

Caixa Preta

Generated by Doxygen 1.8.11

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	Button Class Reference	5
3.1.1	Constructor & Destructor Documentation	5
3.1.1.1	Button()	5
3.1.2	Member Function Documentation	5
3.1.2.1	readBuffer(char *key)	5
3.1.2.2	readButtons()	5
3.2	EEPROM Class Reference	6
3.2.1	Constructor & Destructor Documentation	6
3.2.1.1	EEPROM()	6
3.2.1.2	EEPROM(uint8_t frecuencia, bool byteA1, bool byteA0)	6
3.2.2	Member Function Documentation	6
3.2.2.1	getPosition()	7
3.2.2.2	read(const void *data, uint32_t numBytes)	7
3.2.2.3	read(uint32_t address, const void *data, uint32_t numBytes)	7
3.2.2.4	readByte()	7
3.2.2.5	readFloat()	8
3.2.2.6	readInt()	8

3.2.2.7	readUInt()	8
3.2.2.8	setPosition(uint32_t position)	8
3.2.2.9	write(const void *data, uint32_t numBytes)	8
3.2.2.10	write(uint32_t address, const void *data, uint32_t numBytes)	9
3.2.2.11	writeByte(uint8_t data)	9
3.2.2.12	writeFloat(float data)	9
3.2.2.13	writeInt(int32_t data)	9
3.2.2.14	writeUInt(uint32_t data)	10
3.3	Firmware Class Reference	10
3.4	GPSdata Struct Reference	11
3.5	LCD Class Reference	11
3.5.1	Constructor & Destructor Documentation	12
3.5.1.1	LCD()	12
3.5.2	Member Function Documentation	12
3.5.2.1	backLight(bool on)	12
3.5.2.2	caracter(byte code, byte *vet)	12
3.5.2.3	command(byte dado)	12
3.5.2.4	cursor(byte on)	12
3.5.2.5	isBusy()	12
3.5.2.6	loadBuffer(byte *vet, byte lin, byte col)	13
3.5.2.7	loadBufferWhite()	13
3.5.2.8	loadLCD(byte lin, byte coli, byte colf)	13
3.5.2.9	printChar(byte dado)	13
3.5.2.10	printChar8(byte nr, byte lin, byte col)	13
3.5.2.11	printDec16(word nr, byte lin, byte col)	14
3.5.2.12	printDec8(byte nr, byte lin, byte col)	14
3.5.2.13	printHex16(word nr, byte lin, byte col)	14
3.5.2.14	printHex8(byte nr, byte lin, byte col)	14
3.5.2.15	readAddressCounter()	14
3.5.2.16	setCursor(byte pos)	15

3.5.2.17	startConfig()	15
3.5.3	Member Data Documentation	15
3.5.3.1	lcd_buffer	15
3.5.3.2	lcd_flag0	15
3.5.3.3	lcd_flag1	15
3.6	Leds Class Reference	15
3.6.1	Constructor & Destructor Documentation	15
3.6.1.1	Leds()	15
3.6.2	Member Function Documentation	15
3.6.2.1	green(byte on)	15
3.6.2.2	red(byte on)	16
3.6.2.3	turnOffAll()	16
3.6.2.4	turnOnAll()	16
3.6.2.5	yellow(byte on)	16
3.7	MPU Class Reference	16
3.7.1	Constructor & Destructor Documentation	17
3.7.1.1	MPU()	17
3.7.1.2	MPU(int freq)	17
3.7.2	Member Function Documentation	17
3.7.2.1	calibrate(int16_t *bias, float *valor)	17
3.7.2.2	getAcelRes()	18
3.7.2.3	getGiroRes()	18
3.7.2.4	readAccelTempGyros()	18
3.7.2.5	readAverageAccelTempGyros(word rpt)	18
3.7.2.6	readBlockData(byte reg, byte *dado, byte qtd)	18
3.7.2.7	readRegister(byte reg, uint8_t *dado)	18
3.7.2.8	selfTest()	19
3.7.2.9	setScale(byte gfs, byte afs)	19
3.7.2.10	sleep()	19
3.7.2.11	wakeUp()	19

3.7.2.12	whoAml()	19
3.7.2.13	writeBlockData(byte reg, byte *dado, byte qtd)	19
3.7.2.14	writeRegister(byte reg, byte dado)	20
3.7.3	Member Data Documentation	20
3.7.3.1	axi	20
3.7.3.2	ayi	20
3.7.3.3	azi	20
3.7.3.4	gxi	20
3.7.3.5	gyi	20
3.7.3.6	gzi	21
3.7.3.7	tpi	21
3.8	RawDegrees Struct Reference	21
3.9	SPI Class Reference	21
3.9.1	Constructor & Destructor Documentation	21
3.9.1.1	SPI()	21
3.9.1.2	SPI(int slaveSelect, bool mestre=true)	21
3.9.2	Member Function Documentation	22
3.9.2.1	start()	22
3.9.2.2	stop()	22
3.9.2.3	transferir(uint8_t dado)	22
3.10	SRAM Class Reference	22
3.10.1	Constructor & Destructor Documentation	22
3.10.1.1	SRAM()	22
3.10.1.2	SRAM(int port)	22
3.10.2	Member Function Documentation	23
3.10.2.1	getMode()	23
3.10.2.2	readSeq(uint32_t address, int16_t *data, uint32_t num)	23
3.10.2.3	setMode(uint8_t mode)	23
3.10.2.4	writeSeq(uint32_t address, int16_t *data, uint16_t num)	23
3.11	Timer5 Class Reference	24

3.11.1	Constructor & Destructor Documentation	24
3.11.1.1	Timer5(LCD *initLCD, Button *initBtn)	24
3.12	TinyGPSAltitude Struct Reference	24
3.13	TinyGPSCourse Struct Reference	25
3.14	TinyGPSCustom Class Reference	25
3.15	TinyGPSDate Struct Reference	25
3.16	TinyGPSDecimal Struct Reference	26
3.17	TinyGPSHDOP Struct Reference	26
3.18	TinyGPSInteger Struct Reference	27
3.19	TinyGPSLocation Struct Reference	27
3.20	TinyGPSPlus Class Reference	28
3.21	TinyGPSSpeed Struct Reference	28
3.22	TinyGPSTime Struct Reference	29
3.23	TWI Class Reference	29
3.23.1	Constructor & Destructor Documentation	29
3.23.1.1	TWI(int freq)	29
3.23.2	Member Function Documentation	30
3.23.2.1	readData(bool ack, uint8_t *data)	30
3.23.2.2	sendData(byte dado)	30
3.23.2.3	sendReadAddress(byte address)	30
3.23.2.4	sendRepeatedStart()	30
3.23.2.5	sendStart()	31
3.23.2.6	sendStop()	31
3.23.2.7	sendWriteAddress(byte address)	31
3.24	USART Class Reference	31
3.24.1	Constructor & Destructor Documentation	32
3.24.1.1	USART()	32
3.24.1.2	USART(int serialNumber, long baudRate)	32
3.24.2	Member Function Documentation	32
3.24.2.1	isAvailable()	32
3.24.2.2	print(char dado[])	32
3.24.2.3	print(long dado)	32
3.24.2.4	print(int dado)	33
3.24.2.5	print(double dado)	33
3.24.2.6	print(word dado, int modo=DEC)	33
3.24.2.7	print(byte dado, int modo=DEC)	33
3.24.2.8	println(char dado[])	33
3.24.2.9	println(long dado)	34
3.24.2.10	println(int dado)	34
3.24.2.11	println(double dado)	34
3.24.2.12	println(word dado, int modo=DEC)	34
3.24.2.13	println(byte dado, int modo=DEC)	34
3.24.2.14	readByte()	35

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Button	5
EEPROM	6
Firmware	10
GPSdata	11
LCD	11
Leds	15
MPU	16
RawDegrees	21
SPI	21
SRAM	22
Timer5	24
TinyGPSCustom	25
TinyGPSDate	25
TinyGPSDecimal	26
TinyGPSAltitude	24
TinyGPSCourse	25
TinyGPSHDOP	26
TinyGPSSpeed	28
TinyGPSInteger	27
TinyGPSLocation	27
TinyGPSPlus	28
TinyGPSTime	29
TWI	29
USART	31

Chapter 2

Class Index

2.1 Class List

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

Button	5
EEPROM	6
Firmware	10
GPSdata	11
LCD	11
Leds	15
MPU	16
RawDegrees	21
SPI	21
SRAM	22
Timer5	24
TinyGPSAltitude	24
TinyGPSCourse	25
TinyGPSCustom	25
TinyGPSDate	25
TinyGPSDecimal	26
TinyGPSHDOP	26
TinyGPSInteger	27
TinyGPSLocation	27
TinyGPSPlus	28
TinyGPSSpeed	28
TinyGPSTime	29
TWI	29
USART	31

Chapter 3

Class Documentation

3.1 Button Class Reference

Public Member Functions

- **Button** ()
- void **readButtons** ()
- bool **readBuffer** (char *key)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Button::Button ()

Construtor da classe. Inicializa os botões.

3.1.2 Member Function Documentation

3.1.2.1 bool Button::readBuffer (char * key)

Tira uma chave do buffer e retorna se houve sucesso

Parameters

<i>key</i>	ponteiro para a variável que vai receber o valor da chave acionada
------------	--

Returns

TRUE se havia chave no buffer e FALSE se não havia chave no buffer

3.1.2.2 void Button::readButtons ()

Lê as chaves e coloca no buffer. Monitora as chaves em 20 Hz

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

- headers/Button.h

3.2 EEPROM Class Reference

Public Member Functions

- **EEPROM** ()
- **EEPROM** (uint8_t frequencia, bool byteA1, bool byteA0)
- uint8_t **readByte** ()
- uint32_t **read** (const void *data, uint32_t numBytes)
- uint32_t **read** (uint32_t address, const void *data, uint32_t numBytes)
- bool **writeByte** (uint8_t data)
- uint32_t **write** (const void *data, uint32_t numBytes)
- uint32_t **write** (uint32_t address, const void *data, uint32_t numBytes)
- float **readFloat** ()
- bool **writeFloat** (float data)
- uint32_t **readUInt** ()
- bool **writeUInt** (uint32_t data)
- int32_t **readInt** ()
- bool **writeInt** (int32_t data)
- uint32_t **getPosition** ()
- bool **setPosition** (uint32_t position)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 EEPROM::EEPROM ()

Construtor padrão da classe

3.2.1.2 EEPROM::EEPROM (uint8_t frequencia, bool byteA1, bool byteA0)

Construtor da classe. Recebe a frequência da comunicação **TWI** (p. 29) e os bytes A0 e A1 do seletor do barramento da **EEPROM** (p. 6). Inicializa contadores da classe da posição da **EEPROM** (p. 6). Inicializa também o endereço do chip

Parameters

<i>frequencia</i>	frequencia do TWI (p. 29)
<i>byteA1</i>	MSB do chip selector do barramento da eeprom
<i>byteA0</i>	LSB do chip selector do barramento da eeprom

3.2.2 Member Function Documentation

3.2.2.1 uint32_t EEPROM::getPosition ()

Pega a posição/endereço que a classe está utilizando para fazer as operações

Returns

posição/endereço atual

3.2.2.2 uint32_t EEPROM::read (const void * *data*, uint32_t *numBytes*)

Lê dados da memória utilizando como endereço o local atual na memória, definido pelo contador interno da classe

Parameters

<i>data</i>	valor que foi lido
<i>numBytes</i>	número de bytes que serão lidos

Returns

número de bytes efetivamente lidos

3.2.2.3 uint32_t EEPROM::read (uint32_t *address*, const void * *data*, uint32_t *numBytes*)

Lê dados da memória. Se a leitura passar por mais de 2 páginas, divide as leituras.

Parameters

<i>address</i>	endereço de 17 bits em que vai ser escrito o dado
<i>data</i>	valor que foi lido
<i>numBytes</i>	número de bytes que serão lidos

Returns

número de bytes efetivamente lidos

3.2.2.4 uint8_t EEPROM::readByte ()

Lê um byte da memória utilizando como endereço o local atual na memória, definido pelo contador interno da classe

Returns

valor do byte lido

3.2.2.5 float EEPROM::readFloat ()

Lê um float da memória

Returns

valor do float ou NAN se inválido

3.2.2.6 int32_t EEPROM::readInt ()

Lê um inteiro da memória

Returns

valor do inteiro

3.2.2.7 uint32_t EEPROM::readUInt ()

Lê um unsigned int da memória

Returns

valor do unsigned int

3.2.2.8 bool EEPROM::setPosition (uint32_t *position*)

Define a posição/endereço que a classe vai utilizar para fazer as operações

Parameters

<i>position</i>	nova posição/endereço
-----------------	-----------------------

Returns

TRUE se operação ocorreu com sucesso, FALSE caso contrário

3.2.2.9 uint32_t EEPROM::write (const void * *data*, uint32_t *numBytes*)

Escreve dados na memória utilizando como endereço o local atual na memória, definido pelo contador interno da classe

Parameters

<i>data</i>	valor que vai ser escrito
<i>numBytes</i>	número de bytes que serão escritos

Returns

número de bytes efetivamente escritos

3.2.2.10 `uint32_t EEPROM::write (uint32_t address, const void * data, uint32_t numBytes)`

Escreve dados na memória. Se a escrita passar por mais de 1 página (128 bytes), divide as escritas.

Parameters

<i>address</i>	endereço de 17 bits em que vai ser escrito o dado
<i>data</i>	valor que foi lido
<i>numBytes</i>	número de bytes que serão lidos

Returns

número de bytes efetivamente lidos

3.2.2.11 `bool EEPROM::writeByte (uint8_t data)`

Escreve 1 byte na memória utilizando como endereço o local atual na memória, definido pelo contador interno da classe

Parameters

<i>data</i>	byte que vai ser escrito
-------------	--------------------------

Returns

TRUE se ocorreu tudo bem, FALSE caso contrário

3.2.2.12 `bool EEPROM::writeFloat (float data)`

Escreve um float na memória

Parameters

<i>data</i>	float que vai ser escrito
-------------	---------------------------

Returns

TRUE se operação ocorreu com sucesso, FALSE caso contrário

3.2.2.13 `bool EEPROM::writeInt (int32_t data)`

Escreve um inteiro na memória

Parameters

<i>data</i>	inteiro que vai ser escrito
-------------	-----------------------------

Returns

TRUE se operação ocorreu com sucesso, FALSE caso contrário

3.2.2.14 bool EEPROM::writeUInt (uint32_t data)

Escreve um unsigned int na memória

Parameters

<i>data</i>	unsigned int que vai ser escrito
-------------	----------------------------------

Returns

TRUE se operação ocorreu com sucesso, FALSE caso contrário

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

- headers/EEPROM.h

3.3 Firmware Class Reference

Public Member Functions

- **Firmware** (LCD *lcd, Button *btn)
- void **testGPS** ()
- void **testMPU** ()
- void **rotina** ()
- void **readData** ()
- void **processData** ()
- void **readProcessedData** ()

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

- headers/Firmware.h

3.4 GPSdata Struct Reference

Public Attributes

- float **lat**
- float **lng**
- float **veloc**
- uint8_t **hora**
- uint8_t **minuto**
- uint8_t **segundo**
- uint8_t **dia**
- uint8_t **mes**
- uint16_t **ano**

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

- headers/Firmware.h

3.5 LCD Class Reference

Public Member Functions

- **LCD** ()
- void **startConfig** ()
- void **character** (byte code, byte *vet)
- void **printDec16** (word nr, byte lin, byte col)
- void **printDec8** (byte nr, byte lin, byte col)
- void **printHex16** (word nr, byte lin, byte col)
- void **printHex8** (byte nr, byte lin, byte col)
- void **printChar8** (byte nr, byte lin, byte col)
- void **loadBuffer** (byte *vet, byte lin, byte col)
- void **loadLCD** (byte lin, byte coli, byte colf)
- void **loadBufferWhite** ()
- void **cursor** (byte on)
- void **command** (byte dado)
- void **printChar** (byte dado)
- void **setCursor** (byte pos)
- byte **readAddressCounter** ()
- bool **isBusy** ()
- void **backLight** (bool on)

Public Attributes

- byte **lcd_buffer** [4][20]
- volatile bool **lcd_flag0**
- volatile bool **lcd_flag1**

3.5.1 Constructor & Destructor Documentation

3.5.1.1 LCD::LCD ()

Construtor da classe. Inicia os pinos.

3.5.2 Member Function Documentation

3.5.2.1 void LCD::backLight (bool *on*)

Acende e apaga o Back Light

Parameters

<i>on</i>	TRUE para ligar o back light e FALSE para desligar o back light
-----------	---

3.5.2.2 void LCD::caracter (byte *code*, byte * *vet*)

Cria um caracterer especial

Parameters

<i>code</i>	código do caracter
<i>vet</i>	vetor de 8 posições com o mapa de pontos

3.5.2.3 void LCD::command (byte *dado*)

Escrever um comando (instrução) no **LCD** (p. 11)

Parameters

<i>dado</i>	instrução
-------------	-----------

3.5.2.4 void LCD::cursor (byte *on*)

Liga e desliga o cursor piscante

Parameters

<i>on</i>	ON para ligar o cursor e OFF para desligar o cursor
-----------	---

3.5.2.5 bool LCD::isBusy ()

Verifica se o **LCD** (p. 11) está ocupado

Returns

TRUE se tiver ocupado e FALSE se estiver livre

3.5.2.6 void LCD::loadBuffer (byte * *vet*, byte *lin*, byte *col*)

Trasferir uma string para o lcd_buffer. Transfere até encontrar o "\0"

Parameters

<i>vet</i>	ponteiro para string
<i>lin</i>	linha de início
<i>col</i>	coluna de início

3.5.2.7 void LCD::loadBufferWhite ()

Colocar branco (0x20) em todo o buffer do **LCD** (p. 11)

3.5.2.8 void LCD::loadLCD (byte *lin*, byte *coli*, byte *colf*)

Trasferir do lcd_buffer para o **LCD** (p. 11). Só transfere uma linha

Parameters

<i>lin</i>	linha de início
<i>coli</i>	coluna de início
<i>colf</i>	coluna final (inclusa)

3.5.2.9 void LCD::printChar (byte *dado*)

Escrever uma caracter no **LCD** (p. 11)

Parameters

<i>dado</i>	caracter
-------------	----------

3.5.2.10 void LCD::printChar8 (byte *nr*, byte *lin*, byte *col*)

Imprime uma letra em 8 bits

Parameters

<i>nr</i>	char
<i>lin</i>	linha de início
<i>col</i>	coluna de início

3.5.2.11 void LCD::printDec16 (word *nr*, byte *lin*, byte *col*)

Imprime um decimal de 16 bits

Parameters

<i>nr</i>	número
<i>lin</i>	linha de início
<i>col</i>	coluna de início

3.5.2.12 void LCD::printDec8 (byte *nr*, byte *lin*, byte *col*)

Imprime um decimal de 8 bits

Parameters

<i>nr</i>	número
<i>lin</i>	linha de início
<i>col</i>	coluna de início

3.5.2.13 void LCD::printHex16 (word *nr*, byte *lin*, byte *col*)

Imprime um hexadecimal de 16 bits

Parameters

<i>nr</i>	número
<i>lin</i>	linha de início
<i>col</i>	coluna de início

3.5.2.14 void LCD::printHex8 (byte *nr*, byte *lin*, byte *col*)

Imprime um hexadecimal de 8 bits

Parameters

<i>nr</i>	número
<i>lin</i>	linha de início
<i>col</i>	coluna de início

3.5.2.15 byte LCD::readAddressCounter ()

Ler contador de endereços

Returns

Contador de endereços

3.5.2.16 void LCD::setCursor (byte *pos*)

Posicionar o cursor

Parameters

<i>pos</i>	posição
------------	---------

3.5.2.17 void LCD::startConfig ()

Configura o **LCD** (p. 11) para começar o uso

3.5.3 Member Data Documentation

3.5.3.1 byte LCD::lcd_buffer[4][20]

Buffer para o **LCD** (p. 11) [lin][col] terminar cada linha com Zero

3.5.3.2 volatile bool LCD::lcd_flag0

Flag para indicar que a linha 0 deve atualizar

3.5.3.3 volatile bool LCD::lcd_flag1

Flag para indicar que a linha 1 deve atualizar

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

- headers/LCD.h

3.6 Leds Class Reference

Public Member Functions

- **Leds** ()
- void **green** (byte on)
- void **yellow** (byte on)
- void **red** (byte on)
- void **turnOffAll** ()
- void **turnOnAll** ()

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Leds::Leds ()

Construtor da classe. Inicia as portas dos **Leds** (p. 15) como saída e os desliga.

3.6.2 Member Function Documentation

3.6.2.1 void Leds::green (byte *on*)

Liga, desliga ou troca o estado do led verde

Parameters

<i>on</i>	ON para ligar, OFF para desligar e TOGGLE para trocar o estado
-----------	--

3.6.2.2 void Leds::red (byte *on*)

Liga, desliga ou troca o estado do led vermelho

Parameters

<i>on</i>	ON para ligar, OFF para desligar e TOGGLE para trocar o estado
-----------	--

3.6.2.3 void Leds::turnOffAll ()

Desliga todos os leds

3.6.2.4 void Leds::turnOnAll ()

Liga todos os leds

3.6.2.5 void Leds::yellow (byte *on*)

Liga, desliga ou troca o estado do led amarelo

Parameters

<i>on</i>	ON para ligar, OFF para desligar e TOGGLE para trocar o estado
-----------	--

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

- headers/Leds.h

3.7 MPU Class Reference

Public Member Functions

- **MPU** ()
- **MPU** (int freq)
- bool **writeRegister** (byte reg, byte dado)
- bool **readRegister** (byte reg, uint8_t *dado)
- bool **writeBlockData** (byte reg, byte *dado, byte qtd)
- void **readBlockData** (byte reg, byte *dado, byte qtd)
- byte **whoAml** ()

- void **wakeUp** ()
- void **sleep** ()
- void **setScale** (byte gfs, byte afs)
- void **readAverageAccelTempGyros** (word rpt)
- void **readAccelTempGyros** ()
- bool **selfTest** ()
- void **calibrate** (int16_t *bias, float *valor)
- float **getAcelRes** ()
- float **getGiroRes** ()

Public Attributes

- int **axi**
- int **ayi**
- int **azi**
- int **tpi**
- int **gxi**
- int **gyi**
- int **gzi**

3.7.1 Constructor & Destructor Documentation

3.7.1.1 MPU::MPU ()

Construtor default da classe.

3.7.1.2 MPU::MPU (int *freq*)

Construtor da classe. Coloca o **MPU** (p. 16) num estado conhecido. Algumas operações podem ser redundantes.

Parameters

<i>freq</i>	frequência utilizada
-------------	----------------------

3.7.2 Member Function Documentation

3.7.2.1 void MPU::calibrate (int16_t * *bias*, float * *valor*)

Calibra o acelerômetro e o giroscópio do **MPU** (p. 16)

Parameters

<i>bias</i>	
<i>valor</i>	

3.7.2.2 float MPU::getAcelRes ()

Retorna a resolução do acelerômetro

Returns

float com a resolução

3.7.2.3 float MPU::getGiroRes ()

Retorna a resolução do giroscópio

Returns

float com a resolução

3.7.2.4 void MPU::readAccelTempGyros ()

Lê a aceleração, temperatura e giroscópio

3.7.2.5 void MPU::readAverageAccelTempGyros (word rpt)

Lê a média da aceleração, temperatura e giroscópio

Parameters

<i>rpt</i>	número de leituras que vai ser tirada a média
------------	---

3.7.2.6 void MPU::readBlockData (byte reg, byte * dado, byte qtd)

Lê um bloco de dados no **MPU** (p. 16) a partir de um registrador

Parameters

<i>reg</i>	registrador
<i>dado</i>	bloco de dados em que os dados da leitura serão armazenados
<i>qtd</i>	tamanho do bloco de dados

3.7.2.7 bool MPU::readRegister (byte reg, uint8_t * dado)

Ler um registrador do **MPU** (p. 16)

Parameters

<i>reg</i>	registrador
<i>dado</i>	byte que foi lido

Returns

TRUE se operação ocorreu com sucesso, FALSE caso contrário

3.7.2.8 bool MPU::selfTest ()

Faz um self test do **MPU** (p. 16)

Returns

TRUE se passar do teste, FALSE caso contrário

3.7.2.9 void MPU::setScale (byte *gfs*, byte *afs*)

Seleciona Fundo de Escalas para o **MPU** (p. 16)

Parameters

<i>gfs</i>	FS do giroscópio
<i>afs</i>	FS do acelerômetro

3.7.2.10 void MPU::sleep ()

Dormir o **MPU** (p. 16) e programar para usar relógio Giro X

3.7.2.11 void MPU::wakeUp ()

Acordar o **MPU** (p. 16) e programar para usar relógio Giro X

3.7.2.12 byte MPU::whoAmI ()

Lê o registrador WHO_AM_I

Returns

valor do registrador

3.7.2.13 bool MPU::writeBlockData (byte *reg*, byte * *dado*, byte *qtd*)

Escrever um bloco de dados no **MPU** (p. 16) a partir de um registrador

Parameters

<i>reg</i>	registrador
<i>dado</i>	bloco de dados que vai ser escrito
<i>qtd</i>	tamanho do bloco de dados

Returns

TRUE se dados foram escritos, FALSE caso contrário

3.7.2.14 bool MPU::writeRegister (byte *reg*, byte *dado*)

Escrever em um registrador do **MPU** (p. 16)

Parameters

<i>reg</i>	registrador
<i>dado</i>	dado que vai ser escrito

Returns

TRUE se dado foi escrito, FALSE caso contrário

3.7.3 Member Data Documentation

3.7.3.1 int MPU::axi

leitura instantânea da aceleração no eixo x

3.7.3.2 int MPU::ayi

leitura instantânea da aceleração no eixo y

3.7.3.3 int MPU::azi

leitura instantânea da aceleração no eixo z

3.7.3.4 int MPU::gxi

leitura instantânea do giroscópio no eixo x

3.7.3.5 int MPU::gyi

leitura instantânea do giroscópio no eixo y

3.7.3.6 int MPU::gzi

leitura instantânea do giroscópio no eixo z

3.7.3.7 int MPU::tpi

leitura instantânea da temperatura interna do **MPU** (p. 16)

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

- headers/MPU.h

3.8 RawDegrees Struct Reference

Public Attributes

- uint16_t **deg**
- uint32_t **billionths**
- bool **negative**

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

- headers/TinyGPS++.h

3.9 SPI Class Reference

Public Member Functions

- **SPI** ()
- **SPI** (int slaveSelect, bool mestre=true)
- void **start** ()
- void **stop** ()
- uint8_t **transferir** (uint8_t dado)

3.9.1 Constructor & Destructor Documentation

3.9.1.1 SPI::SPI ()

Construtor default da classe.

3.9.1.2 SPI::SPI (int *slaveSelect*, bool *mestre* =true)

Construtor da classe **SPI** (p. 21). Define a porta que o escravo está e se vai iniciar como escravo ou mestre, sendo default iniciar como mestre slaveSelect porta em que o escravo está conectado mestre TRUE se for mestre (padrão) e FALSE se for escravo

3.9.2 Member Function Documentation

3.9.2.1 void SPI::start ()

Coloca a porta do slave em nível baixo, sinalizando que o Arduino o escolheu

3.9.2.2 void SPI::stop ()

Coloca a porta do slave em nível alto, liberando o canal para outra escolha de escravo

3.9.2.3 uint8_t SPI::transfer (uint8_t *dado*)

Envia e recebe um byte ao mesmo tempo pelo protocolo de comunicação **SPI** (p. 21)

Parameters

<i>dado</i>	dado em 8 bits que vai ser enviado
-------------	------------------------------------

Returns

dado que foi recebido

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

- headers/SPI.h

3.10 SRAM Class Reference

Public Member Functions

- **SRAM** ()
- **SRAM** (int port)
- uint8_t **getMode** ()
- void **setMode** (uint8_t mode)
- void **writeSeq** (uint32_t address, int16_t *data, uint16_t num)
- void **readSeq** (uint32_t address, int16_t *data, uint32_t num)

3.10.1 Constructor & Destructor Documentation

3.10.1.1 SRAM::SRAM ()

Construtor default da classe.

3.10.1.2 SRAM::SRAM (int *port*)

Construtor que recebe qual o pino que o Arduino está usando para selecionar a **SRAM** (p. 22)

Parameters

<i>port</i>	pino do arduino que conecta no SS da SRAM (p. 22)
-------------	--

3.10.2 Member Function Documentation

3.10.2.1 uint8_t SRAM::getMode ()

Lê o registrador de modo de operação da **SRAM** (p. 22)

Returns

byte que representa o modo de operação atual

3.10.2.2 void SRAM::readSeq (uint32_t address, int16_t * data, uint32_t num)

Lê uma sequência de dados da **SRAM** (p. 22) a partir de um endereço

Parameters

<i>address</i>	endereço em que o dado será lido
<i>data</i>	vetor de dados de 16 bits que receberá a informação
<i>num</i>	tamanho do vetor data

3.10.2.3 void SRAM::setMode (uint8_t mode)

Define o modo de operação da **SRAM** (p. 22) escrevendo no registrador de modo

Parameters

<i>byte</i>	que define o modo de operação que será utilizado
-------------	--

3.10.2.4 void SRAM::writeSeq (uint32_t address, int16_t * data, uint16_t num)

Escreve uma sequência de dados na **SRAM** (p. 22) a partir de um endereço

Parameters

<i>address</i>	endereço que o dado será escrito
<i>data</i>	vetor de dados de 16 bits que será escrito
<i>num</i>	tamanho do vetor data

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

- headers/SRAM.h

3.11 Timer5 Class Reference

Public Member Functions

- **Timer5** (**LCD** *initLCD, **Button** *initBtn)

3.11.1 Constructor & Destructor Documentation

3.11.1.1 Timer5::Timer5 (**LCD** * *initLCD*, **Button** * *initBtn*)

Construtor da classe. Inicia os parâmetros do timer 5 e guarda a referencia da classe do **LCD** (p. 11)

Parameters

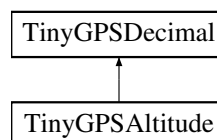
<i>initLCD</i>	endereço da classe que cuida do LCD (p. 11)
<i>initBtn</i>	endereço da classe que cuida dos botões

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

- headers/Timer5.h

3.12 TinyGPSAltitude Struct Reference

Inheritance diagram for TinyGPSAltitude:



Public Member Functions

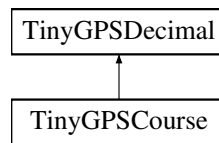
- double **meters** ()
- double **miles** ()
- double **kilometers** ()
- double **feet** ()

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

- headers/TinyGPS++.h

3.13 TinyGPSCourse Struct Reference

Inheritance diagram for TinyGPSCourse:



Public Member Functions

- double **deg** ()

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

- headers/TinyGPS++.h

3.14 TinyGPSCustom Class Reference

Public Member Functions

- **TinyGPSCustom** (**TinyGPSPlus** &gps, const char *sentenceName, int termNumber)
- void **begin** (**TinyGPSPlus** &gps, const char *_sentenceName, int _termNumber)
- bool **isUpdated** () const
- bool **isValid** () const
- uint32_t **age** () const
- const char * **value** ()

Friends

- class **TinyGPSPlus**

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

- headers/TinyGPS++.h

3.15 TinyGPSDate Struct Reference

Public Member Functions

- bool **isValid** () const
- bool **isUpdated** () const
- uint32_t **age** () const
- uint32_t **value** ()
- uint16_t **year** ()
- uint8_t **month** ()
- uint8_t **day** ()

Friends

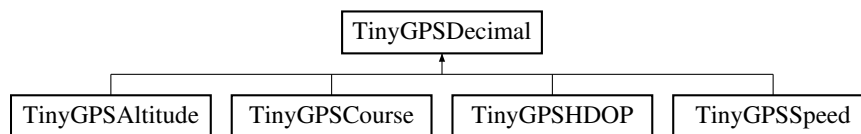
- class **TinyGPSPlus**

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

- headers/TinyGPS++.h

3.16 TinyGPSDecimal Struct Reference

Inheritance diagram for TinyGPSDecimal:



Public Member Functions

- bool **isValid** () const
- bool **isUpdated** () const
- uint32_t **age** () const
- int32_t **value** ()

Friends

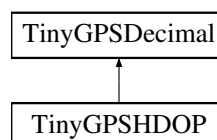
- class **TinyGPSPlus**

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

- headers/TinyGPS++.h

3.17 TinyGPSHDOP Struct Reference

Inheritance diagram for TinyGPSHDOP:



Public Member Functions

- double **hdop** ()

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

- headers/TinyGPS++.h

3.18 TinyGPSInteger Struct Reference

Public Member Functions

- bool **isValid** () const
- bool **isUpdated** () const
- uint32_t **age** () const
- uint32_t **value** ()

Friends

- class **TinyGPSPPlus**

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

- headers/TinyGPS++.h

3.19 TinyGPSLocation Struct Reference

Public Member Functions

- bool **isValid** () const
- bool **isUpdated** () const
- uint32_t **age** () const
- const **RawDegrees** & **rawLat** ()
- const **RawDegrees** & **rawLng** ()
- double **lat** ()
- double **lng** ()

Friends

- class **TinyGPSPPlus**

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

- headers/TinyGPS++.h

3.20 TinyGPSPlus Class Reference

Public Member Functions

- bool **encode** (char c)
- **TinyGPSPlus** & **operator<<** (char c)
- uint32_t **charsProcessed** () const
- uint32_t **sentencesWithFix** () const
- uint32_t **failedChecksum** () const
- uint32_t **passedChecksum** () const

Static Public Member Functions

- static const char * **libraryVersion** ()
- static double **distanceBetween** (double lat1, double long1, double lat2, double long2)
- static double **courseTo** (double lat1, double long1, double lat2, double long2)
- static const char * **cardinal** (double course)
- static int32_t **parseDecimal** (const char *term)
- static void **parseDegrees** (const char *term, **RawDegrees** °)

Public Attributes

- **TinyGPSLocation** location
- **TinyGPSDate** date
- **TinyGPSTime** time
- **TinyGPSSpeed** speed
- **TinyGPSCourse** course
- **TinyGPSAltitude** altitude
- **TinyGPSInteger** satellites
- **TinyGPSHDOP** hdop

Friends

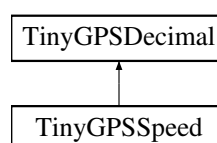
- class **TinyGPSCustom**

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

- headers/TinyGPS++.h

3.21 TinyGPSSpeed Struct Reference

Inheritance diagram for TinyGPSSpeed:



Public Member Functions

- double **knots** ()
- double **mph** ()
- double **mps** ()
- double **kmph** ()

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

- headers/TinyGPS++.h

3.22 TinyGPSTime Struct Reference

Public Member Functions

- bool **isValid** () const
- bool **isUpdated** () const
- uint32_t **age** () const
- uint32_t **value** ()
- uint8_t **hour** ()
- uint8_t **minute** ()
- uint8_t **second** ()
- uint8_t **centisecond** ()

Friends

- class **TinyGPSPlus**

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

- headers/TinyGPS++.h

3.23 TWI Class Reference

Public Member Functions

- **TWI** (int freq)
- bool **sendStart** ()
- bool **sendRepeatedStart** ()
- void **sendStop** ()
- bool **sendReadAddress** (byte address)
- bool **sendWriteAddress** (byte address)
- bool **sendData** (byte dado)
- bool **readData** (bool ack, uint8_t *data)

3.23.1 Constructor & Destructor Documentation

3.23.1.1 TWI::TWI (int freq)

Construtor da classe. Define a frequência que vai ser trabalhada e o endereço do escravo.

Parameters

<i>freq</i>	frequência utilizada
-------------	----------------------

3.23.2 Member Function Documentation

3.23.2.1 bool TWI::readData (bool *ack*, uint8_t * *data*)

Lê um dado.

Parameters

<i>ack</i>	determina se quer receber ACK (TRUE) ou NACK (FALSE)
<i>data</i>	byte lido

Returns

TRUE se ocorreu tudo bem, FALSE caso contrario

3.23.2.2 bool TWI::sendData (byte *dado*)

Envia um dado.

Parameters

<i>dado</i>	dado que vai ser enviado
-------------	--------------------------

Returns

TRUE se o dado foi enviado e FALSE caso contrário

3.23.2.3 bool TWI::sendReadAddress (byte *address*)

Envia o endereço de leitura.

Parameters

<i>address</i>	endereço de leitura
----------------	---------------------

Returns

TRUE se endereço foi enviado e FALSE caso contrário

3.23.2.4 bool TWI::sendRepeatedStart ()

Envia um sinal de start repetido. Caso falhe, repete REPEAT_START vezes.

Returns

TRUE se start repetido ocorreu e FALSE caso contrário

3.23.2.5 bool TWI::sendStart ()

Envia um sinal de start. Caso falhe, repete REPEAT_START vezes.

Returns

TRUE se start ocorreu e FALSE caso contrário

3.23.2.6 void TWI::sendStop ()

Envia um sinal de stop.

3.23.2.7 bool TWI::sendWriteAddress (byte *address*)

Envia o endereço de escrita.

Parameters

<i>address</i>	endereço de escrita
----------------	---------------------

Returns

TRUE se endereço foi enviado e FALSE caso contrário

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

- headers/TWI.h

3.24 USART Class Reference

Public Member Functions

- **USART** ()
- **USART** (int serialNumber, long baudRate)
- bool **isAvailable** ()
- byte **readByte** ()
- void **print** (char dado[])
- void **print** (long dado)
- void **print** (int dado)
- void **print** (double dado)
- void **print** (word dado, int modo=DEC)
- void **print** (byte dado, int modo=DEC)

- void **println** (char dado[])
- void **println** (long dado)
- void **println** (int dado)
- void **println** (double dado)
- void **println** (word dado, int modo=DEC)
- void **println** (byte dado, int modo=DEC)

3.24.1 Constructor & Destructor Documentation

3.24.1.1 USART::USART ()

Construtor default da classe.

3.24.1.2 USART::USART (int *serialNumber*, long *baudRate*)

Construtor da classe. Inicia uma comunicação serial em uma das portas do arduino utilizando o baud rate informado

Parameters

<i>serialNumber</i>	qual porta serial será utilizada. No mega pode ser 0, 1, 2 ou 3
<i>baudRate</i>	baud Rate da comunicação.

3.24.2 Member Function Documentation

3.24.2.1 bool USART::isAvailable ()

Verifica se existe algo para ler na porta serial

Returns

TRUE se existir, FALSE se não

3.24.2.2 void USART::print (char *dado*[])

Imprime uma string

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.3 void USART::print (long *dado*)

Imprime em decimal com sinal um long, retirando os zeros à esquerda

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.4 void USART::print (int *dado*)

Imprime em decimal com sinal um int, retirando os zeros à esquerda

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.5 void USART::print (double *dado*)

Imprime em double xx.xxxxxx

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.6 void USART::print (word *dado*, int *modo* = DEC)

Se o modo for decimal, imprime em decimal sem sinal um W16. Se o modo for hexadecimal, imprime em hexa um palavra de 16 bits. Possui como padrão o modo decimal.

Parameters

<i>dado</i>	dado que será impresso
<i>modo</i>	DEC para decimal e HEX para hexadecimal

3.24.2.7 void USART::print (byte *dado*, int *modo* = DEC)

Se o modo for decimal, imprime em decimal sem sinal um byte retirando os zeros à esquerda. Se o modo for hexadecimal, imprime em Hexa um byte. Possui como padrão o modo decimal.

Parameters

<i>dado</i>	dado que será impresso
<i>modo</i>	DEC para decimal e HEX para hexadecimal

3.24.2.8 void USART::println (char *dado*[])

Imprime uma string e pula uma linha

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.9 void USART::println (long *dado*)

Imprime em decimal com sinal um long e pula uma linha

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.10 void USART::println (int *dado*)

Imprime em decimal com sinal um int e pula uma linha

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.11 void USART::println (double *dado*)

Imprime em double xx.xxxxxx e pula uma linha

Parameters

<i>dado</i>	dado que será impresso
-------------	------------------------

3.24.2.12 void USART::println (word *dado*, int *modo* = DEC)

Se o modo for decimal, imprime em decimal sem sinal um W16 e pula uma linha. Se o modo for hexadecimal, imprime em hexa um palavra de 16 bits e pula uma linha. Possui como padrão o modo decimal.

Parameters

<i>dado</i>	dado que será impresso
<i>modo</i>	DEC para decimal e HEX para hexadecimal

3.24.2.13 void USART::println (byte *dado*, int *modo* = DEC)

Se o modo for decimal, imprime em decimal sem sinal um byte retirando os zeros à esquerda e pula uma linha. Se o modo for hexadecimal, imprime em Hexa um byte e pula uma linha. Possui como padrão o modo decimal.

Parameters

<i>dado</i>	dado que será impresso
<i>modo</i>	DEC para decimal e HEX para hexadecimal

3.24.2.14 byte USART::readByte ()

Lê um byte da porta serial.

Returns

byte lido

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

- headers/USART.h

Index

axi
 MPU, 20

ayi
 MPU, 20

azi
 MPU, 20

backLight
 LCD, 12

Button, 5
 Button, 5
 readBuffer, 5
 readButtons, 5

calibrate
 MPU, 17

character
 LCD, 12

command
 LCD, 12

cursor
 LCD, 12

EEPROM, 6
 EEPROM, 6
 getPosition, 6
 read, 7
 readByte, 7
 readFloat, 7
 readInt, 8
 readUInt, 8
 setPosition, 8
 write, 8, 9
 writeByte, 9
 writeFloat, 9
 writeInt, 9
 writeUInt, 10

Firmware, 10

GPSdata, 11

getAcelRes
 MPU, 17

getGiroRes
 MPU, 18

getMode
 SRAM, 23

getPosition
 EEPROM, 6

green
 Leds, 15

gxi
 MPU, 20

gyi
 MPU, 20

gzi
 MPU, 20

isAvailable
 USART, 32

isBusy
 LCD, 12

LCD, 11
 backLight, 12
 character, 12
 command, 12
 cursor, 12
 isBusy, 12
 LCD, 12
 lcd_buffer, 15
 lcd_flag0, 15
 lcd_flag1, 15
 loadBuffer, 13
 loadBufferWhite, 13
 loadLCD, 13
 printChar, 13
 printChar8, 13
 printDec16, 14
 printDec8, 14
 printHex16, 14
 printHex8, 14
 readAddressCounter, 14
 setCursor, 14
 startConfig, 15

lcd_buffer
 LCD, 15

lcd_flag0
 LCD, 15

lcd_flag1
 LCD, 15

Leds, 15
 green, 15
 Leds, 15
 red, 16
 turnOffAll, 16
 turnOnAll, 16
 yellow, 16

loadBuffer
 LCD, 13
loadBufferWhite

- LCD, 13
- loadLCD
 - LCD, 13
- MPU, 16
 - axi, 20
 - ayi, 20
 - azi, 20
 - calibrate, 17
 - getAcelRes, 17
 - getGiroRes, 18
 - gxi, 20
 - gyi, 20
 - gzi, 20
 - MPU, 17
 - readAccelTempGyros, 18
 - readAverageAccelTempGyros, 18
 - readBlockData, 18
 - readRegister, 18
 - selfTest, 19
 - setScale, 19
 - sleep, 19
 - tpi, 21
 - wakeUp, 19
 - whoAml, 19
 - writeBlockData, 19
 - writeRegister, 20
- print
 - USART, 32, 33
- printChar
 - LCD, 13
- printChar8
 - LCD, 13
- printDec16
 - LCD, 14
- printDec8
 - LCD, 14
- printHex16
 - LCD, 14
- printHex8
 - LCD, 14
- println
 - USART, 33, 34
- RawDegrees, 21
- read
 - EEPROM, 7
- readAccelTempGyros
 - MPU, 18
- readAddressCounter
 - LCD, 14
- readAverageAccelTempGyros
 - MPU, 18
- readBlockData
 - MPU, 18
- readBuffer
 - Button, 5
- readButtons
 - Button, 5
- readByte
 - EEPROM, 7
 - USART, 35
- readData
 - TWI, 30
- readFloat
 - EEPROM, 7
- readInt
 - EEPROM, 8
- readRegister
 - MPU, 18
- readSeq
 - SRAM, 23
- readUInt
 - EEPROM, 8
- red
 - Leds, 16
- SPI, 21
 - SPI, 21
 - start, 22
 - stop, 22
 - transferir, 22
- SRAM, 22
 - getMode, 23
 - readSeq, 23
 - SRAM, 22
 - setMode, 23
 - writeSeq, 23
- selfTest
 - MPU, 19
- sendData
 - TWI, 30
- sendReadAddress
 - TWI, 30
- sendRepeatedStart
 - TWI, 30
- sendStart
 - TWI, 31
- sendStop
 - TWI, 31
- sendWriteAddress
 - TWI, 31
- setCursor
 - LCD, 14
- setMode
 - SRAM, 23
- setPosition
 - EEPROM, 8
- setScale
 - MPU, 19
- sleep
 - MPU, 19
- start
 - SPI, 22
- startConfig
 - LCD, 15
- stop

- SPI, 22
- TWI, 29
 - readData, 30
 - sendData, 30
 - sendReadAddress, 30
 - sendRepeatedStart, 30
 - sendStart, 31
 - sendStop, 31
 - sendWriteAddress, 31
- TWI, 29
- Timer5, 24
 - Timer5, 24
- TinyGPSAltitude, 24
- TinyGPSCourse, 25
- TinyGPSCustom, 25
- TinyGPSDate, 25
- TinyGPSDecimal, 26
- TinyGPSHDOP, 26
- TinyGPSInteger, 27
- TinyGPSLocation, 27
- TinyGPSPlus, 28
- TinyGPSSpeed, 28
- TinyGPSTime, 29
- tpi
 - MPU, 21
- transferir
 - SPI, 22
- turnOffAll
 - Leds, 16
- turnOnAll
 - Leds, 16
- USART, 31
 - isAvailable, 32
 - print, 32, 33
 - println, 33, 34
 - readByte, 35
 - USART, 32
- wakeUp
 - MPU, 19
- whoAml
 - MPU, 19
- write
 - EEPROM, 8, 9
- writeBlockData
 - MPU, 19
- writeByte
 - EEPROM, 9
- writeFloat
 - EEPROM, 9
- writeInt
 - EEPROM, 9
- writeRegister
 - MPU, 20
- writeSeq
 - SRAM, 23
- writeUInt
- EEPROM, 10
- yellow
 - Leds, 16