HAL-062 Operational Panel

Generated by Doxygen 1.9.2

Chapter 1

HAL-062 Operational Panel

This is the official HAL-062 operational panel documentation. It consists of description of the code variables and most important functions.

The panel constitutes of several switches, potentiometers, lights and 3 joysticks. The system uses communications standards like I2C, CAN, UART, and bluetooth and ethernet modules, which are supposed to engage communication with the rover.

Project consists of files specifying operation of each module, which are located in /Modules folder. Used communication standards and modules:

- I2C
- UART
- bluetooth
- ethernet - W7500S2E-R1

Used modules

- MAX11616EEE+ ADC converter
- MCP23017 GPIO expander

1.0.1 To read the description of files, please open "Files" bookmark

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 cameraSwitch Struct Reference

abcd to sczytywanie wartosci ze switcha

```
#include <camera_switch.h>
```

Public Attributes

- uint8_t **cameraNumber**
- GPIO_PinState **channel_A**
- GPIO_PinState **channel_B**
- GPIO_PinState **channel_C**
- GPIO_PinState **channel_D**

4.1.1 Detailed Description

abcd to sczytywanie wartosci ze switcha

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

- C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062_panel/Modules/camera_switch/camera_switch.h

4.2 currentLEDstate Struct Reference

Public Attributes

- uint8_t **dev1portA**
- uint8_t **dev1portB**
- uint8_t **dev2portA**
- uint8_t **dev2portB**

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

- C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062_panel/Modules/LED_switch/LED_const.h

4.3 Joystick Struct Reference

Public Attributes

- uint8_t number
- uint16_t xVal
- uint16_t yVal
- uint16_t zVal
- uint16_t midVal
- int16_t xPos
- int16_t yPos
- int16_t zPos

4.3.1 Member Data Documentation

4.3.1.1 midVal

```
uint16_t Joystick::midVal
```

Reference value used to set middle value based on raw value

4.3.1.2 number

```
uint8_t Joystick::number
```

Joystick number (1, 2 or 3)

4.3.1.3 xPos

```
int16_t Joystick::xPos
```

Calculated final x value, which is sent in frame

4.3.1.4 xVal

```
uint16_t Joystick::xVal
```

Read joystick raw value in x axis

4.3.1.5 yPos

```
int16_t Joystick::yPos
```

Calculated final y value, which is sent in frame

4.3.1.6 yVal

```
uint16_t Joystick::yVal
```

Read joystick raw value in y axis

4.3.1.7 zPos

```
int16_t Joystick::zPos
```

Calculated final z value, which is sent in frame

4.3.1.8 zVal

```
uint16_t Joystick::zVal
```

Read joystick raw value in z axis

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

- C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062_panel/Modules/joystick/joystick_const.h

Chapter 5

File Documentation

5.1 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062↩ _panel/Modules/buttons/buttons.c File Reference

Buttons functionality.

```
#include "buttons.h"
#include "buttons_const.h"
#include "LED_switch/LED_switch.h"
#include "LED_switch/LED_const.h"
#include <stm32h7xx_hal_gpio.h>
#include <stdbool.h>
```

Functions

- void **Buttons_Init** (void)

  Initializes GPIO ports and inputs for buttons.
- void Set_LED_For_Bistable (void)

  Sets LED lights state depending on bistable button.
- void **ADC1_Init** (void)

  Initializes ADC module.
- void **ADC_Try_Read** (void)

  Reads value from potentiometer.
- void **EXTI15_10_IRQHandler** (void)
- void **EXTI4_IRQHandler** (void)
- void **EXTI9_5_IRQHandler** (void)

  This function handles EXTI line[9:5] interrupts.
- void **HAL_GPIO_EXTI_Callback** (uint16_t GPIO_Pin)

Variables

- static uint8_t **buttonsState** = 0x00
- I2C_HandleTypeDef **hi2c1**
- ADC_HandleTypeDef **hadc1**
- double val = 0.0625

5.1.1 Detailed Description

Buttons functionality.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.1.2 Function Documentation

5.1.2.1 Set_LED_For_Bistable()

```
void Set_LED_For_Bistable (
             void  )
```

Sets LED lights state depending on bistable button.

This functionality is used to indicate on the panel, whether bistable button is pressed. The function uses bit masks to read button state and sets proper I2C signal (according to I2C module documentation) which at the end is being written by the interrupt.

5.1.3 Variable Documentation

5.1.3.1 val

```
double val = 0.0625
```

Upper value for joystick values divider raw value

5.2 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062↵ _panel/Modules/buttons/buttons.h File Reference

Buttons header file.

Functions

- void **Buttons_Init** (void)

  Initializes GPIO ports and inputs for buttons.
- void Set_LED_For_Bistable (void)

  Sets LED lights state depending on bistable button.
- void **ADC1_Init** (void)

  Initializes ADC module.
- void **ADC_Try_Read** (void)

  Reads value from potentiometer.

5.2.1 Detailed Description

Buttons header file.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.2.2 Function Documentation

5.2.2.1 Set_LED_For_Bistable()

```
void Set_LED_For_Bistable (
             void  )
```

Sets LED lights state depending on bistable button.

This functionality is used to indicate on the panel, whether bistable button is pressed. The function uses bit masks to read button state and sets proper I2C signal (according to I2C module documentation) which at the end is being written by the interrupt.

5.3 buttons.h

Go to the documentation of this file.

```
#ifndef BUTTONS_BUTTONS_H_
#define BUTTONS_BUTTONS_H_


void Buttons_Init(void);

void Set_LED_For_Bistable(void);


void ADC1_Init(void);

void ADC_Try_Read(void);

#endif //BUTTON_BUTTON_H_
```

5.4 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062↩_panel/Modules/buttons/buttons_const.h File Reference

This a file containing macro definitions for GPIO inputs and outputs.

```
#include <stm32h7xx_hal.h>
#include <stm32h7xx_hal_gpio.h>
```

Macros

- #define **MONO_BUTTON_JOY_RED_Pin** GPIO_PIN_4
- #define **MONO_BUTTON_JOY_RED_GPIO_Port** GPIOD
- #define **MONO_BUTTON_JOY_RED_EXTI_IRQn** EXTI4_IRQn
- #define **MONO_BUTTON_JOY_BLUE_Pin** GPIO_PIN_5
- #define **MONO_BUTTON_JOY_BLUE_GPIO_Port** GPIOD
- #define **MONO_BUTTON_JOY_BLUE_EXTI_IRQn** EXTI9_5_IRQn
- #define **MONO_BUTTON_JOY_GREEN_Pin** GPIO_PIN_6
- #define **MONO_BUTTON_JOY_GREEN_GPIO_Port** GPIOD
- #define **MONO_BUTTON_JOY_GREEN_EXTI_IRQn** EXTI9_5_IRQn
- #define **MONO_BUTTON_RED_1_Pin** GPIO_PIN_7
- #define **MONO_BUTTON_RED_1_GPIO_Port** GPIOD
- #define **MONO_BUTTON_RED_1_EXTI_IRQn** EXTI9_5_IRQn
- #define **MONO_BUTTON_BLACK_1_Pin** GPIO_PIN_9
- #define **MONO_BUTTON_BLACK_1_GPIO_Port** GPIOG
- #define **MONO_BUTTON_BLACK_1_EXTI_IRQn** EXTI9_5_IRQn
- #define **MONO_BUTTON_GREEN_1_Pin** GPIO_PIN_10
- #define **MONO_BUTTON_GREEN_1_GPIO_Port** GPIOG
- #define **MONO_BUTTON_GREEN_1_EXTI_IRQn** EXTI15_10_IRQn
- #define **MONO_BUTTON_BLUE_1_Pin** GPIO_PIN_11
- #define **MONO_BUTTON_BLUE_1_GPIO_Port** GPIOG
- #define **MONO_BUTTON_BLUE_1_EXTI_IRQn** EXTI15_10_IRQn
- #define **MONO_BUTTON_RED_2_Pin** GPIO_PIN_12
- #define **MONO_BUTTON_RED_2_GPIO_Port** GPIOG
- #define **MONO_BUTTON_RED_2_EXTI_IRQn** EXTI15_10_IRQn
- #define **MONO_BUTTON_BLACK_2_Pin** GPIO_PIN_13
- #define **MONO_BUTTON_BLACK_2_GPIO_Port** GPIOG
- #define **MONO_BUTTON_BLACK_2_EXTI_IRQn** EXTI15_10_IRQn
- #define **MONO_BUTTON_GREEN_2_Pin** GPIO_PIN_14
- #define **MONO_BUTTON_GREEN_2_GPIO_Port** GPIOG
- #define **MONO_BUTTON_GREEN_2_EXTI_IRQn** EXTI15_10_IRQn
- #define **MONO_BUTTON_BLUE_2_Pin** GPIO_PIN_15
- #define **MONO_BUTTON_BLUE_2_GPIO_Port** GPIOG
- #define **MONO_BUTTON_BLUE_2_EXTI_IRQn** EXTI15_10_IRQn
- #define **BI_BUTTON_RED_1_Pin** GPIO_PIN_3
- #define **BI_BUTTON_RED_1_GPIO_Port** GPIOB
- #define **BI_BUTTON_RED_2_Pin** GPIO_PIN_4
- #define **BI_BUTTON_RED_2_GPIO_Port** GPIOB
- #define **BI_BUTTON_BLUE_1_Pin** GPIO_PIN_5
- #define **BI_BUTTON_BLUE_1_GPIO_Port** GPIOB
- #define **BI_BUTTON_BLUE_2_Pin** GPIO_PIN_6
- #define **BI_BUTTON_BLUE_2_GPIO_Port** GPIOB
- #define **BI_BUTTON_BLUE_3_Pin** GPIO_PIN_7
- #define **BI_BUTTON_BLUE_3_GPIO_Port** GPIOB
- #define **BI_BUTTON_GREEN_1_Pin** GPIO_PIN_0
- #define **BI_BUTTON_GREEN_1_GPIO_Port** GPIOE
- #define **BI_BUTTON_GREEN_2_Pin** GPIO_PIN_1
- #define **BI_BUTTON_GREEN_2_GPIO_Port** GPIOE
- #define **BI_BUTTON_GREEN_3_Pin** GPIO_PIN_2
- #define **BI_BUTTON_GREEN_3_GPIO_Port** GPIOE
- #define **CAM_SWITCH_3_A_Pin** GPIO_PIN_15
- #define **CAM_SWITCH_3_A_GPIO_Port** GPIOB
- #define **CAM_SWITCH_3_B_Pin** GPIO_PIN_14
- #define **CAM_SWITCH_3_B_GPIO_Port** GPIOD

- #define **CAM_SWITCH_3_C_Pin** GPIO_PIN_15
- #define **CAM_SWITCH_3_C_GPIO_Port** GPIOD
- #define **CAM_SWITCH_3_D_Pin** GPIO_PIN_2
- #define **CAM_SWITCH_3_D_GPIO_Port** GPIOG
- #define **CAM_SWITCH_2_A_Pin** GPIO_PIN_3
- #define **CAM_SWITCH_2_A_GPIO_Port** GPIOG
- #define **CAM_SWITCH_2_B_Pin** GPIO_PIN_4
- #define **CAM_SWITCH_2_B_GPIO_Port** GPIOG
- #define **CAM_SWITCH_2_C_Pin** GPIO_PIN_5
- #define **CAM_SWITCH_2_C_GPIO_Port** GPIOG
- #define **CAM_SWITCH_2_D_Pin** GPIO_PIN_6
- #define **CAM_SWITCH_2_D_GPIO_Port** GPIOG
- #define **CAM_SWITCH_1_A_Pin** GPIO_PIN_7
- #define **CAM_SWITCH_1_A_GPIO_Port** GPIOG
- #define **CAM_SWITCH_1_B_Pin** GPIO_PIN_8
- #define **CAM_SWITCH_1_B_GPIO_Port** GPIOG
- #define **CAM_SWITCH_1_C_Pin** GPIO_PIN_8
- #define **CAM_SWITCH_1_C_GPIO_Port** GPIOC
- #define **CAM_SWITCH_1_D_Pin** GPIO_PIN_9
- #define **CAM_SWITCH_1_D_GPIO_Port** GPIOC

5.4.1 Detailed Description

This a file containing macro definitions for GPIO inputs and outputs.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

This file consists of macro definitions for all modules using GPIO pin, which are buttons, lights and camera switches, each specyfing the GPIO port and GPIO pin number This is used for easier decoding, better readability of the code.

5.5 buttons_const.h

Go to the documentation of this file.

```
#ifndef BUTTONS_BUTTONS_CONST_H_
#define BUTTONS_BUTTONS_CONST_H_

/* Includes ------------------------------------------------------------ */
#include <stm32h7xx_hal.h>
#include <stm32h7xx_hal_gpio.h>

#define MONO_BUTTON_JOY_RED_Pin GPIO_PIN_4 // IN9
#define MONO_BUTTON_JOY_RED_GPIO_Port GPIOD
#define MONO_BUTTON_JOY_RED_EXTI_IRQn EXTI4_IRQn
#define MONO_BUTTON_JOY_BLUE_Pin GPIO_PIN_5 // IN 10
#define MONO_BUTTON_JOY_BLUE_GPIO_Port GPIOD
#define MONO_BUTTON_JOY_BLUE_EXTI_IRQn EXTI9_5_IRQn
#define MONO_BUTTON_JOY_GREEN_Pin GPIO_PIN_6 // IN 11
#define MONO_BUTTON_JOY_GREEN_GPIO_Port GPIOD
#define MONO_BUTTON_JOY_GREEN_EXTI_IRQn EXTI9_5_IRQn
#define MONO_BUTTON_RED_1_Pin GPIO_PIN_7 // IN 12
#define MONO_BUTTON_RED_1_GPIO_Port GPIOD
#define MONO_BUTTON_RED_1_EXTI_IRQn EXTI9_5_IRQn
#define MONO_BUTTON_BLACK_1_Pin GPIO_PIN_9 // IN 13
#define MONO_BUTTON_BLACK_1_GPIO_Port GPIOG
#define MONO_BUTTON_BLACK_1_EXTI_IRQn EXTI9_5_IRQn
#define MONO_BUTTON_GREEN_1_Pin GPIO_PIN_10 // IN 14
```

```
#define MONO_BUTTON_GREEN_1_GPIO_Port GPIOG
#define MONO_BUTTON_GREEN_1_EXTI_IRQn EXTI15_10_IRQn
#define MONO_BUTTON_BLUE_1_Pin GPIO_PIN_11 // IN 15
#define MONO_BUTTON_BLUE_1_GPIO_Port GPIOG
#define MONO_BUTTON_BLUE_1_EXTI_IRQn EXTI15_10_IRQn
#define MONO_BUTTON_RED_2_Pin GPIO_PIN_12 // IN 16
#define MONO_BUTTON_RED_2_GPIO_Port GPIOG
#define MONO_BUTTON_RED_2_EXTI_IRQn EXTI15_10_IRQn
#define MONO_BUTTON_BLACK_2_Pin GPIO_PIN_13 // IN 17
#define MONO_BUTTON_BLACK_2_GPIO_Port GPIOG
#define MONO_BUTTON_BLACK_2_EXTI_IRQn EXTI15_10_IRQn
#define MONO_BUTTON_GREEN_2_Pin GPIO_PIN_14 // IN 18
#define MONO_BUTTON_GREEN_2_GPIO_Port GPIOG
#define MONO_BUTTON_GREEN_2_EXTI_IRQn EXTI15_10_IRQn
#define MONO_BUTTON_BLUE_2_Pin GPIO_PIN_15 // IN 19
#define MONO_BUTTON_BLUE_2_GPIO_Port GPIOG
#define MONO_BUTTON_BLUE_2_EXTI_IRQn EXTI15_10_IRQn

#define BI_BUTTON_RED_1_Pin GPIO_PIN_3 // IN 20
#define BI_BUTTON_RED_1_GPIO_Port GPIOB
#define BI_BUTTON_RED_2_Pin GPIO_PIN_4 // IN 21
#define BI_BUTTON_RED_2_GPIO_Port GPIOB
#define BI_BUTTON_BLUE_1_Pin GPIO_PIN_5 // IN 22
#define BI_BUTTON_BLUE_1_GPIO_Port GPIOB
#define BI_BUTTON_BLUE_2_Pin GPIO_PIN_6 // IN 23
#define BI_BUTTON_BLUE_2_GPIO_Port GPIOB
#define BI_BUTTON_BLUE_3_Pin GPIO_PIN_7 // IN 24
#define BI_BUTTON_BLUE_3_GPIO_Port GPIOB
#define BI_BUTTON_GREEN_1_Pin GPIO_PIN_0 // IN 25
#define BI_BUTTON_GREEN_1_GPIO_Port GPIOE
#define BI_BUTTON_GREEN_2_Pin GPIO_PIN_1 // IN 26
#define BI_BUTTON_GREEN_2_GPIO_Port GPIOE
#define BI_BUTTON_GREEN_3_Pin GPIO_PIN_2 // IN 27
#define BI_BUTTON_GREEN_3_GPIO_Port GPIOE

#define CAM_SWITCH_3_A_Pin GPIO_PIN_15
#define CAM_SWITCH_3_A_GPIO_Port GPIOB
#define CAM_SWITCH_3_B_Pin GPIO_PIN_14
#define CAM_SWITCH_3_B_GPIO_Port GPIOD
#define CAM_SWITCH_3_C_Pin GPIO_PIN_15
#define CAM_SWITCH_3_C_GPIO_Port GPIOD
#define CAM_SWITCH_3_D_Pin GPIO_PIN_2
#define CAM_SWITCH_3_D_GPIO_Port GPIOG
#define CAM_SWITCH_2_A_Pin GPIO_PIN_3
#define CAM_SWITCH_2_A_GPIO_Port GPIOG
#define CAM_SWITCH_2_B_Pin GPIO_PIN_4
#define CAM_SWITCH_2_B_GPIO_Port GPIOG
#define CAM_SWITCH_2_C_Pin GPIO_PIN_5
#define CAM_SWITCH_2_C_GPIO_Port GPIOG
#define CAM_SWITCH_2_D_Pin GPIO_PIN_6
#define CAM_SWITCH_2_D_GPIO_Port GPIOG
#define CAM_SWITCH_1_A_Pin GPIO_PIN_7
#define CAM_SWITCH_1_A_GPIO_Port GPIOG
#define CAM_SWITCH_1_B_Pin GPIO_PIN_8
#define CAM_SWITCH_1_B_GPIO_Port GPIOG
#define CAM_SWITCH_1_C_Pin GPIO_PIN_8
#define CAM_SWITCH_1_C_GPIO_Port GPIOC
#define CAM_SWITCH_1_D_Pin GPIO_PIN_9
#define CAM_SWITCH_1_D_GPIO_Port GPIOC



#endif //BUTTONS_BUTTONS_CONST_H_
```

5.6 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062↩_panel/Modules/camera_switch/camera_switch.c File Reference

Functionality for camera switch.

```
#include "camera_switch.h"
#include "buttons/buttons_const.h"
#include "error_handlers/error_handlers.h"
#include "ethernet/ethernet.h"
#include "LED_switch/LED_switch.h"
#include "LED_switch/LED_const.h"
#include <stdbool.h>
#include <stm32h7xx_hal_gpio.h>
```

Functions

- HAL_StatusTypeDef **Check_Camera_State** (struct cameraSwitch camSw)
- void **Read_Camera_Switch_Value** (void)
- void **Send_Cameras_State** (void)

  Send_Cameras_State() creates a frame that sends 48 stands for 0 in ASCII code 78 stands for x in ASCII.
- void **Set_Camera_LED** (void)

Variables

- static struct cameraSwitch **yellowCamera** = {1,0,0,0,0}
- static struct cameraSwitch **blueCamera** = {2,0,0,0,0}
- static struct cameraSwitch **redCamera** = {3,0,0,0,0}
- bool **ethTxLineOpen**
- static uint8_t **cameraMsgID** [2] = {0x53,0x53}
- static uint8_t **cameraMsgData** [16]
- uint8_t **currentCameraLight** = 0

5.6.1 Detailed Description

Functionality for camera switch.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

This

5.7 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062↩ _panel/Modules/camera_switch/camera_switch.h File Reference

Buttons functionality.

```
#include <stm32h7xx_hal.h>
```

Classes

- struct cameraSwitch

  abcd to sczytywanie wartosci ze switcha

Functions

- HAL_StatusTypeDef **Check_Camera_State** (struct cameraSwitch camSw)
- void **Read_Camera_Switch_Value** (void)
- void **Send_Cameras_State** (void)

  Send_Cameras_State() creates a frame that sends 48 stands for 0 in ASCII code 78 stands for x in ASCII.
- void **Set_Camera_LED** (void)

5.7.1 Detailed Description

Buttons functionality.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.8 camera_switch.h

Go to the documentation of this file.

```
#ifndef CAMERA_SWITCH_CAMERA_SWITCH_H_
#define CAMERA_SWITCH_CAMERA_SWITCH_H_

#include <stm32h7xx_hal.h>


struct cameraSwitch{
    uint8_t cameraNumber;
    GPIO_PinState channel_A;
    GPIO_PinState channel_B;
    GPIO_PinState channel_C;
    GPIO_PinState channel_D;
};

HAL_StatusTypeDef Check_Camera_State(struct cameraSwitch camSw);

void Read_Camera_Switch_Value(void);

void Send_Cameras_State(void);

void Set_Camera_LED(void);

#endif //CAMERA_SWITCH_CAMERA_SWITCH_H_
```

5.9 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/HAL062↩_panel/Modules/error_handlers/error_handlers.c File Reference

: Error handlers - functionality

```
#include <stm32h7xx_hal.h>
#include "error_handlers/error_handlers.h"
#include "LED_switch/LED_switch.h"
```

Functions

- void **Error_Handler** (enum errorCode code)
- void **Error_Handler** (void)

5.9.1 Detailed Description

: Error handlers - functionality

Author

: K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.10 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/error_handlers/error_handlers.h File Reference

Error handlers - headers file.

Enumerations

- enum errorCode { ReceivedFrameError = 0 , TransmittedFrameError = 1 , ConnectionLostError = 2 , CriticalSystemError = 3 }

Functions

- void **Error_Handler** (void)

5.10.1 Detailed Description

Error handlers - headers file.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.10.2 Enumeration Type Documentation

5.10.2.1 errorCode

`enum errorCode`

Enumerator

ReceivedFrameError	Raises error of unproperly constructed frame
TransmittedFrameError	Raises error of unproperly construced frame to send
ConnectionLostError	Raises error of lost connection
CriticalSystemError	Raises error of any other system malfunction

5.11 error_handlers.h

Go to the documentation of this file.

```
#ifndef ERROR_HANDLERS_ERROR_HANDLERS_H_
```

```
#define ERROR_HANDLERS_ERROR_HANDLERS_H_

enum errorCode{
    ReceivedFrameError = 0,
    TransmittedFrameError = 1,
    ConnectionLostError = 2,
    CriticalSystemError = 3
};

void Error_Handler(void);

#endif /* ERROR_HANDLERS_ERROR_HANDLERS_H_ */
```

5.12 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/ethernet/ethernet.c File Reference

Ethernet - functionality.

```
#include "ethernet.h"
#include "error_handlers/error_handlers.h"
#include <stdbool.h>
#include "LED_switch/LED_const.h"
#include "LED_switch/LED_switch.h"
```

Functions

- void Eth_Init ()
  Initializing ethernet module: GPIO, UART.
- void Eth_Send_Massage (uint8_t ∗frameID, uint8_t ∗msgData)
  sending message using special frame
- void Eth_Receive_Massage ()
  Begins to listening of data.
- void **HAL_UART_RxCpltCallback** (UART_HandleTypeDef ∗huart)
- void **HAL_UART_TxCpltCallback** (UART_HandleTypeDef ∗huart)

Variables

- UART_HandleTypeDef **huart3**
- DMA_HandleTypeDef **hdma_usart3_rx**
- DMA_HandleTypeDef **hdma_usart3_tx**
- static uint8_t **ethTxBuffer** [19]
- static uint8_t **ethRxBuffer**
- volatile uint8_t **boudryButtonStates** [3]
- bool **ethTxLineOpen** = true
- bool **test** = false
- uint8_t **taken** = 0
- bool **foundHash**
- bool **toSend** = false

5.12.1 Detailed Description

Ethernet - functionality.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.12.2 Function Documentation

5.12.2.1 Eth_Init()

```
void Eth_Init ( )
```

Initializing ethernet module: GPIO, UART.

Parameters

None	

Returns

void

5.12.2.2 Eth_Receive_Massage()

```
void Eth_Receive_Massage ( )
```

Begins to listening of data.

Returns

void

5.12.2.3 Eth_Send_Massage()

```
void Eth_Send_Massage (
            uint8_t * frameID,
            uint8_t * msgData )
```

sending message using special frame

Parameters

frameID	array of uint8_t, specifying rover modules
msgData	array of uint8_t, contains data connected to ID

Returns

void

5.13 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/ethernet/ethernet.h File Reference

Ethernet - headers file.

```
#include <stm32h7xx_hal.h>
```

Functions

- void Eth_Init ()

  Initializing ethernet module: GPIO, UART.
- void Eth_Send_Massage (uint8_t ∗frameID, uint8_t ∗msgData)

  sending message using special frame
- void Eth_Receive_Massage ()

  Begins to listening of data.

5.13.1 Detailed Description

Ethernet - headers file.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.13.2 Function Documentation

5.13.2.1 Eth_Init()

```
void Eth_Init ( )
```

Initializing ethernet module: GPIO, UART.

Parameters

None	

Returns

void

5.13.2.2 Eth_Receive_Massage()

```
void Eth_Receive_Massage ( )
```

Begins to listening of data.

Returns

void

5.13.2.3 Eth_Send_Massage()

```
void Eth_Send_Massage (
             uint8_t * frameID,
             uint8_t * msgData )
```

sending message using special frame

Parameters

frameID	array of uint8_t, specifying rover modules
msgData	array of uint8_t, contains data connected to ID

Returns

void

5.14 ethernet.h

Go to the documentation of this file.

```
#ifndef ETHERNET_ETHERNET_H_
#define ETHERNET_ETHERNET_H_

#include <stm32h7xx_hal.h>

void Eth_Init();


void Eth_Send_Massage(uint8_t *frameID, uint8_t *msgData);


void Eth_Receive_Massage();


#endif /* ETHERNET_ETHERNET_H_ */
```

5.15 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/joystick/joystick.c File Reference

: Joystick module - implementing functionality and initialization

```
#include "joystick.h"
#include "error_handlers/error_handlers.h"
#include "joystick_const.h"
#include "ethernet/ethernet.h"
#include <stdbool.h>
#include "buttons/buttons.h"
#include "buttons/buttons_const.h"
```

Functions

- void **Joystick_I2C_Init** (void)
- void **Joystick_Write_Conditions** (void)
- void **Joystick_Read_Value_Start** (void)
- void **Jostick_Read_value_Done** (void)
- void **HAL_I2C_MasterTxCpltCallback** (I2C_HandleTypeDef *hi2c)
- void **HAL_I2C_MasterRxCpltCallback** (I2C_HandleTypeDef *hi2c)
- void **Joystick_Send_Readings** (void)

Variables

- I2C_HandleTypeDef hi2c2
- bool **receiveIsReady** = false
- bool joyInitFinished = false
- static uint8_t **receiveData** [24]
- struct Joystick **motorJoy**
- struct Joystick **manipJoy**
- struct Joystick **gripperJoy**
- double val
- static uint8_t **currentReading** = 0
- bool **ethTxLineOpen**
- static uint8_t **data**

5.15.1 Detailed Description

: Joystick module - implementing functionality and initialization

Author

: K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.15.2 Variable Documentation

5.15.2.1 hi2c2

```
I2C_HandleTypeDef hi2c2
```

I2C handler

5.15.2.2 joyInitFinished

```
bool joyInitFinished = false
```

Flag used to complete I2C initialization for joysticks

5.15.2.3 val

```
double val  [extern]
```

Upper value for joystick values divider raw value

5.16 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/joystick/joystick.h File Reference

: Joystick module - header file

```
#include <stm32h7xx_hal.h>
```

Functions

- void **Joystick_I2C_Init** (void)
- void **Joystick_Write_Conditions** (void)
- void **Joystick_Read_Value_Start** (void)
- void **Joystick_Send_Readings** (void)
- void **Jostick_Read_value_Done** (void)

5.16.1 Detailed Description

: Joystick module - header file

: Ethernet - headers file

Author

: K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.17 joystick.h

Go to the documentation of this file.

```
#ifndef JOYSTICK_JOYSTICK_H_
#define JOYSTICK_JOYSTICK_H_

#include <stm32h7xx_hal.h>

// @brief Initializing I2C module:  GPIO
void Joystick_I2C_Init(void);

// @brief configuring ADC inverter
void Joystick_Write_Conditions(void);

// @brief setting flag - start receive joysticks value
void Joystick_Read_Value_Start(void);

// @brief sending do rover joystick values
void Joystick_Send_Readings(void);

// @brief reading value done
void Jostick_Read_value_Done(void);

#endif // JOYSTICK_JOYSTICK_H

```

5.18 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/joystick/joystick_const.h File Reference

Joystick variables structure and ADC converter configuration variables.

`#include <stm32h7xx_hal.h>`

Classes

- struct Joystick

Macros

- #define SLAVE_ADDRESS 0x35
- #define CONFIG_DATA 0x17
- #define SETUP_DATA 0x82

5.18.1 Detailed Description

Joystick variables structure and ADC converter configuration variables.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

See also

MAX.... module configuration in datasheet

5.18.2 Macro Definition Documentation

5.18.2.1 CONFIG_DATA

```
#define CONFIG_DATA 0x17
```

Configuration data for ADC converter

5.18.2.2 SETUP_DATA

```
#define SETUP_DATA 0x82
```

Setup data for ADC converter

5.18.2.3 SLAVE_ADDRESS

```
#define SLAVE_ADDRESS 0x35
```

I2C slave address for ADC converter

5.19 joystick_const.h

Go to the documentation of this file.

```

#ifndef JOYSTICK_JOYSTICK_CONST_H
#define JOYSTICK_JOYSTICK_CONST_H

#include <stm32h7xx_hal.h>

#define SLAVE_ADDRESS 0x35
#define CONFIG_DATA 0x17
#define SETUP_DATA 0x82
struct Joystick{
    uint8_t number;
    uint16_t xVal;
    uint16_t yVal;
    uint16_t zVal;
    uint16_t midVal;
    int16_t xPos;
    int16_t yPos;
    int16_t zPos;
};

#endif // JOYSTICK_JOYSTICK_CONST_H
```

5.20 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/LED_switch/LED_const.h File Reference

: Switch do LEDs - important constant

Classes

- struct currentLEDstate

Macros

- #define **LIGHT1** 0x401201
- #define **LIGHT2** 0x401202
- #define **LIGHT3** 0x401204
- #define **LIGHT4** 0x401208
- #define **LIGHT5** 0x401210
- #define **LIGHT6** 0x401220
- #define **LIGHT7** 0x401240
- #define **LIGHT8** 0x401280
- #define **LIGHT9** 0x401301
- #define **LIGHT10** 0x401302
- #define **LIGHT11** 0x401304
- #define **LIGHT12** 0x401308
- #define **LIGHT13** 0x401310
- #define **LIGHT14** 0x401320
- #define **LIGHT15** 0x401340
- #define **LIGHT16** 0x401380
- #define **LIGHT17** 0x421201
- #define **LIGHT18** 0x421202
- #define **LIGHT19** 0x421204
- #define **LIGHT20** 0x421208
- #define **LIGHT21** 0x421210
- #define **LIGHT22** 0x421220
- #define **LIGHT23** 0x421240
- #define **LIGHT24** 0x421280
- #define **LIGHT25** 0x401301
- #define **LIGHT26** 0x401302
- #define **LIGHT27** 0x401304
- #define **LIGHT28** 0x401308
- #define **LIGHT29** 0x401310
- #define **LIGHT30** 0x401320
- #define **LIGHT31** 0x401340
- #define **LIGHT32** 0x401380
- #define **DEV_1** 0x40
- #define **DEV_2** 0x42
- #define **PORT_A** 0x12
- #define **PORT_B** 0x13

5.20.1 Detailed Description

: Switch do LEDs - important constant

Author

: K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.21 LED_const.h

Go to the documentation of this file.

```
#ifndef LED_CONST_LED_CONST_H_
#define LED_CONST_LED_CONST_H_


//dev 0x40 , port A, pins 0-7
#define LIGHT1  0x401201
#define LIGHT2  0x401202
#define LIGHT3  0x401204
#define LIGHT4  0x401208
#define LIGHT5  0x401210
#define LIGHT6  0x401220
#define LIGHT7  0x401240
#define LIGHT8  0x401280


//dev 0x40 , port B, pins 0-7
#define LIGHT9  0x401301
#define LIGHT10 0x401302
#define LIGHT11 0x401304
#define LIGHT12 0x401308
#define LIGHT13 0x401310
#define LIGHT14 0x401320
#define LIGHT15 0x401340
#define LIGHT16 0x401380


//dev 0x42 , port A, pins 0-7
#define LIGHT17 0x421201
#define LIGHT18 0x421202
#define LIGHT19 0x421204
#define LIGHT20 0x421208
#define LIGHT21 0x421210
#define LIGHT22 0x421220
#define LIGHT23 0x421240
#define LIGHT24 0x421280


//dev 0x42 , port B, pins 0-7
#define LIGHT25 0x401301
#define LIGHT26 0x401302
#define LIGHT27 0x401304
#define LIGHT28 0x401308
#define LIGHT29 0x401310
#define LIGHT30 0x401320
#define LIGHT31 0x401340
#define LIGHT32 0x401380

// defining devices and ports
#define DEV_1   0x40
#define DEV_2   0x42
#define PORT_A  0x12
#define PORT_B  0x13


//structure to remember state of LED
struct currentLEDstate{
    uint8_t dev1portA;
    uint8_t dev1portB;
    uint8_t dev2portA;
    uint8_t dev2portB;
};

#endif //   LED_CONST_LED_CONST_H_
```

5.22 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/←HAL062_panel/Modules/LED_switch/LED_switch.c File Reference

: Switch do LEDs - initialization, function to set/reset LED

```
#include "LED_switch.h"
#include "error_handlers/error_handlers.h"
#include "LED_const.h"
#include <stdbool.h>
```

Functions

- void **LED_Init** (void)
- void **LED_Set** (uint32_t lightCode, uint8_t state)

Variables

- I2C_HandleTypeDef **hi2c1**
- static struct currentLEDstate **currentState**
- uint8_t **pinNum** = 0x00
- uint8_t **devAddr**
- uint8_t **memAddr**
- uint8_t **boundryLed** = 0
- bool **i2cLedLineOpen** = false

5.22.1 Detailed Description

: Switch do LEDs - initialization, function to set/reset LED

Author

: K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.23 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Modules/LED_switch/LED_switch.h File Reference

: Switch do LEDs - headers file

```
#include <stm32h7xx_hal.h>
```

Functions

- void **LED_Init** (void)
- void **LED_Set** (uint32_t lightCode, uint8_t state)
- void **Set_LED_For_Manip_Bounds** (uint8_t ∗states)

5.23.1 Detailed Description

: Switch do LEDs - headers file

Author

: K. Czechowicz, A. Rybojad, S. Kołodziejczyk

5.24 LED_switch.h

Go to the documentation of this file.

```
#ifndef LED_SWITCH_LED_SWITCH_H_
#define LED_SWITCH_LED_SWITCH_H_

#include <stm32h7xx_hal.h>


// @brief Initializing I2C module:  GPIO
// @param None
// @returns void
void LED_Init(void);

// @brief Setting/Reseting LED
// @param lightCode, predefined values "LIGHTn"
// @param state, 1 - set LED, other - reset LED
// @returns void
void LED_Set(uint32_t lightCode, uint8_t state);

void Set_LED_For_Manip_Bounds(uint8_t* states);

#endif //LED_SWITCH_LED_SWITCH_H
```

5.25 buttons_timer.h

```
#ifndef BUTTONS_BUTTONS_TIMER_H_
#define BUTTONS_BUTTONS_TIMER_H_

#include <stm32h7xx_hal.h>

void Buttons_Timer_Init(void);

#endif //BUTTON_BUTTON_TIMER_H_
```

5.26 joystick_timer.h

```
#ifndef JOYSTICK_JOYSTICK_TIMER_H
#define JOYSTICK_JOYSTICK_TIMER_H

#include <stm32h7xx_hal.h>


// @brief Initializing joystick module timer
// @param None
// @returns void
void Joystick_Timer_Init(void);


#endif // JOYSTICK_JOYSTICK_TIMER_H


```

5.27 watchdog.h

```
#ifndef WATCHDOGS_WATCHDOG_H_
#define WATCHDOGS_WATCHDOG_H_

#include <stm32h7xx_hal_iwdg.h>

void MX_IWDG1_Init(void);

#endif //WATCHDOGS_WATCHDOG_H_
```

5.28 C:/Users/ddd/Desktop/driver/HAL-062/SUB-HAL-062-panel/↩ HAL062_panel/Src/main.c File Reference

Rover's operational body main program body version 1.0.

```
#include <stm32h7xx_hal.h>
#include <stm32h7xx_hal_gpio.h>
#include <stdbool.h>
#include "error_handlers/error_handlers.h"
#include "ethernet/ethernet.h"
#include "LED_switch/LED_switch.h"
#include "LED_switch/LED_const.h"
#include "joystick/joystick.h"
#include "timers/joystick_timer.h"
#include "buttons/buttons.h"
#include "timers/buttons_timer.h"
#include "watchdogs/watchdog.h"
```

Functions

- void SystemClock_Config (void)

  SystemClock_Config() is the is used to initialize all required clocks for all modules.
- int **main** (void)

  main() is the main program function that is used only to initialize all required modules and to refresh watchdog

Variables

- bool joyInitFinished
- IWDG_HandleTypeDef hiwdg1

5.28.1 Detailed Description

Rover's operational body main program body version 1.0.

Author

K. Czechowicz, A. Rybojad, S. Kołodziejczyk

This is the main program body of HAL-062 rover's operational panel. The panel constitutes of several switches, potentiometers, lights and 3 joysticks. The system uses communications standards like I2C, CAN, UART, and bluetooth and ethernet modules, which are supposed to engage communication with the rover.

Project consists of files specifying operation of each module, which are located in /Modules folder.

Used communication standards and modules:

- I2C
- UART
- bluetooth
- ethernet - W7500S2E-R1

Used modules

- MAX11616EEE+ GPIO expander
- MCP23017 GPIO expander

5.28.2 Function Documentation

5.28.2.1 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

SystemClock_Config() is the is used to initialize all required clocks for all modules.

Supply configuration update enable

Configure the main internal regulator output voltage

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

Initializes the CPU, AHB and APB buses clocks

5.28.3 Variable Documentation

5.28.3.1 hiwdg1

```
IWDG_HandleTypeDef hiwdg1  [extern]
```

Watchdog handler used for initialization in watchdog.c

5.28.3.2 joyInitFinished

```
bool joyInitFinished  [extern]
```

Flag used to complete I2C initialization for joysticks