Sistema de Monitoramento Embarcado

Generated by Doxygen 1.8.11

Contents

Chapter 1

Sistema de Monitoramento Embarcado (SIMONE)

1.1 Introdução

O software embarcado (firmware) foi projetado seguindo modelo de camadas convencional, totalizando 4 camadas:

- 1. Camada HAL drivers - camada de abstração de hardware (HAL) que contém drivers para acesso aos periféricos do controlador. Os seguintes periféricos são utilizados.
  - a. GPIO – entradas e saídas digitais de propósito geral. Utilizado para acionamento de LEDs e leitura de sensores.
  - b. SPI – comunicação serial síncrona para periféricos, como cartão SD.
  - c. UART - comunicação serial assíncrona para comunicação com periféricos RS485 e modem.
  - d. USB – comunicação serial universal para comunicação com computador por porta USB.
  - e. A/D – entradas analógicas com conversão para valores digitais.
  - f. Timer – contador de tempo para funções de temporização, como relógio do sistema, alarmes e atrasos.
- 2. Camada de dispositivos (devices) – contém as implementações para acesso através de drivers aos dispositivos periféricos externos, como:
  - cartão SD
  - RS485
  - modem/comandos AT
  - LEDs
  - sensores.
- 3. Camada de sistema – contém as implementações relativas ao sistema operacional de tempo real (RTOS) e bibliotecas de middleware para:
  - sistemas de arquivos (FAT)
  - protocolo Modbus RTU
  - protocolo HTTP para comunicação com sistema de monitoramento (SIMON)
- 4. Camada de tarefas/aplicações – contém as implementações da lógica do sistema de monitoramento, incluindo:
  - configuração
  - terminal de comandos
  - sincronização
  - relógio
  - leitura dos equipamentos e sensores, processamento, armazenamento e transmissão de dados.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

Module Documentation

4.1 LEDs

Macros

- #define **LED_ON** 1
- #define **LED_OFF** 0

Enumerations

- enum **led_color_t** { **RED_LED** =4, **YELLOW_LED** =5, **GREEN_LED** =6 }

Functions

- void **led_onboard_init** (void)
- int **led_onboard_state** (led_color_t led_color)
- void **led_onboard_on** (led_color_t led_color)
- void **led_onboard_off** (led_color_t led_color)
- void **led_onboard_toggle** (led_color_t led_color)

4.1.1 Detailed Description

4.2 MCU

Functions

- void **Mcu_Init** (void)
- void **MCG_Init** (void)
- void **System_Init** (void)

4.2.1 Detailed Description

4.3 SPI

Files

- file spi.h

  Serial peripheral interface driver function prototypes.

Macros

- #define **ENABLE_SPI1** TRUE
- #define **ENABLE_SPI2** FALSE

Functions

- void **init_SPI** (unsigned char spi)
- void **SPI1_SendChar** (unsigned char data)
- unsigned char **SPI1_GetChar** (void)
- void **SPI2_SendChar** (unsigned char data)
- unsigned char **SPI2_GetChar** (void)

4.3.1 Detailed Description

4.4 UART

Files

- file uart.h

  Rotinas para transferir e receber dados via UART.

Macros

- #define **BAUD**(x) ((configCPU_CLOCK_HZ/16/(x)) - 1)
- #define **ENABLE_UART0** TRUE
- #define **ENABLE_UART1** TRUE
- #define **ENABLE_UART2** TRUE
- #define **UART0** 0
- #define **UART1** 1
- #define **UART2** 2
- #define **UART0_MUTEX** 1
- #define **UART1_MUTEX** 0
- #define **UART2_MUTEX** 0
- #define **UART0_MUTEX_PRIO** 9
- #define **UART1_MUTEX_PRIO** 10
- #define **UART2_MUTEX_PRIO** 11
- #define CR 13

  ASCII code for carry return.
- #define LF 10

  ASCII code for line feed.
- #define TX_TIMEOUT 5

  timeout in miliseconds for characters transmission

Functions

- void **uart_init** (INT8U uart, INT16U baudrate, INT16U buffersize, INT8U mutex, INT8U priority)
- void **SerialReset** (INT8U Comm)
- void **uart0_acquire** (void)
- void **uart0_release** (void)
- char **putchar_uart0** (char caracter)
- INT8U **getchar_uart0** (char ∗caracter, INT16U timeout)
- void **printf_uart0** (char ∗string)
- void **printP_uart0** (char const ∗string)
- void **uart0_tx** (void)
- void **uart0_rx** (void)
- void **uart0_error** (void)
- void **uart0_RxEnable** (void)
- void **uart0_RxDisable** (void)
- void **uart0_RxEnableISR** (void)
- void **uart0_RxDisableISR** (void)
- void **uart0_TxEnableISR** (void)
- void **uart0_TxDisableISR** (void)
- void **uart1_acquire** (void)
- void **uart1_release** (void)
- char **putchar_uart1** (char caracter)

- void **printf_uart1** (char ∗string)
- void **printP_uart1** (char const ∗string)
- void uart1_tx (void)

  ISR para transmissao de dados.
- void uart1_rx (void)

  ISR para recepcao de dados.
- void **uart1_error** (void)
- void **uart1_RxEnable** (void)
- void **uart1_RxDisable** (void)
- void **uart1_RxEnableISR** (void)
- void **uart1_RxDisableISR** (void)
- void **uart1_TxEnableISR** (void)
- void **uart1_TxDisableISR** (void)
- void **uart2_acquire** (void)
- void **uart2_release** (void)
- char **putchar_uart2** (char caracter)
- void **printf_uart2** (char ∗string)
- void **printP_uart2** (char const ∗string)
- void uart2_tx (void)

  ISR para transmissao de dados.
- void uart2_rx (void)

  ISR para recepcao de dados.
- void **uart2_error** (void)
- void **uart2_RxEnableISR** (void)
- void **uart2_RxDisableISR** (void)
- void **uart2_TxEnableISR** (void)
- void **uart2_TxDisableISR** (void)

4.4.1 Detailed Description

4.5 Relógio do sistema

Macros

- #define **__ENABLE_WATCHDOG**()

Functions

- void **System_Time** (void)

4.5.1 Detailed Description

4.6 Terminal de Comandos

Macros

- #define **TERM_BUFSIZE** 36
- #define **TERM_MUTEX** TRUE
- #define **TERM_BAUDRATE** 19200
- #define **TERM_MUTEX_PRIO** UART0_MUTEX_PRIO
- #define **TERM_OUTPUT** putchar_uart0

Functions

- void **Terminal_Task** (void)
- void **term_cmd_ver** (char ∗param)
- void **term_cmd_top** (char ∗param)
- void **term_cmd_rst** (char ∗param)
- void **term_cmd_cat** (char ∗param)
- void **term_cmd_ls** (char ∗param)
- void **term_cmd_cd** (char ∗param)
- void **term_cmd_mount** (char ∗param)
- void **term_cmd_sr** (char ∗param)
- void **term_cmd_rm** (char ∗param)
- void **term_cmd_rn** (char ∗param)
- void **term_cmd_cr** (char ∗param)
- void **term_cmd_mkdir** (char ∗param)
- void **term_cmd_cp** (char ∗param)
- void **term_cmd_wt** (char ∗param)
- void **term_cmd_echo** (char ∗param)
- void **echo** (char ∗string, char Terminalbackup)
- void **term_cmd_echo_out** (char ∗param)
- void **term_cmd_temp** (char ∗param)
- void **term_cmd_setget_time** (char ∗param)
- void **term_cmd_sin2da** (char ∗param)
- void **term_cmd_esp** (char ∗param)
- void **term_cmd_null_modem** (char ∗param)
- void **term_cmd_m590** (char ∗param)
- void **term_cmd_modem** (char ∗param)
- void **term_cmd_modbus** (char ∗param)
- void **term_cmd_monitor** (char ∗param)
- void **mcu_reset** (void)

Variables

- CONST command_t **ver_cmd**
- CONST command_t **top_cmd**
- CONST command_t **rst_cmd**
- CONST command_t **cat_cmd**
- CONST command_t **ls_cmd**
- CONST command_t **cd_cmd**
- CONST command_t **mount_cmd**
- CONST command_t **sr_cmd**
- CONST command_t **rm_cmd**

- CONST command_t **rn_cmd**
- CONST command_t **cr_cmd**
- CONST command_t **mkdir_cmd**
- CONST command_t **cp_cmd**
- CONST command_t **wt_cmd**
- CONST command_t **echo_cmd**
- CONST command_t **echo_stdout_cmd**
- CONST command_t **temp_cmd**
- CONST command_t **setget_time_cmd**
- CONST command_t **sin2da_cmd**
- CONST command_t **esp_cmd**
- CONST command_t **null_modem_cmd**
- CONST command_t **m590_cmd**
- CONST command_t **modem_cmd**
- CONST command_t **modbus_cmd**
- CONST command_t **monitor_cmd**

4.6.1 Detailed Description

4.7 FatFS

Data Structures

- struct FATFS
- struct FIL
- struct DIR
- struct FILINFO

Macros

- #define **_FATFS** 80376 /∗ Revision ID ∗/
- #define **LD2PD**(vol) (BYTE)(vol) /∗ Each logical drive is bound to the same physical drive number ∗/
- #define **LD2PT**(vol) 0 /∗ Find first valid partition or in SFD ∗/
- #define **_T**(x) x
- #define **_TEXT**(x) x
- #define **f_eof**(fp) ((int)((fp)->fptr == (fp)->fsize))
- #define **f_error**(fp) ((fp)->err)
- #define **f_tell**(fp) ((fp)->fptr)
- #define **f_size**(fp) ((fp)->fsize)
- #define **EOF** (-1)
- #define **FA_READ** 0x01
- #define **FA_OPEN_EXISTING** 0x00
- #define **FA_WRITE** 0x02
- #define **FA_CREATE_NEW** 0x04
- #define **FA_CREATE_ALWAYS** 0x08
- #define **FA_OPEN_ALWAYS** 0x10
- #define **FA__WRITTEN** 0x20
- #define **FA__DIRTY** 0x40
- #define **FS_FAT12** 1
- #define **FS_FAT16** 2
- #define **FS_FAT32** 3
- #define **AM_RDO** 0x01 /∗ Read only ∗/
- #define **AM_HID** 0x02 /∗ Hidden ∗/
- #define **AM_SYS** 0x04 /∗ System ∗/
- #define **AM_VOL** 0x08 /∗ Volume label ∗/
- #define **AM_LFN** 0x0F /∗ LFN entry ∗/
- #define **AM_DIR** 0x10 /∗ Directory ∗/
- #define **AM_ARC** 0x20 /∗ Archive ∗/
- #define **AM_MASK** 0x3F /∗ Mask of defined bits ∗/
- #define **CREATE_LINKMAP** 0xFFFFFFFF
- #define **LD_WORD**(ptr) (WORD)(((WORD)∗((BYTE∗)(ptr)+1)<<8)|(WORD)∗(BYTE∗)(ptr))
- #define **LD_DWORD**(ptr) (DWORD)(((DWORD)∗((BYTE∗)(ptr)+3)<<24)|((DWORD)∗((BYTE∗)(ptr)+2)<<16)|((W↩ORD)∗((BYTE∗)(ptr)+1)<<8)|∗(BYTE∗)(ptr))
- #define **ST_WORD**(ptr, val) ∗(BYTE∗)(ptr)=(BYTE)(val); ∗((BYTE∗)(ptr)+1)=(BYTE)((WORD)(val)>>8)
- #define **ST_DWORD**(ptr, val) ∗(BYTE∗)(ptr)=(BYTE)(val); ∗((BYTE∗)(ptr)+1)=(BYTE)((WORD)(val)>>8); ∗((BYTE∗)(ptr)+2)=(BYTE)((DWORD)(val)>>16); ∗((BYTE∗)(ptr)+3)=(BYTE)((DWORD)(val)>>24)
- #define **SD_FAT_MUTEX_EN** 1
- #define **SD_BMP** 0
- #define **SD_GLCD_CALIB** 0
- #define **SD_WAVE** 0
- #define API_COMMAND_FAIL (INT8U)0x80

  SD defines.

- #define **API_COMMAND_OK** (INT8U)0x81
- #define **API_FILENAME_ERROR** (INT8U)0x82
- #define **NO_CAPS** (INT8U)0x83
- #define **CAPS_1** (INT8U)0x84
- #define **CAPS_2** (INT8U)0x85
- #define **CAPS_12** (INT8U)0x86
- #define **WRITE_BUFFER_SIZE** 512

Typedefs

- typedef char **TCHAR**

Enumerations

- enum **FRESULT** {
  FR_OK = 0, **FR_DISK_ERR**, **FR_INT_ERR**, **FR_NOT_READY**,
  FR_NO_FILE, **FR_NO_PATH**, **FR_INVALID_NAME**, **FR_DENIED**,
  FR_EXIST, **FR_INVALID_OBJECT**, **FR_WRITE_PROTECTED**, **FR_INVALID_DRIVE**,
  FR_NOT_ENABLED, **FR_NO_FILESYSTEM**, **FR_MKFS_ABORTED**, **FR_TIMEOUT**,
  FR_LOCKED, **FR_NOT_ENOUGH_CORE**, **FR_TOO_MANY_OPEN_FILES**, **FR_INVALID_PARAMETER** }
- enum **SD_STATE** {
  SD_FILE_RENAMED, **SD_FILE_DELETED**, **SD_DELETE_FILE_DENIED**, **SD_FILE_READ**,
  SD_FILE_COPIED, **SD_COPY_FILE_FAILURE**, **SD_FILE_FOUND**, **SD_CREATE_FILE_FAILURE**,
  SD_CREATE_FILE_OK, **SD_CREATE_DIR_OK**, **SD_CREATE_DIR_FAILURE**, **SD_OPEN_DIR_OK**,
  SD_OPEN_DIR_FAILURE, **SD_FILE_WRITE_FAILURE**, **SD_FILE_WRITTEN**, **SD_FILE_SUPPORTED**,
  SD_FILE_NOT_SUPPORTED, **SD_FILE_NOT_FOUND**, **SD_FAT_OK**, **SD_FAT_ERROR**,
  VERBOSE_ON, **VERBOSE_OFF** }
- enum {
  SD_CARD_STATUS = 0, **FILE_NOT_FOUND**, **FILE_INVALID**, **SD_CARD_NOT_PRESENT**,
  FILE_REMOVED, **SD_CARD_ERROR**, **SD_CARD_MOUNTED**, **SD_CARD_DETECTED**,
  SD_CARD_MOUNT_FAILURE, **SD_CARD_INIT_FAILURE** }
- enum { **NOME**, **EXTENSAO**, **FIM** }

Functions

- FRESULT **f_open** (FIL ∗fp, const TCHAR ∗path, BYTE mode)
- FRESULT **f_close** (FIL ∗fp)
- FRESULT **f_read** (FIL ∗fp, void ∗buff, UINT btr, UINT ∗br)
- FRESULT **f_write** (FIL ∗fp, const void ∗buff, UINT btw, UINT ∗bw)
- FRESULT **f_forward** (FIL ∗fp, UINT(∗func)(const BYTE ∗, UINT), UINT btf, UINT ∗bf)
- FRESULT **f_lseek** (FIL ∗fp, DWORD ofs)
- FRESULT **f_truncate** (FIL ∗fp)
- FRESULT **f_sync** (FIL ∗fp)
- FRESULT **f_opendir** (DIR ∗dp, const TCHAR ∗path)
- FRESULT **f_closedir** (DIR ∗dp)
- FRESULT **f_readdir** (DIR ∗dp, FILINFO ∗fno)
- FRESULT **f_mkdir** (const TCHAR ∗path)
- FRESULT **f_unlink** (const TCHAR ∗path)
- FRESULT **f_rename** (const TCHAR ∗path_old, const TCHAR ∗path_new)
- FRESULT **f_stat** (const TCHAR ∗path, FILINFO ∗fno)
- FRESULT **f_chmod** (const TCHAR ∗path, BYTE value, BYTE mask)
- FRESULT **f_utime** (const TCHAR ∗path, const FILINFO ∗fno)

- FRESULT **f_chdir** (const TCHAR ∗path)
- FRESULT **f_chdrive** (const TCHAR ∗path)
- FRESULT **f_getcwd** (TCHAR ∗buff, UINT len)
- FRESULT **f_getfree** (const TCHAR ∗path, DWORD ∗nclst, FATFS ∗∗fatfs)
- FRESULT **f_getlabel** (const TCHAR ∗path, TCHAR ∗label, DWORD ∗vsn)
- FRESULT **f_setlabel** (const TCHAR ∗label)
- FRESULT **f_mount** (FATFS ∗fs, const TCHAR ∗path, BYTE opt)
- FRESULT **f_mkfs** (const TCHAR ∗path, BYTE sfd, UINT au)
- FRESULT **f_fdisk** (BYTE pdrv, const DWORD szt[ ], void ∗work)
- int **f_putc** (TCHAR c, FIL ∗fp)
- int **f_puts** (const TCHAR ∗str, FIL ∗cp)
- int **f_printf** (FIL ∗fp, const TCHAR ∗str,...)
- TCHAR ∗ **f_gets** (TCHAR ∗buff, int len, FIL ∗fp)
- DWORD **get_fattime** (void)
- int **ff_cre_syncobj** (BYTE vol, _SYNC_t ∗sobj)
- int **ff_req_grant** (_SYNC_t sobj)
- void **ff_rel_grant** (_SYNC_t sobj)
- int **ff_del_syncobj** (_SYNC_t sobj)
- INT8U **SDCard_Init** (INT8U verbose)
- INT8U **SDCard_SafeRemove** (INT8U verbose)
- void **ListFiles** (CHAR8 ∗pname1)
- INT8U **ReadFile** (CHAR8 ∗FileName, INT8U verbose)
- INT8U **RenameFile** (CHAR8 ∗OldFileName, CHAR8 ∗NewFileName, INT8U verbose)
- INT8U **CreateFile** (CHAR8 ∗FileName, INT8U verbose)
- INT8U **CreateDir** (CHAR8 ∗FileName, INT8U verbose)
- INT8U **DeleteFile** (CHAR8 ∗FileName, INT8U verbose)
- INT8U **file_name_verify** (CHAR8 ∗pname1, CHAR8 ∗pname2, INT8U ∗pfile, INT8U num)
- INT8U **ChangeDir** (CHAR8 ∗FileName, INT8U verbose)
- INT8U **CopyFile** (CHAR8 ∗SrcFileName, CHAR8 ∗DstFileName, INT8U verbose)
- INT8U **WriteUptimeLog** (INT8U verbose)
- BRTOS_Mutex ∗ **SDCard_ResourceInit** (INT8U priority)
- INT8U **GetLastCreatedFileName** (char fileName[ ])
- INT8U **WriteFile** (FIL ∗fp, const char ∗filename, INT8U ∗ptr_data, INT8U length)
- FRESULT **open_append** (FIL ∗fp, const char ∗path)
- FRESULT **empty_directory** (char ∗path)
- void **CSVListFiles** (char ∗∗files)
- void **SDCard_PrintStatus** (INT8U verbose, INT8U status)

4.7.1 Detailed Description

4.8 minINI

Macros

- #define **INI_BUFFERSIZE** 256 /∗ maximum line length, maximum path length ∗/
- #define **INI_FILETYPE** FIL
- #define **ini_openread**(filename, file) (f_open((file), (filename), FA_READ+FA_OPEN_EXISTING) == FR_↩ OK)
- #define **ini_openwrite**(filename, file) (f_open((file), (filename), FA_WRITE+FA_CREATE_ALWAYS) == F↩ R_OK)
- #define **ini_close**(file) (f_close(file) == FR_OK)
- #define **ini_read**(buffer, size, file) f_gets((buffer), (size),(file))
- #define **ini_write**(buffer, file) f_puts((buffer), (file))
- #define **ini_remove**(filename) (f_unlink(filename) == FR_OK)
- #define **INI_FILEPOS** DWORD
- #define **ini_tell**(file, pos) (∗(pos) = f_tell((file)))
- #define **ini_seek**(file, pos) (f_lseek((file), ∗(pos)) == FR_OK)
- #define **NULL** (void∗)0
- #define **mTCHAR** char

Typedefs

- typedef int(∗ **INI_CALLBACK**) (const mTCHAR ∗Section, const mTCHAR ∗Key, const mTCHAR ∗Value, const void ∗UserData)

Functions

- int **ini_getbool** (const mTCHAR ∗Section, const mTCHAR ∗Key, int DefValue, const mTCHAR ∗Filename)
- long **ini_getl** (const mTCHAR ∗Section, const mTCHAR ∗Key, long DefValue, const mTCHAR ∗Filename)
- int **ini_gets** (const mTCHAR ∗Section, const mTCHAR ∗Key, const mTCHAR ∗DefValue, mTCHAR ∗Buffer, int BufferSize, const mTCHAR ∗Filename)
- int **ini_getsection** (int idx, mTCHAR ∗Buffer, int BufferSize, const mTCHAR ∗Filename)
- int **ini_getkey** (const mTCHAR ∗Section, int idx, mTCHAR ∗Buffer, int BufferSize, const mTCHAR ∗Filename)
- int **ini_putl** (const mTCHAR ∗Section, const mTCHAR ∗Key, long Value, const mTCHAR ∗Filename)
- int **ini_puts** (const mTCHAR ∗Section, const mTCHAR ∗Key, const mTCHAR ∗Value, const mTCHAR ∗Filename)
- int **ini_browse** (INI_CALLBACK Callback, const void ∗UserData, const mTCHAR ∗Filename)

4.8.1 Detailed Description

4.9 CRC16

Functions

- uint16_t **ModbusCrc16** (const uint8_t ∗const _pBuff, uint32_t _len)

4.9.1 Detailed Description

4.10 Master

Data Structures

- struct __MB_QUERY_BUILD
- struct __MB_ANSW_READY_DATA
- struct __MB_QUERY_SEND
- struct __MB_QUERY

Enumerations

- enum **__MB_PARS_ANSW** { **eMB_PARS_SLAVE_ADDR** = 0, **eMB_PARS_FUNC**, **eMB_PARS_DATA** }

Functions

- sint32_t ModbusMaster_open (const uint8_t _slave, const uint8_t _func, uint8_t ∗const _pQuery, __MB_Q←UERY ∗m_query)
- void **ModbusMaster_close** (void)
- sint32_t **Modbus_make_query** (const __MB_QUERY_BUILD ∗const _pQueryData)
- sint32_t **Modbus_prepare_receiver** (__MB_ANSW_READY_DATA ∗const m_pAnsw, uint8_t ∗const answBuff)
- sint32_t **Modbus_receive** (const uint8_t _byte)
- sint32_t **Modbus_process_answ** (uint8_t ∗ptr_data, uint16_t num_regs)
- sint32_t **Modbus_GetData** (INT8U slave, INT8U func, INT8U ∗data_ptr, INT16U start_address, INT8U num_regs)
- uint8_t **Modbus_init** (void)
- void set_bits_from_byte (uint8_t ∗dest, int address, const uint8_t value)
- void **set_bits_from_bytes** (uint8_t ∗dest, int address, int nb_bits, const uint8_t ∗tab_byte)
- uint8_t **get_byte_from_bits** (const uint8_t ∗src, int address, int nb_bits)

4.10.1 Detailed Description

4.10.2 Function Documentation

4.10.2.1 sint32_t ModbusMaster_open (const uint8_t _*slave*, const uint8_t _*func*, uint8_t ∗const _*pQuery*, __MB_QUERY ∗ *m_query*)

MODBUS uses a big-Endian Ex.: 16-bits 0x1234 the first byte sent is 0x12 then 0x34

4.10.2.2 void set_bits_from_byte (uint8_t ∗ *dest*, int *address*, const uint8_t *value*)

UTILS FUNCTIONS

Utils

4.11 Slave PM210

Data Structures

- union modbus_pm210_input_register_list1
- union modbus_pm210_input_register_list2
- union modbus_pm210_holding_register_list

Macros

- #define **PM210_REGLIST1_INPUT_START** 4000
- #define **PM210_REGLIST2_INPUT_START** 4105
- #define **PM210_REGLIST_HOLDING_START** 7000
- #define **PM210_REGLIST1_INPUT_NREGS** (36)
- #define **PM210_REGLIST2_INPUT_NREGS** 13
- #define **PM210_REGLIST_HOLDING_NREGS** 7
- #define **PM210_SLAVE_ADDRESS** (0xAA)
- #define **PM210_REG_OFFSET** (4)

4.11.1 Detailed Description

4.12 Slave NULL

Data Structures

- union modbus_null_input_register_list

Macros

- #define **NULL_REGLIST_OFFSET_NREGS** 4
- #define **NULL_REGLIST_INPUT_NREGS** 5
- #define **NULL_REGLIST_INPUT_START** 0
- #define **NULL_SLAVE_ADDRESS** (0x00)

Functions

- void **Modus_slave_null_init** (void)

4.12.1 Detailed Description

4.13 Slaves

Data Structures

- struct modbus_slave_t

Macros

- #define **MODBUS_NUM_SLAVES** (4)

Typedefs

- typedef uint8_t(∗ **_reader**) (uint8_t slave_addr, uint8_t ∗buf, uint8_t max_len)

Enumerations

- enum **slave_num_t** { **MS_NULL** = 0, **MS_PM210** = 1, **MS_TS** = 2, **MS_T500** = 3 }
- enum **eMBSlaves** {
  MODBUS_NULL = 0, **MODBUS_PM210** = 1, **MODBUS_TS** = 2, **MODBUS_T500** = 3,
  MODBUS_NONE }

Functions

- uint8_t **SetModbusHeader** (uint8_t device_id, uint8_t ∗data_ptr)
- uint8_t **SetTimeStamp** (uint8_t device_id, uint8_t ∗data_ptr, OSTime ∗timestamp)

4.13.1 Detailed Description

4.14 Slave T500

Data Structures

- union modbus_t500_input_register_list1

Macros

- #define **T500_REGLIST1_INPUT_START** 2
- #define **T500_REGLIST2_INPUT_START** 236
- #define **T500_REGLIST1_INPUT_NREGS** 11
- #define **T500_REGLIST2_INPUT_NREGS** 13
- #define **T500_SLAVE_ADDRESS** (0x01)
- #define **T500_REG_OFFSET** (4)

4.14.1 Detailed Description

4.15 Slave TS

Data Structures

- union U8
- union Estado_Reles_t
- union Opcionais_t
- union Alarmes_t
- union modbus_ts_input_register_list
- union modbus_ts_holding_register_list

Macros

- #define **TS_REG_INPUT_START** 1001
- #define **TS_REG_INPUT_NREGS** 16
- #define **TS_REG_HOLDING_START** 0000
- #define **TS_REG_HOLDING_NREGS** 48
- #define **TS_REG_OFFSET** (4)
- #define **TS_SLAVE_ADDRESS** (0x01)

4.15.1 Detailed Description

4.16 Monitor

Data Structures

- struct timestamp_t
- struct monitor_entry_t
- struct monitor_headerl1_t
- struct monitor_headerl2_t
- struct monitor_header_t
- struct timer
- struct monitor_state_t
- struct monitors_state_t
- union monitor_config_ok_t

Macros

- #define **puts**(x) printf_lib(x)
- #define **NULL** (void∗)0
- #define **FATFS_ENABLE** 1
- #define **LOG_BUFFERSIZE** 256 /∗ maximum line length, maximum path length ∗/
- #define **LOG_FILETYPE** FIL
- #define **monitor_openread**(filename, file) (f_open((file), (filename), FA_READ+FA_OPEN_EXISTING) == FR_OK)
- #define **monitor_openwrite**(filename, file) (f_open((file), (filename), FA_WRITE+FA_CREATE_ALWAYS) == FR_OK)
- #define **monitor_openappend**(filename, file) (f_open((file), (filename), FA_WRITE) == FR_OK)
- #define **monitor_close**(file) (f_close(file) == FR_OK)
- #define **monitor_read**(buffer, size, file) f_gets((buffer), (size),(file))
- #define **monitor_write**(buffer, file) (f_puts((buffer), (file)) != EOF)
- #define **monitor_remove**(filename) (f_unlink(filename) == FR_OK)
- #define **LOG_FILEPOS** DWORD
- #define **monitor_tell**(file, pos) (∗(pos) = f_tell((file)))
- #define **monitor_seek**(file, pos) (f_lseek((file), ∗(pos)) == FR_OK)
- #define **monitor_seek_end**(file) (f_lseek((file), f_size((file))) == FR_OK)
- #define **LOG_DIRTYPE** DIR
- #define **LOG_DIRINFO** FILINFO
- #define **LOG_FILEINFO** FILINFO
- #define **monitor_stat**(filename, fileinfo) (f_stat((filename), (fileinfo)) == FR_OK)
- #define **monitor_opendir**(dirname, dir) (f_opendir(&(dir),dirname) == FR_OK)
- #define **monitor_closedir**(dir) f_closedir(&(dir))
- #define **monitor_readdir**(dirinfo, dir) (f_readdir(&(dir), &(dirinfo)) == FR_OK)
- #define **monitor_chdir**(dirname) f_chdir(dirname)
- #define **monitor_mkdir**(dirname) (f_mkdir(dirname) == FR_OK)
- #define **LOG_HEADER_LEN** 50
- #define **LOG_MAX_ENTRY_SIZE** 256
- #define **FILENAME_MAX_LENGTH** 13
- #define **LOG_FILENAME_START** "99123123.txt"
- #define **LOG_METAFILE** "metafile.txt"
- #define **MAX_NUM_OF_ENTRIES** (2880)
- #define **MAX_NUM_OF_MONITORES** 4
- #define **NUM_OF_FIELDS** 5

Typedefs

- typedef struct timer **mon_timer_t**
- typedef struct pt **pt_t**
- typedef uint8_t(∗ **data_reader**) (uint8_t slave_addr, uint8_t ∗buf, uint8_t max_len)

Enumerations

- enum **monitor_used_t** { **UNUSED** = 0, **IN_USE** = 1 }

Functions

- void **test_logger** (void)
- uint8_t **monitor_init** (uint8_t monitor_num)
- void **monitor_sync** (uint8_t monitor_num, const char ∗)
- void **monitor_makeheader** (char monitor_header[ ], monitor_header_t ∗h)
- uint8_t **monitor_setheader** (const char ∗filename, monitor_header_t ∗h)
- uint8_t **monitor_getheader** (const char ∗filename, monitor_header_t ∗h)
- uint8_t **monitor_newheader** (const char ∗filename, uint8_t monitor_id, uint16_t interval, uint16_t entry_size)
- uint8_t **monitor_validateheader** (const char ∗filename, uint8_t monitor_id, uint16_t interval, uint16_t entry↩ _size)
- void **monitor_createentry** (char ∗string, uint16_t ∗dados, uint8_t len)
- uint16_t **monitor_writeentry** (const char ∗filename, char ∗entry, uint8_t monitor_num)
- uint32_t **monitor_readentry** (uint8_t monitor_num, const char ∗filename, monitor_entry_t ∗entry, uint8_↩ t enable_send, uint8_t send_ok)
- uint32_t **monitor_confirm_entry_sent** (uint8_t monitor_num, const char ∗filename)
- uint8_t **monitor_gettimestamp** (struct tm ∗ts, uint32_t time_elapsed_s)
- void **monitor_settimestamp** (uint8_t monitor_num, const char ∗filename)
- char ∗ **monitor_getfilename_to_write** (uint8_t monitor_num)
- char ∗ **monitor_getfilename_to_read** (uint8_t monitor_num)
- void **main_monitor** (void)
- uint16_t **monitor_reader** (uint8_t monitor_num)
- void **monitor_writer** (uint8_t monitor_num)
- uint16_t **monitor_reader_multiple** (uint8_t monitor_num)
- clock_t **clock_time** (void)

Variables

- union {
  char **int8_t_incorrect** [sizeof(int8_t)==1]
  char **uint8_t_incorrect** [sizeof(uint8_t)==1]
  char **int16_t_incorrect** [sizeof(int16_t)==2]
  char **uint16_t_incorrect** [sizeof(uint16_t)==2]
  char **int32_t_incorrect** [sizeof(int32_t)==4]
  char **uint32_t_incorrect** [sizeof(uint32_t)==4]
  } **u**

4.16.1 Detailed Description

4.17 Comandos AT

Files

- file at_commands.h

  Implementação de comandos AT para modems.

Macros

- #define **MODEM_APN** "tim.br"
- #define **MODEM_PWD** "tim"
- #define **AT_def** "AT\r\n"
- #define **CREG_def** "AT+CREG?\r\n"
- #define **XISP_def** "AT+XISP=0\r\n"
- #define **GPRS0_def** "AT#GPRS=0\r\n"
- #define **GPRS1_def** "AT#GPRS=1\r\n"
- #define **GPRS_def** "AT#GPRS?\r\n"
- #define **XIIC1_def** "AT+XIIC=1\r\n"
- #define **XIIC_def** "AT+XIIC?\r\n"
- #define **IPSTAT_def** "AT+IPSTATUS=0\r"
- #define **CLK_def** "AT+CCLK?\r\n"
- #define **CLOSE0_def** "AT+TCPCLOSE=0\r\n"
- #define **CLOSE1_def** "AT+TCPCLOSE=1\r\n"
- #define **CGDCONT_def** ("AT+CGDCONT=1,\"IP\",\"" MODEM_APN "\"\r\n")
- #define **XGAUTH_def** ("AT+XGAUTH=1,1,\"" MODEM_PWD "\",\"" MODEM_PWD "\"\r\n")
- #define **SKTRST_def** "AT#SKTRST\r\n"
- #define **ATZ_def** "ATZ\r\n"

Enumerations

- enum **at_enum_cmd** {
  AT = 0, **CREG**, **XISP**, **GPRS0**,
  GPRS1, **GPRS**, **XIIC1**, **XIIC**,
  IPSTAT, **CLK**, **CLOSE0**, **CLOSE1**,
  CGDCONT, **XGAUTH**, **SKTRST**, **ATZ** }

Variables

- const char ∗const **modem_init_cmd** [ ]

4.17.1 Detailed Description

4.18 Memória EEPROM

Files

- file eeprom.h

  Interface para ler/escrever na memoria eeprom.

Macros

- #define WRITE_CYCLE_TIME 10

  EEPROM requer 10ms para ser escrita.
- #define EEPROMAddress 0xA0

  Endereco do dispositivo EEPROM (escravo)

Functions

- void **EEPROM_ByteWrite** (u16 endr, u08 dado)
- u08 **EEPROM_RandomRead** (u16 endr)
- u08 EEPROM_CurrentAddressRead (void)

  Leitura na EEPROM.
- void **EEPROM_AckPolling** (void)
- void EEPROM_Init (void)

  Configura os pinos para o protocolo IIC.

4.18.1 Detailed Description

4.18.2 Function Documentation

4.18.2.1 u08 EEPROM_CurrentAddressRead (void)

Leitura na EEPROM.

Returns

Valor lido no ultimo endereco de memoria acessado

4.19 Modem ESP8266

Files

- file esp8266_at.h

  Interface para ler/escrever dados no modem ESP8266.

Macros

- #define **ESP_ENABLE** 0
- #define **ESP_BAUD** 9600
- #define **ESP_UART** 2
- #define **ESP_TCP_PORT** 80
- #define **ESP_TCP_LOCAL_PORT** 10201
- #define **ESP_TCP_CTX_NUM** 0
- #define **ESP_TCP_CTX_SIZE** 2048
- #define **ESP_AP** "GISELE_e_CARLOS"
- #define **ESP_PWD** "01122007"
- #define **ESP_UART_BUFSIZE** 64
- #define **ESP_UART_TIMEOUT** 2000

Typedefs

- typedef state_t **esp_state_t**

Enumerations

- enum **esp_ret_t** { **ESP_OK**, **ESP_STATE_ERR**, **ESP_APCONN_ERR**, **ESP_TCPCONN_ERR** }

Functions

- esp_ret_t **at_esp_init** (void)
- esp_ret_t **at_esp_open** (void)
- esp_ret_t **at_esp_send** (INT8U ∗dados)
- esp_ret_t **at_esp_receive** (CHAR8 ∗buff, INT8U ∗len)
- esp_ret_t **at_esp_close** (void)
- CHAR8 **at_esp_getchar** (void)
- INT8U **esp_set_hostname** (CHAR8 ∗host)
- INT8U **esp_get_ip** (void)
- INT8U **esp_set_ip** (CHAR8 ∗_ip)

4.19.1 Detailed Description

4.20 Modem GC864

Files

- file gc864_modem.h

  Interface para ler/escrever dados no modem GC864.

Macros

- #define **MODEM_UART_BUFSIZE** 32
- #define **MODEM_UART_TIMEOUT** 10
- #define **MODEM_BAUD** 19200
- #define **USE_UART_MODEM** USE_UART1
- #define **modem_printP**(x) printSer(USE_UART_MODEM,(char∗)x);
- #define **modem_printR**(x) printSer(USE_UART_MODEM,(char∗)x);
- #define **modem_putchar**(x) putcharSer(USE_UART_MODEM,x)
- #define **modem_acquire**() uart1_acquire()
- #define **modem_release**() uart1_release()

Functions

- modem_ret_t **at_modem_init** (void)
- modem_ret_t **at_modem_open** (INT8U host_or_ip, char ∗dados)
- modem_ret_t **at_modem_send** (char ∗dados)
- modem_ret_t **at_modem_receive** (char ∗buff, uint16_t len)
- modem_ret_t **at_modem_close** (void)
- modem_ret_t **at_modem_server** (void)
- modem_ret_t **at_modem_dns** (char ∗param)
- modem_ret_t **at_modem_time** (void)
- CHAR8 **gc864_modem_getchar** (void)
- uint8_t **gc864_modem_init** (void)
- uint8_t **gc864_modem_open** (void)
- uint8_t **gc864_modem_close** (void)
- uint8_t **gc864_modem_get_time** (void)
- uint8_t **gc864_modem_receive** (char ∗buff, uint16_t ∗len)
- uint8_t **gc864_modem_send** (char ∗dados, uint16_t tam)
- uint8_t **gc864_modem_set_ip** (char ∗_ip)
- char ∗ **gc864_modem_get_ip** (void)
- uint8_t **gc864_modem_set_hostname** (char ∗host)
- char ∗ **gc864_modem_get_hostname** (void)
- uint8_t **gc864_modem_resolve_ip** (char ∗host, char ∗_ip)
- uint8_t **gc864_modem_check_connection** (void)

4.20.1 Detailed Description

4.21 LCD

Files

- file lcd.h

  Alphanumeric LCD function prototypes.

Macros

- #define **LCD_DATA_BUS** 4
- #define **LCD_USE_BRTOS** 1
- #define **LCD_CPU_CLOCK** 24000000
- #define **LCD_FOR_NUMBER_OF_CYCLES** 19
- #define **RS** PTDD_PTDD2
- #define **RS_DIR** PTDDD_PTDDD2
- #define **E** PTDD_PTDD3
- #define **E_DIR** PTDDD_PTDDD3
- #define **DATA** PTDD
- #define **DATA_DIR** PTDDD
- #define **BUSY_FLAG** PTDD_PTDD7
- #define **DATA_SHIFT** 0
- #define **BACKLIGHT_DIR** PTCDD_PTCDD7
- #define **BACKLIGHT** PTCD_PTCD7
- #define **delay_450ns**()
- #define **delay_600ns**()

Functions

- void **printf_lcd** (char ∗string)
- void **instr_lcd** (char comando)
- void **putchar_lcd** (char dado)
- void **write_number_lcd** (unsigned char numero)
- void **init_lcd** (void)
- void **init_resource_lcd** (unsigned char priority)
- void **acquire_lcd** (void)
- void **release_lcd** (void)
- void **xy_position_lcd** (unsigned char linha, unsigned char coluna)
- void **clear_lcd** (void)
- void **Delay_ms** (unsigned int DelayTime)

4.21.1 Detailed Description

4.22 Modem M590

Files

- file m590_at.h

  Interface para ler/escrever dados no modem M590.

Macros

- #define **M590_ENABLE** 0
- #define **M590_BAUD** 9600
- #define **M590_UART** MODEM_UART
- #define **M590_TCP_SERVER_NAME** "emon-gpsnetcms.rhcloud.com"
- #define **M590_TCP_SERVER_IP** "54.160.189.224"
- #define **M590_TCP_PORT** 80
- #define **M590_TCP_LOCAL_PORT** 10201
- #define **M590_TCP_CTX_NUM** 0
- #define **M590_TCP_CTX_SIZE** 2048
- #define **M590_APN** "tim.br"
- #define **M590_PWD** "tim"
- #define **M590_UART_BUFSIZE** 64
- #define **M590_UART_TIMEOUT** 10

Enumerations

- enum **m590_state_t** { **M590_SETUP**, **M590_INIT**, **M590_OPEN**, **M590_CLOSE** }
- enum **m590_ret_t** {
  M590_OK, **M590_ERR**, **M590_STATE_ERR**, **M590_APCONN_ERR**,
  M590_TCPCONN_ERR }

Functions

- m590_ret_t **at_m590_init** (void)
- m590_ret_t **at_m590_open** (void)
- m590_ret_t **at_m590_send** (char ∗dados)
- m590_ret_t **at_m590_receive** (char ∗buff, uint16_t len)
- m590_ret_t **at_m590_close** (void)
- m590_ret_t **at_m590_server** (void)
- m590_ret_t **at_m590_dns** (char ∗param)
- m590_ret_t **at_m590_time** (void)
- CHAR8 **m590_getchar** (void)
- uint8_t **m590_init** (void)
- uint8_t **m590_open** (void)
- uint8_t **m590_close** (void)
- uint8_t **m590_get_time** (void)
- uint8_t **m590_receive** (char ∗buff, uint16_t ∗len)
- uint8_t m590_send (char ∗dados, uint16_t tam)
- uint8_t **m590_set_ip** (char ∗_ip)
- char ∗ **m590_get_ip** (void)
- uint8_t **m590_set_hostname** (char ∗host)
- char ∗ **m590_get_hostname** (void)
- uint8_t **m590_host_ip** (void)
- uint8_t **m590_check_connection** (void)

4.22.1 Detailed Description

4.22.2 Function Documentation

4.22.2.1 uint8_t m590_send (char ∗ *dados,* uint16_t *tam*)

testar isso

4.23 Modem

Files

- file modem.h

  Definicoes de interface para modems.

Enumerations

- enum **state_t** { **SETUP**, **INIT**, **OPEN**, **CLOSE** }
- enum **modem_ret_t** { **MODEM_OK**, **MODEM_ERR**, **MODEM_STATE_ERR** }

4.23.1 Detailed Description

4.24 RS485

Files

- file rs485.h

  Rotinas para transferir e receber dados via RS485/UART.

Functions

- void rs485_init (void)

  Inicializa RS485/UART.
- void **rs485_acquire** (void)
- void **rs485_release** (void)
- void **rs485_putchar** (INT8U caracter)
- void **rs485_print** (CHAR8 *string)
- INT8U **rs485_rx** (CHAR8 *caracter, INT16U timeout)
- void **rs485_tx** (const INT8U *data, const INT16U len)
- void **rs485_rx_flush** (void)
- void **rs485_enable_rx** (void)
- void **rs485_enable_tx** (void)

4.24.1 Detailed Description

4.25 RTC DS1307

Data Structures

- struct RTC_DS1307

  Estrutura para manter informacoes do calendario.

Macros

- #define DS1307Address 0xD0

  Rotinas para ler e escrever dados no DS1307. Funcoes de leitura/escrita sao feitas via IIC.
- #define SEC_ADDRESS 0x00

  Endereco de memoria dos segundos no DS1307.
- #define MIN_ADDRESS 0x01

  Endereco de memoria dos minutos no DS1307.
- #define HOUR_ADDRESS 0x02

  Endereco de memoria das horas no DS1307.
- #define WEEK_DAY_ADDRESS 0x03

  Endereco de memoria do dia da semana no DS1307.
- #define DAY_ADDRESS 0x04

  Endereco de memoria do dia do mes no DS1307.
- #define MONTH_ADDRESS 0x05

  Endereco de memoria dos meses no DS1307.
- #define YEAR_ADDRESS 0x06

  Endereco de memoria dos anos no DS1307.
- #define **RTC_YEAR_INIT** (2000)

Functions

- void RTC_ByteWrite (INT8U Address, INT8U Data)

  Escreve Data em Address.
- INT8U RTC_CurrentAddressRead (void)

  Leitura no DS1307.
- INT8U RTC_RandomRead (INT8U Address)

  Leitura aleatoria no DS1307.
- void **RTC_AckPolling** (void)
- INT8U RTC_DS1307_Init (void)

  Configura os pinos para o protocolo IIC.
- INT8U RTC_DS1307_GetSeconds (void)
- INT8U RTC_DS1307_GetMinutes (void)
- INT8U RTC_DS1307_GetHours (void)
- INT8U RTC_DS1307_GetDayOfMonth (void)
- INT8U RTC_DS1307_GetMonth (void)
- INT8U RTC_DS1307_GetYear (void)
- INT8U **RTC_DS1307_GetStatus** (void)
- void **RTC_DS1307_SetStatus** (INT8U st)
- void RTC_DS1307_Set_Time (INT8U hour, INT8U min, INT8U sec)

  Grava nova hora no DS1307.
- void RTC_DS1307_Set_Date (INT8U year, INT8U month, INT8U day)

  Grava nova data no DS1307.

- void RTC_DS1307_Update (RTC_DS1307 ∗rtc_timer)

  Atualiza o calendario da estrutura rtc_timer lendo os dados no DS1307.
- void **RTC_DS1307_Config** (void)
- void **RTC_DS1307_Start_OSC** (void)
- INT8U **Get_Hour_Format** (void)
- void Set_24h_Format (void)

  Habilita formato 24h no DS1307.
- void Set_AM_PM_Mode (void)

  Habilita modo AM/PM no DS1307.

4.25.1 Detailed Description

4.25.2 Macro Definition Documentation

4.25.2.1 #define DS1307Address 0xD0

Rotinas para ler e escrever dados no DS1307. Funcoes de leitura/escrita sao feitas via IIC.

Endereco do DS1307 (escravo).

4.25.3 Function Documentation

4.25.3.1 void RTC_ByteWrite (INT8U *Address*, INT8U *Data*)

Escreve Data em Address.

Parameters

Data	Valor a ser gravado
Address	Endereco a ser gravado

4.25.3.2 INT8U RTC_CurrentAddressRead (void)

Leitura no DS1307.

Returns

Valor lido no ultimo endereco de memoria acessado

4.25.3.3 INT8U RTC_DS1307_GetDayOfMonth (void)

Returns

Dia do mes do DS1307.

4.25.3.4 INT8U RTC_DS1307_GetHours (void)

Returns

Hras do DS1307.

4.25.3.5 INT8U RTC_DS1307_GetMinutes (void)

Returns

Minutos do DS1307.

4.25.3.6 INT8U RTC_DS1307_GetMonth (void)

Returns

Mes do DS1307.

4.25.3.7 INT8U RTC_DS1307_GetSeconds (void)

Returns

Segundos do DS1307.

4.25.3.8 INT8U RTC_DS1307_GetYear (void)

Returns

Ano do DS1307.

4.25.3.9 void RTC_DS1307_Set_Date (INT8U *year,* INT8U *month,* INT8U *day*)

Grava nova data no DS1307.

Parameters

year	Novo ano a ser gravado.
month	Novo mes a ser gravado.
day	Novo dia a ser gravado.

4.25.3.10 void RTC_DS1307_Set_Time (INT8U *hour,* INT8U *min,* INT8U *sec*)

Grava nova hora no DS1307.

Parameters

hour	Nova hora a ser gravada.
min	Novo minuto a ser gravado.
sec	Novo segundo a ser gravado.

4.25.3.11 void RTC_DS1307_Update (RTC_DS1307 * *rtc_timer*)

Atualiza o calendario da estrutura rtc_timer lendo os dados no DS1307.

Parameters

rtc_timer	Estrutura a ser atualizada.

4.25.3.12 INT8U RTC_RandomRead (INT8U *Address*)

Leitura aleatoria no DS1307.

Parameters

Address	Endereco a ser lido

Returns

Valor lido no endereco Address

4.26 Cartão SD

Files

- file SD.h

  Interface para ler/escrever dados no SD.

Data Structures

- union T32_8
- union T16_8

Macros

- #define **USE_OS** 1
- #define **SD_BLOCK_512**
- #define **SD_WAIT_CYCLES** 30
- #define **_OUT** 1
- #define **_IN** 0
- #define **SD_BLOCK_SIZE** (0x00000200)
- #define **SD_BLOCK_SHIFT** (9)
- #define **BLOCK_SIZE** 512
- #define **SD_CS** dummy /∗ Slave Select 1 ∗/
- #define **_SD_CS**
- #define **SD_AUSENT** 1
- #define **_SD_AUSENT**
- #define **SD_AUSENT_PULLUP**
- #define **SD_WP**
- #define **_SD_WP**
- #define **SD_WP_PULLUP**
- #define **FCLK_SLOW**()
- #define **FCLK_FAST**()
- #define **SD_PRESENT** (!SD_AUSENT)
- #define **ENABLE** 0
- #define **DISABLE** 1
- #define **CS_LOW**() /∗ MMC CS = L ∗/
- #define **CS_HIGH**() /∗ MMC CS = H ∗/
- #define **SOCKINS** 0 /∗ Card detected. yes:true, no:false, default:true ∗/
- #define **CMD0** (0) /∗ GO_IDLE_STATE ∗/
- #define **CMD1** (1) /∗ SEND_OP_COND (MMC) ∗/
- #define **ACMD41** (0x80+41) /∗ SEND_OP_COND (SDC) ∗/
- #define **CMD8** (8) /∗ SEND_IF_COND ∗/
- #define **CMD9** (9) /∗ SEND_CSD ∗/
- #define **CMD10** (10) /∗ SEND_CID ∗/
- #define **CMD12** (12) /∗ STOP_TRANSMISSION ∗/
- #define **ACMD13** (0x80+13) /∗ SD_STATUS (SDC) ∗/
- #define **CMD16** (16) /∗ SET_BLOCKLEN ∗/
- #define **CMD17** (17) /∗ READ_SINGLE_BLOCK ∗/
- #define **CMD18** (18) /∗ READ_MULTIPLE_BLOCK ∗/
- #define **CMD23** (23) /∗ SET_BLOCK_COUNT (MMC) ∗/
- #define **ACMD23** (0x80+23) /∗ SET_WR_BLK_ERASE_COUNT (SDC) ∗/
- #define **CMD24** (24) /∗ WRITE_BLOCK ∗/

- #define **CMD25** (25) /∗ WRITE_MULTIPLE_BLOCK ∗/
- #define **CMD32** (32) /∗ ERASE_ER_BLK_START ∗/
- #define **CMD33** (33) /∗ ERASE_ER_BLK_END ∗/
- #define **CMD38** (38) /∗ ERASE ∗/
- #define **CMD55** (55) /∗ APP_CMD ∗/
- #define **CMD58** (58) /∗ READ_OCR ∗/
- #define **CT_MMC** 0x01 /∗ MMC ver 3 ∗/
- #define **CT_SD1** 0x02 /∗ SD ver 1 ∗/
- #define **CT_SD2** 0x04 /∗ SD ver 2 ∗/
- #define **CT_SDC** (CT_SD1|CT_SD2) /∗ SD ∗/
- #define **CT_BLOCK** 0x08 /∗ Block addressing ∗/

Enumerations

- enum {
  SD_OK, **COMMAND_FAILS**, **INIT_FAILS**, **WRITE_COMMAND_FAILS**,
  WRITE_DATA_FAILS, **READ_COMMAND_FAILS**, **READ_DATA_FAILS**, **NO_SD_CARD**,
  INIT_SD_FAILS, **MOUNT_SD_FAILS** }

Functions

- void **disk_timerproc** (void)
- void **SD_CLKDelay** (INT8U)
- INT8U **GetCardType** (void)
- INT8U **GetCardStat** (void)
- INT8U **GetCardInit** (void)
- void **SetCardStat** (INT8U state)
- void **GetFatTimer** (INT32U ∗time)
- void **SetFatTimer** (INT32U time)
- DWORD **get_fattime** (void)
- void **xmit_spi** (INT8U dat)

4.26.1 Detailed Description

4.27 Sensors

Files

- file sensors.h

    Interface para ler dados de sensores.

Macros

- #define **LEVEL_MIN** (0)
- #define **LEVEL_MED** (1)
- #define **LEVEL_MAX** (2)

Enumerations

- enum **sensor_id_t** { **PRESSURE_VALVE** = 0, **SENSOR_LEVEL** = 1 }

Functions

- void **sensors_init** (void)
- uint8_t **sensors_status** (void)
- uint8_t **sensors_read** (sensor_id_t)
- uint8_t **sensors_read_all** (void)

4.27.1 Detailed Description

4.28 Terminal I/O

Files

- file terminal_io.h

  Interface para ler/escrever dados no terminal.

Data Structures

- struct command_t

Macros

- #define **MAX_CMDS** 20
- #define **MAX_CMD_SIZE** 8
- #define **CONSOLE_BUFFER_SIZE** (64)
- #define **CONST** const
- #define **INROM** 1
- #define **DEL** 0x7F
- #define **USE_USB** 0
- #define **USE_UART0** 0
- #define **USE_UART1** 1
- #define **USE_UART2** 2

Typedefs

- typedef void(**cmd_func**) (char ∗params)
- typedef CHAR8(∗ **term_input**) (CHAR8 ∗)
- typedef CHAR8(∗ **term_output**) (CHAR8)

Functions

- int **terminal_add_cmd** (command_t ∗cmd)
- int **terminal_delete_cmd** (command_t ∗cmd)
- void **terminal_init** (void(∗putch_)(char))
- void **terminal_process** (void)
- int **term_skipp_space** (char ∗cmd_line, int start)
- int **term_find_word** (char ∗cmd_line, int start)
- int **term_cmp_str** (char ∗a, char ∗b)
- void **SetSilentMode** (char mode)
- unsigned char **TerminalBackup** (char ∗backup)
- void **printf_terminal** (const char ∗s)
- void **putchar_terminal** (char c)
- int **getchar_terminal** (char ∗c, int timeout)
- void **terminal_acquire** (void)
- void **terminal_release** (void)
- void **terminal_newline** (void)
- int **is_terminal_idle** (void)
- void **terminal_set_idle** (char state)
- void **printf_terminal_P** (const char ∗s)
- void **terminal_set_input** (term_input _input)
- void **terminal_set_output** (term_output _output)
- void **terminal_input** (CHAR8 ∗c)
- void **terminal_output** (CHAR8 c)
- void **printSer** (INT8U SerialPort, const CHAR8 ∗string)
- void **putcharSer** (INT8U SerialPort, CHAR8 caracter)

4.28.1 Detailed Description

4.29 Drivers

Modules

- LEDs
- MCU
- SPI
- UART

4.29.1 Detailed Description

4.30 Modbus

Modules

- CRC16
- Master
- Slave PM210
- Slave NULL
- Slaves
- Slave T500
- Slave TS

Data Structures

- union __UNION_DWORD

Macros

- #define **MB_RS485** 1
- #define **MODBUSMASTER_LOCK**() rs485_acquire();
- #define **MODBUSMASTER_UNLOCK**() rs485_release();
- #define **MODBUSMASTER_PUTCHAR**(x) rs485_putchar(x)
- #define **RS485_TIMEOUT_RX** 10
- #define **QUERY_BUFSIZE** (8)
- #define **ANSWER_BUFSIZE** (36∗2 + 8)
- #define **_STDINT_H** 1
- #define **TRUE_T** (1)
- #define **FALSE_T** (0)
- #define **NULL** ((void∗)0L)
- #define **MODBUS_OPEN** (1)
- #define **MODBUS_CLOSE** (0)
- #define **MODBUS_OK** (0)
- #define **MODBUS_ERROR** (-1)
- #define **HEADER_LENGTH_RTU** (0)
- #define **PRESET_QUERY_LENGTH_RTU** (6)
- #define **PRESET_RESPONSE_LENGTH_RTU** (2)
- #define **CHECKSUM_LENGTH_RTU** (2)
- #define **MIN_QUERY_LENGTH** (8)
- #define **MIN_ANSWER_LENGTH** (5)
- #define **MAX_MESSAGE_LENGTH** (256)
- #define **MASTER_BUFSIZE** (80)
- #define **MAX_STATUS** (512)
- #define **MAX_REGISTERS** (36)
- #define **REPORT_SLAVE_ID_LENGTH** (75)
- #define **MB_OFF** (0)
- #define **MB_ON** (1)
- #define **FC_READ_COIL_STATUS** (0x01)
- #define **FC_READ_INPUT_STATUS** (0x02)
- #define **FC_READ_HOLDING_REGISTERS** (0x03)
- #define **FC_READ_INPUT_REGISTERS** (0x04)
- #define **FC_FORCE_SINGLE_COIL** (0x05)
- #define **FC_PRESET_SINGLE_REGISTER** (0x06)

- #define **FC_READ_EXCEPTION_STATUS** (0x07)
- #define **FC_FORCE_MULTIPLE_COILS** (0x0F)
- #define **FC_PRESET_MULTIPLE_REGISTERS** (0x10)
- #define **FC_REPORT_SLAVE_ID** (0x11)
- #define **ILLEGAL_FUNCTION** (0x01)
- #define **ILLEGAL_DATA_ADDRESS** (0x02)
- #define **ILLEGAL_DATA_VALUE** (0x03)
- #define **SLAVE_DEVICE_FAILURE** (0x04)
- #define **SERVER_FAILURE** (0x04)
- #define **ACKNOWLEDGE** (0x05)
- #define **SLAVE_DEVICE_BUSY** (0x06)
- #define **NEGATIVE_ACKNOWLEDGE** (0x07)
- #define **MEMORY_PARITY_ERROR** (0x08)
- #define **MSG_LENGTH_UNDEFINED** (0)
- #define **MB_MASTER_ERR_OK** (0)
- #define **MB_MASTER_ERR_LEN** (-1)
- #define **MB_MASTER_ERR_SLAVE** (-2)
- #define **MB_MASTER_ERR_FUNC** (-3)
- #define **MB_MASTER_ERR_CRC** (-4)
- #define **MB_MASTER_ERR_UNDEF** (-5)
- #define **MB_MASTER_ERR_TIMEOUT** (-6)
- #define **MB_MASTER_ERR_DATA** (-7)

Typedefs

- typedef signed char **sint8_t**
- typedef signed short **sint16_t**
- typedef unsigned int **bool_t**
- typedef signed int **sint32_t**
- typedef long **long32_t**
- typedef unsigned long **ulong32_t**
- typedef signed long **sling32_t**
- typedef float **float32_t**
- typedef double **float64_t**

Functions

- void **Task_modbus_master_test** (void)
- uint8_t **Modbus_init** (void)
- sint32_t **Modbus_GetData** (INT8U slave, INT8U func, INT8U ∗data_ptr, INT16U start_address, INT8U num_regs)

Variables

- uint8_t **queryBuffer** [QUERY_BUFSIZE]
- uint8_t **answerBuffer** [ANSWER_BUFSIZE]

4.30.1 Detailed Description

4.31 App

Modules

- Relógio do sistema
- Terminal de Comandos
- Monitor

4.31.1 Detailed Description

4.32 Sistema

Modules

- FatFS
- minINI
- Brtos
- Modbus
- Pt
- Simon API

4.32.1 Detailed Description

4.33 Brtos

Files

- file BRTOS.h

    BRTOS kernel main defines, functions prototypes and structs declaration.
- file OS_RTC.h

    System Time managment struct declarations and functions prototypes.
- file timers.h

    OS Soft Timers service functions.

Data Structures

- struct Context
- struct BRTOS_Sem
- struct BRTOS_Mutex
- struct BRTOS_Mbox
- struct BRTOS_Queue
- struct OS_QUEUE
- struct OS_DQUEUE
- struct OS_QUEUE_16
- struct OS_QUEUE_32
- struct OSTime
- struct OSDate
- struct OSTimeDate
- struct OSDateTime
- struct _OSRTC
- struct ContextType
- struct OSTime_Date

Macros

- #define **BRTOS_VERSION** "BRTOS Ver. 1.79"
- #define FALSE 0

    False and True defines.
- #define **TRUE** 1
- #define **NULL** (void∗)0
- #define **READY_LIST_VAR**
- #define **BRTOS_BIG_ENDIAN** (0)
- #define **BRTOS_LITTLE_ENDIAN** (1)
- #define **BRTOS_TH** OS_CPU_TYPE
- #define READY (INT8U)0

    Task States.
- #define SUSPENDED (INT8U)1

    Task is suspended.
- #define BLOCKED (INT8U)2

    Task is blocked - Will not run until be released.
- #define **MUTEX_PRIO** (INT8U)0xFE
- #define **EMPTY_PRIO** (INT8U)0xFF
- #define NO_TIMEOUT (INT16U)65000

    Timer defines.

- #define **EXIT_BY_TIMEOUT** (INT16U)65001
- #define TICK_COUNT_OVERFLOW (INT16U)64000

  Determines the tick timer overflow.
- #define TickCountOverFlow (INT16U)64000

  Compatibility with BRTOS less than or equal to 1.7.
- #define OK (INT8U)0

  Error codes.
- #define NO_MEMORY (INT8U)1

  Error - Lack of memory to allocate a task.
- #define STACK_SIZE_TOO_SMALL (INT8U)2

  Error - Stack size too small to allocate a task.
- #define END_OF_AVAILABLE_PRIORITIES (INT8U)3

  Error - There are no more priorities available.
- #define BUSY_PRIORITY (INT8U)4

  Error - Priority is being used by another task.
- #define INVALID_TIME (INT8U)5

  Error - Informed time is out of the limits.
- #define TIMEOUT (INT8U)6

  Error - Timeout.
- #define CANNOT_ASSIGN_IDLE_TASK_PRIO (INT8U)7

  Error - A task can not be assigned into the idle task slot.
- #define NOT_VALID_TASK (INT8U)8

  There current task number is not valid for this function.
- #define NO_TASK_DELAY (INT8U)9

  Error - No valid time to wait.
- #define END_OF_AVAILABLE_TCB (INT8U)10

  Error - There are no more task control blocks (Context task)
- #define ALLOC_EVENT_OK (INT8U)0

  Event allocated with success.
- #define NO_AVAILABLE_EVENT (INT8U)1

  No event control blocks available.
- #define NO_AVAILABLE_MEMORY (INT8U)2

  Error - Lack of memory to allocate an event.
- #define INVALID_PARAMETERS (INT8U)3

  There is at least one invalid parameter.
- #define IRQ_PEND_ERR (INT8U)4

  Function can not be called inside an interrupt.
- #define ERR_SEM_OVF (INT8U)5

  Semaphore counter overflow.
- #define ERR_MUTEX_OVF (INT8U)6

  Mutex counter overflow.
- #define ERR_EVENT_NO_CREATED (INT8U)7

  There are no task waiting for the event.
- #define NULL_EVENT_POINTER (INT8U)8

  The passed event pointer is NULL.
- #define ERR_EVENT_OWNER (INT8U)9

  Function caller is not the owner of the event control block. Used to mutex implementation.
- #define DELETE_EVENT_OK (INT8U)10

  Event deleted with success.
- #define AVAILABLE_RESOURCE (INT8U)11

  The resource is available.

- #define BUSY_RESOURCE (INT8U)12

  The resource is busy.
- #define AVAILABLE_MESSAGE (INT8U)13

  There is a message.
- #define NO_MESSAGE (INT8U)14

  There is no message.
- #define READ_BUFFER_OK 0

  New data successfully read.
- #define WRITE_BUFFER_OK 0

  New data successfully written.
- #define BUFFER_UNDERRUN 1

  Queue overflow.
- #define CLEAN_BUFFER_OK 2

  Queue successfully cleaned.
- #define NO_ENTRY_AVAILABLE 3

  Queue is empty.
- #define DELAY 0

  Suspended Types.
- #define SEMAPHORE 1

  Task suspended by semaphore.
- #define MAILBOX 2

  Task suspended by mailbox.
- #define QUEUE 3

  Task suspended by queue.
- #define MUTEX 4

  Task suspended by mutex.
- #define configMAX_TASK_INSTALL 8

  Task Defines.
- #define **configMAX_TASK_PRIORITY** 7
- #define **CONST**
- #define **OS_INT_ENTER**() iNesting++;
- #define **OS_INT_EXIT**()
- #define **RemoveFromDelayList**()
- #define **IncludeTaskIntoDelayList**()

Typedefs

- typedef INT8U **PriorityType**
- typedef struct Context **ContextType**
- typedef struct _OSRTC **OS_RTC**

Functions

- INT8U **InstallTask** (void(∗FctPtr)(void), const CHAR8 ∗TaskName, INT16U USER_STACKED_BYTES, IN↩ T8U iPriority, OS_CPU_TYPE ∗TaskHandle)
- INT8U InstallIdle (void(∗FctPtr)(void), INT16U USER_STACKED_BYTES)

  Install the idle task. Initial state = running.
- void Idle (void)

  Idle Task. May be used to implement low power commands.
- void **OS_TICK_HANDLER** (void)

- INT8U BRTOSStart (void)

    Start the Operating System Scheduler The user must call this function to start the tasks execution.
- INT8U DelayTask (INT16U time)

    Wait for a specified period. A task that calling this function will be suspended for a certain time. When this time is reached the task back to ready state.
- INT8U DelayTaskHMSM (INT8U hours, INT8U minutes, INT8U seconds, INT16U miliseconds)

    Wait for a specified period (in hours, minutes, seconds and miliseconds). A task that calling this function will be suspended for a certain time. When this time is reached the task back to ready state.
- INT16U **OSGetTickCount** (void)
- INT16U **OSGetCount** (void)
- void OSIncCounter (void)

    Update the tick counter.
- void PreInstallTasks (void)

    Function that initialize the kernel main variables. This function resets the kernel main variables, preparing the system to be started.
- INT8U BlockPriority (INT8U iPriority)

    Blocks a specific priority Blocks the task that is associated with the specified priority. The user must be careful when using this function in together with mutexes. This can lead to undesired results due the "cealing priority" property used in the mutex.
- INT8U UnBlockPriority (INT8U iPriority)

    UnBlock a specific priority UnBlocks the task that is associated with the specified priority. The user must be careful when using this function in together with mutexes. This can lead to undesired results due the "cealing priority" property used in the mutex.
- INT8U **BlockTask** (BRTOS_TH iTaskNumber)
- INT8U **UnBlockTask** (BRTOS_TH iTaskNumber)
- INT8U BlockMultipleTask (INT8U TaskStart, INT8U TaskNumber)

    Blocks a set of tasks.
- INT8U UnBlockMultipleTask (INT8U TaskStart, INT8U TaskNumber)

    UnBlocks a set of tasks.
- void BRTOS_Init (void)

    Initialize BRTOS control blocks and tick timer (Internal kernel function).
- INT8U OSSchedule (void)

    Priority Preemptive Scheduler (Internal kernel function).
- INT8U SAScheduler (PriorityType ReadyList)

    Sucessive Aproximation Scheduler (Internal kernel function).
- void initEvents (void)

    Initialize event control blocks.
- void **OSUpdateTime** (void)
- void **OSUpdateDate** (void)
- void **OSResetTime** (void)
- void **OSResetDate** (void)
- void **OSUpdateUptime** (void)
- OSTime **OSUptime** (void)
- OSDate **OSUpDate** (void)
- void **CalendarInputSet** (void(∗input)(OS_RTC ∗))
- INT8U **Init_Calendar** (void)
- void **Resync_calendar** (void)
- void **OSUpdateCalendar** (void)
- void **GetCalendar** (OS_RTC ∗rtc)
- void **SetCalendar** (OS_RTC ∗rtc)
- void **GetDateTime** (OSDateTime ∗dt)
- void **GetCalendarTime** (OSTime ∗t)
- void **GetCalendarDate** (OSDate ∗d)

- INT8S **CompareDateTime** (OS_RTC const ∗rtc1, OS_RTC const ∗rtc2)
- void BRTOS_TimerHook (void)

  Provide to the user a function sincronized with the timer tick This function can be used to perform simple tests syncronized with the timer tick.

Variables

- PriorityType **OSReadyList**
- PriorityType **OSBlockedList**
- const PriorityType **PriorityMask** [configMAX_TASK_PRIORITY+1]
- ContextType ∗ **Tail**
- ContextType ∗ **Head**
- INT8U iNesting

  Used to inform if the current code position is an interrupt handler code.
- volatile INT8U currentTask

  Current task being executed.
- volatile INT8U **SelectedTask**
- ContextType ContextTask [NUMBER_OF_TASKS+2]
- INT16U iStackAddress

  Virtual stack counter - Informs the stack occupation in bytes.
- INT8U NumberOfInstalledTasks

  Number of Installed tasks at the moment.
- volatile INT32U OSDuty

  Used to compute the CPU load.
- INT8U PriorityVector [configMAX_TASK_INSTALL]

  Allocate task priorities.
- volatile INT32U OSDutyTmp

  Used to compute the CPU load.
- volatile INT16U LastOSDuty

  Last CPU load computed.
- INT32U TaskAlloc

  Used to search a empty task control block.
- INT16U iQueueAddress

  Queue heap control.
- PGM_P CONST BRTOSStringTable[ ] PROGMEM

  Informs BRTOS version.

4.33.1 Detailed Description

4.33.2 Macro Definition Documentation

4.33.2.1 #define configMAX_TASK_INSTALL 8

Task Defines.

Defines the maximum number of tasks that can be installed

4.33.2.2 #define DELAY 0

Suspended Types.

Task suspended by delay

4.33.2.3 #define IncludeTaskIntoDelayList( )

Value:

```
if(Tail != NULL)                                \
        {                                               \
          /* Insert task into list */                   \
          Tail->Next = Task;                            \
          Task->Previous = Tail;                        \
          Tail = Task;                                  \
          Tail->Next = NULL;                            \
        }                                               \
        else{                                           \
           /* Init delay list */                        \
           Tail = Task;                                 \
           Head = Task;                                 \
           Task->Next = NULL;                           \
           Task->Previous = NULL;                       \
        }
```

4.33.2.4 #define OK (INT8U)0

Error codes.

OK define

4.33.2.5 #define OS_INT_EXIT( )

Value:

```
CriticalDecNesting();                                                   \
  if (!iNesting)                                                          \
  {                                                                       \
    SelectedTask = OSSchedule();                                          \
    if (currentTask != SelectedTask){                                     \
        OS_SAVE_CONTEXT();                                                \
        OS_SAVE_SP();                                                     \
        ContextTask[currentTask].StackPoint = SPvalue;                    \
          currentTask = SelectedTask;                                       \
        SPvalue = ContextTask[currentTask].StackPoint;                    \
        OS_RESTORE_SP();                                                  \
        OS_RESTORE_CONTEXT();                                             \
    }                                                                     \
  }                                                                       \
```

4.33.2.6 #define READY (INT8U)0

Task States.

Task is ready to be executed - waiting for the scheduler authorization

4.33.2.7 #define RemoveFromDelayList( )

Value:

```
if(Task == Head)                            \
        {                                           \
          if(Task == Tail)                          \
          {                                         \
            Tail = NULL;                            \
            Head = NULL;                            \
          }                                         \
          else                                      \
          {                                         \
            Head = Task->Next;                      \
            Head->Previous = NULL;                  \
          }                                         \
        }                                           \
        else                                        \
        {                                           \
          if(Task == Tail)                          \
          {                                         \
            Tail = Task->Previous;                  \
            Tail->Next = NULL;                      \
          }                                         \
          else                                      \
          {                                         \
            Task->Next->Previous = Task->Previous;  \
            Task->Previous->Next = Task->Next;      \
          }                                         \
        }
```

4.33.3 Function Documentation

4.33.3.1 INT8U BlockMultipleTask (INT8U *TaskStart,* INT8U *TaskNumber*)

Blocks a set of tasks.

Parameters

TaskStart	Number of the first task to be blocked
TaskNumber	Number of tasks to be blocked from the specified task start

Returns

OK - Success
IRQ_PEND_ERR - Can not use block multiple tasks function from interrupt handler code

4.33.3.2 INT8U BlockPriority (INT8U *iPriority*)

Blocks a specific priority Blocks the task that is associated with the specified priority. The user must be careful when using this function in together with mutexes. This can lead to undesired results due the "cealing priority" property used in the mutex.

Parameters

iPriority	Priority to be blocked

Returns

OK - Success
IRQ_PEND_ERR - Can not use block priority function from interrupt handler code

4.33.3.3 void BRTOS_TimerHook (void)

Provide to the user a function sincronized with the timer tick This function can be used to perform simple tests syncronized with the timer tick.

Returns

NONE

4.33.3.4 INT8U BRTOSStart (void)

Start the Operating System Scheduler The user must call this function to start the tasks execution.

Returns

OK Success
NO_MEMORY There was not enough memory to start all tasks

4.33.3.5 INT8U DelayTask (INT16U *time_wait*)

Wait for a specified period. A task that calling this function will be suspended for a certain time. When this time is reached the task back to ready state.

Parameters

time	Time in ticks to delay. System default = 1ms. The user can change the time value.

Returns

OK Success
IRQ_PEND_ERR - Can not use block priority function from interrupt handler code

4.33.3.6 INT8U DelayTaskHMSM (INT8U *hours,* INT8U *minutes,* INT8U *seconds,* INT16U *miliseconds*)

Wait for a specified period (in hours, minutes, seconds and miliseconds). A task that calling this function will be suspended for a certain time. When this time is reached the task back to ready state.

Parameters

hours	Hours to delay
minutes	Minutes to delay
seconds	Seconds to delay
miliseconds	Miliseconds to delay

Returns

OK Success
INVALID_TIME The specified parameters are outside of the permitted range

4.33.3.7 void Idle (void)

Idle Task. May be used to implement low power commands.

Returns

NONE

4.33.3.8 void initEvents (void)

Initialize event control blocks.

Returns

NONE

4.33.3.9 INT8U InstallIdle (void(∗)(void) *FctPtr*, INT16U *USER_STACKED_BYTES*)

Install the idle task. Initial state = running.

Parameters

∗*FctPtr*	Pointer to the task to be installed
USER_STACKED_BYTES	Size of the task virtual stack.

Returns

OK Idle task successfully installed
NO_MEMORY Not enough memory available to install the idle task

4.33.3.10 void OSIncCounter (void)

Update the tick counter.

Returns

NONE

4.33.3.11 INT8U OSSchedule (void)

Priority Preemptive Scheduler (Internal kernel function).

BRTOS Scheduler function (Internal kernel function).

4.33.3.12 void PreInstallTasks (void)

Function that initialize the kernel main variables. This function resets the kernel main variables, preparing the system to be started.

Returns

NONE

4.33.3.13 INT8U SAScheduler (PriorityType *ReadyList*)

Sucessive Aproximation Scheduler (Internal kernel function).

Parameters

ReadyList	List of the tasks ready to run

Returns

The priority of the highest priority task ready to run

4.33.3.14 INT8U UnBlockMultipleTask (INT8U *TaskStart,* INT8U *TaskNumber*)

UnBlocks a set of tasks.

Parameters

TaskStart	Number of the first task to be unblocked
TaskNumber	Number of tasks to be unblocked from the specified task start

Returns

OK - Success
IRQ_PEND_ERR - Can not use unblock multiple tasks function from interrupt handler code

4.33.3.15 INT8U UnBlockPriority (INT8U *iPriority*)

UnBlock a specific priority UnBlocks the task that is associated with the specified priority. The user must be careful when using this function in together with mutexes. This can lead to undesired results due the "cealing priority" property used in the mutex.

Parameters

iPriority	Priority to be unblocked

Returns

OK - Success
IRQ_PEND_ERR - Can not use unblock priority function from interrupt handler code

4.33.4 Variable Documentation

4.33.4.1 ContextType ContextTask[NUMBER_OF_TASKS+2]

Task context info ContextTask[0] not used Last ContexTask is the Idle Task

4.34 Simon API

Data Structures

- struct modem_driver_t

Macros

- #define **API_KEY** "90a004390f3530d0ba10199ac2b1ac3d"
- #define **SERVER_NAME** "emon-gpsnetcms.rhcloud.com"
- #define **FALSE** 0
- #define **TRUE** 1
- #define **MODEM_OK** (0)
- #define **MODEM_ERR** (1)
- #define **MAX_HOSTNAME_LEN** (32+1)
- #define **MAX_HOSTIP_LEN** (15+1)
- #define **MAX_APIKEY_LEN** (32+1)
- #define **MAX_GPRS_LEN** (15+1)

Typedefs

- typedef uint8_t(∗ **initialize**) (void)
- typedef uint8_t(∗ **input**) (char ∗, uint16_t ∗)
- typedef uint8_t(∗ **output**) (char ∗, uint16_t)
- typedef uint8_t(∗ **set_host**) (char ∗)
- typedef uint8_t(∗ **set_ip**) (char ∗)
- typedef uint8_t(∗ **get_connect**) (void)
- typedef uint8_t(∗ **resolve_ip**) (char ∗host, char ∗_ip)

Functions

- uint8_t **simon_init** (const modem_driver_t ∗modem)
- uint8_t **simon_get_time** (struct tm ∗t)
- uint8_t simon_send_data (uint8_t ∗buf, uint16_t len, uint8_t mon_id, time_t time)
- uint8_t simon_send_multiple_data (uint8_t ∗buf, uint16_t length, time_t time)
- char ∗ **simon_get_apikey** (void)
- char ∗ **simon_get_hostname** (void)
- char ∗ **simon_get_hostip** (void)
- void **simon_set_apikey** (const char ∗)
- void **simon_set_hostname** (const char ∗)
- void **simon_set_hostip** (const char ∗)
- uint8_t **simon_check_connection** (void)
- void **simon_set_gprs_config** (const char ∗gprs_cfg)
- void **simon_clock_update** (void)
- void **simon_clock_set** (time_t now)
- time_t **simon_clock_get** (void)
- uint8_t **is_simon_clock_synched** (void)
- uint8_t **get_server_time** (char ∗server_reply, struct tm ∗ts)

4.34.1 Detailed Description

4.34.2 Function Documentation

4.34.2.1 uint8_t simon_send_data (uint8_t ∗ *buf,* uint16_t *len,* uint8_t *mon_id,* time_t *time*)

Form request

4.34.2.2 uint8_t simon_send_multiple_data (uint8_t ∗ *buf,* uint16_t *length,* time_t *time*)

Form request

4.35 Devices

Modules

- Comandos AT
- Memória EEPROM
- LCD
- RS485
- RTC DS1307
- Cartão SD
- Sensors
- Terminal I/O
- Modems

4.35.1 Detailed Description

4.36 Modems

Modules

- Modem ESP8266
- Modem GC864
- Modem M590
- Modem

Macros

- #define **NULL_MODEM_UART_BUFSIZE** 32
- #define **NULL_MODEM_UART_TIMEOUT** 10
- #define **NULL_MODEM_MUTEX** FALSE
- #define **NULL_MODEM_MUTEX_PRIO** 0
- #define **null_modem_acquire**()
- #define **null_modem_release**()

Functions

- modem_ret_t **at_null_modem_init** (void)
- modem_ret_t **at_null_modem_open** (void)
- modem_ret_t **at_null_modem_send** (char ∗dados)
- modem_ret_t **at_null_modem_receive** (char ∗buff, uint16_t len)
- modem_ret_t **at_null_modem_close** (void)
- modem_ret_t **at_null_modem_server** (void)
- modem_ret_t **at_null_modem_dns** (char ∗param)
- modem_ret_t **at_null_modem_time** (void)
- CHAR8 **null_modem_getchar** (void)
- uint8_t **null_modem_init** (void)
- uint8_t **null_modem_open** (void)
- uint8_t **null_modem_close** (void)
- uint8_t **null_modem_get_time** (void)
- uint8_t **null_modem_receive** (char ∗buff, uint16_t ∗len)
- uint8_t **null_modem_send** (char ∗dados, uint16_t tam)
- uint8_t **null_modem_set_ip** (char ∗_ip)
- char ∗ **null_modem_get_ip** (void)
- uint8_t **null_modem_set_hostname** (char ∗host)
- char ∗ **null_modem_get_hostname** (void)
- uint8_t **null_modem_host_ip** (void)
- uint8_t **null_modem_check_connection** (void)

4.36.1 Detailed Description

Chapter 5

Data Structure Documentation

5.1 __MB_ANSW_READY_DATA Struct Reference

Data Fields

- uint8_t ∗ **pAnsw**
- uint32_t **answLen**
- uint8_t **errCode**

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

- E:/carlos/PeD/proj-simone/modbus_master/modbus.h

5.2 __MB_QUERY Struct Reference

Data Fields

- uint32_t **state**
- uint8_t **slave**
- uint8_t **func**
- uint8_t **expectedLen**
- uint8_t **queryLen**
- uint8_t ∗ **pQuery**

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

- E:/carlos/PeD/proj-simone/modbus_master/modbus.h

5.3 __MB_QUERY_BUILD Struct Reference

Data Fields

- uint16_t **addr**
- uint16_t **value**
- uint8_t ∗ **pData**

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

- E:/carlos/PeD/proj-simone/modbus_master/modbus.h

5.4 __MB_QUERY_SEND Struct Reference

Data Fields

- __MB_ANSW_READY_DATA **answ**
- __MB_QUERY_BUILD **query**

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

- E:/carlos/PeD/proj-simone/modbus_master/modbus.h

5.5 __UNION_DWORD Union Reference

Data Fields

- uint32_t **data32**
- sint32_t **sdata32**
- float32_t **dataF**
- uint16_t **data16** [2]
- uint8_t **data8** [4]

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

- E:/carlos/PeD/proj-simone/modbus_master/data_types.h

5.6 _OSRTC Struct Reference

Data Fields

- INT16U **Year**
- INT8U **Month**
- INT8U **Day**
- INT8U **Hour**
- INT8U **Min**
- INT8U **Sec**

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

- E:/carlos/PeD/proj-simone/brtos/includes/OS_RTC.h

5.7 Alarmes_t Union Reference

Data Fields

- uint8_t **Val**
- struct {
  uint8_t **Alarme_Temperatura_enrolamento**:1
  uint8_t **Alarme_Temperatura_oleo**:1
  uint8_t **Desligamento_Temperatura_enrolamento**:1
  uint8_t **Desligamento_Temperatura_oleo**:1
  uint8_t **__pad0__**:1
  uint8_t **__pad1__**:1
  uint8_t **__pad2__**:1
  uint8_t **__pad3__**:1
  } **Bits**

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_ts.h

5.8 BRTOS_Mbox Struct Reference

```
#include <BRTOS.h>
```

Data Fields

- INT8U OSEventAllocated

  Indicate if the event is allocated or not.
- INT8U OSEventWait

  Counter of waiting Tasks.
- INT8U OSEventState

  Mailbox state - Defines if the message is available or not.
- PriorityType OSEventWaitList

  Task wait list for event to occur.
- void ∗ OSEventPointer

  Pointer to the message structure / type.

5.8.1 Detailed Description

MailBox Control Block Structure

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.9 BRTOS_Mutex Struct Reference

```
#include <BRTOS.h>
```

Data Fields

- INT8U OSEventAllocated

  Indicate if the event is allocated or not.
- INT8U OSEventState

  Mutex state - Defines if the resource is available or not.
- INT8U OSEventOwner

  Defines mutex owner.
- INT8U OSMaxPriority

  Defines max priority accessing resource.
- INT8U OSOriginalPriority

  Save original priority of Mutex owner task - used to the priority ceiling implementation.
- INT8U OSEventWait

  Counter of waiting Tasks.
- PriorityType OSEventWaitList

  Task wait list for event to occur.

5.9.1 Detailed Description

Mutex Control Block Structure

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.10 BRTOS_Queue Struct Reference

```
#include <BRTOS.h>
```

Data Fields

- INT8U OSEventAllocated

  Indicate if the event is allocated or not.
- INT8U OSEventCount

  Queue Event Count - This value is increased with a post and decremented with a pend.
- INT8U OSEventWait

  Counter of waiting Tasks.
- void ∗ OSEventPointer

  Pointer to queue structure.
- PriorityType OSEventWaitList

  Task wait list for event to occur.

5.10.1 Detailed Description

Queue Control Block Structure

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.11 BRTOS_Sem Struct Reference

```
#include <BRTOS.h>
```

Data Fields

- INT8U OSEventAllocated

    Indicate if the event is allocated or not.
- INT8U OSEventCount

    Semaphore Count - This value is increased with a post and decremented with a pend.
- INT8U OSEventWait

    Counter of waiting Tasks.
- PriorityType OSEventWaitList

    Task wait list for event to occur.

5.11.1 Detailed Description

Semaphore Control Block Structure

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.12 command_t Struct Reference

Data Fields

- const char ∗ **txt**
- cmd_func ∗ **func**
- const char ∗ **help_txt**

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

- E:/carlos/PeD/proj-simone/terminal/terminal.h

5.13 Context Struct Reference

Data Fields

- const CHAR8 ∗ TaskName

  Task name.
- INT16U StackPoint

  Current position of virtual stack pointer.
- INT16U StackInit

  Virtual stack pointer init.
- INT16U TimeToWait

  Time to wait - could be used by delay or timeout.
- INT8U Priority

  Task priority.
- struct Context ∗ **Next**
- struct Context ∗ **Previous**

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.14 ContextType Struct Reference

```
#include <BRTOS.h>
```

5.14.1 Detailed Description

Context Task Structure Used by the task control block

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.15 DIR Struct Reference

Data Fields

- FATFS ∗ **fs**
- WORD **id**
- WORD **index**
- DWORD **sclust**
- DWORD **clust**
- DWORD **sect**
- BYTE ∗ **dir**
- BYTE ∗ **fn**

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

- E:/carlos/PeD/proj-simone/FatFS/ff.h

5.16 Estado_Reles_t Union Reference

Data Fields

- uint8_t **Val**
- struct {
  uint8_t **Estado_Rele_RF1**:1
  uint8_t **Estado_Rele_RF2**:1
  uint8_t **Estado_Rele_Autodiagnostico**:1
  uint8_t **__pad0__**:1
  uint8_t **Estado_Rele_1**:1
  uint8_t **Estado_Rele_2**:1
  uint8_t **Estado_Rele_3**:1
  uint8_t **Estado_Rele_4**:1
  } **Bits**

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_ts.h

5.17 FATFS Struct Reference

Data Fields

- BYTE **fs_type**
- BYTE **drv**
- BYTE **csize**
- BYTE **n_fats**
- BYTE **wflag**
- BYTE **fsi_flag**
- WORD **id**
- WORD **n_rootdir**
- _SYNC_t **sobj**
- DWORD **last_clust**
- DWORD **free_clust**
- DWORD **cdir**
- DWORD **n_fatent**
- DWORD **fsize**
- DWORD **volbase**
- DWORD **fatbase**
- DWORD **dirbase**
- DWORD **database**
- DWORD **winsect**
- BYTE **win** [_MAX_SS]

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

- E:/carlos/PeD/proj-simone/FatFS/ff.h

5.18 FIL Struct Reference

Data Fields

- FATFS ∗ **fs**
- WORD **id**
- BYTE **flag**
- BYTE **err**
- DWORD **fptr**
- DWORD **fsize**
- DWORD **sclust**
- DWORD **clust**
- DWORD **dsect**
- DWORD **dir_sect**
- BYTE ∗ **dir_ptr**
- DWORD ∗ **cltbl**

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

- E:/carlos/PeD/proj-simone/FatFS/ff.h

5.19 FILINFO Struct Reference

Data Fields

- DWORD **fsize**
- WORD **fdate**
- WORD **ftime**
- BYTE **fattrib**
- TCHAR **fname** [13]

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

- E:/carlos/PeD/proj-simone/FatFS/ff.h

5.20 minIni Class Reference

Public Member Functions

- **minIni** (const wxString &filename)
- bool **getbool** (const wxString &Section, const wxString &Key, bool DefValue=false) const
- long **getl** (const wxString &Section, const wxString &Key, long DefValue=0) const
- int **geti** (const wxString &Section, const wxString &Key, int DefValue=0) const
- wxString **gets** (const wxString &Section, const wxString &Key, const wxString &DefValue=wxT("")) const
- wxString **getsection** (int idx) const
- wxString **getkey** (const wxString &Section, int idx) const
- bool **put** (const wxString &Section, const wxString &Key, long Value) const
- bool **put** (const wxString &Section, const wxString &Key, int Value) const
- bool **put** (const wxString &Section, const wxString &Key, bool Value) const
- bool **put** (const wxString &Section, const wxString &Key, const wxString &Value) const
- bool **put** (const wxString &Section, const wxString &Key, const char ∗Value) const
- bool **del** (const wxString &Section, const wxString &Key) const
- bool **del** (const wxString &Section) const

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

- E:/carlos/PeD/proj-simone/minINI/wxMinIni.h

5.21 modbus_null_input_register_list Union Reference

Data Fields

- struct {
  uint8_t **Device_id**
  uint8_t **Entradas**
  uint8_t **Ano**
  uint8_t **Mes**
  uint8_t **Dia**
  uint8_t **Hora**
  uint8_t **Minuto**
  uint8_t **Segundo**
  uint32_t **SD_bytes_available**
  uint32_t **Local_time**
  uint8_t **Pressure_Valve**
  uint8_t **Oil_Level**
  } **Reg**

- uint8_t **Regs8** [NULL_REGLIST_INPUT_NREGS ∗2+NULL_REGLIST_OFFSET_NREGS ∗2]
- uint16_t **Regs16** [NULL_REGLIST_INPUT_NREGS+NULL_REGLIST_OFFSET_NREGS]
- uint32_t **Regs32** [NULL_REGLIST_INPUT_NREGS/2+NULL_REGLIST_OFFSET_NREGS/2]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_slave_null.h

5.22 modbus_pm210_holding_register_list Union Reference

Data Fields

- struct {
  uint16_t **Firmware_Version_Reset_System**
  uint16_t **Firmware_Version_Operating_System**
  uint16_t **Serial_Number_H**
  uint16_t **Serial_Number_L**
  uint16_t **Device_ID**
  uint16_t **Modbus_Address**
  uint16_t **Baud_rate**
  } **Reg**

- uint8_t **Regs8** [PM210_REGLIST_HOLDING_NREGS ∗2]
- uint16_t **Regs16** [PM210_REGLIST_HOLDING_NREGS]
- uint32_t **Regs32** [PM210_REGLIST_HOLDING_NREGS/2]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_pm210.h

5.23 modbus_pm210_input_register_list1 Union Reference

Data Fields

- struct {
  uint8_t **Device_id**
  uint8_t **Entradas**
  uint8_t **Ano**
  uint8_t **Mes**
  uint8_t **Dia**
  uint8_t **Hora**
  uint8_t **Minuto**
  uint8_t **Segundo**
  uint16_t **Real_Energy_Consumption_H**
  uint16_t **Real_Energy_Consumption_L**
  uint16_t **Apparent_Energy_Consumption_H**
  uint16_t **Apparent_Energy_Consumption_L**
  uint16_t **Reactive_Energy_Consumption_H**
  uint16_t **Reactive_Energy_Consumption_L**
  uint16_t **Total_Real_Power**
  uint16_t **Total_Apparent_Power**
  uint16_t **Total_Reactive_Power**
  uint16_t **Total_Power_Factor**
  uint16_t **Frequency**
  uint16_t **Total_Real_Power_Present_Demand**
  uint16_t **Total_Apparent_Power_Present_Demand**
  uint16_t **Total_Reactive_Power_Present_Demand**
  uint16_t **Total_Real_Power_Max_Demand**
  uint16_t **Total_Apparent_Power_Max_Demand**
  uint16_t **Total_Reactive_Power_Max_Demand**
  uint16_t **Current_Instantaneous_Phase_A**
  uint16_t **Current_Instantaneous_Phase_B**
  uint16_t **Current_Instantaneous_Phase_C**
  uint16_t **Current_Present_Demand_Phase_A**
  uint16_t **Current_Present_Demand_Phase_B**
  uint16_t **Current_Present_Demand_Phase_C**
  uint16_t **Current_Max_Demand_Phase_A**
  uint16_t **Current_Max_Demand_Phase_B**
  uint16_t **Current_Max_Demand_Phase_C**
  uint16_t **Voltage_Phase_A_B**
  uint16_t **Voltage_Phase_B_C**
  uint16_t **Voltage_Phase_C_A**
  uint16_t **Voltage_Phase_A_N**
  uint16_t **Voltage_Phase_B_N**
  uint16_t **Voltage_Phase_C_N**
  uint16_t **Scale_Factor_I**
  uint16_t **Scale_Factor_V**
  uint16_t **Scale_Factor_W**
  uint16_t **Scale_Factor_E**
  } **Reg**

- uint8_t **Regs8** [PM210_REGLIST1_INPUT_NREGS ∗2+PM210_REG_OFFSET ∗2]
- uint16_t **Regs16** [PM210_REGLIST1_INPUT_NREGS+PM210_REG_OFFSET]
- uint32_t **Regs32** [PM210_REGLIST1_INPUT_NREGS/2+PM210_REG_OFFSET/2]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_pm210.h

5.24 modbus_pm210_input_register_list2 Union Reference

Data Fields

- struct {
  uint8_t **Device_id**
  uint8_t **Entradas**
  uint8_t **Ano**
  uint8_t **Mes**
  uint8_t **Dia**
  uint8_t **Hora**
  uint8_t **Minuto**
  uint8_t **Segundo**
  uint16_t **Error_Bitmap**
  uint16_t **Thermal_Demand_Interval**
  uint16_t **Power_Block_Demand_Interval**
  uint16_t **Power_Block_Demand_Sub_Intervals**
  uint16_t **CT_Ratio_Primary**
  uint16_t **CT_Ratio_Secondary**
  uint16_t **PT_Ratio_Primary**
  uint16_t **PT_Ratio_Scale**
  uint16_t **PT_Ratio_Secondary**
  uint16_t **Service_Frequency**
  uint16_t **Reset**
  uint16_t **System_Type**
  uint16_t **Units**
  } **Reg**
- uint8_t **Regs8** [PM210_REGLIST2_INPUT_NREGS ∗2+PM210_REG_OFFSET ∗2]
- uint16_t **Regs16** [PM210_REGLIST2_INPUT_NREGS+PM210_REG_OFFSET]
- uint32_t **Regs32** [PM210_REGLIST2_INPUT_NREGS/2+PM210_REG_OFFSET/2]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_pm210.h

5.25 modbus_slave_t Struct Reference

Data Fields

- slave_num_t **num**
- char ∗ **nome**
- _reader **slave_reader**

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_slaves.h

5.26 modbus_t500_input_register_list1 Union Reference

Data Fields

- struct {
  uint8_t **Device_id**
  uint8_t **Entradas**
  uint8_t **Ano**
  uint8_t **Mes**
  uint8_t **Dia**
  uint8_t **Hora**
  uint8_t **Minuto**
  uint8_t **Segundo**
  uint32_t **Voltage_Phase_Avg**
  uint32_t **Current_Phase_Avg**
  uint32_t **Voltage_Line_Avg**
  uint32_t **Total_Power_Factor_Sign**
  uint32_t **Total_Real_Power**
  uint32_t **Total_Reactive_Power**
  uint32_t **Total_Apparent_Power**
  uint32_t **Current_Angle_Phase_A**
  uint32_t **Total_Power_Factor**
  uint32_t **Caract_Power_Factor**
  uint32_t **Frequency**
  } **Reg**
- uint32_t **Regs32** [T500_REGLIST1_INPUT_NREGS+T500_REG_OFFSET/2]
- uint16_t **Regs16** [T500_REGLIST1_INPUT_NREGS ∗2+T500_REG_OFFSET]
- uint8_t **Regs8** [T500_REGLIST1_INPUT_NREGS ∗4+T500_REG_OFFSET ∗2]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_t500.h

5.27 modbus_ts_holding_register_list Union Reference

Data Fields

- struct {
  uint8_t **Parametro_ALMO**
  uint8_t **Parametro_DSLO**
  uint8_t **Parametro_RDSO**
  uint8_t **Parametro_ALME**
  uint8_t **Parametro_DSLE**
  uint8_t **Parametro_RDSE**
  uint8_t **Parametro_IDI**
  uint8_t **Parametro_RTDS**
  uint8_t **Parametro_DISP**
  uint8_t **Parametro_ALO**
  uint8_t **Parametro_DSO**
  uint8_t **Parametro_ALE**
  uint8_t **Parametro_DSE**
  uint8_t **Parametro_RL**

uint8_t **Parametro_VSAN**
uint8_t **Parametro_FSAN**
uint8_t **Parametro_FESA**
uint8_t **Parametro_IESA**
uint8_t **Parametro_GEO**
uint8_t **Parametro_TE**
uint8_t **Parametro_HS_MAIS**
uint8_t **Parametro_HS_AST**
uint8_t **Parametro_2M**
uint8_t **Parametro_CNT**
uint8_t **Parametro_CNS**
uint8_t **Parametro_RF1**
uint8_t **Parametro_RF2**
uint8_t **Parametro_HIS**
uint8_t **Parametro_ALT**
uint8_t **Parametro_CV1**
uint8_t **Parametro_CV2**
uint8_t **Parametro_HIC**
uint8_t **Parametro_EVH**
uint8_t **Parametro_EVM**
uint8_t **Parametro_TEV**
uint8_t **Parametro_HLOG**
uint8_t **Parametro_TLOG**
uint8_t **Parametro_RLOG**
uint8_t **Parametro_MES**
uint8_t **Parametro_DIA**
uint8_t **Parametro_ANO**
uint8_t **Parametro_HORA**
uint8_t **Parametro_MIN**
uint8_t **Modo_RF1**
uint8_t **Modo_RF2**
} **Reg**

- uint8_t **Regs8** [TS_REG_HOLDING_NREGS]
- uint16_t **Regs16** [TS_REG_HOLDING_NREGS/2]
- uint32_t **Regs32** [TS_REG_HOLDING_NREGS/4]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_ts.h

5.28 modbus_ts_input_register_list Union Reference

Data Fields

- struct {
uint8_t **Device_id**
uint8_t **Entradas**
uint8_t **Ano**
uint8_t **Mes**
uint8_t **Dia**
uint8_t **Hora**
uint8_t **Minuto**
uint8_t **Segundo**

uint16_t **Temperatura_oleo**
uint16_t **Temperatura_enrolamento**
uint16_t **Temperatura_RTD2**
uint16_t **Temperatura_RTD3**
uint16_t **Temperatura_maxima_oleo**
uint16_t **Temperatura_maxima_enrolamento**
uint16_t **Temperatura_maxima_RTD2**
uint16_t **Temperatura_maxima_RTD3**
uint16_t **Gradiente_Final_Temperatura**
uint16_t **Percentual_carga**
uint16_t **Corrente_secundario_TC**
uint16_t **Corrente_transformador**
uint16_t **Estado_Reles**
uint16_t **Variavel_erros**
uint16_t **Opcionais**
uint16_t **Reles**
} **Reg**

- uint8_t **Regs8** [TS_REG_INPUT_NREGS ∗2+TS_REG_OFFSET ∗2]
- uint16_t **Regs16** [TS_REG_INPUT_NREGS+TS_REG_OFFSET]
- uint32_t **Regs32** [TS_REG_INPUT_NREGS/2+TS_REG_OFFSET/2]

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_ts.h

5.29 modem_driver_t Struct Reference

Data Fields

- initialize **init**
- input **receive**
- output **send**
- set_host **sethost**
- set_ip **setip**
- get_connect **is_connected**
- resolve_ip **resolveip**

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

- E:/carlos/PeD/proj-simone/simon-api/simon-api.h

5.30 monitor_config_ok_t Union Reference

Data Fields

- uint8_t **byte**
- struct {
  uint8_t **num_mon_ok**:1
  uint8_t **server_ok**:1
  uint8_t **ip_ok**:1
  uint8_t **key_ok**:1
  uint8_t **gprs_apn_ok**:1
  uint8_t **gprs_user_ok**:1
  uint8_t **gprs_pwd_ok**:1
  } **bit**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.31 monitor_entry_t Struct Reference

Data Fields

- time_t **ts**
- uint8_t **size**
- uint8_t ∗ **values**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.32 monitor_header_t Struct Reference

Data Fields

- monitor_headerl1_t **h1**
- monitor_headerl2_t **h2**
- uint16_t **last_idx**
- uint16_t **count**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.33 monitor_headerl1_t Struct Reference

Data Fields

- uint8_t **version**
- uint8_t **mon_id**
- uint16_t **entry_size**
- uint16_t **time_interv**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.34 monitor_headerl2_t Struct Reference

Data Fields

- uint16_t **year**
- uint8_t **mon**
- uint8_t **mday**
- uint8_t **hour**
- uint8_t **min**
- uint8_t **sec**
- uint8_t **synched**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.35 monitor_state_t Struct Reference

Data Fields

- monitor_used_t **state**
- char **monitor_name_writing** [FILENAME_MAX_LENGTH]
- char **monitor_name_reading** [FILENAME_MAX_LENGTH]
- char **monitor_dir_name** [FILENAME_MAX_LENGTH]
- monitor_headerl1_t **config_h**
- mon_timer_t **read_timer**
- mon_timer_t **write_timer**
- pt_t **read_pt**
- pt_t **write_pt**
- uint8_t **slave_addr**
- uint8_t **codigo**
- uint8_t **sinc**
- uint8_t **sending**
- uint8_t **uploading**
- data_reader **read_data**

- uint32_t **written_entries**
- uint32_t **total_written_entries**
- uint32_t **read_entries**
- uint32_t **sent_entries**
- uint32_t **failed_tx**
- time_t **last_timestamp**
- time_t **sinc_time**
- uint32_t **tx_start**
- uint32_t **tx_time**
- uint32_t **tx_time_avg**
- clock_t **reader_upload_start_time**
- uint32_t **reader_upload_time**
- uint32_t **reader_upload_time_avg**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.36 monitors_state_t Struct Reference

Data Fields

- time_t **time_started**
- uint8_t **monitores_em_uso**
- uint8_t **is_idle**
- uint8_t **uploading**
- uint8_t **running**
- uint8_t **is_connected**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.37 Opcionais_t Union Reference

Data Fields

- uint8_t **Val**
- struct {
    uint8_t **Opcional_RTDs_adicionais**:1
    uint8_t **Opcional_Saida_Analogica**:1
    uint8_t **Opcional_RS485**:1
    uint8_t **Unused**:1
    uint8_t **Opcional_Memoria_Massa**:1
    uint8_t **Opcional_Pre_resfriamento**:1
    uint8_t **Opcional_Exercicio_ventiladores**:1
    uint8_t **__pad0__**:1
  } **Bits**

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_ts.h

5.38 OS_DQUEUE Struct Reference

Data Fields

- INT8U ∗ OSQStart

    Pointer to the queue start.
- INT8U ∗ OSQEnd

    Pointer to the queue end.
- INT8U ∗ OSQIn

    Pointer to the next queue entry.
- INT8U ∗ OSQOut

    Pointer to the next data in the queue output.
- INT16U OSQTSize

    Size of the queue type - Defined in the create queue function.
- INT16U OSQLength

    Length of the queue - Defined in the create queue function.
- INT16U OSQEntries

    Size of data inside the queue.

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.39 OS_QUEUE Struct Reference

```
#include <BRTOS.h>
```

Data Fields

- INT8U ∗ OSQStart

    Pointer to the queue start.
- INT8U ∗ OSQEnd

    Pointer to the queue end.
- INT8U ∗ OSQIn

    Pointer to the next queue entry.
- INT8U ∗ OSQOut

    Pointer to the next data in the queue output.
- INT16U OSQSize

    Size of the queue - Defined in the create queue function.
- INT16U OSQEntries

    Size of data inside the queue.

5.39.1 Detailed Description

Queue Control Block Structure

Dynamic Queue Control Block Structure

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.40 OS_QUEUE_16 Struct Reference

Data Fields

- INT16U ∗ OSQStart

    Pointer to the queue start.
- INT16U ∗ OSQEnd

    Pointer to the queue end.
- INT16U ∗ OSQIn

    Pointer to the next queue entry.
- INT16U ∗ OSQOut

    Pointer to the next data in the queue output.
- INT16U OSQSize

    Size of the queue - Defined in the create queue function.
- INT16U OSQEntries

    Size of data inside the queue.

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.41 OS_QUEUE_32 Struct Reference

Data Fields

- INT32U ∗ OSQStart

    Pointer to the queue start.
- INT32U ∗ OSQEnd

    Pointer to the queue end.
- INT32U ∗ OSQIn

    Pointer to the next queue entry.
- INT32U ∗ OSQOut

    Pointer to the next data in the queue output.
- INT16U OSQSize

    Size of the queue - Defined in the create queue function.
- INT16U OSQEntries

    Size of data inside the queue.

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

- E:/carlos/PeD/proj-simone/brtos/includes/BRTOS.h

5.42 OSDate Struct Reference

```
#include <OS_RTC.h>
```

Data Fields

- INT8U RTC_Day

  Day of the date.
- INT8U RTC_Month

  Month of the date.
- INT16U RTC_Year

  Year of the date.

5.42.1 Detailed Description

Operating System Date - Shows the current day, month and year ou the uptime info

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

- E:/carlos/PeD/proj-simone/brtos/includes/OS_RTC.h

5.43 OSDateTime Struct Reference

```
#include <OS_RTC.h>
```

Data Fields

- OSDate **date**
- OSTime **time**

5.43.1 Detailed Description

Operating System Date and time - Shows the current date and time

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

- E:/carlos/PeD/proj-simone/brtos/includes/OS_RTC.h

5.44 OSTime Struct Reference

```
#include <OS_RTC.h>
```

Data Fields

- INT8U RTC_Second

  Seconds of the clock.
- INT8U RTC_Minute

  Minutes of the clock.
- INT8U RTC_Hour

  Hours of the clock.

5.44.1 Detailed Description

Real time clock - shows the current hours, minutes and seconds or the uptime info

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

- E:/carlos/PeD/proj-simone/brtos/includes/OS_RTC.h

5.45 OSTime_Date Struct Reference

```
#include <OS_RTC.h>
```

5.45.1 Detailed Description

Operating System Date and time - Shows the current time and date

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

- E:/carlos/PeD/proj-simone/brtos/includes/OS_RTC.h

5.46 OSTimeDate Struct Reference

Data Fields

- OSTime **time**
- OSDate **date**

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

- E:/carlos/PeD/proj-simone/brtos/includes/OS_RTC.h

5.47 pt Struct Reference

Data Fields

- lc_t **lc**

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

- E:/carlos/PeD/proj-simone/monitor/pt/pt.h

5.48 pt_sem Struct Reference

Data Fields

- unsigned int **count**

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

- E:/carlos/PeD/proj-simone/monitor/pt/pt-sem.h

5.49 putbuff Struct Reference

Data Fields

- FIL ∗ **fp**
- int **idx**
- int **nchr**
- BYTE **buf** [64]

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

- E:/carlos/PeD/proj-simone/FatFS/ff.c

5.50 RTC_DS1307 Struct Reference

Estrutura para manter informacoes do calendario.

```
#include <rtc_ds1307.h>
```

Data Fields

- INT8U Sec
- INT8U Min
- INT8U Hour
- INT8U Day
- INT8U DayOfWeek
- INT8U Month
- INT16U Year

5.50.1 Detailed Description

Estrutura para manter informacoes do calendario.

Os dados da estrutura devem ser lidos do DS1307 usando as funcoes deste arquivo.

5.50.2 Field Documentation

5.50.2.1 INT8U Day

Dia do mes (1-31)

5.50.2.2 INT8U DayOfWeek

Dia da semana (1-7)

5.50.2.3 INT8U Hour

Horas (1-12 no formato AM/PM) (0-23 no formato 24h)

5.50.2.4 INT8U Min

Minutos (0-59)

5.50.2.5 INT8U Month

Mes (1-12)

5.50.2.6 INT8U Sec

Segundos (0-59)

5.50.2.7 INT16U Year

Ano (2000-2099)

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

- E:/carlos/PeD/proj-simone/devices/rtc_ds1307.h

5.51 T16_8 Union Reference

Data Fields

- uint8_t **u8** [2]
- uint16_t **u16**

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

- E:/carlos/PeD/proj-simone/devices/SD.h

5.52 T32_8 Union Reference

Data Fields

- uint8_t **bytes** [4]
- uint32_t **lword**

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

- E:/carlos/PeD/proj-simone/devices/SD.h

5.53 timer Struct Reference

Data Fields

- clock_t **start**
- clock_t **interval**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.54 timestamp_t Struct Reference

Data Fields

- uint16_t **year**
- uint8_t **mon**
- uint8_t **mday**
- uint8_t **hour**
- uint8_t **min**
- uint8_t **sec**

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

- E:/carlos/PeD/proj-simone/monitor/monitor.h

5.55 U8 Union Reference

Data Fields

- uint8_t **Byte**
- struct {
  uint8_t **b0**:1
  uint8_t **b1**:1
  uint8_t **b2**:1
  uint8_t **b3**:1
  uint8_t **b4**:1
  uint8_t **b5**:1
  uint8_t **b6**:1
  uint8_t **b7**:1
  } **Bits**

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

- E:/carlos/PeD/proj-simone/modbus_slaves/modbus_ts.h