actuator_controller

v1.0

Generated by Doxygen 1.9.8

Chapter 1

actuator_controller

Actuator controller on STM32

Electronic control unit is based on an STM32 MCU. The ECU controls a linear actuator. The actuator can be extended, shrinked or kept in steady state. Moving the actuator is controlled by two digital outputs (GPIO). Setting one pin enables moving in one direction, another - in opposite direction. Endpoints of the actuator's working range are configured mechanically by adjusting 2 end switches. Actuator can pass the switches. Moving velocity is constant (and probably not equal) in both directions.

For testing 2 LEDs and 2 buttons are used. LEDs are connected to the control outputs and indicate movement of the actuator. Buttons are used to imitate the end switches.

Here the Bluepill on STM32F103C8T6 is used for evaluation

Project files are provided for Keil and for plain Linux Makefile project (gnu-arm-none-eabi-gcc) Files are made by STM32CubeMX wizard.

1.1 Pinout:

Forward output: PB15 Backward output: PB14

Forward endpoint: PB13 (normally set to 1 by pull up) Backward endpoint:.PB12.(normally set to 1 by pull up)

1.2 Prototype picture

1.3 Building

Use MDK-ARM v5 or Linux machine with Make (or eclipse with make environment). Open project or write "make" in project folder.

1.4 Evaluation

Normally started, we check for endpoints status.

- If no switched endpoints, move backwards until endpoint. Timeout is set to 10 seconds. After touching backward endpoint we calibrate forward velocity by measuring time, and then backward velocity again by measuring time.
- If we are on forward endpoint, move backward is in previous case
- If we are on backward endpoint, move forward until endpoint, and then measure backward and forward velocities by measured time.
- From the last endpoint, move in opposite direction by corresponding time / 2. Stop here and treat it as a middle point.

If any error while working, actuator stops and blinker on PC13 is activated. Direction inputs are debounced by 20 samples.

To evaluate such a behaviour, press a backward endpoint button soon after start, then press forward endpoint button, and then again backward point(calibration of time). Then wait for middle point position.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Topic Documentation

4.1 CMSIS

Modules

- Stm32f1xx_system

4.1.1 Detailed Description

4.1.2 Stm32f1xx_system

Modules

- STM32F1xx_System_Private_Includes
- STM32F1xx_System_Private_TypesDefinitions
- STM32F1xx_System_Private_Defines
- STM32F1xx_System_Private_Macros
- STM32F1xx_System_Private_Variables
- STM32F1xx_System_Private_FunctionPrototypes
- STM32F1xx_System_Private_Functions

4.1.2.1 Detailed Description

4.1.2.2 STM32F1xx_System_Private_Includes

4.1.2.3 STM32F1xx_System_Private_TypesDefinitions

4.1.2.4 STM32F1xx_System_Private_Defines

Macros

- #define HSE_VALUE 8000000U
- #define HSI_VALUE 8000000U

4.1.2.4.1 Detailed Description

4.1.2.4.2 Macro Definition Documentation

4.1.2.4.2.1 HSE_VALUE

```
#define HSE_VALUE 8000000U
```

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

4.1.2.4.2.2 HSI_VALUE

```
#define HSI_VALUE 8000000U
```

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

4.1.2.5 STM32F1xx_System_Private_Macros

4.1.2.6 STM32F1xx_System_Private_Variables

Variables

- uint32_t **SystemCoreClock** = 16000000
- const uint8_t **AHBPrescTable** [16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8U] = {0, 0, 0, 0, 1, 2, 3, 4}

4.1.2.6.1 Detailed Description

4.1.2.7 STM32F1xx_System_Private_FunctionPrototypes

4.1.2.8 STM32F1xx_System_Private_Functions

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.
- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

4.1.2.8.1 Detailed Description

4.1.2.8.2 Function Documentation

4.1.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f1xx.h file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f1xx.h file (default value 8 MHz or 25 MHz, depending on the product used), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

4.1.2.8.2.2 SystemInit()

```
void SystemInit (
            void  )
```

Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.

Note

This function should be used only after reset.

Parameters

None	

Return values

None	

Chapter 5

File Documentation

5.1 Core/Inc/actuator.h File Reference

Actuator header file for actuator controller project.

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

Macros

- #define **NO_ENDSWITCH** 0
  State of no endswitch signal.
- #define **ENDSWITCH_BACKWARD** 1
  Backward endswitch signal.
- #define **ENDSWITCH_FORWARD** 2
  Forward endswitch signal.
- #define **ENDSWITCH_ERROR** 3
  Endswitch error.
- #define **ACTUATOR_TIMEOUT_IN_TICKS** 10000
- #define **MOVE_ACTUATOR_IDLE** 0
  Idle operation.
- #define **MOVE_ACTUATOR_BACKWARD** 1
  Backward movement operation.
- #define **MOVE_ACTUATOR_FORWARD** 2
  Forward movement operation.

Enumerations

- enum ActuatorState {
  IDLE , CALIBRATION_HOME_FORWARD , CALIBRATION_HOME_BACKWARD , CALIBRATION_SPEED_FORWARD
  ,
  CALIBRATION_SPEED_BACKWARD , MOVING_MIDDLE , ACTUATOR_MIDDLE , POWERUP_UNKNOWN
  ,
  ACTUATOR_ERROR }
  Actuator state is for iterative processing of actuator actions.

Functions

- void **actuator_iteration** (void)

  Main function for processing actuator events.
- void process_unknown (uint8_t endswitch_state)

  Function to process unknown(powerup) state of actuator.
- void process_calibration_home_backward (uint8_t endswitch_state)

  Function to home actuator backward after poweron.
- void process_calibration_home_forward (uint8_t endswitch_state)

  Function to home actuator forward after poweron.
- void process_calibration_speed_forward (uint8_t endswitch_state)

  Function to calibrate forward speed.
- void process_calibration_speed_backward (uint8_t endswitch_state)

  Function to calibrate backward speed.
- void process_moving_middle (uint8_t endswitch_state)

  Function to center actuator between two endpoints.
- void move_actuator (uint8_t operation)

  Function to set actuator outputs.
- uint8_t get_end_switch_state (void)

  Get the end switch state object.

5.1.1 Detailed Description

Actuator header file for actuator controller project.

Author

Fedor Zagumennov

Version

0.1

Date

2023-09-06

Copyright

5.1.2 Enumeration Type Documentation

5.1.2.1 ActuatorState

```
enum ActuatorState
```

Actuator state is for iterative processing of actuator actions.

Enumerator

IDLE	Idle, no movement.
CALIBRATION_HOME_FORWARD	Home to forward after power on.
CALIBRATION_HOME_BACKWARD	Home to backward after power on.
CALIBRATION_SPEED_FORWARD	Calibrate time of backward to forward movement.
CALIBRATION_SPEED_BACKWARD	Calibrate time of forward to backward movement.
MOVING_MIDDLE	Waiting for middle point.
ACTUATOR_MIDDLE	State of being in middle position.
POWERUP_UNKNOWN	State after powerup.
ACTUATOR_ERROR	Calibrate time of backward to forward movement.

5.1.3 Function Documentation

5.1.3.1 get_end_switch_state()

```
uint8_t get_end_switch_state (
            void  )
```

Get the end switch state object.

Returns

uint8_t

5.1.3.2 move_actuator()

```
void move_actuator (
            uint8_t operation )
```

Function to set actuator outputs.

Parameters

operation	

5.1.3.3 process_calibration_home_backward()

```
void process_calibration_home_backward (
            uint8_t endswitch_state )
```

Function to home actuator backward after poweron.

Parameters

endswitch_state	

5.1.3.4 process_calibration_home_forward()

```
void process_calibration_home_forward (
            uint8_t endswitch_state )
```

Function to home actuator forward after poweron.

Parameters

endswitch_state	

5.1.3.5 process_calibration_speed_backward()

```
void process_calibration_speed_backward (
            uint8_t endswitch_state )
```

Function to calibrate backward speed.

Parameters

endswitch_state	

5.1.3.6 process_calibration_speed_forward()

```
void process_calibration_speed_forward (
            uint8_t endswitch_state )
```

Function to calibrate forward speed.

Parameters

endswitch_state	

5.1.3.7 process_moving_middle()

```
void process_moving_middle (
            uint8_t endswitch_state )
```

Function to center actuator between two endpoints.

Parameters

endswitch_state	

5.1.3.8 process_unknown()

```
void process_unknown (
            uint8_t endswitch_state )
```

Function to process unknown(powerup) state of actuator.

Parameters

endswitch_state	

5.2 actuator.h

Go to the documentation of this file.

```
00001
00012 #ifndef ACTUATOR_H
00013 #define ACTUATOR_H
00014
00015 #include <stdint.h>
00016
00018 enum ActuatorState {
00019     IDLE,
00020     CALIBRATION_HOME_FORWARD,
00021     CALIBRATION_HOME_BACKWARD,
00022     CALIBRATION_SPEED_FORWARD,
00023     CALIBRATION_SPEED_BACKWARD,
00024     MOVING_MIDDLE,
00025     ACTUATOR_MIDDLE,
00026     POWERUP_UNKNOWN,
00027     ACTUATOR_ERROR
00028 };
00029
00030 #define NO_ENDSWITCH 0
00031 #define ENDSWITCH_BACKWARD 1
00032 #define ENDSWITCH_FORWARD 2
00033 #define ENDSWITCH_ERROR 3
00034
00035 #define ACTUATOR_TIMEOUT_IN_TICKS 10000 //10s
00036 #define MOVE_ACTUATOR_IDLE 0
00037 #define MOVE_ACTUATOR_BACKWARD 1
00038 #define MOVE_ACTUATOR_FORWARD 2
00039
00044 void actuator_iteration(void);
00045
00051 void process_unknown(uint8_t endswitch_state);
00057 void process_calibration_home_backward(uint8_t endswitch_state);
00063 void process_calibration_home_forward(uint8_t endswitch_state);
00069 void process_calibration_speed_forward(uint8_t endswitch_state);
00075 void process_calibration_speed_backward(uint8_t endswitch_state);
00081 void process_moving_middle(uint8_t endswitch_state);
00082
00088 void move_actuator(uint8_t operation);
00094 uint8_t get_end_switch_state(void);
00095
00096 #endif
```

5.3 Core/Inc/main.h File Reference

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

```
#include "stm32f1xx_hal.h"
```

Macros

- #define **LED_Pin** GPIO_PIN_13
- #define **LED_GPIO_Port** GPIOC
- #define **ENDSWITCH_BACKWARD_Pin** GPIO_PIN_12
- #define **ENDSWITCH_BACKWARD_GPIO_Port** GPIOB
- #define **ENDSWITCH_FORWARD_Pin** GPIO_PIN_13
- #define **ENDSWITCH_FORWARD_GPIO_Port** GPIOB
- #define **MOT_BACKWARD_Pin** GPIO_PIN_14
- #define **MOT_BACKWARD_GPIO_Port** GPIOB
- #define **MOT_FORWARD_Pin** GPIO_PIN_15
- #define **MOT_FORWARD_GPIO_Port** GPIOB

Functions

- void Error_Handler (void)

  This function is executed in case of error occurrence.

5.3.1 Detailed Description

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

Attention

5.3.2 Function Documentation

5.3.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

5.4 main.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -------------------------------------*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes ------------------------------------------------------------------*/
00030 #include "stm32f1xx_hal.h"
00031
00032 /* Private includes ----------------------------------------------------------*/
00033 /* USER CODE BEGIN Includes */
00034
00035 /* USER CODE END Includes */
00036
00037 /* Exported types ------------------------------------------------------------*/
00038 /* USER CODE BEGIN ET */
00039
00040 /* USER CODE END ET */
00041
00042 /* Exported constants --------------------------------------------------------*/
00043 /* USER CODE BEGIN EC */
00044
00045 /* USER CODE END EC */
00046
00047 /* Exported macro ------------------------------------------------------------*/
00048 /* USER CODE BEGIN EM */
00049
00050 /* USER CODE END EM */
00051
00052 /* Exported functions prototypes ---------------------------------------------*/
00053 void Error_Handler(void);
00054
00055 /* USER CODE BEGIN EFP */
00056
00057 /* USER CODE END EFP */
00058
00059 /* Private defines -----------------------------------------------------------*/
00060 #define LED_Pin GPIO_PIN_13
00061 #define LED_GPIO_Port GPIOC
00062 #define ENDSWITCH_BACKWARD_Pin GPIO_PIN_12
00063 #define ENDSWITCH_BACKWARD_GPIO_Port GPIOB
00064 #define ENDSWITCH_FORWARD_Pin GPIO_PIN_13
00065 #define ENDSWITCH_FORWARD_GPIO_Port GPIOB
00066 #define MOT_BACKWARD_Pin GPIO_PIN_14
00067 #define MOT_BACKWARD_GPIO_Port GPIOB
00068 #define MOT_FORWARD_Pin GPIO_PIN_15
00069 #define MOT_FORWARD_GPIO_Port GPIOB
00070
00071 /* USER CODE BEGIN Private defines */
00072
00073 /* USER CODE END Private defines */
00074
00075 #ifdef __cplusplus
00076 }
00077 #endif
00078
00079 #endif /* __MAIN_H */
```

5.5 setup.h

```
00001 #ifndef SETUP_H
00002 #define SETUP_H
00003
00004
00005
00006 #endif
```

5.6 Core/Inc/stm32f1xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32f1xx_hal_rcc.h"
#include "stm32f1xx_hal_gpio.h"
#include "stm32f1xx_hal_exti.h"
#include "stm32f1xx_hal_dma.h"
#include "stm32f1xx_hal_cortex.h"
#include "stm32f1xx_hal_flash.h"
#include "stm32f1xx_hal_pwr.h"
#include "stm32f1xx_hal_tim.h"
```

Macros

- #define **HAL_MODULE_ENABLED**

  This is the list of modules to be used in the HAL driver.
- #define **HAL_GPIO_MODULE_ENABLED**
- #define **HAL_TIM_MODULE_ENABLED**
- #define **HAL_CORTEX_MODULE_ENABLED**
- #define **HAL_DMA_MODULE_ENABLED**
- #define **HAL_FLASH_MODULE_ENABLED**
- #define **HAL_EXTI_MODULE_ENABLED**
- #define **HAL_GPIO_MODULE_ENABLED**
- #define **HAL_PWR_MODULE_ENABLED**
- #define **HAL_RCC_MODULE_ENABLED**
- #define HSE_VALUE 8000000U

  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSE_STARTUP_TIMEOUT 100U
- #define HSI_VALUE 8000000U

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define LSI_VALUE 40000U

  Internal Low Speed oscillator (LSI) value.
- #define LSE_VALUE 32768U

  External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.
- #define LSE_STARTUP_TIMEOUT 5000U
- #define VDD_VALUE 3300U

  This is the HAL system configuration section.
- #define TICK_INT_PRIORITY 15U
- #define **USE_RTOS** 0U
- #define **PREFETCH_ENABLE** 1U
- #define **USE_HAL_ADC_REGISTER_CALLBACKS** 0U /∗ ADC register callback disabled ∗/
- #define **USE_HAL_CAN_REGISTER_CALLBACKS** 0U /∗ CAN register callback disabled ∗/
- #define **USE_HAL_CEC_REGISTER_CALLBACKS** 0U /∗ CEC register callback disabled ∗/
- #define **USE_HAL_DAC_REGISTER_CALLBACKS** 0U /∗ DAC register callback disabled ∗/
- #define **USE_HAL_ETH_REGISTER_CALLBACKS** 0U /∗ ETH register callback disabled ∗/
- #define **USE_HAL_HCD_REGISTER_CALLBACKS** 0U /∗ HCD register callback disabled ∗/
- #define **USE_HAL_I2C_REGISTER_CALLBACKS** 0U /∗ I2C register callback disabled ∗/
- #define **USE_HAL_I2S_REGISTER_CALLBACKS** 0U /∗ I2S register callback disabled ∗/
- #define **USE_HAL_MMC_REGISTER_CALLBACKS** 0U /∗ MMC register callback disabled ∗/
- #define **USE_HAL_NAND_REGISTER_CALLBACKS** 0U /∗ NAND register callback disabled ∗/
- #define **USE_HAL_NOR_REGISTER_CALLBACKS** 0U /∗ NOR register callback disabled ∗/
- #define **USE_HAL_PCCARD_REGISTER_CALLBACKS** 0U /∗ PCCARD register callback disabled ∗/
- #define **USE_HAL_PCD_REGISTER_CALLBACKS** 0U /∗ PCD register callback disabled ∗/

- #define **USE_HAL_RTC_REGISTER_CALLBACKS** 0U /* RTC register callback disabled */
- #define **USE_HAL_SD_REGISTER_CALLBACKS** 0U /* SD register callback disabled */
- #define **USE_HAL_SMARTCARD_REGISTER_CALLBACKS** 0U /* SMARTCARD register callback disabled */
- #define **USE_HAL_IRDA_REGISTER_CALLBACKS** 0U /* IRDA register callback disabled */
- #define **USE_HAL_SRAM_REGISTER_CALLBACKS** 0U /* SRAM register callback disabled */
- #define **USE_HAL_SPI_REGISTER_CALLBACKS** 0U /* SPI register callback disabled */
- #define **USE_HAL_TIM_REGISTER_CALLBACKS** 0U /* TIM register callback disabled */
- #define **USE_HAL_UART_REGISTER_CALLBACKS** 0U /* UART register callback disabled */
- #define **USE_HAL_USART_REGISTER_CALLBACKS** 0U /* USART register callback disabled */
- #define **USE_HAL_WWDG_REGISTER_CALLBACKS** 0U /* WWDG register callback disabled */
- #define **MAC_ADDR0** 2U

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define **MAC_ADDR1** 0U
- #define **MAC_ADDR2** 0U
- #define **MAC_ADDR3** 0U
- #define **MAC_ADDR4** 0U
- #define **MAC_ADDR5** 0U
- #define **ETH_RX_BUF_SIZE** ETH_MAX_PACKET_SIZE /* buffer size for receive */
- #define **ETH_TX_BUF_SIZE** ETH_MAX_PACKET_SIZE /* buffer size for transmit */
- #define **ETH_RXBUFNB** 8U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
- #define **ETH_TXBUFNB** 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
- #define **DP83848_PHY_ADDRESS** 0x01U
- #define **PHY_RESET_DELAY** 0x000000FFU
- #define **PHY_CONFIG_DELAY** 0x00000FFFU
- #define **PHY_READ_TO** 0x0000FFFFU
- #define **PHY_WRITE_TO** 0x0000FFFFU
- #define PHY_BCR ((uint16_t)0x00)
- #define PHY_BSR ((uint16_t)0x01)
- #define PHY_RESET ((uint16_t)0x8000)
- #define PHY_LOOPBACK ((uint16_t)0x4000)
- #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100)
- #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000)
- #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100)
- #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000)
- #define PHY_AUTONEGOTIATION ((uint16_t)0x1000)
- #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200)
- #define PHY_POWERDOWN ((uint16_t)0x0800)
- #define PHY_ISOLATE ((uint16_t)0x0400)
- #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020)
- #define PHY_LINKED_STATUS ((uint16_t)0x0004)
- #define PHY_JABBER_DETECTION ((uint16_t)0x0002)
- #define PHY_SR ((uint16_t)0x10U)
- #define PHY_SPEED_STATUS ((uint16_t)0x0002U)
- #define PHY_DUPLEX_STATUS ((uint16_t)0x0004U)
- #define **USE_SPI_CRC** 0U
- #define **assert_param**(expr) ((void)0U)

  Include module's header file.

5.6.1 Detailed Description

HAL configuration file.

Attention

5.6.2 Macro Definition Documentation

5.6.2.1 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT 100U
```

Time out for HSE start up, in ms

5.6.2.2 HSE_VALUE

```
#define HSE_VALUE 8000000U
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

5.6.2.3 HSI_VALUE

```
#define HSI_VALUE 8000000U
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

5.6.2.4 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

5.6.2.5 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.

< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External oscillator in Hz

5.6.2.6 LSI_VALUE

```
#define LSI_VALUE 40000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

5.6.2.7 PHY_AUTONEGO_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020)
```

Auto-Negotiation process completed

5.6.2.8 PHY_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000)
```

Enable auto-negotiation function

5.6.2.9 PHY_BCR

```
#define PHY_BCR ((uint16_t)0x00)
```

Transceiver Basic Control Register

5.6.2.10 PHY_BSR

```
#define PHY_BSR ((uint16_t)0x01)
```

Transceiver Basic Status Register

5.6.2.11 PHY_DUPLEX_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t)0x0004U)
```

PHY Duplex mask

5.6.2.12 PHY_FULLDUPLEX_100M

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100)
```

Set the full-duplex mode at 100 Mb/s

5.6.2.13 PHY_FULLDUPLEX_10M

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100)
```

Set the full-duplex mode at 10 Mb/s

5.6.2.14 PHY_HALFDUPLEX_100M

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000)
```

Set the half-duplex mode at 100 Mb/s

5.6.2.15 PHY_HALFDUPLEX_10M

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000)
```

Set the half-duplex mode at 10 Mb/s

5.6.2.16 PHY_ISOLATE

```
#define PHY_ISOLATE ((uint16_t)0x0400)
```

Isolate PHY from MII

5.6.2.17 PHY_JABBER_DETECTION

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002)
```

Jabber condition detected

5.6.2.18 PHY_LINKED_STATUS

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004)
```

Valid link established

5.6.2.19 PHY_LOOPBACK

```
#define PHY_LOOPBACK ((uint16_t)0x4000)
```

Select loop-back mode

5.6.2.20 PHY_POWERDOWN

```
#define PHY_POWERDOWN ((uint16_t)0x0800)
```

Select the power down mode

5.6.2.21 PHY_RESET

```
#define PHY_RESET ((uint16_t)0x8000)
```

PHY Reset

5.6.2.22 PHY_RESTART_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200)
```

Restart auto-negotiation function

5.6.2.23 PHY_SPEED_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t)0x0002U)
```

PHY Speed mask

5.6.2.24 PHY_SR

```
#define PHY_SR ((uint16_t)0x10U)
```

PHY status register Offset

5.6.2.25 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority (lowest by default)

5.6.2.26 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

5.7 stm32f1xx_hal_conf.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32F1xx_HAL_CONF_H
00022 #define __STM32F1xx_HAL_CONF_H
00023
00024 #ifdef __cplusplus
00025  extern "C" {
00026 #endif
00027
00028 /* Exported types ------------------------------------------------------------*/
00029 /* Exported constants --------------------------------------------------------*/
00030
00031 /* ########################## Module Selection ############################## */
00036 #define HAL_MODULE_ENABLED
00037   /*#define HAL_ADC_MODULE_ENABLED   */
00038 /*#define HAL_CRYP_MODULE_ENABLED   */
00039 /*#define HAL_CAN_MODULE_ENABLED   */
00040 /*#define HAL_CAN_LEGACY_MODULE_ENABLED   */
00041 /*#define HAL_CEC_MODULE_ENABLED   */
00042 /*#define HAL_CORTEX_MODULE_ENABLED   */
00043 /*#define HAL_CRC_MODULE_ENABLED   */
00044 /*#define HAL_DAC_MODULE_ENABLED   */
00045 /*#define HAL_DMA_MODULE_ENABLED   */
00046 /*#define HAL_ETH_MODULE_ENABLED   */
00047 /*#define HAL_FLASH_MODULE_ENABLED   */
00048 #define HAL_GPIO_MODULE_ENABLED
00049 /*#define HAL_I2C_MODULE_ENABLED   */
00050 /*#define HAL_I2S_MODULE_ENABLED   */
00051 /*#define HAL_IRDA_MODULE_ENABLED   */
00052 /*#define HAL_IWDG_MODULE_ENABLED   */
00053 /*#define HAL_NOR_MODULE_ENABLED   */
00054 /*#define HAL_NAND_MODULE_ENABLED   */
00055 /*#define HAL_PCCARD_MODULE_ENABLED   */
00056 /*#define HAL_PCD_MODULE_ENABLED   */
00057 /*#define HAL_HCD_MODULE_ENABLED   */
00058 /*#define HAL_PWR_MODULE_ENABLED   */
00059 /*#define HAL_RCC_MODULE_ENABLED   */
00060 /*#define HAL_RTC_MODULE_ENABLED   */
00061 /*#define HAL_SD_MODULE_ENABLED   */
00062 /*#define HAL_MMC_MODULE_ENABLED   */
00063 /*#define HAL_SDRAM_MODULE_ENABLED   */
00064 /*#define HAL_SMARTCARD_MODULE_ENABLED   */
00065 /*#define HAL_SPI_MODULE_ENABLED   */
00066 /*#define HAL_SRAM_MODULE_ENABLED   */
00067 #define HAL_TIM_MODULE_ENABLED
00068 /*#define HAL_UART_MODULE_ENABLED   */
00069 /*#define HAL_USART_MODULE_ENABLED   */
00070 /*#define HAL_WWDG_MODULE_ENABLED   */
00071
00072 #define HAL_CORTEX_MODULE_ENABLED
00073 #define HAL_DMA_MODULE_ENABLED
```

```
00074 #define HAL_FLASH_MODULE_ENABLED
00075 #define HAL_EXTI_MODULE_ENABLED
00076 #define HAL_GPIO_MODULE_ENABLED
00077 #define HAL_PWR_MODULE_ENABLED
00078 #define HAL_RCC_MODULE_ENABLED
00079
00080 /* ########################## Oscillator Values adaptation ####################*/
00086 #if !defined  (HSE_VALUE)
00087   #define HSE_VALUE    8000000U
00088 #endif /* HSE_VALUE */
00089
00090 #if !defined  (HSE_STARTUP_TIMEOUT)
00091   #define HSE_STARTUP_TIMEOUT    100U
00092 #endif /* HSE_STARTUP_TIMEOUT */
00093
00099 #if !defined  (HSI_VALUE)
00100   #define HSI_VALUE    8000000U
00101 #endif /* HSI_VALUE */
00102
00106 #if !defined  (LSI_VALUE)
00107  #define LSI_VALUE               40000U
00108 #endif /* LSI_VALUE */
00116 #if !defined  (LSE_VALUE)
00117   #define LSE_VALUE    32768U
00118 #endif /* LSE_VALUE */
00119
00120 #if !defined  (LSE_STARTUP_TIMEOUT)
00121   #define LSE_STARTUP_TIMEOUT    5000U
00122 #endif /* LSE_STARTUP_TIMEOUT */
00123
00124 /* Tip: To avoid modifying this file each time you need to use different HSE,
00125    ===  you can define the HSE value in your toolchain compiler preprocessor. */
00126
00127 /* ########################### System Configuration ######################### */
00131 #define  VDD_VALUE                    3300U
00132 #define  TICK_INT_PRIORITY            15U
00133 #define  USE_RTOS                     0U
00134 #define  PREFETCH_ENABLE              1U
00135
00136 #define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
00137 #define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
00138 #define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
00139 #define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
00140 #define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback disabled       */
00141 #define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback disabled       */
00142 #define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
00143 #define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
00144 #define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback disabled       */
00145 #define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback disabled      */
00146 #define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback disabled       */
00147 #define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback disabled    */
00148 #define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
00149 #define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
00150 #define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback disabled        */
00151 #define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
00152 #define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
00153 #define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback disabled      */
00154 #define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
00155 #define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
00156 #define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
00157 #define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
00158 #define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */
00159
00160 /* ########################## Assert Selection ############################## */
00165 /* #define USE_FULL_ASSERT    1U */
00166
00167 /* ################### Ethernet peripheral configuration ##################### */
00168
00169 /* Section 1 : Ethernet peripheral configuration */
00170
00171 /* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
00172 #define MAC_ADDR0   2U
00173 #define MAC_ADDR1   0U
00174 #define MAC_ADDR2   0U
00175 #define MAC_ADDR3   0U
00176 #define MAC_ADDR4   0U
00177 #define MAC_ADDR5   0U
00178
00179 /* Definition of the Ethernet driver buffers size and count */
00180 #define ETH_RX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for receive               */
00181 #define ETH_TX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for transmit              */
00182 #define ETH_RXBUFNB                    8U       /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
00183 #define ETH_TXBUFNB                    4U       /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */
00184
00185 /* Section 2: PHY configuration section */
00186
00187 /* DP83848_PHY_ADDRESS Address*/
```

```
00188 #define DP83848_PHY_ADDRESS           0x01U
00189 /* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
00190 #define PHY_RESET_DELAY                0x000000FFU
00191 /* PHY Configuration delay */
00192 #define PHY_CONFIG_DELAY               0x00000FFFU
00193
00194 #define PHY_READ_TO                    0x0000FFFFU
00195 #define PHY_WRITE_TO                   0x0000FFFFU
00196
00197 /* Section 3: Common PHY Registers */
00198
00199 #define PHY_BCR                        ((uint16_t)0x00)
00200 #define PHY_BSR                        ((uint16_t)0x01)
00202 #define PHY_RESET                      ((uint16_t)0x8000)
00203 #define PHY_LOOPBACK                   ((uint16_t)0x4000)
00204 #define PHY_FULLDUPLEX_100M            ((uint16_t)0x2100)
00205 #define PHY_HALFDUPLEX_100M            ((uint16_t)0x2000)
00206 #define PHY_FULLDUPLEX_10M             ((uint16_t)0x0100)
00207 #define PHY_HALFDUPLEX_10M             ((uint16_t)0x0000)
00208 #define PHY_AUTONEGOTIATION            ((uint16_t)0x1000)
00209 #define PHY_RESTART_AUTONEGOTIATION    ((uint16_t)0x0200)
00210 #define PHY_POWERDOWN                  ((uint16_t)0x0800)
00211 #define PHY_ISOLATE                    ((uint16_t)0x0400)
00213 #define PHY_AUTONEGO_COMPLETE          ((uint16_t)0x0020)
00214 #define PHY_LINKED_STATUS              ((uint16_t)0x0004)
00215 #define PHY_JABBER_DETECTION           ((uint16_t)0x0002)
00217 /* Section 4: Extended PHY Registers */
00218 #define PHY_SR                         ((uint16_t)0x10U)
00220 #define PHY_SPEED_STATUS               ((uint16_t)0x0002U)
00221 #define PHY_DUPLEX_STATUS              ((uint16_t)0x0004U)
00223 /* ################## SPI peripheral configuration ########################## */
00224
00225 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00226 * Activated: CRC code is present inside driver
00227 * Deactivated: CRC code cleaned from driver
00228 */
00229
00230 #define USE_SPI_CRC                    0U
00231
00232 /* Includes ------------------------------------------------------------------*/
00237 #ifdef HAL_RCC_MODULE_ENABLED
00238 #include "stm32f1xx_hal_rcc.h"
00239 #endif /* HAL_RCC_MODULE_ENABLED */
00240
00241 #ifdef HAL_GPIO_MODULE_ENABLED
00242 #include "stm32f1xx_hal_gpio.h"
00243 #endif /* HAL_GPIO_MODULE_ENABLED */
00244
00245 #ifdef HAL_EXTI_MODULE_ENABLED
00246 #include "stm32f1xx_hal_exti.h"
00247 #endif /* HAL_EXTI_MODULE_ENABLED */
00248
00249 #ifdef HAL_DMA_MODULE_ENABLED
00250 #include "stm32f1xx_hal_dma.h"
00251 #endif /* HAL_DMA_MODULE_ENABLED */
00252
00253 #ifdef HAL_ETH_MODULE_ENABLED
00254 #include "stm32f1xx_hal_eth.h"
00255 #endif /* HAL_ETH_MODULE_ENABLED */
00256
00257 #ifdef HAL_CAN_MODULE_ENABLED
00258 #include "stm32f1xx_hal_can.h"
00259 #endif /* HAL_CAN_MODULE_ENABLED */
00260
00261 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00262   #include "Legacy/stm32f1xx_hal_can_legacy.h"
00263 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00264
00265 #ifdef HAL_CEC_MODULE_ENABLED
00266 #include "stm32f1xx_hal_cec.h"
00267 #endif /* HAL_CEC_MODULE_ENABLED */
00268
00269 #ifdef HAL_CORTEX_MODULE_ENABLED
00270 #include "stm32f1xx_hal_cortex.h"
00271 #endif /* HAL_CORTEX_MODULE_ENABLED */
00272
00273 #ifdef HAL_ADC_MODULE_ENABLED
00274 #include "stm32f1xx_hal_adc.h"
00275 #endif /* HAL_ADC_MODULE_ENABLED */
00276
00277 #ifdef HAL_CRC_MODULE_ENABLED
00278 #include "stm32f1xx_hal_crc.h"
00279 #endif /* HAL_CRC_MODULE_ENABLED */
00280
00281 #ifdef HAL_DAC_MODULE_ENABLED
00282 #include "stm32f1xx_hal_dac.h"
00283 #endif /* HAL_DAC_MODULE_ENABLED */
```

```
00284
00285 #ifdef HAL_FLASH_MODULE_ENABLED
00286 #include "stm32f1xx_hal_flash.h"
00287 #endif /* HAL_FLASH_MODULE_ENABLED */
00288
00289 #ifdef HAL_SRAM_MODULE_ENABLED
00290 #include "stm32f1xx_hal_sram.h"
00291 #endif /* HAL_SRAM_MODULE_ENABLED */
00292
00293 #ifdef HAL_NOR_MODULE_ENABLED
00294 #include "stm32f1xx_hal_nor.h"
00295 #endif /* HAL_NOR_MODULE_ENABLED */
00296
00297 #ifdef HAL_I2C_MODULE_ENABLED
00298 #include "stm32f1xx_hal_i2c.h"
00299 #endif /* HAL_I2C_MODULE_ENABLED */
00300
00301 #ifdef HAL_I2S_MODULE_ENABLED
00302 #include "stm32f1xx_hal_i2s.h"
00303 #endif /* HAL_I2S_MODULE_ENABLED */
00304
00305 #ifdef HAL_IWDG_MODULE_ENABLED
00306 #include "stm32f1xx_hal_iwdg.h"
00307 #endif /* HAL_IWDG_MODULE_ENABLED */
00308
00309 #ifdef HAL_PWR_MODULE_ENABLED
00310 #include "stm32f1xx_hal_pwr.h"
00311 #endif /* HAL_PWR_MODULE_ENABLED */
00312
00313 #ifdef HAL_RTC_MODULE_ENABLED
00314 #include "stm32f1xx_hal_rtc.h"
00315 #endif /* HAL_RTC_MODULE_ENABLED */
00316
00317 #ifdef HAL_PCCARD_MODULE_ENABLED
00318 #include "stm32f1xx_hal_pccard.h"
00319 #endif /* HAL_PCCARD_MODULE_ENABLED */
00320
00321 #ifdef HAL_SD_MODULE_ENABLED
00322 #include "stm32f1xx_hal_sd.h"
00323 #endif /* HAL_SD_MODULE_ENABLED */
00324
00325 #ifdef HAL_NAND_MODULE_ENABLED
00326 #include "stm32f1xx_hal_nand.h"
00327 #endif /* HAL_NAND_MODULE_ENABLED */
00328
00329 #ifdef HAL_SPI_MODULE_ENABLED
00330 #include "stm32f1xx_hal_spi.h"
00331 #endif /* HAL_SPI_MODULE_ENABLED */
00332
00333 #ifdef HAL_TIM_MODULE_ENABLED
00334 #include "stm32f1xx_hal_tim.h"
00335 #endif /* HAL_TIM_MODULE_ENABLED */
00336
00337 #ifdef HAL_UART_MODULE_ENABLED
00338 #include "stm32f1xx_hal_uart.h"
00339 #endif /* HAL_UART_MODULE_ENABLED */
00340
00341 #ifdef HAL_USART_MODULE_ENABLED
00342 #include "stm32f1xx_hal_usart.h"
00343 #endif /* HAL_USART_MODULE_ENABLED */
00344
00345 #ifdef HAL_IRDA_MODULE_ENABLED
00346 #include "stm32f1xx_hal_irda.h"
00347 #endif /* HAL_IRDA_MODULE_ENABLED */
00348
00349 #ifdef HAL_SMARTCARD_MODULE_ENABLED
00350 #include "stm32f1xx_hal_smartcard.h"
00351 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00352
00353 #ifdef HAL_WWDG_MODULE_ENABLED
00354 #include "stm32f1xx_hal_wwdg.h"
00355 #endif /* HAL_WWDG_MODULE_ENABLED */
00356
00357 #ifdef HAL_PCD_MODULE_ENABLED
00358 #include "stm32f1xx_hal_pcd.h"
00359 #endif /* HAL_PCD_MODULE_ENABLED */
00360
00361 #ifdef HAL_HCD_MODULE_ENABLED
00362 #include "stm32f1xx_hal_hcd.h"
00363 #endif /* HAL_HCD_MODULE_ENABLED */
00364
00365 #ifdef HAL_MMC_MODULE_ENABLED
00366 #include "stm32f1xx_hal_mmc.h"
00367 #endif /* HAL_MMC_MODULE_ENABLED */
00368
00369 /* Exported macro ------------------------------------------------------------*/
00370 #ifdef  USE_FULL_ASSERT
```

```
00379 #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00380 /* Exported functions ------------------------------------------------------- */
00381 void assert_failed(uint8_t* file, uint32_t line);
00382 #else
00383 #define assert_param(expr) ((void)0U)
00384 #endif /* USE_FULL_ASSERT */
00385
00386 #ifdef __cplusplus
00387 }
00388 #endif
00389
00390 #endif /* __STM32F1xx_HAL_CONF_H */
00391
```

5.8 Core/Inc/stm32f1xx_it.h File Reference

This file contains the headers of the interrupt handlers.

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **MemManage_Handler** (void)

  This function handles Memory management fault.
- void **BusFault_Handler** (void)

  This function handles Prefetch fault, memory access fault.
- void **UsageFault_Handler** (void)

  This function handles Undefined instruction or illegal state.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **DebugMon_Handler** (void)

  This function handles Debug monitor.
- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.
- void **SysTick_Handler** (void)

  This function handles System tick timer.
- void **TIM2_IRQHandler** (void)

  This function handles TIM2 global interrupt.

5.8.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

5.9 stm32f1xx_it.h

Go to the documentation of this file.

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -------------------------------------*/
00021 #ifndef __STM32F1xx_IT_H
00022 #define __STM32F1xx_IT_H
00023
00024 #ifdef __cplusplus
00025  extern "C" {
00026 #endif
00027
00028 /* Private includes ----------------------------------------------------------*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types ------------------------------------------------------------*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants --------------------------------------------------------*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
00043 /* Exported macro ------------------------------------------------------------*/
00044 /* USER CODE BEGIN EM */
00045
00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes ---------------------------------------------*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void MemManage_Handler(void);
00052 void BusFault_Handler(void);
00053 void UsageFault_Handler(void);
00054 void SVC_Handler(void);
00055 void DebugMon_Handler(void);
00056 void PendSV_Handler(void);
00057 void SysTick_Handler(void);
00058 void TIM2_IRQHandler(void);
00059 /* USER CODE BEGIN EFP */
00060
00061 /* USER CODE END EFP */
00062
00063 #ifdef __cplusplus
00064 }
00065 #endif
00066
00067 #endif /* __STM32F1xx_IT_H */
```

5.10 Core/Src/actuator.c File Reference

Actuator C file for actuator controller project.

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

Functions

- void process_unknown (uint8_t endswitch_state)

  Function to process unknown(powerup) state of actuator.
- void process_calibration_home_backward (uint8_t endswitch_state)

  Function to home actuator backward after poweron.
- void process_calibration_home_forward (uint8_t endswitch_state)

Function to home actuator forward after poweron.

- void process_calibration_speed_forward (uint8_t endswitch_state)

  Function to calibrate forward speed.
- void process_calibration_speed_backward (uint8_t endswitch_state)

  Function to calibrate backward speed.
- void process_moving_middle (uint8_t endswitch_state)

  Function to center actuator between two endpoints.
- void **actuator_iteration** (void)

  Main function for processing actuator events.
- void move_actuator (uint8_t operation)

  Function to set actuator outputs.
- uint8_t get_end_switch_state (void)

  Get the end switch state object.

Variables

- uint8_t **errorblink** = 0
- uint32_t **operationTimestamp** = 0
- uint32_t **targetTimestamp** = 0
- uint32_t **timeForward** = 0
- uint32_t **timeBackward** = 0

5.10.1 Detailed Description

Actuator C file for actuator controller project.

Author

Fedor Zagumennov

Version

0.1

Date

2023-09-06

Copyright

Copyright (c) 2023

5.10.2 Function Documentation

5.10.2.1 get_end_switch_state()

```
uint8_t get_end_switch_state (
            void  )
```

Get the end switch state object.

Returns

uint8_t

5.10.2.2 move_actuator()

```
void move_actuator (
            uint8_t operation )
```

Function to set actuator outputs.

Parameters

operation	

5.10.2.3 process_calibration_home_backward()

```
void process_calibration_home_backward (
            uint8_t endswitch_state )
```

Function to home actuator backward after poweron.

Parameters

endswitch_state	

5.10.2.4 process_calibration_home_forward()

```
void process_calibration_home_forward (
            uint8_t endswitch_state )
```

Function to home actuator forward after poweron.

Parameters

endswitch_state	

5.10.2.5 process_calibration_speed_backward()

```
void process_calibration_speed_backward (
            uint8_t endswitch_state )
```

Function to calibrate backward speed.

Parameters

endswitch_state	

5.10.2.6 process_calibration_speed_forward()

```
void process_calibration_speed_forward (
            uint8_t endswitch_state )
```

Function to calibrate forward speed.

Parameters

endswitch_state	

5.10.2.7 process_moving_middle()

```
void process_moving_middle (
            uint8_t endswitch_state )
```

Function to center actuator between two endpoints.

Parameters

endswitch_state	

5.10.2.8 process_unknown()

```
void process_unknown (
            uint8_t endswitch_state )
```

Function to process unknown(powerup) state of actuator.

Parameters

endswitch_state	

5.11 Core/Src/main.c File Reference

: Main program body

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

Functions

- void SystemClock_Config (void)

  System Clock Configuration.

- int main (void)

  The application entry point.
- void **HAL_TIM_PeriodElapsedCallback** (TIM_HandleTypeDef ∗htim)
- void Error_Handler (void)

  This function is executed in case of error occurrence.

Variables

- TIM_HandleTypeDef **htim2**

5.11.1 Detailed Description

: Main program body

Attention

5.11.2 Function Documentation

5.11.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

5.11.2.2 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

5.11.2.3 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

System Clock Configuration.

Return values

None	

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

5.12 Core/Src/stm32f1xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

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

Functions

- void HAL_MspInit (void)
- void HAL_TIM_Base_MspInit (TIM_HandleTypeDef ∗htim_base)

  TIM_Base MSP Initialization This function configures the hardware resources used in this example.
- void HAL_TIM_Base_MspDeInit (TIM_HandleTypeDef ∗htim_base)

  TIM_Base MSP De-Initialization This function freeze the hardware resources used in this example.

5.12.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

5.12.2 Function Documentation

5.12.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void  )
```

Initializes the Global MSP. NOJTAG: JTAG-DP Disabled and SW-DP Enabled

5.12.2.2 HAL_TIM_Base_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
            TIM_HandleTypeDef * htim_base )
```

TIM_Base MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

htim_base	TIM_Base handle pointer

Return values

None	

5.12.2.3 HAL_TIM_Base_MspInit()

```
void HAL_TIM_Base_MspInit (
            TIM_HandleTypeDef * htim_base )
```

TIM_Base MSP Initialization This function configures the hardware resources used in this example.

Parameters

htim_base	TIM_Base handle pointer

Return values

None	

5.13 Core/Src/stm32f1xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f1xx_it.h"
```

Functions

- void **NMI_Handler** (void)

  This function handles Non maskable interrupt.
- void **HardFault_Handler** (void)

  This function handles Hard fault interrupt.
- void **MemManage_Handler** (void)

  This function handles Memory management fault.
- void **BusFault_Handler** (void)

  This function handles Prefetch fault, memory access fault.
- void **UsageFault_Handler** (void)

  This function handles Undefined instruction or illegal state.
- void **SVC_Handler** (void)

  This function handles System service call via SWI instruction.
- void **DebugMon_Handler** (void)

This function handles Debug monitor.

- void **PendSV_Handler** (void)

  This function handles Pendable request for system service.

- void **SysTick_Handler** (void)

  This function handles System tick timer.

- void **TIM2_IRQHandler** (void)

  This function handles TIM2 global interrupt.

Variables

- TIM_HandleTypeDef **htim2**

5.13.1 Detailed Description

Interrupt Service Routines.

Attention

5.14 Core/Src/system_stm32f1xx.c File Reference

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

```
#include "stm32f1xx.h"
```

Macros

- #define HSE_VALUE 8000000U
- #define HSI_VALUE 8000000U

Functions

- void SystemInit (void)

  Setup the microcontroller system Initialize the Embedded Flash Interface, the PLL and update the SystemCoreClock variable.

- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t **SystemCoreClock** = 16000000
- const uint8_t **AHBPrescTable** [16U] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t **APBPrescTable** [8U] = {0, 0, 0, 0, 1, 2, 3, 4}

5.14.1 Detailed Description

CMSIS Cortex-M3 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

1. This file provides two functions and one global variable to be called from user application:
   - SystemInit(): Setups the system clock (System clock source, PLL Multiplier factors, AHB/APBx prescalers and Flash settings). This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f1xx_xx.s" file.
   - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
   - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then SystemInit() function is called, in "startup_stm32f1xx_xx.s" file, to configure the system clock before to branch to main program.
3. The default value of HSE crystal is set to 8 MHz (or 25 MHz, depending on the product used), refer to "HSE↩ _VALUE". When HSE is used as system clock source, directly or through PLL, and you are using different crystal you have to adapt the HSE value to your own configuration.

Attention

Index