**Biblioteca MPU para a Caixa Preta**

Versão 2.0, 18/04/2020

Funções

|  |  |  |
| --- | --- | --- |
|  |  |  |
| void | mpu\_rd\_ac\_gi\_mg | (word \*vetor) |
| void | mpu\_rd\_mg | (word \*vetor) |
| void | mpu\_mag\_config | (void) |
|  |  |  |
| int | mpu\_rd\_bw | (void) |
| int | mpu\_rd\_freq | (void) |
| int | mpu\_rd\_esc\_giro | (void) |
| int | mpu\_rd\_esc\_acel | (void) |
| void | mpu\_int | (void) |
| int | mpu\_rd\_tp | (void) |
| void | mpu\_rd\_ac\_tp\_gi | (word \*vetor) |
| void | mpu\_rd\_ac\_gi | (word \*vetor) |
| void | mpu\_acorda | (void) |
| void | mpu\_dorme | (void) |
| byte | mpu\_whoami | (void) |
| void | mpu\_inicializa | (void) |
| void | mpu\_escalas | (byte gfs, byte afs) |
| void | void mpu\_sample\_rt | (byte sample\_rate) |
| byte | mpu\_self\_test | (int \*vt, byte prn) |
| void | mpu\_wr | (byte reg, byte dado) |
| byte | mpu\_rd | (byte reg) |
| void | mpu\_wr\_blk | (byte reg, byte \*dado, byte qtd) |
| void | mpu\_rd\_blk | (byte reg, byte \*dado, byte qtd) |
|  |  |  |
|  |  |  |
|  |  |  |

* void **mpu\_rd\_ac\_gi\_mg** (word \*vetor)

Ler Aceleração, Giro e Mag. Retorna vetor de 18 bytes. Não monta as palavras de 16 bits.

Retorna: [axh axl ayh ayl azh azl gxh gxl gyh gyl gzh gzl hxh hxl hyh hyl hzh hzl].

* void **mpu\_rd\_mg** (word \*vetor)

Ler Magnetômetro. Retorna vetor de 3 words [hx hy hz].

* void **mpu\_mag\_config** (void)

Inicializar Magnetômetro – Ainda não sabemos como fazer

* int **mpu\_rd\_bw** (void)

Ler no MPU a banda do filtro programada. Retorna 5, 10, 21, ..., 260 (Escala do Acelerômetro).

* int **mpu\_rd\_freq** (void)

Ler no MPU a freq de amostragem. Retorna 100, 200, ..., 1000. Considera Gyro Rate = 1.000 (DLPF=0).

* int **mpu\_rd\_esc\_acel** (void)

Ler no MPU a escala usada para o acelerômetro. Retorna 2, 4, 8 ou 16 g.

* **int mpu\_rd\_esc\_giro** (void)

Ler no MPU a escala usada para o giroscópio. Retorna 250, 500, 1000 ou 2000 graus/seg.

* void **mpu\_int** (void)

Preparar para MPU usar INT4

Pino PE4 entrada com pullup e habilitar INT4 para flanco de descida

MPU: interrupção em baixo com push-pull, pulso de 50 useg

MPU: Habilitar interrupção dado pronto

* void **mpu\_des\_int** (void)

Desabilitar o uso de interrupção pelo MPU.

Desabilitar INT4 e MPU: Desabilitar interrupção dado pronto.

* int **mpu\_rd\_tp** (void)

Retorna a leitura do registrador de temperatura.

* void **mpu\_rd\_ac\_tp\_gi** (word \*vetor)

Ler Aceleração, temperatura e giro. Retorna vetor de 7 words [ax ay az tp gx gy gz].

* void **mpu\_rd\_ac\_gi** (word \*vetor)

Ler Aceleração e giro. Retorna vetor de 6 words [ax ay az gx gy gz].

* void **mpu\_acorda** (void)

Acordar o MPU e programar para usar relógio Giro X.

* void **mpu\_dorme** (void)

Dormir o MPU e programar para usar relógio Giro X.

* byte **mpu\_whoami** (void)

Retornar a leitura do registrador WHO\_AM\_I.

* void **mpu\_config** (void)

Colocar o MPU num estado conhecido.

Taxa = 1 kHz, Banda: Acel=5 Hz e Giro=5 Hz e delay=19 mseg

Taxa de amostragem = taxa/(1+SMPLRT\_DIV) = 1k/10 = 100Hz

Escalas acel = +/2g e giro = +/-250 gr/s

* void **mpu\_escalas** (byte gfs, byte afs)

Selecionar Fundo de Escalas para o MPU.

Acel: 0=+/-2g, 1=+/-4g, 2=+/-8g, 3=+/-16g.

Gyro: 0=+/-250gr/s, 1=+/-500gr/s, 2=+/-1000gr/s, 3=+/-2000gr/s.

#define GIRO\_FS\_250 0 // +/- 250 graus/seg

#define GIRO\_FS\_500 1 // +/- 500 graus/seg

#define GIRO\_FS\_1000 2 // +/- 1000 graus/seg

#define GIRO\_FS\_2000 3 // +/- 2000 graus/seg

#define ACEL\_FS\_2G 0 // +/- 2g

#define ACEL\_FS\_4G 1 // +/- 4g

#define ACEL\_FS\_8G 2 // +/- 8g

#define ACEL\_FS\_16G 3 // +/- 16g

* void **mpu\_sample\_rt** (byte sample\_rate)

Selecionar Sample Rate, considerando Taxa = 1kHz (Registrador CONFIG)

// Valores para o Sample Rate, Registrador SMPLRT\_DIV

// Considerando Taxa = 1kHz (Registrador CONFIG)

#define SAMPLE\_RT\_1kHz 0 // 1.000/(0+1) = 1000

#define SAMPLE\_RT\_500Hz 1 // 1.000/(1+1) = 500

#define SAMPLE\_RT\_333Hz 2 // 1.000/(2+1) = 333,33

#define SAMPLE\_RT\_250Hz 3 // 1.000/(3+1) = 250

#define SAMPLE\_RT\_200Hz 4 // 1.000/(4+1) = 200

#define SAMPLE\_RT\_166Hz 5 // 1.000/(5+1) = 166,66

#define SAMPLE\_RT\_142Hz 6 // 1.000/(6+1) = 142,85

#define SAMPLE\_RT\_125Hz 7 // 1.000/(7+1) = 125

#define SAMPLE\_RT\_111Hz 8 //1.000 /(8+1) = 111,11

#define SAMPLE\_RT\_100Hz 9 //1.000 /(9+1) = 100

* byte **mpu\_self\_test** (int \*vt, byte prn)

MPU: Realizar Self-Test (ST) e retornar se passou (TRUE) ou falhou (FALSE).

Opção prn = TRUE, imprime resultados, prn=FALSE, nada imprime.

Vetor **int vt[24]** recebe resultados de passos intermediários

vt[ 0, ..., 5] 🡪 leitura dos eixos com selt test OFF

vt[ 6, ..., 11] 🡪 leitura dos eixos com selt test ON

vt[12, ..., 17] 🡪 leitura dos registradores de self test

vt[18, ..., 23] 🡪 resultado para comparar com tolerância de 14%

* void **mpu\_wr** (byte reg, byte dado)

(10) Escrever num registrador do MPU.

* byte **mpu\_rd** (byte reg)

(20) Ler um registrador do MPU.

* void **mpu\_wr\_blk** (byte reg, byte \*dado, byte qtd)

(30) Escrever um bloco de dados no MPU a partir de um registrador.

* void **mpu\_rd\_blk** (byte reg, byte \*dado, byte qtd)

(40) Ler um bloco do MPU a partir de um registrador.