intLib

Generated by Doxygen 1.8.6

Tue Apr 8 2014 11:02:20

Contents

Chapter 1

uIntPLib

Universal Integrated Peripheral Library

This is a library made with functions masks to medium level programming. Intended to make code more portable, while maintaning its performance.

Doxyen generated documentation is located at latex/refman.pdf Complete documentation is under construction.

Chapter 2

uIntPLib

Universal Integrated Peripheral Library

This is a library made with functions masks to medium level programming. Intended to make code more portable, while maintaning its performance.

Doxyen generated documentation is located at latex/refman.pdf Complete documentation is under construction.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Data Structure Documentation

5.1 CommandInstance Struct Reference

`#include <cmd_sort.h>`

Data Fields

- uint8_t charIn
- uint8_t cmdBuffer [MAX_BUFFER_SIZE]
- uint16_t charOut [MAX_BUFFER_SIZE]
- uint8_t charOutPtr

5.1.1 Detailed Description

Definition at line 15 of file cmd_sort.h.

5.1.2 Field Documentation

5.1.2.1 uint8_t CommandInstance::charIn

Definition at line 16 of file cmd_sort.h.

5.1.2.2 uint16_t CommandInstance::charOut[MAX_BUFFER_SIZE]

Definition at line 18 of file cmd_sort.h.

5.1.2.3 uint8_t CommandInstance::charOutPtr

Definition at line 19 of file cmd_sort.h.

5.1.2.4 uint8_t CommandInstance::cmdBuffer[MAX_BUFFER_SIZE]

Definition at line 17 of file cmd_sort.h.

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

- my_lib/cmd_sort.h

5.2 IRInstance Struct Reference

`#include <ir.h>`

Data Fields

- uint16_t Mode
- uint8_t CarrierFrequency
- uint16_t CarrierPeriod
- uint32_t TxPin
- uint32_t TxPort
- uint32_t RxPin
- uint32_t RxPort
- uint16_t ReceiveAddress
- uint16_t ReceiveBuffer
- uint16_t Pulses
- uint8_t LastData

5.2.1 Detailed Description

Definition at line 83 of file ir.h.

5.2.2 Field Documentation

5.2.2.1 uint8_t IRInstance::CarrierFrequency

Definition at line 85 of file ir.h.

5.2.2.2 uint16_t IRInstance::CarrierPeriod

Definition at line 86 of file ir.h.

5.2.2.3 uint8_t IRInstance::LastData

Definition at line 94 of file ir.h.

5.2.2.4 uint16_t IRInstance::Mode

Definition at line 84 of file ir.h.

5.2.2.5 uint16_t IRInstance::Pulses

Definition at line 93 of file ir.h.

5.2.2.6 uint16_t IRInstance::ReceiveAddress

Definition at line 91 of file ir.h.

5.2.2.7 uint16_t IRInstance::ReceiveBuffer

Definition at line 92 of file ir.h.

5.2.2.8 uint32_t IRInstance::RxPin

Definition at line 89 of file ir.h.

5.2.2.9 uint32_t IRInstance::RxPort

Definition at line 90 of file ir.h.

5.2.2.10 uint32_t IRInstance::TxPin

Definition at line 87 of file ir.h.

5.2.2.11 uint32_t IRInstance::TxPort

Definition at line 88 of file ir.h.

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

- my_lib/ir.h

5.3 LCDStatus Struct Reference

`#include <lcd.h>`

Data Fields

- uint8_t row
- uint8_t col
- uint8_t display
- uint8_t shift
- uint8_t cgramAdress
- uint8_t specialChar [8]

5.3.1 Detailed Description

Definition at line 118 of file lcd.h.

5.3.2 Field Documentation

5.3.2.1 uint8_t LCDStatus::cgramAdress

Definition at line 124 of file lcd.h.

5.3.2.2 uint8_t LCDStatus::col

Definition at line 121 of file lcd.h.

5.3.2.3 uint8_t LCDStatus::display

Definition at line 122 of file lcd.h.

5.3.2.4 uint8_t LCDStatus::row

Definition at line 120 of file lcd.h.

5.3.2.5 uint8_t LCDStatus::shift

Definition at line 123 of file lcd.h.

5.3.2.6 uint8_t LCDStatus::specialChar[8]

Definition at line 125 of file lcd.h.

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

- my_lib/lcd.h

5.4 UARTInstance Struct Reference

`#include <myUart.h>`

Data Fields

- uint8_t RxBuffer [UART_BUFFER_SIZE]
- uint8_t RxBufferPtr
- uint8_t TxBuffer [UART_BUFFER_SIZE]
- uint8_t TxBufferPtr
- uint16_t Mode
- uint8_t TxLastSent [UART_BUFFER_SIZE]
- uint8_t TxLastSentPtr

5.4.1 Detailed Description

Definition at line 23 of file myUart.h.

5.4.2 Field Documentation

5.4.2.1 uint16_t UARTInstance::Mode

Definition at line 28 of file myUart.h.

5.4.2.2 uint8_t UARTInstance::RxBuffer[UART_BUFFER_SIZE]

Definition at line 24 of file myUart.h.

5.4.2.3 uint8_t UARTInstance::RxBufferPtr

Definition at line 25 of file myUart.h.

5.4.2.4 uint8_t UARTInstance::TxBuffer[UART_BUFFER_SIZE]

Definition at line 26 of file myUart.h.

5.4.2.5 uint8_t UARTInstance::TxBufferPtr

Definition at line 27 of file myUart.h.

5.4.2.6 uint8_t UARTInstance::TxLastSent[UART_BUFFER_SIZE]

Definition at line 29 of file myUart.h.

5.4.2.7 uint8_t UARTInstance::TxLastSentPtr

Definition at line 30 of file myUart.h.

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

- my_lib/myUart.h

Chapter 6

File Documentation

6.1 depl_spc/chip_specific.c File Reference

```
#include "chip_specific.h"
```

6.2 chip_specific.c

```
00001 /*
00002  * chip_specific.c
00003  *
00004  *  Created on: Mar 25, 2014
00005  *      Author: rikardo
00006  */
00007
00008
00009 #include "chip_specific.h"
00010
00011
00012
00013
00014
```

6.3 depl_spc/chip_specific.h File Reference

```
#include "includeAll_sw.h"
#include "includeAll_hw.h"
```

6.4 chip_specific.h

```
00001 /*
00002  * chip_specific.h
00003  *
00004  *  Created on: Mar 25, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef CHIP_SPECIFIC_H_
00009 #define CHIP_SPECIFIC_H_
00010
00011 #include "includeAll_sw.h"
00012 #include "includeAll_hw.h"
00013
00014
00015
00016 #endif /* CHIP_SPECIFIC_H_ */
```

6.5 depl_spc/cmd_list.h File Reference

6.6 cmd_list.h

```
00001 /*
00002  * cmd_list.h
00003  *
00004  *  Created on: Mar 25, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef CMD_LIST_H_
00009 #define CMD_LIST_H_
00010
00011
00012
00013 #endif /* CMD_LIST_H_ */
```

6.7 depl_spc/device_init/hardwareInit.c File Reference

```
#include "hardwareInit.h"
```

Functions

- void HardwareInit (void)

6.7.1 Function Documentation

6.7.1.1 void HardwareInit (void)

Definition at line 13 of file hardwareInit.c.

6.8 hardwareInit.c

```
00001 /*
00002  * hardwareInit.c
00003  *
00004  *  Created on: Feb 5, 2014
00005  *      Author: rikardo
00006  */
00007
00008
00009
00010 #include "hardwareInit.h"
00011
00012
00013 void HardwareInit(void)
00014 {
00015
00016 }
00017
00018
00019
```

6.9 depl_spc/device_init/hardwareInit.h File Reference

```
#include "depl_spc/includeAll_hw.h"
```

Functions

- void HardwareInit (void)

6.9.1 Function Documentation

6.9.1.1 void HardwareInit (void)

Definition at line 13 of file hardwareInit.c.

6.10 hardwareInit.h

```
00001 /*
00002  * hardwareInit.h
00003  *
00004  *  Created on: Feb 5, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef HARDWAREINIT_H_
00009 #define HARDWAREINIT_H_
00010
00011 #include "depl_spc/includeAll_hw.h"
00012
00013
00014
00015
00016
00017
00018
00019 void HardwareInit(void);
00020
00021 #endif /* HARDWAREINIT_H_ */
```

6.11 depl_spc/device_init/softwareInit.c File Reference

```
#include "softwareInit.h"
```

Functions

- void SoftwareInit (void)

6.11.1 Function Documentation

6.11.1.1 void SoftwareInit (void)

Definition at line 12 of file softwareInit.c.

6.12 softwareInit.c

```
00001 /*
00002  * softwareInit.c
00003  *
00004  *  Created on: Feb 5, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #include "softwareInit.h"
00009
00010
```

```
00011
00012 void SoftwareInit(void)
00013 {
00014
00015 }
00016
```

6.13 depl_spc/device_init/softwareInit.h File Reference

```
#include "depl_spc/includeAll_sw.h"
```

Functions

- void SoftwareInit (void)

6.13.1 Function Documentation

6.13.1.1 void SoftwareInit (void)

Definition at line 12 of file softwareInit.c.

6.14 softwareInit.h

```
00001 /*
00002  * softwareInit.h
00003  *
00004  *  Created on: Feb 5, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef SOFTWAREINIT_H_
00009 #define SOFTWAREINIT_H_
00010
00011 #include "depl_spc/includeAll_sw.h"
00012
00013
00014
00015
00016
00017
00018 void SoftwareInit(void);
00019
00020 #endif /* SOFTWAREINIT_H_ */
```

6.15 depl_spc/globalParam.h File Reference

Macros

- #define PROJECT_NAME ("your projects name here")
- #define LCD_SPLASHSCREEN1 1
- #define LCD_SPLASHSCREEN 1
- #define CPU_CLOCK 48
- #define BUS_CLOCK CPU_CLOCK/2
- #define CPUHZ_CLOCK 48000000
- #define BUSHZ_CLOCK CPUHZ_CLOCK/2

6.15.1 Macro Definition Documentation

6.15.1.1 #define BUS_CLOCK CPU_CLOCK/2

Definition at line 19 of file globalParam.h.

6.15.1.2 #define BUSHZ_CLOCK CPUHZ_CLOCK/2

Definition at line 21 of file globalParam.h.

6.15.1.3 #define CPU_CLOCK 48

Definition at line 18 of file globalParam.h.

6.15.1.4 #define CPUHZ_CLOCK 48000000

Definition at line 20 of file globalParam.h.

6.15.1.5 #define LCD_SPLASHSCREEN 1

Definition at line 15 of file globalParam.h.

6.15.1.6 #define LCD_SPLASHSCREEN1 1

Definition at line 14 of file globalParam.h.

6.15.1.7 #define PROJECT_NAME ("your projects name here")

Definition at line 13 of file globalParam.h.

6.16 globalParam.h

```
00001 /*
00002  * globalParam.h
00003  *
00004  *  Created on: Mar 26, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef GLOBALPARAM_H_
00009 #define GLOBALPARAM_H_
00010
00011
00012
00013 #define PROJECT_NAME        ("your projects name here")
00014 #define LCD_SPLASHSCREEN1   1   //enables proejct name in 2 secs splash
00015 #define LCD_SPLASHSCREEN    1   //enables date and time of compilation
00016
00017
00018 #define CPU_CLOCK   48
00019 #define BUS_CLOCK   CPU_CLOCK/2
00020 #define CPUHZ_CLOCK 48000000
00021 #define BUSHZ_CLOCK CPUHZ_CLOCK/2
00022
00023
00024
00025
00026
00027 #endif /* GLOBALPARAM_H_ */
```

6.17 depl_spc/includeAll_hw.h File Reference

```
#include "globalParam.h"
#include "depl_spc/device_init/hardwareInit.h"
#include "depl_spc/lib_comp/external_cons.h"
#include "chip_specific.h"
```

6.18 includeAll_hw.h

```
00001 /*
00002  * includeAll_hw.h
00003  *
00004  *  Created on: Feb 5, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef INCLUDEALL_HW_H_
00009 #define INCLUDEALL_HW_H_
00010
00011 //program definitions
00012 #include "globalParam.h"
00013
00014
00015 //masks for chip
00016
00017 //functions for peripherals
00018
00019
00020
00021 #include "depl_spc/device_init/hardwareInit.h"
00022 #include "depl_spc/lib_comp/external_cons.h"
00023 #include "chip_specific.h"
00024
00025
00026 #endif /* INCLUDEALL_HW_H_ */
```

6.19 depl_spc/includeAll_sw.h File Reference

```
#include "stdint.h"
#include "stdbool.h"
#include "depl_spc/lib_comp/libraryCompatible.h"
#include "my_use.h"
#include "lcd.h"
#include "depl_spc/device_init/softwareInit.h"
#include "depl_spc/device_init/hardwareInit.h"
```

6.20 includeAll_sw.h

```
00001 /*
00002  * includeAll_sw.h
00003  *
00004  *  Created on: Mar 25, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef INCLUDEALL_SW_H_
00009 #define INCLUDEALL_SW_H_
00010
00011 #include "stdint.h"
00012 #include "stdbool.h"
00013
00014 //basic low level functions
00015 #include "depl_spc/lib_comp/libraryCompatible.h"
00016 #include "my_use.h"
00017
```

```
00018 //external peripherals
00019 #include "lcd.h"
00020
00021
00022
00023 #include "depl_spc/device_init/softwareInit.h"
00024 #include "depl_spc/device_init/hardwareInit.h"
00025
00026 #endif /* INCLUDEALL_SW_H_ */
```

6.21 depl_spc/lib_comp/external_cons.h File Reference

Macros

- #define LCD_RS_Port PTE_BASE_PTR
- #define LCD_RS_Pin IOPin_30
- #define LCD_EN_Port PTE_BASE_PTR
- #define LCD_EN_Pin IOPin_29
- #define LCD_DTA_Port PTE_BASE_PTR
- #define LCD_DTA_Pin IOPin_22
- #define LCD_CLK_Port PTE_BASE_PTR
- #define LCD_CLK_Pin IOPin_23
- #define LCD_row_num 2
- #define LCD_col_num 16
- #define LCD_char_heigh 8
- #define LCD_char_width 5

6.21.1 Macro Definition Documentation

6.21.1.1 #define LCD_char_heigh 8

Definition at line 29 of file external_cons.h.

6.21.1.2 #define LCD_char_width 5

Definition at line 30 of file external_cons.h.

6.21.1.3 #define LCD_CLK_Pin IOPin_23

Definition at line 25 of file external_cons.h.

6.21.1.4 #define LCD_CLK_Port PTE_BASE_PTR

Definition at line 24 of file external_cons.h.

6.21.1.5 #define LCD_col_num 16

Definition at line 28 of file external_cons.h.

6.21.1.6 #define LCD_DTA_Pin IOPin_22

Definition at line 22 of file external_cons.h.

6.21.1.7 #define LCD_DTA_Port PTE_BASE_PTR

Definition at line 21 of file external_cons.h.

6.21.1.8 #define LCD_EN_Pin IOPin_29

Definition at line 19 of file external_cons.h.

6.21.1.9 #define LCD_EN_Port PTE_BASE_PTR

Definition at line 18 of file external_cons.h.

6.21.1.10 #define LCD_row_num 2

Definition at line 27 of file external_cons.h.

6.21.1.11 #define LCD_RS_Pin IOPin_30

Definition at line 16 of file external_cons.h.

6.21.1.12 #define LCD_RS_Port PTE_BASE_PTR

Definition at line 15 of file external_cons.h.

6.22 external_cons.h

```
00001 #ifndef external_cons_h
00002 #define external_cons_h
00003
00004
00005 /*
00006  * file used to declare masks to external peripherals
00007  *
00008  */
00009
00010
00011 /*
00012  * Definitions for LCD peripheral
00013  */
00014 //LCD
00015 #define LCD_RS_Port     PTE_BASE_PTR
00016 #define LCD_RS_Pin      IOPin_30
00017
00018 #define LCD_EN_Port     PTE_BASE_PTR
00019 #define LCD_EN_Pin      IOPin_29
00020
00021 #define LCD_DTA_Port        PTE_BASE_PTR
00022 #define LCD_DTA_Pin     IOPin_22
00023
00024 #define LCD_CLK_Port        PTE_BASE_PTR
00025 #define LCD_CLK_Pin     IOPin_23
00026
00027 #define LCD_row_num     2
00028 #define LCD_col_num     16
00029 #define LCD_char_heigh  8
00030 #define LCD_char_width  5
00031
00032
00033
00034 #endif//external_cons_h
```

6.23 depl_spc/lib_comp/libraryCompatible.h File Reference

```
#include "depl_spc/includeAll_hw.h"
#include "gpioPin_masks.h"
```

Macros

- #define PinSet(port, pin) (port##_PSOR = pin)
- #define PinClear(port, pin) (port##_PCOR = pin)
- #define PinToogle(port, pin) (port##_PTOR = pin)
- #define PinAddrSet(port, pin) (GPIO_PSOR_REG((GPIO_MemMapPtr)port) = pin)
- #define PinAddrClear(port, pin) (GPIO_PCOR_REG((GPIO_MemMapPtr)port) = pin)
- #define SysDelay(time) SysDelayFRDM(time)
- #define SysDelayUs(time) SysDelay((time∗BUS_CLOCK)/6)
- #define SysDelayMs(time) SysDelayUs(time∗1000)

6.23.1 Macro Definition Documentation

6.23.1.1 #define PinAddrClear(*port, pin*) (GPIO_PCOR_REG((GPIO_MemMapPtr)port) = pin)

Definition at line 22 of file libraryCompatible.h.

6.23.1.2 #define PinAddrSet(*port, pin*) (GPIO_PSOR_REG((GPIO_MemMapPtr)port) = pin)

Definition at line 21 of file libraryCompatible.h.

6.23.1.3 #define PinClear(*port, pin*) (port##_PCOR = pin)

Definition at line 17 of file libraryCompatible.h.

6.23.1.4 #define PinSet(*port, pin*) (port##_PSOR = pin)

Definition at line 16 of file libraryCompatible.h.

6.23.1.5 #define PinToogle(*port, pin*) (port##_PTOR = pin)

Definition at line 18 of file libraryCompatible.h.

6.23.1.6 #define SysDelay(*time*) SysDelayFRDM(time)

Definition at line 26 of file libraryCompatible.h.

6.23.1.7 #define SysDelayMs(*time*) SysDelayUs(time∗1000)

Definition at line 28 of file libraryCompatible.h.

6.23.1.8 #define SysDelayUs(*time*) SysDelay((time∗BUS_CLOCK)/6)

Definition at line 27 of file libraryCompatible.h.

6.24 libraryCompatible.h

```
00001 /*
00002  * libraryCompatible.h
00003  *
00004  *  Created on: Feb 5, 2014
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef LIBRARYCOMPATIBLE_H_
00009 #define LIBRARYCOMPATIBLE_H_
00010
00011 #include "depl_spc/includeAll_hw.h"
00012 #include "gpioPin_masks.h"
00013
00014
00015 //direct setting, uses a pre-casted object
00016 #define PinSet(port, pin)       (port##_PSOR = pin)
00017 #define PinClear(port, pin)     (port##_PCOR = pin)
00018 #define PinToogle(port, pin)    (port##_PTOR = pin)
00019
00020 //casts the address to the structure referenced in the memory mapping file
00021 #define PinAddrSet(port, pin)   (GPIO_PSOR_REG((GPIO_MemMapPtr)port) = pin)
00022 #define PinAddrClear(port, pin) (GPIO_PCOR_REG((GPIO_MemMapPtr)port) = pin)
00023
00024
00025
00026 #define SysDelay(time)          SysDelayFRDM(time)          //chip specific
00027 #define SysDelayUs(time)        SysDelay((time*BUS_CLOCK)/6)
00028 #define SysDelayMs(time)        SysDelayUs(time*1000)
00029
00030
00031
00032
00033
00034 #endif /* LIBRARYCOMPATIBLE_H_ */
```

6.25 depl_spc/variables.h File Reference

Macros

- #define variable_h

6.25.1 Macro Definition Documentation

6.25.1.1 #define variable_h

Definition at line 2 of file variables.h.

6.26 variables.h

```
00001 #ifndef variables_h
00002 #define variable_h
00003
00004
00005
00006 #endif
```

6.27 my_lib/ascii.h File Reference

Macros

- #define ASCII_NULL 0
- #define ASCII_SOH 1
- #define ASCII_STX 2

- #define ASCII_ETX 3
- #define ASCII_EOT 4
- #define ASCII_ENQ 5
- #define ASCII_ACK 6
- #define ASCII_BEL 7
- #define ASCII_BS 8
- #define ASCII_HT 9
- #define ASCII_LF 10
- #define ASCII_VT 11
- #define ASCII_FF 12
- #define ASCII_CR 13
- #define ASCII_SO 14
- #define ASCII_SI 15
- #define ASCII_DLE 16
- #define ASCII_DC1 17
- #define ASCII_DC2 18
- #define ASCII_DC3 19
- #define ASCII_DC4 20
- #define ASCII_NAK 21
- #define ASCII_SYN 22
- #define ASCII_ETB 23
- #define ASCII_CAN 24
- #define ASCII_EM 25
- #define ASCII_SUB 26
- #define ASCII_ESC 27
- #define ASCII_FS 28
- #define ASCII_GS 29
- #define ASCII_RS 30
- #define ASCII_US 31

6.27.1 Macro Definition Documentation

6.27.1.1 #define ASCII_ACK 6

Definition at line 22 of file ascii.h.

6.27.1.2 #define ASCII_BEL 7

Definition at line 23 of file ascii.h.

6.27.1.3 #define ASCII_BS 8

Definition at line 24 of file ascii.h.

6.27.1.4 #define ASCII_CAN 24

Definition at line 40 of file ascii.h.

6.27.1.5 #define ASCII_CR 13

Definition at line 29 of file ascii.h.

6.27.1.6 #define ASCII_DC1 17

Definition at line 33 of file ascii.h.

6.27.1.7 #define ASCII_DC2 18

Definition at line 34 of file ascii.h.

6.27.1.8 #define ASCII_DC3 19

Definition at line 35 of file ascii.h.

6.27.1.9 #define ASCII_DC4 20

Definition at line 36 of file ascii.h.

6.27.1.10 #define ASCII_DLE 16

Definition at line 32 of file ascii.h.

6.27.1.11 #define ASCII_EM 25

Definition at line 41 of file ascii.h.

6.27.1.12 #define ASCII_ENQ 5

Definition at line 21 of file ascii.h.

6.27.1.13 #define ASCII_EOT 4

Definition at line 20 of file ascii.h.

6.27.1.14 #define ASCII_ESC 27

Definition at line 43 of file ascii.h.

6.27.1.15 #define ASCII_ETB 23

Definition at line 39 of file ascii.h.

6.27.1.16 #define ASCII_ETX 3

Definition at line 19 of file ascii.h.

6.27.1.17 #define ASCII_FF 12

Definition at line 28 of file ascii.h.

6.27.1.18 #define ASCII_FS 28

Definition at line 44 of file ascii.h.

6.27.1.19 #define ASCII_GS 29

Definition at line 45 of file ascii.h.

6.27.1.20 #define ASCII_HT 9

Definition at line 25 of file ascii.h.

6.27.1.21 #define ASCII_LF 10

Definition at line 26 of file ascii.h.

6.27.1.22 #define ASCII_NAK 21

Definition at line 37 of file ascii.h.

6.27.1.23 #define ASCII_NULL 0

Definition at line 16 of file ascii.h.

6.27.1.24 #define ASCII_RS 30

Definition at line 46 of file ascii.h.

6.27.1.25 #define ASCII_SI 15

Definition at line 31 of file ascii.h.

6.27.1.26 #define ASCII_SO 14

Definition at line 30 of file ascii.h.

6.27.1.27 #define ASCII_SOH 1

Definition at line 17 of file ascii.h.

6.27.1.28 #define ASCII_STX 2

Definition at line 18 of file ascii.h.

6.27.1.29 #define ASCII_SUB 26

Definition at line 42 of file ascii.h.

6.27.1.30 #define ASCII_SYN 22

Definition at line 38 of file ascii.h.

6.27.1.31 #define ASCII_US 31

Definition at line 47 of file ascii.h.

6.27.1.32 #define ASCII_VT 11

Definition at line 27 of file ascii.h.

6.28 ascii.h

```
00001 /*
00002  * ascii.h
00003  *
00004  *  Created on: Nov 25, 2013
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef ASCII_H_
00009 #define ASCII_H_
00010
00011 /*
00012  * File contaning ASCII Masks
00013  */
00014
00015
00016 #define ASCII_NULL          0           //Null Char
00017 #define ASCII_SOH           1           //Start of Header
00018 #define ASCII_STX           2           //Start of Text
00019 #define ASCII_ETX           3           //End of Text
00020 #define ASCII_EOT           4           //End of Transmission
00021 #define ASCII_ENQ           5           //Enquiry
00022 #define ASCII_ACK           6           //Ack
00023 #define ASCII_BEL           7           //Bell
00024 #define ASCII_BS            8           //BackSpace
00025 #define ASCII_HT            9           //Horizontal Tab
00026 #define ASCII_LF            10          //Line Feed
00027 #define ASCII_VT            11          //Vertical Tab
00028 #define ASCII_FF            12          //Form Feed
00029 #define ASCII_CR            13          //Carriage Return
00030 #define ASCII_SO            14          //Shift Out
00031 #define ASCII_SI            15          //Shift In
00032 #define ASCII_DLE           16          //Data Link Escape
00033 #define ASCII_DC1           17          //Device Control 1
00034 #define ASCII_DC2           18
00035 #define ASCII_DC3           19
00036 #define ASCII_DC4           20
00037 #define ASCII_NAK           21          //Negative Ack
00038 #define ASCII_SYN           22          //Synchronous idle
00039 #define ASCII_ETB           23          //End of Transmission Block
00040 #define ASCII_CAN           24          //Cancel
00041 #define ASCII_EM            25          //End of Medium
00042 #define ASCII_SUB           26          //Substitute
00043 #define ASCII_ESC           27          //Escape
00044 #define ASCII_FS            28          //File Separator
00045 #define ASCII_GS            29          //Group Separtor
00046 #define ASCII_RS            30          //Record Separator
00047 #define ASCII_US            31          //Unit Separator
00048
00049
00050
00051
00052
00053
00054
00055 #endif /* ASCII_H_ */
```

6.29 my_lib/cmd_sort.c File Reference

```
#include "cmd_sort.h"
```

Functions

- void CommandSort (uint8_t *cmdString)

6.29.1 Function Documentation

6.29.1.1 void CommandSort (uint8_t * *cmdString*)

Definition at line 16 of file cmd_sort.c.

6.30 cmd_sort.c

```
00001 /*
00002  * cmd_sort.c
00003  *
00004  *  Created on: Nov 28, 2013
00005  *      Author: rikardo
00006  */
00007
00008 #include "cmd_sort.h"
00009
00010
00011 /*
00012  * Processes a string as a command
00013  * todo: make software interrupt for routines, call from here
00014  * todo: return function pointer
00015  */
00016 void CommandSort(uint8_t *cmdString)
00017 {
00018
00019 }
00020
00021
```

6.31 my_lib/cmd_sort.h File Reference

```
#include "includeAll.h"
```

Data Structures

- struct CommandInstance

Macros

- #define MAX_BUFFER_SIZE 30

Functions

- void CommandSort (uint8_t *cmdString)

6.31.1 Macro Definition Documentation

6.31.1.1 #define MAX_BUFFER_SIZE 30

Definition at line 13 of file cmd_sort.h.

6.31.2 Function Documentation

6.31.2.1 void CommandSort (uint8_t * *cmdString*)

Definition at line 16 of file cmd_sort.c.

6.32 cmd_sort.h

```
00001 /*
00002  * cmd_sort.h
00003  *
00004  *  Created on: Nov 28, 2013
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef CMD_SORT_H_
00009 #define CMD_SORT_H_
00010
00011 #include "includeAll.h"
00012
00013 #define MAX_BUFFER_SIZE             30
00014
00015 typedef struct{
00016     uint8_t charIn;
00017     uint8_t cmdBuffer[MAX_BUFFER_SIZE];
00018     uint16_t charOut[MAX_BUFFER_SIZE];
00019     uint8_t charOutPtr;
00020 } CommandInstance;
00021
00022
00023
00024 void CommandSort(uint8_t *cmdString);
00025
00026
00027
00028
00029 #endif /* CMD_SORT_H_ */
```

6.33 my_lib/gpioPin_masks.h File Reference

Macros

- #define IOPin_0 0x00000001
- #define IOPin_1 0x00000002
- #define IOPin_2 0x00000004
- #define IOPin_3 0x00000008
- #define IOPin_4 0x00000010
- #define IOPin_5 0x00000020
- #define IOPin_6 0x00000040
- #define IOPin_7 0x00000080
- #define IOPin_8 0x00000100
- #define IOPin_9 0x00000200
- #define IOPin_10 0x00000400
- #define IOPin_11 0x00000800
- #define IOPin_12 0x00001000
- #define IOPin_13 0x00002000

- #define IOPin_14 0x00004000
- #define IOPin_15 0x00008000
- #define IOPin_16 0x00010000
- #define IOPin_17 0x00020000
- #define IOPin_18 0x00040000
- #define IOPin_19 0x00080000
- #define IOPin_20 0x00100000
- #define IOPin_21 0x00200000
- #define IOPin_22 0x00400000
- #define IOPin_23 0x00800000
- #define IOPin_24 0x01000000
- #define IOPin_25 0x02000000
- #define IOPin_26 0x04000000
- #define IOPin_27 0x08000000
- #define IOPin_28 0x10000000
- #define IOPin_29 0x20000000
- #define IOPin_30 0x40000000
- #define IOPin_31 0x08000000

6.33.1 Macro Definition Documentation

6.33.1.1 #define IOPin_0 0x00000001

Definition at line 5 of file gpioPin_masks.h.

6.33.1.2 #define IOPin_1 0x00000002

Definition at line 6 of file gpioPin_masks.h.

6.33.1.3 #define IOPin_10 0x00000400

Definition at line 15 of file gpioPin_masks.h.

6.33.1.4 #define IOPin_11 0x00000800

Definition at line 16 of file gpioPin_masks.h.

6.33.1.5 #define IOPin_12 0x00001000

Definition at line 17 of file gpioPin_masks.h.

6.33.1.6 #define IOPin_13 0x00002000

Definition at line 18 of file gpioPin_masks.h.

6.33.1.7 #define IOPin_14 0x00004000

Definition at line 19 of file gpioPin_masks.h.

6.33.1.8 #define IOPin_15 0x00008000

Definition at line 20 of file gpioPin_masks.h.

6.33.1.9 #define IOPin_16 0x00010000

Definition at line 21 of file gpioPin_masks.h.

6.33.1.10 #define IOPin_17 0x00020000

Definition at line 22 of file gpioPin_masks.h.

6.33.1.11 #define IOPin_18 0x00040000

Definition at line 23 of file gpioPin_masks.h.

6.33.1.12 #define IOPin_19 0x00080000

Definition at line 24 of file gpioPin_masks.h.

6.33.1.13 #define IOPin_2 0x00000004

Definition at line 7 of file gpioPin_masks.h.

6.33.1.14 #define IOPin_20 0x00100000

Definition at line 25 of file gpioPin_masks.h.

6.33.1.15 #define IOPin_21 0x00200000

Definition at line 26 of file gpioPin_masks.h.

6.33.1.16 #define IOPin_22 0x00400000

Definition at line 27 of file gpioPin_masks.h.

6.33.1.17 #define IOPin_23 0x00800000

Definition at line 28 of file gpioPin_masks.h.

6.33.1.18 #define IOPin_24 0x01000000

Definition at line 29 of file gpioPin_masks.h.

6.33.1.19 #define IOPin_25 0x02000000

Definition at line 30 of file gpioPin_masks.h.

6.33.1.20 #define IOPin_26 0x04000000

Definition at line 31 of file gpioPin_masks.h.

6.33.1.21 #define IOPin_27 0x08000000

Definition at line 32 of file gpioPin_masks.h.

6.33.1.22 #define IOPin_28 0x10000000

Definition at line 33 of file gpioPin_masks.h.

6.33.1.23 #define IOPin_29 0x20000000

Definition at line 34 of file gpioPin_masks.h.

6.33.1.24 #define IOPin_3 0x00000008

Definition at line 8 of file gpioPin_masks.h.

6.33.1.25 #define IOPin_30 0x40000000

Definition at line 35 of file gpioPin_masks.h.

6.33.1.26 #define IOPin_31 0x08000000

Definition at line 36 of file gpioPin_masks.h.

6.33.1.27 #define IOPin_4 0x00000010

Definition at line 9 of file gpioPin_masks.h.

6.33.1.28 #define IOPin_5 0x00000020

Definition at line 10 of file gpioPin_masks.h.

6.33.1.29 #define IOPin_6 0x00000040

Definition at line 11 of file gpioPin_masks.h.

6.33.1.30 #define IOPin_7 0x00000080

Definition at line 12 of file gpioPin_masks.h.

6.33.1.31 #define IOPin_8 0x00000100

Definition at line 13 of file gpioPin_masks.h.

6.33.1.32 #define IOPin_9 0x00000200

Definition at line 14 of file gpioPin_masks.h.

6.34 gpioPin_masks.h

```
00001 #ifndef GPIOPIN_MASKS
00002 #define GPIOPIN_MASKS
00003
00004
00005 #define IOPin_0     0x00000001
00006 #define IOPin_1     0x00000002
00007 #define IOPin_2     0x00000004
00008 #define IOPin_3     0x00000008
00009 #define IOPin_4     0x00000010
00010 #define IOPin_5     0x00000020
00011 #define IOPin_6     0x00000040
00012 #define IOPin_7     0x00000080
00013 #define IOPin_8     0x00000100
00014 #define IOPin_9     0x00000200
00015 #define IOPin_10    0x00000400
00016 #define IOPin_11    0x00000800
00017 #define IOPin_12    0x00001000
00018 #define IOPin_13    0x00002000
00019 #define IOPin_14    0x00004000
00020 #define IOPin_15    0x00008000
00021 #define IOPin_16    0x00010000
00022 #define IOPin_17    0x00020000
00023 #define IOPin_18    0x00040000
00024 #define IOPin_19    0x00080000
00025 #define IOPin_20    0x00100000
00026 #define IOPin_21    0x00200000
00027 #define IOPin_22    0x00400000
00028 #define IOPin_23    0x00800000
00029 #define IOPin_24    0x01000000
00030 #define IOPin_25    0x02000000
00031 #define IOPin_26    0x04000000
00032 #define IOPin_27    0x08000000
00033 #define IOPin_28    0x10000000
00034 #define IOPin_29    0x20000000
00035 #define IOPin_30    0x40000000
00036 #define IOPin_31    0x08000000
00037
00038
00039
00040 #endif //gpiopin_masks
```

6.35 my_lib/ir.c File Reference

```
#include "ir.h"
```

Functions

- void IRInit (IRInstance *instPtr)
- void IRSend (IRInstance *instPtr, uint16_t address, uint16_t byte)
- void __inline IRByteBySoftware (IRInstance *instPtr, uint16_t address, uint16_t byte)
- void __inline IRRepeat (uint32_t port, uint32_t pin, uint8_t pulses, uint16_t delay)

6.35.1 Function Documentation

6.35.1.1 void __inline IRByteBySoftware (IRInstance * *instPtr*, uint16_t *address*, uint16_t *byte*)

Definition at line 146 of file ir.c.

6.35.1.2 void IRInit (IRInstance * *instPtr*)

Definition at line 16 of file ir.c.

6.35.1.3 void __inline IRRepeat (uint32_t *port*, uint32_t *pin*, uint8_t *pulses*, uint16_t *delay*)

Definition at line 212 of file ir.c.

6.35.1.4 void IRSend (IRInstance * *instPtr*, uint16_t *address*, uint16_t *byte*)

Definition at line 57 of file ir.c.

6.36 ir.c

```
00001 #include "ir.h"
00002
00003 #include "ir.h"
00004
00005
00006
00007
00008
00009
00010 /*
00011  * Calls IR init
00012  * Modes: IR_BY_SOFTWARE
00013  *        IR_BY_UART
00014  *        IR_BY_TIMER
00015  */
00016 void IRInit(IRInstance *instPtr)
00017 {
00018
00019     if((instPtr->Mode&IR_BY_SOFTWARE)!=0)
00020     {
00021         instPtr->CarrierPeriod = (uint16_t) 1000/instPtr->
      CarrierFrequency;
00022         #ifdef IR_BY_SOFTWARE_EN
00023         /*
00024          * for software modulation, configure delay timing
00025          */
00026         if((instPtr->Mode & (IR_NEC_PROTOCOL|IR_NEC_EXTENDED))!=0)
00027             instPtr->Pulses = (uint16_t) (NEC_PULSE_TIME*((uint16_t) instPtr->
      CarrierFrequency))/2000;
00028         if((instPtr->Mode&IR_RC5_PROTOCOL)!=0)
00029             instPtr->Pulses = (uint16_t) (RC5_PULSE_TIME*((uint16_t) instPtr->
      CarrierFrequency))/1000;
00030         #endif
00031     }
00032
00033
00034     if(instPtr->Mode == IR_BY_UART)
00035     {
00036         #ifdef IR_BY_UART_EN
00037
00038         #endif
00039     }
00040
00041
00042     if(instPtr->Mode == IR_BY_TIMER)
00043     {
00044         #ifdef IR_BY_TIMER_EN
00045
00046         #endif
00047     }
00048
00049
00050 }
00051
00052
00053
00054 /*
00055  * Sends IR data according to instance
00056  */
00057 void IRSend(IRInstance *instPtr, uint16_t address, uint16_t byte)
00058 {
00059     uint32_t data;
```

```
00060     uint8_t tempAddress=0;
00061     uint8_t tempByte=0;
00062
00063 #ifdef IR_BY_SOFTWARE_EN
00064     uint16_t pulses;
00065     uint8_t roller;
00066     uint16_t delay = instPtr->CarrierPeriod/2;
00067 #endif
00068
00069     if((instPtr->Mode&IR_NEC_PROTOCOL)!=0)                          //inversdo enderee
      dados
00070     {
00071         tempAddress = ~address;
00072         address = ((address&0xFF))|((tempAddress&0xFF)<<8);
00073         tempByte = ~byte;
00074         byte = ((byte&0xFF))|((tempByte&0xFF)<<8);
00075     }
00076     if((instPtr->Mode&(IR_NEC_PROTOCOL|IR_NEC_EXTENDED))!=0)
00077             data = address|byte<<16;
00078
00079 #ifdef IR_BY_SOFTWARE_EN
00080     if((instPtr->Mode&IR_BY_SOFTWARE)!=0)
00081     {
00082         if((instPtr->Mode&(IR_NEC_EXTENDED|IR_NEC_PROTOCOL))!=0)      //
     padrde envio
00083         {
00084             pulses = instPtr->Pulses*32;
00085             roller = 32;
00086             while(pulses>0)                     //start signal send 9ms
00087             {
00088                 IRPinSet(instPtr->TxPort, instPtr->TxPin);
00089                 IRDelayUs(delay);
00090                 IRPinClear(instPtr->TxPort, instPtr->TxPin);
00091                 IRDelayUs(delay);
00092                 pulses--;
00093             }
00094             IRDelayMs(4);                   //protocol wait time
00095             IRDelayUs(500);
00096             while(roller>0)
00097             {
00098                 pulses = instPtr->Pulses;
00099                 while(pulses>0)                 //carrier send
00100                 {
00101                     IRPinSet(instPtr->TxPort, instPtr->TxPin);
00102                     IRDelayUs(delay);
00103                     IRPinClear(instPtr->TxPort, instPtr->TxPin);
00104                     IRDelayUs(delay);
00105                     pulses--;
00106                 }
00107                 if((data&0x1)!=0)
00108                     IRDelayUs(NEC_PULSE_TIME*2);
00109                 IRDelayUs(NEC_PULSE_TIME);
00110                 data >>= 1;
00111                 roller --;
00112             }
00113             pulses = instPtr->Pulses;
00114             while(pulses>0)                 //end signal send 562.5 us
00115             {
00116                 IRPinSet(instPtr->TxPort, instPtr->TxPin);
00117                 IRDelayUs(delay);
00118                 IRPinClear(instPtr->TxPort, instPtr->TxPin);
00119                 IRDelayUs(delay);
00120                 pulses--;
00121             }
00122         }
00123         if((instPtr->Mode&IR_RC5_PROTOCOL)!=0)
00124         {
00125             //todo: to be implemented. sem saco anymore.
00126         }
00127     }
00128 #endif
00129 #ifdef IR_BY_UART_EN
00130     if((instPtr->Mode&IR_BY_UART)!=0)
00131     {
00132
00133         //todo: make uart send buffer/command
00134     }
00135 #endif
00136 #ifdef IR_BY_TIMER_EN
00137
00138 #endif
00139 }
00140
00141
00142 #ifdef IR_BY_SOFTWARE
00143 /*
00144  * sends a modulated bit
```

```
00145  */
00146 void __inline IRByteBySoftware(IRInstance *instPtr, uint16_t address, uint16_t
      byte)
00147 {
00148     uint8_t tempAddress=0;
00149     uint8_t tempByte=0;
00150     uint16_t pulses;
00151     uint32_t data;
00152     uint8_t roller;
00153     uint16_t delay = instPtr->CarrierPeriod/2;
00154 
00155     if((instPtr->Mode&IR_NEC_PROTOCOL)!=0)                        //inversdo enderee
       dados
00156     {
00157         tempAddress = ~address;
00158         address = ((address&0xFF))|((tempAddress&0xFF)<<8);
00159         tempByte = ~byte;
00160         byte = ((byte&0xFF))|((tempByte&0xFF)<<8);
00161     }
00162     if((instPtr->Mode&(IR_NEC_EXTENDED|IR_NEC_PROTOCOL))!=0)      //
      padrde envio
00163     {
00164         data = address|byte<<16;
00165         pulses = instPtr->Pulses*32;
00166         roller = 32;
00167         while(pulses>0)                     //start signal send 9ms
00168         {
00169             IRPinSet(instPtr->TxPort, instPtr->TxPin);
00170             IRDelayUs(delay);
00171             IRPinClear(instPtr->TxPort, instPtr->TxPin);
00172             IRDelayUs(delay);
00173             pulses--;
00174         }
00175         IRDelayMs(4);                   //protocol wait time
00176         IRDelayUs(500);
00177         while(roller>0)
00178         {
00179             pulses = instPtr->Pulses;
00180             while(pulses>0)                 //carrier send
00181             {
00182                 IRPinSet(instPtr->TxPort, instPtr->TxPin);
00183                 IRDelayUs(delay);
00184                 IRPinClear(instPtr->TxPort, instPtr->TxPin);
00185                 IRDelayUs(delay);
00186                 pulses--;
00187             }
00188             if((data&0x1)!=0)
00189                 IRDelayUs(NEC_PULSE_TIME*2);
00190             IRDelayUs(NEC_PULSE_TIME);
00191             data >>= 1;
00192             roller --;
00193         }
00194         pulses = instPtr->Pulses;
00195         while(pulses>0)                 //end signal send 562.5 us
00196         {
00197             IRPinSet(instPtr->TxPort, instPtr->TxPin);
00198             IRDelayUs(delay);
00199             IRPinClear(instPtr->TxPort, instPtr->TxPin);
00200             IRDelayUs(delay);
00201             pulses--;
00202         }
00203     }
00204     if((instPtr->Mode&IR_RC5_PROTOCOL)!=0)
00205     {
00206         //todo: to be implemented. sem saco anymore.
00207     }
00208 }
00209 
00210 
00211 
00212 void __inline IRRepeat(uint32_t port, uint32_t pin, uint8_t pulses, uint16_t delay)
00213 {
00214     uint8_t tempPulses;
00215     //fixme: repeat codes should be sent at 108ms intervals
00216     tempPulses = pulses;
00217     pulses *= 16;
00218     delay /= 2;
00219     while(pulses>0)                 //start signal send 9ms
00220     {
00221         IRPinSet(port, pin);
00222         IRDelayUs(delay);
00223         IRPinClear(port, pin);
00224         IRDelayUs(delay);
00225         pulses--;
00226     }
00227     IRDelayMs(2);
00228     IRDelayUs(250);
```

```
00229     pulses = tempPulses;
00230     while(pulses>0)                 //end signal send 562.5 us
00231     {
00232         IRPinSet(port, pin);
00233         IRDelayUs(delay);
00234         IRPinClear(port, pin);
00235         IRDelayUs(delay);
00236         pulses--;
00237     }
00238 }
00239
00240
00241
00242
00243 #endif //ir_by_software
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
```

6.37 my_lib/ir.h File Reference

```
#include "includeAll.h"
```

Data Structures

- struct IRInstance

Macros

- #define IR_MAX_INSTANCES 4
- #define IR_BY_SOFTWARE 0x0001
- #define IR_BY_UART 0x0002
- #define IR_BY_TIMER 0x0004
- #define IR_BY_EXTERNAL_TIEMR 0x0008
- #define IR_NEC_PROTOCOL 0x0010
- #define IR_NEC_EXTENDED 0x0020
- #define IR_MY_PROTOCOL 0x0040
- #define IR_RC5_PROTOCOL 0x0080
- #define IR_REPEAT_COMMAND_ENABLE 0x0100
- #define IR_REPEAT_COMMAND_DISABLE 0x0000
- #define NEC_PULSE_TIME 562
- #define RC5_PULSE_TIME 889
- #define IRPinSet(port, pin) PinSet(port, pin)
- #define IRPinClear(port, pin) PinClear(port, pin)
- #define IRDelayMs(delay) SysDelayMs(delay)
- #define IRDelayUs(delay) SysDelayUs(delay)
- #define IRDelay(delay) SysDelay(delay)

Functions

- void IRInit (IRInstance *instPtr)
- void IRSend (IRInstance *instPtr, uint16_t address, uint16_t byte)
- void IRByteBySoftware (IRInstance *instPtr, uint16_t address, uint16_t byte)
- void IRRepeat (uint32_t port, uint32_t pin, uint8_t pulses, uint16_t delay)

6.37.1 Macro Definition Documentation

6.37.1.1 #define IR_BY_EXTERNAL_TIEMR 0x0008

Definition at line 63 of file ir.h.

6.37.1.2 #define IR_BY_SOFTWARE 0x0001

Definition at line 60 of file ir.h.

6.37.1.3 #define IR_BY_TIMER 0x0004

Definition at line 62 of file ir.h.

6.37.1.4 #define IR_BY_UART 0x0002

Definition at line 61 of file ir.h.

6.37.1.5 #define IR_MAX_INSTANCES 4

Definition at line 55 of file ir.h.

6.37.1.6 #define IR_MY_PROTOCOL 0x0040

Definition at line 67 of file ir.h.

6.37.1.7 #define IR_NEC_EXTENDED 0x0020

Definition at line 66 of file ir.h.

6.37.1.8 #define IR_NEC_PROTOCOL 0x0010

Definition at line 65 of file ir.h.

6.37.1.9 #define IR_RC5_PROTOCOL 0x0080

Definition at line 68 of file ir.h.

6.37.1.10 #define IR_REPEAT_COMMAND_DISABLE 0x0000

Definition at line 71 of file ir.h.

6.37.1.11 #define IR_REPEAT_COMMAND_ENABLE 0x0100

Definition at line 70 of file ir.h.

6.37.1.12 #define IRDelay(*delay*) SysDelay(delay)

Definition at line 102 of file ir.h.

6.37.1.13 #define IRDelayMs(*delay*) SysDelayMs(delay)

Definition at line 100 of file ir.h.

6.37.1.14 #define IRDelayUs(*delay*) SysDelayUs(delay)

Definition at line 101 of file ir.h.

6.37.1.15 #define IRPinClear(*port, pin*) PinClear(port, pin)

Definition at line 99 of file ir.h.

6.37.1.16 #define IRPinSet(*port, pin*) PinSet(port, pin)

Definition at line 98 of file ir.h.

6.37.1.17 #define NEC_PULSE_TIME 562

Definition at line 77 of file ir.h.

6.37.1.18 #define RC5_PULSE_TIME 889

Definition at line 79 of file ir.h.

6.37.2 Function Documentation

6.37.2.1 void IRByteBySoftware (IRInstance * *instPtr,* uint16_t *address,* uint16_t *byte*)

Definition at line 146 of file ir.c.

6.37.2.2 void IRInit (IRInstance * *instPtr*)

Definition at line 16 of file ir.c.

6.37.2.3 void IRRepeat (uint32_t *port,* uint32_t *pin,* uint8_t *pulses,* uint16_t *delay*)

Definition at line 212 of file ir.c.

6.37.2.4 void IRSend (IRInstance * *instPtr,* uint16_t *address,* uint16_t *byte*)

Definition at line 57 of file ir.c.

6.38 ir.h

```
00001 #ifndef ir_h
00002 #define ir_h
00003
00004
00005
00006 //main file header
00007 #include "includeAll.h"
00008
00009
00010 /*
00011  * need to declare:
00012  *
00013
00014 #define IRTX_FREQ           38000
00015 #define IRTX_PORT           GPIO_PORTB_BASE
00016 #define IRTX_PIN            GPIO_PIN_1
00017 #define IRRX_PORT           GPIO_PORTB_BASE
00018 #define IRRX_PIN            GPIO_PIN_0
00019 //larger compiled file
00020 #define IR_BY_SOFTWARE_EN
00021 #define IR_BY_UART_EN
00022 #define IR_UART_BASE        UART1_BASE
00023 #define IR_BY_TIMER_EN
00024 #define IR_TIMER_BASE       TIMER3_BASE
00025
00026
00027
00028  */
00029
00030
00031
00032 /*
00033  * InfraRed transceiver library
00034  * todo:    uart
00035  *          timer
00036  *          software
00037  *
00038  * -> uart peripheral support
00039  *      initiated for stellaris/tivaC uCs, uart IR coding support
00040  *  -> timer peripheral support
00041  *      common to all uCs
00042  *      carrier frequency generation
00043  *  -> software support
00044  *      support for full software control and emulation
00045  *      delay by cpu time use
00046  *
00047  *
00048  *  devBy: rnm (17/11/13)
00049  */
00050
00051
00052 /*
00053  * Op. Param.
00054  */
00055 #define IR_MAX_INSTANCES            4
00056
00057 /*
00058  * Op. Masks
00059  */
00060 #define IR_BY_SOFTWARE              0x0001
00061 #define IR_BY_UART                  0x0002
00062 #define IR_BY_TIMER                 0x0004
00063 #define IR_BY_EXTERNAL_TIEMR        0x0008
00064
00065 #define IR_NEC_PROTOCOL             0x0010
00066 #define IR_NEC_EXTENDED             0x0020
00067 #define IR_MY_PROTOCOL              0x0040
00068 #define IR_RC5_PROTOCOL             0x0080
00069
00070 #define IR_REPEAT_COMMAND_ENABLE    0x0100
00071 #define IR_REPEAT_COMMAND_DISABLE   0x0000
00072
00073 /*
00074  * Protocol Definitions
00075  */
00076
00077 #define NEC_PULSE_TIME              562
00078
00079 #define RC5_PULSE_TIME              889
00080
00081
00082
00083 typedef struct{
00084     uint16_t Mode;              //IR_BY_XX | IR_XX_PROTOCOL
```

```
00085     uint8_t CarrierFrequency;   //in kHZ
00086     uint16_t CarrierPeriod;        // in uS
00087     uint32_t TxPin;
00088     uint32_t TxPort;
00089     uint32_t RxPin;
00090     uint32_t RxPort;
00091     uint16_t ReceiveAddress;
00092     uint16_t ReceiveBuffer;
00093     uint16_t Pulses;
00094     uint8_t LastData;
00095 }IRInstance;
00096
00097
00098 #define IRPinSet(port, pin)     PinSet(port, pin)
00099 #define IRPinClear(port, pin)   PinClear(port, pin)
00100 #define IRDelayMs(delay)        SysDelayMs(delay)
00101 #define IRDelayUs(delay)        SysDelayUs(delay)
00102 #define IRDelay(delay)          SysDelay(delay)
00103
00104
00105
00106 void IRInit(IRInstance *instPtr);
00107 void IRSend(IRInstance *instPtr, uint16_t address, uint16_t byte);
00108 void IRByteBySoftware(IRInstance *instPtr, uint16_t address, uint16_t byte);
00109 void IRRepeat(uint32_t port, uint32_t pin, uint8_t pulses, uint16_t delay);
00110
00111
00112
00113 #endif// if_h
```

6.39 my_lib/lcd.c File Reference

```
#include "lcd.h"
```

Macros

- #define true 1
- #define false 0
- #define trueDefinedLCD
- #define lcd_vector_index 9

Functions

- void LCDInit (void)

  Initializes the LCD Module.
- void LCDSendCmd (uint8_t cmd)
- void LCDSendChar (uint8_t txt)

  send single character to LCD.
- __inline void LCDSend (uint8_t send)

  Send data to LCD, no RS control.
- void LCDPosition (uint8_t row, uint8_t col)

  Set LCD write position.
- void LCDPositionNoDelay (uint8_t row, uint8_t col)

  Set LCD write position, no delay in function.
- void LCDSendString (uint8_t *string, uint8_t breakLine)

  Send string to LCD.
- void LCDSendNumStrict (int64_t num, uint8_t length, uint8_t isSigned, uint8_t showZeros)
- void LCDSendNum (int64_t num, uint8_t length, uint8_t isSigned, uint8_t showZeros)
- void LCDSendNumArray (uint8_t *index)
- void LCDClear (void)
- void LCDDisplayOn (uint8_t onOff)

- void LCDSendHex (uint8_t *array)
- void numToArray (int32_t num, uint8_t *array, uint8_t length, uint16_t base)
- void LCDRegisterSpecial (uint8_t number, uint8_t *character)
- void LCDShift (uint8_t shift)
- void LCDHome (void)
- void arrayToNum (uint8_t *array, uint32_t *num, uint8_t base)
- void LCDSendVU (uint32_t num, uint32_t base)

Variables

- const char LCD_CmdInit_Vector [lcd_vector_index]
- const unsigned int LCD_InitDelay_Vector [lcd_vector_index]

6.39.1 Macro Definition Documentation

6.39.1.1 #define false 0

Definition at line 12 of file lcd.c.

6.39.1.2 #define lcd_vector_index 9

Definition at line 21 of file lcd.c.

6.39.1.3 #define true 1

Checks and defines boolean values.

Definition at line 11 of file lcd.c.

6.39.1.4 #define trueDefinedLCD

Definition at line 13 of file lcd.c.

6.39.2 Function Documentation

6.39.2.1 void arrayToNum (uint8_t * *array*, uint32_t * *num*, uint8_t *base*)

Definition at line 433 of file lcd.c.

6.39.2.2 void LCDClear (void)

Definition at line 297 of file lcd.c.

6.39.2.3 void LCDDisplayOn (uint8_t *onOff*)

Definition at line 311 of file lcd.c.

6.39.2.4 void LCDHome (void)

Definition at line 425 of file lcd.c.

6.39.2.5 void LCDInit (void)

Initializes the LCD Module.

Called once at startup. Takes no parameters.

Returns

None.

Definition at line 47 of file lcd.c.

6.39.2.6 void LCDPosition (uint8_t *row*, uint8_t *col*)

Set LCD write position.

Parameters

row	uint8_t row.
col	uint8_t column.

Definition at line 118 of file lcd.c.

6.39.2.7 void LCDPositionNoDelay (uint8_t *row*, uint8_t *col*)

Set LCD write position, no delay in function.

Parameters

row	uint8_t row.
col	uint8_t column.

Definition at line 138 of file lcd.c.

6.39.2.8 void LCDRegisterSpecial (uint8_t *number*, uint8_t * *character*)

Definition at line 398 of file lcd.c.

6.39.2.9 __inline void LCDSend (uint8_t *send*)

Send data to LCD, no RS control.

Parameters

send	uint8_t data to be sent.

Definition at line 102 of file lcd.c.

6.39.2.10 void LCDSendChar (uint8_t *txt*)

send single character to LCD.

Parameters

txt	uint8_t type data to be sent, 8 bits.

Definition at line 89 of file lcd.c.

6.39.2.11 void LCDSendCmd (uint8_t *cmd*)

Definition at line 77 of file lcd.c.

6.39.2.12 void LCDSendHex (uint8_t * *array*)

Definition at line 325 of file lcd.c.

6.39.2.13 void LCDSendNum (int64_t *num*, uint8_t *length*, uint8_t *isSigned*, uint8_t *showZeros*)

Definition at line 246 of file lcd.c.

6.39.2.14 void LCDSendNumArray (uint8_t * *index*)

Definition at line 284 of file lcd.c.

6.39.2.15 void LCDSendNumStrict (int64_t *num*, uint8_t *length*, uint8_t *isSigned*, uint8_t *showZeros*)

Definition at line 199 of file lcd.c.

6.39.2.16 void LCDSendString (uint8_t * *string*, uint8_t *breakLine*)

Send string to LCD.

Parameters

**string*	uint8_t string to be sent.
breakLine	uint8_t break line at the end of LCD length.

Writes a string of characteres on display Processes according to the ASCII code 0 - NULL

Definition at line 162 of file lcd.c.

6.39.2.17 void LCDSendVU (uint32_t *num*, uint32_t *base*)

Definition at line 445 of file lcd.c.

6.39.2.18 void LCDShift (uint8_t *shift*)

Definition at line 416 of file lcd.c.

6.39.2.19 void numToArray (int32_t *num*, uint8_t * *array*, uint8_t *length*, uint16_t *base*)

Definition at line 358 of file lcd.c.

6.39.3 Variable Documentation

6.39.3.1 const char LCD_CmdInit_Vector[lcd_vector_index]

Initial value:

```
=
        {
            0x03, 0x38, 0x38, 0x38, 0x01, LCD_DISPLAY_CONFIG,
      LCD_DISPLAY_INCREMENT, 0x01, 0x02,
        }
```

Definition at line 22 of file lcd.c.

6.39.3.2 const unsigned int LCD_InitDelay_Vector[lcd_vector_index]

Initial value:

```
=
        {
            8000, 800, 800, 800, 600, 600, 200, 200, 200
        }
```

LCD Init command delay vector, in uS

Definition at line 32 of file lcd.c.

6.40 lcd.c

```
00001
00002 #include "lcd.h"
00003
00004
00005
00006
00007 /**
00008  * Checks and defines boolean values.
00009  */
00010 #ifndef true
00011 #define true 1
00012 #define false 0
00013 #define trueDefinedLCD
00014 #endif
00015
00016
00017 /*
00018  * Initialization Sequence:
00019  * TODO: create masks for LCD commands
00020  */
00021 #define lcd_vector_index    9
00022 const char LCD_CmdInit_Vector [lcd_vector_index] = \
00023         {
00024             0x03, 0x38, 0x38, 0x38, 0x01, LCD_DISPLAY_CONFIG,
      LCD_DISPLAY_INCREMENT, 0x01, 0x02,
00025         };
00026 /*
00027  * Delay time in uSs
00028  */
00029 /**
00030  * LCD Init command delay vector, in uS
00031  */
00032 const unsigned int LCD_InitDelay_Vector[lcd_vector_index] = \
00033         {
00034             8000, 800, 800, 800, 600, 600, 200, 200, 200
00035         };
00036
00037
00038
00039
00040 /**
00041  *  \brief Initializes the LCD Module
00042  *
00043  *  Called once at startup. Takes no parameters.
00044  *
00045  *  \return None.
00046  */
00047 void LCDInit(void)
00048 {
00049     uint8_t Vector_Scan = 0;
00050     LCDDelay(15000);
00051     for(Vector_Scan=0; Vector_Scan < lcd_vector_index; Vector_Scan++)
00052     {
```

```
00053         LCDSendCmd(LCD_CmdInit_Vector[Vector_Scan]);
00054         LCDDelay(LCD_InitDelay_Vector[Vector_Scan]);
00055     }
00056     //splash screen
00057 #if LCD_SPLASHSCREEN1 == 1
00058     LCDPosition(1,1);
00059     LCDSendString(LCD_splashscreen_row1, 0);
00060     LCDPosition(2,1);
00061     LCDSendString(LCD_splashscreen_row2, 0);
00062     LCDDelay(2*1000*1000);
00063 #endif
00064 #if LCD_SPLASHSCREEN2 == 1
00065     LCDPosition(1,1);
00066     LCDSendString(LCD_splashscreen2_row1, false);
00067     LCDPosition(2,1);
00068     LCDSendString(LCD_splashscreen2_row2, false);
00069     LCDDelay(2*1000*1000);
00070 #endif
00071     LCDClear();
00072 }
00073
00074 /*
00075  * Send a Command to the LCD
00076  */
00077 void LCDSendCmd(uint8_t cmd)
00078 {
00079     LCD_RS_Low;
00080     LCDSend(cmd);
00081 }
00082
00083
00084 /**
00085  * \brief send single character to LCD.
00086  *
00087  * \param txt uint8_t type data to be sent, 8 bits.
00088  */
00089 void LCDSendChar(uint8_t txt)
00090 {
00091     LCD_RS_High;
00092     LCDSend(txt);
00093     LCD0Status.col ++;
00094     LCD_RS_Low;
00095 }
00096
00097 /**
00098  * \brief Send data to LCD, no RS control
00099  *
00100  * \param send uint8_t data to be sent.
00101  */
00102 __inline void LCDSend(uint8_t send)
00103 {
00104     LCD_EN_Low;
00105     LCD_DTA_Send(send);
00106     LCDDelay(4);
00107     LCD_EN_High;
00108     LCDDelay(4);
00109     LCD_EN_Low;
00110 }
00111
00112 /**
00113  * \brief Set LCD write position
00114  *
00115  * \param row uint8_t row.
00116  * \param col uint8_t column.
00117  */
00118 void LCDPosition(uint8_t row, uint8_t col)
00119 {
00120     LCD0Status.row = row;
00121     LCD0Status.col = col;
00122     col--;
00123     if(row==1)
00124         row = 0x80;
00125     if(row==2)
00126         row = 0xC0;
00127     LCDSendCmd(row+col);
00128     LCDDelay(20);
00129 }
00130
00131
00132 /**
00133  * \brief Set LCD write position, no delay in function
00134  *
00135  * \param row uint8_t row.
00136  * \param col uint8_t column.
00137  */
00138 void LCDPositionNoDelay(uint8_t row, uint8_t col)
00139 {
```

```
00140     LCD0Status.row = row;
00141     LCD0Status.col = col;
00142     col--;
00143     if(row==1)
00144         row = 0x80;
00145     if(row==2)
00146         row = 0xC0;
00147     LCDSendCmd(row+col);
00148 }
00149
00150
00151 /**
00152  * \brief Send string to LCD
00153  *
00154  * \param *string uint8_t string to be sent.
00155  * \param breakLine uint8_t break line at the end of LCD length.
00156  *
00157  *
00158  * Writes a string of characteres on display
00159  * Processes according to the ASCII code
00160  * 0 - NULL
00161  */
00162 void LCDSendString(uint8_t *string, uint8_t breakLine)
00163 {
00164     while(*string)
00165     {
00166         LCDSendChar(*string);
00167         string++;
00168         if(LCD0Status.col==LCD_col_num && breakLine==true)
00169         {
00170             if(LCD0Status.row<=LCD_row_num)
00171                 LCDPosition(LCD0Status.row+1, 1);
00172             else
00173                 LCDPosition(0, 1);
00174         }
00175     }
00176 }
00177
00178
00179
00180
00181 /*
00182  * TODO: make function to print string and remaining spaces in LCD
00183  */
00184
00185
00186 /*
00187  * Prints a number, from a variable, to the LCD
00188  * PARAM: num, length, isSigned, showZeros
00189  * IF signed
00190  *      Limits to a max of 10 digits to a positive number
00191  *      Limits to a max of 9 digits to a negative number
00192  * ELSE
00193  *      limits to 10 chars
00194  * IF showzeros
00195  *      shows all leading zeros
00196  * else
00197  *      supresses zeros; places space instead
00198  */
00199 void LCDSendNumStrict(int64_t num, uint8_t length, uint8_t isSigned, uint8_t showZeros)
00200 {
00201     uint8_t index =0;
00202     uint8_t out;
00203     uint64_t multiple = 1;
00204     limitCeilValue(length,10);
00205     if(num<0 && isSigned==true)
00206     {
00207         LCDSendChar('-');
00208         num *= -1;
00209         length--;
00210     }
00211     index = length;
00212     while(length>1)
00213     {
00214         multiple *= 10;
00215         length--;
00216     }
00217     while(index >= 1)
00218     {
00219         out = (uint32_t) (num/multiple);
00220         num -= out*(multiple);
00221         if(out!=0)
00222             showZeros = true;
00223         if(out==0 && showZeros==false)
00224             out -= 16;
00225         LCDSendChar(out+48);
00226         multiple /= 10;
```

```
00227         index--;
00228     }
00229 }
00230
00231
00232
00233 /*
00234  * Prints a number, from a variable, to the LCD
00235  * PARAM: num, length, isSigned, showZeros
00236  * IF signed
00237  *      Limits to a max of 10 digits to a positive number
00238  *      Limits to a max of 9 digits to a negative number
00239  * ELSE
00240  *      limits to 10 chars
00241  * IF showzeros
00242  *      shows all leading zeros
00243  * else
00244  *      supresses zeros; places space instead
00245  */
00246 void LCDSendNum(int64_t num, uint8_t length, uint8_t isSigned, uint8_t showZeros)
00247 {
00248     uint8_t index =0;
00249     uint8_t out = ' ';
00250     uint64_t multiple = 1;
00251     limitCeilValue(length,10);
00252     if(num<0 && isSigned==true)
00253     {
00254         out = '-';
00255         num *= -1;
00256     }
00257     LCDSendChar(out);
00258     index = length;
00259     while(length>1)
00260     {
00261         multiple *= 10;
00262         length--;
00263     }
00264     while(index >= 1)
00265     {
00266         out = (uint32_t) (num/multiple);
00267         num -= out*(multiple);
00268         if(out!=0)
00269             showZeros = true;
00270         if(out==0 && showZeros==false)
00271             out -= 16;
00272         LCDSendChar(out+48);
00273         multiple /= 10;
00274         index--;
00275     }
00276 }
00277
00278
00279 /*
00280  * Sends numerical values to LCD
00281  * Values between 0 and base;
00282  * max base value is defined as 32 (32bit wide buses)
00283  */
00284 void LCDSendNumArray(uint8_t *index)
00285 {
00286     while(*index<33)
00287     {
00288         LCDSendChar(*index+'0');
00289         index++;
00290     }
00291 }
00292
00293 /*
00294  * Clears display
00295  * Updates LCDStatus
00296  */
00297 void LCDClear(void)
00298 {
00299     LCDSendCmd(0x01);
00300     LCD0Status.row=1;
00301     LCD0Status.col=1;
00302     LCDDelay(800);
00303 }
00304
00305 /*
00306  * Turns
00307  * LCD_DISPLAY_ON/OFF
00308  * LCD_CURSOR_ON/OFF
00309  * LCD_BLINK_ON/OFF
00310  */
00311 void LCDDisplayOn(uint8_t onOff)
00312 {
00313     LCD0Status.display = onOff;
```

```
00314     LCDSendCmd(onOff);
00315 }
00316
00317
00318
00319
00320 /*
00321  * Prints the value of the array in hex format
00322  * As HEX, it'll print in base 16
00323  * Takes out 2 leading digits
00324  */
00325 void LCDSendHex(uint8_t *array)
00326 {
00327     uint8_t offset, temp;
00328     LCDSendChar('0');
00329     LCDSendChar('x');
00330     array += 2;
00331     while(*array<=32)
00332     {
00333         temp = *array;
00334         if(temp>9)
00335         {
00336             temp -= 10;
00337             offset = 'A';
00338         }
00339         else
00340             offset = '0';
00341         LCDSendChar(temp+offset);
00342         array++;
00343     }
00344 }
00345
00346
00347
00348 //void LCDSendNum(long num, char length, uint8_t isSigned, uint8_t showZeros)
00349
00350 /*
00351  * Passes a number to a vector
00352  * num -> number
00353  * vector -> pointer to vector
00354  * base -> base of output (max: 32)
00355  *
00356  * Last number in vector output is 33
00357  */
00358 void numToArray(int32_t num, uint8_t *array, uint8_t length, uint16_t base)
00359 {
00360     uint16_t index =1;
00361     uint8_t out;
00362     uint64_t multiple = 1;
00363
00364     limitCeilValue(length, (unsigned char) 1<<64/base);
00365     limitCeilValue(length, maxLengthOut);
00366
00367
00368     //create multiple number
00369     while(index<length)
00370     {
00371         multiple *= base;
00372         index++;
00373     }
00374     //sort multiples
00375     while(index >= 1)
00376     {
00377         //determines the multiple
00378         out = (uint8_t) (num/multiple);
00379         //takes out multiple
00380         num -= out*(multiple);
00381
00382         //escreve no vetor, desloca indice
00383         *array = out;
00384         array++;
00385         multiple /= base;
00386         //change multiple position
00387         index--;
00388     }
00389     *array = 33;
00390 }
00391
00392 /*
00393  * registers special characteres
00394  * number -> from 0 to 7
00395  * *character -> first index to 8 bytes long vector
00396  *               scans char downward
00397  */
00398 void LCDRegisterSpecial(uint8_t number, uint8_t *character)
00399 {
00400     uint8_t scan=0, data=0;
```

```
00401     LCDSendCmd(0x40+(number<<3));
00402     do
00403     {
00404         data = *(character+scan);
00405         LCDDelay(640);
00406         LCDSendChar(data&0x1F);
00407         scan++;
00408     }
00409     while(scan<8);
00410     LCDDelay(320);
00411 }
00412
00413 /*
00414  * Shifts data on LCD Display
00415  */
00416 void LCDShift(uint8_t shift)
00417 {
00418     LCDSendCmd(shift|LCD_SHIFT);
00419 }
00420
00421 /*
00422  * Sends LCD cursor to home position
00423  * PARAM: none
00424  */
00425 void LCDHome(void)
00426 {
00427     LCDSendCmd(0x02);
00428     LCDDelay(1500);
00429 }
00430
00431
00432
00433 void arrayToNum(uint8_t *array, uint32_t *num, uint8_t base)
00434 {
00435     while(*array<33)
00436     {
00437         *num += *array * base;
00438         array++;
00439     }
00440 }
00441
00442
00443
00444
00445 void LCDSendVU(uint32_t num, uint32_t base)
00446 {
00447     uint8_t index, pass=1;
00448     num  = (unsigned int) num*(LCD_col_num*LCD_char_width)/base;
00449     while(num>0)
00450     {
00451         index = LCD_char_width;
00452         while(num<LCD_char_width)
00453         {
00454             index--;
00455             num++;
00456         }
00457         LCDSendChar(index);
00458         num -= LCD_char_width;
00459         pass++;
00460     }
00461     while(pass<=LCD_col_num)
00462     {
00463         pass++;
00464         LCDSendChar(0);
00465     }
00466 }
00467
00468
00469
00470
00471 #ifdef trueDefinedLCD
00472 #undef true
00473 #undef false
00474 #endif
00475
00476
00477
```

6.41 my_lib/lcd.h File Reference

```
#include "includeAll.h"
```

Data Structures

- struct LCDStatus

Macros

- #define LCD_splashscreen_row1 PROJECT_NAME

  geneartion of project name in LCD
- #define LCD_splashscreen_row2 ("rnm sys undvpd")

  creator's watermark
- #define LCD_splashscreen2_row1 __DATE__

  compile date, used as program version
- #define LCD_splashscreen2_row2 __TIME__

  compile time, used as program version
- #define LCDDelay(x) SysDelayUs(x)
- #define LCDPinSet(x, y) PinAddrSet(x, y)
- #define LCDPinClear(x, y) PinAddrClear(x,y)
- #define LCD_RS_High LCDPinSet(LCD_RS_Port, LCD_RS_Pin)
- #define LCD_RS_Low LCDPinClear(LCD_RS_Port, LCD_RS_Pin)
- #define LCD_EN_High LCDPinSet(LCD_EN_Port, LCD_EN_Pin)
- #define LCD_EN_Low LCDPinClear(LCD_EN_Port, LCD_EN_Pin)
- #define LCD_CLK_High LCDPinSet(LCD_CLK_Port, LCD_CLK_Pin)
- #define LCD_CLK_LoW LCDPinClear(LCD_CLK_Port, LCD_CLK_Pin)
- #define LCD_DTA_Send(text)
- #define LCD_DISPLAY_ON 0x0C
- #define LCD_DISPLAY_OFF 0x08
- #define LCD_CURSOR_ON 0x0A
- #define LCD_CURSOR_OFF 0x08
- #define LCD_BLINK_ON 0x09
- #define LCD_BLINK_OFF 0x08
- #define LCD_SHIFT 0x10
- #define LCD_SHIFT_DISPLAY 0x08
- #define LCD_SHIFT_CURSOR 0x02
- #define LCD_SHIFT_RIGHT 0x04
- #define LCD_SHIFT_LEFT 0x00
- #define LCD_SET_CGRAM 0x40
- #define LCD_INCREMENT 0X04
- #define LCD_INCREMENT_NO_SHIFT 0x00
- #define LCD_INCREMENT_SHIFT 0x01
- #define LCD_INCREMENT_POSITIVE 0x02
- #define LCD_INCREMENT_NEGATIVE 0x00
- #define LCD_DISPLAY_CONFIG (LCD_DISPLAY_ON|LCD_CURSOR_OFF|LCD_BLINK_OFF)
- #define LCD_DISPLAY_INCREMENT (LCD_INCREMENT|LCD_INCREMENT_NO_SHIFT)
- #define maxLengthOut 16

Functions

- void LCDInit (void)

  Initializes the LCD Module.
- void LCDSendCmd (uint8_t cmd)
- void LCDSendChar (uint8_t txt)

  send single character to LCD.
- void LCDSend (uint8_t send)

Send data to LCD, no RS control.

- void LCDPosition (uint8_t row, uint8_t col)

  Set LCD write position.
- void LCDPositionNoDelay (uint8_t row, uint8_t col)

  Set LCD write position, no delay in function.
- void LCDSendString (uint8_t *string, uint8_t breakLine)

  Send string to LCD.
- void LCDSendNumStrict (int64_t num, uint8_t length, uint8_t isSigned, uint8_t showZeros)
- void LCDSendNum (int64_t num, uint8_t length, uint8_t isSigned, uint8_t showZeros)
- void LCDSendNumArray (uint8_t *vector)
- void LCDClear (void)
- void LCDSendHex (uint8_t *array)
- void numToArray (int32_t num, uint8_t *array, uint8_t length, uint16_t base)
- void LCDRegisterSpecial (uint8_t number, uint8_t *character)
- void LCDShift (uint8_t shift)
- void LCDHome (void)
- void arrayToNum (uint8_t *array, uint32_t *num, uint8_t base)
- void LCDSendVU (uint32_t num, uint32_t base)

Variables

- LCDStatus LCD0Status

6.41.1 Macro Definition Documentation

6.41.1.1 #define LCD_BLINK_OFF 0x08

Definition at line 99 of file lcd.h.

6.41.1.2 #define LCD_BLINK_ON 0x09

Definition at line 98 of file lcd.h.

6.41.1.3 #define LCD_CLK_High LCDPinSet(LCD_CLK_Port, LCD_CLK_Pin)

Definition at line 85 of file lcd.h.

6.41.1.4 #define LCD_CLK_LoW LCDPinClear(LCD_CLK_Port, LCD_CLK_Pin)

Definition at line 86 of file lcd.h.

6.41.1.5 #define LCD_CURSOR_OFF 0x08

Definition at line 97 of file lcd.h.

6.41.1.6 #define LCD_CURSOR_ON 0x0A

Definition at line 96 of file lcd.h.

6.41.1.7 #define LCD_DISPLAY_CONFIG (LCD_DISPLAY_ON|LCD_CURSOR_OFF|LCD_BLINK_OFF)

Definition at line 114 of file lcd.h.

6.41.1.8 #define LCD_DISPLAY_INCREMENT (LCD_INCREMENT|LCD_INCREMENT_NO_SHIFT)

Definition at line 115 of file lcd.h.

6.41.1.9 #define LCD_DISPLAY_OFF 0x08

Definition at line 95 of file lcd.h.

6.41.1.10 #define LCD_DISPLAY_ON 0x0C

Definition at line 94 of file lcd.h.

6.41.1.11 #define LCD_DTA_Send(*text*)

Value:

```
ShiftSerialSend(LCD_DTA_Port,\
                                LCD_DTA_Pin,\
                                LCD_CLK_Port,\
                                LCD_CLK_Pin, text)
```

Definition at line 87 of file lcd.h.

6.41.1.12 #define LCD_EN_High LCDPinSet(LCD_EN_Port, LCD_EN_Pin)

Definition at line 83 of file lcd.h.

6.41.1.13 #define LCD_EN_Low LCDPinClear(LCD_EN_Port, LCD_EN_Pin)

Definition at line 84 of file lcd.h.

6.41.1.14 #define LCD_INCREMENT 0X04

Definition at line 106 of file lcd.h.

6.41.1.15 #define LCD_INCREMENT_NEGATIVE 0x00

Definition at line 110 of file lcd.h.

6.41.1.16 #define LCD_INCREMENT_NO_SHIFT 0x00

Definition at line 107 of file lcd.h.

6.41.1.17 #define LCD_INCREMENT_POSITIVE 0x02

Definition at line 109 of file lcd.h.

6.41.1.18 #define LCD_INCREMENT_SHIFT 0x01

Definition at line 108 of file lcd.h.

6.41.1.19 #define LCD_RS_High LCDPinSet(LCD_RS_Port, LCD_RS_Pin)

Definition at line 81 of file lcd.h.

6.41.1.20 #define LCD_RS_Low LCDPinClear(LCD_RS_Port, LCD_RS_Pin)

Definition at line 82 of file lcd.h.

6.41.1.21 #define LCD_SET_CGRAM 0x40

Definition at line 105 of file lcd.h.

6.41.1.22 #define LCD_SHIFT 0x10

Definition at line 100 of file lcd.h.

6.41.1.23 #define LCD_SHIFT_CURSOR 0x02

Definition at line 102 of file lcd.h.

6.41.1.24 #define LCD_SHIFT_DISPLAY 0x08

Definition at line 101 of file lcd.h.

6.41.1.25 #define LCD_SHIFT_LEFT 0x00

Definition at line 104 of file lcd.h.

6.41.1.26 #define LCD_SHIFT_RIGHT 0x04

Definition at line 103 of file lcd.h.

6.41.1.27 #define LCD_splashscreen2_row1 __DATE__

compile date, used as program version

Definition at line 11 of file lcd.h.

6.41.1.28 #define LCD_splashscreen2_row2 __TIME__

compile time, used as program version

Definition at line 12 of file lcd.h.

6.41.1.29 #define LCD_splashscreen_row1 PROJECT_NAME

geneartion of project name in LCD

Definition at line 8 of file lcd.h.

6.41.1.30 #define LCD_splashscreen_row2 ("rnm sys undvpd")

creator's watermark

Definition at line 9 of file lcd.h.

6.41.1.31 #define LCDDelay(*x*) SysDelayUs(x)

Definition at line 74 of file lcd.h.

6.41.1.32 #define LCDPinClear(*x, y*) PinAddrClear(x,y)

Definition at line 76 of file lcd.h.

6.41.1.33 #define LCDPinSet(*x, y*) PinAddrSet(x, y)

Definition at line 75 of file lcd.h.

6.41.1.34 #define maxLengthOut 16

Definition at line 153 of file lcd.h.

6.41.2 Function Documentation

6.41.2.1 void arrayToNum (uint8_t * *array,* uint32_t * *num,* uint8_t *base*)

Definition at line 433 of file lcd.c.

6.41.2.2 void LCDClear (void)

Definition at line 297 of file lcd.c.

6.41.2.3 void LCDHome (void)

Definition at line 425 of file lcd.c.

6.41.2.4 void LCDInit (void)

Initializes the LCD Module.

Called once at startup. Takes no parameters.

Returns

None.

Definition at line 47 of file lcd.c.

6.41.2.5 void LCDPosition (uint8_t *row*, uint8_t *col*)

Set LCD write position.

Parameters

row	uint8_t row.
col	uint8_t column.

Definition at line 118 of file lcd.c.

6.41.2.6 void LCDPositionNoDelay (uint8_t *row*, uint8_t *col*)

Set LCD write position, no delay in function.

Parameters

row	uint8_t row.
col	uint8_t column.

Definition at line 138 of file lcd.c.

6.41.2.7 void LCDRegisterSpecial (uint8_t *number*, uint8_t * *character*)

Definition at line 398 of file lcd.c.

6.41.2.8 void LCDSend (uint8_t *send*)

Send data to LCD, no RS control.

Parameters

send	uint8_t data to be sent.

Definition at line 102 of file lcd.c.

6.41.2.9 void LCDSendChar (uint8_t *txt*)

send single character to LCD.

Parameters

txt	uint8_t type data to be sent, 8 bits.

Definition at line 89 of file lcd.c.

6.41.2.10 void LCDSendCmd (uint8_t *cmd*)

Definition at line 77 of file lcd.c.

6.41.2.11 void LCDSendHex (uint8_t * *array*)

Definition at line 325 of file lcd.c.

6.41.2.12 void LCDSendNum (int64_t *num*, uint8_t *length*, uint8_t *isSigned*, uint8_t *showZeros*)

Definition at line 246 of file lcd.c.

6.41.2.13 void LCDSendNumArray (uint8_t * *vector*)

Definition at line 284 of file lcd.c.

6.41.2.14 void LCDSendNumStrict (int64_t *num*, uint8_t *length*, uint8_t *isSigned*, uint8_t *showZeros*)

Definition at line 199 of file lcd.c.

6.41.2.15 void LCDSendString (uint8_t * *string*, uint8_t *breakLine*)

Send string to LCD.

Parameters

**string*	uint8_t string to be sent.
breakLine	uint8_t break line at the end of LCD length.

Writes a string of characteres on display Processes according to the ASCII code 0 - NULL

Definition at line 162 of file lcd.c.

6.41.2.16 void LCDSendVU (uint32_t *num*, uint32_t *base*)

Definition at line 445 of file lcd.c.

6.41.2.17 void LCDShift (uint8_t *shift*)

Definition at line 416 of file lcd.c.

6.41.2.18 void numToArray (int32_t *num*, uint8_t * *array*, uint8_t *length*, uint16_t *base*)

Definition at line 358 of file lcd.c.

6.41.3 Variable Documentation

6.41.3.1 LCDStatus LCD0Status

6.42 lcd.h

```
00001 #ifndef lcd_h
00002 #define lcd_h
00003
00004
00005 #include "includeAll.h"
00006
00007
00008 #define LCD_splashscreen_row1   PROJECT_NAME            //!< geneartion of project name in LCD
00009 #define LCD_splashscreen_row2   ("rnm sys undvpd")      //!< creator's watermark
00010
00011 #define LCD_splashscreen2_row1  __DATE__                //!< compile date, used as program version
00012 #define LCD_splashscreen2_row2  __TIME__                //!< compile time, used as program version
00013
00014
00015
00016 /*
00017 NEED TO DECLARE
00018
00019 //LCD
00020 #define LCD_RS          J1_05   //E5
00021 #define LCD_RS_Port     GPIO_PORTE_BASE
00022 #define LCD_RS_Pin      GPIO_PIN_5
00023
00024 #define LCD_EN          J1_06   //E4
00025 #define LCD_EN_Port     GPIO_PORTE_BASE
00026 #define LCD_EN_Pin      GPIO_PIN_4
00027
00028 #define LCD_DTA         J2_09   //A2
00029 #define LCD_DTA_Port    GPIO_PORTA_BASE
00030 #define LCD_DTA_Pin     GPIO_PIN_3
00031
```

```
00032 #define LCD_CLK         J2_10   //A3
00033 #define LCD_CLK_Port    GPIO_PORTA_BASE
00034 #define LCD_CLK_Pin     GPIO_PIN_2
00035
00036 #define LCD_row_num     2
00037 #define LCD_col_num     16
00038
00039 LCDStatus LCD0Status;
00040
00041
00042 #define LCD_splashscreen_row1   ("odqd")
00043 #define LCD_splashscreen_row2   ("rnm sys undvpd")
00044
00045
00046
00047 uint8_t specialChar[8][8] = {   //ultima coluna, linha de baixo, reservada para cursor
00048         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x07, 0x00, 0x00,\
00049         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x06, 0x01, 0x00,\
00050         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x05, 0x02, 0x00,\
00051         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x04, 0x03, 0x00,\
00052         0x1F, 0x0F, 0x07, 0x03, 0x00, 0x03, 0x04, 0x00,\
00053         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x02, 0x05, 0x00,\
00054         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x01, 0x06, 0x00,\
00055         0x1F, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x07, 0x00
00056         };
00057
00058
00059
00060 */
00061
00062
00063
00064 /*
00065  * LIBRARY FOR LCD USE
00066  * SERIAL COMM
00067  * USE OF SHIFT REGISTERS FOR DATA
00068  */
00069
00070
00071
00072 //sub-function masks
00073 //External Function Masks
00074 #define LCDDelay(x)             SysDelayUs(x)
00075 #define LCDPinSet(x, y)         PinAddrSet(x, y)
00076 #define LCDPinClear(x, y)       PinAddrClear(x,y)
00077
00078
00079
00080 //sub-function masks
00081 #define LCD_RS_High             LCDPinSet(LCD_RS_Port, LCD_RS_Pin)
00082 #define LCD_RS_Low              LCDPinClear(LCD_RS_Port, LCD_RS_Pin)
00083 #define LCD_EN_High             LCDPinSet(LCD_EN_Port, LCD_EN_Pin)
00084 #define LCD_EN_Low              LCDPinClear(LCD_EN_Port, LCD_EN_Pin)
00085 #define LCD_CLK_High            LCDPinSet(LCD_CLK_Port, LCD_CLK_Pin)
00086 #define LCD_CLK_LoW             LCDPinClear(LCD_CLK_Port, LCD_CLK_Pin)
00087 #define LCD_DTA_Send(text)      ShiftSerialSend(LCD_DTA_Port,\
00088                                 LCD_DTA_Pin,\
00089                                 LCD_CLK_Port,\
00090                                 LCD_CLK_Pin, text)
00091
00092
00093 //LCD Command Masks
00094 #define LCD_DISPLAY_ON          0x0C
00095 #define LCD_DISPLAY_OFF         0x08
00096 #define LCD_CURSOR_ON           0x0A
00097 #define LCD_CURSOR_OFF          0x08
00098 #define LCD_BLINK_ON            0x09
00099 #define LCD_BLINK_OFF           0x08
00100 #define LCD_SHIFT               0x10
00101 #define LCD_SHIFT_DISPLAY       0x08
00102 #define LCD_SHIFT_CURSOR        0x02
00103 #define LCD_SHIFT_RIGHT         0x04
00104 #define LCD_SHIFT_LEFT          0x00
00105 #define LCD_SET_CGRAM           0x40
00106 #define LCD_INCREMENT           0X04
00107 #define LCD_INCREMENT_NO_SHIFT  0x00
00108 #define LCD_INCREMENT_SHIFT     0x01
00109 #define LCD_INCREMENT_POSITIVE  0x02
00110 #define LCD_INCREMENT_NEGATIVE  0x00
00111
00112
00113 //LCD Command Initial State - Config
00114 #define LCD_DISPLAY_CONFIG      (LCD_DISPLAY_ON|LCD_CURSOR_OFF|LCD_BLINK_OFF)
00115 #define LCD_DISPLAY_INCREMENT   (LCD_INCREMENT|LCD_INCREMENT_NO_SHIFT)
00116
00117
00118 typedef struct
```

```
00119 {
00120     uint8_t row;
00121     uint8_t col;
00122     uint8_t display;
00123     uint8_t shift;
00124     uint8_t cgramAdress;
00125     uint8_t specialChar[8];     //defined by fonts
00126 }LCDStatus;
00127
00128
00129
00130 // \TODO: fix this shit. decide if is to be used with structs or no use at all.
00131 extern LCDStatus   LCD0Status;
00132
00133
00134
00135
00136 //functions declarations
00137 extern void LCDInit(void);
00138 extern void LCDSendCmd(uint8_t cmd);
00139 extern void LCDSendChar(uint8_t txt);
00140 extern void LCDSend(uint8_t send);
00141 extern void LCDPosition(uint8_t row, uint8_t col);
00142 extern void LCDPositionNoDelay(uint8_t row, uint8_t col);
00143 extern void LCDSendString(uint8_t *string, uint8_t breakLine);
00144 extern void LCDSendNumStrict(int64_t num, uint8_t length,\
00145         uint8_t isSigned, uint8_t showZeros);
00146 extern void LCDSendNum(int64_t num, uint8_t length,\
00147         uint8_t isSigned, uint8_t showZeros);
00148 extern void LCDSendNumArray(uint8_t *vector);
00149 extern void LCDClear(void);
00150 extern void LCDSendHex(uint8_t *array);
00151
00152 //limited by uint64 max counting
00153 #define maxLengthOut    16
00154 extern void numToArray(int32_t num, uint8_t *array,\
00155                 uint8_t length, uint16_t base);
00156
00157 extern void LCDRegisterSpecial(uint8_t number,\
00158                         uint8_t *character);
00159 extern void LCDShift(uint8_t shift);
00160 extern void LCDHome(void);
00161
00162
00163 extern void arrayToNum(uint8_t *array, uint32_t *num, uint8_t base);
00164
00165
00166 extern void LCDSendVU(uint32_t num, uint32_t base);
00167
00168 #endif
```

6.43 my_lib/my_use.c File Reference

```
#include "my_use.h"
```

Functions

- void __inline ShiftSerialSend (uint32_t data_port, uint32_t data_pin, uint32_t clk_port, uint32_t clk_pin, uint8_t text)

6.43.1 Function Documentation

6.43.1.1 void __inline ShiftSerialSend (uint32_t *data_port,* uint32_t *data_pin,* uint32_t *clk_port,* uint32_t *clk_pin,* uint8_t *text*)

Definition at line 9 of file my_use.c.

6.44 my_use.c

```
00001 #include "my_use.h"
```

```
00002
00003
00004
00005
00006 /*
00007  * Shift Serial Send function
00008  */
00009 void __inline ShiftSerialSend(uint32_t data_port, uint32_t data_pin, \
00010                     uint32_t clk_port, uint32_t clk_pin, uint8_t text)
00011 {
00012
00013     PinAddrClear(clk_port, clk_pin);
00014     char i=8;
00015     while(i>0)
00016     {
00017         if((text&0x80)==0)
00018             PinAddrClear(data_port, data_pin);
00019         else
00020             PinAddrSet(data_port, data_pin);
00021         text <<= 1;
00022         i--;
00023         PinAddrSet(clk_port, clk_pin);
00024         SysDelay(2);
00025         PinAddrClear(clk_port, clk_pin);
00026         SysDelay(2);
00027     }
00028
00029 }
00030
00031
00032
00033
00034
```

6.45 my_lib/my_use.h File Reference

```
#include "depl_spc/includeAll_sw.h"
#include "depl_spc/includeAll_hw.h"
```

Macros

- #define bTrue0 0x01
- #define bTrue1 0x02
- #define bTrue2 0x04
- #define bTrue3 0x08
- #define bTrue4 0x10
- #define bTrue5 0x20
- #define bTrue6 0x40
- #define bTrue7 0x80
- #define charDecadeLength 3
- #define charBinaryLength 8
- #define shortDecadeLength 5
- #define shortBinaryLength 16
- #define intDecadeLength 10
- #define intBinaryLength 32
- #define limitCeilValue(value, lim)
- #define limitCycleValueUpZero(value, lim)
- #define limitCycleValueUpOff(value, lim, reset)
- #define limitFloorValue(value, lim)

Functions

- void ShiftSerialSend (uint32_t data_port, uint32_t data_pin, uint32_t clk_port, uint32_t clk_pin, uint8_t text)

6.45.1 Macro Definition Documentation

6.45.1.1 #define bTrue0 0x01

Definition at line 7 of file my_use.h.

6.45.1.2 #define bTrue1 0x02

Definition at line 8 of file my_use.h.

6.45.1.3 #define bTrue2 0x04

Definition at line 9 of file my_use.h.

6.45.1.4 #define bTrue3 0x08

Definition at line 10 of file my_use.h.

6.45.1.5 #define bTrue4 0x10

Definition at line 11 of file my_use.h.

6.45.1.6 #define bTrue5 0x20

Definition at line 12 of file my_use.h.

6.45.1.7 #define bTrue6 0x40

Definition at line 13 of file my_use.h.

6.45.1.8 #define bTrue7 0x80

Definition at line 14 of file my_use.h.

6.45.1.9 #define charBinaryLength 8

Definition at line 17 of file my_use.h.

6.45.1.10 #define charDecadeLength 3

Definition at line 16 of file my_use.h.

6.45.1.11 #define intBinaryLength 32

Definition at line 23 of file my_use.h.

6.45.1.12 #define intDecadeLength 10

Definition at line 22 of file my_use.h.

6.45.1.13 #define limitCeilValue(*value, lim*)

Value:

```
if(value>=lim)\
                                                value=lim;
```

Definition at line 28 of file my_use.h.

6.45.1.14 #define limitCycleValueUpOff(*value, lim, reset*)

Value:

```
if(value>=lim)\
                                                value=reset;
```

Definition at line 32 of file my_use.h.

6.45.1.15 #define limitCycleValueUpZero(*value, lim*)

Value:

```
if(value>=lim)\
                                                value=0;
```

Definition at line 30 of file my_use.h.

6.45.1.16 #define limitFloorValue(*value, lim*)

Value:

```
if(value<=lim)\
                                                value=lim;
```

Definition at line 34 of file my_use.h.

6.45.1.17 #define shortBinaryLength 16

Definition at line 20 of file my_use.h.

6.45.1.18 #define shortDecadeLength 5

Definition at line 19 of file my_use.h.

6.45.2 Function Documentation

6.45.2.1 void ShiftSerialSend (uint32_t *data_port,* uint32_t *data_pin,* uint32_t *clk_port,* uint32_t *clk_pin,* uint8_t *text*)

Definition at line 9 of file my_use.c.

6.46 my_use.h

```
00001 #ifndef my_use_h
00002 #define my_use_h
00003
00004 #include "depl_spc/includeAll_sw.h"
00005 #include "depl_spc/includeAll_hw.h"
00006
00007 #define bTrue0              0x01
00008 #define bTrue1              0x02
00009 #define bTrue2              0x04
00010 #define bTrue3              0x08
00011 #define bTrue4              0x10
00012 #define bTrue5              0x20
00013 #define bTrue6              0x40
00014 #define bTrue7              0x80
00015
00016 #define charDecadeLength    3
00017 #define charBinaryLength    8
00018
00019 #define shortDecadeLength   5
00020 #define shortBinaryLength   16
00021
00022 #define intDecadeLength     10
00023 #define intBinaryLength     32
00024
00025
00026
00027 //function masks
00028 #define limitCeilValue(value, lim)                  if(value>=lim)\
00029                                                     value=lim;
00030 #define limitCycleValueUpZero(value,lim)            if(value>=lim)\
00031                                                     value=0;
00032 #define limitCycleValueUpOff(value, lim, reset)     if(value>=lim)\
00033                                                     value=reset;
00034 #define limitFloorValue(value, lim)                 if(value<=lim)\
00035                                                     value=lim;
00036
00037
00038
00039
00040 /*
00041  * Function Declarations
00042  */
00043 void ShiftSerialSend(uint32_t data_port, uint32_t data_pin, \
00044         uint32_t clk_port, uint32_t clk_pin, uint8_t text);
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054 #endif// my_use_h
```

6.47 my_lib/myUart.c File Reference

```
#include "myUart.h"
```

Functions

- void myUARTSendString (uint32_t instance, uint8_t *string)

6.47.1 Function Documentation

6.47.1.1 void myUARTSendString (uint32_t *instance*, uint8_t * *string*)

Definition at line 17 of file myUart.c.

6.48 myUart.c

```
00001 /*
00002  * myUart.c
00003  *
00004  *  Created on: Nov 25, 2013
00005  *      Author: rikardo
00006  */
00007
00008
00009 #include "myUart.h"
00010
00011
00012
00013 /*
00014  * Sends a string, NULL terminator
00015  * returns none
00016  */
00017 void myUARTSendString(uint32_t instance, uint8_t *string)
00018 {
00019     while(*string)
00020     {
00021         myUARTSend(instance, *string);
00022         string++;
00023         myUARTDelay(1);
00024     }
00025 }
00026
```

6.49 my_lib/myUart.h File Reference

```
#include "includeAll.h"
```

Data Structures

- struct UARTInstance

Macros

- #define UART_NORMAL_OP_MODE 0x0001
- #define UART_DIRECT_TRANSFER_MODE 0x0002
- #define UART_BUFFER_SIZE 30
- #define myUARTPC UART0_BASE
- #define myUARTSend(instance, charToGo) MAP_UARTCharPutNonBlocking(instance, charToGo)
- #define myUARTDelay(delay) SysDelay(delay)
- #define myUARTPCSend(charToGo) myUARTSend(myUARTPC, charToGo)

Functions

- void myUARTSendString (uint32_t instance, uint8_t *string)

6.49.1 Macro Definition Documentation

6.49.1.1 #define myUARTDelay(*delay*) SysDelay(delay)

Definition at line 37 of file myUart.h.

6.49.1.2 #define myUARTPC UART0_BASE

Definition at line 34 of file myUart.h.

6.49.1.3 #define myUARTPCSend(*charToGo*) myUARTSend(myUARTPC, charToGo)

Definition at line 38 of file myUart.h.

6.49.1.4 #define myUARTSend(*instance, charToGo*) MAP_UARTCharPutNonBlocking(instance, charToGo)

Definition at line 36 of file myUart.h.

6.49.1.5 #define UART_BUFFER_SIZE 30

Definition at line 21 of file myUart.h.

6.49.1.6 #define UART_DIRECT_TRANSFER_MODE 0x0002

Definition at line 17 of file myUart.h.

6.49.1.7 #define UART_NORMAL_OP_MODE 0x0001

Definition at line 16 of file myUart.h.

6.49.2 Function Documentation

6.49.2.1 void myUARTSendString (uint32_t *instance,* uint8_t * *string*)

Definition at line 17 of file myUart.c.

6.50 myUart.h

```
00001 /*
00002  * myUart.h
00003  *
00004  *  Created on: Nov 25, 2013
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef MYUART_H_
00009 #define MYUART_H_
00010
00011 #include "includeAll.h"
00012
00013 /*
00014  * UART operation mode masks
00015  */
00016 #define UART_NORMAL_OP_MODE                0x0001
00017 #define UART_DIRECT_TRANSFER_MODE          0x0002
00018
00019
00020
00021 #define UART_BUFFER_SIZE    30
00022
00023 typedef struct{
00024     uint8_t RxBuffer[UART_BUFFER_SIZE];
00025     uint8_t RxBufferPtr;
00026     uint8_t TxBuffer[UART_BUFFER_SIZE];
00027     uint8_t TxBufferPtr;
00028     uint16_t Mode;
00029     uint8_t TxLastSent[UART_BUFFER_SIZE];
00030     uint8_t TxLastSentPtr;
00031 }UARTInstance;
00032
00033
00034 #define myUARTPC                           UART0_BASE
00035
00036 #define myUARTSend(instance, charToGo)     MAP_UARTCharPutNonBlocking(instance, charToGo)
00037 #define myUARTDelay(delay)                 SysDelay(delay)
```

```
00038 #define myUARTPCSend(charToGo)              myUARTSend(myUARTPC, charToGo)
00039
00040
00041
00042 void myUARTSendString(uint32_t instance, uint8_t *string);
00043
00044
00045
00046
00047 #endif /* MYUART_H_ */
```

6.51 my_lib/uk_mapping.h File Reference

Macros

- #define UKM_SPACE 32
- #define UKM_BSPACE 127
- #define UKM_BS 8
- #define UKM_ENTER 13
- #define UKM_TILDA 126
- #define UKM_ESCAPE 27
- #define UKM_TAB 9
- #define UKM_CTRL_E 5
- #define UKM_ASCII_TAB 9
- #define UKM_ASCII_LF 10
- #define UKM_LF 10
- #define UKM_LINEFEED 10
- #define UKM_CR 13
- #define UKM_ASCII_VT 11
- #define UKM_VT 11
- #define UKM_ASCII_FF 12
- #define UKM_CLS 12

6.51.1 Macro Definition Documentation

6.51.1.1 #define UKM_ASCII_FF 12

Definition at line 28 of file uk_mapping.h.

6.51.1.2 #define UKM_ASCII_LF 10

Definition at line 22 of file uk_mapping.h.

6.51.1.3 #define UKM_ASCII_TAB 9

Definition at line 21 of file uk_mapping.h.

6.51.1.4 #define UKM_ASCII_VT 11

Definition at line 26 of file uk_mapping.h.

6.51.1.5 #define UKM_BS 8

Definition at line 15 of file uk_mapping.h.

6.51.1.6 #define UKM_BSPACE 127

Definition at line 14 of file uk_mapping.h.

6.51.1.7 #define UKM_CLS 12

Definition at line 29 of file uk_mapping.h.

6.51.1.8 #define UKM_CR 13

Definition at line 25 of file uk_mapping.h.

6.51.1.9 #define UKM_CTRL_E 5

Definition at line 20 of file uk_mapping.h.

6.51.1.10 #define UKM_ENTER 13

Definition at line 16 of file uk_mapping.h.

6.51.1.11 #define UKM_ESCAPE 27

Definition at line 18 of file uk_mapping.h.

6.51.1.12 #define UKM_LF 10

Definition at line 23 of file uk_mapping.h.

6.51.1.13 #define UKM_LINEFEED 10

Definition at line 24 of file uk_mapping.h.

6.51.1.14 #define UKM_SPACE 32

Definition at line 13 of file uk_mapping.h.

6.51.1.15 #define UKM_TAB 9

Definition at line 19 of file uk_mapping.h.

6.51.1.16 #define UKM_TILDA 126

Definition at line 17 of file uk_mapping.h.

6.51.1.17 #define UKM_VT 11

Definition at line 27 of file uk_mapping.h.

6.52 uk_mapping.h

```
00001 /*
00002  * uart_keyboard_mapping.h
00003  *
00004  *  Created on: Nov 27, 2013
00005  *      Author: rikardo
00006  */
00007
00008 #ifndef UART_KEYBOARD_MAPPING_H_
00009 #define UART_KEYBOARD_MAPPING_H_
00010
00011
00012
00013 #define UKM_SPACE           32
00014 #define UKM_BSPACE          127
00015 #define UKM_BS              8
00016 #define UKM_ENTER           13
00017 #define UKM_TILDA           126
00018 #define UKM_ESCAPE          27
00019 #define UKM_TAB             9
00020 #define UKM_CTRL_E          5
00021 #define UKM_ASCII_TAB       9
00022 #define UKM_ASCII_LF        10
00023 #define UKM_LF              10
00024 #define UKM_LINEFEED        10
00025 #define UKM_CR              13
00026 #define UKM_ASCII_VT        11
00027 #define UKM_VT              11
00028 #define UKM_ASCII_FF        12
00029 #define UKM_CLS             12
00030
00031
00032
00033 #endif /* UART_KEYBOARD_MAPPING_H_ */
```

6.53 README 1.md File Reference

6.54 README 1.md

```
00001 uIntPLib
00002 ========
00003
00004 Universal Integrated Peripheral Library
00005
00006 This is a library made with functions masks to medium level programming.
00007 Intended to make code more portable, while maintaning its performance.
00008
00009
00010 Doxyen generated documentation is located at latex/refman.pdf
00011 Complete documentation is under construction.
00012
```

6.55 README.md File Reference

6.56 README.md

```
00001 uIntPLib
00002 ========
00003
00004 Universal Integrated Peripheral Library
00005
00006 This is a library made with functions masks to medium level programming.
00007 Intended to make code more portable, while maintaning its performance.
00008
00009
00010 Doxyen generated documentation is located at latex/refman.pdf
00011 Complete documentation is under construction.
00012
```

Index