Biblioteca Timers para o Arduino

Timer1 🡪 gerar interrupção de 50 Hz (20 mseg);

Timer2 🡪 nada

Timer3 🡪 nada

Timer4 🡪 nada

Timer5 🡪 Cronômetro;

**Cálculo para 50 Hz (interrupção a cada 20 mseg):**

CS12:0 = 011B 🡪 Configurar prescaler = 64

WGM13:0 = 0100B 🡪 Modo 4 CTC na coincidência com OCR1A;

50 Hz (20 mseg) 🡪

100 Hz (10 mseg) 🡪

Gabarito para configurar os registradores do TC1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **TCCR1A** | COM**1**A1 | COM**1**A0 | COM**1**B1 | COM**1**B0 | COM**1**C1 | COM**1**C0 | WGM**1**1 | WGM**1**0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **TCCR1B** | ICNC**1** | ICES**1** | - | WGM**1**3 | WGM**1**2 | CS**1**2 | CS**1**1 | CS**1**0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **TCCR1C** | FOC**1**A | FOC**1**B | FOC**1**C | - | - | - | - | - |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **TIMSK1** | - | - | ICIE**1** | - | OCIE**1**C | OCIE**1**B | OCIE**1**A | TOIE**1** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **TIRF1** | - | - | ICF**1** | - | OCF**1**C | OCF**1**B | OCF**1**A | TOV**1** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Funções para Serial 0

|  |  |  |
| --- | --- | --- |
|  |  |  |
| void | lcd\_float | (float fx, byte prec) |
| void | lcd\_dec32 | (long dt) |
| void | lcd\_dec32u | (long dt) |
| void | lcd\_dec32nz | (long dt) |
| void | lcd\_dec32unz | (long dt) |
| void | lcd\_hex32 | (long dt) |
|  |  |  |
| void | lcd\_dec16 | (int dt) |
| void | lcd\_dec16u | (word dt) |
| void | lcd\_dec16nz | (int dt) |
| void | lcd\_dec16unz | (word dt) |
| void | lcd\_hex16 | (word dt) |
|  |  |  |
| void | lcd\_dec8 | (byte dt) |
| void | lcd\_dec8u | (byte dt) |
| void | lcd\_dec8nz | (byte dt) |
| void | lcd\_dec8unz | (byte dt) |
| void | lcd\_hex8 | (byte dt) |
|  |  |  |
| void | ser\_crlf | (byte qtd) |
| void | ser\_cr | (byte qtd) |
| void | ser\_lf | (byte qtd) |
| void | ser\_spc | (byte qtd) |
| void | ser\_str | (byte \*msg) |
| void | ser\_char | (byte dt) |
| void | ser\_tx\_fila\_config | (void) |
| char | ser\_tx\_poe | (char dt) |
| char | ser\_tx\_tira | (char \*dt) |
| void | ser\_config | (long br) |
|  | ISR(USART0\_TX\_vect) |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* void **ser\_crlf** (byte qtd)

Colocar na fila qtd de pares CR (0xD = ‘\r’) e LF (0xA = ‘\n’.

* void **ser\_cr** (byte qtd)

Colocar na fila qtd Carriage Return (CR = 0xD = ‘\r’).

* void **ser\_lf** (byte qtd)

Colocar na fila qtd Line Feed (LF = 0xA = ‘\n’).

* void **ser\_spc** (byte qtd)

Colocar na fila qtd brancos (0x20).

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

Colocar a string apontada por msg no buffer serial.

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

Colocar um char no buffer serial. Gera alerta no LCD se fila serial encher.

* void **ser\_tx\_fila\_config** (void)

Inicializar fila de transmissão serial.

* char **ser\_tx\_poe** (char cha)

Colocar um byte na fila de transmissão. Retorna TRUE se conseguiu. Retorna FALSE se fila cheia.

* char **ser\_tx\_tira** (char \*cha)

Coloca no ponteiro \*char o byte retirado da fila de transmissão e retorna TRUE.

Se retornar FALSE é porque a fila está vazia.

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

Configurar porta serial com o baud rate = BR. Habilitar interrupção por transmissão.

* **ISR (USART0\_TX\_vect)**

ISR: Interrupção por dado enviado, faz ser\_tx\_ok=TRUE para indicar que transmitiu um dado.