cdfr2020CarteCerveauProg

Generated by Doxygen 1.8.17

1 Module Index		1
1.1 Modules		1
2 Data Structure Index		3
2.1 Data Structures		3
3 File Index		5
3.1 File List		5
4 Module Documentation		7
4.1 actuator_tim		7
4.1.1 Detailed Description		7
4.2 arm		8
4.2.1 Detailed Description		8
4.3 flag		9
4.3.1 Detailed Description		9
4.4 arm_limit_switch		10
4.4.1 Detailed Description		10
4.5 flag_limit_switch		11
4.5.1 Detailed Description		11
4.6 flash_memory		12
4.6.1 Detailed Description		12
4.6.2 Function Documentation		12
4.6.2.1 flash_program_data()		12
4.6.2.2 flash_read_data()		13
4.7 I2C		14
4.7.1 Detailed Description		14
4.8 tof_shiftr		15
4.8.1 Detailed Description		15
4.9 Range Ranging Profile		16
4.9.1 Detailed Description		16
4.10 tof_tim		17
4.10.1 Detailed Description		17
4.11 debug_uart		18
4.11.1 Detailed Description	1	18
5 Data Structure Documentation		19
5.1 can_msg_buffer_list_t Struct Reference		19
5.1.1 Detailed Description		19
5.2 Can_rx_msg Struct Reference		19
5.2.1 Detailed Description	2	20
5.3 Can_tx_msg Struct Reference	2	20
5.4 can_tx_msg Struct Reference	2	21
5.4.1 Detailed Description		21

	5.5 VL53L0X_Calibration_Parameter_S Struct Reference	21
	5.5.1 Detailed Description	21
6	File Documentation	23
	6.1 lowlevel/canmsgs.c File Reference	23
	6.1.1 Detailed Description	23
	6.1.2 Function Documentation	24
	6.1.2.1 can_setup()	24
	6.2 lowlevel/exti.c File Reference	24
	6.2.1 Detailed Description	24
	6.2.2 Function Documentation	25
	6.2.2.1 _limit_switch_init()	25
	6.2.2.2 exti9_5_isr()	25
	6.2.2.3 exti_setup()	25
	6.3 lowlevel/flash.c File Reference	26
	6.3.1 Detailed Description	26
	6.4 lowlevel/include/actuator.h File Reference	26
	6.4.1 Detailed Description	28
	6.4.2 Macro Definition Documentation	28
	6.4.2.1 PWM_PERIOD	28
	6.4.2.2 PWM_PRESCALE	28
	6.4.3 Enumeration Type Documentation	28
	6.4.3.1 actuator_sel	28
	6.4.4 Function Documentation	28
	6.4.4.1 actuator_set()	28
	6.4.4.2 actuator_setup()	29
	6.5 lowlevel/include/canmsgs.h File Reference	29
	6.5.1 Detailed Description	30
	6.5.2 Function Documentation	31
	6.5.2.1 _can_msg_buffer_append()	31
	6.5.2.2 can_msg_buffer_pop()	31
	6.5.2.3 can_setup()	31
	6.5.2.4 cec_can_isr()	31
	6.5.2.5 receive()	32
	6.5.2.6 transmit()	32
	6.6 lowlevel/include/clock.h File Reference	32
	6.6.1 Detailed Description	33
	6.6.2 Function Documentation	33
	6.6.2.1 clock_setup()	33
	6.6.2.2 delay_ms()	33
	6.7 lowlevel/include/exti.h File Reference	33
	6.7.1 Detailed Description	34

6.7.2 Function Documentation	35
6.7.2.1 _limit_switch_init()	35
6.7.2.2 exti9_5_isr()	35
6.7.2.3 exti_setup()	35
6.8 lowlevel/include/gpio.h File Reference	36
6.8.1 Detailed Description	36
6.8.2 Function Documentation	36
6.8.2.1pulse()	37
6.8.2.2 _gpio_setup_pin()	38
6.8.2.3 _gpio_setup_pin_af()	38
6.9 lowlevel/include/i2c.h File Reference	39
6.9.1 Detailed Description	40
6.9.2 Typedef Documentation	40
6.9.2.1 I2C_status	40
6.9.3 Enumeration Type Documentation	40
6.9.3.1 I2C_Status_E	40
6.9.4 Function Documentation	40
6.9.4.1 i2c_read7()	40
6.9.4.2 i2c_setup()	41
6.9.4.3 i2c_write7()	41
6.10 lowlevel/include/timer.h File Reference	42
6.10.1 Detailed Description	42
6.10.2 Function Documentation	43
6.10.2.1 _timer_setup()	43
6.10.2.2 _timer_setup_output_c()	43
6.10.2.3 _timer_start()	43
6.11 lowlevel/include/tof.h File Reference	44
6.11.1 Detailed Description	46
6.11.2 Function Documentation	46
6.11.2.1 _shift_reg()	46
6.11.2.2 _shift_reg_init()	46
6.11.2.3 _tof_1_setup()	46
6.11.2.4 _tof_calibration()	47
6.11.2.5 _tof_config()	47
6.11.2.6 _tof_init_struct()	48
6.11.2.7 _tof_poke()	48
6.11.2.8 _tof_set_address()	48
6.11.2.9 _tof_setup_addr()	49
6.11.2.10 _tof_setup_calib()	49
6.11.2.11 tof_perform_measure()	49
6.11.2.12 tof_print_calib_info()	50
6.11.2.13 tof_print_data_measure()	50

6.11.2.14 tof_print_device_info()	51
6.11.2.15 tof_print_device_mode()	51
6.11.2.16 tof_print_int_status()	51
6.11.2.17 tof_print_PAL_state()	52
6.11.2.18 tof_print_ranging_status()	52
6.11.2.19 tof_reset()	53
6.11.2.20 tof_setup()	53
6.12 lowlevel/include/tof_timer.h File Reference	53
6.12.1 Detailed Description	54
6.12.2 Macro Definition Documentation	54
6.12.2.1 TOF_TIM_PERIOD	54
6.12.2.2 TOF_TIM_PRESCALER	54
6.12.3 Function Documentation	55
6.12.3.1 tim4_isr()	55
6.12.3.2 timer_setup_interrupt()	55
6.13 lowlevel/include/uart.h File Reference	55
6.13.1 Detailed Description	56
6.13.2 Function Documentation	57
6.13.2.1 _read()	57
6.13.2.2 _write()	57
6.13.2.3 uart_setup()	58
Index	59

# **Module Index**

## 1.1 Modules

Here is a list of all modules:

actuator_tim													 			 						7
arm													 			 						8
flag													 			 						9
arm_limit_switch .													 			 						10
flag_limit_switch .													 			 						-11
flash_memory													 			 						12
I2C																						
tof_shiftr													 			 						15
Range Ranging Pr	ofile	Э											 			 						16
tof_tim													 			 						17
debug uart													 			 						18

2 Module Index

# **Data Structure Index**

## 2.1 Data Structures

Here are the data structures with brief descriptions:

can_msg_buffer_list_t	
FIFO linked list to store impeding CAN messages (software side)	19
Can_rx_msg	
Frame of stantard received CAN messages	19
Can_tx_msg	20
can_tx_msg	
Frame of stantard transmitted CAN messages	21
VL53L0X_Calibration_Parameter_S	
Storage of all Calibration Parameter for the TOF (VL53L1X)	21

Data Structure Index

# File Index

## 3.1 File List

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

lowlevel/canmsgs.c	
This implements the setup of CAN protocol to allow F3, F4 and other potential computers to com-	
<pre>municate Source: https://www.rhye.org/post/stm32-with-opencm3-3-can.</pre>	bus/
23	
lowlevel/exti.c	
This implements the setup of the sensors linked to the actuators: the arm and the flag $\dots \dots$	24
lowlevel/flash.c	
This implements function to read and flash data in the rom sector	26
lowlevel/include/actuator.h	
This implements the setup of the actuators: the arm and the flag	26
lowlevel/include/canmsgs.h	
This implements the setup of communication between F3, F4 and other potential computers us-	
<pre>ing CAN protocol. Source: https://www.rhye.org/post/stm32-with-opencm3- 29</pre>	-3-canbus
lowlevel/include/clock.h	
This implements the setup of the system clock, acces function (debug) and temporal fonction	
(delay)	
32	
lowlevel/include/exti.h	
This implements the setup of the sensors linked to the actuators: the arm and the flag $\dots \dots$	33
lowlevel/include/flash.h	??
lowlevel/include/gpio.h	
This implements the setup of a gpio pin	
36	
lowlevel/include/i2c.h	
This implements the setup of an I2C peripheral	39
lowlevel/include/timer.h	
This implements the functions required setup a timer and its output channel	
42	
lowlevel/include/tof.h	
This implements all needed peripheral to use the tof	44
lowlevel/include/tof_timer.h	
This implements a routine for periodic call of a function	53
lowlevel/include/uart.h	
This implements the setup of the actuators: the arm and the flag	55

6 File Index

## **Module Documentation**

## 4.1 actuator\_tim

Internal timer used to pilot the motors of the actuators with a PWM. Both use TIM3.

#### **Macros**

- #define ACTUATOR TIM RCC RCC TIM3
- #define ACTUATOR\_TIM TIM3
- #define COMM\_RCC\_USART RCC\_USART1
- #define COMM\_USART USART1
- #define **COMM\_UART\_SPEED** (9600)
- #define COMM\_PORT\_TX GPIOA
- #define COMM\_PORT\_TX\_RCC RCC\_GPIOB
- #define COMM\_PIN\_TX GPIO9
- #define COMM\_AF\_TX GPIO\_AF7
- #define COMM\_PORT\_RX GPIOA
- #define COMM\_PORT\_RX\_RCC RCC\_GPIOB
- #define COMM\_PIN\_RX GPIO10
- #define COMM\_AF\_RX GPIO\_AF7
- #define COMM UART EXTI EXTI25
- #define COMM\_UART\_NVIC NVIC\_USART1\_IRQ

## 4.1.1 Detailed Description

Internal timer used to pilot the motors of the actuators with a PWM. Both use TIM3.

Uart used for communication between devices.

Two channels are used for the ARM and FLAG

Baudrate is 9600

## 4.2 arm

Definitions for the arm.

#### **Macros**

- #define ARM\_GPIO\_RCC\_EN RCC\_GPIOC
- #define ARM\_PORT\_EN GPIOC
- #define ARM\_PIN\_EN GPIO7
- #define ARM AF GPIO AF2
- #define ARM\_OC\_ID TIM\_OC2
- #define ARM\_OC\_MODE TIM\_OCM\_PWM1
- #define ARM\_GPIO\_RCC\_DIR\_1 RCC\_GPIOB
- #define ARM PORT DIR 1 GPIOB
- #define ARM\_PIN\_DIR\_1 GPIO12
- #define ARM\_GPIO\_RCC\_DIR\_2 RCC\_GPIOB
- #define ARM\_PORT\_DIR\_2 GPIOB
- #define ARM PIN DIR 2 GPIO13
- #define ARM INIT DIR 0
- #define ARM\_INVERT\_DIR (-1)

## 4.2.1 Detailed Description

Definitions for the arm.

EN stands for enable (output of the PWM signal)

We use OC\_ID to select a specific channel of the output comparator as a PWM\_output

DIR\_1/2 stands for direction (boolean value)

INIT\_DIR is the initial direction of the motor INVERT\_DIR allows to define the forward direction in motor\_set (must be 1 or -1) Pinmap used here: EN on PC7 (with TIM3\_CH2), DIR\_1 on PB12, DIR\_2 on PB13

4.3 flag 9

## 4.3 flag

Definitions for the flag.

#### **Macros**

- #define FLAG\_GPIO\_RCC\_EN RCC\_GPIOC
- #define FLAG PORT EN GPIOC
- #define FLAG\_PIN\_EN GPIO6
- #define FLAG AF GPIO AF2
- #define FLAG\_OC\_ID TIM\_OC1
- #define FLAG\_OC\_MODE TIM\_OCM\_PWM1
- #define FLAG\_GPIO\_RCC\_DIR\_1 RCC\_GPIOB
- #define FLAG\_PORT\_DIR\_1 GPIOB
- #define FLAG PIN DIR 1 GPIO14
- #define FLAG\_GPIO\_RCC\_DIR\_2 RCC\_GPIOB
- #define FLAG\_PORT\_DIR\_2 GPIOB
- #define FLAG PIN DIR 2 GPIO15
- #define FLAG\_INIT\_DIR 0
- #define FLAG\_INVERT\_DIR (-1)

## 4.3.1 Detailed Description

Definitions for the flag.

EN stands for enable (output of the PWM signal)

We use  $OC\_ID$  to select a specific channel of the output comparator as a  $PWM\_output$ 

DIR 1/2 stands for direction (boolean value)

INIT\_DIR is the initial direction of the motor INVERT\_DIR allows to define the forward direction in motor\_set (must be 1 or -1) Pinmap used here: EN on PC6 (with TIM3\_CH1), DIR\_1 on PB14, DIR\_2 on PB15

## 4.4 arm limit switch

INterruption when the actuator reaches limit switch.

#### **Macros**

- #define ARM\_LIMITSWITCH\_RCC RCC\_GPIOC
- #define ARM LIMITSWITCH PORT GPIOC
- #define ARM\_LIMITSWITCH\_PIN GPIO9
- #define ARM\_NVIC\_INTERRUPT\_NUMBER NVIC\_EXTI9\_5\_IRQ
- #define ARM\_LIMITSWITCH\_EXTI EXTI9
- #define ARM\_PRIORITY (3\*16)

## 4.4.1 Detailed Description

INterruption when the actuator reaches limit switch.

EXTI: External Interrupt, peripheral that is linked to a pin and generates interrupts NVIC: Nested vectored interrupt controller. It is a table that makes the link between the interruption event and the code (interrupt routine) to execute PRIORITY: from 0 to 255 in steps of 16 (for the time being not really important)

4.5 flag\_limit\_switch

## 4.5 flag limit switch

Interruption when the actuator is done(touches the limitswitch) EXTI: External Interrupt, peripheral that is linked to a pin and generates interrupts NVIC: Nested vectored interrupt controller. It is a table that makes the link between the interruption event and the code (interrupt routine) to execute PRIORITY: from 0 to 255 in steps of 16 (for the time being not really important)

#### **Macros**

- #define FLAG LIMITSWITCH RCC RCC GPIOC
- #define FLAG\_LIMITSWITCH\_PORT GPIOC
- #define FLAG\_LIMITSWITCH\_PIN GPIO8
- #define FLAG\_NVIC\_INTERRUPT\_NUMBER NVIC\_EXTI9\_5\_IRQ
- #define FLAG\_LIMITSWITCH\_EXTI EXTI8
- #define FLAG\_PRIORITY (4\*16)

## 4.5.1 Detailed Description

Interruption when the actuator is done(touches the limitswitch) EXTI: External Interrupt, peripheral that is linked to a pin and generates interrupts NVIC: Nested vectored interrupt controller. It is a table that makes the link between the interruption event and the code (interrupt routine) to execute PRIORITY: from 0 to 255 in steps of 16 (for the time being not really important)

## 4.6 flash memory

## **Functions**

- uint32\_t flash\_program\_data (uint8\_t sector, uint8\_t \*input\_data, uint16\_t num\_elements)

  This function programs data into the rom of the STM32.
- void flash\_read\_data (uint32\_t start\_address, uint16\_t num\_elements, uint8\_t \*poutput\_data)

  This function reads data in the rom of the STM32.

## Rom memory structure

These define the addresses use to operate on the rom part of the memory This organization is dependant on the very structure of the memory on the  $\mu$ controller You must check the documentation and allocate enough memory for your program and ram and reserve some of it for your storage in the linked script

- #define FLASH OPERATION ADDRESS ((uint32 t)0x08020000)
- #define FLASH\_SECTOR\_NUM\_MAX 3
- #define FLASH SECTOR SIZE 128000
- #define RESULT\_OK 0
- #define FLASH\_PROGRAM\_SIZE 0

#### 4.6.1 Detailed Description

#### 4.6.2 Function Documentation

#### 4.6.2.1 flash program data()

This function programs data into the rom of the STM32.

STM32 F4 has big sectors that can be programmed

#### **Parameters**

in	sector	adress of the sector to program (it will be overwritten /!)
out	input_data	data to program
in	num_elements	number of byte of input_data

#### Returns

error status (success=0)

4.6 flash\_memory

## 4.6.2.2 flash\_read\_data()

This function reads data in the rom of the STM32.

## **Parameters**

in	start_address	memory adress to start reading from
in	num_elements	number of bytes to read
out	poutput_data	pointer to the byte array where the read data will be stored

## 4.7 I2C

Definitions for the I2C serial protocol.

#### **Macros**

- #define I2C\_GPIO\_OTYPE GPIO\_OTYPE\_OD
- #define I2C GPIO PULL UP GPIO PUPD PULLUP
- #define I2C1\_SCL\_GPIO\_PORT GPIOB
- #define I2C1 SCL GPIO RCC RCC GPIOB
- #define I2C1\_SCL\_GPIO\_PIN GPIO6
- #define I2C1\_SCL\_AF GPIO\_AF4
- #define I2C1\_SDA\_GPIO\_PORT GPIOB
- #define I2C1\_SDA\_GPIO\_RCC RCC\_GPIOB
- #define I2C1 SDA GPIO PIN GPIO7
- #define I2C1\_SDA\_AF GPIO\_AF4

## 4.7.1 Detailed Description

Definitions for the I2C serial protocol.

OD: Open drain (req. for the protocol arbitration) PULLUP: the two I2C lines have to be pulled up SCL: Clock Pin SDA: Data pin

4.8 tof\_shiftr 15

## 4.8 tof shiftr

Resetting the tof is done via this shift register.

#### **Macros**

- #define SHIFTR DSAB RCC RCC GPIOC
- #define SHIFTR DSAB PORT GPIOC
- #define SHIFTR\_DSAB\_PIN GPIO1
- #define SHIFTR CP RCC RCC GPIOC
- #define SHIFTR CP PORT GPIOC
- #define SHIFTR\_CP\_PIN GPIO0

#### 4.8.1 Detailed Description

Resetting the tof is done via this shift register.

DSAB: data pin CP: clock pin for the shift

Registre à décalage : à cause de la configuration interne des ToFs, il n'est pas possible de les reset tous en même temps car alors les adresses I2C se réattribueraient un peu n'importe comment et ne permettraient plus au microcontrôleur de savoir quel ToF a quelle adresse. C'est pourquoi on a choisi d'utiliser un registre à décalage : Lorsqu'il faut reset les ToFs, on passe DSAB à 1 puis on envoie une impulsion sur CP pour mettre ce 1 dans le premier bit du registre. Ceci aura pour effet de reset le premier ToF qui ne connaît alors plus son adresse. On passe DSAB à 0 pour éviter de reset plusieurs ToFs d'un coup. Ensuite on répète : "On envoie une impulsion sur CP pour faire passer le 1 du bit n au bit n+1 dans le registre. Ce faisant, le n ème ToF n'est plus en état de reset et on peut lui réattribuer sa propre adresse sans crainte de confusion." On reset successivement tous les ToFs en leur réattribuant leur propre adresse à chaque fois

## 4.9 Range Ranging Profile

## **Macros**

- #define VL53L0X\_LR\_SIGNAL\_LIMIT (FixPoint1616\_t)(0.25\*65536)
- #define VL53L0X\_LR\_SIGMA\_LIMIT (FixPoint1616\_t)(18\*65536)
- #define VL53L0X\_LR\_TIMING\_BUDGET 33000
- #define VL53L0X\_LR\_VCSEL\_PERIOD\_PRE\_RANGE 14
- #define VL53L0X\_LR\_VCSEL\_PERIOD\_FINAL\_RANGE 10

## 4.9.1 Detailed Description

4.10 tof\_tim 17

## 4.10 tof\_tim

Internal timer that will generate interrupts to get tof sensor measurement TIM4 DIER: DMA/INterrupt enable register (we use an interrupt) SR: Status Register UI: Update interrupt.

## **Macros**

- #define TOF\_TIM\_RCC RCC\_TIM4
- #define TOF\_TIM TIM4
- #define TOF\_TIM\_NVIC NVIC\_TIM4\_IRQ
- #define **TOF\_TIM\_DIER\_UIE** TIM\_DIER\_UIE
- #define TOF\_TIM\_SR\_UIF TIM\_SR\_UIF

## 4.10.1 Detailed Description

Internal timer that will generate interrupts to get tof sensor measurement TIM4 DIER: DMA/INterrupt enable register (we use an interrupt) SR: Status Register UI: Update interrupt.

## 4.11 debug\_uart

Uart used for debugging via a usb to a pc.

#### **Macros**

- #define **DEBUG\_RCC\_USART** RCC\_USART2
- #define **DEBUG USART** USART2
- #define **DEBUG\_UART\_SPEED** (9600)
- #define **DEBUG\_PORT\_TX** GPIOA
- #define **DEBUG\_PORT\_TX\_RCC** RCC\_GPIOA
- #define **DEBUG\_PIN\_TX** GPIO2
- #define **DEBUG\_AF\_TX** GPIO\_AF7
- #define **DEBUG\_PORT\_RX** GPIOA
- #define DEBUG\_PORT\_RX\_RCC RCC GPIOA
- #define **DEBUG\_PIN\_RX** GPIO3
- #define **DEBUG\_AF\_RX** GPIO\_AF7
- #define **DEBUG\_UART\_EXTI** EXTI26
- #define DEBUG\_UART\_NVIC NVIC\_USART2\_IRQ

## 4.11.1 Detailed Description

Uart used for debugging via a usb to a pc.

Baudrate is 9600

## **Data Structure Documentation**

## 5.1 can\_msg\_buffer\_list\_t Struct Reference

FIFO linked list to store impeding CAN messages (software side)

```
#include <canmsgs.h>
```

#### **Data Fields**

- Can\_rx\_msg data
- can\_msg\_buffer\_list\_t \* next

## 5.1.1 Detailed Description

FIFO linked list to store impeding CAN messages (software side)

#### **Parameters**

data	A can mesage
next	Pointer to the next element in the list

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

• lowlevel/include/canmsgs.h

## 5.2 Can\_rx\_msg Struct Reference

Frame of stantard received CAN messages.

```
#include <canmsgs.h>
```

## **Data Fields**

- uint32\_t std\_id
- bool ext\_id
- · bool rtr
- uint8\_t fmi
- uint8\_t dlc
- uint8\_t data [8]
- uint8\_t crc
- · uint8 t ack
- uint16\_t **ts**

## 5.2.1 Detailed Description

Frame of stantard received CAN messages.

#### **Parameters**

std⊷ _id	Unique identifier which also represents the message priority
ext← _id	Dominant for standard frame. Recessive for extended frame
rtr	Dominant for data frames. Recessive for request frames
fmi	ID of the matched filter
dlc	Data length code. Number of bytes of data
data	Data to be transmitted
crc	Cyclic redundancy check. Error detecting code
ack	Acknowledge the receipt of a valid CAN frame (dominant)
ts	{Timestamp. Pointer to store the message timestamp. Only valid on time triggered CAN. Use NULL to ignore.}

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

• lowlevel/include/canmsgs.h

## 5.3 Can\_tx\_msg Struct Reference

## **Data Fields**

- uint32\_t std\_id
- bool ext\_id
- bool rtr
- uint8\_t fmi
- uint8\_t dlc
- uint8\_t data [8]
- uint8\_t crc
- uint8\_t ack
- uint16\_t **ts**

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

• lowlevel/include/canmsgs.h

## 5.4 can\_tx\_msg Struct Reference

Frame of stantard transmitted CAN messages.

#include <canmsgs.h>

## 5.4.1 Detailed Description

Frame of stantard transmitted CAN messages.

#### **Parameters**

std← _id	Unique identifier which also represents the message priority
ext← _id	Dominant for standard frame. Recessive for extended frame
rtr	Dominant for data frames. Recessive for request frames
fmi	ID of the matched filter
dlc	Data length code. Number of bytes of data
data	Data to be transmitted
crc	Cyclic redundancy check. Error detecting code
ack	Acknowledge the receipt of a valid CAN frame (dominant)
ts	{Timestamp. Pointer to store the message timestamp. Only valid on time triggered CAN. Use NULL to ignore.}

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

lowlevel/include/canmsgs.h

## 5.5 VL53L0X\_Calibration\_Parameter\_S Struct Reference

Storage of all Calibration Parameter for the TOF (VL53L1X)

#include <tof.h>

## **Data Fields**

- uint8\_t VhvSettings
- uint8\_t PhaseCal
- uint32\_t refSpadCount
- uint8\_t isApertureSpads
- int32\_t OffsetMicroMeter
- FixPoint1616\_t XTalkCompensationRateMegaCps

## 5.5.1 Detailed Description

Storage of all Calibration Parameter for the TOF (VL53L1X)

TODO: write all parameter

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

· lowlevel/include/tof.h

## **File Documentation**

## 6.1 lowlevel/canmsgs.c File Reference

```
#include "canmsgs.h"
#include <stdlib.h>
```

#### **Functions**

void can\_setup ()
 Startup configuration of the CAN system.

## 6.1.1 Detailed Description

This implements the setup of CAN protocol to allow F3, F4 and other potential computers to communicate Source: https://www.rhye.org/post/stm32-with-opencm3-3-canbus/.

This file is part of cdfr2020CarteCerveauProg

Date

10/2020

Licence:

Robotronik Phelma

Author

NPXav Benano JamesWright

24 File Documentation

## 6.1.2 Function Documentation

#### 6.1.2.1 can\_setup()

```
void can_setup ( )
```

Startup configuration of the CAN system.

## 6.2 lowlevel/exti.c File Reference

This implements the setup of the sensors linked to the actuators: the arm and the flag.

```
#include "exti.h"
```

#### **Functions**

void exti9\_5\_isr ()

interrupt routine for interrution of exti 9 to exti 5

void exti\_setup ()

initialize the peripheral that managed the exti line (syscfg)

- void \_limit\_switch\_init (uint32\_t exti, uint32\_t gpio\_port, uint8\_t interrupt\_number, enum exti\_trigger\_type trig)

  This function initializes the exti interrupt and nvic interrupts will be received from gpio\_port with the pin matching the number of the exti.
- void \_arm\_limit\_switch\_init ()

Initialize the GPIO and interrupts for the limit switch of the ARM.

• void \_flag\_limit\_switch\_init ()

Initialize the GPIO and interrupts for the limit switch of the FLAG.

#### 6.2.1 Detailed Description

This implements the setup of the sensors linked to the actuators: the arm and the flag.

This file is part of cdfr2020CarteCerveauProg

Date

09/2020

Licence:

Robotronik Phelma

Author

NPXav Benano Trukbidule

## 6.2.2 Function Documentation

## 6.2.2.1 \_limit\_switch\_init()

This function initializes the exti interrupt and nvic interrupts will be received from gpio\_port with the pin matching the number of the exti.

#### **Parameters**

exti	the external interrupt peripheral linked to the gpio pin (number must match !)
gpio_port	the port on which the limit switch will be plugged
interrupt_number	the interrupt number in the NVIC table
trig	the type of event that will trigger the interrupt (rising,falling,both)

### 6.2.2.2 exti9\_5\_isr()

```
void exti9_5_isr ( )
```

interrupt routine for interrution of exti 9 to exti 5

## Warning

You may need to edit this function to change the interrupt routine for the given functionality

## 6.2.2.3 exti\_setup()

```
void exti_setup ( )
```

initialize the peripheral that managed the exti line (syscfg)

## Warning

We assume you already setup your actuator and uart

26 File Documentation

## 6.3 lowlevel/flash.c File Reference

This implements function to read and flash data in the rom sector.

```
#include "flash.h"
```

#### **Functions**

- void setup\_flash\_rom ()
- uint32\_t flash\_program\_data (uint8\_t sector, uint8\_t \*input\_data, uint16\_t num\_elements)

This function programs data into the rom of the STM32.

void flash\_read\_data (uint32\_t start\_address, uint16\_t num\_elements, uint8\_t \*poutput\_data)

This function reads data in the rom of the STM32.

#### 6.3.1 Detailed Description

This implements function to read and flash data in the rom sector.

Date

Wed Jun 9 21:06:39 2021

Author

benano NPXav

Copyright

```
Robotronik phelma This file is part of cdfr2020CerveauProg useful reference: libopencm3 example: https-
://github.com/libopencm3/libopencm3-examples/blob/master/examples/stm32/f1/stm32-h107/f1
_rw_example/flash_rw_example.c
```

## 6.4 lowlevel/include/actuator.h File Reference

This implements the setup of the actuators: the arm and the flag.

```
#include "gpio.h"
#include "timer.h"
#include "exti.h"
```

#### **Macros**

- #define PWM PRESCALE (64)
- #define PWM PERIOD (20000)
- #define ACTUATOR\_TIM\_RCC RCC TIM3
- #define ACTUATOR\_TIM TIM3
- #define ARM\_GPIO\_RCC\_EN RCC\_GPIOC
- #define ARM\_PORT\_EN GPIOC
- #define ARM\_PIN\_EN GPIO7
- #define ARM\_AF GPIO\_AF2
- #define ARM OC ID TIM OC2
- #define ARM OC MODE TIM OCM PWM1
- #define ARM GPIO RCC DIR 1 RCC GPIOB
- #define ARM\_PORT\_DIR\_1 GPIOB
- #define ARM\_PIN\_DIR\_1 GPIO12
- #define ARM\_GPIO\_RCC\_DIR\_2 RCC\_GPIOB
- #define ARM\_PORT\_DIR\_2 GPIOB
- #define ARM PIN DIR 2 GPIO13
- #define ARM\_INIT\_DIR 0
- #define ARM\_INVERT\_DIR (-1)
- #define FLAG\_GPIO\_RCC\_EN RCC\_GPIOC
- #define FLAG PORT EN GPIOC
- #define FLAG PIN EN GPIO6
- #define FLAG\_AF GPIO AF2
- #define FLAG\_OC\_ID TIM\_OC1
- #define FLAG\_OC\_MODE TIM\_OCM\_PWM1
- #define FLAG\_GPIO\_RCC\_DIR\_1 RCC\_GPIOB
- #define FLAG\_PORT\_DIR\_1 GPIOB
- #define FLAG\_PIN\_DIR\_1 GPIO14
- #define FLAG\_GPIO\_RCC\_DIR\_2 RCC\_GPIOB
- #define FLAG\_PORT\_DIR\_2 GPIOB
- #define FLAG\_PIN\_DIR\_2 GPIO15
- #define FLAG\_INIT\_DIR 0
- #define FLAG\_INVERT\_DIR (-1)

#### **Enumerations**

enum actuator sel { ARM, FLAG }

enum of the actuators, used to identify them in some functions (like function actuators\_set)

#### **Functions**

void actuator\_setup ()

This function initializes the timers (including the timer output comparator) and GPIOs to pilot by PWM the propulsion motors + the GPIOs for the direction.

· void actuator set (enum actuator sel sel, int8 t value)

This function pilots the sel with a value between -100(backward full speed) and +100 (forward full speed). The forward direction depends on the sign of ACT\_X\_INVER\_DIR.

28 File Documentation

## 6.4.1 Detailed Description

This implements the setup of the actuators: the arm and the flag.

This file is part of cdfr2020CarteCerveauProg

Date

07/2020

Licence:

Robotronik Phelma

**Author** 

PhenixRobotik NPXav Benano Trukbidule

#### 6.4.2 Macro Definition Documentation

#### 6.4.2.1 PWM PERIOD

```
#define PWM_PERIOD (20000)
```

We need a 50 Hz period (1000 / 20ms = 50), thus divide 100000 by 50 = 20000 (us).

## 6.4.2.2 PWM\_PRESCALE

```
#define PWM_PRESCALE (64)
```

Prescale 64000000 Hz system clock by 64 = 1000000 Hz.

## 6.4.3 Enumeration Type Documentation

#### 6.4.3.1 actuator\_sel

```
enum actuator_sel
```

enum of the actuators, used to identify them in some functions (like function actuators\_set)

#### 6.4.4 Function Documentation

## 6.4.4.1 actuator\_set()

This function pilots the sel with a value between -100(backward full speed) and +100 (forward full speed). The forward direction depends on the sign of ACT\_X\_INVER\_DIR.

#### **Parameters**

sel	The actuator that will be piloted (eg ARM)
value	value is between -100 and +100, controls the speed and direction of the motor sel (eg +54)

This function pilots the sel with a value between -100(backward full speed) and +100 (forward full speed). The forward direction depends on the sign of ACT\_X\_INVER\_DIR.

#### **Parameters**

sel	The motor that will be piloted (eg ARM)
value	value is between -100 and +100, controls the speed and direction of the motor sel (eg +54)

#### 6.4.4.2 actuator\_setup()

```
void actuator_setup ( )
```

This function initializes the timers (including the timer output comparator) and GPIOs to pilot by PWM the propulsion motors + the GPIOs for the direction.

This function initializes the timers (including the timer output comparator) and GPIOs to pilot by PWM the propulsion motors + the GPIOs for the direction.

## 6.5 lowlevel/include/canmsgs.h File Reference

This implements the setup of communication between F3, F4 and other potential computers using CAN protocol. Source: https://www.rhye.org/post/stm32-with-opencm3-3-canbus/.

```
#include "gpio.h"
#include <stdio.h>
#include <libopencm3/cm3/nvic.h>
#include <libopencm3/stm32/can.h>
#include <libopencm3/stm32/exti.h>
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/usart.h>
```

#### **Data Structures**

- struct Can\_tx\_msg
- struct Can\_rx\_msg

Frame of stantard received CAN messages.

• struct can\_msg\_buffer\_list\_t

FIFO linked list to store impeding CAN messages (software side)

30 File Documentation

#### **Macros**

- #define PARAM\_SJW CAN BTR SJW 1TQ
- #define PARAM\_TS1 CAN BTR TS1 10TQ
- #define PARAM\_TS2 CAN BTR TS2 3TQ
- #define PARAM BRP 16
- #define CAN1\_RX\_PORT GPIOB
- #define CAN1 RX PIN GPIO8
- #define CAN1\_RX\_RCC RCC\_GPIOB
- #define CAN1 RX AF GPIO AF9
- #define CAN1 TX PORT GPIOB
- #define CAN1 TX PIN GPIO9
- #define CAN1\_TX\_RCC RCC\_GPIOB
- #define CAN1\_TX\_AF GPIO\_AF9
- #define CAN1\_NVIC\_TX NVIC\_CAN1\_TX\_IRQ
- #define CAN1\_NVIC\_RX0 NVIC\_CAN1\_RX0\_IRQ
- #define CAN1\_NVIC\_RX1 NVIC\_CAN1\_RX1\_IRQ
- #define CAN1\_NVIC\_SCE NVIC\_CAN1\_SCE\_IRQ

#### **Typedefs**

- typedef struct Can\_tx\_msg Can\_tx\_msg
- typedef struct Can\_rx\_msg Can\_rx\_msg
- typedef struct can msg buffer list t can msg buffer list t

#### **Functions**

void \_can\_msg\_buffer\_append (Can\_rx\_msg rx\_msg)

Appends a can message at the end of the global can buffer list.

int can\_msg\_buffer\_pop (Can\_rx\_msg \*rx\_msg)

Pops the first element of the global can buffer list.

void can\_setup ()

Startup configuration of the CAN system.

void cec\_can\_isr ()

This function manages messages pending on FIFO 0 and 1.

void receive (uint8\_t fifo)

This function receives the message and push it in a FIFO.

• void transmit (uint32\_t id, Can\_tx\_msg tx\_msg)

This function transmits a message.

## 6.5.1 Detailed Description

This implements the setup of communication between F3, F4 and other potential computers using CAN protocol. Source: https://www.rhye.org/post/stm32-with-opencm3-3-canbus/.

This file is part of cdfr2020CerveauProg

Date

10/2020

Licence:

Robotronik Phelma

**Author** 

NPXav Benano Trukbidule JamesWright Floorcows

### 6.5.2 Function Documentation

### 6.5.2.1 \_can\_msg\_buffer\_append()

Appends a can message at the end of the global can buffer list.

#### **Parameters**

rx_msg	The can message to be appended
--------	--------------------------------

#### 6.5.2.2 can\_msg\_buffer\_pop()

Pops the first element of the global can buffer list.

### **Parameters**

rx\_msg | Pointer to the variable where the can message will be stored

### Returns

0: a can message was found and stored in rx\_msg, 1: the list was empty

### 6.5.2.3 can\_setup()

```
void can_setup ( )
```

Startup configuration of the CAN system.

### 6.5.2.4 cec\_can\_isr()

```
void cec_can_isr ( )
```

This function manages messages pending on FIFO 0 and 1.

#### 6.5.2.5 receive()

```
void receive (
          uint8_t fifo )
```

This function receives the message and push it in a FIFO.

#### **Parameters**

```
fifo The FIFO in which the message is pushed
```

### 6.5.2.6 transmit()

This function transmits a message.

#### **Parameters**

id	id of the message to be transmitted
tx_msg	structure of the message to transmit

### 6.6 lowlevel/include/clock.h File Reference

This implements the setup of the system clock, acces function (debug) and temporal fonction (delay)

```
#include <stdint.h>
#include <libopencm3/cm3/systick.h>
#include <libopencm3/stm32/rcc.h>
```

### **Functions**

• void clock\_setup ()

This function setup the system clock.

• uint32\_t \_clock\_get\_systicks ()

This function gets the number of systicks since starting.

void delay\_ms (uint32\_t ms)

This function implements a delay in ms.

### 6.6.1 Detailed Description

This implements the setup of the system clock, acces function (debug) and temporal fonction (delay)

This file is part of cdfr2020CerveauProg

Date

07/2020

Licence:

Robotronik Phelma

**Author** 

PhenixRobotik NPXav Benano Trukbidule

### 6.6.2 Function Documentation

### 6.6.2.1 clock\_setup()

```
void clock_setup ( )
```

This function setup the system clock.

#### 6.6.2.2 delay\_ms()

This function implements a delay in ms.

**Parameters** 

ms value of delay in ms

# 6.7 lowlevel/include/exti.h File Reference

This implements the setup of the sensors linked to the actuators: the arm and the flag.

```
#include <stdint.h>
#include <stdio.h>
#include "libopencm3/stm32/exti.h"
#include "libopencm3/cm3/nvic.h"
#include "gpio.h"
#include "actuator.h"
```

#### **Macros**

- #define ARM LIMITSWITCH RCC RCC GPIOC
- #define ARM\_LIMITSWITCH\_PORT GPIOC
- #define ARM LIMITSWITCH PIN GPIO9
- #define ARM\_NVIC\_INTERRUPT\_NUMBER NVIC\_EXTI9\_5\_IRQ
- #define ARM\_LIMITSWITCH\_EXTI EXTI9
- #define ARM\_PRIORITY (3\*16)
- #define FLAG LIMITSWITCH RCC RCC GPIOC
- #define FLAG LIMITSWITCH PORT GPIOC
- #define FLAG\_LIMITSWITCH\_PIN GPIO8
- #define FLAG NVIC INTERRUPT NUMBER NVIC EXTI9 5 IRQ
- #define FLAG\_LIMITSWITCH\_EXTI EXTI8
- #define FLAG\_PRIORITY (4\*16)

#### **Functions**

- void \_limit\_switch\_init (uint32\_t exti, uint32\_t gpio\_port, uint8\_t interrupt\_number, enum exti\_trigger\_type trig)

  This function initializes the exti interrupt and nvic interrupts will be received from gpio\_port with the pin matching the number of the exti.
- void \_flag\_limit\_switch\_init ()

Initialize the GPIO and interrupts for the limit switch of the FLAG.

void \_arm\_limit\_switch\_init ()

Initialize the GPIO and interrupts for the limit switch of the ARM.

void exti\_setup ()

initialize the peripheral that managed the exti line (syscfg)

void exti9\_5\_isr ()

interrupt routine for interrution of exti 9 to exti 5

#### 6.7.1 Detailed Description

This implements the setup of the sensors linked to the actuators: the arm and the flag.

This file is part of cdfr2020CarteCerveauProg

Date

07/2020

Licence:

Robotronik Phelma

**Author** 

PhenixRobotik NPXav Benano Trukbidule

### 6.7.2 Function Documentation

### 6.7.2.1 \_limit\_switch\_init()

This function initializes the exti interrupt and nvic interrupts will be received from gpio\_port with the pin matching the number of the exti.

#### **Parameters**

exti	the external interrupt peripheral linked to the gpio pin (number must match !)
gpio_port	the port on which the limit switch will be plugged
interrupt_number	the interrupt number in the NVIC table
trig	the type of event that will trigger the interrupt (rising,falling,both)

### 6.7.2.2 exti9\_5\_isr()

```
void exti9_5_isr ( )
```

interrupt routine for interrution of exti 9 to exti 5

### Warning

You may need to edit this function to change the interrupt routine for the given functionality

### 6.7.2.3 exti\_setup()

```
void exti_setup ( )
```

initialize the peripheral that managed the exti line (syscfg)

### Warning

We assume you already setup your actuator and uart

# 6.8 lowlevel/include/gpio.h File Reference

This implements the setup of a gpio pin

```
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/gpio.h>
#include "clock.h"
```

#### **Enumerations**

• enum pulse\_active { low, high }

enum of the pulse possible directions active low is a high->low->high transition active high is a low->high->low transition

#### **Functions**

• void <u>gpio\_setup\_pin\_af</u> (enum rcc\_periph\_clken rcc\_clken, uint32\_t gpio\_port, uint16\_t gpio\_pin, uint8\_t gpio\_altfun, uint8\_t pull\_up\_down, uint8\_t otype)

This function setup a pin for an alternate function.

void \_gpio\_setup\_pin (enum rcc\_periph\_clken clken, uint32\_t port, uint16\_t pin, uint8\_t mode, uint8\_t pull
 \_up\_down, uint8\_t otype)

This function setup a GPIO pin for standard input or output.

void \_\_pulse (uint32\_t port, uint16\_t pin, enum pulse\_active dir, uint16\_t delay)

This function write a short pulse on the output pin.

### 6.8.1 Detailed Description

This implements the setup of a gpio pin

This file is part of cdfr2020CarteCerveauProg

Date

07/2020

Licence:

Robotronik Phelma

**Author** 

NPXav Benano Trukbidule

#### 6.8.2 Function Documentation

# 6.8.2.1 \_\_pulse()

This function write a short pulse on the output pin.

#### **Parameters**

port	the port to enable
pin	the pint to enable
dir	active high or low (pulse direction)
delay	duration of the pulse

### 6.8.2.2 \_gpio\_setup\_pin()

This function setup a GPIO pin for standard input or output.

### **Parameters**

clken	the clock of the port to enable
port	the port to enable
pin	the pint to enable
mode	the mode of your GPIO (GPIO_MODE_INPUT,GPIO_MODE_OUTPUT)
pull_up_down	the type of pull for the pin (GPIO_PUPD_NONE, GPIO_PUPD_PULLUP, GPIO_PUPD_PULLDOWN)
otype	the type of output for the pin (GPIO_OTYPE_OD open drain or GPIO_OTYPE_PP push pull)

### 6.8.2.3 \_gpio\_setup\_pin\_af()

This function setup a pin for an alternate function.

### **Parameters**

rcc_clken	reset clock control for the pin (usualy RCC_X with X the gpio_port)
gpio_port	port of the selected pin
gpio_pin	number of the selected pin

#### **Parameters**

gpio_altfun identifier for the alternate function (usualy GPIO_AFX with X the number for alt		identifier for the alternate function (usualy GPIO_AFX with X the number for altfun)
	pull_up_down	the type of pull for the pin (GPIO_PUPD_NONE, GPIO_PUPD_PULLUP, GPIO_PUPD_PULLDOWN)
	otype	the type of output for the pin (GPIO_OTYPE_OD open drain or GPIO_OTYPE_PP push pull)

### 6.9 lowlevel/include/i2c.h File Reference

This implements the setup of an I2C peripheral.

```
#include <libopencm3/stm32/i2c.h>
#include <stdio.h>
#include "gpio.h"
```

#### **Macros**

- #define I2C MAX TIMEOUT 10
- #define I2C\_GPIO\_OTYPE GPIO\_OTYPE\_OD
- #define I2C\_GPIO\_PULL\_UP GPIO\_PUPD\_PULLUP
- #define I2C1\_SCL\_GPIO\_PORT GPIOB
- #define I2C1 SCL GPIO RCC RCC GPIOB
- #define I2C1\_SCL\_GPIO\_PIN GPIO6
- #define I2C1 SCL AF GPIO AF4
- #define I2C1\_SDA\_GPIO\_PORT GPIOB
- #define I2C1\_SDA\_GPIO\_RCC RCC\_GPIOB
- #define I2C1\_SDA\_GPIO\_PIN GPIO7
- #define I2C1 SDA AF GPIO AF4

### **Typedefs**

typedef enum I2C\_Status\_E I2C\_status
 enum of the possible I2C status, used for status monitoring

#### **Enumerations**

enum I2C\_Status\_E { I2C\_OK, I2C\_TIMEOUT }
 enum of the possible I2C status, used for status monitoring

#### **Functions**

void i2c\_setup (uint32\_t i2c\_peripheral)

Set the application-specific I2C configuration.

• I2C\_status i2c\_write7 (uint32\_t i2c, int addr, uint8\_t \*data, size\_t n)

This function re-implement Libopencm3 write on I2C bus with 7 bit address.

• I2C\_status i2c\_read7 (uint32\_t i2c, int addr, uint8\_t \*res, size\_t n)

This function re-implement Libopencm3 read on I2C bus with 7 bit address.

### 6.9.1 Detailed Description

This implements the setup of an I2C peripheral.

This file is part of cdfr2020CarteCerveauProg

Date

10/2020

Licence:

Robotronik Phelma

Author

NPXav Benano PhoenixRobotics (Antonin H.)

# 6.9.2 Typedef Documentation

#### 6.9.2.1 I2C\_status

```
typedef enum I2C_Status_E I2C_status
```

enum of the possible I2C status, used for status monitoring

### 6.9.3 Enumeration Type Documentation

### 6.9.3.1 I2C\_Status\_E

```
enum I2C_Status_E
```

enum of the possible I2C status, used for status monitoring

### 6.9.4 Function Documentation

### 6.9.4.1 i2c\_read7()

This function re-implement Libopencm3 read on I2C bus with 7 bit address.

#### **Parameters**

i2c	I2C peripheral used
addr	address of slave
res	data that have been read
n	size of data in byte

### Returns

I2C bus status

#### See also

libopencm3 i2c\_read7

### 6.9.4.2 i2c\_setup()

Set the application-specific I2C configuration.

### **Parameters**

### 6.9.4.3 i2c\_write7()

This function re-implement Libopencm3 write on I2C bus with 7 bit address.

#### **Parameters**

i2c	I2C peripheral used
addr	address of slave
data	data to be sent
n	size of data in byte

Returns

I2C bus status

See also

libopencm3 i2c\_write7

### 6.10 lowlevel/include/timer.h File Reference

This implements the functions required setup a timer and its output channel

```
#include <stdint.h>
#include <libopencm3/stm32/timer.h>
#include <libopencm3/stm32/rcc.h>
```

### **Functions**

void \_timer\_setup (enum rcc\_periph\_clken rcc\_clken, uint32\_t timer\_peripheral, uint32\_t prescaler, uint32\_t period)

This function setup an internal timer with the given parameters.

void \_timer\_setup\_output\_c (uint32\_t timer\_peripheral, enum tim\_oc\_id oc\_id, enum tim\_oc\_mode oc\_mode, uint32\_t oc\_value)

This function configure the output comparator of a channel for the timer specified.

void <u>\_timer\_start</u> (uint32\_t timer\_peripheral)

This function starts the given timer.

# 6.10.1 Detailed Description

This implements the functions required setup a timer and its output channel

This file is part of cdfr2020CerveauProg

Date

07/2020

Licence:

Robotronik Phelma

Author

NPXav Benano Trukbidule

### 6.10.2 Function Documentation

### 6.10.2.1 \_timer\_setup()

This function setup an internal timer with the given parameters.

#### **Parameters**

rcc_clken	reset and clock control enable for the timer (clock tree)
timer_peripheral	timer selected
prescaler	the input frequency of the timer (sys_clk) is divided by this factor
period	period of the timer in us

### 6.10.2.2 \_timer\_setup\_output\_c()

This function configure the output comparator of a channel for the timer specified.

#### **Parameters**

timer_peripheral	selected timer
oc_id	selected channel of the output comparator
oc_mode	different mode used for the timer
oc_value	initial value of the duty cycle

### 6.10.2.3 \_timer\_start()

This function starts the given timer.

#### **Parameters**

timer_peripheral	selected timer
------------------	----------------

### 6.11 lowlevel/include/tof.h File Reference

This implements all needed peripheral to use the tof.

```
#include "gpio.h"
#include "i2c.h"
#include "v15310x_api.h"
```

#### **Data Structures**

• struct VL53L0X\_Calibration\_Parameter\_S

Storage of all Calibration Parameter for the TOF (VL53L1X)

### **Macros**

- #define **TOF\_COR\_FACTOR** ((int) (0.5 \* 256))
- #define TOF\_DEFAULT\_ADDR 0x52
- #define TOF\_DELAY 50
- #define SHIFTR\_DSAB\_RCC RCC\_GPIOC
- #define SHIFTR DSAB PORT GPIOC
- #define SHIFTR\_DSAB\_PIN GPIO1
- #define SHIFTR\_CP\_RCC RCC\_GPIOC
- #define SHIFTR\_CP\_PORT GPIOC
- #define SHIFTR\_CP\_PIN GPIO0
- #define VL53L0X\_LR\_SIGNAL\_LIMIT (FixPoint1616\_t)(0.25\*65536)
- #define VL53L0X\_LR\_SIGMA\_LIMIT (FixPoint1616\_t)(18\*65536)
- #define VL53L0X\_LR\_TIMING\_BUDGET 33000
- #define VL53L0X LR VCSEL PERIOD PRE RANGE 14
- #define VL53L0X\_LR\_VCSEL\_PERIOD\_FINAL\_RANGE 10

### **Typedefs**

• typedef struct VL53L0X\_Calibration\_Parameter\_S VL53L0X\_Calibration\_Parameter

#### **Functions**

```
    VL53L0X_Error tof_setup (VL53L0X_DEV *t_dev, uint8_t tof_number)

     setup all peripheral req. for tof usage

    VL53L0X_Error _tof_1_setup (VL53L0X_DEV dev, uint8_t tof_addr)

     setup a tof

    void <u>tof_init_struct</u> (VL53L0X_DEV dev)

     setup the structure with the standard address and I2C peripheral

    VL53L0X_Error _tof_poke (VL53L0X_DEV dev)

      Check if the tof is answering.
• VL53L0X_Error _tof_set_address (VL53L0X_DEV dev, uint8_t addr)
     set the tof I2C slave address

    VL53L0X_Error _tof_setup_addr (VL53L0X_DEV dev, uint8_t addr)

     setup the tof with its address (calling poke and set address)

    VL53L0X_Error _tof_config (VL53L0X_DEV dev)

      Configure the tof with its calibration data and ranging profile.

    VL53L0X_Error _tof_calibration (VL53L0X_DEV dev, VL53L0X_Calibration_Parameter *calib_param, Fix

  Point1616_t offset_cal_distance, FixPoint1616_t xTalk_cal_distance)
     Function to calibrate a tof (called to calibrate a specific tof)

    VL53L0X_Error _tof_setup_calib (VL53L0X_DEV dev, VL53L0X_Calibration_Parameter *calib_param)

     Function to calibrate a tof without target (called at every tof setup)

    VL53L0X Error tof perform measure (VL53L0X DEV dev)

     Function that performs a single measurement coming from the tof defined by dev.

    VL53L0X_Error tof_print_device_info (VL53L0X_DEV dev)

     Function to print tof device information.

    VL53L0X_Error tof_print_calib_info (VL53L0X_DEV dev)

     Function to print tof calibration information.

    VL53L0X_Error tof_print_PAL_state (VL53L0X_DEV dev)

     Function to print PAL state.

    VL53L0X Error tof print device mode (VL53L0X DEV dev)

     Function to print device mode.

    VL53L0X Error tof print ranging status (VL53L0X RangingMeasurementData t measure data)

     Function to print range status.

    VL53L0X Error tof print data measure (VL53L0X RangingMeasurementData t measure data)

     Function to print measured data in detail.

    VL53L0X Error tof print int status (VL53L0X DEV dev)

     Function to print interrupt status.
• void tof reset ()
     reset all tof via the shift register

    void shift reg init ()

     reset the shiftregister to start counting from 0
```

• void shift reg (int i)

pulse at the output +i @params i amount of shifting

# 6.11.1 Detailed Description

This implements all needed peripheral to use the tof.

This file is part of cdfr2020CarteCerveauProg

Date

03/2021

Licence:

Robotronik Phelma

**Author** 

NPXav benano

#### 6.11.2 Function Documentation

### 6.11.2.1 \_shift\_reg()

```
void \_shift\_reg (  int i )
```

pulse at the output +i @params i amount of shifting

#### 6.11.2.2 \_shift\_reg\_init()

```
void _shift_reg_init ( )
```

reset the shiftregister to start counting from 0

### 6.11.2.3 \_tof\_1\_setup()

setup a tof

#### **Parameters**

dev	our tof object
tof_addr	address to be given to tof object

### 6.11.2.4 \_tof\_calibration()

Function to calibrate a tof (called to calibrate a specific tof)

#### **Parameters**

dev	our tof object	
calib_param	structure to store all calibration parameter	
offset_cal_distance	distance to the white target in millimeter	
xTalk_cal_distance	distance to the grey target in millimeter	

#### Returns

return the error type from the API

### 6.11.2.5 \_tof\_config()

```
\begin{tabular}{lll} VL53L0X\_Error tof\_config ( \\ VL53L0X\_DEV dev ) \end{tabular}
```

Configure the tof with its calibration data and ranging profile.

### Parameters

```
dev our tof object
```

#### Returns

return the error type from the API

# 6.11.2.6 \_tof\_init\_struct()

```
void _tof_init_struct ( \label{eq:vl53l0X_DEV} {\tt VL53L0X\_DEV} \ {\tt dev} \ )
```

setup the structure with the standard address and I2C peripheral

#### **Parameters**

```
dev our tof object
```

### 6.11.2.7 \_tof\_poke()

```
\begin{tabular}{lll} VL53L0X\_Error \_tof\_poke ( & & VL53L0X\_DEV $dev$ ) \end{tabular}
```

Check if the tof is answering.

#### **Parameters**

dev	our tof object
-----	----------------

#### Returns

return the error type from the API

## 6.11.2.8 \_tof\_set\_address()

set the tof I2C slave address

#### **Parameters**

dev	our tof object	
addr	the slave address	

### Returns

return the error type from the API

### 6.11.2.9 \_tof\_setup\_addr()

setup the tof with its address (calling poke and set address)

#### **Parameters**

dev	our tof object	
addr	the slave address	

#### Returns

return the error type from the API

#### 6.11.2.10 \_tof\_setup\_calib()

Function to calibrate a tof without target (called at every tof setup)

#### **Parameters**

dev	our tof object
calib_param	structure to store all calibration parameter

#### Returns

return the error type from the API

### 6.11.2.11 tof\_perform\_measure()

```
\begin{tabular}{ll} VL53L0X\_Error tof\_perform\_measure ( \\ VL53L0X\_DEV $dev$ ) \end{tabular}
```

Function that performs a single measurement coming from the tof defined by dev.

#### Warning

this function is a blocking call for 1-2 ms IF THE CALLED ARE SPACED BY MORE THAN 40ms (because you need the time to actually expose the sensor, if you call the function faster you will have delay required to obtain a measurement)

#### **Parameters**

dev the defining structure of the tof

#### Returns

VL53L0X\_Error

### 6.11.2.12 tof\_print\_calib\_info()

```
\begin{tabular}{ll} VL53L0X\_Error tof\_print\_calib\_info ( \\ VL53L0X\_DEV $dev $) \end{tabular}
```

Function to print tof calibration information.

#### Warning

We assume you already setup your tof and uart

#### **Parameters**

dev our tof object

### 6.11.2.13 tof\_print\_data\_measure()

```
\label{eq:VL53L0X} $$ VL53L0X_{encoder} = ($$ VL53L0
```

Function to print measured data in detail.

#### Warning

We assume you already setup your tof and uart

#### **Parameters**

measure data	measured data buffer

#### See also

VL53L0X Ranging Measurement Data

#### 6.11.2.14 tof\_print\_device\_info()

```
\begin{tabular}{ll} VL53L0X\_Error tof\_print\_device\_info ( \\ VL53L0X\_DEV $dev$ ) \end{tabular}
```

Function to print tof device information.

Warning

We assume you already setup your tof and uart

#### **Parameters**

```
dev our tof object
```

### 6.11.2.15 tof\_print\_device\_mode()

```
\begin{tabular}{ll} VL53L0X\_Error tof\_print\_device\_mode ( \\ VL53L0X\_DEV $dev $) \end{tabular}
```

Function to print device mode.

Warning

We assume you already setup your tof and uart

#### **Parameters**

```
dev our tof object
```

See also

VL53L0X DeviceModes group (I181 vl53l0x\_def.h)

### 6.11.2.16 tof\_print\_int\_status()

```
\begin{tabular}{ll} VL53L0X\_Error tof\_print\_int\_status ( \\ VL53L0X\_DEV $dev $) \end{tabular}
```

Function to print interrupt status.

Warning

We assume you already setup your tof and uart

#### **Parameters**

```
dev our tof object
```

See also

VL53L0X\_REG\_SYSTEM\_INTERRUPT (I152 vl53l0x\_device.h)

#### 6.11.2.17 tof\_print\_PAL\_state()

```
\begin{tabular}{ll} VL53L0X\_Error tof\_print\_PAL\_state ( \\ VL53L0X\_DEV $dev$ ) \end{tabular}
```

Function to print PAL state.

Warning

We assume you already setup your tof and uart

#### **Parameters**

```
dev our tof object
```

See also

VL53L0X State group (I273 vI53I0x\_def.h)

### 6.11.2.18 tof\_print\_ranging\_status()

```
\label{eq:vl53l0x_error} $$ VL53L0X_Error tof\_print\_ranging\_status ( $$ VL53L0X_RangingMeasurementData\_t $$ measure\_data $$ )
```

Function to print range status.

Warning

We assume you already setup your tof and uart

#### **Parameters**

<i>measure_data</i> r	measured data buffer
-----------------------	----------------------

See also

VL53L0X Range status (p16 User Manual)

#### 6.11.2.19 tof\_reset()

```
void tof_reset ( )
```

reset all tof via the shift register

### 6.11.2.20 tof\_setup()

```
\label{eq:vl53l0x_dev} $$ VL53L0X_DEV * t_dev, $$ uint8_t tof_number $$ )
```

setup all peripheral req. for tof usage

#### **Parameters**

t_dev	table of tof allocated outside the function
tof_number	number of tof currently used (between 1 and 8)

Warning

You have to allocate t\_dev before calling this function

# 6.12 lowlevel/include/tof\_timer.h File Reference

This implements a routine for periodic call of a function.

```
#include <libopencm3/stm32/timer.h>
#include <libopencm3/cm3/nvic.h>
#include "timer.h"
#include "uart.h"
#include "tof.h"
```

### **Macros**

- #define TOF\_TIM\_PRESCALER (42000)
- #define TOF\_TIM\_PERIOD (50)
- #define TOF\_TIM\_RCC RCC TIM4
- #define TOF\_TIM TIM4
- #define TOF\_TIM\_NVIC NVIC TIM4 IRQ
- #define TOF\_TIM\_DIER\_UIE TIM\_DIER\_UIE
- #define TOF\_TIM\_SR\_UIF TIM\_SR\_UIF

### **Functions**

```
    void timer_setup_interrupt ()
        setup a regular interruption routine
    void tim4_isr ()
        definition of the interrupt routine
```

### 6.12.1 Detailed Description

This implements a routine for periodic call of a function.

This file is part of cdfr2020CarteCerveauProg

#### See also

```
reference used code by Ken Sarkies ksarkies@trinity.asn.au: https://github. ← com/ksarkies/ARM-Ports/blob/master/test-libopencm3-stm32f1/timer-interrupt-oc-et-stm c
```

Date

12/2020

Licence:

Robotronik Phelma

Author

NPXav Benano

### 6.12.2 Macro Definition Documentation

### 6.12.2.1 TOF\_TIM\_PERIOD

```
#define TOF_TIM_PERIOD (50)
```

Period for the Timer [ms]

### 6.12.2.2 TOF\_TIM\_PRESCALER

```
#define TOF_TIM_PRESCALER (42000)
```

Prescale 84000000 Hz system clock by 84000 = 1000 Hz.

### 6.12.3 Function Documentation

#### 6.12.3.1 tim4\_isr()

```
void tim4_isr ( )
```

definition of the interrupt routine

#### Warning

You may need to edit this function to change the interrupt routine for the given functionality

#### 6.12.3.2 timer\_setup\_interrupt()

```
void timer_setup_interrupt ( )
```

setup a regular interruption routine

### 6.13 lowlevel/include/uart.h File Reference

This implements the setup of the actuators: the arm and the flag.

```
#include <stdarg.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <libopencm3/stm32/usart.h>
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/exti.h>
#include <libopencm3/stm32/exti.h>
#include <libopencm3/cm3/nvic.h>
```

#### **Macros**

- #define DEBUG RCC USART RCC USART2
- #define DEBUG\_USART USART2
- #define DEBUG UART SPEED (9600)
- #define **DEBUG\_PORT\_TX** GPIOA
- #define DEBUG\_PORT\_TX\_RCC RCC\_GPIOA
- #define **DEBUG\_PIN\_TX** GPIO2
- #define DEBUG\_AF\_TX GPIO\_AF7
- #define DEBUG\_PORT RX GPIOA
- #define DEBUG PORT RX RCC RCC GPIOA
- #define DEBUG PIN RX GPIO3
- #define DEBUG\_AF\_RX GPIO\_AF7
- #define **DEBUG\_UART\_EXTI** EXTI26
- #define DEBUG UART NVIC NVIC USART2 IRQ
- #define COMM\_RCC\_USART RCC\_USART1
- #define COMM\_USART USART1
- #define COMM UART SPEED (9600)
- #define COMM PORT TX GPIOA
- #define COMM PORT TX RCC RCC GPIOB
- #define COMM PIN TX GPIO9
- #define COMM AF TX GPIO AF7
- #define COMM PORT RX GPIOA
- #define COMM\_PORT\_RX\_RCC RCC\_GPIOB
- #define COMM PIN RX GPIO10
- #define COMM AF RX GPIO AF7
- #define COMM UART EXTI EXTI25
- #define COMM UART NVIC NVIC USART1 IRQ

#### **Functions**

void uart setup ()

setup communication uart and debug uart(usb through the stlink)

• int \_write (int file, const char \*ptr, ssize\_t len)

implementation of write that redirects stdout on the communication uart and stderr on the debug uart This function is never actually called by us: use fprintf and fscanf to communicate

• int \_read (int file, char \*ptr, ssize\_t len)

implementation of read that redirects stdout on the communication uart and stderr on the debug uart This function is never actually called by us: use fprintf and fscanf to communicate

### 6.13.1 Detailed Description

This implements the setup of the actuators: the arm and the flag.

This file is part of cdfr2020CarteCerveauProg

Date

07/2020

Licence:

#include "canmsgs.h" \* Robotronik Phelma

**Author** 

NPXav Benano

### 6.13.2 Function Documentation

### 6.13.2.1 \_read()

implementation of read that redirects stdout on the communication uart and stderr on the debug uart This function is never actually called by us: use fprintf and fscanf to communicate

#### **Parameters**

file	
ptr	
len	

#### Returns

int

### 6.13.2.2 \_write()

implementation of write that redirects stdout on the communication uart and stderr on the debug uart This function is never actually called by us: use fprintf and fscanf to communicate

### Parameters

file	
ptr	
len	

#### Returns

int

# 6.13.2.3 uart\_setup()

```
void uart_setup ( )
```

setup communication uart and debug uart(usb through the stlink)

# Index

pulse	actuator set
gpio.h, 36	actuator.h, 28
_can_msg_buffer_append	actuator_setup
canmsgs.h, 31	actuator.h, 29
_gpio_setup_pin	actuator tim, 7
gpio.h, 38	arm, 8
_gpio_setup_pin_af	arm_limit_switch, 10
gpio.h, 38	ao
limit switch init	can_msg_buffer_list_t, 19
exti.c, 25	can_msg_buffer_pop
exti.h, 35	canmsgs.h, 31
read	Can_rx_msg, 19
uart.h, 57	can setup
_shift_reg	canmsgs.c, 24
	canmsgs.h, 31
_shift_reg_init	Can_tx_msg, 20
tof.h, 46	can_tx_msg, 21
_timer_setup	canmsgs.c
timer.h, 43	can_setup, 24
_timer_setup_output_c	canmsgs.h
timer.h, 43	_can_msg_buffer_append, 31
_timer_start	can_msg_buffer_pop, 31
timer.h, 43	can setup, 31
_tof_1_setup	cec_can_isr, 31
tof.h, 46	receive, 31
tof calibration	transmit, 32
tof.h, 47	cec can isr
_tof_config	canmsgs.h, 31
tof.h, 47	clock.h
_tof_init_struct	clock_setup, 33
tof.h, 47	delay_ms, 33
_tof_poke	clock_setup
tof.h, 48	clock.h, 33
tof set address	,
tof.h, 48	debug_uart, 18
_tof_setup_addr	delay_ms
tof.h, 48	clock.h, 33
_tof_setup_calib	
tof.h, 49	exti.c
write	_limit_switch_init, 25
uart.h, 57	exti9_5_isr, 25
	exti_setup, 25
actuator.h	exti.h
actuator_sel, 28	_limit_switch_init, 35
actuator_set, 28	exti9_5_isr, 35
actuator_setup, 29	exti_setup, 35
PWM_PERIOD, 28	exti9_5_isr
PWM_PRESCALE, 28	exti.c, 25
actuator_sel	exti.h, 35
actuator.h, 28	exti_setup

60 INDEX

exti.c, 25	tof_timer.h, 55
exti.h, 35	timer.h
flog 0	_timer_setup, 43
flag, 9 flag limit switch, 11	_timer_setup_output_c, 43
<del>-</del>	_timer_start, 43
flash_memory, 12	timer_setup_interrupt
flash_program_data, 12 flash_read_data, 13	tof_timer.h, 55
flash program data	tof.h
flash_memory, 12	_shift_reg, 46
flash read data	_shift_reg_init, 46
flash memory, 13	_tof_1_setup, 46
nasi_memory, re	_tof_calibration, 47
gpio.h	_tof_config, 47
pulse, 36	_tof_init_struct, 47
 _gpio_setup_pin, 38	_tof_poke, 48
_gpio_setup_pin_af, 38	_tof_set_address, 48
	_tof_setup_addr, 48
I2C, 14	_tof_setup_calib, 49
i2c.h	tof_perform_measure, 49
i2c_read7, 40	tof_print_calib_info, 50
i2c_setup, 41	tof_print_data_measure, 50
I2C_status, 40	tof_print_device_info, 50
I2C_Status_E, 40	tof_print_device_mode, 51
i2c_write7, 41	tof_print_int_status, 51
i2c_read7	tof_print_PAL_state, 52
i2c.h, 40	tof_print_ranging_status, 52
i2c_setup	tof_reset, 53
i2c.h, 41	tof_setup, 53
I2C_status	tof_perform_measure
i2c.h, 40	tof.h, 49
I2C_Status_E	tof_print_calib_info
i2c.h, 40	tof.h, 50
i2c_write7	tof_print_data_measure
i2c.h, 41	tof.h, 50
lowlevel/canmsgs.c, 23	tof_print_device_info
lowlevel/exti.c, 24	tof.h, 50
lowlevel/flash.c, 26	tof_print_device_mode tof.h, 51
lowlevel/include/actuator.h, 26	tof_print_int_status
lowlevel/include/canmsgs.h, 29	tof_print_int_status tof.h, 51
lowlevel/include/clock.h, 32	tof_print_PAL_state
lowlevel/include/exti.h, 33	tof_print_1 A2_state tof.h, 52
lowlevel/include/gpio.h, 36	tof_print_ranging_status
lowlevel/include/i2c.h, 39	tof.h, 52
lowlevel/include/timer.h, 42	tof reset
lowlevel/include/tof.h, 44	tof.h, 53
lowlevel/include/tof_timer.h, 53	tof_setup
lowlevel/include/uart.h, 55	tof_setup tof.h, 53
	tof_shiftr, 15
PWM_PERIOD	tof tim, 17
actuator.h, 28	TOF TIM PERIOD
PWM_PRESCALE	tof_timer.h, 54
actuator.h, 28	TOF TIM PRESCALER
Panga Panging Profile 40	tof timer.h, 54
Range Ranging Profile, 16	tof_timer.h
receive	tim4 isr, 55
canmsgs.h, 31	timer_setup_interrupt, 55
tim4_isr	TOF TIM PERIOD, 54
	. J

INDEX 61

```
TOF_TIM_PRESCALER, 54
transmit
    canmsgs.h, 32

uart.h
    _read, 57
    _write, 57
    uart_setup, 57

uart_setup
    uart.h, 57

VL53L0X_Calibration_Parameter_S, 21
```