RobotROS

Generated by Doxygen 1.9.8

Chapter 1

Robot ROS

Documentation about the pinout of the robot

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 111 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 113 of file main.c.

4.1.2.2 data

```
int data
```

Represent the mode of the robot

Definition at line 114 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 120 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 122 of file main.c.

4.2.2.2 mode

```
int mode
```

Represent the mode of the robot

Definition at line 123 of file main.c.

4.2.2.3 speed

```
int speed
```

Represent the speed of the robot

Definition at line 124 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 130 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 132 of file main.c.

4.3.2.2 mode

```
int mode
```

Represent the mode send by the IHM

Definition at line 135 of file main.c.

4.3.2.3 speed

```
int speed
```

Represent the speed send by the IHM

Definition at line 136 of file main.c.

4.3.2.4 x

```
int x
```

Represent the x position send by the camera

Definition at line 133 of file main.c.

4.3.2.5 y

```
int y
```

Represent the y position send by the camera

Definition at line 134 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 301 of file VL53L0X.h.

4.4.2 Field Documentation

4.4.2.1 dss

```
uint8_t dss
```

Definition at line 302 of file VL53L0X.h.

4.4.2.2 final_range

```
uint8_t final_range
```

Definition at line 302 of file VL53L0X.h.

4.4.2.3 msrc

```
uint8_t msrc
```

Definition at line 302 of file VL53L0X.h.

4.4.2.4 pre_range

```
uint8_t pre_range
```

Definition at line 302 of file VL53L0X.h.

4.4.2.5 tcc

```
uint8_t tcc
```

Definition at line 302 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 305 of file VL53L0X.h.

4.5.2 Field Documentation

4.5.2.1 final_range_mclks

```
uint16_t final_range_mclks
```

Definition at line 308 of file VL53L0X.h.

4.5.2.2 final_range_us

```
uint32_t final_range_us
```

Definition at line 309 of file VL53L0X.h.

4.5.2.3 final_range_vcsel_period_pclks

```
uint16_t final_range_vcsel_period_pclks
```

Definition at line 306 of file VL53L0X.h.

4.5.2.4 msrc_dss_tcc_mclks

```
uint16_t msrc_dss_tcc_mclks
```

Definition at line 308 of file VL53L0X.h.

4.5.2.5 msrc_dss_tcc_us

```
uint32_t msrc_dss_tcc_us
```

Definition at line 309 of file VL53L0X.h.

4.5.2.6 pre_range_mclks

```
uint16_t pre_range_mclks
```

Definition at line 308 of file VL53L0X.h.

4.5.2.7 pre_range_us

```
uint32_t pre_range_us
```

Definition at line 309 of file VL53L0X.h.

4.5.2.8 pre_range_vcsel_period_pclks

```
uint16_t pre_range_vcsel_period_pclks
```

Definition at line 306 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 203 of file VL53L0X.h.

4.6.2 Field Documentation

4.6.2.1 ambientCnt

```
uint16_t ambientCnt
```

Definition at line 206 of file VL53L0X.h.

4.6.2.2 rangeStatus

```
uint8_t rangeStatus
```

Definition at line 208 of file VL53L0X.h.

4.6.2.3 rawDistance

```
uint16_t rawDistance
```

Definition at line 204 of file VL53L0X.h.

4.6.2.4 signalCnt

```
uint16_t signalCnt
```

Definition at line 205 of file VL53L0X.h.

4.6.2.5 spadCnt

```
uint16_t spadCnt
```

Definition at line 207 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 218 of file main.h.

5.8.2.2 B1_Pin

```
#define B1_Pin GPIO_PIN_13
```

Definition at line 217 of file main.h.

5.8.2.3 LD2_GPIO_Port

```
#define LD2_GPIO_Port GPIOA
```

Definition at line 224 of file main.h.

5.8.2.4 LD2_Pin

```
#define LD2_Pin GPIO_PIN_5
```

Definition at line 223 of file main.h.

5.8.2.5 SWO_GPIO_Port

```
#define SWO_GPIO_Port GPIOB
```

Definition at line 230 of file main.h.

5.8.2.6 SWO_Pin

```
#define SWO_Pin GPIO_PIN_3
```

Definition at line 229 of file main.h.

5.8.2.7 TCK_GPIO_Port

```
#define TCK_GPIO_Port GPIOA
```

Definition at line 228 of file main.h.

5.8.2.8 TCK_Pin

```
#define TCK_Pin GPIO_PIN_14
```

Definition at line 227 of file main.h.

5.8.2.9 TMS_GPIO_Port

```
#define TMS_GPIO_Port GPIOA
```

Definition at line 226 of file main.h.

5.8.2.10 TMS_Pin

```
#define TMS_Pin GPIO_PIN_13
```

Definition at line 225 of file main.h.

5.8.2.11 USART_RX_GPIO_Port

```
#define USART_RX_GPIO_Port GPIOA
```

Definition at line 222 of file main.h.

5.8.2.12 USART_RX_Pin

```
#define USART_RX_Pin GPIO_PIN_3
```

Definition at line 221 of file main.h.

5.8.2.13 USART_TX_GPIO_Port

```
#define USART_TX_GPIO_Port GPIOA
```

Definition at line 220 of file main.h.

5.8.2.14 USART_TX_Pin

```
#define USART_TX_Pin GPIO_PIN_2
```

Definition at line 219 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 154 of file main.c.

5.8.3.2 Error_Handler()

```
void Error_Handler (
             void  )
```

Definition at line 914 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 752 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 168 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 156 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	

Definition at line 463 of file main.c.

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 385 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 411 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 498 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	

Definition at line 438 of file main.c.

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 868 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 845 of file main.c.

5.8.3.13 test_vl53()

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

Use to test the VL53 sensor

Definition at line 857 of file main.c.

5.9 main.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00046 /* USER CODE END Header */
00047
00048 /* Define to prevent recursive inclusion -------------------------------------*/
00049 #ifndef __MAIN_H
00050 #define __MAIN_H
00051
00052 #ifdef __cplusplus
00053 extern "C" {
00054 #endif
00055
00056 /* Includes ------------------------------------------------------------------*/
00057 #include "stm32f4xx_hal.h"
00058
00059 #include <stdio.h>
00060 #include <rcl/rcl.h>
00061 #include <rcl/error_handling.h>
00062 #include <rclc/rclc.h>
00063 #include <rclc/executor.h>
00064 #include <uxr/client/transport.h>
00065 #include <rmw_microxrcedds_c/config.h>
00066 #include <rmw_microros/rmw_microros.h>
00067
00068 #include <std_msgs/msg/int32.h>
00069 #include <std_msgs/msg/string.h>
00070 #include <std_msgs/msg/header.h>
00071
00072 #include "FreeRTOS.h"
00073 #include "task.h"
00074 #include "queue.h"
00075 #include "semphr.h"
00076
00077 #include "systemclock.h"
00078 #include "drv_uart.h"
00079 #include "drv_gpio.h"
00080 #include "drv_i2c.h"
00081 #include "cmsis_os.h"
00082
00083 #include "microROS.h" //Custom microRos utils
00084 #include "retarget.h" //To redirect printf and scanf on UART2
00085
00086 /* Private includes ----------------------------------------------------------*/
00087 /* USER CODE BEGIN Includes */
00088
00089 /* USER CODE END Includes */
00090
00091 /* Exported types ------------------------------------------------------------*/
00092 /* USER CODE BEGIN ET */
00093
00094 /* USER CODE END ET */
00095
00096 /* Exported constants --------------------------------------------------------*/
00097 /* USER CODE BEGIN EC */
00098
00099 /* USER CODE END EC */
00100
00101 /* Exported macro ------------------------------------------------------------*/
00102 /* USER CODE BEGIN EM */
00103
00104 /* USER CODE END EM */
00105
00106 /* Exported functions prototypes ---------------------------------------------*/
00107 void Error_Handler(void);
00108
00109 /* USER CODE BEGIN EFP */
```

```
00114 void CHECKMRRET(rcl_ret_t ret, char* msg);
00115
00120 void SubscriberCallbackFunction(const void *msgin);
00121
00140 void microros_task(void *argument);
00141
00146 void task_Motor_Left(void *pvParameters);
00147
00152 void task_Motor_Right(void *pvParameters);
00153
00158 void task_VL53(void *pvParameters);
00159
00168 void task_Grove_LCD(void *pvParameters);
00169
00184 void task_Supervision(void *pvParameters);
00185
00188 void test_uart2(void *pvParameters);
00190 void test_vl53(void *pvParameters);
00192 void test_motor(void *pvParameters);
00212 int main(void);
00213
00214 /* USER CODE END EFP */
00215
00216 /* Private defines -----------------------------------------------------------*/
00217 #define B1_Pin GPIO_PIN_13
00218 #define B1_GPIO_Port GPIOC
00219 #define USART_TX_Pin GPIO_PIN_2
00220 #define USART_TX_GPIO_Port GPIOA
00221 #define USART_RX_Pin GPIO_PIN_3
00222 #define USART_RX_GPIO_Port GPIOA
00223 #define LD2_Pin GPIO_PIN_5
00224 #define LD2_GPIO_Port GPIOA
00225 #define TMS_Pin GPIO_PIN_13
00226 #define TMS_GPIO_Port GPIOA
00227 #define TCK_Pin GPIO_PIN_14
00228 #define TCK_GPIO_Port GPIOA
00229 #define SWO_Pin GPIO_PIN_3
00230 #define SWO_GPIO_Port GPIOB
00231 /* USER CODE BEGIN Private defines */
00232
00233 /* USER CODE END Private defines */
00234
00235 #ifdef __cplusplus
00236 }
00237 #endif
00238
00239 #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 custom 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 custom 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 10 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 29 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 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/retarget.h File Reference

: Contain function to add printf and scanf function use the UART2 All credit to Carmine Noviello for this code https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030↩ R8/system/src/retarget/

```
#include "stm32f4xx_hal.h"
#include <sys/stat.h>
```

Functions

- void RetargetInit (UART_HandleTypeDef *huart)
- int _isatty (int fd)
- int _write (int fd, char *ptr, int len)
- int _close (int fd)
- int _lseek (int fd, int ptr, int dir)
- int _read (int fd, char *ptr, int len)
- int _fstat (int fd, struct stat *st)
- int _getpid (void)
- int _kill (int pid, int sig)

5.14.1 Detailed Description

: Contain function to add printf and scanf function use the UART2 All credit to Carmine Noviello for this code https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/src/retarget/

Definition in file retarget.h.

5.14.2 Macro Definition Documentation

5.14.2.1 _RETARGET_H__

```
#define _RETARGET_H__
```

Definition at line 12 of file retarget.h.

5.14.3 Function Documentation

5.14.3.1 _close()

```
int _close (
            int fd )
```

Definition at line 57 of file retarget.c.

5.14.3.2 _fstat()

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

Definition at line 88 of file retarget.c.

5.14.3.3 _getpid()

```
int _getpid (
             void  )
```

Definition at line 98 of file retarget.c.

5.14.3.4 _isatty()

```
int _isatty (
             int fd )
```

Definition at line 35 of file retarget.c.

5.14.3.5 _kill()

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

Definition at line 103 of file retarget.c.

5.14.3.6 _lseek()

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

Definition at line 65 of file retarget.c.

5.14.3.7 _read()

```
int _read (
             int fd,
             char * ptr,
             int len )
```

Definition at line 74 of file retarget.c.

5.14.3.8 _write()

```
int _write (
             int fd,
             char * ptr,
             int len )
```

Definition at line 43 of file retarget.c.

5.14.3.9 RetargetInit()

```
void RetargetInit (
             UART_HandleTypeDef * huart )
```

Reconfigure stdin, stdout and stderr to use UART

Parameters

huart	Structure wich represent an UART

Definition at line 27 of file retarget.c.

5.15 retarget.h

Go to the documentation of this file.

```
00001
00008 #ifndef INC_RETARGET_H_
00009 #define INC_RETARGET_H_
00010
00011 #ifndef _RETARGET_H__
00012 #define _RETARGET_H__
00013
00014 #include "stm32f4xx_hal.h"
00015 #include <sys/stat.h>
00016
00021 void RetargetInit(UART_HandleTypeDef *huart);
00022 int _isatty(int fd);
00023 int _write(int fd, char* ptr, int len);
00024 int _close(int fd);
00025 int _lseek(int fd, int ptr, int dir);
00026 int _read(int fd, char* ptr, int len);
00027 int _fstat(int fd, struct stat* st);
00028 int _getpid(void);
00029 int _kill(int pid, int sig);
00030
00031 #endif //#ifndef _RETARGET_H__
00032
00033 #endif /* INC_RETARGET_H_ */
```

5.16 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 */
```

5.17 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)

5.17.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

Definition in file stm32f4xx_it.h.

5.17.2 Function Documentation

5.17.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

Definition at line 38 of file stm32f4xx_it.c.

5.17.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

Definition at line 55 of file stm32f4xx_it.c.

5.17.2.3 DMA1_Stream5_IRQHandler()

```
void DMA1_Stream5_IRQHandler (
             void  )
```

Definition at line 66 of file stm32f4xx_it.c.

5.17.2.4 DMA1_Stream6_IRQHandler()

```
void DMA1_Stream6_IRQHandler (
             void  )
```

Definition at line 72 of file stm32f4xx_it.c.

5.17.2.5 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
            void  )
```

Definition at line 97 of file stm32f4xx_it.c.

5.17.2.6 DMA2_Stream7_IRQHandler()

```
void DMA2_Stream7_IRQHandler (
            void  )
```

Definition at line 102 of file stm32f4xx_it.c.

5.17.2.7 HardFault_Handler()

```
void HardFault_Handler (
            void  )
```

Definition at line 22 of file stm32f4xx_it.c.

5.17.2.8 MemManage_Handler()

```
void MemManage_Handler (
            void  )
```

Definition at line 30 of file stm32f4xx_it.c.

5.17.2.9 NMI_Handler()

```
void NMI_Handler (
            void  )
```

Definition at line 15 of file stm32f4xx_it.c.

5.17.2.10 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

Definition at line 48 of file stm32f4xx_it.c.

5.17.2.11 USART1_IRQHandler()

```
void USART1_IRQHandler (
            void  )
```

Definition at line 87 of file stm32f4xx_it.c.

5.17.2.12 USART2_IRQHandler()

```
void USART2_IRQHandler (
            void  )
```

Definition at line 92 of file stm32f4xx_it.c.

5.18 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 */
```

5.19 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_ */
```

5.20 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_ */
```

5.21 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Inc/VL53L0X.h File Reference

: VL53L0X API STSW-IMG005 portage Most of the functionality of this library is based on the VL53L0X API provided by ST (STSW-IMG005), and some of the explanatory comments are quoted or paraphrased from the API source code, API user manual (UM2039), and the VL53L0X datasheet. I2C protocole for VLX530X

Data Structures

- struct statInfo_t
- struct SequenceStepEnables
- struct SequenceStepTimeouts

Macros

- #define true 1
- #define false 0
- #define SYSRANGE_START 0x00
- #define SYSTEM_THRESH_HIGH 0x0C
- #define SYSTEM_THRESH_LOW 0x0E
- #define SYSTEM_SEQUENCE_CONFIG 0x01

- #define SYSTEM_RANGE_CONFIG 0x09
- #define SYSTEM_INTERMEASUREMENT_PERIOD 0x04
- #define SYSTEM_INTERRUPT_GPIO_CONFIG 0x0A
- #define GPIO_HV_MUX_ACTIVE_HIGH 0x84
- #define SYSTEM_INTERRUPT_CLEAR 0x0B
- #define I2C_MODE 0x88
- #define RESULT_INTERRUPT_STATUS 0x13
- #define RESULT_RANGE_STATUS 0x14
- #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
- #define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
- #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
- #define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
- #define RESULT_PEAK_SIGNAL_RATE_REF 0xB6
- #define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28
- #define MSRC_CONFIG_CONTROL 0x60
- #define PRE_RANGE_CONFIG_MIN_SNR 0x27
- #define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
- #define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
- #define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
- #define FINAL_RANGE_CONFIG_MIN_SNR 0x67
- #define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
- #define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
- #define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44
- #define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
- #define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
- #define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
- #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
- #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52
- #define INTERNAL_TUNING_1 0x91
- #define INTERNAL_TUNING_2 0xFF
- #define SYSTEM_HISTOGRAM_BIN 0x81
- #define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
- #define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
- #define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
- #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
- #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
- #define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
- #define MSRC_CONFIG_TIMEOUT_MACROP 0x46
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF0 0x0B0
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF1 0x0B1
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF2 0x0B2
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF3 0x0B3
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF4 0x0B4
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF5 0x0B5
- #define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
- #define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
- #define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
- #define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
- #define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
- #define ALGO_PHASECAL_LIM 0x30
- #define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30
- #define SYSTEM_THRESH_HIGH 0x0C
- #define SYSTEM_THRESH_LOW 0x0E
- #define SYSTEM_SEQUENCE_CONFIG 0x01
- #define SYSTEM_RANGE_CONFIG 0x09

- #define SYSTEM_INTERMEASUREMENT_PERIOD 0x04
- #define SYSTEM_INTERRUPT_CONFIG_GPIO 0x0A
- #define GPIO_HV_MUX_ACTIVE_HIGH 0x84
- #define SYSTEM_INTERRUPT_CLEAR 0x0B
- #define RESULT_INTERRUPT_STATUS 0x13
- #define RESULT_RANGE_STATUS 0x14
- #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
- #define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
- #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
- #define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
- #define RESULT_PEAK_SIGNAL_RATE_REF 0xB6
- #define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28
- #define I2C_SLAVE_DEVICE_ADDRESS 0x8A
- #define MSRC_CONFIG_CONTROL 0x60
- #define PRE_RANGE_CONFIG_MIN_SNR 0x27
- #define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
- #define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
- #define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
- #define FINAL_RANGE_CONFIG_MIN_SNR 0x67
- #define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
- #define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
- #define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44
- #define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
- #define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
- #define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
- #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
- #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52
- #define SYSTEM_HISTOGRAM_BIN 0x81
- #define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
- #define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
- #define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
- #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
- #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
- #define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
- #define MSRC_CONFIG_TIMEOUT_MACROP 0x46
- #define SOFT_RESET_GO2_SOFT_RESET_N 0xBF
- #define IDENTIFICATION_MODEL_ID 0xC0
- #define IDENTIFICATION_REVISION_ID 0xC2
- #define OSC_CALIBRATE_VAL 0xF8
- #define GLOBAL_CONFIG_VCSEL_WIDTH 0x32
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF_0 0xB0
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF_1 0xB1
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF_2 0xB2
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF_3 0xB3
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF_4 0xB4
- #define GLOBAL_CONFIG_SPAD_ENABLES_REF_5 0xB5
- #define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
- #define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
- #define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
- #define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
- #define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
- #define ALGO_PHASECAL_LIM 0x30
- #define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30
- #define ADDRESS_DEFAULT 0b01010010
- #define ADDRESS_DEFAULT2 0b00101001

- #define startTimeout() (g_timeoutStartMs = millis())
- #define checkTimeoutExpired() (g_ioTimeout > 0 && ((uint16_t)millis() - g_timeoutStartMs) > g_ioTimeout)
- #define decodeVcselPeriod(reg_val) (((reg_val) + 1) << 1)
- #define encodeVcselPeriod(period_pclks) (((period_pclks) >> 1) - 1)
- #define calcMacroPeriod(vcsel_period_pclks) ((((uint32_t)2304 * (vcsel_period_pclks) * 1655) + 500) / 1000)

Enumerations

- enum **vcselPeriodType** { **VcselPeriodPreRange** , **VcselPeriodFinalRange** }

Functions

- void setAddress (uint8_t new_addr)
- uint8_t getAddress (void)
- uint8_t initVL53L0X ()
- uint8_t setSignalRateLimit (float limit_Mcps)
- float getSignalRateLimit (void)
- uint8_t **setMeasurementTimingBudget** (uint32_t budget_us)
- uint32_t **getMeasurementTimingBudget** (void)
- uint8_t **setVcselPulsePeriod** (vcselPeriodType type, uint8_t period_pclks)
- uint8_t **getVcselPulsePeriod** (vcselPeriodType type)
- void **startContinuous** (uint32_t period_ms)
- void **stopContinuous** (void)
- uint16_t **readRangeContinuousMillimeters** ()
- uint16_t readRangeSingleMillimeters ()
- void **setTimeout** (uint16_t timeout)
- uint16_t **getTimeout** (void)
- bool **timeoutOccurred** (void)
- void writeReg (uint8_t reg, uint8_t value)
- void writeReg16Bit (uint8_t reg, uint16_t value)
- void writeReg32Bit (uint8_t reg, uint32_t value)
- uint8_t readReg (uint8_t reg)
- uint16_t readReg16Bit (uint8_t reg)
- uint32_t readReg32Bit (uint8_t reg)
- void writeMulti (uint8_t reg, uint8_t const *src, uint8_t count)

5.21.1 Detailed Description

: VL53L0X API STSW-IMG005 portage Most of the functionality of this library is based on the VL53L0X API provided by ST (STSW-IMG005), and some of the explanatory comments are quoted or paraphrased from the API source code, API user manual (UM2039), and the VL53L0X datasheet. `I2C protocole for VLX530X`

Definition in file VL53L0X.h.

5.21.2 Macro Definition Documentation

5.21.2.1 ADDRESS_DEFAULT

```
#define ADDRESS_DEFAULT 0b01010010
```

Definition at line 175 of file VL53L0X.h.

5.21.2.2 ADDRESS_DEFAULT2

```
#define ADDRESS_DEFAULT2 0b00101001
```

Definition at line 176 of file VL53L0X.h.

5.21.2.3 ALGO_PART_TO_PART_RANGE_OFFSET_MM [1/2]

```
#define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28
```

Definition at line 40 of file VL53L0X.h.

5.21.2.4 ALGO_PART_TO_PART_RANGE_OFFSET_MM [2/2]

```
#define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28
```

Definition at line 40 of file VL53L0X.h.

5.21.2.5 ALGO_PHASECAL_CONFIG_TIMEOUT [1/2]

```
#define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30
```

Definition at line 86 of file VL53L0X.h.

5.21.2.6 ALGO_PHASECAL_CONFIG_TIMEOUT [2/2]

```
#define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30
```

Definition at line 86 of file VL53L0X.h.

5.21.2.7 ALGO_PHASECAL_LIM [1/2]

```
#define ALGO_PHASECAL_LIM 0x30
```

Definition at line 85 of file VL53L0X.h.

5.21.2.8 ALGO_PHASECAL_LIM [2/2]

```
#define ALGO_PHASECAL_LIM 0x30
```

Definition at line 85 of file VL53L0X.h.

5.21.2.9 calcMacroPeriod

```
#define calcMacroPeriod(
              vcsel_period_pclks ) ((((uint32_t)2304 * (vcsel_period_pclks) * 1655) + 500) /
1000)
```

Definition at line 195 of file VL53L0X.h.

5.21.2.10 checkTimeoutExpired

```
#define checkTimeoutExpired( ) (g_ioTimeout > 0 && ((uint16_t)millis() - g_timeoutStartMs) >
g_ioTimeout)
```

Definition at line 181 of file VL53L0X.h.

5.21.2.11 CROSSTALK_COMPENSATION_PEAK_RATE_MCPS [1/2]

```
#define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
```

Definition at line 72 of file VL53L0X.h.

5.21.2.12 CROSSTALK_COMPENSATION_PEAK_RATE_MCPS [2/2]

```
#define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
```

Definition at line 72 of file VL53L0X.h.

5.21.2.13 decodeVcselPeriod

```
#define decodeVcselPeriod(
              reg_val ) (((reg_val) + 1) << 1)
```

Definition at line 186 of file VL53L0X.h.

5.21.2.14 DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD [1/2]

```
#define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
```

Definition at line 81 of file VL53L0X.h.

5.21.2.15 DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD [2/2]

```
#define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
```

Definition at line 81 of file VL53L0X.h.

5.21.2.16 DYNAMIC_SPAD_REF_EN_START_OFFSET [1/2]

```
#define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
```

Definition at line 82 of file VL53L0X.h.

5.21.2.17 DYNAMIC_SPAD_REF_EN_START_OFFSET [2/2]

```
#define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
```

Definition at line 82 of file VL53L0X.h.

5.21.2.18 encodeVcselPeriod

```
#define encodeVcselPeriod(
             period_pclks ) (((period_pclks) >> 1) - 1)
```

Definition at line 190 of file VL53L0X.h.

5.21.2.19 false

```
#define false 0
```

Definition at line 13 of file VL53L0X.h.

5.21.2.20 FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT [1/2]

```
#define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44
```

Definition at line 51 of file VL53L0X.h.

5.21.2.21 FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT [2/2]

```
#define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44
```

Definition at line 51 of file VL53L0X.h.

5.21.2.22 FINAL_RANGE_CONFIG_MIN_SNR [1/2]

```
#define FINAL_RANGE_CONFIG_MIN_SNR 0x67
```

Definition at line 48 of file VL53L0X.h.

5.21.2.23 FINAL_RANGE_CONFIG_MIN_SNR [2/2]

```
#define FINAL_RANGE_CONFIG_MIN_SNR 0x67
```

Definition at line 48 of file VL53L0X.h.

5.21.2.24 FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI [1/2]

```
#define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
```

Definition at line 70 of file VL53L0X.h.

5.21.2.25 FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI [2/2]

```
#define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
```

Definition at line 70 of file VL53L0X.h.

5.21.2.26 FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO [1/2]

```
#define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
```

Definition at line 71 of file VL53L0X.h.

5.21.2.27 FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO [2/2]

```
#define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
```

Definition at line 71 of file VL53L0X.h.

5.21.2.28 FINAL_RANGE_CONFIG_VALID_PHASE_HIGH [1/2]

```
#define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
```

Definition at line 50 of file VL53L0X.h.

5.21.2.29 FINAL_RANGE_CONFIG_VALID_PHASE_HIGH [2/2]

```
#define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
```

Definition at line 50 of file VL53L0X.h.

5.21.2.30 FINAL_RANGE_CONFIG_VALID_PHASE_LOW [1/2]

```
#define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
```

Definition at line 49 of file VL53L0X.h.

5.21.2.31 FINAL_RANGE_CONFIG_VALID_PHASE_LOW [2/2]

```
#define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
```

Definition at line 49 of file VL53L0X.h.

5.21.2.32 FINAL_RANGE_CONFIG_VCSEL_PERIOD [1/2]

```
#define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
```

Definition at line 69 of file VL53L0X.h.

5.21.2.33 FINAL_RANGE_CONFIG_VCSEL_PERIOD [2/2]

```
#define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
```

Definition at line 69 of file VL53L0X.h.

5.21.2.34 GLOBAL_CONFIG_REF_EN_START_SELECT [1/2]

```
#define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
```

Definition at line 80 of file VL53L0X.h.

5.21.2.35 GLOBAL_CONFIG_REF_EN_START_SELECT [2/2]

```
#define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
```

Definition at line 80 of file VL53L0X.h.

5.21.2.36 GLOBAL_CONFIG_SPAD_ENABLES_REF0

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF0 0x0B0
```

Definition at line 74 of file VL53L0X.h.

5.21.2.37 GLOBAL_CONFIG_SPAD_ENABLES_REF1

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF1 0x0B1
```

Definition at line 75 of file VL53L0X.h.

5.21.2.38 GLOBAL_CONFIG_SPAD_ENABLES_REF2

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF2 0x0B2
```

Definition at line 76 of file VL53L0X.h.

5.21.2.39 GLOBAL_CONFIG_SPAD_ENABLES_REF3

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF3 0x0B3
```

Definition at line 77 of file VL53L0X.h.

5.21.2.40 GLOBAL_CONFIG_SPAD_ENABLES_REF4

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF4 0x0B4
```

Definition at line 78 of file VL53L0X.h.

5.21.2.41 GLOBAL_CONFIG_SPAD_ENABLES_REF5

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF5 0x0B5
```

Definition at line 79 of file VL53L0X.h.

5.21.2.42 GLOBAL_CONFIG_SPAD_ENABLES_REF_0

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF_0 0xB0
```

Definition at line 153 of file VL53L0X.h.

5.21.2.43 GLOBAL_CONFIG_SPAD_ENABLES_REF_1

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF_1 0xB1
```

Definition at line 154 of file VL53L0X.h.

5.21.2.44 GLOBAL_CONFIG_SPAD_ENABLES_REF_2

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF_2 0xB2
```

Definition at line 155 of file VL53L0X.h.

5.21.2.45 GLOBAL_CONFIG_SPAD_ENABLES_REF_3

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF_3 0xB3
```

Definition at line 156 of file VL53L0X.h.

5.21.2.46 GLOBAL_CONFIG_SPAD_ENABLES_REF_4

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF_4 0xB4
```

Definition at line 157 of file VL53L0X.h.

5.21.2.47 GLOBAL_CONFIG_SPAD_ENABLES_REF_5

```
#define GLOBAL_CONFIG_SPAD_ENABLES_REF_5 0xB5
```

Definition at line 158 of file VL53L0X.h.

5.21.2.48 GLOBAL_CONFIG_VCSEL_WIDTH

```
#define GLOBAL_CONFIG_VCSEL_WIDTH 0x32
```

Definition at line 152 of file VL53L0X.h.

5.21.2.49 GPIO_HV_MUX_ACTIVE_HIGH [1/2]

```
#define GPIO_HV_MUX_ACTIVE_HIGH 0x84
```

Definition at line 26 of file VL53L0X.h.

5.21.2.50 GPIO_HV_MUX_ACTIVE_HIGH [2/2]

```
#define GPIO_HV_MUX_ACTIVE_HIGH 0x84
```

Definition at line 26 of file VL53L0X.h.

5.21.2.51 HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT [1/2]

```
#define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
```

Definition at line 67 of file VL53L0X.h.

5.21.2.52 HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT [2/2]

```
#define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
```

Definition at line 67 of file VL53L0X.h.

5.21.2.53 HISTOGRAM_CONFIG_READOUT_CTRL [1/2]

```
#define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
```

Definition at line 68 of file VL53L0X.h.

5.21.2.54 HISTOGRAM_CONFIG_READOUT_CTRL [2/2]

```
#define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
```

Definition at line 68 of file VL53L0X.h.

5.21.2.55 I2C_MODE

```
#define I2C_MODE 0x88
```

Definition at line 28 of file VL53L0X.h.

5.21.2.56 I2C_SLAVE_DEVICE_ADDRESS

```
#define I2C_SLAVE_DEVICE_ADDRESS 0x8A
```

Definition at line 113 of file VL53L0X.h.

5.21.2.57 IDENTIFICATION_MODEL_ID

```
#define IDENTIFICATION_MODEL_ID 0xC0
```

Definition at line 147 of file VL53L0X.h.

5.21.2.58 IDENTIFICATION_REVISION_ID

```
#define IDENTIFICATION_REVISION_ID 0xC2
```

Definition at line 148 of file VL53L0X.h.

5.21.2.59 INTERNAL_TUNING_1

```
#define INTERNAL_TUNING_1 0x91
```

Definition at line 61 of file VL53L0X.h.

5.21.2.60 INTERNAL_TUNING_2

```
#define INTERNAL_TUNING_2 0xFF
```

Definition at line 62 of file VL53L0X.h.

5.21.2.61 MSRC_CONFIG_CONTROL [1/2]

```
#define MSRC_CONFIG_CONTROL 0x60
```

Definition at line 43 of file VL53L0X.h.

5.21.2.62 MSRC_CONFIG_CONTROL [2/2]

```
#define MSRC_CONFIG_CONTROL 0x60
```

Definition at line 43 of file VL53L0X.h.

5.21.2.63 MSRC_CONFIG_TIMEOUT_MACROP [1/2]

```
#define MSRC_CONFIG_TIMEOUT_MACROP 0x46
```

Definition at line 73 of file VL53L0X.h.

5.21.2.64 MSRC_CONFIG_TIMEOUT_MACROP [2/2]

```
#define MSRC_CONFIG_TIMEOUT_MACROP 0x46
```

Definition at line 73 of file VL53L0X.h.

5.21.2.65 OSC_CALIBRATE_VAL

```
#define OSC_CALIBRATE_VAL 0xF8
```

Definition at line 150 of file VL53L0X.h.

5.21.2.66 POWER_MANAGEMENT_GO1_POWER_FORCE [1/2]

```
#define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
```

Definition at line 83 of file VL53L0X.h.

5.21.2.67 POWER_MANAGEMENT_GO1_POWER_FORCE [2/2]

```
#define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
```

Definition at line 83 of file VL53L0X.h.

5.21.2.68 PRE_RANGE_CONFIG_MIN_SNR [1/2]

```
#define PRE_RANGE_CONFIG_MIN_SNR 0x27
```

Definition at line 44 of file VL53L0X.h.

5.21.2.69 PRE_RANGE_CONFIG_MIN_SNR [2/2]

```
#define PRE_RANGE_CONFIG_MIN_SNR 0x27
```

Definition at line 44 of file VL53L0X.h.

5.21.2.70 PRE_RANGE_CONFIG_SIGMA_THRESH_HI [1/2]

```
#define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
```

Definition at line 54 of file VL53L0X.h.

5.21.2.71 PRE_RANGE_CONFIG_SIGMA_THRESH_HI [2/2]

```
#define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
```

Definition at line 54 of file VL53L0X.h.

5.21.2.72 PRE_RANGE_CONFIG_SIGMA_THRESH_LO [1/2]

```
#define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
```

Definition at line 55 of file VL53L0X.h.

5.21.2.73 PRE_RANGE_CONFIG_SIGMA_THRESH_LO [2/2]

```
#define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
```

Definition at line 55 of file VL53L0X.h.

5.21.2.74 PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI [1/2]

```
#define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
```

Definition at line 57 of file VL53L0X.h.

5.21.2.75 PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI [2/2]

```
#define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
```

Definition at line 57 of file VL53L0X.h.

5.21.2.76 PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO [1/2]

```
#define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52
```

Definition at line 58 of file VL53L0X.h.

5.21.2.77 PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO [2/2]

```
#define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52
```

Definition at line 58 of file VL53L0X.h.

5.21.2.78 PRE_RANGE_CONFIG_VALID_PHASE_HIGH [1/2]

```
#define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
```

Definition at line 46 of file VL53L0X.h.

5.21.2.79 PRE_RANGE_CONFIG_VALID_PHASE_HIGH [2/2]

```
#define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
```

Definition at line 46 of file VL53L0X.h.

5.21.2.80 PRE_RANGE_CONFIG_VALID_PHASE_LOW [1/2]

```
#define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
```

Definition at line 45 of file VL53L0X.h.

5.21.2.81 PRE_RANGE_CONFIG_VALID_PHASE_LOW [2/2]

```
#define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
```

Definition at line 45 of file VL53L0X.h.

5.21.2.82 PRE_RANGE_CONFIG_VCSEL_PERIOD [1/2]

```
#define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
```

Definition at line 56 of file VL53L0X.h.

5.21.2.83 PRE_RANGE_CONFIG_VCSEL_PERIOD [2/2]

```
#define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
```

Definition at line 56 of file VL53L0X.h.

5.21.2.84 PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT [1/2]

```
#define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
```

Definition at line 47 of file VL53L0X.h.

5.21.2.85 PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT [2/2]

```
#define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
```

Definition at line 47 of file VL53L0X.h.

5.21.2.86 RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF [1/2]

```
#define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
```

Definition at line 35 of file VL53L0X.h.

5.21.2.87 RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF [2/2]

```
#define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
```

Definition at line 35 of file VL53L0X.h.

5.21.2.88 RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN [1/2]

```
#define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
```

Definition at line 33 of file VL53L0X.h.

5.21.2.89 RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN [2/2]

```
#define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
```

Definition at line 33 of file VL53L0X.h.

5.21.2.90 RESULT_CORE_RANGING_TOTAL_EVENTS_REF [1/2]

```
#define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
```

Definition at line 36 of file VL53L0X.h.

5.21.2.91 RESULT_CORE_RANGING_TOTAL_EVENTS_REF [2/2]

```
#define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
```

Definition at line 36 of file VL53L0X.h.

5.21.2.92 RESULT_CORE_RANGING_TOTAL_EVENTS_RTN [1/2]

```
#define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
```

Definition at line 34 of file VL53L0X.h.

5.21.2.93 RESULT_CORE_RANGING_TOTAL_EVENTS_RTN [2/2]

```
#define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
```

Definition at line 34 of file VL53L0X.h.

5.21.2.94 RESULT_INTERRUPT_STATUS [1/2]

```
#define RESULT_INTERRUPT_STATUS 0x13
```

Definition at line 31 of file VL53L0X.h.

5.21.2.95 RESULT_INTERRUPT_STATUS [2/2]

```
#define RESULT_INTERRUPT_STATUS 0x13
```

Definition at line 31 of file VL53L0X.h.

5.21.2.96 RESULT_PEAK_SIGNAL_RATE_REF [1/2]

```
#define RESULT_PEAK_SIGNAL_RATE_REF 0xB6
```

Definition at line 37 of file VL53L0X.h.

5.21.2.97 RESULT_PEAK_SIGNAL_RATE_REF [2/2]

```
#define RESULT_PEAK_SIGNAL_RATE_REF 0xB6
```

Definition at line 37 of file VL53L0X.h.

5.21.2.98 RESULT_RANGE_STATUS [1/2]

```
#define RESULT_RANGE_STATUS 0x14
```

Definition at line 32 of file VL53L0X.h.

5.21.2.99 RESULT_RANGE_STATUS [2/2]

```
#define RESULT_RANGE_STATUS 0x14
```

Definition at line 32 of file VL53L0X.h.

5.21.2.100 SOFT_RESET_GO2_SOFT_RESET_N

```
#define SOFT_RESET_GO2_SOFT_RESET_N 0xBF
```

Definition at line 146 of file VL53L0X.h.

5.21.2.101 startTimeout

```
#define startTimeout( ) (g_timeoutStartMs = millis())
```

Definition at line 178 of file VL53L0X.h.

5.21.2.102 SYSRANGE_START

```
#define SYSRANGE_START 0x00
```

Definition at line 15 of file VL53L0X.h.

5.21.2.103 SYSTEM_HISTOGRAM_BIN [1/2]

```
#define SYSTEM_HISTOGRAM_BIN 0x81
```

Definition at line 66 of file VL53L0X.h.

5.21.2.104 SYSTEM_HISTOGRAM_BIN [2/2]

```
#define SYSTEM_HISTOGRAM_BIN 0x81
```

Definition at line 66 of file VL53L0X.h.

5.21.2.105 SYSTEM_INTERMEASUREMENT_PERIOD [1/2]

```
#define SYSTEM_INTERMEASUREMENT_PERIOD 0x04
```

Definition at line 20 of file VL53L0X.h.

5.21.2.106 SYSTEM_INTERMEASUREMENT_PERIOD [2/2]

```
#define SYSTEM_INTERMEASUREMENT_PERIOD 0x04
```

Definition at line 20 of file VL53L0X.h.

5.21.2.107 SYSTEM_INTERRUPT_CLEAR [1/2]

```
#define SYSTEM_INTERRUPT_CLEAR 0x0B
```

Definition at line 27 of file VL53L0X.h.

5.21.2.108 SYSTEM_INTERRUPT_CLEAR [2/2]

```
#define SYSTEM_INTERRUPT_CLEAR 0x0B
```

Definition at line 27 of file VL53L0X.h.

5.21.2.109 SYSTEM_INTERRUPT_CONFIG_GPIO

```
#define SYSTEM_INTERRUPT_CONFIG_GPIO 0x0A
```

Definition at line 96 of file VL53L0X.h.

5.21.2.110 SYSTEM_INTERRUPT_GPIO_CONFIG

```
#define SYSTEM_INTERRUPT_GPIO_CONFIG 0x0A
```

Definition at line 23 of file VL53L0X.h.

5.21.2.111 SYSTEM_RANGE_CONFIG [1/2]

```
#define SYSTEM_RANGE_CONFIG 0x09
```

Definition at line 19 of file VL53L0X.h.

5.21.2.112 SYSTEM_RANGE_CONFIG [2/2]

```
#define SYSTEM_RANGE_CONFIG 0x09
```

Definition at line 19 of file VL53L0X.h.

5.21.2.113 SYSTEM_SEQUENCE_CONFIG [1/2]

```
#define SYSTEM_SEQUENCE_CONFIG 0x01
```

Definition at line 18 of file VL53L0X.h.

5.21.2.114 SYSTEM_SEQUENCE_CONFIG [2/2]

```
#define SYSTEM_SEQUENCE_CONFIG 0x01
```

Definition at line 18 of file VL53L0X.h.

5.21.2.115 SYSTEM_THRESH_HIGH [1/2]

```
#define SYSTEM_THRESH_HIGH 0x0C
```

Definition at line 16 of file VL53L0X.h.

5.21.2.116 SYSTEM_THRESH_HIGH [2/2]

```
#define SYSTEM_THRESH_HIGH 0x0C
```

Definition at line 16 of file VL53L0X.h.

5.21.2.117 SYSTEM_THRESH_LOW [1/2]

```
#define SYSTEM_THRESH_LOW 0x0E
```

Definition at line 17 of file VL53L0X.h.

5.21.2.118 SYSTEM_THRESH_LOW [2/2]

```
#define SYSTEM_THRESH_LOW 0x0E
```

Definition at line 17 of file VL53L0X.h.

5.21.2.119 true

```
#define true 1
```

Definition at line 12 of file VL53L0X.h.

5.21.2.120 VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV [1/2]

```
#define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
```

Definition at line 84 of file VL53L0X.h.

5.21.2.121 VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV [2/2]

```
#define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
```

Definition at line 84 of file VL53L0X.h.

5.21.3 Enumeration Type Documentation

5.21.3.1 vcselPeriodType

```
enum vcselPeriodType
```

Definition at line 200 of file VL53L0X.h.

5.21.4 Function Documentation

5.21.4.1 getAddress()

```
uint8_t getAddress (
            void  )
```

Definition at line 103 of file VL53L0X.c.

5.21.4.2 getSignalRateLimit()

```
float getSignalRateLimit (
            void  )
```

Definition at line 329 of file VL53L0X.c.

5.21.4.3 initVL53L0X()

```
uint8_t initVL53L0X ( )
```

Definition at line 115 of file VL53L0X.c.

5.21.4.4 readRangeSingleMillimeters()

```
uint16_t readRangeSingleMillimeters ( )
```

Definition at line 342 of file VL53L0X.c.

5.21.4.5 readReg()

```
uint8_t readReg (
            uint8_t reg )
```

Definition at line 62 of file VL53L0X.c.

5.21.4.6 readReg16Bit()

```
uint16_t readReg16Bit (
            uint8_t reg )
```

Definition at line 70 of file VL53L0X.c.

5.21.4.7 readReg32Bit()

```
uint32_t readReg32Bit (
            uint8_t reg )
```

Definition at line 79 of file VL53L0X.c.

5.21.4.8 setAddress()

```
void setAddress (
            uint8_t new_addr )
```

Definition at line 98 of file VL53L0X.c.

5.21.4.9 setSignalRateLimit()

```
uint8_t setSignalRateLimit (
            float limit_Mcps )
```

Definition at line 319 of file VL53L0X.c.

5.21.4.10 writeMulti()

```
void writeMulti (
            uint8_t reg,
            uint8_t const * src,
            uint8_t count )
```

Definition at line 89 of file VL53L0X.c.

5.21.4.11 writeReg()

```
void writeReg (
            uint8_t reg,
            uint8_t value )
```

Definition at line 39 of file VL53L0X.c.

5.21.4.12 writeReg16Bit()

```
void writeReg16Bit (
            uint8_t reg,
            uint16_t value )
```

Definition at line 44 of file VL53L0X.c.

5.21.4.13 writeReg32Bit()

```
void writeReg32Bit (
            uint8_t reg,
            uint32_t value )
```

Definition at line 52 of file VL53L0X.c.

5.22 VL53L0X.h

Go to the documentation of this file.

```
00001
00011 //#define bool  uint8_t
00012 #define true  1
00013 #define false 0
00014
00015 #define SYSRANGE_START 0x00
00016 #define SYSTEM_THRESH_HIGH 0x0C
00017 #define SYSTEM_THRESH_LOW 0x0E
00018 #define SYSTEM_SEQUENCE_CONFIG 0x01
00019 #define SYSTEM_RANGE_CONFIG 0x09
00020 #define SYSTEM_INTERMEASUREMENT_PERIOD 0x04
00021
00022
00023 #define SYSTEM_INTERRUPT_GPIO_CONFIG 0x0A
00024
00025 //GPIO Config
00026 #define GPIO_HV_MUX_ACTIVE_HIGH 0x84
00027 #define SYSTEM_INTERRUPT_CLEAR 0x0B
00028 #define I2C_MODE 0x88
00029
00030 // Result registers
00031 #define RESULT_INTERRUPT_STATUS 0x13
00032 #define RESULT_RANGE_STATUS 0x14
00033 #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN 0xBC
00034 #define RESULT_CORE_RANGING_TOTAL_EVENTS_RTN 0xC0
00035 #define RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF 0xD0
00036 #define RESULT_CORE_RANGING_TOTAL_EVENTS_REF 0xD4
00037 #define RESULT_PEAK_SIGNAL_RATE_REF 0xB6
00038
00039 //Algo Register
00040 #define ALGO_PART_TO_PART_RANGE_OFFSET_MM 0x28
00041
00042 //Check limit register
00043 #define MSRC_CONFIG_CONTROL 0x60
00044 #define PRE_RANGE_CONFIG_MIN_SNR 0x27
00045 #define PRE_RANGE_CONFIG_VALID_PHASE_LOW 0x56
00046 #define PRE_RANGE_CONFIG_VALID_PHASE_HIGH 0x57
```

```
00047 #define PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT 0x64
00048 #define FINAL_RANGE_CONFIG_MIN_SNR 0x67
00049 #define FINAL_RANGE_CONFIG_VALID_PHASE_LOW 0x47
00050 #define FINAL_RANGE_CONFIG_VALID_PHASE_HIGH 0x48
00051 #define FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT 0x44
00052
00053 // PRE RANGE registers
00054 #define PRE_RANGE_CONFIG_SIGMA_THRESH_HI 0x61
00055 #define PRE_RANGE_CONFIG_SIGMA_THRESH_LO 0x62
00056 #define PRE_RANGE_CONFIG_VCSEL_PERIOD 0x50
00057 #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x51
00058 #define PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x52
00059
00060 //Internal tuning registers
00061 #define INTERNAL_TUNING_1 0x91
00062 #define INTERNAL_TUNING_2 0xFF
00063
00064
00065 //Other registers
00066 #define SYSTEM_HISTOGRAM_BIN 0x81
00067 #define HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT 0x33
00068 #define HISTOGRAM_CONFIG_READOUT_CTRL 0x55
00069 #define FINAL_RANGE_CONFIG_VCSEL_PERIOD 0x70
00070 #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI 0x71
00071 #define FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO 0x72
00072 #define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS 0x20
00073 #define MSRC_CONFIG_TIMEOUT_MACROP 0x46
00074 #define GLOBAL_CONFIG_SPAD_ENABLES_REF0 0x0B0
00075 #define GLOBAL_CONFIG_SPAD_ENABLES_REF1 0x0B1
00076 #define GLOBAL_CONFIG_SPAD_ENABLES_REF2 0x0B2
00077 #define GLOBAL_CONFIG_SPAD_ENABLES_REF3 0x0B3
00078 #define GLOBAL_CONFIG_SPAD_ENABLES_REF4 0x0B4
00079 #define GLOBAL_CONFIG_SPAD_ENABLES_REF5 0x0B5
00080 #define GLOBAL_CONFIG_REF_EN_START_SELECT 0xB6
00081 #define DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD 0x4E
00082 #define DYNAMIC_SPAD_REF_EN_START_OFFSET 0x4F
00083 #define POWER_MANAGEMENT_GO1_POWER_FORCE 0x80
00084 #define VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV 0x89
00085 #define ALGO_PHASECAL_LIM 0x30
00086 #define ALGO_PHASECAL_CONFIG_TIMEOUT 0x30
00087
00088
00089 #define SYSTEM_THRESH_HIGH                              0x0C
00090 #define SYSTEM_THRESH_LOW                              0x0E
00091
00092 #define  SYSTEM_SEQUENCE_CONFIG                          0x01
00093 #define  SYSTEM_RANGE_CONFIG                             0x09
00094 #define  SYSTEM_INTERMEASUREMENT_PERIOD                  0x04
00095
00096 #define  SYSTEM_INTERRUPT_CONFIG_GPIO                    0x0A
00097
00098 #define  GPIO_HV_MUX_ACTIVE_HIGH                         0x84
00099
00100 #define  SYSTEM_INTERRUPT_CLEAR                          0x0B
00101
00102 #define RESULT_INTERRUPT_STATUS                         0x13
00103 #define  RESULT_RANGE_STATUS                             0x14
00104
00105 #define  RESULT_CORE_AMBIENT_WINDOW_EVENTS_RTN           0xBC
00106 #define  RESULT_CORE_RANGING_TOTAL_EVENTS_RTN            0xC0
00107 #define  RESULT_CORE_AMBIENT_WINDOW_EVENTS_REF           0xD0
00108 #define  RESULT_CORE_RANGING_TOTAL_EVENTS_REF            0xD4
00109 #define  RESULT_PEAK_SIGNAL_RATE_REF                     0xB6
00110
00111 #define  ALGO_PART_TO_PART_RANGE_OFFSET_MM               0x28
00112
00113 #define  I2C_SLAVE_DEVICE_ADDRESS                        0x8A
00114 //#define  I2C_SLAVE_DEVICE_ADDRESS                        0x53
00115
00116 #define  MSRC_CONFIG_CONTROL                             0x60
00117
00118 #define  PRE_RANGE_CONFIG_MIN_SNR                        0x27
00119 #define  PRE_RANGE_CONFIG_VALID_PHASE_LOW                0x56
00120 #define  PRE_RANGE_CONFIG_VALID_PHASE_HIGH               0x57
00121 #define  PRE_RANGE_MIN_COUNT_RATE_RTN_LIMIT              0x64
00122
00123 #define  FINAL_RANGE_CONFIG_MIN_SNR                      0x67
00124 #define  FINAL_RANGE_CONFIG_VALID_PHASE_LOW              0x47
00125 #define  FINAL_RANGE_CONFIG_VALID_PHASE_HIGH             0x48
00126 #define  FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT     0x44
00127
00128 #define  PRE_RANGE_CONFIG_SIGMA_THRESH_HI                0x61
00129 #define  PRE_RANGE_CONFIG_SIGMA_THRESH_LO                0x62
00130
00131 #define  PRE_RANGE_CONFIG_VCSEL_PERIOD                   0x50
00132 #define  PRE_RANGE_CONFIG_TIMEOUT_MACROP_HI             0x51
00133 #define  PRE_RANGE_CONFIG_TIMEOUT_MACROP_LO              0x52
```

```
00134
00135 #define  SYSTEM_HISTOGRAM_BIN                          0x81
00136 #define  HISTOGRAM_CONFIG_INITIAL_PHASE_SELECT         0x33
00137 #define  HISTOGRAM_CONFIG_READOUT_CTRL                 0x55
00138
00139 #define  FINAL_RANGE_CONFIG_VCSEL_PERIOD               0x70
00140 #define  FINAL_RANGE_CONFIG_TIMEOUT_MACROP_HI          0x71
00141 #define  FINAL_RANGE_CONFIG_TIMEOUT_MACROP_LO          0x72
00142 #define CROSSTALK_COMPENSATION_PEAK_RATE_MCPS        0x20
00143
00144 #define MSRC_CONFIG_TIMEOUT_MACROP                    0x46
00145
00146 #define  SOFT_RESET_GO2_SOFT_RESET_N                   0xBF
00147 #define  IDENTIFICATION_MODEL_ID                       0xC0
00148 #define  IDENTIFICATION_REVISION_ID                   0xC2
00149
00150 #define  OSC_CALIBRATE_VAL                             0xF8
00151
00152 #define  GLOBAL_CONFIG_VCSEL_WIDTH                     0x32
00153 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_0              0xB0
00154 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_1              0xB1
00155 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_2              0xB2
00156 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_3             0xB3
00157 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_4              0xB4
00158 #define  GLOBAL_CONFIG_SPAD_ENABLES_REF_5              0xB5
00159
00160 #define  GLOBAL_CONFIG_REF_EN_START_SELECT             0xB6
00161 #define  DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD           0x4E
00162 #define  DYNAMIC_SPAD_REF_EN_START_OFFSET              0x4F
00163 #define  POWER_MANAGEMENT_GO1_POWER_FORCE              0x80
00164
00165 #define  VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV             0x89
00166
00167 #define  ALGO_PHASECAL_LIM                             0x30
00168 #define  ALGO_PHASECAL_CONFIG_TIMEOUT                  0x30
00169
00170 //------------------------------------------------------------
00171 // Defines
00172 //------------------------------------------------------------
00173 // I use a 8-bit number for the address, LSB must be 0 so that I can
00174 // OR over the last bit correctly based on reads and writes
00175 #define ADDRESS_DEFAULT 0b01010010
00176 #define ADDRESS_DEFAULT2 0b00101001
00177 // Record the current time to check an upcoming timeout against
00178 #define startTimeout() (g_timeoutStartMs = millis())
00179
00180 // Check if timeout is enabled (set to nonzero value) and has expired
00181 #define checkTimeoutExpired() (g_ioTimeout > 0 && ((uint16_t)millis() - g_timeoutStartMs) >
      g_ioTimeout)
00182
00183 // Decode VCSEL (vertical cavity surface emitting laser) pulse period in PCLKs
00184 // from register value
00185 // based on VL53L0X_decode_vcsel_period()
00186 #define decodeVcselPeriod(reg_val)      (((reg_val) + 1) « 1)
00187
00188 // Encode VCSEL pulse period register value from period in PCLKs
00189 // based on VL53L0X_encode_vcsel_period()
00190 #define encodeVcselPeriod(period_pclks) (((period_pclks) » 1) - 1)
00191
00192 // Calculate macro period in *nanoseconds* from VCSEL period in PCLKs
00193 // based on VL53L0X_calc_macro_period_ps()
00194 // PLL_period_ps = 1655; macro_period_vclks = 2304
00195 #define calcMacroPeriod(vcsel_period_pclks) ((((uint32_t)2304 * (vcsel_period_pclks) * 1655) + 500) /
      1000)
00196
00197 // register addresses from API vl53l0x_device.h (ordered as listed there)
00198
00199
00200 typedef enum { VcselPeriodPreRange, VcselPeriodFinalRange }vcselPeriodType;
00201
00202 // Additional info for one measurement
00203 typedef struct{
00204   uint16_t rawDistance; //uncorrected distance  [mm],   uint16_t
00205   uint16_t signalCnt;   //Signal  Counting Rate [mcps], uint16_t, fixpoint9.7
00206   uint16_t ambientCnt;  //Ambient Counting Rate [mcps], uint16_t, fixpoint9.7
00207   uint16_t spadCnt;     //Effective SPAD return count,  uint16_t, fixpoint8.8
00208   uint8_t  rangeStatus; //Ranging status (0-15)
00209 } statInfo_t;
00210
00211
00212 //------------------------------------------------------------
00213 // API Functions
00214 //------------------------------------------------------------
00215 // configures chip i2c and lib for `new_addr` (8 bit, LSB=0)
00216 void setAddress(uint8_t new_addr);
00217 // Returns the current I2C address.
00218 uint8_t getAddress(void);
```

```
00219
00220 // Iniitializes and configures the sensor.
00221 // If the optional argument io_2v8 is 1, the sensor is configured for 2V8 mode (2.8 V I/O);
00222 // if 0, the sensor is left in 1V8 mode. Returns 1 if the initialization completed successfully.
00223 uint8_t initVL53L0X();
00224
00225 // Sets the return signal rate limit to the given value in units of MCPS (mega counts per second).
00226 // This is the minimum amplitude of the signal reflected from the target and received by the sensor
00227 //  necessary for it to report a valid reading. Setting a lower limit increases the potential range
00228 // of the sensor but also increases the likelihood of getting an inaccurate reading because of
00229 //  reflections from objects other than the intended target. This limit is initialized to 0.25 MCPS
00230 //  by default. The return value is a boolean indicating whether the requested limit was valid.
00231 uint8_t setSignalRateLimit(float limit_Mcps);
00232
00233 // Returns the current return signal rate limit in MCPS.
00234 float getSignalRateLimit(void);
00235
00236 // Set the measurement timing budget in microseconds, which is the time allowed
00237 // for one measurement; the ST API and this library take care of splitting the
00238 // timing budget among the sub-steps in the ranging sequence. A longer timing
00239 // budget allows for more accurate measurements. Increasing the budget by a
00240 // factor of N decreases the range measurement standard deviation by a factor of
00241 // sqrt(N). Defaults to about 33 milliseconds; the minimum is 20 ms.
00242 // based on VL53L0X_set_measurement_timing_budget_micro_seconds()
00243 uint8_t setMeasurementTimingBudget(uint32_t budget_us);
00244
00245 // Returns the current measurement timing budget in microseconds.
00246 uint32_t getMeasurementTimingBudget(void);
00247
00248 // Sets the VCSEL (vertical cavity surface emitting laser) pulse period for the given period type
00249 // (VcselPeriodPreRange or VcselPeriodFinalRange) to the given value (in PCLKs).
00250 // Longer periods increase the potential range of the sensor. Valid values are (even numbers only):
00251 // Pre: 12 to 18 (initialized to 14 by default)
00252 // Final: 8 to 14 (initialized to 10 by default)
00253 // The return value is a boolean indicating whether the requested period was valid.
00254 uint8_t setVcselPulsePeriod(vcselPeriodType type, uint8_t period_pclks);
00255
00256 // Returns the current VCSEL pulse period for the given period type.
00257 uint8_t getVcselPulsePeriod(vcselPeriodType type);
00258
00259 // Starts continuous ranging measurements. If the argument period_ms is 0,
00260 // continuous back-to-back mode is used (the sensor takes measurements as often as possible);
00261 // if it is nonzero, continuous timed mode is used, with the specified inter-measurement period
00262 // in milliseconds determining how often the sensor takes a measurement.
00263 void startContinuous(uint32_t period_ms);
00264
00265 // Stops continuous mode.
00266 void stopContinuous(void);
00267
00268 // Returns a range reading in millimeters when continuous mode is active.
00269 // Additional measurement data will be copied into `extraStats` if it is non-zero.
00270 uint16_t readRangeContinuousMillimeters(/* statInfo_t *extraStats*/ );
00271
00272 // Performs a single-shot ranging measurement and returns the reading in millimeters.
00273 // Additional measurement data will be copied into `extraStats` if it is non-zero.
00274 uint16_t readRangeSingleMillimeters( /*statInfo_t *extraStats */);
00275
00276 // Sets a timeout period in milliseconds after which read operations will abort
00277 // if the sensor is not ready. A value of 0 disables the timeout.
00278 void setTimeout(uint16_t timeout);
00279
00280 // Returns the current timeout period setting.
00281 uint16_t getTimeout(void);
00282
00283 // Indicates whether a read timeout has occurred since the last call to timeoutOccurred().
00284 bool timeoutOccurred(void);
00285
00286 //------------------------------------------------------------
00287 // I2C communication Functions
00288 //------------------------------------------------------------
00289 void writeReg(uint8_t reg, uint8_t value);        // Write an 8-bit register
00290 void writeReg16Bit(uint8_t reg, uint16_t value);  // Write a 16-bit register
00291 void writeReg32Bit(uint8_t reg, uint32_t value);  // Write a 32-bit register
00292 uint8_t readReg(uint8_t reg);                     // Read an 8-bit register
00293 uint16_t readReg16Bit(uint8_t reg);               // Read a 16-bit register
00294 uint32_t readReg32Bit(uint8_t reg);               // Read a 32-bit register
00295 // Write `count` number of bytes from `src` to the sensor, starting at `reg`
00296 void writeMulti(uint8_t reg, uint8_t const *src, uint8_t count);
00297
00298 // TCC: Target CentreCheck
00299 // MSRC: Minimum Signal Rate Check
00300 // DSS: Dynamic Spad Selection
00301 typedef struct {
00302   uint8_t tcc, msrc, dss, pre_range, final_range;
00303 }SequenceStepEnables;
00304
00305 typedef struct {
```

```
00306   uint16_t pre_range_vcsel_period_pclks, final_range_vcsel_period_pclks;
00307
00308   uint16_t msrc_dss_tcc_mclks, pre_range_mclks, final_range_mclks;
00309   uint32_t msrc_dss_tcc_us,    pre_range_us,    final_range_us;
00310 }SequenceStepTimeouts;
00311
```

5.23 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 //================================================================
```

5.24 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
```

5.25 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 }
```

5.26 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 
```

5.27 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 }
```

5.28 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
```

5.29 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 
```

5.30 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 SEUIL_DIST_SENSOR 800
- #define ROS_DOMAIN_ID 0
- #define LCD 1
- #define VL53 1
- #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 =100 }
- 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_VL53 (void ∗pvParameters)
- void task_Grove_LCD (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)
- 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

5.30.1 Detailed Description

file that contain the main code

Definition in file main.c.

5.30.2 Macro Definition Documentation

5.30.2.1 CAMERA_X_MAX

```
#define CAMERA_X_MAX 640
```

Define maximal x position return by camera

Definition at line 73 of file main.c.

5.30.2.2 CAMERA_X_MIN

```
#define CAMERA_X_MIN 0
```

Define minimal x position return by camera

Definition at line 72 of file main.c.

5.30.2.3 CAMERA_Y_MAX

```
#define CAMERA_Y_MAX 480
```

Define maximal y position return by camera

Definition at line 75 of file main.c.

5.30.2.4 CAMERA_Y_MIN

```
#define CAMERA_Y_MIN 0
```

Define minimal y position return by camera

Definition at line 74 of file main.c.

5.30.2.5 CMD

```
#define CMD 1000
```

Can be use as default speed

Definition at line 66 of file main.c.

5.30.2.6 DEBUG_MOTOR

```
#define DEBUG_MOTOR 0
```

Activate motor debug print

Definition at line 56 of file main.c.

5.30.2.7 DEBUG_PRINTF

```
#define DEBUG_PRINTF 0
```

Activate debug print

Definition at line 55 of file main.c.

5.30.2.8 DEFAULT_DIR

```
#define DEFAULT_DIR STOP
```

Default direction at startup

Definition at line 82 of file main.c.

5.30.2.9 DEFAULT_MODE

```
#define DEFAULT_MODE MODE_ZIG
```

Default mode at startup

Definition at line 80 of file main.c.

5.30.2.10 DEFAULT_SPEED

```
#define DEFAULT_SPEED LOW
```

Default speed at startup

Definition at line 81 of file main.c.

5.30.2.11 EXCORRECTOR

```
#define EXCORRECTOR 2
```

Code to calibrate your correcteur

Definition at line 41 of file main.c.

5.30.2.12 EXFINAL

```
#define EXFINAL 6
```

Final code

Definition at line 45 of file main.c.

5.30.2.13 EXSTARTUP

```
#define EXSTARTUP 0
```

startup code

Definition at line 39 of file main.c.

5.30.2.14 EXTEST_MICROROS

```
#define EXTEST_MICROROS 5
```

Test Micro ROS subscriber and publisher

Definition at line 44 of file main.c.

5.30.2.15 EXTEST_UART2

```
#define EXTEST_UART2 1
```

Test printf and scanf function

Definition at line 40 of file main.c.

5.30.2.16 EXTEST_VL53

```
#define EXTEST_VL53 4
```

Test VL530X sensor

Definition at line 43 of file main.c.

5.30.2.17 EXTESTCORRECTOR

```
#define EXTESTCORRECTOR 3
```

Code to test your correcteur

Definition at line 42 of file main.c.

5.30.2.18 LCD

```
#define LCD 1
```

Activate LCD task

Definition at line 52 of file main.c.

5.30.2.19 LKi

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

Ki factor for the left motor

Definition at line 61 of file main.c.

5.30.2.20 LKp

```
#define LKp 0.001
```

Kp factor for the left motor

Definition at line 60 of file main.c.

5.30.2.21 MICROROS

```
#define MICROROS 1
```

Activate MicroROS task

Definition at line 54 of file main.c.

5.30.2.22 NB

```
#define NB 200
```

Number of samples in correcteur calibration task

Definition at line 85 of file main.c.

5.30.2.23 RKi

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

Kp factor for the right motor

Definition at line 63 of file main.c.

5.30.2.24 RKp

```
#define RKp 0.001
```

Kp factor for the right motor

Definition at line 62 of file main.c.

5.30.2.25 ROS_DOMAIN_ID

```
#define ROS_DOMAIN_ID 0
```

Define ROS domain id

Definition at line 51 of file main.c.

5.30.2.26 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.

5.30.2.27 SEUIL_DIST_SENSOR

```
#define SEUIL_DIST_SENSOR 800
```

Define the trigger for forward sensors4

Definition at line 50 of file main.c.

5.30.2.28 SYNCHRO_EX

```
#define SYNCHRO_EX EXFINAL
```

Define wich config are executed

Definition at line 47 of file main.c.

5.30.2.29 Te

```
#define Te SAMPLING_PERIOD_ms
```

Definition at line 59 of file main.c.

5.30.2.30 TEST_CORRECTOR_DUTY

```
#define TEST_CORRECTOR_DUTY 150
```

Duty cycle to apply to calibrate the correcteur

Definition at line 86 of file main.c.

5.30.2.31 TEST_CORRECTOR_SPEEDL

```
#define TEST_CORRECTOR_SPEEDL -100
```

Speed to test the left correcteur

Definition at line 87 of file main.c.

5.30.2.32 TEST_CORRECTOR_SPEEDR

```
#define TEST_CORRECTOR_SPEEDR -100
```

Speed to test the right correcteur

Definition at line 88 of file main.c.

5.30.2.33 TEST_LEFT_MOTOR

```
#define TEST_LEFT_MOTOR 1
```

Calibrate left motor correcteur or not

Definition at line 89 of file main.c.

5.30.2.34 VITESSE_CAM

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

Default speed for camera mode

Definition at line 69 of file main.c.

5.30.2.35 VITESSE_KART

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

Default speed for manual mode

Definition at line 67 of file main.c.

5.30.2.36 VITESSE_OBS

```
#define VITESSE_OBS CMD
```

Default speed for obstacle mode

Definition at line 68 of file main.c.

5.30.2.37 VL53

```
#define VL53 1
```

Activate VL530X task

Definition at line 53 of file main.c.

5.30.3 Enumeration Type Documentation

5.30.3.1 anonymous enum

```
anonymous enum
```

enumerate mode of robot

Definition at line 31 of file main.c.

5.30.3.2 anonymous enum

```
anonymous enum
```

enumerate speed

Definition at line 33 of file main.c.

5.30.3.3 anonymous enum

```
anonymous enum
```

enumerate direction

Definition at line 35 of file main.c.

5.30.4 Function Documentation

5.30.4.1 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 154 of file main.c.

5.30.4.2 Error_Handler()

```
void Error_Handler (
             void  )
```

Definition at line 914 of file main.c.

5.30.4.3 HAL_TIM_PeriodElapsedCallback()

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

Definition at line 906 of file main.c.

5.30.4.4 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 752 of file main.c.

5.30.4.5 microros_allocate()

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

Definition at line 14 of file microros_allocators.c.

5.30.4.6 microros_deallocate()

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

Definition at line 22 of file microros_allocators.c.

5.30.4.7 microros_reallocate()

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

Definition at line 31 of file microros_allocators.c.

5.30.4.8 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 168 of file main.c.

5.30.4.9 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.

5.30.4.10 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 156 of file main.c.

5.30.4.11 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.

5.30.4.12 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	

Definition at line 463 of file main.c.

5.30.4.13 task_Motor_Left()

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

Task use to control the left motor of the robot

Parameters

argument	

Definition at line 385 of file main.c.

5.30.4.14 task_Motor_Right()

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

Task use to control the right motor of the robot

Parameters

argument	

Definition at line 411 of file main.c.

5.30.4.15 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 498 of file main.c.

5.30.4.16 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	

Definition at line 438 of file main.c.

5.30.4.17 test_motor()

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

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

Definition at line 868 of file main.c.

5.30.4.18 test_uart2()

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

Use to test printf and scanf function

Definition at line 845 of file main.c.

5.30.4.19 test_vl53()

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

Use to test the VL53 sensor

Definition at line 857 of file main.c.

5.30.5 Variable Documentation

5.30.5.1 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.

5.30.5.2 defaultTaskHandle

```
osThreadId_t defaultTaskHandle
```

Definition at line 23 of file main.c.

5.30.5.3 hdma_usart1_rx

```
DMA_HandleTypeDef hdma_usart1_rx  [extern]
```

Definition at line 6 of file drv_uart.c.

5.30.5.4 hdma_usart1_tx

```
DMA_HandleTypeDef hdma_usart1_tx  [extern]
```

Definition at line 7 of file drv_uart.c.

5.30.5.5 hdma_usart2_rx

```
DMA_HandleTypeDef hdma_usart2_rx  [extern]
```

Definition at line 8 of file drv_uart.c.

5.30.5.6 hdma_usart2_tx

```
DMA_HandleTypeDef hdma_usart2_tx  [extern]
```

Definition at line 9 of file drv_uart.c.

5.30.5.7 hi2c1

```
I2C_HandleTypeDef hi2c1  [extern]
```

Definition at line 6 of file drv_i2c.c.

5.30.5.8 huart1

```
UART_HandleTypeDef huart1  [extern]
```

Definition at line 4 of file drv_uart.c.

5.30.5.9 huart2

```
UART_HandleTypeDef huart2  [extern]
```

Definition at line 5 of file drv_uart.c.

5.30.5.10 q_mot_L

```
xQueueHandle q_mot_L = NULL
```

Queue to communicate with left motor task

Definition at line 97 of file main.c.

5.30.5.11 q_mot_R

```
xQueueHandle q_mot_R = NULL
```

Queue to communicate with right motor task

Definition at line 98 of file main.c.

5.30.5.12 qhLCD

```
xQueueHandle qhLCD = NULL
```

Queue to communicate with LCD task

Definition at line 101 of file main.c.

5.30.5.13 qhMR_pub

```
xQueueHandle qhMR_pub = NULL
```

Queue to communicate with microRos task

Definition at line 100 of file main.c.

5.30.5.14 qhMR_sub

```
xQueueHandle qhMR_sub = NULL
```

Queue to get information from microRos task

Definition at line 99 of file main.c.

5.30.5.15 qhVl53

```
xQueueHandle qhVl53 = NULL
```

Queue to communicate with VL53 task

Definition at line 102 of file main.c.

5.30.5.16 tab_speed

```
int16_t tab_speed[NB]
```

use to store speed of motor during calibration of the correcteur

Definition at line 105 of file main.c.

5.30.5.17 xSem_Supervision

```
xSemaphoreHandle xSem_Supervision = NULL
```

Semaphore use in decision task

Definition at line 94 of file main.c.

5.31 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=100};
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 SEUIL_DIST_SENSOR 800
00051 #define ROS_DOMAIN_ID 0
00052 #define LCD 1
00053 #define VL53 1
00054 #define MICROROS 1
00055 #define DEBUG_PRINTF 0
00056 #define DEBUG_MOTOR 0
00059 #define Te SAMPLING_PERIOD_ms
00060 #define LKp 0.001
00061 #define LKi (5.0/(0.1*40.0))
00062 #define RKp 0.001
00063 #define RKi (5.0/(0.1*40.0))
00066 #define CMD 1000
00067 #define VITESSE_KART CMD/2
00068 #define VITESSE_OBS CMD
00069 #define VITESSE_CAM CMD/3
00072 #define CAMERA_X_MIN 0
00073 #define CAMERA_X_MAX 640
00074 #define CAMERA_Y_MIN 0
00075 #define CAMERA_Y_MAX 480
00076 //#define CAMERA_X_TIER (CAMERA_X_MAX-CAMERA_X_MIN)/3 /**< */
00077 //#define CAMERA_Y_TIER (CAMERA_Y_MAX-CAMERA_Y_MIN)/3 /**< */
00080 #define DEFAULT_MODE MODE_ZIG
00081 #define DEFAULT_SPEED LOW
00082 #define DEFAULT_DIR STOP
00085 #define NB 200
00086 #define TEST_CORRECTOR_DUTY 150
00087 #define TEST_CORRECTOR_SPEEDL -100
00088 #define TEST_CORRECTOR_SPEEDR -100
00089 #define TEST_LEFT_MOTOR 1
00092 // Déclaration des objets synchronisants !! Ne pas oublier de les créer
00094 xSemaphoreHandle xSem_Supervision = NULL;
00097 xQueueHandle q_mot_L = NULL;
00098 xQueueHandle q_mot_R = NULL;
00099 xQueueHandle qhMR_sub = NULL;
00100 xQueueHandle qhMR_pub = NULL;
00101 xQueueHandle qhLCD = NULL;
00102 xQueueHandle qhVl53 = NULL;
00105 int16_t tab_speed[NB];
00111 typedef struct
00112 {
00113     char command;
00114     int data;
00115 } AMessage;
00116
00120 typedef struct
00121 {
00122     char dir;
00123     int mode;
00124     int speed;
00125 } MicroRosPubMsg;
00126
00130 typedef struct
00131 {
00132     int dir;
00133     int x;
00134     int y;
00135     int mode;
00136     int speed;
00137 } MicroRosSubMsg;
00140 //Robot function
00141 void SystemClock_Config(void);
00142
00143 //Micro-Ros function
00144 bool cubemx_transport_open(struct uxrCustomTransport * transport);
00145 bool cubemx_transport_close(struct uxrCustomTransport * transport);
00146 size_t cubemx_transport_write(struct uxrCustomTransport* transport, const uint8_t * buf, size_t len,
      uint8_t * err);
00147 size_t cubemx_transport_read(struct uxrCustomTransport* transport, uint8_t* buf, size_t len, int
      timeout, uint8_t* err);
```

```
00148
00149 void * microros_allocate(size_t size, void * state);
00150 void microros_deallocate(void * pointer, void * state);
00151 void * microros_reallocate(void * pointer, size_t size, void * state);
00152 void * microros_zero_allocate(size_t number_of_elements, size_t size_of_element, void * state);
00153
00154 void CHECKMRRET(rcl_ret_t ret, char* msg){if (ret != RCL_RET_OK){ if (DEBUG_PRINTF){printf("Error :
      %d\r\nMsg : %s\r\n", (int)ret, msg); }}}
00155
00156 void SubscriberCallbackFunction(const void *msgin){
00157 #if SYNCHRO_EX == EXTEST_MICROROS
00158     std_msgs__msg__String * msg = (std_msgs__msg__String * )msgin;
00159     printf("\r\nMessage recue : %s\r\n", msg->data->data);
00160 #elif SYNCHRO_EX == EXFINAL
00161     std_msgs__msg__Int32 * msg = (std_msgs__msg__Int32 * )msgin;
00162     if (DEBUG_PRINTF)
00163         printf("\r\nMessage recue : %ld\r\n", msg->data);
00164 #endif //SYNCHRO_EX
00165 }
00166
00167 // https://github.com/lFatality/stm32_micro_ros_setup
00168 void microros_task(void *argument)
00169 {
00170     // micro-ROS app variable
00171     rclc_support_t support; //Contain information about how config microros
00172     rcl_allocator_t allocator; //Contain information about how microRos can allocate memory
00173     rcl_node_t node; //microRos structure wich represent a node ROS
00174     rcl_node_options_t node_opt; //microRos structure wich represent option of a node ROS
00175     rclc_executor_t executor; //microRos structure wich represent an executor wich can be use to
    receive message
00176
00177     // micro-ROS configuration with freertos
00178     rmw_uros_set_custom_transport(
00179         true,
00180         (void *) &huart1,
00181         cubemx_transport_open,
00182         cubemx_transport_close,
00183         cubemx_transport_write,
00184         cubemx_transport_read);
00185
00186     rcl_allocator_t freeRTOS_allocator = rcutils_get_zero_initialized_allocator();
00187     freeRTOS_allocator.allocate = microros_allocate;
00188     freeRTOS_allocator.deallocate = microros_deallocate;
00189     freeRTOS_allocator.reallocate = microros_reallocate;
00190     freeRTOS_allocator.zero_allocate =  microros_zero_allocate;
00191
00192     if (!rcutils_set_default_allocator(&freeRTOS_allocator)) {
00193         printf("Error on default allocators (line %d)\r\n", __LINE__);
00194     }
00195
00196     allocator = rcl_get_default_allocator();
00197
00198     //create init_options
00199     CHECKMRRET(rclc_support_init(&support, 0, NULL, &allocator), "error on init support");
00200     // create node
00201     node_opt = rcl_node_get_default_options(); //Get default node options
00202     node_opt.domain_id = ROS_DOMAIN_ID; //Set the ROS_DOMAIN_ID
00203     CHECKMRRET(rclc_node_init_with_options(&node, "STM32_node", "", &support, &node_opt), "error on
    init node");
00204
00205
00206 #if SYNCHRO_EX == EXSTARTUP
00207     static int counter = 0;
00208     rcl_ret_t ret; //Use to store the return of microRos function
00209     rcl_publisher_t publisher; //microRos structure wich represent a publisher
00210     std_msgs__msg__String msg; //microRos structure wich represent a String ROS message
00211
00212     CHECKMRRET(rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs,
      msg, String), "cubemx_publisher"),
00213             "Error when create publisher"); //Create a default publisher wich publish on topic named
      "cubemx_publisher"
00214
00215     //Allocate memory for string message
00216     msg.data.data = (char * ) malloc(100 * sizeof(char));
00217     msg.data.size = 0;
00218     msg.data.capacity = 100; //Capacity need to be less than or equal to the allocution memory space
00219
00220     for (;;)
00221     {
00222         sprintf(msg.data.data, "Hello from micro-ROS #%d", counter++); //Write string in message
00223         msg.data.size = strlen(msg.data.data); //Set the size of the message
00224         ret = rcl_publish(&publisher, &msg, NULL); //Publish the message
00225         if (ret != RCL_RET_OK)
00226             printf("Error publishing (line %d)\r\n", __LINE__); //If the message are not publish print
      an error
00227         vTaskDelay(SAMPLING_PERIOD_ms);
00228     }
```

```
00229 #elif SYNCHRO_EX == EXTEST_MICROROS
00230     //micro-ros topic variable
00231     rcl_ret_t ret; //Use to store the return of microRos function
00232     rcl_publisher_t publisher; //microRos structure wich represent a publisher
00233     rcl_subscription_t subscriber; //microRos structure wich represent a subsriber
00234     std_msgs__msg__String msg; //microRos structure wich represent a String ROS message
00235 
00236     //create default publisher wich publish on topic named "cubemx_publisher"
00237     CHECKMRRET(rclc_publisher_init_default(&publisher, &node, ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs,
    msg, String), "cubemx_publisher"),
00238             "Error when create publisher");
00239 
00240     //create default subscriber wich listen to the topic named "cubemx_subscriber"
00241     subscriber = rcl_get_zero_initialized_subscription();
00242     CHECKMRRET(rclc_subscription_init_default(&subscriber, &node,
    ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, String),"cubemx_subscriber"),
00243             "Error when create subscriber");
00244 
00245     //Init string msg
00246     msg.data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
00247     msg.data.size = 0;
00248     msg.data.capacity = ARRAY_LEN;
00249 
00250     // Init executor by indicate how many subscriber we will put in it
00251     CHECKMRRET(rclc_executor_init(
00252             &executor, //executor structure
00253             &support.context,
00254             1, //number of subscriber that will be add
00255             &allocator), "Error on init executor");
00256     //Add subsciber to the executor
00257     CHECKMRRET(rclc_executor_add_subscription(&executor, //executor structure
00258             &subscriber, //subscriber structure
00259             &msg, //msg structure
00260             &SubscriberCallbackFunction, ON_NEW_DATA), "error add subscriber");
00261 
00262     for (;;)
00263     {
00264         //Execute the executor to receive message
00265         ret = rclc_executor_spin_some(&executor, 100*1000*1000);
00266         vTaskDelay(SAMPLING_PERIOD_ms);
00267     }
00268 #elif SYNCHRO_EX == EXFINAL
00269     //Init the queue mesage
00270     MicroRosPubMsg MsgToPub = {'N', 0, 0};
00271     MicroRosSubMsg SubToMsg = {DEFAULT_DIR, 0, 0, DEFAULT_MODE, DEFAULT_SPEED};
00272     /* PUBLISHER */
00273     //Use to publish the direction of robot in sensor mode
00274     rcl_publisher_t capteur_dir_pub;
00275     char* capteur_dir_topic = CAPTEUR_DIR_TOPIC;
00276     std_msgs__msg__Int32 capteur_dir_msg;
00277     //Use to publish the actual mode of the robot
00278     rcl_publisher_t etat_mode_pub;
00279     char* etat_mode_topic = ETAT_MODE_TOPIC;
00280     std_msgs__msg__Int32 etat_mode_msg;
00281     //Use to publish the actual speed of the robot
00282     rcl_publisher_t etat_speed_pub;
00283     char* etat_speed_topic = ETAT_SPEED_TOPIC;
00284     std_msgs__msg__Int32 etat_speed_msg;
00285     /* SUBSCRIBER */
00286     //Use to receive the x position of object see by the camera
00287     rcl_subscription_t camera_x_sub;
00288     char* camera_x_topic = CAMERA_X_TOPIC;
00289     std_msgs__msg__Int32 camera_x_msg;
00290     //Use to receive the y position of object see by the camera
00291     rcl_subscription_t camera_y_sub;
00292     char* camera_y_topic = CAMERA_Y_TOPIC;
00293     std_msgs__msg__Int32 camera_y_msg;
00294     //Use to receive the remote control in remote mode
00295     rcl_subscription_t telecommande_dir_sub;
00296     char* telecommande_dir_topic = TELECOMMANDE_DIR_TOPIC;
00297     std_msgs__msg__Int32 telecommande_dir_msg;
00298     //Use to receive the mode config
00299     rcl_subscription_t config_mode_sub;
00300     char* config_mode_topic = CONFIG_MODE_TOPIC;
00301     std_msgs__msg__Int32 config_mode_msg;
00302     //Use to receive the speed config
00303     rcl_subscription_t config_speed_sub;
00304     char* config_speed_topic = CONFIG_SPEED_TOPIC;
00305     std_msgs__msg__Int32 config_speed_msg;
00306 
00307     // create publisher
00308     createPublisher(&capteur_dir_pub, &node,
00309         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00310         capteur_dir_topic, &capteur_dir_msg);
00311 
00312     createPublisher(&etat_mode_pub, &node,
00313         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
```

```
00314         etat_mode_topic, &etat_mode_msg);
00315
00316     createPublisher(&etat_speed_pub, &node,
00317         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00318         etat_speed_topic, &etat_speed_msg);
00319
00320     //create subscriber
00321     createSubscriber(&camera_x_sub, &node,
00322         ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00323         camera_x_topic, &camera_x_msg);
00324
00325     createSubscriber(&camera_y_sub, &node,
00326             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00327             camera_y_topic, &camera_y_msg);
00328
00329     createSubscriber(&telecommande_dir_sub, &node,
00330             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00331             telecommande_dir_topic, &telecommande_dir_msg);
00332
00333     createSubscriber(&config_mode_sub, &node,
00334             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00335             config_mode_topic, &config_mode_msg);
00336
00337     createSubscriber(&config_speed_sub, &node,
00338             ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg, Int32),
00339             config_speed_topic, &config_speed_msg);
00340
00341     //Init the executor
00342     CHECKMRRET(rclc_executor_init(&executor, &support.context, 5, &allocator), "Error on init
    executor");
00343     /*Add subscriber to executor to let it check if message is receive on this
00344     topic and store the data on the message structure after call the callback*/
00345     CHECKMRRET(rclc_executor_add_subscription(&executor, &camera_x_sub, &camera_x_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add camera_x_sub");
00346     CHECKMRRET(rclc_executor_add_subscription(&executor, &camera_y_sub, &camera_y_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add camera_y_sub");
00347     CHECKMRRET(rclc_executor_add_subscription(&executor, &telecommande_dir_sub, &telecommande_dir_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add telecommande_dir_sub");
00348     CHECKMRRET(rclc_executor_add_subscription(&executor, &config_mode_sub, &config_mode_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add config_mode_sub");
00349     CHECKMRRET(rclc_executor_add_subscription(&executor, &config_speed_sub, &config_speed_msg,
    &SubscriberCallbackFunction, ON_NEW_DATA), "error add config_speed_sub");
00350
00351     for(;;)
00352     {
00353         if (!uxQueueMessagesWaiting(qhMR_sub)) //If no message in 'output' queue
00354             xQueueSend(qhMR_sub, ( void * ) &SubToMsg, portMAX_DELAY); //Send queue message
00355         rclc_executor_spin_some(&executor, 1*1000*1000); //Execute executor
00356
00357         //Put the receive data into the queue message structure
00358         SubToMsg.dir = telecommande_dir_msg.data;
00359         SubToMsg.x = camera_x_msg.data;
00360         SubToMsg.y = camera_y_msg.data;
00361         SubToMsg.mode = config_mode_msg.data;
00362         SubToMsg.speed = config_speed_msg.data;
00363
00364         if (uxQueueMessagesWaiting(qhMR_pub)) //If no message in 'input' queue
00365         {
00366             xQueueReceive(qhMR_pub, &MsgToPub, portMAX_DELAY); //Receive data
00367             capteur_dir_msg.data = (int)MsgToPub.dir;
00368             etat_mode_msg.data = MsgToPub.mode;
00369             etat_speed_msg.data = MsgToPub.speed;
00370
00371             //Publish data
00372             CHECKMRRET(rcl_publish(&capteur_dir_pub, &capteur_dir_msg, NULL), "erreur publish
    capteur_dir_pub");
00373             CHECKMRRET(rcl_publish(&etat_mode_pub, &etat_mode_msg, NULL), "erreur publish
    etat_mode_pub");
00374             CHECKMRRET(rcl_publish(&etat_speed_pub, &etat_speed_msg, NULL), "erreur publish
    etat_speed_pub");
00375
00376             #if DEBUG_PRINTF
00377             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);
00378             #endif //DEBUG_PRINTF
00379         }
00380         vTaskDelay(SAMPLING_PERIOD_ms);
00381     }
00382 #endif //SYNCHRO_EX
00383 }
00384
00385 void task_Motor_Left(void *pvParameters)
00386 {
00387     int16_t consigne = 0; //Store the desirate speed
00388
00389     float ui = 0.0; //Integral term of the correcteur
00390     float up = 0.0; //Proportionnal term of the correcteur
```

```
00391     int err = 0; //Error term of the correcteur
00392     int speed = 0; //Actual speed of motor
00393 
00394     for (;;)
00395     {
00396         xQueueReceive(q_mot_L, &consigne, portMAX_DELAY); //receive wanted speed
00397 
00398         speed = quadEncoder_GetSpeedL(); //Get actual speed
00399         //Calculate term of correcteur
00400         err=consigne-speed;
00401         up=LKp*(float)err;
00402         ui=ui+LKp*LKi*(float)err;
00403 
00404         motorLeft_SetDuty(100+(int)(up+ui)); //Set duty cycle of the motor
00405 
00406         xSemaphoreGive(xSem_Supervision); //Give semaphore to liberate the decision task
00407         vTaskDelay(SAMPLING_PERIOD_ms);
00408     }
00409 }
00410 
00411 void task_Motor_Right(void *pvParameters)
00412 {
00413     int16_t consigne = 0; //Store the desirate speed
00414 
00415     float ui = 0.0; //Integral term of the correcteur
00416     float up = 0.0; //Proportionnal term of the correcteur
00417     int err = 0; //Error term of the correcteur
00418     int speed = 0; //Actual speed of motor
00419 
00420     for (;;)
00421     {
00422         xQueueReceive(q_mot_R, &consigne, portMAX_DELAY); //receive wanted speed
00423 
00424         speed = quadEncoder_GetSpeedR(); //Get actual speed
00425         //Calculate term of correcteur
00426         err=consigne-speed;
00427         up=RKp*(float)err;
00428         ui=ui+RKp*RKi*(float)err;
00429 
00430         motorRight_SetDuty(100+(int)(up+ui)); //Set duty cycle of the motor
00431 
00432         xSemaphoreGive(xSem_Supervision);//Give semaphore to liberate the decision task
00433         vTaskDelay(SAMPLING_PERIOD_ms);
00434     }
00435 }
00436 
00437 #if VL53
00438 void task_VL53(void *pvParameters)
00439 {
00440     static uint16_t dist;
00441     static const int SEUIL = 20; //Trigger
00442     int obs = 0; //Bool to indicate if we detect an obstacle or not
00443 
00444     for(;;)
00445     {
00446         dist = readRangeSingleMillimeters()/10; //Get the distance from the sensor
00447 
00448         if (dist < SEUIL && dist != 0) //If distance is less than the trigger
00449             obs = 1; //We detect an obstacle
00450         else
00451             obs = 0; //We do not detect an obstacle
00452 
00453         if (!uxQueueMessagesWaiting(qhVl53)) //If no data in queue
00454             xQueueSend(qhVl53, (void *)&obs, portMAX_DELAY); //Send data
00455 
00456         vTaskDelay(SAMPLING_PERIOD_ms);
00457     }
00458 }
00459 #endif //VL53
00460 
00461 
00462 #if LCD
00463 void task_Grove_LCD(void *pvParameters)
00464 {
00465 #if SYNCHRO_EX == EXSTARTUP
00466     for (;;)
00467     {
00468          groveLCD_setCursor(0,0); //Set cursor position to 0,0
00469          groveLCD_term_printf("TEST LCD"); //Write TEST LCD on the screen
00470          vTaskDelay(100);
00471     }
00472 #elif SYNCHRO_EX == EXFINAL
00473     AMessage pxRxedMessage;
00474 
00475     for(;;)
00476     {
00477         if (uxQueueMessagesWaiting(qhLCD)) //If data in the queue
```

```
00478         {
00479             xQueueReceive(qhLCD, &pxRxedMessage, portMAX_DELAY); //Receive data
00480             int mode = pxRxedMessage.data;
00481             char direction=pxRxedMessage.command;
00482             groveLCD_setCursor(0,0);
00483             //Write on screen information about mode
00484             if (mode == MODE_OBS)
00485                 groveLCD_term_printf("M:Obstacle  D:%c", direction);
00486             else if (mode == MODE_ZIG)
00487                 groveLCD_term_printf("M:Manuel        ");
00488             else if (mode == MODE_CAM)
00489                 groveLCD_term_printf("M:Camera        ");
00490         }
00491 
00492         vTaskDelay(SAMPLING_PERIOD_ms);
00493     }
00494 #endif //SYNCHRO_EX
00495 }
00496 #endif //LCD
00497 
00498 void task_Supervision(void *pvParameters)
00499 {
00500 #if SYNCHRO_EX == EXSTARTUP
00501     int16_t consigne_G=0; //Motor left speed
00502     int16_t consigne_D=0; //Motor rigth speed
00503 
00504     int tab_mes_ir[2]; //VL53L0X sensors values
00505     uint16_t mes_vl53=0; //VL530X sensor value
00506 
00507     vTaskDelay(100);
00508     for (;;)
00509     {
00510         //Get sensor value
00511         captDistIR_Get(tab_mes_ir);
00512         //mes_vl53 = readRangeSingleMillimeters()/10;
00513 
00514         if((tab_mes_ir[0]>2000)||(tab_mes_ir[1]>2000))
00515         { // !! obstacle
00516             consigne_G=0;
00517             consigne_D=0;
00518         }
00519         else
00520         {
00521             consigne_G=1000;
00522             consigne_D=1000;
00523         }
00524 
00525         xQueueSend( q_mot_L, ( void * ) &consigne_G,  portMAX_DELAY );
00526         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00527 
00528         xQueueSend( q_mot_R, ( void * ) &consigne_D,  portMAX_DELAY );
00529         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00530 
00531         vTaskDelay(SAMPLING_PERIOD_ms);
00532     }
00533 #elif SYNCHRO_EX == EXTESTCORRECTOR
00534     int16_t speedLeft = TEST_CORRECTOR_SPEEDL;
00535     int16_t speedRight = TEST_CORRECTOR_SPEEDR;
00536 
00537     for (;;)
00538     {
00539         xQueueSend(q_mot_L, (void *)&speedLeft, portMAX_DELAY);
00540         xSemaphoreTake(xSem_Supervision, portMAX_DELAY);
00541 
00542         xQueueSend(q_mot_R, (void *)&speedRight, portMAX_DELAY);
00543         xSemaphoreTake(xSem_Supervision, portMAX_DELAY);
00544 
00545         vTaskDelay(SAMPLING_PERIOD_ms);
00546     }
00547 #elif SYNCHRO_EX == EXFINAL
00548     int16_t speedLeft; //Motor left speed
00549     int16_t speedRight; //Motor rigth speed
00550 
00551     int table[2]; //VL53L0X sensors values
00552     #if VL53
00553     int vl53 = 0; //VL530X detect an obstacle or not
00554     #endif //VL53
00555 
00556     static int obs = 0; //store the number of different obstacle detected without break
00557     static char dir = 'f'; //represent the direction of the robot in obstacle mode
00558     static int direction = DEFAULT_DIR; //default direction of the robot
00559     static int speed = DEFAULT_SPEED; //default speed of the robot
00560     static int mode = DEFAULT_MODE; //default mode of the robot
00561     static int x = 0; //position x of the object detect by the camera
00562     static int y = 0; //position y of the object detect by the camera
00563 
00564 #if LCD
```

```
00565      AMessage pxMessage; //LCD queue message
00566 #endif
00567
00568 #if MICROROS
00569     MicroRosSubMsg SubToMsg; //ROS subscriber queue message
00570     MicroRosPubMsg MsgToPub; //ROS publisher queue message
00571 #endif //MICROROS
00572
00573     for (;;)
00574     {
00575         #if MICROROS
00576         if (uxQueueMessagesWaiting(qhMR_sub)) //If data  are in the the queue
00577         {
00578             xQueueReceive(qhMR_sub, &SubToMsg, portMAX_DELAY); //Receive data
00579             //Set mode, speed and direction if the data is correct
00580             if (SubToMsg.mode >= 0 && SubToMsg.mode < LAST_MODE)
00581                 mode = SubToMsg.mode;
00582             if (SubToMsg.dir >= 0 && SubToMsg.dir < LAST_DIR)
00583                 direction = SubToMsg.dir;
00584             if (SubToMsg.speed > 0 && SubToMsg.speed < LAST_SPEED)
00585                 speed = SubToMsg.speed;
00586             //Set x and y position
00587             x = SubToMsg.x;
00588             y = SubToMsg.y;
00589             #if DEBUG_PRINTF
00590             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);
00591             #endif //DEBUG_PRINTF
00592         }
00593         #endif //MICROROS
00594
00595         if (mode == MODE_ZIG) //Mode manual
00596         {
00597             dir = 'N'; //No direction information
00598             obs = 0; //No obstacle
00599             switch(direction) //Set the motor speed depending of the direction variable
00600             {
00601                 case STOP:
00602                     speedLeft = 0;
00603                     speedRight = 0;
00604                     break;
00605                 case AVANT:
00606                     speedLeft = VITESSE_KART+(8*(speed-50));
00607                     speedRight = VITESSE_KART+(8*(speed-50));
00608                     break;
00609                 case RECULE:
00610                     speedLeft = -(VITESSE_KART+(8*(speed-50)));
00611                     speedRight = -(VITESSE_KART+(8*(speed-50)));
00612                     break;
00613                 case DROITE:
00614                     speedLeft = VITESSE_KART+(8*(speed-50));
00615                     speedRight = -(VITESSE_KART+(8*(speed-50)));
00616                     break;
00617                 case GAUCHE:
00618                     speedLeft = -(VITESSE_KART+(8*(speed-50)));
00619                     speedRight = VITESSE_KART+(8*(speed-50));
00620                     break;
00621                 case AVANT_GAUCHE:
00622                     speedLeft = (VITESSE_KART/2)+(8*(speed-50));
00623                     speedRight = VITESSE_KART+(8*(speed-50));
00624                     break;
00625                 case AVANT_DROITE:
00626                     speedLeft = VITESSE_KART+(8*(speed-50));
00627                     speedRight = (VITESSE_KART/2)+(8*(speed-50));
00628                     break;
00629                 case RECULE_GAUCHE:
00630                     speedLeft = -(VITESSE_KART+(8*(speed-50)));
00631                     speedRight = -((VITESSE_KART/2)+(8*(speed-50)));
00632                     break;
00633                 case RECULE_DROITE:
00634                     speedLeft = -((VITESSE_KART/2)+(8*(speed-50)));
00635                     speedRight = -(VITESSE_KART+(8*(speed-50)));
00636                     break;
00637                 default:
00638                     speedLeft = 0;
00639                     speedRight = 0;
00640                     break;
00641             }
00642         }
00643         else if (mode == MODE_OBS) //Mode obstacle
00644         {
00645             //Get sensors informations
00646             captDistIR_Get(table);
00647             #if VL53
00648             if (uxQueueMessagesWaiting(qhVl53))
00649                 xQueueReceive(qhVl53, &vl53, portMAX_DELAY);
00650             else
```

```
00651                 vl53 = 0;
00652
00653             if (vl53 == 1) //if an obstacle is detected on the back we stop
00654             {
00655                 if (dir != 'S')
00656                     printf("Detection d'un obstacle à l'arrière");
00657                 speedLeft = 0;
00658                 speedRight = 0;
00659                 dir = 'S';
00660                 obs = 1;
00661             }
00662             else
00663             #endif //VL53
00664             if (table[0] > SEUIL_DIST_SENSOR || table[1] > SEUIL_DIST_SENSOR) //We have an obstacle in
      front of the robot
00665             {
00666                 if (obs > 10) //If we detect more than 10 different obstacle we turn on the left until
      they are no more obstacle
00667                 {
00668                     speedLeft = VITESSE_OBS/2;
00669                     speedRight = -VITESSE_OBS/2;
00670                     dir = 'G';
00671                 }
00672                 else
00673                 {
00674                     speedLeft = 0;
00675                     speedRight = 0;
00676
00677                     if (table[0] > table[1] && table[0] > SEUIL_DIST_SENSOR) //We have an obstacle on
      our right
00678                     {
00679                         dir = 'G';
00680                         speedLeft = -VITESSE_OBS/2;
00681                         speedRight = VITESSE_OBS/2;
00682                         if (obs%2 == 0)
00683                             obs++;
00684                     }
00685                     else if (table[0] < table[1] && table[1] > SEUIL_DIST_SENSOR) //We have an
      obstacle on left right
00686                     {
00687                         dir = 'D';
00688                         speedLeft = VITESSE_OBS/2;
00689                         speedRight = -VITESSE_OBS/2;
00690                         if (obs%2 == 1)
00691                             obs++;
00692                     }
00693                 }
00694             }
00695             else //No obstacle
00696             {
00697                 speedLeft = VITESSE_OBS;
00698                 speedRight = VITESSE_OBS;
00699                 dir = 'F';
00700                 obs = 0;
00701             }
00702         }
00703         else if (mode == MODE_CAM) //Mode camera
00704         {
00705             dir = 'N';
00706             obs = 0;
00707
00708             if(x < 0 || y < 0) //No object
00709             {
00710                 speedLeft = 0;
00711                 speedRight = 0;
00712             }
00713             else //Try to keep the object on the center
00714             {
00715                 speedLeft = VITESSE_CAM - ((CAMERA_X_MAX/2 - x))/3; // (int) (((float)
      ((x-CAMERA_X_MAX/2)/CAMERA_X_MAX))*500);
00716                 speedRight = VITESSE_CAM + ((CAMERA_X_MAX/2 - x))/3; // (int) (((float)
      (x/CAMERA_X_MAX))*500);
00717             }
00718         }
00719
00720         #if DEBUG_MOTOR
00721         printf("Motor L : %d || R : %d\r\n", speedLeft, speedRight);
00722         #endif
00723
00724         xQueueSend( q_mot_L, ( void * ) &speedLeft,  portMAX_DELAY ); //Send motor left speed
00725         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00726
00727         xQueueSend( q_mot_R, ( void * ) &speedRight,  portMAX_DELAY ); //Send motor right speed
00728         xSemaphoreTake( xSem_Supervision, portMAX_DELAY );
00729
00730     #if MICROROS
00731         MsgToPub.dir = dir;
```

```
00732         MsgToPub.mode = mode;
00733         MsgToPub.speed = speed;
00734         if (!uxQueueMessagesWaiting(qhMR_pub)) //If no data in queue
00735             xQueueSend(qhMR_pub, ( void * ) &MsgToPub, portMAX_DELAY); //Send data
00736     #endif //MICROROS
00737
00738     #if LCD
00739         if (!uxQueueMessagesWaiting(qhLCD)) //If no data in queue
00740         {
00741             pxMessage.data=mode;
00742             pxMessage.command=dir;
00743             xQueueSend( qhLCD, ( void * ) &pxMessage, portMAX_DELAY); //Send data
00744         }
00745     #endif //LCD
00746
00747         vTaskDelay(SAMPLING_PERIOD_ms);
00748     }
00749 #endif //SYNCHRO_EX
00750 }
00751
00752 int main(void)
00753 {
00754   HAL_Init();
00755   SystemClock_Config();
00756   MX_GPIO_Init();
00757   MX_DMA_Init();
00758   MX_USART2_UART_Init();
00759   MX_I2C1_Init();
00760   MX_USART1_UART_Init();
00761
00762   RetargetInit(&huart2); //make printf and scanf work with uart2
00763   printf("%cc%c[2J%c[0;0HTitouan//Jeremy//Louanne//Donald\r\n", 0x1b, 0x1b, 0x1b);
00764
00765   motorCommand_Init();
00766   quadEncoder_Init();
00767   captDistIR_Init();
00768
00769   HAL_Delay(500);
00770
00771 #if VL53
00772   initVL53L0X();
00773   HAL_Delay(500);
00774 #endif //VL53
00775
00776   // Test Ecran LCD
00777 #if LCD
00778   groveLCD_begin(16,2,0); // !! cette fonction prend du temps
00779   HAL_Delay(100);
00780   groveLCD_setCursor(0,0);
00781   groveLCD_setColor(1);
00782   groveLCD_term_printf("Titouan//Jeremy//Louanne");
00783   HAL_Delay(1000);
00784 #endif //LCD
00785
00786   osKernelInitialize();
00787   //defaultTaskHandle = osThreadNew(microros_task, NULL, &defaultTask_attributes);
00788
00789   //Create the diffrent task depending of the config
00790 #if SYNCHRO_EX == EXSTARTUP
00791     #if MICROROS
00792     xTaskCreate( microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, 24,
      NULL);
00793     #endif //MICROROS
00794     xTaskCreate( task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */,
      NULL, 27, NULL);
00795     xTaskCreate( task_Motor_Left, ( const portCHAR * ) "task Mot L", 128 /* stack size */, NULL, 25,
      NULL);
00796     xTaskCreate( task_Motor_Right, ( const portCHAR * ) "task Mot R", 128 /* stack size */, NULL, 26,
      NULL);
00797     #if LCD
00798     xTaskCreate( task_Grove_LCD, ( const portCHAR * ) "task Mot R", 128 /* stack size */, NULL, 23,
      NULL);
00799     #endif
00800 #elif SYNCHRO_EX == EXTEST_UART2
00801     xTaskCreate(test_uart2, ( const portCHAR * ) "task print uart 2", 128 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00802 #elif SYNCHRO_EX == EXCORRECTOR
00803     xTaskCreate(test_motor, ( const portCHAR * ) "task test motor", 128 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00804 #elif SYNCHRO_EX == EXTESTCORRECTOR
00805     xTaskCreate(task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL,
      27, NULL);
00806     xTaskCreate(task_Motor_Left, ( const portCHAR * ) "task Motor Left", 128 /* stack size */, NULL,
      25, NULL);
00807     xTaskCreate(task_Motor_Right, ( const portCHAR * ) "task Motor Right", 128 /* stack size */, NULL,
      26, NULL);
00808 #elif SYNCHRO_EX == EXTEST_VL53
```

```
00809     xTaskCreate(test_vl53, ( const portCHAR * ) "test_vl53", 128 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00810 #elif SYNCHRO_EX == EXTEST_MICROROS
00811     xTaskCreate(microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL,
      tskIDLE_PRIORITY, NULL);
00812 #elif SYNCHRO_EX == EXFINAL
00813     #if MICROROS
00814     xTaskCreate(microros_task, ( const portCHAR * ) "microros_task", 3000 /* stack size */, NULL, 24,
      NULL);
00815     #endif //MICROROS
00816     xTaskCreate(task_Supervision, ( const portCHAR * ) "task Supervision", 128 /* stack size */, NULL,
      27, NULL);
00817     xTaskCreate(task_Motor_Left, ( const portCHAR * ) "task Motor Left", 128 /* stack size */, NULL,
      25, NULL);
00818     xTaskCreate(task_Motor_Right, ( const portCHAR * ) "task Motor Right", 128 /* stack size */, NULL,
      26, NULL);
00819
00820     #if VL53
00821     xTaskCreate(task_VL53, ( const portCHAR * ) "task VL53", 128 /* stack size */, NULL, 23, NULL);
00822     #endif //VL53
00823
00824     #if LCD
00825     xTaskCreate(task_Grove_LCD, ( const portCHAR * ) "task LCD", 128 /* stack size */, NULL, 23,
      NULL);
00826     #endif //LCD
00827 #endif //SYNCHRO_EX
00828
00829     //Create the semaphore
00830     vSemaphoreCreateBinary(xSem_Supervision);
00831
00832     //Init all the queue
00833     q_mot_L = xQueueCreate(1, sizeof(int16_t));
00834     q_mot_R = xQueueCreate(1, sizeof(int16_t));
00835     qhVl53 = xQueueCreate(1, sizeof(int));
00836
00837     qhMR_sub = xQueueCreate(1, sizeof(MicroRosSubMsg));
00838     qhMR_pub = xQueueCreate(1, sizeof(MicroRosPubMsg));
00839     qhLCD = xQueueCreate(1, sizeof(AMessage));
00840
00841     osKernelStart();
00842     while(1){}
00843 }
00844
00845 void test_uart2(void *pvParameters)
00846 {
00847     char buf[100] = "";
00848     for(;;)
00849     {
00850         printf("Veuillez saisir votre nom :\r\n");
00851         scanf("%s", buf);
00852         printf("bonjour et bienvenue %s\r\n", buf);
00853         vTaskDelay(SAMPLING_PERIOD_ms);
00854     }
00855 }
00856
00857 void test_vl53(void *pvParameters)
00858 {
00859     uint16_t val;
00860
00861     for(;;)
00862     {
00863         val = readRangeSingleMillimeters()/10;
00864         printf("Distance capteur : %d\r\n", val);
00865     }
00866 }
00867
00868 void test_motor(void *pvParameters)
00869 {
00870     int16_t  consigne = TEST_CORRECTOR_DUTY;
00871     if (consigne < 0 || consigne > 200)
00872         consigne = 150;
00873     int speed = 0;
00874     int i = 0;
00875
00876     for (;;)
00877     {
00878         #if TEST_LEFT_MOTOR
00879         motorLeft_SetDuty(consigne);
00880         speed = quadEncoder_GetSpeedL();
00881         #else
00882         motorRight_SetDuty(consigne);
00883         speed = quadEncoder_GetSpeedR();
00884         #endif
00885
00886         if(i<NB)
00887         {
00888             tab_speed[i]=speed;
```

```
00889             i++;
00890         }
00891         else
00892             printf("sampling end");
00893         vTaskDelay(SAMPLING_PERIOD_ms);
00894     }
00895 }
00896
00897 //=========================================================================
00898 /*
00899   * @brief  Period elapsed callback in non blocking mode
00900   * @note   This function is called  when TIM1 interrupt took place, inside
00901   * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
00902   * a global variable "uwTick" used as application time base.
00903   * @param  htim : TIM handle
00904   * @retval None
00905   */
00906 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
00907 {
00908   if (htim->Instance == TIM4)
00909   {
00910     HAL_IncTick();
00911   }
00912 }
00913 //=========================================================================
00914 void Error_Handler(void)
00915 {
00916   __disable_irq();
00917   while (1)
00918   {}
00919 }
00920 //=========================================================================
00921 #ifdef  USE_FULL_ASSERT
00922 /*
00923   * @brief  Reports the name of the source file and the source line number
00924   *         where the assert_param error has occurred.
00925   * @param  file: pointer to the source file name
00926   * @param  line: assert_param error line source number
00927   * @retval None
00928   */
00929 void assert_failed(uint8_t *file, uint32_t line)
00930 {
00931   /* USER CODE BEGIN 6 */
00932   /* User can add his own implementation to report the file name and line number,
00933      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
00934   /* USER CODE END 6 */
00935 }
00936 #endif /* USE_FULL_ASSERT */
```

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

: Contain microROS default custom creator for subscriber and publisher

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

Macros

- #define STRING 0

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)

5.32.1 Detailed Description

: Contain microROS default custom creator for subscriber and publisher

Definition in file microROS.c.

5.32.2 Macro Definition Documentation

5.32.2.1 STRING

```
#define STRING 0
```

Definition at line 8 of file microROS.c.

5.32.3 Function Documentation

5.32.3.1 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 10 of file microROS.c.

5.32.3.2 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 29 of file microROS.c.

5.33 microROS.c

Go to the documentation of this file.

```
00001
00006 #include "main.h"
00007
00008 #define STRING 0
00009
00010 void createPublisher(rcl_publisher_t* publisher,
00011     const rcl_node_t* node,
00012     const rosidl_message_type_support_t* type_support,
00013     const char* topic_name,
00014     std_msgs__msg__Int32* msg)
00015 {
00016     rcl_ret_t ret = rclc_publisher_init_default(publisher, node, type_support, topic_name);
00017     printf("Publisher %s is created with result %d\r\n", topic_name, (int)ret);
00018
00019 #if STRING == 1
00020     (*msg).data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
00021     (*msg).data.size = 0;
00022     (*msg).data.capacity = ARRAY_LEN;
00023 #else
00024     (*msg).data = 0;
00025 #endif
00026
00027 }
00028
00029 void createSubscriber(rcl_subscription_t* subscription,
00030     rcl_node_t* node,
00031     const rosidl_message_type_support_t* type_support,
00032     const char* topic_name,
00033     std_msgs__msg__Int32* msg)
00034 {
00035     *subscription = rcl_get_zero_initialized_subscription();
00036
00037     rcl_ret_t ret = rclc_subscription_init_default(subscription, node,
00038         type_support, topic_name);
00039     printf("Subscription %s is created with result %d\r\n", topic_name, (int)ret);
00040
00041 #if STRING == 1
00042     (*msg).data.data = (char * ) malloc(ARRAY_LEN * sizeof(char));
00043     (*msg).data.size = 0;
00044     (*msg).data.capacity = ARRAY_LEN;
00045 #else
00046     (*msg).data = 0;
00047 #endif
00048 }
```

5.34 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 }
```

5.35 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 }
```

5.36 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
```

5.37 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
```

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

: Contain function to add printf and scanf function use the UART2 All credit to Carmine Noviello for this code https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030↩R8/system/src/retarget/

```
#include <_ansi.h>
#include <_syslist.h>
#include <errno.h>
#include <sys/time.h>
#include <sys/times.h>
```

```
#include <limits.h>
#include <signal.h>
#include <../Inc/retarget.h>
#include <stdint.h>
#include <stdio.h>
```

Macros

- #define STDIN_FILENO 0
- #define STDOUT_FILENO 1
- #define STDERR_FILENO 2

Functions

- void RetargetInit (UART_HandleTypeDef *huart)
- int _isatty (int fd)
- int _write (int fd, char *ptr, int len)
- int _close (int fd)
- int _lseek (int fd, int ptr, int dir)
- int _read (int fd, char *ptr, int len)
- int _fstat (int fd, struct stat *st)
- int _getpid (void)
- int _kill (int pid, int sig)

Variables

- UART_HandleTypeDef * gHuart

5.38.1 Detailed Description

: Contain function to add printf and scanf function use the UART2 All credit to Carmine Noviello for this code https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/src/retarget/

Definition in file retarget.c.

5.38.2 Macro Definition Documentation

5.38.2.1 STDERR_FILENO

```
#define STDERR_FILENO 2
```

Definition at line 23 of file retarget.c.

5.38.2.2 STDIN_FILENO

```
#define STDIN_FILENO 0
```

Definition at line 21 of file retarget.c.

5.38.2.3 STDOUT_FILENO

```
#define STDOUT_FILENO 1
```

Definition at line 22 of file retarget.c.

5.38.3 Function Documentation

5.38.3.1 _close()

```
int _close (
            int fd )
```

Definition at line 57 of file retarget.c.

5.38.3.2 _fstat()

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

Definition at line 88 of file retarget.c.

5.38.3.3 _getpid()

```
int _getpid (
            void  )
```

Definition at line 98 of file retarget.c.

5.38.3.4 _isatty()

```
int _isatty (
            int fd )
```

Definition at line 35 of file retarget.c.

5.38.3.5 _kill()

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

Definition at line 103 of file retarget.c.

5.38.3.6 _lseek()

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

Definition at line 65 of file retarget.c.

5.38.3.7 _read()

```
int _read (
              int fd,
              char * ptr,
              int len )
```

Definition at line 74 of file retarget.c.

5.38.3.8 _write()

```
int _write (
              int fd,
              char * ptr,
              int len )
```

Definition at line 43 of file retarget.c.

5.38.3.9 RetargetInit()

```
void RetargetInit (
              UART_HandleTypeDef * huart )
```

Reconfigure stdin, stdout and stderr to use UART

Parameters

huart	Structure wich represent an UART

Definition at line 27 of file retarget.c.

5.38.4 Variable Documentation

5.38.4.1 gHuart

```
UART_HandleTypeDef* gHuart
```

Definition at line 25 of file retarget.c.

5.39 retarget.c

Go to the documentation of this file.

```
00001
00008 #include <_ansi.h>
00009 #include <_syslist.h>
00010 #include <errno.h>
00011 #include <sys/time.h>
00012 #include <sys/times.h>
00013 #include <limits.h>
00014 #include <signal.h>
00015 #include <../Inc/retarget.h>
00016 #include <stdint.h>
00017 #include <stdio.h>
00018
00019 #if !defined(OS_USE_SEMIHOSTING)
00020
00021 #define STDIN_FILENO  0
00022 #define STDOUT_FILENO 1
00023 #define STDERR_FILENO 2
00024
00025 UART_HandleTypeDef *gHuart;
00026
00027 void RetargetInit(UART_HandleTypeDef *huart) {
00028   gHuart = huart;
00029
00030   /* Disable I/O buffering for STDOUT stream, so that
00031    * chars are sent out as soon as they are printed. */
00032   setvbuf(stdout, NULL, _IONBF, 0);
00033 }
00034
00035 int _isatty(int fd) {
00036   if (fd >= STDIN_FILENO && fd <= STDERR_FILENO)
00037     return 1;
00038
00039   errno = EBADF;
00040   return 0;
00041 }
00042
00043 int _write(int fd, char* ptr, int len) {
00044   HAL_StatusTypeDef hstatus;
00045
00046   if (fd == STDOUT_FILENO || fd == STDERR_FILENO) {
00047     hstatus = HAL_UART_Transmit(gHuart, (uint8_t *) ptr, len, HAL_MAX_DELAY);
00048     if (hstatus == HAL_OK)
00049       return len;
00050     else
00051       return EIO;
00052   }
00053   errno = EBADF;
00054   return -1;
00055 }
00056
00057 int _close(int fd) {
00058   if (fd >= STDIN_FILENO && fd <= STDERR_FILENO)
00059     return 0;
00060
00061   errno = EBADF;
00062   return -1;
00063 }
00064
00065 int _lseek(int fd, int ptr, int dir) {
00066   (void) fd;
00067   (void) ptr;
00068   (void) dir;
00069
00070   errno = EBADF;
00071   return -1;
00072 }
00073
00074 int _read(int fd, char* ptr, int len) {
00075   HAL_StatusTypeDef hstatus;
00076
00077   if (fd == STDIN_FILENO) {
00078     hstatus = HAL_UART_Receive(gHuart, (uint8_t *) ptr, 1, HAL_MAX_DELAY);
00079     if (hstatus == HAL_OK)
00080       return 1;
00081     else
00082       return EIO;
00083   }
00084   errno = EBADF;
00085   return -1;
00086 }
00087
00088 int _fstat(int fd, struct stat* st) {
```

```
00089   if (fd >= STDIN_FILENO && fd <= STDERR_FILENO) {
00090     st->st_mode = S_IFCHR;
00091     return 0;
00092   }
00093
00094   errno = EBADF;
00095   return 0;
00096 }
00097
00098 int _getpid(void)
00099 {
00100     return 1;
00101 }
00102
00103 int _kill(int pid, int sig)
00104 {
00105     errno = EINVAL;
00106     return -1;
00107 }
00108
00109 #endif //#if !defined(OS_USE_SEMIHOSTING)
```

5.40 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 EXTI10  --> 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); //PA8 & PA9
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); //PB10
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 EXTI0   --> 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); //PA0 & PA1
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); //PC0
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 - TIMER3 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); //PA6
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); //PC7
00130
00131       // ENABLE MOTEUR : SORTIE LOGIQUE PB3
00132       GPIO_InitStruct.Pin = GPIO_PIN_3;
00133       GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
00134       GPIO_InitStruct.Pull = GPIO_PULLUP;
00135       GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
00136
00137       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct); //PB3
00138       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, 1);
00139
00140       /* ENABLE MOTEUR carte proto
00141       GPIO_InitStruct.Pin = GPIO_PIN_7;
00142       GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
00143       GPIO_InitStruct.Pull = GPIO_NOPULL;
00144
00145       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
00146       HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, 1);*/
```

```
00147 }
00148
00149 /**************************************************************
00150             ADC
00151 ADC1_4  --> PA4
00152 ADC1_8  --> PB0
00153 http://stm32f4-discovery.com/2014/04/library-06-ad-converter-on-stm32f4xx/
00154 **************************************************************/
00155 void HAL_adcir_MspInit(void)
00156 {
00157       GPIO_InitTypeDef  GPIO_InitStruct;
00158       /* Peripheral clock enable */
00159       __ADC1_CLK_ENABLE();
00160
00161       GPIO_InitStruct.Pin = GPIO_PIN_4 ;
00162       GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
00163       GPIO_InitStruct.Pull = GPIO_NOPULL;
00164
00165       HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //PA4
00166
00167       GPIO_InitStruct.Pin = GPIO_PIN_0 ;
00168       GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
00169       GPIO_InitStruct.Pull = GPIO_NOPULL;
00170
00171       HAL_GPIO_Init(GPIOB, &GPIO_InitStruct); //PB0
00172
00173 }
00174
00175
00176 /**************************************************************
00177 * @brief I2C MSP Initialization
00178 * This function configures the hardware resources used in this example
00179 * I2C1 GPIO Configuration
00180 * PB8     ------> I2C1_SCL
00181 * PB9     ------> I2C1_SDA
00182 * @param hi2c: I2C handle pointer
00183 * @retval None
00184 **************************************************************/
00185 void HAL_I2C_MspInit(I2C_HandleTypeDef* hi2c)
00186 {
00187   GPIO_InitTypeDef GPIO_InitStruct = {0};
00188   if(hi2c->Instance==I2C1)
00189   {
00190     __HAL_RCC_GPIOB_CLK_ENABLE();
00191
00192     GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
00193     GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
00194     GPIO_InitStruct.Pull = GPIO_NOPULL;
00195     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00196     GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
00197     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct); //PB8 & PB9
00198
00199     /* Peripheral clock enable */
00200     __HAL_RCC_I2C1_CLK_ENABLE();
00201
00202       HAL_NVIC_SetPriority(I2C1_ER_IRQn, I2C1_ER_IRQ_PRIO, 0);
00203       HAL_NVIC_EnableIRQ(I2C1_ER_IRQn);
00204       HAL_NVIC_SetPriority(I2C1_EV_IRQn, I2C1_EV_IRQ_PRIO, 0);
00205       HAL_NVIC_EnableIRQ(I2C1_EV_IRQn);
00206   }
00207 }
00208
00209 /**************************************************************
00210 * @brief I2C MSP De-Initialization
00211 * This function freeze the hardware resources used in this example
00212 * @param hi2c: I2C handle pointer
00213 * @retval None
00214 **************************************************************/
00215 void HAL_I2C_MspDeInit(I2C_HandleTypeDef* hi2c)
00216 {
00217   if(hi2c->Instance==I2C1)
00218   {
00219   /* USER CODE BEGIN I2C1_MspDeInit 0 */
00220
00221   /* USER CODE END I2C1_MspDeInit 0 */
00222     /* Peripheral clock disable */
00223     __HAL_RCC_I2C1_CLK_DISABLE();
00224
00229     HAL_GPIO_DeInit(GPIOB, GPIO_PIN_6);
00230
00231     HAL_GPIO_DeInit(GPIOB, GPIO_PIN_7);
00232
00233   /* USER CODE BEGIN I2C1_MspDeInit 1 */
00234
00235   /* USER CODE END I2C1_MspDeInit 1 */
00236   }
00237
```

```
00238 }
00239
00240 /**********************************************************************
00241 * @brief UART MSP Initialization
00242 * This function configures the hardware resources used in this example
00243 * USART1 GPIO Configuration
00244 * PB6     ------> USART1_TX
00245 * PA10     ------> USART1_RX
00246 *
00247 * USART2 GPIO Configuration
00248 * PA2     ------> USART2_TX
00249 * PA3     ------> USART2_RX
00250 * @param huart: UART handle pointer
00251 * @retval None
00252 **********************************************************************/
00253 void HAL_UART_MspInit(UART_HandleTypeDef* huart)
00254 {
00255   GPIO_InitTypeDef GPIO_InitStruct = {0};
00256   if(huart->Instance==USART1)
00257   {
00258   /* USER CODE BEGIN USART1_MspInit 0 */
00259
00260   /* USER CODE END USART1_MspInit 0 */
00261     /* Peripheral clock enable */
00262     __HAL_RCC_USART1_CLK_ENABLE();
00263
00264     __HAL_RCC_GPIOA_CLK_ENABLE();
00265
00266     // USART1
00267     GPIO_InitStruct.Pin = GPIO_PIN_10;
00268     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00269     GPIO_InitStruct.Pull = GPIO_NOPULL;
00270     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00271     GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
00272     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //PA10
00273
00274     GPIO_InitStruct.Pin = GPIO_PIN_6;
00275     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00276     GPIO_InitStruct.Pull = GPIO_NOPULL;
00277     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00278     GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
00279     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct); //PB6
00280
00281     /* USART1 DMA Init */
00282     /* USART1_RX Init */
00283     hdma_usart1_rx.Instance = DMA2_Stream2;
00284     hdma_usart1_rx.Init.Channel = DMA_CHANNEL_4;
00285     hdma_usart1_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
00286     hdma_usart1_rx.Init.PeriphInc = DMA_PINC_DISABLE;
00287     hdma_usart1_rx.Init.MemInc = DMA_MINC_ENABLE;
00288     hdma_usart1_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00289     hdma_usart1_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00290     hdma_usart1_rx.Init.Mode = DMA_CIRCULAR;
00291     hdma_usart1_rx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00292     hdma_usart1_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00293     if (HAL_DMA_Init(&hdma_usart1_rx) != HAL_OK)
00294     {
00295       Error_Handler();
00296     }
00297
00298     __HAL_LINKDMA(huart,hdmarx,hdma_usart1_rx);
00299
00300     /* USART1_TX Init */
00301     hdma_usart1_tx.Instance = DMA2_Stream7;
00302     hdma_usart1_tx.Init.Channel = DMA_CHANNEL_4;
00303     hdma_usart1_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
00304     hdma_usart1_tx.Init.PeriphInc = DMA_PINC_DISABLE;
00305     hdma_usart1_tx.Init.MemInc = DMA_MINC_ENABLE;
00306     hdma_usart1_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00307     hdma_usart1_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00308     hdma_usart1_tx.Init.Mode = DMA_NORMAL;
00309     hdma_usart1_tx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00310     hdma_usart1_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00311     if (HAL_DMA_Init(&hdma_usart1_tx) != HAL_OK)
00312     {
00313       Error_Handler();
00314     }
00315
00316     __HAL_LINKDMA(huart,hdmatx,hdma_usart1_tx);
00317
00318     /* USART1 interrupt Init */
00319     HAL_NVIC_SetPriority(USART1_IRQn, 5, 0);
00320     HAL_NVIC_EnableIRQ(USART1_IRQn);
00321   /* USER CODE BEGIN USART1_MspInit 1 */
00322
00323   /* USER CODE END USART1_MspInit 1 */
00324   }
```

```
00325   else if(huart->Instance==USART2)
00326   {
00327   /* USER CODE BEGIN USART2_MspInit 0 */
00328
00329   /* USER CODE END USART2_MspInit 0 */
00330     /* Peripheral clock enable */
00331     __HAL_RCC_USART2_CLK_ENABLE();
00332
00333     __HAL_RCC_GPIOA_CLK_ENABLE();
00334
00335     // USART2
00336     GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
00337     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
00338     GPIO_InitStruct.Pull = GPIO_NOPULL;
00339     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
00340     GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
00341     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //PA4 & PA8
00342
00343     /* USART2 DMA Init */
00344     /* USART2_RX Init */
00345     hdma_usart2_rx.Instance = DMA1_Stream5;
00346     hdma_usart2_rx.Init.Channel = DMA_CHANNEL_4;
00347     hdma_usart2_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
00348     hdma_usart2_rx.Init.PeriphInc = DMA_PINC_DISABLE;
00349     hdma_usart2_rx.Init.MemInc = DMA_MINC_ENABLE;
00350     hdma_usart2_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00351     hdma_usart2_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00352     hdma_usart2_rx.Init.Mode = DMA_CIRCULAR;
00353     hdma_usart2_rx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00354     hdma_usart2_rx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00355     if (HAL_DMA_Init(&hdma_usart2_rx) != HAL_OK)
00356     {
00357       Error_Handler();
00358     }
00359
00360     __HAL_LINKDMA(huart,hdmarx,hdma_usart2_rx);
00361
00362     /* USART2_TX Init */
00363     hdma_usart2_tx.Instance = DMA1_Stream6;
00364     hdma_usart2_tx.Init.Channel = DMA_CHANNEL_4;
00365     hdma_usart2_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
00366     hdma_usart2_tx.Init.PeriphInc = DMA_PINC_DISABLE;
00367     hdma_usart2_tx.Init.MemInc = DMA_MINC_ENABLE;
00368     hdma_usart2_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
00369     hdma_usart2_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
00370     hdma_usart2_tx.Init.Mode = DMA_NORMAL;
00371     hdma_usart2_tx.Init.Priority = DMA_PRIORITY_VERY_HIGH;
00372     hdma_usart2_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
00373     if (HAL_DMA_Init(&hdma_usart2_tx) != HAL_OK)
00374     {
00375       Error_Handler();
00376     }
00377
00378     __HAL_LINKDMA(huart,hdmatx,hdma_usart2_tx);
00379
00380     /* USART2 interrupt Init */
00381     HAL_NVIC_SetPriority(USART2_IRQn, 5, 0);
00382     HAL_NVIC_EnableIRQ(USART2_IRQn);
00383   /* USER CODE BEGIN USART2_MspInit 1 */
00384
00385   /* USER CODE END USART2_MspInit 1 */
00386   }
00387
00388 }
00389
00390 /**********************************************************************
00391 * @brief UART MSP De-Initialization
00392 * This function freeze the hardware resources used in this example
00393 * @param huart: UART handle pointer
00394 * @retval None
00395 **********************************************************************/
00396 void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
00397 {
00398   if(huart->Instance==USART1)
00399   {
00400   /* USER CODE BEGIN USART1_MspDeInit 0 */
00401
00402   /* USER CODE END USART1_MspDeInit 0 */
00403     /* Peripheral clock disable */
00404     __HAL_RCC_USART1_CLK_DISABLE();
00405
00410     HAL_GPIO_DeInit(GPIOA, GPIO_PIN_9|GPIO_PIN_10);
00411
00412     /* USART1 DMA DeInit */
00413     HAL_DMA_DeInit(huart->hdmarx);
00414     HAL_DMA_DeInit(huart->hdmatx);
00415
```

```
00416     /* USART1 interrupt DeInit */
00417     HAL_NVIC_DisableIRQ(USART1_IRQn);
00418   /* USER CODE BEGIN USART1_MspDeInit 1 */
00419
00420   /* USER CODE END USART1_MspDeInit 1 */
00421   }
00422   else if(huart->Instance==USART2)
00423   {
00424   /* USER CODE BEGIN USART2_MspDeInit 0 */
00425
00426   /* USER CODE END USART2_MspDeInit 0 */
00427     /* Peripheral clock disable */
00428     __HAL_RCC_USART2_CLK_DISABLE();
00429
00434     HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
00435
00436     /* USART2 DMA DeInit */
00437     HAL_DMA_DeInit(huart->hdmarx);
00438     HAL_DMA_DeInit(huart->hdmatx);
00439
00440     /* USART2 interrupt DeInit */
00441     HAL_NVIC_DisableIRQ(USART2_IRQn);
00442   /* USER CODE BEGIN USART2_MspDeInit 1 */
00443
00444   /* USER CODE END USART2_MspDeInit 1 */
00445   }
00446
00447 }
00448
00449 /* USER CODE BEGIN 1 */
00450
00451 /* USER CODE END 1 */
```

5.41 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

5.41.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

Definition in file stm32f4xx_hal_timebase_tim.c.

5.41.2 Function Documentation

5.41.2.1 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.

5.41.2.2 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.

5.41.2.3 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.

5.41.3 Variable Documentation

5.41.3.1 htim4

```
TIM_HandleTypeDef htim4
```

Definition at line 28 of file stm32f4xx_hal_timebase_tim.c.

5.42 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
```

5.43 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
```

5.44 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

5.44.1 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.

5.44.2 Function Documentation

5.44.2.1 __attribute__()

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

Definition at line 65 of file syscalls.c.

5.44.2.2 __io_getchar()

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

Definition at line 36 of file syscalls.c.

5.44.2.3 _close()

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

Definition at line 88 of file syscalls.c.

5.44.2.4 _execve()

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

Definition at line 151 of file syscalls.c.

5.44.2.5 _exit()

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

Definition at line 59 of file syscalls.c.

5.44.2.6 _fork()

```
int _fork (
            void  )
```

Definition at line 145 of file syscalls.c.

5.44.2.7 _fstat()

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

Definition at line 94 of file syscalls.c.

5.44.2.8 _getpid()

```
int _getpid (
             void  )
```

Definition at line 48 of file syscalls.c.

5.44.2.9 _isatty()

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

Definition at line 100 of file syscalls.c.

5.44.2.10 _kill()

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

Definition at line 53 of file syscalls.c.

5.44.2.11 _link()

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

Definition at line 139 of file syscalls.c.

5.44.2.12 _lseek()

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

Definition at line 105 of file syscalls.c.

5.44.2.13 _open()

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

Definition at line 110 of file syscalls.c.

5.44.2.14 _stat()

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

Definition at line 133 of file syscalls.c.

5.44.2.15 _times()

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

Definition at line 128 of file syscalls.c.

5.44.2.16 _unlink()

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

Definition at line 122 of file syscalls.c.

5.44.2.17 _wait()

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

Definition at line 116 of file syscalls.c.

5.44.2.18 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

Definition at line 44 of file syscalls.c.

5.44.3 Variable Documentation

5.44.3.1 environ

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

Definition at line 40 of file syscalls.c.

5.45 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 }
```

5.46 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

5.46.1 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.

5.46.2 Function Documentation

5.46.2.1 _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.

5.46.3 Variable Documentation

5.46.3.1 __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.

5.47 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 }
```

5.48 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 }
```

5.49 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 //================================================================
```

5.50 C:/Users/Administrator/Documents/GitHub/ENIB_Robot_Mobile_↩ Ros2/WORKSPACE_F411_uROS6/base_robot/Core/Src/VL53L0X.c File Reference

: VL53L0X API STSW-IMG005 portage Most of the functionality of this library is based on the VL53L0X API provided by ST (STSW-IMG005), and some of the explanatory comments are quoted or paraphrased from the API source code, API user manual (UM2039), and the VL53L0X datasheet.

```
#include "main.h"
#include <unistd.h>
#include <stdint.h>
#include "VL53L0X.h"
#include "drv_i2c.h"
```

Macros

- #define ACTIVE_WHILE 0
- #define IO_2V8 1
- #define ADDRESS 0x52

Functions

- uint8_t performSingleRefCalibration (uint8_t vhv_init_byte)
- void writeReg (uint8_t reg, uint8_t value)
- void writeReg16Bit (uint8_t reg, uint16_t value)
- void writeReg32Bit (uint8_t reg, uint32_t value)
- uint8_t readReg (uint8_t reg)
- uint16_t readReg16Bit (uint8_t reg)
- uint32_t readReg32Bit (uint8_t reg)
- void writeMulti (uint8_t reg, uint8_t const *src, uint8_t count)
- void setAddress (uint8_t new_addr)
- uint8_t getAddress ()
- uint8_t initVL53L0X ()
- uint8_t setSignalRateLimit (float limit_Mcps)
- float getSignalRateLimit (void)
- uint16_t readRangeSingleMillimeters ()

Variables

- uint8_t g_i2cAddr = ADDRESS
- uint8_t g_stopVariable
- uint16_t addr_read = 0

5.50.1 Detailed Description

: VL53L0X API STSW-IMG005 portage Most of the functionality of this library is based on the VL53L0X API provided by ST (STSW-IMG005), and some of the explanatory comments are quoted or paraphrased from the API source code, API user manual (UM2039), and the VL53L0X datasheet.

Definition in file VL53L0X.c.

5.50.2 Macro Definition Documentation

5.50.2.1 ACTIVE_WHILE

```
#define ACTIVE_WHILE 0
```

Definition at line 19 of file VL53L0X.c.

5.50.2.2 ADDRESS

```
#define ADDRESS 0x52
```

Definition at line 21 of file VL53L0X.c.

5.50.2.3 IO_2V8

```
#define IO_2V8 1
```

Definition at line 20 of file VL53L0X.c.

5.50.3 Function Documentation

5.50.3.1 getAddress()

```
uint8_t getAddress (
             void  )
```

Definition at line 103 of file VL53L0X.c.

5.50.3.2 getSignalRateLimit()

```
float getSignalRateLimit (
             void  )
```

Definition at line 329 of file VL53L0X.c.

5.50.3.3 initVL53L0X()

```
uint8_t initVL53L0X ( )
```

Definition at line 115 of file VL53L0X.c.

5.50.3.4 performSingleRefCalibration()

```
uint8_t performSingleRefCalibration (
            uint8_t vhv_init_byte )
```

Definition at line 369 of file VL53L0X.c.

5.50.3.5 readRangeSingleMillimeters()

```
uint16_t readRangeSingleMillimeters ( )
```

Definition at line 342 of file VL53L0X.c.

5.50.3.6 readReg()

```
uint8_t readReg (
            uint8_t reg )
```

Definition at line 62 of file VL53L0X.c.

5.50.3.7 readReg16Bit()

```
uint16_t readReg16Bit (
            uint8_t reg )
```

Definition at line 70 of file VL53L0X.c.

5.50.3.8 readReg32Bit()

```
uint32_t readReg32Bit (
            uint8_t reg )
```

Definition at line 79 of file VL53L0X.c.

5.50.3.9 setAddress()

```
void setAddress (
            uint8_t new_addr )
```

Definition at line 98 of file VL53L0X.c.

5.50.3.10 setSignalRateLimit()

```
uint8_t setSignalRateLimit (
            float limit_Mcps )
```

Definition at line 319 of file VL53L0X.c.

5.50.3.11 writeMulti()

```
void writeMulti (
            uint8_t reg,
            uint8_t const * src,
            uint8_t count )
```

Definition at line 89 of file VL53L0X.c.

5.50.3.12 writeReg()

```
void writeReg (
            uint8_t reg,
            uint8_t value )
```

Definition at line 39 of file VL53L0X.c.

5.50.3.13 writeReg16Bit()

```
void writeReg16Bit (
            uint8_t reg,
            uint16_t value )
```

Definition at line 44 of file VL53L0X.c.

5.50.3.14 writeReg32Bit()

```
void writeReg32Bit (
            uint8_t reg,
            uint32_t value )
```

Definition at line 52 of file VL53L0X.c.

5.50.4 Variable Documentation

5.50.4.1 addr_read

```
uint16_t addr_read = 0
```

Definition at line 29 of file VL53L0X.c.

5.50.4.2 g_i2cAddr

```
uint8_t g_i2cAddr = ADDRESS
```

Definition at line 26 of file VL53L0X.c.

5.50.4.3 g_stopVariable

```
uint8_t g_stopVariable
```

Definition at line 27 of file VL53L0X.c.

5.51 VL53L0X.c

Go to the documentation of this file.

```
00001
00010 //"${workspace_loc:/${ProjName}/Drivers/vl53l0x}" into include path of c++ buider properties
00011
00012 #include "main.h"
00013 #include <unistd.h>
00014
00015 #include <stdint.h>
00016 #include "VL53L0X.h"
00017 #include "drv_i2c.h"
00018
00019 #define ACTIVE_WHILE 0
00020 #define IO_2V8 1
00021 #define ADDRESS 0x52
00022
00023 //------------------------------------------------------------
00024 // Local variables within this file (private)
00025 //------------------------------------------------------------
00026 uint8_t g_i2cAddr = ADDRESS;
00027 uint8_t g_stopVariable; // read by init and used when starting measurement; is StopVariable field of
      VL53L0X_DevData_t structure in API
00028
00029 uint16_t addr_read = 0;
00030
00031 //------------------------------------------------------------
00032 // Locally used functions (private)
00033 //------------------------------------------------------------
00034 uint8_t performSingleRefCalibration(uint8_t vhv_init_byte);
00035 //------------------------------------------------------------
00036 // I2C communication Functions
00037 //------------------------------------------------------------
00038 // Write an 8-bit register
00039 void writeReg(uint8_t reg, uint8_t value) {
00040     i2c1_WriteRegBuffer(g_i2cAddr,reg,&value,1);
00041 }
00042
00043 // Write a 16-bit register
00044 void writeReg16Bit(uint8_t reg, uint16_t value){
00045     uint8_t tab[2];
00046     tab[0]= ((value » 8));
00047     tab[1] = ((value ) & 0xFF);
00048     i2c1_WriteRegBuffer(g_i2cAddr,reg,tab,2);
00049 }
00050
00051 // Write a 32-bit register
00052 void writeReg32Bit(uint8_t reg, uint32_t value){
00053     uint8_t tab[4];
00054         tab[3]= ((value » 24) & 0xFF);
00055         tab[2]= ((value » 16) & 0xFF);
00056         tab[1]= ((value » 8) & 0xFF);
00057         tab[0] = ((value ) & 0xFF);
00058         i2c1_WriteRegBuffer(g_i2cAddr,reg,tab,4);
00059 }
00060
00061 // Read an 8-bit register
00062 uint8_t readReg(uint8_t reg) {
00063     uint8_t value=0;
00064     i2c1_WriteBuffer(g_i2cAddr, &reg, 1);
00065     i2c1_ReadBuffer(g_i2cAddr|0x01, &value, 1);
```

```
00066     return value;
00067 }
00068
00069 // Read a 16-bit register
00070 uint16_t readReg16Bit(uint8_t reg) {
00071     uint8_t tab[2];
00072     i2c1_WriteBuffer(g_i2cAddr, &reg, 1);
00073     i2c1_ReadBuffer(g_i2cAddr|0x01, tab, 2);
00074     uint16_t value= ((uint16_t)tab[0] « 8) | (uint16_t)tab[1];
00075     return value;
00076 }
00077
00078 // Read a 32-bit register
00079 uint32_t readReg32Bit(uint8_t reg) {
00080   uint8_t tab[4];
00081   i2c1_WriteBuffer(g_i2cAddr, &reg, 1);
00082   i2c1_ReadBuffer(g_i2cAddr|0x01, tab, 4);
00083   uint32_t value= (tab[3] « 24) | (tab[2] « 16 ) | (tab[1] « 8) | tab[0];
00084   return value;
00085 }
00086
00087 // Write an arbitrary number of bytes from the given array to the sensor,
00088 // starting at the given register
00089 void writeMulti(uint8_t reg, uint8_t const *src, uint8_t count){
00090   while ( count-- > 0 ) {
00091     i2c1_WriteRegBuffer(g_i2cAddr,reg,(uint8_t *)src,1);
00092   }
00093 }
00094
00095
00096
00097 // Public Methods //////////////////////////////////////////////////////////////
00098 void setAddress(uint8_t new_addr) {
00099   writeReg( I2C_SLAVE_DEVICE_ADDRESS, (new_addr»1) & 0x7F );
00100   g_i2cAddr = new_addr;
00101 }
00102
00103 uint8_t getAddress() {
00104   return g_i2cAddr;
00105 }
00106
00107 // Initialize sensor using sequence based on VL53L0X_DataInit(),
00108 // VL53L0X_StaticInit(), and VL53L0X_PerformRefCalibration().
00109 // This function does not perform reference SPAD calibration
00110 // (VL53L0X_PerformRefSpadManagement()), since the API user manual says that it
00111 // is performed by ST on the bare modules; it seems like that should work well
00112 // enough unless a cover glass is added.
00113 // If io_2v8 (optional) is true or not given, the sensor is configured for 2V8
00114 // mode.
00115 uint8_t initVL53L0X( ){
00116   // VL53L0X_DataInit() begin
00117
00118
00119   // sensor uses 1V8 mode for I/O by default; switch to 2V8 mode if necessary
00120   if (IO_2V8)
00121   {
00122     writeReg(VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV,
00123     readReg(VHV_CONFIG_PAD_SCL_SDA__EXTSUP_HV) | 0x01); // set bit 0
00124   }
00125
00126   // "Set I2C standard mode"
00127   writeReg(0x88, 0x00);
00128
00129   writeReg(0x80, 0x01);
00130   writeReg(0xFF, 0x01);
00131   writeReg(0x00, 0x00);
00132   g_stopVariable = readReg(0x91);
00133   writeReg(0x00, 0x01);
00134   writeReg(0xFF, 0x00);
00135   writeReg(0x80, 0x00);
00136
00137   // disable SIGNAL_RATE_MSRC (bit 1) and SIGNAL_RATE_PRE_RANGE (bit 4) limit checks
00138   writeReg(MSRC_CONFIG_CONTROL, readReg(MSRC_CONFIG_CONTROL) | 0x12);
00139
00140   // set final range signal rate limit to 0.25 MCPS (million counts per second)
00141   setSignalRateLimit(0.25);
00142
00143   writeReg(SYSTEM_SEQUENCE_CONFIG, 0xFF);
00144
00145   // VL53L0X_DataInit() end
00146
00147   // VL53L0X_StaticInit() begin
00148
00149   // The SPAD map (RefGoodSpadMap) is read by VL53L0X_get_info_from_device() in
00150   // the API, but the same data seems to be more easily readable from
00151   // GLOBAL_CONFIG_SPAD_ENABLES_REF_0 through _6, so read it from there
00152
```

```
00153   // -- VL53L0X_set_reference_spads() begin (assume NVM values are valid)
00154
00155   writeReg(0xFF, 0x01);
00156   writeReg(DYNAMIC_SPAD_REF_EN_START_OFFSET, 0x00);
00157   writeReg(DYNAMIC_SPAD_NUM_REQUESTED_REF_SPAD, 0x2C);
00158   writeReg(0xFF, 0x00);
00159   writeReg(GLOBAL_CONFIG_REF_EN_START_SELECT, 0xB4);
00160
00161
00162   // -- VL53L0X_set_reference_spads() end
00163
00164   // -- VL53L0X_load_tuning_settings() begin
00165   // DefaultTuningSettings from vl53l0x_tuning.h
00166
00167   writeReg(0xFF, 0x01);
00168   writeReg(0x00, 0x00);
00169
00170   writeReg(0xFF, 0x00);
00171   writeReg(0x09, 0x00);
00172   writeReg(0x10, 0x00);
00173   writeReg(0x11, 0x00);
00174
00175   writeReg(0x24, 0x01);
00176   writeReg(0x25, 0xFF);
00177   writeReg(0x75, 0x00);
00178
00179   writeReg(0xFF, 0x01);
00180   writeReg(0x4E, 0x2C);
00181   writeReg(0x48, 0x00);
00182   writeReg(0x30, 0x20);
00183
00184   writeReg(0xFF, 0x00);
00185   writeReg(0x30, 0x09);
00186   writeReg(0x54, 0x00);
00187   writeReg(0x31, 0x04);
00188   writeReg(0x32, 0x03);
00189   writeReg(0x40, 0x83);
00190   writeReg(0x46, 0x25);
00191   writeReg(0x60, 0x00);
00192   writeReg(0x27, 0x00);
00193   writeReg(0x50, 0x06);
00194   writeReg(0x51, 0x00);
00195   writeReg(0x52, 0x96);
00196   writeReg(0x56, 0x08);
00197   writeReg(0x57, 0x30);
00198   writeReg(0x61, 0x00);
00199   writeReg(0x62, 0x00);
00200   writeReg(0x64, 0x00);
00201   writeReg(0x65, 0x00);
00202   writeReg(0x66, 0xA0);
00203
00204   writeReg(0xFF, 0x01);
00205   writeReg(0x22, 0x32);
00206   writeReg(0x47, 0x14);
00207   writeReg(0x49, 0xFF);
00208   writeReg(0x4A, 0x00);
00209
00210   writeReg(0xFF, 0x00);
00211   writeReg(0x7A, 0x0A);
00212   writeReg(0x7B, 0x00);
00213   writeReg(0x78, 0x21);
00214
00215   writeReg(0xFF, 0x01);
00216   writeReg(0x23, 0x34);
00217   writeReg(0x42, 0x00);
00218   writeReg(0x44, 0xFF);
00219   writeReg(0x45, 0x26);
00220   writeReg(0x46, 0x05);
00221   writeReg(0x40, 0x40);
00222   writeReg(0x0E, 0x06);
00223   writeReg(0x20, 0x1A);
00224   writeReg(0x43, 0x40);
00225
00226   writeReg(0xFF, 0x00);
00227   writeReg(0x34, 0x03);
00228   writeReg(0x35, 0x44);
00229
00230   writeReg(0xFF, 0x01);
00231   writeReg(0x31, 0x04);
00232   writeReg(0x4B, 0x09);
00233   writeReg(0x4C, 0x05);
00234   writeReg(0x4D, 0x04);
00235
00236   writeReg(0xFF, 0x00);
00237   writeReg(0x44, 0x00);
00238   writeReg(0x45, 0x20);
00239   writeReg(0x47, 0x08);
```

```
00240   writeReg(0x48, 0x28);
00241   writeReg(0x67, 0x00);
00242   writeReg(0x70, 0x04);
00243   writeReg(0x71, 0x01);
00244   writeReg(0x72, 0xFE);
00245   writeReg(0x76, 0x00);
00246   writeReg(0x77, 0x00);
00247
00248   writeReg(0xFF, 0x01);
00249   writeReg(0x0D, 0x01);
00250
00251   writeReg(0xFF, 0x00);
00252   writeReg(0x80, 0x01);
00253   writeReg(0x01, 0xF8);
00254
00255   writeReg(0xFF, 0x01);
00256   writeReg(0x8E, 0x01);
00257   writeReg(0x00, 0x01);
00258   writeReg(0xFF, 0x00);
00259   writeReg(0x80, 0x00);
00260
00261   // -- VL53L0X_load_tuning_settings() end
00262
00263   // "Set interrupt config to new sample ready"
00264   // -- VL53L0X_SetGpioConfig() begin
00265
00266   writeReg(SYSTEM_INTERRUPT_CONFIG_GPIO, 0x04);
00267   writeReg(GPIO_HV_MUX_ACTIVE_HIGH, readReg(GPIO_HV_MUX_ACTIVE_HIGH) & ~0x10); // active low
00268   writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);
00269
00270   // -- VL53L0X_SetGpioConfig() end
00271
00272
00273
00274   // "Disable MSRC and TCC by default"
00275   // MSRC = Minimum Signal Rate Check
00276   // TCC = Target CentreCheck
00277   // -- VL53L0X_SetSequenceStepEnable() begin
00278
00279   writeReg(SYSTEM_SEQUENCE_CONFIG, 0xE8);
00280
00281   // -- VL53L0X_SetSequenceStepEnable() end
00282
00283
00284
00285   // VL53L0X_StaticInit() end
00286
00287   // VL53L0X_PerformRefCalibration() begin (VL53L0X_perform_ref_calibration())
00288
00289   // -- VL53L0X_perform_vhv_calibration() begin
00290
00291   writeReg(SYSTEM_SEQUENCE_CONFIG, 0x01);
00292   if (performSingleRefCalibration(0x40)) { return 1; }
00293
00294   // -- VL53L0X_perform_vhv_calibration() end
00295
00296   // -- VL53L0X_perform_phase_calibration() begin
00297
00298   writeReg(SYSTEM_SEQUENCE_CONFIG, 0x02);
00299   if (performSingleRefCalibration(0x00)) { return 1; }
00300
00301   // -- VL53L0X_perform_phase_calibration() end
00302
00303   // "restore the previous Sequence Config"
00304   writeReg(SYSTEM_SEQUENCE_CONFIG, 0xE8);
00305
00306   // VL53L0X_PerformRefCalibration() end
00307
00308   return 0;
00309 }
00310
00311 // Set the return signal rate limit check value in units of MCPS (mega counts
00312 // per second). "This represents the amplitude of the signal reflected from the
00313 // target and detected by the device"; setting this limit presumably determines
00314 // the minimum measurement necessary for the sensor to report a valid reading.
00315 // Setting a lower limit increases the potential range of the sensor but also
00316 // seems to increase the likelihood of getting an inaccurate reading because of
00317 // unwanted reflections from objects other than the intended target.
00318 // Defaults to 0.25 MCPS as initialized by the ST API and this library.
00319 uint8_t setSignalRateLimit(float limit_Mcps)
00320 {
00321   if (limit_Mcps < 0 || limit_Mcps > 511.99) { return false; }
00322
00323   // Q9.7 fixed point format (9 integer bits, 7 fractional bits)
00324   writeReg16Bit(FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT, limit_Mcps * (1 << 7));
00325   return 0;
00326 }
```

```
00327
00328 // Get the return signal rate limit check value in MCPS
00329 float getSignalRateLimit(void)
00330 {
00331   return (float)readReg16Bit(FINAL_RANGE_CONFIG_MIN_COUNT_RATE_RTN_LIMIT) / (1 « 7);
00332 }
00333
00334
00335
00336
00337
00338 // Performs a single-shot range measurement and returns the reading in
00339 // millimeters
00340 // based on VL53L0X_PerformSingleRangingMeasurement()
00341 // extraStats provides additional info for this measurment. Set to 0 if not needed.
00342 uint16_t readRangeSingleMillimeters( /*statInfo_t *extraStats */) {
00343   writeReg(0x80, 0x01);
00344   writeReg(0xFF, 0x01);
00345   writeReg(0x00, 0x00);
00346   writeReg(0x91, g_stopVariable);
00347   writeReg(0x00, 0x01);
00348   writeReg(0xFF, 0x00);
00349   writeReg(0x80, 0x00);
00350   writeReg(SYSRANGE_START, 0x01);
00351
00352   uint16_t temp;
00353
00354   if (ACTIVE_WHILE)
00355   {
00356       while (readReg(SYSRANGE_START) & 0x01){};
00357       while ((readReg(RESULT_INTERRUPT_STATUS) & 0x07) == 0){};
00358   }
00359   // assumptions: Linearity Corrective Gain is 1000 (default);
00360   // fractional ranging is not enabled
00361   temp = readReg16Bit(RESULT_RANGE_STATUS + 10);
00362   temp+=0;
00363
00364   return temp;
00365 }
00366
00367
00368 // based on VL53L0X_perform_single_ref_calibration()
00369 uint8_t performSingleRefCalibration(uint8_t vhv_init_byte)
00370 {
00371   writeReg(SYSRANGE_START, 0x01 | vhv_init_byte); // VL53L0X_REG_SYSRANGE_MODE_START_STOP
00372
00373   if (ACTIVE_WHILE)
00374       while ((readReg(RESULT_INTERRUPT_STATUS) & 0x07) == 0){};
00375
00376   writeReg(SYSTEM_INTERRUPT_CLEAR, 0x01);
00377
00378   writeReg(SYSRANGE_START, 0x00);
00379
00380   return 0;
00381 }
```

Index