

Barbary_GPROJECT_M13
SMARTHOME

Generated by Doxygen 1.9.1

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 APP/APP_SMARTHOME_interface.h File Reference	3
2.1.1 Detailed Description	4
2.1.2 Function Documentation	4
2.1.2.1 APP_authority()	4
2.1.2.2 APP_powerGarageDoor()	5
2.1.2.3 APP_powerLeds()	5
2.1.2.4 APP_recvCMD()	5
2.1.2.5 APP_sendCMD()	6
2.1.2.6 APP_useCursorLCD()	6
2.1.2.7 APP_useWriteLCD()	6
2.2 APP/APP_SMARTHOME_program.c File Reference	6
2.2.1 Detailed Description	7
2.2.2 Function Documentation	8
2.2.2.1 APP_authority()	8
2.2.2.2 APP_powerGarageDoor()	8
2.2.2.3 APP_powerLeds()	8
2.2.2.4 APP_recvCMD()	9
2.2.2.5 APP_sendCMD()	9
2.2.2.6 APP_useCursorLCD()	9
2.2.2.7 APP_useWriteLCD()	10
2.3 HAL/BLUETOOTH/HAL_BLUETOOTH_interface.h File Reference	10
2.3.1 Detailed Description	10
2.3.2 Function Documentation	11
2.3.2.1 HAL_establishBluetooth()	11
2.3.2.2 HAL_recvBluetooth()	11
2.4 HAL/BLUETOOTH/HAL_BLUETOOTH_program.c File Reference	11
2.4.1 Detailed Description	12
2.4.2 Function Documentation	12
2.4.2.1 HAL_establishBluetooth()	12
2.4.2.2 HAL_recvBluetooth()	12
2.5 HAL/LCD/HAL_LCD_interface.h File Reference	13
2.5.1 Detailed Description	14
2.5.2 Function Documentation	14
2.5.2.1 HAL_LCD_displayCharacter()	14
2.5.2.2 HAL_LCD_displayString()	14
2.5.2.3 HAL_LCD_init()	15
2.5.2.4 HAL_LCD_putAtLoc()	15
2.6 HAL/LCD/HAL_LCD_private.h File Reference	15

2.6.1 Detailed Description	16
2.7 HAL/LCD/HAL_LCD_program.c File Reference	16
2.7.1 Detailed Description	17
2.7.2 Function Documentation	17
2.7.2.1 HAL_LCD_displayCharacter()	17
2.7.2.2 HAL_LCD_displayString()	17
2.7.2.3 HAL_LCD_init()	18
2.7.2.4 HAL_LCD_putAtLoc()	18
2.8 HAL/LED/HAL_LED_interface.h File Reference	18
2.8.1 Detailed Description	18
2.8.2 Function Documentation	19
2.8.2.1 HAL_LED_power()	19
2.9 HAL/LED/HAL_LED_program.c File Reference	19
2.9.1 Detailed Description	20
2.9.2 Function Documentation	20
2.9.2.1 HAL_LED_power()	20
2.10 LSTD/LSTD_BITMATH.h File Reference	20
2.10.1 Detailed Description	21
2.11 LSTD/LSTD_TYPES.h File Reference	21
2.11.1 Detailed Description	21
2.12 main.c File Reference	22
2.12.1 Detailed Description	22
2.12.2 Function Documentation	23
2.12.2.1 main()	23
2.13 MCAL/GPIO/MCAL_GPIO_interface.h File Reference	23
2.13.1 Detailed Description	24
2.13.2 Function Documentation	24
2.13.2.1 MCAL_GPIO_GetPinState()	24
2.13.2.2 MCAL_GPIO_PinMode()	24
2.13.2.3 MCAL_GPIO_PinState()	25
2.13.2.4 MCAL_GPIO_TogglePin()	25
2.14 MCAL/GPIO/MCAL_GPIO_private.h File Reference	26
2.14.1 Detailed Description	26
2.14.2 Macro Definition Documentation	26
2.14.2.1 MCAL_PORTA	27
2.15 MCAL/GPIO/MCAL_GPIO_program.c File Reference	27
2.15.1 Detailed Description	27
2.15.2 Function Documentation	28
2.15.2.1 MCAL_GPIO_GetPinState()	28
2.15.2.2 MCAL_GPIO_PinMode()	28
2.15.2.3 MCAL_GPIO_PinState()	28
2.15.2.4 MCAL_GPIO_TogglePin()	30

2.16 MCAL/SPI/MCAL_SPI_interface.h File Reference	30
2.16.1 Detailed Description	31
2.16.2 Function Documentation	31
2.16.2.1 MCAL_SPI_init()	31
2.16.2.2 MCAL_SPI_masterSendRecvByte()	32
2.16.2.3 MCAL_SPI_slaveSendRecvByte()	32
2.17 MCAL/SPI/MCAL_SPI_private.h File Reference	32
2.17.1 Detailed Description	32
2.18 MCAL/SPI/MCAL_SPI_program.c File Reference	33
2.18.1 Detailed Description	34
2.18.2 Function Documentation	34
2.18.2.1 MCAL_SPI_init()	34
2.18.2.2 MCAL_SPI_masterSendRecvByte()	34
2.18.2.3 MCAL_SPI_slaveSendRecvByte()	35
2.19 MCAL/UART/MCAL_UART_interface.h File Reference	35
2.19.1 Detailed Description	36
2.19.2 Function Documentation	36
2.19.2.1 MCAL_UART_init()	36
2.19.2.2 MCAL_UART_recvByte()	36
2.19.2.3 MCAL_UART_sendByte()	37
2.19.2.4 MCAL_UART_sendStream()	37
2.20 MCAL/UART/MCAL_UART_private.h File Reference	37
2.20.1 Detailed Description	37
2.21 MCAL/UART/MCAL_UART_program.c File Reference	38
2.21.1 Detailed Description	39
2.21.2 Function Documentation	39
2.21.2.1 MCAL_UART_init()	39
2.21.2.2 MCAL_UART_recvByte()	39
2.21.2.3 MCAL_UART_sendByte()	40
2.21.2.4 MCAL_UART_sendStream()	40
Index	41

Chapter 1

File Index

1.1 File List

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

main.c	Main file for the SMARTHOME Graduation project from AMIT LEARNING'S DIPLOMA (THIS IS THE MASTER FILE)(!FOUND AND BUILT IN DEBUG FOLDER!)	22
APP/APP_SMARTHOME_interface.h	Interfacing file that contains the MACROS and Function prototypes used to make the SMARTHOME DRIVER	3
APP/APP_SMARTHOME_program.c	Program file that contains the implementation of the function prototypes defined in the header files for the SMARTHOME DRIVER	6
HAL/BLUETOOTH/HAL_BLUETOOTH_interface.h	Interfacing file that contains the MACROS and Function prototypes used to make the BLUE-TOOTH DRIVER	10
HAL/BLUETOOTH/HAL_BLUETOOTH_program.c	Program file that contains the implementation of the functions defined in the header file for the BLUETOOTH DRIVER	11
HAL/LCD/HAL_LCD_interface.h	This is the h file that is used for our macros, function prototypes and declaration used in our LCD16x2 Driver. !PLEASE DON'T FORGET TO USE DELAYS!	13
HAL/LCD/HAL_LCD_private.h	This is the h file that is used to store the private variables and declarations of our LCD16x2 Driver. !PLEASE DON'T FORGET TO USE DELAYS!	15
HAL/LCD/HAL_LCD_program.c	HAL_LCD_program.c is the file that contains the implementation for the function prototypes found in the HAL_LCD_interface.h file	16
HAL/LED/HAL_LED_interface.h	Interfacing file that contains macros and function prototypes for the LED DRIVER	18
HAL/LED/HAL_LED_program.c	Program file that contains the implementation for the function prototypes defined in the header file	19
LSTD/LSTD_BITMATH.h	This is a standard library layer file that contains bitmath macros that can come in handy while coding	20
LSTD/LSTD_TYPES.h	This is a standard library layer file that is used to make aliases for the standard data types inorder to make the code more portable and to avoid changes in data type sizes when using different compilers. giving our standard data types new aliases: unsigned char -> u8_t and s8_t. unsigned short int and signed short int -> u16_t and s16_t. unsigned long int and signed long int -> u32_t and s32_t. float -> f32_t. double -> f64_t	21

MCAL/GPIO/ MCAL_GPIO_interface.h	
This .h file contains the interfacing macros, declarations and function prototypes for the GPIO Driver	23
MCAL/GPIO/ MCAL_GPIO_private.h	
This .h file contains the private macros and declarations for the GPIO Driver	26
MCAL/GPIO/ MCAL_GPIO_program.c	
This c file contains the implementation for the function prototypes used in MCAL_GPIO_interface.h	27
MCAL/SPI/ MCAL_SPI_interface.h	
Interfacing file that contains all the needed interfacing macros as well as function prototypes . .	30
MCAL/SPI/ MCAL_SPI_private.h	
Private file that contains the addresses for the registers to be used in the SPI DRIVER	32
MCAL/SPI/ MCAL_SPI_program.c	
Program file that contains the implementation of the function prototypes defined in the header files for the SPI DRIVER	33
MCAL/UART/ MCAL_UART_interface.h	
Interfacing file that contains interfacing macros and function prototypes for the UART driver . .	35
MCAL/UART/ MCAL_UART_private.h	
Private file that contains the addresses of the registers used in the UART driver	37
MCAL/UART/ MCAL_UART_program.c	
Program file that contains the implementation for the function prototypes defined in the header files	38

Chapter 2

File Documentation

2.1 APP/APP_SMARTHOME_interface.h File Reference

Interfacing file that contains the MACROS and Function prototypes used to make the SMARTHOME DRIVER.

Macros

- `#define MASTER_MCU (0)`
- `#define SLAVE_MCU (1)`
- `#define APP_LCD_ROW00 (0x80)`
- `#define APP_LCD_ROW01 (0xC0)`
- `#define APP_LCD_COL00 (0)`
- `#define APP_LCD_COL01 (1)`
- `#define APP_LCD_COL02 (2)`
- `#define APP_LCD_COL03 (3)`
- `#define APP_LCD_COL04 (4)`
- `#define APP_LCD_COL05 (5)`
- `#define APP_LCD_COL06 (6)`
- `#define APP_LCD_COL07 (7)`
- `#define APP_LCD_COL08 (8)`
- `#define APP_LCD_COL09 (9)`
- `#define APP_LCD_COL10 (10)`
- `#define APP_LCD_COL11 (11)`
- `#define APP_LCD_COL12 (12)`
- `#define APP_LCD_COL13 (13)`
- `#define APP_LCD_COL14 (14)`
- `#define APP_LCD_COL15 (15)`
- `#define APP_LED_KITCHEN (1)`
- `#define APP_LED_GARAGE (3)`
- `#define APP_LED_NULL (0)`
- `#define APP_GARAGE_DOOR_OFF (0)`
- `#define APP_GARAGE_DOOR_ON (1)`

Functions

- void **APP_authority** (u8_t au8_Authority)
Used to assign the authority of the calling MCU unit.
- void **APP_init** (void)
Initializes the application (MASTER/SLAVE have different initializations).
- void **APP_recvCMD** (u8_t *pu8_dataHolder)
Receives a command via BLUETOOTH/SPI DEPENDING ON THE AUTHORITY. (MASTER/SLAVE COMMAND)
- void **APP_sendCMD** (u8_t au8_dataSent, u8_t *pu8_dataRecv)
Sends a command via SPI (MASTER ONLY COMMAND)
- void **APP_useWriteLCD** (u8_t *pu8_string)
Used to write strings on the LCD. (SLAVE ONLY COMMAND)
- void **APP_useClearLCD** ()
Used to clear the LCD. (SLAVE ONLY COMMAND)
- void **APP_useCursorLCD** (u8_t au8_row, u8_t au8_col)
Used to put the cursor of the LCD at a specific location. (SLAVE ONLY COMMAND)
- void **APP_powerLeds** (u8_t au8_ledName)
Used to power LED given MACRO NAMES or NUMBERS. (SLAVE ONLY COMMAND)
- void **APP_powerGarageDoor** (u8_t au8_state)
Used to power the garage MOTOR. (SLAVE ONLY COMMAND)

2.1.1 Detailed Description

Interfacing file that contains the MACROS and Function prototypes used to make the SMARTHOME DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

03-04-2021 3:21:54 AM

Copyright

Copyright GPL(c) 2021

2.1.2 Function Documentation

2.1.2.1 APP_authority()

```
void APP_authority (
    u8_t au8_selectAuthority )
```

Used to assign the authority of the calling MCU unit.

Parameters

<i>au8_Authority</i>	the variable containing the Authority of the MCU (!USE THE MACROS!)
----------------------	---

2.1.2.2 APP_powerGarageDoor()

```
void APP_powerGarageDoor (
    u8_t au8_state )
```

Used to power the garage MOTOR. (SLAVE ONLY COMMAND)

Parameters

<i>au8_state</i>	The variable that stores the values to toggle the MOTOR. (!USE THE MACROS!)
------------------	---

2.1.2.3 APP_powerLeds()

```
void APP_powerLeds (
    u8_t au8_ledName )
```

Used to power LED given MACRO NAMES or NUMBERS. (SLAVE ONLY COMMAND)

Parameters

<i>au8_ledName</i>	The variable that stores the values of the LEDS to be powered.
--------------------	--

2.1.2.4 APP_recvCMD()

```
void APP_recvCMD (
    u8_t * pu8_dataHolder )
```

Receives a command via BLUETOOTH/SPI DEPENDING ON THE AUTHORITY. (MASTER/SLAVE COMMAND)

Parameters

<i>pu8_dataHolder</i>	A pointer to the variable that will hold the data received.
-----------------------	---

2.1.2.5 APP_sendCMD()

```
void APP_sendCMD (
    u8_t au8_dataSent,
    u8_t * pu8_dataRecv )
```

Sends a command via SPI (MASTER ONLY COMMAND)

Parameters

<i>au8_dataSent</i>	A variable containing the data to be sent.
<i>pu8_dataRecv</i>	A pointer to a variable for the data to be received.

2.1.2.6 APP_useCursorLCD()

```
void APP_useCursorLCD (
    u8_t au8_row,
    u8_t au8_col )
```

Used to put the cursor of the LCD at a specific location. (SLAVE ONLY COMMAND)

Parameters

<i>au8_row</i>	The variable containing the ROW where we want our cursor to be.
<i>au8_col</i>	The variable containing the COLUMN where we want our cursor to be.

2.1.2.7 APP_useWriteLCD()

```
void APP_useWriteLCD (
    u8_t * pu8_string )
```

Used to write strings on the LCD. (SLAVE ONLY COMMAND)

Parameters

<i>pu8_string</i>	A pointer to a variable that contains the string to be shown on the LCD.
-------------------	--

2.2 APP/APP_SMARTHOME_program.c File Reference

Program file that contains the implementation of the function prototypes defined in the header files for the SMARTHOME DRIVER.

```
#include "LSTD_TYPES.h"
#include "LSTD_BITMATH.h"
#include "MCAL_GPIO_interface.h"
#include "MCAL_UART_interface.h"
#include "MCAL_SPI_interface.h"
#include "HAL_BLUETOOTH_interface.h"
#include "APP_SMARTHOME_interface.h"
#include <util/delay.h>
```

Macros

- `#define F_CPU 16000000UL`

Functions

- void `APP_authority` (u8_t au8_selectAuthority)
Used to assign the authority of the calling MCU unit.
- void `APP_init` (void)
Initializes the application (MASTER/SLAVE have different initializations).
- void `APP_recvCMD` (u8_t *pu8_dataHolder)
Receives a command via BLUETOOTH/SPI DEPENDING ON THE AUTHORITY. (MASTER/SLAVE COMMAND)
- void `APP_sendCMD` (u8_t au8_dataSent, u8_t *pu8_dataRecv)
Sends a command via SPI (MASTER ONLY COMMAND)
- void `APP_useWriteLCD` (u8_t *pu8_string)
Used to write strings on the LCD. (SLAVE ONLY COMMAND)
- void `APP_useClearLCD` ()
Used to clear the LCD. (SLAVE ONLY COMMAND)
- void `APP_useCursorLCD` (u8_t au8_row, u8_t au8_col)
Used to put the cursor of the LCD at a specific location. (SLAVE ONLY COMMAND)
- void `APP_powerLeds` (u8_t au8_ledName)
Used to power LED given MACRO NAMES or NUMBERS. (SLAVE ONLY COMMAND)
- void `APP_powerGarageDoor` (u8_t au8_state)
Used to power the garage MOTOR. (SLAVE ONLY COMMAND)

Variables

- u8_t `gu8_Authority` = 2

2.2.1 Detailed Description

Program file that contains the implementation of the function prototypes defined in the header files for the SMARTHOME DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

03-04-2021 3:22:13 AM

Copyright

Copyright GPL(c) 2021

2.2.2 Function Documentation

2.2.2.1 APP_authority()

```
void APP_authority (
    u8_t au8_selectAuthority )
```

Used to assign the authority of the calling MCU unit.

Parameters

<i>au8_Authority</i>	the variable containing the Authority of the MCU (!USE THE MACROS!)
----------------------	---

2.2.2.2 APP_powerGarageDoor()

```
void APP_powerGarageDoor (
    u8_t au8_state )
```

Used to power the garage MOTOR. (SLAVE ONLY COMMAND)

Parameters

<i>au8_state</i>	The variable that stores the values to toggle the MOTOR. (!USE THE MACROS!)
------------------	---

2.2.2.3 APP_powerLeds()

```
void APP_powerLeds (
    u8_t au8_ledName )
```

Used to power LED given MACRO NAMES or NUMBERS. (SLAVE ONLY COMMAND)

Parameters

<i>au8_ledName</i>	The variable that stores the values of the LEDS to be powered.
--------------------	--

2.2.2.4 APP_recvCMD()

```
void APP_recvCMD (
    u8_t * pu8_dataHolder )
```

Receives a command via BLUETOOTH/SPI DEPENDING ON THE AUTHORITY. (MASTER/SLAVE COMMAND)

Parameters

<i>pu8_dataHolder</i>	A pointer to the variable that will hold the data received.
-----------------------	---

2.2.2.5 APP_sendCMD()

```
void APP_sendCMD (
    u8_t au8_dataSent,
    u8_t * pu8_dataRecv )
```

Sends a command via SPI (MASTER ONLY COMMAND)

Parameters

<i>au8_dataSent</i>	A variable containing the data to be sent.
<i>pu8_dataRecv</i>	A pointer to a variable for the data to be received.

2.2.2.6 APP_useCursorLCD()

```
void APP_useCursorLCD (
    u8_t au8_row,
    u8_t au8_col )
```

Used to put the cursor of the LCD at a specific location. (SLAVE ONLY COMMAND)

Parameters

<i>au8_row</i>	The variable containing the ROW where we want our cursor to be.
<i>au8_col</i>	The variable containing the COLUMN where we want our cursor to be.

2.2.2.7 APP_useWriteLCD()

```
void APP_useWriteLCD (
    u8_t * pu8_string )
```

Used to write strings on the LCD. (SLAVE ONLY COMMAND)

Parameters

<i>pu8_string</i>	A pointer to a variable that contains the string to be shown on the LCD.
-------------------	--

2.3 HAL/BLUETOOTH/HAL_BLUETOOTH_interface.h File Reference

Interfacing file that contains the MACROS and Function prototypes used to make the BLUETOOTH DRIVER.

Macros

- #define **BLUETOOTH_DATA_BAUDRATE** (103)

Functions

- void [HAL_establishBluetooth](#) (u16_t au16_baudRate)
Establishes a Bluetooth connection with a given BAUD RATE. (PLEASE USE THE DEFAULT BAUD-RATE FOR BLUETOOTH!)
- void [HAL_recvBluetooth](#) (u8_t *pu8_dataHolder)
Receive data from an already established BLUETOOTH connection with a given device.

2.3.1 Detailed Description

Interfacing file that contains the MACROS and Function prototypes used to make the BLUETOOTH DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

03-04-2021 2:49:17 AM

Copyright

Copyright GPL(c) 2021

2.3.2 Function Documentation

2.3.2.1 HAL_establishBluetooth()

```
void HAL_establishBluetooth (
    u16_t au16_baudRate )
```

Establishes a Bluetooth connection with a given BAUD RATE. (!PLEASE USE THE DEFAULT BAUD-RATE FOR BLUETOOTH!)

Parameters

<i>au16_baudRate</i>	variable that contains the BAUD RATE to be passed to the function.
----------------------	--

2.3.2.2 HAL_recvBluetooth()

```
void HAL_recvBluetooth (
    u8_t * pu8_dataHolder )
```

Receive data from an already established BLUETOOTH connection with a given device.

Parameters

<i>pu8_dataHolder</i>	pointer to the variable that contains the data received from the BLUETOOTH Transmission.
-----------------------	--

2.4 HAL/BLUETOOTH/HAL_BLUETOOTH_program.c File Reference

Program file that contains the implementation of the functions defined in the header file for the BLUETOOTH DRIVER.

```
#include "LSTD_TYPES.h"
#include "LSTD_BITMATH.h"
#include "MCAL_UART_interface.h"
#include "MCAL_GPIO_interface.h"
#include "HAL_BLUETOOTH_interface.h"
```

Functions

- void [HAL_establishBluetooth](#) (u16_t au16_baudRate)
Establishes a Bluetooth connection with a given BAUD RATE. (!PLEASE USE THE DEFAULT BAUD-RATE FOR BLUETOOTH!)
- void [HAL_recvBluetooth](#) (u8_t *pu8_dataHolder)
Receive data from an already established BLUETOOTH connection with a given device.

2.4.1 Detailed Description

Program file that contains the implementation of the functions defined in the header file for the BLUETOOTH DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

03-04-2021 2:44:53 AM

Copyright

Copyright GPL(c) 2021

2.4.2 Function Documentation

2.4.2.1 HAL_establishBluetooth()

```
void HAL_establishBluetooth (
    u16_t au16_baudRate )
```

Establishes a Bluetooth connection with a given BAUD RATE. (!PLEASE USE THE DEFAULT BAUD-RATE FOR BLUETOOTH!)

Parameters

<i>au16_baudRate</i>	variable that contains the BAUD RATE to be passed to the function.
----------------------	--

2.4.2.2 HAL_recvBluetooth()

```
void HAL_recvBluetooth (
    u8_t * pu8_dataHolder )
```

Receive data from an already established BLUETOOTH connection with a given device.

Parameters

<code>pu8_dataHolder</code>	pointer to the variable that contains the data received from the BLUETOOTH Transmission.
-----------------------------	--

2.5 HAL/LCD/HAL_LCD_interface.h File Reference

This is the h file that is used for our macros, function prototypes and declaration used in our LCD16x2 Driver.
!PLEASE DON'T FORGET TO USE DELAYS!

Macros

- `#define HAL_LCD_ROW00` (0x80)
- `#define HAL_LCD_ROW01` (0xC0)
- `#define HAL_LCD_COL00` (0)
- `#define HAL_LCD_COL01` (1)
- `#define HAL_LCD_COL02` (2)
- `#define HAL_LCD_COL03` (3)
- `#define HAL_LCD_COL04` (4)
- `#define HAL_LCD_COL05` (5)
- `#define HAL_LCD_COL06` (6)
- `#define HAL_LCD_COL07` (7)
- `#define HAL_LCD_COL08` (8)
- `#define HAL_LCD_COL09` (9)
- `#define HAL_LCD_COL10` (10)
- `#define HAL_LCD_COL11` (11)
- `#define HAL_LCD_COL12` (12)
- `#define HAL_LCD_COL13` (13)
- `#define HAL_LCD_COL14` (14)
- `#define HAL_LCD_COL15` (15)

Functions

- void `HAL_LCD_init` (void)
HAL_LCD_init is the LCD16x2 Initializing function.
- void `HAL_LCD_displayCharacter` (u8_t au8_charData)
HAL_LCD_displayCharacter is a function that displays the character passed as a parameter on the LCD16x2.
- void `HAL_LCD_displayString` (u8_t *pu8_srtData)
HAL_LCD_displayString is a function that displays a given string on the LCD16x2 screen (!Be careful of character overflows!)
- void `HAL_LCD_putAtLoc` (u8_t au8_row, u8_t au8_col)
HAL_LCD_putAtLoc is a function that moves the cursor of the LCD16x2 to the DDRAM address passed.
- void `HAL_LCD_clearDisplay` (void)
HAL_LCD_clearDisplay is a function that clears the display of the LCD16x2.

2.5.1 Detailed Description

This is the h file that is used for our macros, function prototypes and declaration used in our LCD16x2 Driver.
!PLEASE DON'T FORGET TO USE DELAYS!

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

29-01-2021 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.5.2 Function Documentation

2.5.2.1 HAL_LCD_displayCharacter()

```
void HAL_LCD_displayCharacter (
    u8_t au8_charData )
```

HAL_LCD_displayCharacter is a function that displays the character passed as a parameter on the LCD16x2.

Parameters

<i>au8_charData</i>	is the data that will be passed to the LCD16x2.
---------------------	---

2.5.2.2 HAL_LCD_displayString()

```
void HAL_LCD_displayString (
    u8_t * pu8_srtData )
```

HAL_LCD_displayString is a function that displays a given string on the LCD16x2 screen (!Be careful of character overflows!)

Parameters

<i>pu8_srtData</i>	is a pointer to the 8 bits of dat that will be displayed.
--------------------	---

2.5.2.3 HAL_LCD_init()

```
void HAL_LCD_init (
    void )
```

HAL_LCD_init is the LCD16x2 Initializing function.

SUPER IMPORTANT!!!

2.5.2.4 HAL_LCD_putAtLoc()

```
void HAL_LCD_putAtLoc (
    u8_t au8_row,
    u8_t au8_col )
```

HAL_LCD_putAtLoc is a function that moves the cursor of the LCD16x2 to the DDRAM address passed.

Parameters

<i>au8_row</i>	is the variable containing the row where we want to move (!DO NOT FORGET TO USE THE MACROS!)
<i>au8_col</i>	is the variable containing the column where we want to move (!DO NOT FORGET TO USE THE MACROS!)

2.6 HAL/LCD/HAL_LCD_private.h File Reference

This is the h file that is used to store the private variables and declarations of our LCD16x2 Driver. !PLEASE DON'T FORGET TO USE DELAYS!

Macros

- #define **HAL_LCD_CTRL_PORT** (PORTB)
- #define **HAL_LCD_RS_PIN** (PIN1)
- #define **HAL_LCD_RW_PIN** (PIN2)
- #define **HAL_LCD_EN_PIN** (PIN3)
- #define **HAL_LCD_DATA_PORT** (PORTA)
- #define **HAL_LCD_D0_PIN** (PIN0)
- #define **HAL_LCD_D1_PIN** (PIN1)
- #define **HAL_LCD_D2_PIN** (PIN2)
- #define **HAL_LCD_D3_PIN** (PIN3)
- #define **HAL_LCD_D4_PIN** (PIN4)
- #define **HAL_LCD_D5_PIN** (PIN5)
- #define **HAL_LCD_D6_PIN** (PIN6)
- #define **HAL_LCD_D7_PIN** (PIN7)

2.6.1 Detailed Description

This is the h file that is used to store the private variables and declarations of our LCD16x2 Driver. !PLEASE DON'T FORGET TO USE DELAYS!

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

29-01-2021 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.7 HAL/LCD/HAL_LCD_program.c File Reference

[HAL_LCD_program.c](#) is the file that contains the implementation for the function prototypes found in the [HAL_LCD_interface.h](#) file.

```
#include "LSTD_BITMATH.h"
#include "LSTD_TYPES.h"
#include "MCAL_GPIO_interface.h"
#include "HAL_LCD_private.h"
#include "HAL_LCD_interface.h"
#include <util/delay.h>
```

Macros

- `#define F_CPU 16000000UL`

Functions

- void [HAL_LCD_init](#) (void)
HAL_LCD_init is the LCD16x2 Initializing function.
- void [HAL_LCD_displayCharacter](#) (u8_t au8_charData)
HAL_LCD_displayCharacter is a function that displays the character passed as a parameter on the LCD16x2.
- void [HAL_LCD_displayString](#) (u8_t *pu8_srtData)
HAL_LCD_displayString is a function that displays a given string on the LCD16x2 screen (!Be careful of character overflows!)
- void [HAL_LCD_putAtLoc](#) (u8_t au8_row, u8_t au8_col)
HAL_LCD_putAtLoc is a function that moves the cursor of the LCD16x2 to the DDRAM address passed.
- void [HAL_LCD_clearDisplay](#) (void)
HAL_LCD_clearDisplay is a function that clears the display of the LCD16x2.

2.7.1 Detailed Description

[HAL_LCD_program.c](#) is the file that contains the implementation for the function prototypes found in the [HAL_LCD_interface.h](#) file.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

25-03-2021 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.7.2 Function Documentation

2.7.2.1 HAL_LCD_displayCharacter()

```
void HAL_LCD_displayCharacter (
    u8_t au8_charData )
```

HAL_LCD_displayCharacter is a function that displays the character passed as a parameter on the LCD16x2.

Parameters

<i>au8_charData</i>	is the data that will be passed to the LCD16x2.
---------------------	---

2.7.2.2 HAL_LCD_displayString()

```
void HAL_LCD_displayString (
    u8_t * pu8_srtData )
```

HAL_LCD_displayString is a function that displays a given string on the LCD16x2 screen (!Be careful of character overflows!)

Parameters

<code>pu8_srtData</code>	is a pointer to the 8 bits of dat that will be displayed.
--------------------------	---

2.7.2.3 HAL_LCD_init()

```
void HAL_LCD_init (
    void )
```

HAL_LCD_init is the LCD16x2 Initializing function.

SUPER IMPORTANT!!!

2.7.2.4 HAL_LCD_putAtLoc()

```
void HAL_LCD_putAtLoc (
    u8_t au8_row,
    u8_t au8_col )
```

HAL_LCD_putAtLoc is a function that moves the cursor of the LCD16x2 to the DDRAM address passed.

Parameters

<code>au8_row</code>	is the variable containing the row where we want to move (!DO NOT FORGET TO USE THE MACROS!)
<code>au8_col</code>	is the variable containing the column where we want to move (!DO NOT FORGET TO USE THE MACROS!)

2.8 HAL/LED/HAL_LED_interface.h File Reference

Interfacing file that contains macros and function prototypes for the LED DRIVER.

Functions

- void [HAL_LED_init](#) (void)
Initializer function that sets up the ports for given leds as OUTPUT using the GPIO DRIVER.
- void [HAL_LED_power](#) (u8_t au8_ledNumber)
Powers up given LEDS/LED MACROS using the GPIO DRIVER.

2.8.1 Detailed Description

Interfacing file that contains macros and function prototypes for the LED DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 3:29:53 PM

Copyright

Copyright GPL(c) 2021

2.8.2 Function Documentation

2.8.2.1 HAL_LED_power()

```
void HAL_LED_power (
    u8_t au8_ledNumber )
```

Powers up given LEDS/LED MACROS using the GPIO DRIVER.

Parameters

<i>au8_ledNumber</i>	variable that contains the number of the leds that will be used.
----------------------	--

2.9 HAL/LED/HAL_LED_program.c File Reference

Program file that contains the implementation for the function prototypes defined in the header file.

```
#include "LSTD_TYPES.h"
#include "MCAL_GPIO_interface.h"
```

Functions

- void [HAL_LED_init](#) (void)
Initializer function that sets up the ports for given leds as OUTPUT using the GPIO DRIVER.
- void [HAL_LED_power](#) (u8_t au8_ledNumber)
Powers up given LEDS/LED MACROS using the GPIO DRIVER.

2.9.1 Detailed Description

Program file that contains the implementation for the function prototypes defined in the header file.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 3:30:09 PM

Copyright

Copyright GPL(c) 2021

2.9.2 Function Documentation

2.9.2.1 HAL_LED_power()

```
void HAL_LED_power (
    u8_t au8_ledNumber )
```

Powers up given LEDS/LED MACROS using the GPIO DRIVER.

Parameters

<i>au8_ledNumber</i>	variable that contains the number of the leds that will be used.
----------------------	--

2.10 LSTD/LSTD_BITMATH.h File Reference

This is a standard library layer file that contains bitmath macros that can come in handy while coding.

Macros

- #define **setBit**(REG, POS) (REG |= (1 << POS))
- #define **clearBit**(REG, POS) (REG &= ~(1 << POS))
- #define **toggleBit**(REG, POS) (REG ^= (1 << POS))
- #define **getBit**(REG, POS) ((REG >> POS) & 1)

2.10.1 Detailed Description

This is a standard library layer file that contains bitmath macros that can come in handy while coding.

Author

Mohamed El Barbary (Mohmbarbary@gmail.com)

Version

1.0

Date

2021-01-29 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.11 LSTD/LSTD_TYPES.h File Reference

This is a standard library layer file that is used to make aliases for the standard data types inorder to make the code more portable and to avoid changes in data type sizes when using different compilers. giving our standard data types new aliases: unsigned char and signed char -> u8_t and s8_t. unsigned short int and signed short int -> u16_t and s16_t. unsigned long int and signed long int -> u32_t and s32_t. float -> f32_t. double -> f64_t.

Typedefs

- typedef unsigned char **u8_t**
- typedef signed char **s8_t**
- typedef unsigned short int **u16_t**
- typedef signed short int **s16_t**
- typedef unsigned long int **u32_t**
- typedef signed long int **s32_t**
- typedef float **f32_t**
- typedef double **f64_t**

2.11.1 Detailed Description

This is a standard library layer file that is used to make aliases for the standard data types inorder to make the code more portable and to avoid changes in data type sizes when using different compilers. giving our standard data types new aliases: unsigned char and signed char -> u8_t and s8_t. unsigned short int and signed short int -> u16_t and s16_t. unsigned long int and signed long int -> u32_t and s32_t. float -> f32_t. double -> f64_t.

Author

Mohamed El Barbary (Mohmbarbary@gmail.com)

Version

1.0

Date

2021-01-29 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.12 main.c File Reference

Main file for the SMARTHOME Graduation project from AMIT LEARNING'S DIPLOMA (THIS IS THE MASTER FILE)(!FOUND AND BUILT IN DEBUG FOLDER!).

```
#include "LSTD_TYPES.h"
#include "APP_SMARTHOME_interface.h"
```

Functions

- int [main](#) (void)
main entry point for the program.

2.12.1 Detailed Description

Main file for the SMARTHOME Graduation project from AMIT LEARNING'S DIPLOMA (THIS IS THE MASTER FILE)(!FOUND AND BUILT IN DEBUG FOLDER!).

Main file for the SMARTHOME Graduation project from AMIT LEARNING'S DIPLOMA (THIS IS THE SLAVE FILE)(!FOUND AND BUILT IN RELEASE FOLDER!).

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

02-04-2021 6:58:46 PM

Copyright

Copyright GPL(c) 2021

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

02-04-2021 6:58:46 PM

Copyright

Copyright (c) 2021

2.12.2 Function Documentation

2.12.2.1 main()

```
int main (
    void )
```

main entry point for the program.

Returns

int (0 on successful completion of the main or anything else for errors).

2.13 MCAL/GPIO/MCAL_GPIO_interface.h File Reference

This .h file contains the interfacing macros, declarations and function prototypes for the GPIO Driver.

Macros

- #define **PORTA** (0)
- #define **PORTB** (1)
- #define **PORTC** (2)
- #define **PORTD** (3)
- #define **PIN0** (0b00000001)

creating macros for the PIN registers, we will be writing them in binary, so that we can do bit operations on them for ease of use.
- #define **PIN1** (0b00000010)
- #define **PIN2** (0b00000100)
- #define **PIN3** (0b00001000)
- #define **PIN4** (0b00010000)
- #define **PIN5** (0b00100000)
- #define **PIN6** (0b01000000)
- #define **PIN7** (0b10000000)
- #define **INPUT_FLOAT** (0)

creating a macro for the data direction types.
- #define **INPUT_PULLUP** (1)
- #define **OUTPUT** (2)
- #define **LOW** (0)

creating a macro for the possible states.
- #define **HIGH** (1)

Functions

- void **MCAL_GPIO_PinMode** (u8_t au8_port, u8_t au8_pin, u8_t au8_type)

MCAL_GPIO_PinMode is used to change the Mode of a pin/pins from any given port.
- void **MCAL_GPIO_PinState** (u8_t au8_port, u8_t au8_pin, u8_t au8_state)

MCAL_GPIO_PinState is used to change the State of a pin/pins from any given port to HIGH or LOW.
- void **MCAL_GPIO_TogglePin** (u8_t au8_port, u8_t au8_pin)

MCAL_GPIO_TogglePin is used to toggle the State of a pin given a port.
- u8_t **MCAL_GPIO_GetPinState** (u8_t au8_port, u8_t au8_pin)

MCAL_GPIO_GetPinState is a function that gets the state of a given PORT and PIN combination.

2.13.1 Detailed Description

This .h file contains the interfacing macros, declarations and function prototypes for the GPIO Driver.

Author

Mohamed El Barbary (Mohmbarbary@gmail.com)

Version

1.0

Date

29-01-2021 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.13.2 Function Documentation

2.13.2.1 MCAL_GPIO_GetPinState()

```
u8_t MCAL_GPIO_GetPinState (
    u8_t au8_port,
    u8_t au8_pin )
```

MCAL_GPIO_GetPinState is a function that gets the state of a given PORT and PIN combination.

Parameters

<i>au8_port</i>	the given PORT from our macros list.
<i>au8_pin</i>	the given PIN from our macros list.

Returns

u8_t returns true if the state is HIGH and false if the state is LOW.

2.13.2.2 MCAL_GPIO_PinMode()

```
void MCAL_GPIO_PinMode (
    u8_t au8_port,
```

```

u8_t au8_pin,
u8_t au8_type )

```

MCAL_GPIO_PinMode is used to change the Mode of a pin/pins from any given port.

Parameters

<i>au8_port</i>	is the port to be selected from our macro list PORTA, PORTB, PORTC or PORTD.
<i>au8_pin</i>	is the port to be selected from our macro list PIN1 ... PIN7.
<i>au8_type</i>	is the mode selected from our macro list INPUT_FLOAT, INPUT_PULLUP or OUTPUT.

We will switch over the au_8port given and once we find it, we will switch over the au8_type and then we set up our mode.

The registers used in order to alter the I/P or O/P modes. MCAL_DDRx, MCAL_PORTx.

2.13.2.3 MCAL_GPIO_PinState()

```

void MCAL_GPIO_PinState (
    u8_t au8_port,
    u8_t au8_pin,
    u8_t au8_state )

```

MCAL_GPIO_PinState is used to change the State of a pin/pins from any given port to HIGH or LOW.

Parameters

<i>au8_port</i>	is the port to be selected from our macro list PORTA, PORTB, PORTC or PORTD.
<i>au8_pin</i>	is the port to be selected from our macro list PIN1 ... PIN7.
<i>au8_state</i>	is the mode selected from our macro list HIGH or LOW.

We will switch over the au_8port given and once we find it, we will switch over the au8_type and then we set up our state.

The registers used in order to alter the I/P or O/P states. MCAL_PORTx.

2.13.2.4 MCAL_GPIO_TogglePin()

```

void MCAL_GPIO_TogglePin (
    u8_t au8_port,
    u8_t au8_pin )

```

MCAL_GPIO_TogglePin is used to toggle the State of a pin given a port.

Parameters

<i>au8_port</i>	The PORT used in the toggling operation.
<i>au8_pin</i>	The PIN to be toggled.

2.14 MCAL/GPIO/MCAL_GPIO_private.h File Reference

This .h file contains the private macros and declarations for the GPIO Driver.

Macros

- #define **MCAL_PORTA** (*(volatile u8_t*)(0x3B))
Header guard for the .h file.
- #define **MCAL_DDRA** (*(volatile u8_t*)(0x3A))
- #define **MCAL_PINA** (*(volatile u8_t*)(0x39))
- #define **MCAL_PORTB** (*(volatile u8_t*)(0x38))
defining the memory mapped addresses for the PORTB, DDRB, PINB Registers.
- #define **MCAL_DDRB** (*(volatile u8_t*)(0x37))
- #define **MCAL_PINB** (*(volatile u8_t*)(0x36))
- #define **MCAL_PORTC** (*(volatile u8_t*)(0x35))
defining the memory mapped addresses for the PORTC, DDRC, PINC Registers.
- #define **MCAL_DDRC** (*(volatile u8_t*)(0x34))
- #define **MCAL_PINC** (*(volatile u8_t*)(0x33))
- #define **MCAL_PORTD** (*(volatile u8_t*)(0x32))
defining the memory mapped addresses for the PORTD, DDRD, PIND Registers.
- #define **MCAL_DDRD** (*(volatile u8_t*)(0x31))
- #define **MCAL_PIND** (*(volatile u8_t*)(0x30))

2.14.1 Detailed Description

This .h file contains the private macros and declarations for the GPIO Driver.

Author

Mohamed El Barbary (Mohmbarbary@gmail.com)

Version

1.0

Date

29-01-2021 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.14.2 Macro Definition Documentation

2.14.2.1 MCAL_PORTA

```
#define MCAL_PORTA (*(volatile u8_t*) (0x3B))
```

Header guard for the .h file.

defining the memory mapped addresses for the PORTA, DDRA, PINA Registers.

2.15 MCAL/GPIO/MCAL_GPIO_program.c File Reference

This c file contains the implementation for the function prototypes used in [MCAL_GPIO_interface.h](#).

```
#include "LSTD_BITMATH.h"
#include "LSTD_TYPES.h"
#include "MCAL_GPIO_private.h"
#include "MCAL_GPIO_interface.h"
```

Functions

- void [MCAL_GPIO_PinMode](#) (u8_t au8_port, u8_t au8_pin, u8_t au8_type)
MCAL_GPIO_PinMode is used to change the Mode of a pin/pins from any given port.
- void [MCAL_GPIO_PinState](#) (u8_t au8_port, u8_t au8_pin, u8_t au8_state)
MCAL_GPIO_PinState is used to change the State of a pin/pins from any given port to HIGH or LOW.
- void [MCAL_GPIO_TogglePin](#) (u8_t au8_port, u8_t au8_pin)
MCAL_GPIO_TogglePin is used to toggle the State of a pin given a port.
- u8_t [MCAL_GPIO_GetPinState](#) (u8_t au8_port, u8_t au8_pin)
MCAL_GPIO_GetPinState is a function that gets the state of a given PORT and PIN combination.

2.15.1 Detailed Description

This c file contains the implementation for the function prototypes used in [MCAL_GPIO_interface.h](#).

Author

Mohamed El Barbary (Mohmbarbary@gmail.com)

Version

1.0

Date

29-01-2021 10:19:20 PM

Copyright

Copyright GPL(c) 2021

2.15.2 Function Documentation

2.15.2.1 MCAL_GPIO_GetPinState()

```
u8_t MCAL_GPIO_GetPinState (
    u8_t au8_port,
    u8_t au8_pin )
```

MCAL_GPIO_GetPinState is a function that gets the state of a given PORT and PIN combination.

Parameters

<i>au8_port</i>	the given PORT from our macros list.
<i>au8_pin</i>	the given PIN from our macros list.

Returns

u8_t returns true if the state is HIGH and false if the state is LOW.

2.15.2.2 MCAL_GPIO_PinMode()

```
void MCAL_GPIO_PinMode (
    u8_t au8_port,
    u8_t au8_pin,
    u8_t au8_type )
```

MCAL_GPIO_PinMode is used to change the Mode of a pin/pins from any given port.

Parameters

<i>au8_port</i>	is the port to be selected from our macro list PORTA, PORTB, PORTC or PORTD.
<i>au8_pin</i>	is the port to be selected from our macro list PIN1 ... PIN7.
<i>au8_type</i>	is the mode selected from our macro list INPUT_FLOAT, INPUT_PULLUP or OUTPUT.

We will switch over the au_8port given and once we find it, we will switch over the au8_type and then we set up our mode.

The registers used in order to alter the I/P or O/P modes. MCAL_DDRx, MCAL_PORTx.

2.15.2.3 MCAL_GPIO_PinState()

```
void MCAL_GPIO_PinState (
    u8_t au8_port,
```



```
u8_t au8_pin,  
u8_t au8_state )
```

MCAL_GPIO_PinState is used to change the State of a pin/pins from any given port to HIGH or LOW.

Parameters

<i>au8_port</i>	is the port to be selected from our macro list PORTA, PORTB, PORTC or PORTD.
<i>au8_pin</i>	is the port to be selected from our macro list PIN1 ... PIN7.
<i>au8_state</i>	is the mode selected from our macro list HIGH or LOW.

We will switch over the au_8port given and once we find it, we will switch over the au8_type and then we set up our state.

The registers used in order to alter the I/P or O/P states. MCAL_PORTx.

2.15.2.4 MCAL_GPIO_TogglePin()

```
void MCAL_GPIO_TogglePin (
    u8_t au8_port,
    u8_t au8_pin )
```

MCAL_GPIO_TogglePin is used to toggle the State of a pin given a port.

Parameters

<i>au8_port</i>	The PORT used in the toggling operation.
<i>au8_pin</i>	The PIN to be toggled.

2.16 MCAL/SPI/MCAL_SPI_interface.h File Reference

Interfacing file that contains all the needed interfacing macros as well as function prototypes.

Macros

- #define **MCAL_SPI_MASTER** (1)
- #define **MCAL_SPI_SLAVE** (0)
- #define **MCAL_SPI_MSB** (0)
- #define **MCAL_SPI_LSB** (1)
- #define **MCAL_SPI_SAMPLE_R_SETUP_F** (0)
- #define **MCAL_SPI_SETUP_R_SAMPLE_F** (1)
- #define **MCAL_SPI_SAMPLE_F_SETUP_R** (2)
- #define **MCAL_SPI_SETUP_F_SAMPLE_R** (3)
- #define **MCAL_SPI_CLK_BY_2** (0)
- #define **MCAL_SPI_CLK_BY_4** (1)
- #define **MCAL_SPI_CLK_BY_8** (2)
- #define **MCAL_SPI_CLK_BY_16** (3)
- #define **MCAL_SPI_CLK_BY_32** (4)
- #define **MCAL_SPI_CLK_BY_64** (5)
- #define **MCAL_SPI_CLK_BY_128** (7)

Functions

- void [MCAL_SPI_init](#) (u8_t au8_spiMode, u8_t au8_dataOutMode, u8_t au8_clockMode, u8_t au8_spiSpeed)
Initializer function for an SPI connection. (!USE THE MACROS!)
- void [MCAL_SPI_masterSendRecvByte](#) (u8_t au8_sentData, u8_t *pu8_recvData)
Sending or Receiving one byte as a MASTER.
- void [MCAL_SPI_slaveSendRecvByte](#) (u8_t au8_sentData, u8_t *pu8_recvData)
Sending or Receiving one byte as a SLAVE.

2.16.1 Detailed Description

Interfacing file that contains all the needed interfacing macros as well as function prototypes.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 8:15:25 PM

Copyright

Copyright GPL(c) 2021

2.16.2 Function Documentation

2.16.2.1 MCAL_SPI_init()

```
void MCAL_SPI_init (
    u8_t au8_spiMode,
    u8_t au8_dataOutMode,
    u8_t au8_clockMode,
    u8_t au8_spiSpeed )
```

Initializer function for an SPI connection. (!USE THE MACROS!)

Parameters

<i>au8_spiMode</i>	MASTER or Slave mode variable.
<i>au8_dataOutMode</i>	MSB or LSB DATA OUT MODES.
<i>au8_clockMode</i>	CLOCK MODES (LEADING AND TRAILING EDGES respectively defined in MACROS).
<i>au8_spiSpeed</i>	connection speed variable.

2.16.2.2 MCAL_SPI_masterSendRecvByte()

```
void MCAL_SPI_masterSendRecvByte (
    u8_t au8_sentData,
    u8_t * pu8_recvData )
```

Sending or Receiving one byte as a MASTER.

Parameters

<i>au8_sentData</i>	data to be sent.
<i>pu8_recvData</i>	data to be received.

2.16.2.3 MCAL_SPI_slaveSendRecvByte()

```
void MCAL_SPI_slaveSendRecvByte (
    u8_t au8_sentData,
    u8_t * pu8_recvData )
```

Sending or Receiving one byte as a SLAVE.

Parameters

<i>au8_sentData</i>	data to be sent.
<i>pu8_recvData</i>	data to be received.

2.17 MCAL/SPI/MCAL_SPI_private.h File Reference

Private file that contains the addresses for the registers to be used in the SPI DRIVER.

Macros

- #define **MCAL_SPI_SPCR** (*(volatile u8_t*)(0x2D))
- #define **MCAL_SPI_SPSR** (*(volatile u8_t*)(0x2E))
- #define **MCAL_SPI_SPDR** (*(volatile u8_t*)(0x2F))

2.17.1 Detailed Description

Private file that contains the addresses for the registers to be used in the SPI DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 8:15:38 PM

Copyright

Copyright GPL(c) 2021

2.18 MCAL/SPI/MCAL_SPI_program.c File Reference

Program file that contains the implementation of the function prototypes defined in the header files for the SPI DRIVER.

```
#include "../LSTD/LSTD_TYPES.h"
#include "../LSTD/LSTD_BITMATH.h"
#include "MCAL_SPI_private.h"
#include "MCAL_SPI_interface.h"
#include <util/delay.h>
```

Macros

- #define **F_CPU** 16000000UL
- #define **TIMEOUT_DELAY** (100)

Functions

- void [MCAL_SPI_init](#) (u8_t au8_spiMode, u8_t au8_dataOutMode, u8_t au8_clockMode, u8_t au8_spiSpeed)
Initializer function for an SPI connection. (!USE THE MACROS!)
- void [MCAL_SPI_masterSendRecvByte](#) (u8_t au8_sentData, u8_t *pu8_recvData)
Sending or Receiving one byte as a MASTER.
- void [MCAL_SPI_slaveSendRecvByte](#) (u8_t au8_sentData, u8_t *pu8_recvData)
Sending or Receiving one byte as a SLAVE.

2.18.1 Detailed Description

Program file that contains the implementation of the function prototypes defined in the header files for the SPI DRIVER.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021

Copyright

Copyright (c) 2021

2.18.2 Function Documentation

2.18.2.1 MCAL_SPI_init()

```
void MCAL_SPI_init (
    u8_t au8_spiMode,
    u8_t au8_dataOutMode,
    u8_t au8_clockMode,
    u8_t au8_spiSpeed )
```

Initializer function for an SPI connection. (!USE THE MACROS!)

Parameters

<i>au8_spiMode</i>	MASTER or Slave mode variable.
<i>au8_dataOutMode</i>	MSB or LSB DATA OUT MODES.
<i>au8_clockMode</i>	CLOCK MODES (LEADING AND TRAILING EDGES respectively defined in MACROS).
<i>au8_spiSpeed</i>	connection speed variable.

2.18.2.2 MCAL_SPI_masterSendRecvByte()

```
void MCAL_SPI_masterSendRecvByte (
    u8_t au8_sentData,
    u8_t * pu8_recvData )
```

Sending or Receiving one byte as a MASTER.

Parameters

<i>au8_sentData</i>	data to be sent.
<i>pu8_recvData</i>	data to be received.

2.18.2.3 MCAL_SPI_slaveSendRecvByte()

```
void MCAL_SPI_slaveSendRecvByte (
    u8_t au8_sentData,
    u8_t * pu8_recvData )
```

Sending or Receiving one byte as a SLAVE.

Parameters

<i>au8_sentData</i>	data to be sent.
<i>pu8_recvData</i>	data to be received.

2.19 MCAL/UART/MCAL_UART_interface.h File Reference

Interfacing file that contains interfacing macros and function prototypes for the UART driver.

Macros

- #define **MCAL_UART_BR_2400** (416)
- #define **MCAL_UART_BR_4800** (207)
- #define **MCAL_UART_BR_9600** (103)
- #define **MCAL_UART_BR_19200** (51)
- #define **MCAL_UART_BR_115200** (8)

Functions

- void [MCAL_UART_init](#) (u16_t au16_baudRate)
Initialize a UART connection with a given BAUD RATE.
- void [MCAL_UART_sendByte](#) (u8_t au8_dataByte)
Send one byte via an already established UART connection.
- void [MCAL_UART_sendStream](#) (u8_t *pu8_dataStream, u8_t au8_dataSize)
Send a stream of data via an already established UART connection.
- void [MCAL_UART_recvByte](#) (u8_t *pu8_dataByte)
Receive one byte via an already established UART connection.

2.19.1 Detailed Description

Interfacing file that contains interfacing macros and function prototypes for the UART driver.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 9:02:39 PM

Copyright

Copyright GPL(c) 2021

2.19.2 Function Documentation

2.19.2.1 MCAL_UART_init()

```
void MCAL_UART_init (
    u16_t au16_baudRate )
```

Initialize a UART connection with a given BAUD RATE.

Parameters

<i>au16_baudRate</i>	the baud rate to be passed for the init function (!Please use the pre-defined MACROS!).
----------------------	---

2.19.2.2 MCAL_UART_recvByte()

```
void MCAL_UART_recvByte (
    u8_t * pu8_dataByte )
```

Receive one byte via an already established UART connection.

Parameters

<i>pu8_dataByte</i>	a pointer to the data to be received.
---------------------	---------------------------------------

2.19.2.3 MCAL_UART_sendByte()

```
void MCAL_UART_sendByte (
    u8_t au8_dataByte )
```

Send one byte via an already established UART connection.

Parameters

<i>au8_dataByte</i>	the data to be sent.
---------------------	----------------------

2.19.2.4 MCAL_UART_sendStream()

```
void MCAL_UART_sendStream (
    u8_t * pu8_dataStream,
    u8_t au8_dataSize )
```

Send a stream of data via an already established UART connection.

Parameters

<i>pu8_dataStream</i>	a pointer to the data to be sent.
<i>au8_dataSize</i>	the size of that data.

2.20 MCAL/UART/MCAL_UART_private.h File Reference

Private file that contains the addresses of the registers used in the UART driver.

Macros

- **#define MCAL_UART_UDR** (*(volatile u8_t*)(0x2C))
- **#define MCAL_UART_UCSRA** (*(volatile u8_t*)(0x2B))
- **#define MCAL_UART_UCSRB** (*(volatile u8_t*)(0x2A))
- **#define MCAL_UART_UCSRC** (*(volatile u8_t*)(0x40))
- **#define MCAL_UART_UBRRH** (*(volatile u8_t*)(0x40))
- **#define MCAL_UART_UBRRL** (*(volatile u8_t*)(0x29))

2.20.1 Detailed Description

Private file that contains the addresses of the registers used in the UART driver.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 9:01:59 PM

Copyright

Copyright GPL(c) 2021

2.21 MCAL/UART/MCAL_UART_program.c File Reference

Program file that contains the implementation for the function prototypes defined in the header files.

```
#include "../LSTD/LSTD_BITMATH.h"
#include "../LSTD/LSTD_TYPES.h"
#include "MCAL_UART_private.h"
#include "MCAL_UART_interface.h"
#include <util/delay.h>
```

Macros

- #define **F_CPU** 16000000UL
- #define **TIMEOUT_DELAY** (10)

Functions

- void [MCAL_UART_init](#) (u16_t au16_baudRate)
Initialize a UART connection with a given BAUD RATE.
- void [MCAL_UART_sendByte](#) (u8_t au8_dataByte)
Send one byte via an already established UART connection.
- void [MCAL_UART_sendStream](#) (u8_t *pu8_dataStream, u8_t au8_dataSize)
Send a stream of data via an already established UART connection.
- void [MCAL_UART_recvByte](#) (u8_t *pu8_dataByte)
Receive one byte via an already established UART connection.

2.21.1 Detailed Description

Program file that contains the implementation for the function prototypes defined in the header files.

Author

Mohamed El Barbary (mohmbarbary@gmail.com)

Version

1.0

Date

05-04-2021 9:03:12 PM

Copyright

Copyright GPL(c) 2021

2.21.2 Function Documentation

2.21.2.1 MCAL_UART_init()

```
void MCAL_UART_init (
    u16_t au16_baudRate )
```

Initialize a UART connection with a given BAUD RATE.

Parameters

<i>au16_baudRate</i>	the baud rate to be passed for the init function (!Please use the pre-defined MACROS!).
----------------------	---

2.21.2.2 MCAL_UART_recvByte()

```
void MCAL_UART_recvByte (
    u8_t * pu8_dataByte )
```

Receive one byte via an already established UART connection.

Parameters

<i>pu8_dataByte</i>	a pointer to the data to be received.
---------------------	---------------------------------------

2.21.2.3 MCAL_UART_sendByte()

```
void MCAL_UART_sendByte (
    u8_t au8_dataByte )
```

Send one byte via an already established UART connection.

Parameters

<i>au8_dataByte</i>	the data to be sent.
---------------------	----------------------

2.21.2.4 MCAL_UART_sendStream()

```
void MCAL_UART_sendStream (
    u8_t * pu8_dataStream,
    u8_t au8_dataSize )
```

Send a stream of data via an already established UART connection.

Parameters

<i>pu8_dataStream</i>	a pointer to the data to be sent.
<i>au8_dataSize</i>	the size of that data.

Index

APP/APP_SMARTHOME_interface.h, [3](#)
APP/APP_SMARTHOME_program.c, [6](#)
APP_authority
 APP_SMARTHOME_interface.h, [4](#)
 APP_SMARTHOME_program.c, [8](#)
APP_powerGarageDoor
 APP_SMARTHOME_interface.h, [5](#)
 APP_SMARTHOME_program.c, [8](#)
APP_powerLeds
 APP_SMARTHOME_interface.h, [5](#)
 APP_SMARTHOME_program.c, [8](#)
APP_recvCMD
 APP_SMARTHOME_interface.h, [5](#)
 APP_SMARTHOME_program.c, [9](#)
APP_sendCMD
 APP_SMARTHOME_interface.h, [5](#)
 APP_SMARTHOME_program.c, [9](#)
APP_SMARTHOME_interface.h
 APP_authority, [4](#)
 APP_powerGarageDoor, [5](#)
 APP_powerLeds, [5](#)
 APP_recvCMD, [5](#)
 APP_sendCMD, [5](#)
 APP_useCursorLCD, [6](#)
 APP_useWriteLCD, [6](#)
APP_SMARTHOME_program.c
 APP_authority, [8](#)
 APP_powerGarageDoor, [8](#)
 APP_powerLeds, [8](#)
 APP_recvCMD, [9](#)
 APP_sendCMD, [9](#)
 APP_useCursorLCD, [9](#)
 APP_useWriteLCD, [10](#)
APP_useCursorLCD
 APP_SMARTHOME_interface.h, [6](#)
 APP_SMARTHOME_program.c, [9](#)
APP_useWriteLCD
 APP_SMARTHOME_interface.h, [6](#)
 APP_SMARTHOME_program.c, [10](#)

HAL/BLUETOOTH/HAL_BLUETOOTH_interface.h, [10](#)
HAL/BLUETOOTH/HAL_BLUETOOTH_program.c, [11](#)
HAL/LCD/HAL_LCD_interface.h, [13](#)
HAL/LCD/HAL_LCD_private.h, [15](#)
HAL/LCD/HAL_LCD_program.c, [16](#)
HAL/LED/HAL_LED_interface.h, [18](#)
HAL/LED/HAL_LED_program.c, [19](#)
HAL_BLUETOOTH_interface.h
 HAL_establishBluetooth, [11](#)
 HAL_recvBluetooth, [11](#)
HAL_BLUETOOTH_program.c
 HAL_establishBluetooth, [12](#)
 HAL_recvBluetooth, [12](#)
HAL_establishBluetooth
 HAL_BLUETOOTH_interface.h, [11](#)
 HAL_BLUETOOTH_program.c, [12](#)
HAL_LCD_displayCharacter
 HAL_LCD_interface.h, [14](#)
 HAL_LCD_program.c, [17](#)
HAL_LCD_displayString
 HAL_LCD_interface.h, [14](#)
 HAL_LCD_program.c, [17](#)
HAL_LCD_init
 HAL_LCD_interface.h, [15](#)
 HAL_LCD_program.c, [18](#)
HAL_LCD_interface.h
 HAL_LCD_displayCharacter, [14](#)
 HAL_LCD_displayString, [14](#)
 HAL_LCD_init, [15](#)
 HAL_LCD_putAtLoc, [15](#)
HAL_LCD_program.c
 HAL_LCD_displayCharacter, [17](#)
 HAL_LCD_displayString, [17](#)
 HAL_LCD_init, [18](#)
 HAL_LCD_putAtLoc, [18](#)
HAL_LCD_putAtLoc
 HAL_LCD_interface.h, [15](#)
 HAL_LCD_program.c, [18](#)
HAL_LED_interface.h
 HAL_LED_power, [19](#)
HAL_LED_power
 HAL_LED_interface.h, [19](#)
 HAL_LED_program.c, [20](#)
HAL_LED_program.c
 HAL_LED_power, [20](#)
HAL_recvBluetooth
 HAL_BLUETOOTH_interface.h, [11](#)
 HAL_BLUETOOTH_program.c, [12](#)

LSTD/LSTD_BITMATH.h, [20](#)
LSTD/LSTD_TYPES.h, [21](#)

main
 main.c, [23](#)
main.c, [22](#)
 main, [23](#)
MCAL/GPIO/MCAL_GPIO_interface.h, [23](#)
MCAL/GPIO/MCAL_GPIO_private.h, [26](#)
MCAL/GPIO/MCAL_GPIO_program.c, [27](#)
MCAL/SPI/MCAL_SPI_interface.h, [30](#)

MCAL/SPI/MCAL_SPI_private.h, 32
 MCAL/SPI/MCAL_SPI_program.c, 33
 MCAL/UART/MCAL_UART_interface.h, 35
 MCAL/UART/MCAL_UART_private.h, 37
 MCAL/UART/MCAL_UART_program.c, 38
 MCAL_GPIO_GetPinState
 MCAL_GPIO_interface.h, 24
 MCAL_GPIO_program.c, 28
 MCAL_GPIO_interface.h
 MCAL_GPIO_GetPinState, 24
 MCAL_GPIO_PinMode, 24
 MCAL_GPIO_PinState, 25
 MCAL_GPIO_TogglePin, 25
 MCAL_GPIO_PinMode
 MCAL_GPIO_interface.h, 24
 MCAL_GPIO_program.c, 28
 MCAL_GPIO_PinState
 MCAL_GPIO_interface.h, 25
 MCAL_GPIO_program.c, 28
 MCAL_GPIO_private.h
 MCAL_PORTA, 26
 MCAL_GPIO_program.c
 MCAL_GPIO_GetPinState, 28
 MCAL_GPIO_PinMode, 28
 MCAL_GPIO_PinState, 28
 MCAL_GPIO_TogglePin, 30
 MCAL_GPIO_TogglePin
 MCAL_GPIO_interface.h, 25
 MCAL_GPIO_program.c, 30
 MCAL_PORTA
 MCAL_GPIO_private.h, 26
 MCAL_SPI_init
 MCAL_SPI_interface.h, 31
 MCAL_SPI_program.c, 34
 MCAL_SPI_interface.h
 MCAL_SPI_init, 31
 MCAL_SPI_masterSendRecvByte, 32
 MCAL_SPI_slaveSendRecvByte, 32
 MCAL_SPI_masterSendRecvByte
 MCAL_SPI_interface.h, 32
 MCAL_SPI_program.c, 34
 MCAL_SPI_program.c
 MCAL_SPI_init, 34
 MCAL_SPI_masterSendRecvByte, 34
 MCAL_SPI_slaveSendRecvByte, 35
 MCAL_SPI_slaveSendRecvByte
 MCAL_SPI_interface.h, 32
 MCAL_SPI_program.c, 35
 MCAL_UART_init
 MCAL_UART_interface.h, 36
 MCAL_UART_program.c, 39
 MCAL_UART_interface.h
 MCAL_UART_init, 36
 MCAL_UART_recvByte, 36
 MCAL_UART_sendByte, 37
 MCAL_UART_sendStream, 37
 MCAL_UART_program.c
 MCAL_UART_init, 39
 MCAL_UART_recvByte, 39
 MCAL_UART_sendByte, 40
 MCAL_UART_sendStream, 40
 MCAL_UART_recvByte
 MCAL_UART_interface.h, 36
 MCAL_UART_program.c, 39
 MCAL_UART_sendByte
 MCAL_UART_interface.h, 37
 MCAL_UART_program.c, 40
 MCAL_UART_sendStream
 MCAL_UART_interface.h, 37
 MCAL_UART_program.c, 40