MyLibrary

AUTHOR
Version
Tue Jan 25 2022

Table of Contents

Table of contents

Module Index

Modules

Here is a list of all modules:

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Module Documentation

MyLibrary

Modules

- **CLIPBOARDS**
  CLIPBOARD Module Drivers.
- **SERIALPROTOCOL**
  SerialProtocol Module Driver.
- **MYSTDLIB**
  MYSTDLIB module driver.

Variables

- **CLIPBOARD_SPI_TypeDef hclipboardR**
- TIM_HandleTypeDef * **htimR**
- **MY_BARGRAPH_BrightnessMode BrightnessMode**
- **CLIPBOARD_SPI_TypeDef hclipboard**
- TIM_HandleTypeDef * **htim**

Detailed Description

Variable Documentation

MY_BARGRAPH_BrightnessMode BrightnessMode

Enum BrightnessMode

Definition at line 57 of file bargraph.h.

CLIPBOARD_SPI_TypeDef hclipboard

Clipboard struct

Definition at line 43 of file rotary.h.

CLIPBOARD_SPI_TypeDef hclipboardR

Clipboard Handle

Definition at line 53 of file bargraph.h.

TIM_HandleTypeDef* htim

Timer struct

Definition at line 45 of file rotary.h.

TIM_HandleTypeDef* htimR

Timer Handle

Definition at line 55 of file bargraph.h.

CLIPBOARDS

CLIPBOARD Module Drivers.

Modules

- **BARGRAPH**
  BARGRAPHL Module Driver.
- **CLIPBOARD**
  CLIPBOARD Module Driver.
- **ROTARY**
  ROTARY Module Drivers.

Detailed Description

CLIPBOARD Module Drivers.

BARGRAPH

BARGRAPHL Module Driver.

Modules

- **BARGRAPH Private Functions**
- **BARGRAPH Exported Functions**
- **BARGRAPH_Private_Function**
- **BARGRAPH Exported Types**

Detailed Description

BARGRAPHL Module Driver.

BARGRAPH Private Functions

Detailed Description

BARGRAPH Exported Functions

Modules

- **BARGRAPH Initialization Functions**
  Funktionen fürs initialisieren des Bargraph Modules.
- **BARGRAPH Operation Functions**
  Operation functions.

Detailed Description

BARGRAPH Initialization Functions

Funktionen fürs initialisieren des Bargraph Modules.

Functions

- void **MY_BARGRAPH_Init_Strobed** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, TIM_HandleTypeDef *htim, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialisiert den Bargraphen mit der Strobed-Helligkeit.

- void **MY_BARGRAPH_Init_Pulsed** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, TIM_HandleTypeDef *htim, uint32_t channel, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialisiert den Bargraphen mit der Pulsed-Helligkeit.

Detailed Description

Funktionen fürs initialisieren des Bargraph Modules.

```
================================================================================
            ##### Initialization Functions  #####
================================================================================
```

[..] Dienen fürs initialisieren des Bargraph Modules

Function Documentation

void MY_BARGRAPH_Init_Pulsed (MY_BARGRAPH_HandleTypeDef * *hbargraph*, TIM_HandleTypeDef * *htim*, uint32_t *channel*, GPIO_TypeDef * *GPIO_MOSI_Port*, uint16_t *GPIO_MOSI_Pin*, GPIO_TypeDef * *GPIO_SCK_Port*, uint16_t *GPIO_SCK_Pin*, GPIO_TypeDef * *GPIO_CS_Port*, uint16_t *GPIO_CS_Pin*)

Initialisiert den Bargraphen mit der Pulsed-Helligkeit.

Parameters

hbargraph	Handle Bargraph
htim	Handle Timer für PWM
channel	PWM Channel
GPIO_MOSI_Port	MOSI_Port
GPIO_MOSI_Pin	MOSI_Pin
GPIO_SCK_Port	SCK_Port
GPIO_SCK_Pin	SCK_Pin
GPIO_CS_Port	CS_Port
GPIO_CS_Pin	CS_Pin

Definition at line 127 of file bargraph.c.

void MY_BARGRAPH_Init_Strobed (MY_BARGRAPH_HandleTypeDef * *hbargraph*, TIM_HandleTypeDef * *htim*, GPIO_TypeDef * *GPIO_MOSI_Port*, uint16_t

***GPIO_MOSI_Pin*, GPIO_TypeDef * *GPIO_SCK_Port*, uint16_t *GPIO_SCK_Pin*, GPIO_TypeDef * *GPIO_CS_Port*, uint16_t *GPIO_CS_Pin*)**

Initialisiert den Bargraphen mit der Strobed-Helligkeit.

Parameters

hbargraph	Handle Bargraph
htim	Handle Timer
GPIO_MOSI_Port	MOSI-Port des Clipboards
GPIO_MOSI_Pin	MOSI-Pin des Clipboards
GPIO_SCK_Port	SCK-Port des Clipboards
GPIO_SCK_Pin	SCK-Pin des Clipboards
GPIO_CS_Port	CS-Pin des Clipboards
GPIO_CS_Pin	CS-Pin des Clipboards

Definition at line 108 of file bargraph.c.

BARGRAPH Operation Functions

Operation functions.

Functions

- void **MY_BARGRAPH_SET_BITS** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, uint16_t *bit_array_of_segments, uint8_t *bit_array_of_brightness)
  Setzt die jeweiligen Segmente.

Detailed Description

Operation functions.

```
==============================================================================
                     #####  Operation Functions  #####
==============================================================================
   [..] Dienen zum steuern des Bargraph Moduls
     (+) Dient zum Ein- und Ausschalten der jeweiligen Segmente
```

(+) Dient zum einstellen der Helligkeit

Function Documentation

void MY_BARGRAPH_SET_BITS (MY_BARGRAPH_HandleTypeDef * *hbargraph*, uint16_t * *bit_array_of_segments*, uint8_t * *bit_array_of_brightness*)

Setzt die jeweiligen Segmente.

Parameters

hbargraph	Bargraph handle.
bit_array_of_leds	Bitarray, welche Segmente Ein- und Ausgeschaltet werden sollen
bit_array_of_brighness	Helligkeitsarray - sorgt für die Helligkeit jedes einzelnen Segmenes 0-100.

Definition at line 161 of file bargraph.c.

BARGRAPH_Private_Function

Detailed Description

BARGRAPH Exported Types

Data Structures

- struct **MY_BARGRAPH_HandleTypeDef**
  Bargraph Structure definition.

Enumerations

- enum **MY_BARGRAPH_BrightnessMode** {
  MY_BARGRAPH_BRITHNESSMODE_PULSED = 0,
  MY_BARGRAPH_BRITHNESSMODE_STROBED = 1 }
  Bargraph-Hellogkeit Enum definition.

Detailed Description

Enumeration Type Documentation

enum MY_BARGRAPH_BrightnessMode

Bargraph-Hellogkeit Enum definition.

Enumerator:

MY_BARGRAPH_BRITHNESSMODE_PULSED	Helligkeit mit PWM steuern
MY_BARGRAPH_BRITHNESSMODE_STROBED	Helligkeit mit Frequenz steuern

Definition at line 41 of file bargraph.h.

CLIPBOARD

CLIPBOARD Module Driver.

Modules

- **CLIPBOARD Exported Functions**
- **CLIPBOARD Exported Types**

Detailed Description

CLIPBOARD Module Driver.

CLIPBOARD Exported Functions

Modules

- **CLIPBOARD Initialization Functions**
  Initialization and Configuration functions.
- **CLIPBOARD SPI-Transfer functions**
  CLIPBOARD SPI-Transfer functions.

Detailed Description

CLIPBOARD Initialization Functions

Initialization and Configuration functions.

Functions

- void **MY_CLIPBOARD_Init** (**CLIPBOARD_SPI_TypeDef** *hclipboard, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Detailed Description

Initialization and Configuration functions.

Initialization and Configuration Functions.

```
==============================================================================
            ##### Initialization Functions  #####
==============================================================================
```

[..] Dienen fürs initialisieren des Clipboards

Function Documentation

void MY_CLIPBOARD_Init (CLIPBOARD_SPI_TypeDef * *hclipboard*, GPIO_TypeDef * *GPIO_MOSI_Port*, uint16_t *GPIO_MOSI_Pin*, GPIO_TypeDef * *GPIO_SCK_Port*, uint16_t *GPIO_SCK_Pin*, GPIO_TypeDef * *GPIO_CS_Port*, uint16_t *GPIO_CS_Pin*)

Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Parameters

GPIOx	where x can be (A..H) to select the GPIO peripheral for STM32L4 family
GPIO_Init	pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

Return values

None	

Definition at line 64 of file clipboard.c.

CLIPBOARD SPI-Transfer functions

CLIPBOARD SPI-Transfer functions.

Functions

- void **MY_CLIPBOARD_SPI_TX** (**CLIPBOARD_SPI_TypeDef** *hclipboard, uint8_t *data)
  Dient dazu um Daten über die SPI-Verbindung an das Clipboard zu senden.

- uint16_t **get_value_bitpositions** (uint16_t zahl, uint16_t stelle)
  Dient dazu um das Bit jeder Stelle zu ermitteln.

Detailed Description

CLIPBOARD SPI-Transfer functions.

```
================================================================================
                    #####  SPI-Transfer Functions  #####
================================================================================
```

[..] Dienen zur Kommunikation mit dem Clipboard

Function Documentation

uint16_t get_value_bitpositions (uint16_t *zahl*, uint16_t *stelle*)

Dient dazu um das Bit jeder Stelle zu ermitteln.

Parameters

zahl	Zahl
stelle	Stelle

Return values

Stellenwert	

Definition at line 120 of file clipboard.c.

void MY_CLIPBOARD_SPI_TX (CLIPBOARD_SPI_TypeDef * *hclipboard*, uint8_t * *data*)

Dient dazu um Daten über die SPI-Verbindung an das Clipboard zu senden.

Parameters

hclipboard	Clipboard Handler
data	Daten

Definition at line 100 of file clipboard.c.

CLIPBOARD Exported Types

Data Structures

- struct **CLIPBOARD_SPI_TypeDef**
  TIM Time base Configuration Structure definition.

Detailed Description

ROTARY

ROTARY Module Drivers.

Modules

- **ROTARY Exported Functions**
- **ROTARY Exported Types**

Detailed Description

ROTARY Module Drivers.

ROTARY Exported Functions

Modules

- **Initialization Functions**
  Initialization Functions.
- **Operation functions**
  Operation functions.

Detailed Description

Initialization Functions

Initialization Functions.

Functions

- void **MY_ROTARY_Init_Strobed** (**MY_ROTARY_HandleTypeDef** *hrotary, TIM_HandleTypeDef *htim, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin, GPIO_TypeDef *GPIO_ENCA_Port, uint16_t GPIO_ENCA_Pin, GPIO_TypeDef *GPIO_ENCB_Port, uint16_t GPIO_ENCB_Pin, GPIO_TypeDef *GPIO_SWITCH_Port, uint16_t GPIO_SWITCH_Pin)
  Funktion Initialisiert Encoder.

Detailed Description

Initialization Functions.

```
==============================================================================
           ##### Initialization Functions  #####
==============================================================================
```

[..] Dienen fürs initialisieren des Rotary Modules

Function Documentation

void MY_ROTARY_Init_Strobed (MY_ROTARY_HandleTypeDef * ***hrotary*****, TIM_HandleTypeDef *** ***htim*****, GPIO_TypeDef *** ***GPIO_MOSI_Port*****, uint16_t** ***GPIO_MOSI_Pin*****, GPIO_TypeDef *** ***GPIO_SCK_Port*****, uint16_t** ***GPIO_SCK_Pin*****, GPIO_TypeDef *** ***GPIO_CS_Port*****, uint16_t** ***GPIO_CS_Pin*****, GPIO_TypeDef *** ***GPIO_ENCA_Port*****, uint16_t** ***GPIO_ENCA_Pin*****, GPIO_TypeDef *** ***GPIO_ENCB_Port*****, uint16_t** ***GPIO_ENCB_Pin*****, GPIO_TypeDef *** ***GPIO_SWITCH_Port*****, uint16_t** ***GPIO_SWITCH_Pin*****)**

Funktion Initialisiert Encoder.

Parameters

hrotary	rotary handle
htim	htim handle
GPIO_MOSI_Port	MOSI-Port des Clipboards
GPIO_MOSI_Pin	MOSI-Pin des Clipboards
GPIO_SCK_Port	SCK-Port des Clipboards
GPIO_SCK_Pin	SCK-Pin des Clipboards
GPIO_CS_Port	MOSI-Port des Clipboards
GPIO_CS_Pin	SCK-Pin des Clipboards
GPIO_ENCA_Port	ENCA-Port des Clipboards
GPIO_ENCA_Pin	ENCA-Pin des Clipboards
GPIO_ENCB_Port	ENCB-Port des Clipboards
GPIO_ENCB_Pin	ENCB-Pin des Clipboards
GPIO_SWITCH_Port	SWITCH-Port des Clipboards
GPIO_SWITCH_P	SWITCH-Pin des Clipboards

in	

Definition at line 115 of file rotary.c.

Operation functions

Operation functions.

Functions

- uint8_t **MY_ROTARY_GetEncoderEvent** ()
  Dient dazu um die Drehrichtung des Encoder festzustellen und ob der Taster gedrückt wurde.

- void **MY_ROTARY_SET_LEDS** (**MY_ROTARY_HandleTypeDef** *hrotary, uint16_t *bit_array_of_leds, uint8_t *bit_array_of_brightness)
  Funktion Setzt die LED's des Encoders.

Detailed Description

Operation functions.

```
==============================================================================
                    #####  Operation Functions  #####
==============================================================================
   [..] Dienen zum steuern des Rotary Moduls
     (+) Dient zum Ein- und Ausschalten der jeweiligen LED's
```

(+) Dient zum Einstellen der Helligkeit

Function Documentation

uint8_t MY_ROTARY_GetEncoderEvent ()

Dient dazu um die Drehrichtung des Encoder festzustellen und ob der Taster gedrückt wurde.

Return values

ENCODER	PinEvent / 1=Rechtsdreh, 2=Linksdreh und 3=Taster gedrückt

Definition at line 152 of file rotary.c.

void MY_ROTARY_SET_LEDS (MY_ROTARY_HandleTypeDef * *hrotary*, uint16_t * *bit_array_of_leds*, uint8_t * *bit_array_of_brightness*)

Funktion Setzt die LED's des Encoders.

Parameters

hrotary	rotary handle
bit_array_of_leds	Jedes Bit steht für eine LED / 0=aus, 1=ein
bit_array_of_brightness	Helligkeit jeder LED 0-100%

Definition at line 175 of file rotary.c.

ROTARY Exported Types

Data Structures

- struct **MY_ROTARY_HandleTypeDef**
  ROTARY Configuration Structure definition.

Detailed Description

SERIALPROTOCOL

SerialProtocol Module Driver.

Modules

- **SERIALPROTOCOL Private Defines**
- **SERIALPROTOCOL Private Functions**
- **SERIALPROTOCOL Exported Functions**
- **SERIALPROTOCOL Exported Types**
- **SerialProtocol Exported Macros**

Detailed Description

SerialProtocol Module Driver.

SERIALPROTOCOL Private Defines

Macros

- #define **CollectionBuffer_SIZE** 65
- #define **STM32_ACK** "STM32-ACK -> "
- #define **STM32_NACK** "STM32-NACK -> "
- #define **NEW_LINE** "\n\r"

Detailed Description

Macro Definition Documentation

#define CollectionBuffer_SIZE 65

Groesse des Sendepuffers

Definition at line 78 of file serialprotocol.c.

#define NEW_LINE "\n\r"

Neue Zeile

Definition at line 81 of file serialprotocol.c.

#define STM32_ACK "STM32-ACK -> "

Nachricht OK

Definition at line 79 of file serialprotocol.c.

#define STM32_NACK "STM32-NACK -> "

Nachricht falsch oder nicht erkannt

Definition at line 80 of file serialprotocol.c.

SERIALPROTOCOL Private Functions

Detailed Description

SERIALPROTOCOL Exported Functions

Modules

- **Initialization Functions**
  Initialization Functions.
- **Callback Functions**
  Callback Functions.

Detailed Description

Initialization Functions

Initialization Functions.

Functions

- void **MYLIB_SERIALPROT_XCHANGE** (**SERIALPROTOCOL_TypeDef** *hserialprot, uint8_t *RxBuffer, uint8_t *TxBuffer)
  Funktion verarbeitet die einzel Eingegebenen Zeichen von RxBuffer und gibt dementsprechend die Antwort im TXBuffer zurück.

Detailed Description

Initialization Functions.

```
================================================================================
            ##### Initialization Functions  #####
================================================================================
```

[..] Dienen fürs initialisieren des Rotary Modules

Function Documentation

void MYLIB_SERIALPROT_XCHANGE (SERIALPROTOCOL_TypeDef * *hserialprot*, uint8_t * *RxBuffer*, uint8_t * *TxBuffer*)

Funktion verarbeitet die einzel Eingegebenen Zeichen von RxBuffer und gibt dementsprechend die Antwort im TXBuffer zurück.

Parameters

hserialprot	SERIALPROT handle
RxBuffer	Ein-Zeichen-Empfangspuffer
TxBuffer	Sendepuffer/Antwortpuffer

Return values

none	

Definition at line 144 of file serialprotocol.c.

Callback Functions

Callback Functions.

Functions

- __weak uint8_t **SERIALPROT_Command_GPO_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "gpo" Befehls abzufragen.

- __weak uint8_t **SERIALPROT_Command_CRS_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crs" Befehls abzufragen.

- __weak uint16_t **SERIALPROT_Command_CRG_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crg" Befehls abzufragen.

- __weak uint8_t **SERIALPROT_Command_CBS_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbs" Befehls abzufragen.

- __weak uint16_t **SERIALPROT_Command_CBG_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbg" Befehls abzufragen.

Detailed Description

Callback Functions.

```
================================================================================
                    #####  Callback Functions  #####
================================================================================
```

[..] Dienen dazu, um die Funktionen in die main.c zum kopieren um Befehle abzufragen und diese zu Handeln

---

Function Documentation

uint16_t SERIALPROT_Command_CBG_Callback (SERIALPROTOCOL_TypeDef * *hserialprot*)

Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbg" Befehls abzufragen.

Parameters

hserialprot	SERIALPROT handle

Return values

query	

Definition at line 294 of file serialprotocol.c.

uint8_t SERIALPROT_Command_CBS_Callback (SERIALPROTOCOL_TypeDef * *hserialprot*)

Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbs" Befehls abzufragen.

Parameters

hserialprot	SERIALPROT handle

Return values

query	

Definition at line 278 of file serialprotocol.c.

uint16_t SERIALPROT_Command_CRG_Callback (SERIALPROTOCOL_TypeDef * *hserialprot*)

Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crg" Befehls abzufragen.

Parameters

hserialprot	SERIALPROT handle

Return values

query	

Definition at line 262 of file serialprotocol.c.

uint8_t SERIALPROT_Command_CRS_Callback (SERIALPROTOCOL_TypeDef * *hserialprot*)

Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crs" Befehls abzufragen.

Parameters

hserialprot	SERIALPROT handle

Return values

query	

Definition at line 246 of file serialprotocol.c.

uint8_t SERIALPROT_Command_GPO_Callback (SERIALPROTOCOL_TypeDef * *hserialprot*)

Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "gpo" Befehls abzufragen.

Parameters

hserialprot	SERIALPROT handle

Return values

query	

Definition at line 230 of file serialprotocol.c.

SERIALPROTOCOL Exported Types

Data Structures

- struct **SERIALPROTOCOL_TypeDef**
  SERIALPROTOCOL Status structures definition.

Enumerations

- enum **SERIALPROTCOL_StatusTypeDef** { **SERIALPROT_OK** = 0x00, **SERIALPROT_ERROR** = 0x01 }
  SERIALPROTOCOL Message Status definition.
- enum **SERIALPROTOCOL_MessageKindTypeDef** { **MESSAGEKIND_TEXT_NUMBER** = 0x00, **MESSAGEKIND_NUMBER_NUMBER** = 0x01, **MESSAGEKIND_TEXT_TEXT** = 0x02, **MESSAGEKIND_NUMBER_TEXT** = 0x03 }
  SERIALPROTOCOL Message Kind definition.

Detailed Description

Enumeration Type Documentation

enum SERIALPROTCOL_StatusTypeDef

SERIALPROTOCOL Message Status definition.

Enumerator:

SERIALPROT_OK	Nachricht OK
SERIALPROT_ERROR	Nachricht nicht OK

Definition at line 36 of file serialprotocol.h.

enum SERIALPROTOCOL_MessageKindTypeDef

SERIALPROTOCOL Message Kind definition.

Enumerator:

MESSAGEKIND_TEXT_NUMBER	Text-Nummer Nachricht
MESSAGEKIND_NUMBER_NUMBER	Nummer-Nummer Nachricht
MESSAGEKIND_TEXT_TEXT	Text-Text Nachricht
MESSAGEKIND_NUMBER_TEXT	Nummer-Text Nachricht

Definition at line 46 of file serialprotocol.h.

SerialProtocol Exported Macros

Macros

- #define **__SERIALPROT_IS_COMMAND**(__HANDLE__, __COMMANDNAME__, __PARAMETER1__, __PARAMETER2__) ((!strcmp((__HANDLE__)->CommandName,(__COMMANDNAME__))) && (!strcmp((__HANDLE__)->Parameter1,(__PARAMETER1__))) && (!strcmp((__HANDLE__)->Parameter2,(__PARAMETER2__))))
  SERIALPROT Kommando-Abfragen.

- #define **__SERIALPROT_IS_COMMANDNAME**(__HANDLE__, __COMMANDNAME__) (!strcmp((__HANDLE__)->CommandName,(__COMMANDNAME__)))
  SERIALPROT Kommando-Abfragen.

- #define **__SERIALPROT_IS_PARAMETER1**(__HANDLE__, __PARAMETER1__) (!strcmp((__HANDLE__)->Parameter1,(__PARAMETER1__)))
  SERIALPROT Kommando-Abfragen.

- #define **__SERIALPROT_IS_PARAMETER2**(__HANDLE__, __PARAMETER2__) (!strcmp((__HANDLE__)->Parameter2,(__PARAMETER2__)))
  SERIALPROT Kommando-Abfragen.

Detailed Description

Macro Definition Documentation

#define __SERIALPROT_IS_COMMAND(__HANDLE__, __COMMANDNAME__, __PARAMETER1__, __PARAMETER2__) ((!strcmp((__HANDLE__)->CommandName,(__COMMANDNAME__))) && (!strcmp((__HANDLE__)->Parameter1,(__PARAMETER1__))) && (!strcmp((__HANDLE__)->Parameter2,(__PARAMETER2__))))

SERIALPROT Kommando-Abfragen.

Parameters

HANDLE	SERIALPROT handle.
COMMANDNAME	SERIALPROT Kommandoname
PARAMETER1	SERIALPROT Parameter1 des Kommandos
PARAMETER2	SERIALPROT Parameter2 des Kommandos

Return values

None	

Definition at line 85 of file serialprotocol.h.

#define __SERIALPROT_IS_COMMANDNAME(__HANDLE__, __COMMANDNAME__) (!strcmp((__HANDLE__)->CommandName,(__COMMANDNAME__)))

SERIALPROT Kommando-Abfragen.

Parameters

HANDLE	SERIALPROT handle.
COMMANDNAME	SERIALPROT Kommandoname
PARAMETER1	SERIALPROT Parameter1 des Kommandos
PARAMETER2	SERIALPROT Parameter2 des Kommandos

Return values

None	

Definition at line 95 of file serialprotocol.h.

#define __SERIALPROT_IS_PARAMETER1(__HANDLE__, __PARAMETER1__) (!strcmp((__HANDLE__)->Parameter1,(__PARAMETER1__)))

SERIALPROT Kommando-Abfragen.

Parameters

HANDLE	SERIALPROT handle.
COMMANDNAME	SERIALPROT Kommandoname
PARAMETER1	SERIALPROT Parameter1 des Kommandos
PARAMETER2	SERIALPROT Parameter2 des Kommandos

Return values

None	

Definition at line 105 of file serialprotocol.h.

#define __SERIALPROT_IS_PARAMETER2(__HANDLE__, __PARAMETER2__) (!strcmp((__HANDLE__)->Parameter2,(__PARAMETER2__)))

SERIALPROT Kommando-Abfragen.

Parameters

HANDLE	SERIALPROT handle.

COMMANDNAME	SERIALPROT Kommandoname
PARAMETER1	SERIALPROT Parameter1 des Kommandos
PARAMETER2	SERIALPROT Parameter2 des Kommandos

Return values

None	

Definition at line 115 of file serialprotocol.h.

MYSTDLIB

MYSTDLIB module driver.

Modules

- **MYSTDLIB Exported Functions**

Detailed Description

MYSTDLIB module driver.

MYSTDLIB Exported Functions

Modules

- **MYSTDLIB Operation functions**
  Operation Functions.

Functions

- void **MY_STDLIB_Delay** (TIM_HandleTypeDef *htim, uint16_t us)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Detailed Description

Function Documentation

void MY_STDLIB_Delay (TIM_HandleTypeDef * *htim*, uint16_t *us*)

Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Parameters

GPIOx	where x can be (A..H) to select the GPIO peripheral for STM32L4 family
GPIO_Init	pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

Return values

None	

Definition at line 74 of file mystdlib.c.

MYSTDLIB Operation functions

Operation Functions.

Functions

- void **MY_STDLIB_Delay** (TIM_HandleTypeDef *htim, uint16_t us)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Detailed Description

Operation Functions.

```
 ==============================================================================
                     #####  Operation Functions  #####
 ==============================================================================
    [..] Dienen zum steuern des Rotary Moduls
      (+) Dient zum Ein- und Ausschalten der jeweiligen LED's
```

(+) Dient zum Einstellen der Helligkeit

Function Documentation

void MY_STDLIB_Delay (TIM_HandleTypeDef * *htim*, uint16_t *us*)

Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Parameters

GPIOx	where x can be (A..H) to select the GPIO peripheral for STM32L4 family
GPIO_Init	pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

Return values

None	

Definition at line 74 of file mystdlib.c.

Data Structure Documentation

CLIPBOARD_SPI_TypeDef Struct Reference

TIM Time base Configuration Structure definition.
`#include <clipboard.h>`

Data Fields

- GPIO_TypeDef * **GPIO_MOSI_Port**
- uint16_t **GPIO_MOSI_Pin**
- GPIO_TypeDef * **GPIO_SCK_Port**
- uint16_t **GPIO_SCK_Pin**
- GPIO_TypeDef * **GPIO_CS_Port**
- uint16_t **GPIO_CS_Pin**

Detailed Description

TIM Time base Configuration Structure definition.

Definition at line 44 of file clipboard.h.

Field Documentation

uint16_t GPIO_CS_Pin

CS-Pin des Clipboards

Definition at line 56 of file clipboard.h.

GPIO_TypeDef* GPIO_CS_Port

CS-Port des Clipboards

Definition at line 54 of file clipboard.h.

uint16_t GPIO_MOSI_Pin

MOSI-Pin des Clipboards

Definition at line 48 of file clipboard.h.

GPIO_TypeDef* GPIO_MOSI_Port

MOSI-Port des Clipboards

Definition at line 46 of file clipboard.h.

uint16_t GPIO_SCK_Pin

SCK-Pin des Clipboards

Definition at line 52 of file clipboard.h.

GPIO_TypeDef* GPIO_SCK_Port

SCK-Port des Clipboards

Definition at line 50 of file clipboard.h.

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

- C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/**clipboard.h**

MY_BARGRAPH_HandleTypeDef Struct Reference

Bargraph Structure definition.
`#include <bargraph.h>`

Data Fields

- **CLIPBOARD_SPI_TypeDef hclipboardR**
- TIM_HandleTypeDef * **htimR**
- **MY_BARGRAPH_BrightnessMode BrightnessMode**

Detailed Description

Bargraph Structure definition.

Definition at line 51 of file bargraph.h.

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

- C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/Bargraph/**bargraph.h**

MY_ROTARY_HandleTypeDef Struct Reference

ROTARY Configuration Structure definition.
`#include <rotary.h>`

Data Fields

- **CLIPBOARD_SPI_TypeDef hclipboard**
- TIM_HandleTypeDef * **htim**

Detailed Description

ROTARY Configuration Structure definition.

Definition at line 41 of file rotary.h.

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

- C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/Rotary/**rotary.h**

SERIALPROTOCOL_TypeDef Struct Reference

SERIALPROTOCOL Status structures definition.
`#include <serialprotocol.h>`

Data Fields

- uint8_t **CommandName** [5]
- **SERIALPROTOCOL_MessageKindTypeDef MessageKind**
- uint8_t **Parameter1** [15]
- uint8_t **Parameter2** [15]

Detailed Description

SERIALPROTOCOL Status structures definition.

Definition at line 58 of file serialprotocol.h.

Field Documentation

uint8_t CommandName[5]

Kommandoname

Definition at line 60 of file serialprotocol.h.

SERIALPROTOCOL_MessageKindTypeDef MessageKind

Nachrichtentyp

Definition at line 62 of file serialprotocol.h.

uint8_t Parameter1[15]

Parameter1 des Kommandos

Definition at line 64 of file serialprotocol.h.

uint8_t Parameter2[15]

Parameter2 des Kommandos

Definition at line 66 of file serialprotocol.h.

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

- C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/SerialProtocol/**serialprotocol.h**

File Documentation

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/Bargraph/bargraph.c File Reference

BARGRAPH module driver. Dieses File dient dazu, um das Bargraph Click-Board anzusteuern bzw. einzelne Segmente Ein- und Ausgeschaltet sowie die Helligkeit eingestellt werden.

```
#include "bargraph.h"
```

Functions

- void **MY_BARGRAPH_Init_Strobed** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, TIM_HandleTypeDef *htim, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialisiert den Bargraphen mit der Strobed-Helligkeit.

- void **MY_BARGRAPH_Init_Pulsed** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, TIM_HandleTypeDef *htim, uint32_t channel, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialisiert den Bargraphen mit der Pulsed-Helligkeit.

- void **MY_BARGRAPH_SET_BITS** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, uint16_t *bit_array_of_segments, uint8_t *bit_array_of_brightness)
  Setzt die jeweiligen Segmente.

Detailed Description

BARGRAPH module driver. Dieses File dient dazu, um das Bargraph Click-Board anzusteuern bzw. einzelne Segmente Ein- und Ausgeschaltet sowie die Helligkeit eingestellt werden.

Author

Reiter Roman

Version

1.0

Date

Created on: Jan 23, 2022

```
==============================================================================
                  ##### How to use this driver #####
==============================================================================
 [..] Als erstes muss das BARGRAPH module initialisiert werden, welches auf
    zwei unterschiedliche Arten erfolgen kann.

  (#) void MY_BARGRAPH_Init_Strobed(..)
      (++) Mit dieser Initialisierung ist es möglich die einzelnen Segmente
         Ein- und Auszuschalten sowie die Helligkeit jedes einzelnen Moduls
         mit der Funktion MY_BARGRAPH_SET_BITS(...) einzustellen.
```

```
(#) void MY_BARGRAPH_Init_Pulsed(..)
    (++) Mit dieser Initialisierung ist es möglich die einzelnen Segmente
       Ein- und Auszuschalten sowie die Helligkeit des gesamten Moduls
```

einzustellen.

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/Bargraph/bargraph.h File Reference

Header file des BARGRAPH modules.
`#include "Clipboards/clipboard.h"`

Data Structures

- struct **MY_BARGRAPH_HandleTypeDef**
  Bargraph Structure definition.

Enumerations

- enum **MY_BARGRAPH_BrightnessMode** {
  MY_BARGRAPH_BRITHNESSMODE_PULSED = 0,
  MY_BARGRAPH_BRITHNESSMODE_STROBED = 1 }
  Bargraph-Hellogkeit Enum definition.

Functions

- void **MY_BARGRAPH_Init_Strobed** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, TIM_HandleTypeDef *htim, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialisiert den Bargraphen mit der Strobed-Helligkeit.

- void **MY_BARGRAPH_Init_Pulsed** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, TIM_HandleTypeDef *htim, uint32_t channel, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialisiert den Bargraphen mit der Pulsed-Helligkeit.

- void **MY_BARGRAPH_SET_BITS** (**MY_BARGRAPH_HandleTypeDef** *hbargraph, uint16_t *bit_array_of_segments, uint8_t *bit_array_of_brightness)
  Setzt die jeweiligen Segmente.

Detailed Description

Header file des BARGRAPH modules.

Author

Reiter Roman

Version

1.0

Date

Created on: Jan 23, 2022

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/clipboard.c File Reference

```
#include <Clipboards/clipboard.h>
```

Functions

- void **MY_CLIPBOARD_Init** (**CLIPBOARD_SPI_TypeDef** *hclipboard, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

- void **MY_CLIPBOARD_SPI_TX** (**CLIPBOARD_SPI_TypeDef** *hclipboard, uint8_t *data)
  Dient dazu um Daten über die SPI-Verbindung an das Clipboard zu senden.

- uint16_t **get_value_bitpositions** (uint16_t zahl, uint16_t stelle)
  Dient dazu um das Bit jeder Stelle zu ermitteln.

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/clipboard.h File Reference

```
#include "stm32l4xx.h"
```

Data Structures

- struct **CLIPBOARD_SPI_TypeDef**
  TIM Time base Configuration Structure definition.

Functions

- void **MY_CLIPBOARD_Init** (**CLIPBOARD_SPI_TypeDef** *hclipboard, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

- void **MY_CLIPBOARD_SPI_TX** (**CLIPBOARD_SPI_TypeDef** *hclipboard, uint8_t *data)
  Dient dazu um Daten über die SPI-Verbindung an das Clipboard zu senden.

- uint16_t **get_value_bitpositions** (uint16_t zahl, uint16_t stelle)
  Dient dazu um das Bit jeder Stelle zu ermitteln.

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/clipboard_def.h File Reference

```
#include "Rotary/rotary.h"
#include "Bargraph/bargraph.h"
```

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/Rotary/rotary.c File Reference

ROTARY module driver. Dieses File dient dazu, um das Rotary Click-Board anzusteuern bzw. einzelne LED's ein- und auszuschalten und deren Helligkeit einzustellen.

```
#include "rotary.h"
```

Functions

- void **MY_ROTARY_Init_Strobed** (**MY_ROTARY_HandleTypeDef** *hrotary, TIM_HandleTypeDef *htim, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin, GPIO_TypeDef *GPIO_ENCA_Port, uint16_t GPIO_ENCA_Pin, GPIO_TypeDef *GPIO_ENCB_Port, uint16_t GPIO_ENCB_Pin, GPIO_TypeDef *GPIO_SWITCH_Port, uint16_t GPIO_SWITCH_Pin)
  Funktion Initialisiert Encoder.

- uint8_t **MY_ROTARY_GetEncoderEvent** ()
  Dient dazu um die Drehrichtung des Encoder festzustellen und ob der Taster gedrückt wurde.

- void **MY_ROTARY_SET_LEDS** (**MY_ROTARY_HandleTypeDef** *hrotary, uint16_t *bit_array_of_leds, uint8_t *bit_array_of_brightness)
  Funktion Setzt die LED's des Encoders.

Detailed Description

ROTARY module driver. Dieses File dient dazu, um das Rotary Click-Board anzusteuern bzw. einzelne LED's ein- und auszuschalten und deren Helligkeit einzustellen.

Author

Reiter Roman

Version

1.0

Date

Created on: Jan 23, 2022

```
==============================================================================
                  ##### How to use this driver #####
==============================================================================
 [..] Als erstes muss das ROTARY module initialisiert werden, welches auf
    zwei unterschiedliche Arten erfolgen kann.

  (#) void MY_ROTARY_Init_Strobed(..)
      (++) Mit dieser Initialisierung ist es möglich einzelnen LED's
```

Ein- und Auszuschalten sowie deren Helligkeit einzelnen einzustellen.

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/Clipboards/Rotary/rotary.h File Reference

Header file of GPIO HAL module.
`#include <Clipboards/clipboard.h>`

Data Structures

- struct **MY_ROTARY_HandleTypeDef**
  ROTARY Configuration Structure definition.

Functions

- void **MY_ROTARY_Init_Strobed** (**MY_ROTARY_HandleTypeDef** *hrotary, TIM_HandleTypeDef *htim, GPIO_TypeDef *GPIO_MOSI_Port, uint16_t GPIO_MOSI_Pin, GPIO_TypeDef *GPIO_SCK_Port, uint16_t GPIO_SCK_Pin, GPIO_TypeDef *GPIO_CS_Port, uint16_t GPIO_CS_Pin, GPIO_TypeDef *GPIO_ENCA_Port, uint16_t GPIO_ENCA_Pin, GPIO_TypeDef *GPIO_ENCB_Port, uint16_t GPIO_ENCB_Pin, GPIO_TypeDef *GPIO_SWITCH_Port, uint16_t GPIO_SWITCH_Pin)
  Funktion Initialisiert Encoder.

- void **MY_ROTARY_SET_LEDS** (**MY_ROTARY_HandleTypeDef** *hrotary, uint16_t *bit_array_of_leds, uint8_t *bit_array_of_brightness)
  Funktion Setzt die LED's des Encoders.

- uint8_t **MY_ROTARY_GetEncoderEvent** ()
  Dient dazu um die Drehrichtung des Encoder festzustellen und ob der Taster gedrückt wurde.

Detailed Description

Header file of GPIO HAL module.

Author

Reiter Roman

Version

1.0

Date

Created on: Jan 23, 2022

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/mylibrary.h File Reference

Header file of MYLIBRARY module.

```
#include <Clipboards/clipboard_def.h>
#include "SerialProtocol/serialprotocol.h"
```

Detailed Description

Header file of MYLIBRARY module.

Author

Reiter Roman

Version

1.0

Date

Created on: 24.12.2021

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/SerialProtocol/serialprotocol.c File Reference

mylib-Serielles Protokoll. Diese Datei bietet Funktionen zur Verwaltung der folgenden Funktionalitäten eines seriellen Protokolls in Verbindung mit dem UART2:

```
#include "stdint.h"
#include "stddef.h"
#include "serialprotocol.h"
#include "stdlib.h"
#include "string.h"
```

Macros

- #define **CollectionBuffer_SIZE** 65
- #define **STM32_ACK** "STM32-ACK -> "
- #define **STM32_NACK** "STM32-NACK -> "
- #define **NEW_LINE** "\n\r"

Functions

- void **MYLIB_SERIALPROT_XCHANGE** (**SERIALPROTOCOL_TypeDef** *hserialprot, uint8_t *RxBuffer, uint8_t *TxBuffer)
  Funktion verarbeitet die einzel Eingegebenen Zeichen von RxBuffer und gibt dementsprechend die Antwort im TXBuffer zurück.

- __weak uint8_t **SERIALPROT_Command_GPO_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "gpo" Befehls abzufragen.

- __weak uint8_t **SERIALPROT_Command_CRS_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crs" Befehls abzufragen.

- __weak uint16_t **SERIALPROT_Command_CRG_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crg" Befehls abzufragen.

- __weak uint8_t **SERIALPROT_Command_CBS_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbs" Befehls abzufragen.

- __weak uint16_t **SERIALPROT_Command_CBG_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbg" Befehls abzufragen.

Detailed Description

mylib-Serielles Protokoll. Diese Datei bietet Funktionen zur Verwaltung der folgenden Funktionalitäten eines seriellen Protokolls in Verbindung mit dem UART2:

Author

Reiter Roman

- IO-Betriebsfunktionen
- Zustands- und Fehlerfunktionen

```
  ==============================================================================
                    ##### How to use this driver #####
  ==============================================================================
   [..] Der SERIALPROT MYLIB-Treiber kann wie folgt verwendet werden:

    (#) Einbinden des UART2
        (+) Damit das serielle Protokoll verwendet werden kann muss der UART2 aktiviert
werden.
            (++) Der UART2 muss auf den Pins PA4 (TX) und PA15(RX) liegen
            (++) Für den UART2 werden die Standardeinstellungen der IDE-CubeMX verwendet
(Baudrate, Wortlänge, Stop-Bit, Parität, Prescaler-Wert..)
            (++) Zusätzlich muss der globale Interrupt in der NVIC-Konfiguration des UART2
aktiviert werden.
            (++) Das Empfangen der Daten erfolgt mit HAL_UART_Receive_IT ()
            (++) Das Senden der Daten erfolgt mit HAL_UART_Transmit ()
            (++) Da der Datenampfang per Interrupt erfolgt, muss die Callback-Methode
HAL_UART_TxCpltCallback () definiert werden.

    (#) Verwenden des seriellen Protokolls
        (+) Für das serielle Protokoll wird der UART2 benötigt.
            (++) Die Daten werden mit dem UART2 per Interrupt zeichenweise empfangen. Dafür
muss ein Empfangspuffer deklariert und initialisiert werden.
                (+++) z.B.: uint8_t RxBuffer[RxBuffer_SIZE]={0};
            (++) Der Funktion HAL_UART_Receive_IT () muss der Empfangspuffer übergeben
werden.
                (+++) z.B.: HAL_UART_Receive_IT(&huart2, RxBuffer, 1)
            (++) Nach einem Zeichenempfang wird die HAL_UART_RxCpltCallback () aufgerufen
                (+++) Um die Eingabe mit dem seriellen Protokoll zu verknüpfen muss ein
exchangePuffer angelegt werden,
                 welcher die Antworten zu den getätigten Eingaben in RxBuffer enthält.
                    (++++) z.B.: uint8_t exchangedMessage[50] ={0};
                (+++) Als Schnittstelle für die Eingabe (RxBuffer) und der Ausgabe
(exchangedMessage) muss die Funktion
                   MYLIB_SERIALPROT_XCHANGE() aufgerufen werden.
                    (++++) z.B.:
MYLIB_SERIALPROT_XCHANGE(&hserialprot,RxBuffer,exchangedMessage);
                (+++) Die erstellte Antwortnachricht muss nun über den UART2
hinausgeschrieben werden HAL_UART_Transmit()
                    (++++) z.B.: HAL_UART_Transmit(&huart2,
exchangedMessage,(uint16_t)strlen(exchangedMessage), 100)
                (+++) Abschließend muss der Interrupt für den UART2-Empfang wieder
aktiviert werden HAL_UART_Receive_IT()
                    (++++) z.B.: HAL_UART_Receive_IT(&huart2, RxBuffer, RxBuffer_SIZE)

    (#) Verwenden der Callback-Funktion SERIALPROT_Command_GPO_Callback()
        (+) Die Funktion dient dazu, um GPIO's ansteuern zu können.
            (++) Dazu wird die Callback-Funktion SERIALPROT_Command_GPO_Callback() in die
main.c kopiert
            (++) Für die Abfragen bzw. das Festlegen der Kommandos können diese einfach
in einer If-Schleige abgefragt werden.
                (+++) zur Vereinfachung der Abfrage wird das Makro
__SERIALPROT_IS_COMMAND() zu verfügung gestellt,
                     wodurch die Eingabeparameter des letzten Kommandos abgefragt werden
können
                     (++++) z.B.:
__SERIALPROT_IS_COMMAND(hserialprot,"gpo","rt","off")
```

(+++) Je nach Ergebnis muss beim erfüllen der Bedingung eine 0, anderfalls eine 1 zurückgegeben werden

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/SerialProtocol/serialprotocol.h File Reference

Header file des SerialProtocol Module.
`#include "stm32l4xx_hal.h"`

Data Structures

- struct **SERIALPROTOCOL_TypeDef**
  SERIALPROTOCOL Status structures definition.

Macros

- #define **__SERIALPROT_IS_COMMAND**(__HANDLE__, __COMMANDNAME__, __PARAMETER1__, __PARAMETER2__) ((!strcmp((__HANDLE__)->CommandName,(__COMMANDNAME__))) && (!strcmp((__HANDLE__)->Parameter1,(__PARAMETER1__))) && (!strcmp((__HANDLE__)->Parameter2,(__PARAMETER2__))))
  SERIALPROT Kommando-Abfragen.

- #define **__SERIALPROT_IS_COMMANDNAME**(__HANDLE__, __COMMANDNAME__) (!strcmp((__HANDLE__)->CommandName,(__COMMANDNAME__)))
  SERIALPROT Kommando-Abfragen.

- #define **__SERIALPROT_IS_PARAMETER1**(__HANDLE__, __PARAMETER1__) (!strcmp((__HANDLE__)->Parameter1,(__PARAMETER1__)))
  SERIALPROT Kommando-Abfragen.

- #define **__SERIALPROT_IS_PARAMETER2**(__HANDLE__, __PARAMETER2__) (!strcmp((__HANDLE__)->Parameter2,(__PARAMETER2__)))
  SERIALPROT Kommando-Abfragen.

Enumerations

- enum **SERIALPROTCOL_StatusTypeDef** { **SERIALPROT_OK** = 0x00, **SERIALPROT_ERROR** = 0x01 }
  SERIALPROTOCOL Message Status definition.
- enum **SERIALPROTOCOL_MessageKindTypeDef** { **MESSAGEKIND_TEXT_NUMBER** = 0x00, **MESSAGEKIND_NUMBER_NUMBER** = 0x01, **MESSAGEKIND_TEXT_TEXT** = 0x02, **MESSAGEKIND_NUMBER_TEXT** = 0x03 }
  SERIALPROTOCOL Message Kind definition.

Functions

- void **MYLIB_SERIALPROT_XCHANGE** (**SERIALPROTOCOL_TypeDef** *hserialprot, uint8_t *RxBuffer, uint8_t *TxBuffer)
  Funktion verarbeitet die einzel Eingegebenen Zeichen von RxBuffer und gibt dementsprechend die Antwort im TXBuffer zurück.

- __weak uint8_t **SERIALPROT_Command_GPO_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)

Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "gpo" Befehls abzufragen.

- __weak uint8_t **SERIALPROT_Command_CRS_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crs" Befehls abzufragen.

- __weak uint16_t **SERIALPROT_Command_CRG_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "crg" Befehls abzufragen.

- __weak uint8_t **SERIALPROT_Command_CBS_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbs" Befehls abzufragen.

- __weak uint16_t **SERIALPROT_Command_CBG_Callback** (**SERIALPROTOCOL_TypeDef** *hserialprot)
  Funktion Callback, welche in die main.c kopiert werden kann um die Eingabe des "cbg" Befehls abzufragen.

Detailed Description

Header file des SerialProtocol Module.

Author

Reiter Roman

Version

1.0

Date

Created on: Jan 23, 2022

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/StdLib/mystdlib.c File Reference

StdLib This file provides firmware functions to manage the following functionalities of the General Purpose Input/Output (GPIO) peripheral:
`#include "StdLib/mystdlib.h"`

Functions

- void **MY_STDLIB_Delay** (TIM_HandleTypeDef *htim, uint16_t us)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Detailed Description

StdLib This file provides firmware functions to manage the following functionalities of the General Purpose Input/Output (GPIO) peripheral:

Author

Reiter Roman

Version

1.0

Date

Created on: 24.12.2021

- Initialization and de-initialization functions
- IO operation functions

```
==============================================================================
                  ##### How to use this driver #####
==============================================================================
 [..] Als erstes muss das ROTARY module initialisiert werden, welches auf
    zwei unterschiedliche Arten erfolgen kann.

  (#) void MY_ROTARY_Init_Strobed(..)
      (++) Mit dieser Initialisierung ist es möglich einzelnen LED's
```

Ein- und Auszuschalten sowie deren Helligkeit einzelnen einzustellen.

C:/Users/reite/STM32CubeIDE/workspace_1.7.0/0_ROTARYBAR/MyLibrary/StdLib/mystdlib.h File Reference

```
#include "stm32l4xx.h"
```

Functions

- void **MY_STDLIB_Delay** (TIM_HandleTypeDef *htim, uint16_t us)
  Initialize the CLIPBOARDx peripheral according to the specified parameters in the GPIO_Init.

Detailed Description

Author

Reiter Roman

Version

1.0

Date

Created on: 24.12.2021

Index

INDEX