**Biblioteca acesso à memória FLASH (24LC1025)**

20/04/2020, versão 1.0

Faixas de endereços das memórias Flash (2 memórias de 128 KB cada)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Início | Fim | Flash | Bloco | Define | Define |
| 0 0000 | 0 FFFF | 1 | 0 | FLASH1\_ADR+0 | 0x50 |
| 1 0000 | 1 FFFF | 1 | 1 | FLASH1\_ADR+4 | 0x54 |
| 2 0000 | 2 FFFF | 2 | 0 | FLASH2\_ADR+0 | 0x51 |
| 3 0000 | 3 FFFF | 2 | 1 | FLASH2\_ADR+4 | 0x55 |

As duas memórias flash vão trabalhar com uma só de 256 KB de 0x0 0000 até 0x3 FFFF.

A partir do endereço, a função seleciona onde ler.

Funções

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| void | flash\_dados\_acid | (void) |
| void | flash\_op\_mostra | (void) |
| void | flash\_dump | (long adr, long qtd) |
| long | flash\_rd\_32b | (long adr) |
| void | flash\_wd\_32b | (long adr, long dado) |
| int | flash\_rd\_16b | (long adr) |
| void | flash\_wr\_16b | (long adr, int dado) |
| void | flash\_rd\_str | (long adr, byte \*msg, word qtd) |
| void | flash\_wr\_str | (long adr, byte \*msg) |
| void | flash\_apg\_blk | (long adr, byte qtd) |
| void | flash\_rd\_blk | (long adr, byte \*vet, word qtd) |
| void | flash\_wr\_blk | (long adr, byte \*vet, byte qtd) |
| byte | flash\_rd | (long adr) |
| void | flash\_wr | (long adr, byte dado) |
| byte | flash\_qual | (long adr) |
| void | flash\_espera | (long adr) |
|  |  |  |

* void **flash\_dados\_acid** (void)

Serve para a Flash Batida. Apresentar todos os dados do acidente. Inclui GPS e MPU.

* void **flash\_op\_mostra** (void)

Mostrar configuração feita ao ligar o carro.

* void **flash\_dump** (long adr, long qtd)

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

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

* long **flash\_rd\_32b** (long adr)

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

* void **flash\_wr\_32b** (long adr, long dado)

Escrever um valor de 32 bits na FLASH, Big Endian.

* int **flash\_rd\_16b** (long adr)

Ler um valor inteiro de 16 bits da FLASH. Big Endian.

* void **flash\_wr\_16b** (long adr, int dado)

Escrever um valor inteiro de 16 bits da FLASH. Big Endian.

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

Ler uma string da FLASH. O ponteiro msg deve ter espaço adequado.

Qtd indica a quantidade máxima, o último byte é o zero final

* void **flash\_wr\_str** (long adr, byte \*msg)

Gravar uma string na FLASH, terminada em zero.

* void **flash\_apg\_blk** (long adr, byte qtd)

(90) Preencher um determinado bloco com 0xFF.

qtd --> precisa estar dentro de página de 128 bytes.

Não é preciso apagar para gravar algo. Esta função é apenas como conveniência.

Usar a função *void flash\_espera(long adr)* para ser avisado assim que a operação terminar.

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

(80) Ler um bloco a partir de um endereço.

qtd --> não há limitação desde que fique dentro dos 16 bits (64 K).

* void **flash\_wr\_blk** (long adr, byte \*vet, byte qtd)

(70) Escrever um bloco a partir de um endereço.

qtd --> precisa estar dentro de página de 128 bytes (apenas os 7 LSbits de endereço variam).

Usar a função *void flash\_espera(long adr)* para ser avisado assim que a operação terminar.

* byte **flash\_rd** (long adr)

(50) Ler um endereço da memória.

* byte **flash\_wr** (long adr, byte dado)

(60) Escrever o dado no endereço adr da memória.

Usar a função *void flash\_espera(long adr)* para ser avisado assim que a operação terminar.

* byte **flash\_qual** (long adr)

Retornar endereço TWI (I2C) da flash a ser usada, em função do endereço a ser acessado.

#define FLASH1\_ADR 0x50 //FLASH1

#define FLASH2\_ADR 0x51 //FLASH2

* + 0 0000 --> 0 FFFF 🡪 retorna FLASH1\_ADR+0 (0x50)
  + 1 0000 --> 1 FFFF 🡪 retorna FLASH1\_ADR+4 (0x54)
  + 2 0000 --> 2 FFFF 🡪 retorna FLASH2\_ADR+0 (0x51)
  + 3 0000 --> 3 FFFF 🡪 retorna FLASH2\_ADR+5 (0x55)

Exemplo:

byte er,et;

er=flash\_qual(endereço);

er=er<<1; //Endereço escravo receptor

et=et+1; //Endereço escravo transmissor

* void **flash\_espera** (long adr)

(100) Esperar Flash terminar gravação no endereço adr.

Ficar acessando com WR até ela responder.

// Endereços da FLASH 24LC1025 (128 KB) TWI

#define FLASH1\_ADR 0x50 //FLASH1

#define FLASH2\_ADR 0x51 //FLASH2

#define FLASH1\_ADR\_B0 FLASH1\_ADR+0 //FLASH1, Bloco 0: 64KB (0x00000 -> 0x0FFFF)

#define FLASH1\_ADR\_B1 FLASH1\_ADR+4 //FLASH1, Bloco 1: 64KB (0x10000 -> 0x1FFFF)

#define FLASH2\_ADR\_B0 FLASH2\_ADR+0 //FLASH2, Bloco 0: 64KB (0x00000 -> 0x0FFFF)

#define FLASH2\_ADR\_B1 FLASH2\_ADR+4 //FLASH2, Bloco 1: 64KB (0x10000 -> 0x1FFFF)

#define FLASH\_PAG 128 //Tamanho da página para gravação

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