047   uint32_t              pFLatency;
00048   HAL_StatusTypeDef     status;
00049
00050   /* Enable TIM1 clock */
00051   __HAL_RCC_TIM4_CLK_ENABLE();
00052
00053   /* Get clock configuration */
00054   HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
00055
00056   /* Compute TIM1 clock */
00057   uwTimclock = 2*HAL_RCC_GetPCLK2Freq();
00058
00059   /* Compute the prescaler value to have TIM1 counter clock equal to 1MHz */
00060   uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000U) - 1U);
00061
00062   /* Initialize TIM1 */
00063   htim4.Instance = TIM4;
00064
00065   /* Initialize TIMx peripheral as follow:
00066   + Period = [(TIM1CLK/1000) - 1]. to have a (1/1000) s time base.
00067   + Prescaler = (uwTimclock/1000000 - 1) to have a 1MHz counter clock.
00068   + ClockDivision = 0
00069   + Counter direction = Up
00070   */
00071   htim4.Init.Period = (1000000U / 1000U) - 1U;
00072   htim4.Init.Prescaler = uwPrescalerValue;
00073   htim4.Init.ClockDivision = 0;
00074   htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
00075   htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
00076
00077   status = HAL_TIM_Base_Init(&htim4);
00078   if (status == HAL_OK)
00079   {
00080     /* Start the TIM time Base generation in interrupt mode */
00081     status = HAL_TIM_Base_Start_IT(&htim4);
00082     if (status == HAL_OK)
00083     {
00084     /* Enable the TIM1 global Interrupt */
00085         HAL_NVIC_EnableIRQ(TIM4_IRQn);
00086       /* Configure the SysTick IRQ priority */
00087       if (TickPriority < (1UL << __NVIC_PRIO_BITS))
00088       {
00089         /* Configure the TIM IRQ priority */
00090         HAL_NVIC_SetPriority(TIM4_IRQn, TickPriority, 0U);
00091         uwTickPrio = TickPriority;
00092       }
00093       else
00094       {
00095         status = HAL_ERROR;
00096       }
00097     }
00098   }
00099
00100  /* Return function status */
00101   return status;
00102 }
00103
00110 void HAL_SuspendTick(void)
00111 {
00112   /* Disable TIM1 update Interrupt */
00113   __HAL_TIM_DISABLE_IT(&htim4, TIM_IT_UPDATE);
00114 }
00115
00122 void HAL_ResumeTick(void)
00123 {
00124   /* Enable TIM1 Update interrupt */
00125   __HAL_TIM_ENABLE_IT(&htim4, TIM_IT_UPDATE);
00126 }
00127
```

5.39 stm32f4xx_it.c

```
00001
00002 #include "main.h"
00003 #include "stm32f4xx_it.h"
00004
00005 extern DMA_HandleTypeDef hdma_usart1_rx;
00006 extern DMA_HandleTypeDef hdma_usart1_tx;
00007 extern DMA_HandleTypeDef hdma_usart2_rx;
00008 extern DMA_HandleTypeDef hdma_usart2_tx;
00009 extern UART_HandleTypeDef huart1;
00010 extern UART_HandleTypeDef huart2;
00011 extern TIM_HandleTypeDef htim4;
00012 extern I2C_HandleTypeDef hi2c1;
00013
00014
00015 void NMI_Handler(void)
00016 {
00017   while (1)
00018   {
00019   }
00020 }
00021
00022 void HardFault_Handler(void)
00023 {
00024
00025   while (1)
00026   {
00027   }
00028 }
00029
00030 void MemManage_Handler(void)
00031 {
00032   while (1)
00033   {
00034   }
00035 }
00036
00037
00038 void BusFault_Handler(void)
00039 {
00040   while (1)
00041   {
00042   }
00043 }
00044
00048 void UsageFault_Handler(void)
00049 {
00050   while (1)
00051   {
00052   }
00053 }
00054
00055 void DebugMon_Handler(void)
00056 {
00057 }
00058
00059 /******************************************************************************/
00060 /* STM32F4xx Peripheral Interrupt Handlers                                    */
00061 /* Add here the Interrupt Handlers for the used peripherals.                  */
00062 /* For the available peripheral interrupt handler names,                      */
00063 /* please refer to the startup file (startup_stm32f4xx.s).                    */
00064 /******************************************************************************/
00065
00066 void DMA1_Stream5_IRQHandler(void)
00067 {
00068   HAL_DMA_IRQHandler(&hdma_usart2_rx);
00069 }
00070
00071
00072 void DMA1_Stream6_IRQHandler(void)
00073 {
00074   HAL_DMA_IRQHandler(&hdma_usart2_tx);
00075 }
00076
00077 /*void TIM1_UP_TIM10_IRQHandler(void)
00078 {
00079   HAL_TIM_IRQHandler(&htim1);
00080 }*/
00081
00082 void TIM4_IRQHandler(void)
00083 {
00084   HAL_TIM_IRQHandler(&htim4);
00085 }
00086
00087 void USART1_IRQHandler(void)
00088 {
```

```
00089   HAL_UART_IRQHandler(&huart1);
00090 }
00091
00092 void USART2_IRQHandler(void)
00093 {
00094   HAL_UART_IRQHandler(&huart2);
00095 }
00096
00097 void DMA2_Stream2_IRQHandler(void)
00098 {
00099   HAL_DMA_IRQHandler(&hdma_usart1_rx);
00100 }
00101
00102 void DMA2_Stream7_IRQHandler(void)
00103 {
00104   HAL_DMA_IRQHandler(&hdma_usart1_tx);
00105 }
00106
00107
00108
00109
00110 //=======================================================
00111 //      ENCODER INDEX LEFT
00112 //=======================================================
00113 void EXTI15_10_IRQHandler(void)
00114 {
00115   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
00116 }
00117 //=======================================================
00118 //      ENCODER INDEX RIGHT
00119 //=======================================================
00120
00121 void EXTI0_IRQHandler(void)
00122 {
00123   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
00124 }
00125
00126
00127 void I2C1_EV_IRQHandler(void)
00128 {
00129       HAL_I2C_EV_IRQHandler(&hi2c1);
00130 }
00131
00132 void I2C1_ER_IRQHandler(void)
00133 {
00134       HAL_I2C_ER_IRQHandler(&hi2c1);
00135 }
00136
```

5.40 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/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

5.40.1 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

Definition in file syscalls.c.

5.40.2 Function Documentation

5.40.2.1 __attribute__()

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

Definition at line 65 of file syscalls.c.

5.40.2.2 __io_getchar()

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

Definition at line 36 of file syscalls.c.

5.40.2.3 _close()

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

Definition at line 88 of file syscalls.c.

5.40.2.4 _execve()

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

Definition at line 151 of file syscalls.c.

5.40.2.5 _exit()

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

Definition at line 59 of file syscalls.c.

5.40.2.6 _fork()

```
int _fork (
            void  )
```

Definition at line 145 of file syscalls.c.

5.40.2.7 _fstat()

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

Definition at line 94 of file syscalls.c.

5.40.2.8 _getpid()

```
int _getpid (
            void  )
```

Definition at line 48 of file syscalls.c.

5.40.2.9 _isatty()

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

Definition at line 100 of file syscalls.c.

5.40.2.10 _kill()

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

Definition at line 53 of file syscalls.c.

5.40.2.11 _link()

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

Definition at line 139 of file syscalls.c.

5.40.2.12 _lseek()

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

Definition at line 105 of file syscalls.c.

5.40.2.13 _open()

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

Definition at line 110 of file syscalls.c.

5.40.2.14 _stat()

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

Definition at line 133 of file syscalls.c.

5.40.2.15 _times()

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

Definition at line 128 of file syscalls.c.

5.40.2.16 _unlink()

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

Definition at line 122 of file syscalls.c.

5.40.2.17 _wait()

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

Definition at line 116 of file syscalls.c.

5.40.2.18 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

Definition at line 44 of file syscalls.c.

5.40.3 Variable Documentation

5.40.3.1 environ

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

Definition at line 40 of file syscalls.c.

5.41 syscalls.c

Go to the documentation of this file.

```
00001
00023 /* Includes */
00024 #include <sys/stat.h>
00025 #include <stdlib.h>
00026 #include <errno.h>
00027 #include <stdio.h>
00028 #include <signal.h>
00029 #include <time.h>
00030 #include <sys/time.h>
00031 #include <sys/times.h>
00032
00033
00034 /* Variables */
00035 extern int __io_putchar(int ch) __attribute__((weak));
00036 extern int __io_getchar(void) __attribute__((weak));
00037
00038
00039 char *__env[1] = { 0 };
00040 char **environ = __env;
00041
00042
00043 /* Functions */
00044 void initialise_monitor_handles()
00045 {
00046 }
00047
00048 int _getpid(void)
00049 {
00050     return 1;
00051 }
00052
00053 int _kill(int pid, int sig)
00054 {
00055     errno = EINVAL;
00056     return -1;
00057 }
00058
00059 void _exit (int status)
00060 {
00061     _kill(status, -1);
00062     while (1) {}        /* Make sure we hang here */
00063 }
00064
00065 __attribute__((weak)) int _read(int file, char *ptr, int len)
00066 {
00067     int DataIdx;
00068
00069     for (DataIdx = 0; DataIdx < len; DataIdx++)
00070     {
00071         *ptr++ = __io_getchar();
00072     }
00073
00074 return len;
00075 }
00076
00077 __attribute__((weak)) int _write(int file, char *ptr, int len)
00078 {
00079     int DataIdx;
00080
00081     for (DataIdx = 0; DataIdx < len; DataIdx++)
00082     {
00083         __io_putchar(*ptr++);
00084     }
00085     return len;
00086 }
00087
00088 int _close(int file)
00089 {
00090     return -1;
00091 }
00092
00093
00094 int _fstat(int file, struct stat *st)
00095 {
00096     st->st_mode = S_IFCHR;
00097     return 0;
00098 }
00099
00100 int _isatty(int file)
00101 {
00102     return 1;
00103 }
```

```
00104
00105 int _lseek(int file, int ptr, int dir)
00106 {
00107     return 0;
00108 }
00109
00110 int _open(char *path, int flags, ...)
00111 {
00112     /* Pretend like we always fail */
00113     return -1;
00114 }
00115
00116 int _wait(int *status)
00117 {
00118     errno = ECHILD;
00119     return -1;
00120 }
00121
00122 int _unlink(char *name)
00123 {
00124     errno = ENOENT;
00125     return -1;
00126 }
00127
00128 int _times(struct tms *buf)
00129 {
00130     return -1;
00131 }
00132
00133 int _stat(char *file, struct stat *st)
00134 {
00135     st->st_mode = S_IFCHR;
00136     return 0;
00137 }
00138
00139 int _link(char *old, char *new)
00140 {
00141     errno = EMLINK;
00142     return -1;
00143 }
00144
00145 int _fork(void)
00146 {
00147     errno = EAGAIN;
00148     return -1;
00149 }
00150
00151 int _execve(char *name, char **argv, char **env)
00152 {
00153     errno = ENOMEM;
00154     return -1;
00155 }
```

5.42 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

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

Functions

- void * _sbrk (ptrdiff_t incr)

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

Variables

- static uint8_t * __sbrk_heap_end = NULL

5.42.1 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

Definition in file sysmem.c.

5.42.2 Function Documentation

5.42.2.1 _sbrk()

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

_sbrk() 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

Definition at line 53 of file sysmem.c.

5.42.3 Variable Documentation

5.42.3.1 __sbrk_heap_end

```
uint8_t* __sbrk_heap_end = NULL  [static]
```

Pointer to the current high watermark of the heap usage

Definition at line 30 of file sysmem.c.

5.43 sysmem.c

Go to the documentation of this file.

```
00001
00023 /* Includes */
00024 #include <errno.h>
00025 #include <stdint.h>
00026
00030 static uint8_t *__sbrk_heap_end = NULL;
00031
00053 void *_sbrk(ptrdiff_t incr)
00054 {
00055   extern uint8_t _end; /* Symbol defined in the linker script */
00056   extern uint8_t _estack; /* Symbol defined in the linker script */
00057   extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
00058   const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
00059   const uint8_t *max_heap = (uint8_t *)stack_limit;
00060   uint8_t *prev_heap_end;
00061
00062   /* Initialize heap end at first call */
00063   if (NULL == __sbrk_heap_end)
00064   {
00065     __sbrk_heap_end = &_end;
00066   }
00067
00068   /* Protect heap from growing into the reserved MSP stack */
00069   if (__sbrk_heap_end + incr > max_heap)
00070   {
00071     errno = ENOMEM;
00072     return (void *)-1;
00073   }
00074
00075   prev_heap_end = __sbrk_heap_end;
00076   __sbrk_heap_end += incr;
00077
00078   return (void *)prev_heap_end;
00079 }
```

5.44 systemclock.c

```
00001 /*
00002  * systemclock.c
00003  *
00004  *  Created on: Mar 13, 2023
00005  *      Author: kerhoas
00006  */
00007
00008
00009 #include "main.h"
00010
00011 void SystemClock_Config(void)
00012 {
00013   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
00014   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
00015
00018   __HAL_RCC_PWR_CLK_ENABLE();
00019   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
00020
00024   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
00025   RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
00026   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
00027   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
```

```
00028   RCC_OscInitStruct.PLL.PLLM = 8;
00029   RCC_OscInitStruct.PLL.PLLN = 432;
00030   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV6;
00031   RCC_OscInitStruct.PLL.PLLQ = 4;
00032   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
00033   {
00034     Error_Handler();
00035   }
00036
00039   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
00040                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
00041   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
00042   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
00043   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
00044   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;
00045
00046   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
00047   {
00048     Error_Handler();
00049   }
00050 }
```

5.45 util.c

```
00001 #include "util.h"
00002
00003 //================================================================
00004 void num2str(char *s, unsigned int number, unsigned int base, unsigned int size, int sp)
00005 {
00006         static char  hexChars[] = "0123456789ABCDEF";
00007
00008         char *p=s;
00009         unsigned int cnt;
00010         unsigned int i;
00011         char tmp;
00012
00013         // get digits
00014         do {
00015                 *s++=hexChars[number % base];
00016         } while (number /= base);
00017         *s='\0';
00018
00019         // reverse string
00020         cnt=s-p;
00021         for (i=0;i<cnt/2;i++) {
00022                 tmp=p[i]; p[i] = p[cnt-i-1]; p[cnt-i-1]=tmp;
00023         }
00024
00025         // add extra space
00026         if (cnt<size) {
00027                 for (i=cnt;i==0;i--)
00028                         {p[i+size-cnt]=p[i];}
00029                 if (sp) tmp=' '; else tmp='0';
00030                 for (i=0;i<size-cnt;i++) p[i]=tmp;
00031         }
00032 }
00033
00034 //================================================================
00035 unsigned int str2num(char *s, unsigned base)
00036 {
00037     unsigned int u=0, d;
00038     char ch=*s++;
00039     while (ch) {
00040         if ((ch>='0') && (ch<='9')) d=ch-'0';
00041         else if ((base==16) && (ch>='A') && (ch<='F')) d=ch-'A'+10;
00042         else if ((base==16) && (ch>='a') && (ch<='f')) d=ch-'a'+10;
00043         else break;
00044         u=d+base*u;
00045         ch=*s++;
00046     }
00047     return u;
00048 }
00049
00050 //================================================================
00051 void reverse(char *str, int len)
00052 {
00053     int i=0, j=len-1, temp;
00054     while (i<j)
00055     {
00056         temp = str[i];
00057         str[i] = str[j];
00058         str[j] = temp;
00059         i++; j--;
```

```
00060     }
00061 }
00062
00063 //=============================================================
00064 int intToStr(int x, char str[], int d)
00065 {
00066     int i = 0;
00067     while (x)
00068     {
00069         str[i++] = (x%10) + '0';
00070         x = x/10;
00071     }
00072
00073     // If number of digits required is more, then
00074     // add 0s at the beginning
00075     while (i < d)
00076         str[i++] = '0';
00077
00078     reverse(str, i);
00079     str[i] = '\0';
00080     return i;
00081 }
00082 //=============================================================
00083 void float2str( char *res, float n, int afterpoint)
00084 {
00085     // Extract integer part
00086     int ipart = (int)n;
00087
00088     // Extract floating part
00089     float fpart = n - (float)ipart;
00090
00091     // convert integer part to string
00092     int i = intToStr(ipart, res, 0);
00093
00094     // check for display option after point
00095     if (afterpoint != 0)
00096     {
00097         res[i] = '.';  // add dot
00098
00099         // Get the value of fraction part upto given no.
00100         // of points after dot. The third parameter is needed
00101         // to handle cases like 233.007
00102         fpart = fpart * (float)myPow(10.0, afterpoint);
00103
00104         intToStr((int)fpart, res + i + 1, afterpoint);
00105     }
00106 }
00107 //=============================================================
00108 double myPow(double x, int n) {
00109     unsigned int p = abs(n);
00110     double result = 1;
00111     while(p > 0)
00112     {
00113         if(p & 1) // if bit is set
00114         {
00115             result = result * x;
00116         }
00117         p = p » 1;
00118         x = x * x;
00119     }
00120
00121     if(n < 0)
00122     {
00123         return 1/result;
00124     }
00125     return result;
00126 }
00127
00128 //=============================================================
00129 void flush_ch(char* ch, int ch_size)
00130 {
00131     int i=0;
00132     for (i=0 ; i<ch_size ; i++)
00133     {
00134         ch[i]=0;
00135
00136     }
00137 }
00138 //=============================================================
00139
00140 int size_ch(char* ch, int ch_size_max)
00141 {
00142
00143     int i=0;
00144     for (i=0 ; i<ch_size_max ; i++)
00145     {
00146         if (ch[i]==0)
```

```
00147             break;
00148     }
00149
00150     return i;
00151 }
00152
00153 //================================================================
```

5.46 VL53L0X.c

```
00001
00002
00003 #include "main.h"
00004 #include <unistd.h>
00005     // Most of the functionality of this library is based on the VL53L0X API
00006 // provided by ST (STSW-IMG005), and some of the explanatory comments are quoted
00007 // or paraphrased from the API source code, API user manual (UM2039), and the
00008 // VL53L0X datasheet.
00009
00010 #include <stdint.h>
00011 #include "VL53L0X.h"
00012 #include "drv_i2c.h"
00013
00014 //------------------------------------------------------------
00015 // Local variables within this file (private)
00016 //------------------------------------------------------------
00017 uint8_t g_i2cAddr = ADDRESS_DEFAULT;
00018 uint8_t g_stopVariable; // read by init and used when starting measurement; is StopVariable field of
     VL53L0X_DevData_t structure in API
00019
00020
00021 //------------------------------------------------------------
00022 // Locally used functions (private)
00023 //------------------------------------------------------------
00024 uint8_t performSingleRefCalibration(uint8_t vhv_init_byte);
00025 //------------------------------------------------------------
00026 // I2C communication Functions
00027 //------------------------------------------------------------
00028 // Write an 8-bit register
00029 void writeReg(uint8_t reg, uint8_t value) {
00030     i2c1_WriteRegBuffer(0x53,reg,&value,1);
00031
00032 }
00033
00034 // Write a 16-bit register
00035 void writeReg16Bit(uint8_t reg, uint16_t value){
00036     uint8_t tab[2];
00037     tab[0]= ((value » 8));
00038     tab[1] = ((value ) & 0xFF);
00039     i2c1_WriteRegBuffer(0x53,reg,tab,2);
00040
00041 }
00042
00043 // Write a 32-bit register
00044 void writeReg32Bit(uint8_t reg, uint32_t value){
00045     uint8_t tab[4];
00046         tab[3]= ((value » 24) & 0xFF);
00047         tab[2]= ((value » 16) & 0xFF);
00048         tab[1]= ((value » 8) & 0xFF);
00049         tab[0] = ((value ) & 0xFF);
00050         i2c1_WriteRegBuffer(0x53,reg,tab,4);
00051 }
00052
00053 // Read an 8-bit register
00054 uint8_t readReg(uint8_t reg) {
00055     uint8_t value=0;
00056     i2c1_ReadRegBuffer(0x53,reg,&value,1);
00057     return value;
00058 }
00059
00060 // Read a 16-bit register
00061 uint16_t readReg16Bit(uint8_t reg) {
00062     uint8_t tab[2];
00063     i2c1_ReadRegBuffer(0x53,reg,tab,2);
00064     uint16_t value= ((uint16_t)tab[0] « 8) | (uint16_t)tab[1];
00065     return value;
00066 }
00067
00068 // Read a 32-bit register
00069 uint32_t readReg32Bit(uint8_t reg) {
00070   uint8_t tab[4];
00071   i2c1_ReadRegBuffer(0x53,reg,tab,4);
00072   uint32_t value= (tab[3] « 24) | (tab[2] « 16 ) | (tab[1] « 8) | tab[0];
```

```
00073   return value;
00074 }
00075
00076 // Write an arbitrary number of bytes from the given array to the sensor,
00077 // starting at the given register
00078 void writeMulti(uint8_t reg, uint8_t const *src, uint8_t count){
00079
00080   while ( count-- > 0 ) {
00081     i2c1_WriteRegBuffer(0x53,reg,(uint8_t *)src,1);
00082   }
00083 }
00084
00085
00086
00087 // Public Methods //////////////////////////////////////////////////////////////
00088
00089 void setAddress(uint8_t new_addr) {
00090   writeReg( I2C_SLAVE_DEVICE_ADDRESS, (new_addr»1) & 0x7F );
00091   g_i2cAddr = new_addr;
00092 }
00093
00094 uint8_t getAddress() {
00095   return g_i2cAddr;
00096 }
00097
00098 // Initialize sensor using sequence based on VL53L0X_DataInit(),
00099 // VL53L0X_StaticInit(), and VL53L0X_PerformRefCalibration().
00100 // This function does not perform reference SPAD calibration
00101 // (VL53L0X_PerformRefSpadManagement()), since the API user manual says that it
00102 // is performed by ST on the bare modules; it seems like that should work well
00103 // enough unless a cover glass is added.
00104 // If io_2v8 (optional) is true or not given, the sensor is configured for 2V8
00105 // mode.
00106 uint8_t initVL53L0X( ){
00107   // VL53L0X_DataInit() begin
00108
00109   // "Set I2C standard mode"
00110   writeReg(0x88, 0x00);
00111
00112   writeReg(0x80, 0x01);
00113   writeReg(0xFF, 0x01);
00114   writeReg(0x00, 0x00);
00115   g_stopVariable = readReg(0x91);
00116   writeReg(0x00, 0x01);
00117   writeReg(0xFF, 0x00);
00118   writeReg(0x80, 0x00);
00119
00120   // disable SIGNAL_RATE_MSRC (bit 1) and SIGNAL_RATE_PRE_RANGE (bit 4) limit checks
00121   writeReg(MSRC_CONFIG_CONTROL, readReg(MSRC_CONFIG_CONTROL) | 0x12);
00122
00123   // set final range signal rate limit to 0.25 MCPS (million counts per second)
00124   setSignalRateLimit(0.25);
00125
00126   writeReg(SYSTEM_SEQUENCE_CONFIG, 0xFF);
00127
00128   // VL53L0X_DataInit() end
00129
00130   // VL53L0X_StaticInit() begin
00131
00132   // The SPAD map (RefGoodSpadMap) is read by VL53L0X_get_info_from_device() in
00133   // the API, but the same data seems to be more easily readable from
00134   // GLOBAL_CONFIG_SPAD_ENABLES_REF_0 through _6, so read it from there
00135
00136   // -- VL53L0X_set_reference_spads() begin (assume NVM values are valid)
00137
00138   writeReg(0xFF, 0x01);
00139   writeReg(DYNAMIC_SPAD_REF_EN_START_OFFSET, 0x00);
00140   writeReg(DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD, 0x2C);
00141   writeReg(0xFF, 0x00);
00142   writeReg(GLOBAL_CONFIG_REF_EN_START_SELECT, 0xB4);
00143
00144
00145   // -- VL53L0X_set_reference_spads() end
00146
00147   // -- VL53L0X_load_tuning_settings() begin
00148   // DefaultTuningSettings from vl53l0x_tuning.h
00149
00150   writeReg(0xFF, 0x01);
00151   writeReg(0x00, 0x00);
00152
00153   writeReg(0xFF, 0x00);
00154   writeReg(0x09, 0x00);
00155   writeReg(0x10, 0x00);
00156   writeReg(0x11, 0x00);
00157
00158   writeReg(0x24, 0x01);
00159   writeReg(0x25, 0xFF);
```

```
00160   writeReg(0x75, 0x00);
00161
00162   writeReg(0xFF, 0x01);
00163   writeReg(0x4E, 0x2C);
00164   writeReg(0x48, 0x00);
00165   writeReg(0x30, 0x20);
00166
00167   writeReg(0xFF, 0x00);
00168   writeReg(0x30, 0x09);
00169   writeReg(0x54, 0x00);
00170   writeReg(0x31, 0x04);
00171   writeReg(0x32, 0x03);
00172   writeReg(0x40, 0x83);
00173   writeReg(0x46, 0x25);
00174   writeReg(0x60, 0x00);
00175   writeReg(0x27, 0x00);
00176   writeReg(0x50, 0x06);
00177   writeReg(0x51, 0x00);
00178   writeReg(0x52, 0x96);
00179   writeReg(0x56, 0x08);
00180   writeReg(0x57, 0x30);
00181   writeReg(0x61, 0x00);
00182   writeReg(0x62, 0x00);
00183   writeReg(0x64, 0x00);
00184   writeReg(0x65, 0x00);
00185   writeReg(0x66, 0xA0);
00186
00187   writeReg(0xFF, 0x01);
00188   writeReg(0x22, 0x32);
00189   writeReg(0x47, 0x14);
00190   writeReg(0x49, 0xFF);
00191   writeReg(0x4A, 0x00);
00192
00193   writeReg(0xFF, 0x00);
00194   writeReg(0x7A, 0x0A);
00195   writeReg(0x7B, 0x00);
00196   writeReg(0x78, 0x21);
00197
00198   writeReg(0xFF, 0x01);
00199   writeReg(0x23, 0x34);
00200   writeReg(0x42, 0x00);
00201   writeReg(0x44, 0xFF);
00202   writeReg(0x45, 0x26);
00203   writeReg(0x46, 0x05);
00204   writeReg(0x40, 0x40);
00205   writeReg(0x0E, 0x06);
00206   writeReg(0x20, 0x1A);
00207   writeReg(0x43, 0x40);
00208
00209   writeReg(0xFF, 0x00);
00210   writeReg(0x34, 0x03);
00211   writeReg(0x35, 0x44);
00212
00213   writeReg(0xFF, 0x01);
00214   writeReg(0x31, 0x04);
00215   writeReg(0x4B, 0x09);
00216   writeReg(0x4C, 0x05);
00217   writeReg(0x4D, 0x04);
00218
00219   writeReg(0xFF, 0x00);
00220   writeReg(0x44, 0x00);
00221   writeReg(0x45, 0x20);
00222   writeReg(0x47, 0x08);
00223   writeReg(0x48, 0x28);
00224   writeReg(0x67, 0x00);
00225   writeReg(0x70, 0x04);
00226   writeReg(0x71, 0x01);
00227   writeReg(0x72, 0xFE);
00228   writeReg(0x76, 0x00);
00229   writeReg(0x77, 0x00);
00230
00231   writeReg(0xFF, 0x01);
00232   writeReg(0x0D, 0x01);
00233
00234   writeReg(0xFF, 0x00);
00235   writeReg(0x80, 0x01);
00236   writeReg(0x01, 0xF8);
00237
00238   writeReg(0xFF, 0x01);
00239   writeReg(0x8E, 0x01);
00240   writeReg(0x00, 0x01);
00241   writeReg(0xFF, 0x00);
00242   writeReg(0x80, 0x00);
00243
00244   // -- VL53L0X_load_tuning_settings() end
00245
00246   // "Set interrupt config to new sample ready"
```

```
00247   // -- VL53L0X_SetGpioConfig() begin
00248
00249   writeReg(SYSTEM_INTERRUPT_CONFIG_GPIO, 0x04);
00250   writeReg(GPIO_HV_MUX_ACTIVE_HIGH, readReg(GPIO_HV_MUX_ACTIVE_HIGH) & ~0x10); // active low
00251   writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);
00252
00253   // -- VL53L0X_SetGpioConfig() end
00254
00255
00256
00257   // "Disable MSRC and TCC by default"
00258   // MSRC = Minimum Signal Rate Check
00259   // TCC = Target CentreCheck
00260   // -- VL53L0X_SetSequenceStepEnable() begin
00261
00262   writeReg(SYSTEM_SEQUENCE_CONFIG, 0xE8);
00263
00264   // -- VL53L0X_SetSequenceStepEnable() end
00265
00266
00267
00268   // VL53L0X_StaticInit() end
00269
00270   // VL53L0X_PerformRefCalibration() begin (VL53L0X_perform_ref_calibration())
00271
00272   // -- VL53L0X_perform_vhv_calibration() begin
00273
00274   writeReg(SYSTEM_SEQUENCE_CONFIG, 0x01);
00275   if (performSingleRefCalibration(0x40)) { return 1; }
00276
00277   // -- VL53L0X_perform_vhv_calibration() end
00278
00279   // -- VL53L0X_perform_phase_calibration() begin
00280
00281   writeReg(SYSTEM_SEQUENCE_CONFIG, 0x02);
00282   if (performSingleRefCalibration(0x00)) { return 1; }
00283
00284   // -- VL53L0X_perform_phase_calibration() end
00285
00286   // "restore the previous Sequence Config"
00287   writeReg(SYSTEM_SEQUENCE_CONFIG, 0xE8);
00288
00289   // VL53L0X_PerformRefCalibration() end
00290
00291   return 0;
00292 }
00293
00294 // Set the return signal rate limit check value in units of MCPS (mega counts
00295 // per second). "This represents the amplitude of the signal reflected from the
00296 // target and detected by the device"; setting this limit presumably determines
00297 // the minimum measurement necessary for the sensor to report a valid reading.
00298 // Setting a lower limit increases the potential range of the sensor but also
00299 // seems to increase the likelihood of getting an inaccurate reading because of
00300 // unwanted reflections from objects other than the intended target.
00301 // Defaults to 0.25 MCPS as initialized by the ST API and this library.
00302 uint8_t setSignalRateLimit(float limit_Mcps)
00303 {
00304   if (limit_Mcps < 0 || limit_Mcps > 511.99) { return false; }
00305
00306   // Q9.7 fixed point format (9 integer bits, 7 fractional bits)
00307   writeReg16Bit(FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT, limit_Mcps * (1 << 7));
00308   return 0;
00309 }
00310
00311 // Get the return signal rate limit check value in MCPS
00312 float getSignalRateLimit(void)
00313 {
00314   return (float)readReg16Bit(FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT) / (1 << 7);
00315 }
00316
00317
00318
00319
00320
00321 // Performs a single-shot range measurement and returns the reading in
00322 // millimeters
00323 // based on VL53L0X_PerformSingleRangingMeasurement()
00324 // extraStats provides additional info for this measurment. Set to 0 if not needed.
00325 uint16_t readRangeSingleMillimeters( /*statInfo_t *extraStats */) {
00326   writeReg(0x80, 0x01);
00327   writeReg(0xFF, 0x01);
00328   writeReg(0x00, 0x00);
00329   writeReg(0x91, g_stopVariable);
00330   writeReg(0x00, 0x01);
00331   writeReg(0xFF, 0x00);
00332   writeReg(0x80, 0x00);
00333   writeReg(SYSRANGE_START, 0x01);
```

```
00334
00335   uint16_t temp;
00336
00337     // assumptions: Linearity Corrective Gain is 1000 (default);
00338     // fractional ranging is not enabled
00339       temp = readReg16Bit(RESULT_RANGE_STATUS + 10);
00340
00341       temp+=0;
00342
00343   writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);
00344   return temp;
00345 }
00346
00347
00348 // based on VL53L0X_perform_single_ref_calibration()
00349 uint8_t performSingleRefCalibration(uint8_t vhv_init_byte)
00350 {
00351   writeReg(SYSRANGE_START, 0x01 | vhv_init_byte); // VL53L0X_REG_SYSRANGE_MODE_START_STOP
00352
00353
00354
00355   writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);
00356
00357   writeReg(SYSRANGE_START, 0x00);
00358
00359   return 0;
00360 }
```

Index