cdfr2020CarteCerveauProg

Generated by Doxygen 1.8.20

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Module Documentation

3.1 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

3.1.1 Detailed Description

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

Two channels are used for the ARM and FLAG

3.2 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)
- #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)

3.2.1 Detailed Description

Definitions for the arm.

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 PC7 (with TIM3_CH2), DIR_1 on PB12, DIR_2 on PB13

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

Chapter 4

File Documentation

4.1 lowlevel/include/actuator.h File Reference

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

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

4.1.1 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

4.1.2 Macro Definition Documentation

4.1.2.1 PWM_PERIOD

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

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

4.1.2.2 PWM_PRESCALE

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

Prescale 64000000 Hz system clock by 64 = 1000000 Hz.

4.1.3 Function Documentation

4.1.3.1 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)

4.1.3.2 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.

4.2 lowlevel/include/clock.h File Reference

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

```
#include <stdint.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.

4.2.1 Detailed Description

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

This file is part of cdfr2020CerveauProg

Date

07/2020

Licence :

Robotronik Phelma

Author

PhenixRobotik NPXav Benano Trukbidule

4.2.2 Function Documentation

4.2.2.1 delay_ms()

```
void delay_ms (
             uint32_t ms )
```

This function implements a delay in ms.

Parameters

ms	value of delay in ms

4.3 lowlevel/include/gpio.h File Reference

This implements the setup of a gpio pin

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

Functions

- void gpio_setup_pin_af (enum rcc_periph_clken rcc_clken, uint32_t gpio_port, uint16_t gpio_pin, uint8_↩
  t gpio_altfun)

  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)

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

4.3.1 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

4.3.2 Function Documentation

4.3.2.1 _gpio_setup_pin()

```
void _gpio_setup_pin (
            enum rcc_periph_clken clken,
            uint32_t port,
            uint16_t pin,
            uint8_t mode )
```

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_OUTPUT,GPIO_MODE_OUTPUT)

4.3.2.2 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 )
```

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
gpio_altfun	identifier for the alternate function (usualy GPIO_AFX with X the number for altfun)

4.4 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.

4.4.1 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

4.4.2 Function Documentation

4.4.2.1 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

4.4.2.2 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

4.4.2.3 timer_start()

```
void timer_start (
            uint32_t timer_peripheral )
```

This function starts the given timer.

Parameters

timer_peripheral	selected timer

4.5 lowlevel/sensor.c File Reference

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

```
#include "sensor.h"
```

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 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.
- void **exti9_5_isr** ()

4.5.1 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

4.5.2 Function Documentation

4.5.2.1 _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)

4.6 lowlevel/sensor.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"
```

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 **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

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.

4.6.1 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

4.6.2 Function Documentation

4.6.2.1 _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)

Index