**Biblioteca acesso à memória EEPROM do Atmega 2560 (4.096 bytes) 4 KB24**

Versão 1.0, 19/04/2020

Funções

|  |  |  |
| --- | --- | --- |
|  |  |  |
| byte | eeprom\_cf\_mostra | (void) |
| void | eeprom\_dump | (word adr, word qtd) |
| long | eeprom\_rd\_32b | (word adr) |
| void | eeprom\_rd\_32b | (word adr, long dado) |
| int | eeprom\_rd\_16b | (word adr) |
| void | eeprom\_rd\_16b | (word adr, word dado) |
| void | eeprom\_rd\_str | (word adr, byte \*msg, word qtd) |
| void | eeprom\_wr\_str | (word adr, byte \*msg) |
| void | eeprom\_wr\_ff | (void) |
| void | eeprom\_rd\_blk | (word adr, byte \*vet, word qtd) |
| void | eeprom\_wr\_blk | (word adr, byte \*vet, byte qtd) |
| byte | eeprom\_rd | (word adr) |
| void | eeprom\_wr | (word adr, byte dado) |
|  |  |  |

(#define EEPROM\_TAM 4096)

Em todo acesso à EEPROM, o campo de endereços é truncado em 12 bits. Não há verificação do limite do endereço fornecido. Durante a escrita, as interrupções são momentaneamente desabilitadas.

* byte **eeprom\_cf\_mostra** (void)

Mostrar Calibração de Fábrica gravada na EEPROM.

* void **eeprom\_dump** (word adr, word qtd)

Dump da EEPROM. Mostra qtd bytes a partir do endereço adr.

Usa a função ser\_dump\_memo (long adr, char \*vet) para mostrar uma linha.

* long **eeprom\_rd\_32b** (word adr)

Ler um valor de 32 bits da EEPROM, Big Endian.

* void **eeprom\_rd\_32b** (word adr, long dado)

Escrever um valor de 32 bits da EEPROM, Big Endian.

* int **eeprom\_rd\_16b** (word adr)

Ler um valor de 16 bits da EEPROM, Big Endian.

* void **eeprom\_rd\_16b** (word adr, word dado)

Escrever um valor de 16 bits da EEPROM, Big Endian.

* void **eeprom\_rd\_str** (word adr, byte \*msg, word qtd)

Ler uma string à partir do endereço adr da EEPROM. Copia o zero final.

Qtd indica a quantidade máxima, e caso se chegue a esse valor, o último byte é o zero final.

* void **eeprom\_wr\_str** (word adr, byte \*msg)

Gravar uma string na EEPROM, incluindo o zero final.

* void **eeprom\_wr\_ff** (void)

Gravar 0xFF em toda a EEPROM.

* void **eeprom\_rd\_blk** (word adr, byte \*vet, word qtd)

Ler um bloco a partir de um endereço. O endereço **adr** é truncado em 12 bits.

* void **eeprom\_wr\_blk** (word adr, byte \*vet, byte qtd)

Escrever um bloco a partir de um endereço. O endereço **adr** é truncado em 12 bits.

* byte **eeprom\_rd** (word adr)

Ler um endereço da EEPROM. O endereço **adr** é truncado em 12 bits.

* byte **eeprom\_wr** (word adr, byte dado)

Escrever o dado no endereço adr da memória. O endereço **adr** é truncado em 12 bits.

Nesta escrita, as interrupções são momentaneamente desabilitadas.

GP RMC - Recebe uma string e se for RMC, a imprime.

Retorna TRUE se era RMC (imprime).

Retorna FALSE se não era (nada imprime).

Seq Vazia = $GPRMC,,V,,,,,,,,,,N\*53

01234567890123456789012

* byte **gps\_gsa\_ser** (char \*vt)

GP GSA - Recbe uma string e se for GSA, a imprime.

Retorna TRUE se era GSA (imprime).

Retorna FALSE se não era (nada imprime).

Seq Vazia = $GPGSA,A,3,,,,,,,,,,,,N\*53

01234567890123456789012

* void **gps\_str** (byte \*msg)

Enviar msg para GPS.

* void **gps\_char** (byte dt)

Enviar um char para o GPS.

Não usa interrupção.

* void **gps\_rx\_fila\_config** (void)

Inicializar fila GPS RX serial.

* char **gps\_rx\_poe** (char cha)

Colocar um byte na fila RX do GPS.

* void **gps\_fila\_rx\_cheia** (void)

Mensagem de Fila RX Cheia do GPS.

* char **gps\_rx\_tira** (char \*cha)

Tirar um byte da fila RX do GPS.

* void **gps\_config** (long br)

Configurar porta serial 3.

Não habilita TX e nem RX.

Não habilita interrupções.

* ISR(USART3\_TX\_vect)

TX3: Interrupção por dado enviado

* ISR(USART3\_RX\_vect)

RX3: Interrupção por dado recebido

volatile char gps\_tx\_fila[GPS\_TX\_FILA\_TAM]; //Espaço para a fila serial de TX

volatile byte gps\_tx\_pin, gps\_tx\_pout; //Ponteiros para usar a fila

volatile byte gps\_tx\_ok; //Indica que terminou transmissão

volatile char gps\_rx\_fila[GPS\_RX\_FILA\_TAM]; //Espaço para a fila serial de RX

volatile byte gps\_rx\_pin, gps\_rx\_pout; //Ponteiros para usar a fila

volatile byte gps\_rx\_ok; //Indica que terminou recepção

volatile byte gps\_msg\_0[200]; //(fase=0) Buffer 0 usado pela interrupção para receber GPRMC

volatile byte gps\_msg\_1[200]; //(fase=1) Buffer 1 usado pela interrupção para receber GPRMC

volatile byte gps\_msg\_ix; //Indexador para escrever nos buffers;

volatile byte gps\_msg\_fase; //Fase para receber GPRMC (0=gps\_rmc\_0 e 1=gps\_rmc\_1);

volatile byte gps\_msg\_ok; //Indica que completou o recebimento de uma mensagem;

------------------------------------------------------------------------------------------------------------------------------------------------

GPRMC

$GPRMC,hhmmss,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode\*cs<CR><LF>

$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A\*57

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| $GPRMC, | hhmmmss.sss, | Stat, | Lat:ddmm.mmmmm, | N/S, | Long:dddmm.mmmmm, | E/W, |
| $GPRMC, | 083559.00, | A, | 4717.11437, | N, | 00833.91522, | E, |
| 7 | 12 | 2 | 11 | 2 | 12 | 2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Speed:ddd.ddd, | Curso:ddd.ddd, | Data:ddmmyy | Mv, | mvE, | Modo | \*Check | CR LF |
| 0.004, | 77.52, | 091202, | , | , | A | \*57 | 0xD 0xA |
| ?8 | ?8 | 7 | ?8 | ?2 | 1 | 3 | 2 |

Tamanh0 = 80 bytes, vou usar tamanho 100.

Lido com Arduino: $GPRMC,131732.00,A,1548.62581,S,04748.65809,W,0.299,,260120,,,A\*72

------------------------------------------------------------------------------------------------------------------------------------------------

GPGSA

$GPGSA,Smode,FS{,sv},PDOP,HDOP,VDOP\*cs<CR><LF>

$GPGSA,A,3,23,29,07,08,09,18,26,28,,,,,1.94,1.18,1.54\*0D

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $GPGSA, | Smode, | Fix, | Sat1, | Sat2, | Sat3, | Sat4, | Sat5, | Sat6, | Sat7, | Sat8 |
| $GPRMC, | A, | 3, | 23, | 29, | 07, | 08, | 09, | 18, | 26, | 28, |
| 7 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Sat9, | Sat10 | Sat11, | Sat12, | PDOP, | HDOP, | VDOP | \*Check | CR LF |
| 18, | 18, | 18, | 18, | 1.94, | 1.18, | 1.54 | \*0D | 0xD 0xA |
| 3 | 3 | 3 | 3 | 5 | 5 | 5 | 3 | 2 |

Tamanho = 67 bytes, vou usar tamanho 100.

Lido com Arduino: $GPGSA,A,3,07,09,16,23,04,01,,,,,,,8.16,1.24,8.06\*09