cdfr2020CarteCerveauProg

Generated by Doxygen 1.8.17

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

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

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

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

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 µcontroller 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()

```
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.

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)

4.6.2.2 flash_read_data()

```
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.

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

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

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

Chapter 5

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

Chapter 6

File Documentation

6.1 lowlevel/canmsgs.c File Reference

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/.

```
#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

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()

```
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.

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

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/fl_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.

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()

```
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.

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)

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()

```
void _can_msg_buffer_append (
            Can_rx_msg rx_msg )
```

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()

```
int can_msg_buffer_pop (
            Can_rx_msg * rx_msg )
```

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()

```
void transmit (
             uint32_t id,
             Can_tx_msg tx_msg )
```

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()

```
void delay_ms (
            uint32_t 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()

```
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.

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 _gpio_setup_pin_af (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()

```
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.

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()

```
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.

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()

```
void _gpio_setup_pin_af (
            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.

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 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()

```
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.

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()

```
void i2c_setup (
            uint32_t i2c_peripheral )
```

Set the application-specific I2C configuration.

Parameters

i2c_peripheral	I2C peripheral used (expected I2C1 or I2C2)

6.9.4.3 i2c_write7()

```
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.

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 _timer_start (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()

```
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.

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()

```
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.

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()

```
void _timer_start (
            uint32_t timer_peripheral )
```

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 "vl53l0x_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 _tof_init_struct (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()

```
VL53L0X_Error _tof_1_setup (
             VL53L0X_DEV dev,
             uint8_t tof_addr )
```

setup a tof

Parameters

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

6.11.2.4 _tof_calibration()

```
VL53L0X_Error _tof_calibration (
            VL53L0X_DEV dev,
            VL53L0X_Calibration_Parameter * calib_param,
            FixPoint1616_t offset_cal_distance,
            FixPoint1616_t xTalk_cal_distance )
```

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()

```
VL53L0X_Error _tof_config (
            VL53L0X_DEV dev )
```

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 (
            VL53L0X_DEV dev )
```

setup the structure with the standard address and I2C peripheral

Parameters

dev	our tof object

6.11.2.7 _tof_poke()

```
VL53L0X_Error _tof_poke (
            VL53L0X_DEV dev )
```

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()

```
VL53L0X_Error _tof_set_address (
            VL53L0X_DEV dev,
            uint8_t addr )
```

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()

```
VL53L0X_Error _tof_setup_addr (
            VL53L0X_DEV dev,
            uint8_t 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()

```
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)

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()

```
VL53L0X_Error tof_perform_measure (
            VL53L0X_DEV dev )
```

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()

```
VL53L0X_Error tof_print_calib_info (
            VL53L0X_DEV dev )
```

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()

```
VL53L0X_Error tof_print_data_measure (
            VL53L0X_RangingMeasurementData_t measure_data )
```

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()

```
VL53L0X_Error tof_print_device_info (
            VL53L0X_DEV dev )
```

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()

```
VL53L0X_Error tof_print_device_mode (
            VL53L0X_DEV dev )
```

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 (l181 vl53l0x_def.h)

6.11.2.16 tof_print_int_status()

```
VL53L0X_Error tof_print_int_status (
            VL53L0X_DEV dev )
```

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 (l152 vl53l0x_device.h)

6.11.2.17 tof_print_PAL_state()

```
VL53L0X_Error tof_print_PAL_state (
            VL53L0X_DEV dev )
```

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 (l273 vl53l0x_def.h)

6.11.2.18 tof_print_ranging_status()

```
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

measure_data	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()

```
VL53L0X_Error tof_setup (
            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/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()

```
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

Parameters

file	
ptr	
len	

Returns

int

6.13.2.2 _write()

```
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

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