**Biblioteca Timers para a Caixa Preta**

Versão 1.0, 12/04/2020

Funções

|  |  |  |
| --- | --- | --- |
|  |  |  |
| void | timer1\_config | void |
| void | timer2\_config | void |
| byte | lcd\_linha\_alterou | (byte qual) |
|  | ISR(TIMER1\_COMPA\_vect) |  |
|  | ISR(TIMER2\_COMPA\_vect) |  |

* void **timer1\_config** (void)

Configurar Timer 1 para FREQ\_T1 (ver define). Coloca no Modo CTC com limite indicado por OCR1A.

* void **timer2\_config** (void)

Configurar Timer 2 para FREQ\_T2 (ver define). Coloca no Modo CTC com limite indicado por OCR2A.

* byte **lcd\_linha\_alterou** (byte qual)

A variável **lcd\_mudou** indica nos seus bits quais linhas foram alteradas. Esta função consulta o bit indicado pela variável **qual** e se for 1, ela atualiza **lcd\_lin** com o endereço do buffer da linha a ser atualizada, apaga o bit na variável **lcd\_mudou** e retorna TRUE.

* **ISR** (**TIMER1\_COMPA\_vect**)

Atender à interrupção do Timer 1: FREQ\_T1 (100 Hz)

* **ISR** (**TIMER2\_COMPA\_vect**){

Atender à interrupção do Timer 2: FREQ\_T2 (1000 Hz)

O Arduino Mega tem 5 timers, usados da forma listada abaixo,

Timer1 🡪 gerar interrupção a cada 100 Hz (10 mseg);

Timer2 🡪 Atualizar linhas do LCD (1 kHz);

Timer3 🡪 nada

Timer4 🡪 nada

Timer5 🡪 Cronômetro;

**------------------- TIMER 1 -----------------------------------**

Timer 1 interrompe com taxa de 100 Hz (10 mseg).

**Timer 1: Provocar interrupção numa dada freq (fOC1A)**

CS12:0 = 011B 🡪 Configurar prescaler = 64 🡪 16 MHz / 64 = 250 kHz.

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

FREQ\_T1🡪 (conta para facilitar ao programa calcular OCR1A)

#define FREQ\_T1 100 //Freq de interrupção do timer 1

...

OCR1A = (25000/(FREQ\_T1/10))-1;

Se for 50 Hz (20 mseg) 🡪

Se for 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 |

**----------------- Sobre o ADC --------------------**

*Gabaritos para configurar ADC (8 bits alinhado pela esquerda, ler apenas ADCH)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **ADMUX** | REFS1 | REFS0 | ADLAR | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| 0,1,1 | 1,1,1 | 1 | 0 | 0 | 0 | 0,0,1 | 0,1,0 |
| **ADCSRA** | ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| **ADCSRB** | - | ACME | - | - | MUX5 | ADST2 | ADST1 | ADST0 |
| - | 0 | - | - | 0 | 1 | 0 | 1 |

Seleção do relógio do ADC 🡪 ADPS2:0 = 3 🡪 16 MHz/8 = 2 MHz

*Tabela 13.5: Tempos de conversão do ADC (em ciclos do ADCCLK)*

|  |  |
| --- | --- |
| **Condição** | **Tempo de conversão (em ciclos)** |
| Primeira conversão | 25 |
| Conversão única, canal simples | 13 |

Ao mudar de canal, conversão consome 25 ciclos 🡪 12,5 µs (80 kHz) 🡪 200 instruções do AVR.

Conversão com o mesmo canal 13 ciclos 🡪 6,5 µs (154 kHz) 🡪 104 instruções do AVR.

Não precisamos de taxas de conversão tão altas e não queremos gastar tempo esperando o ADC converter. A solução é usar o Timer 1 para disparar o ADC numa interrupção e ler o resultado na interrupção seguinte. Porém, precisamos ler 3 entradas analógicas diferentes:

* Canal 0: ler o teclado,
* Canal 1: ler a tensão do carro (12 V) e
* Canal 2: ler a tensão do supercapacitor.

A solução foi usando um contador **timer1\_cont**, contando de 0 até 31. Ele é incrementado a cada interrupção, assim, o Timer 1 sabe o que deve fazer a cada instante. Toda leitura do ADC é fruto da média de duas conversões espaçadas de 10 ms.

* Atualiza tensão do carro (**vcar\_val**), na taxa de 100/32 Hz (~3Hz).
* Atualiza tensão do super cap (**vcap\_val**), na taxa de 100/32 Hz (~3Hz).
* Leitura das teclas (**sw\_val**) na taxa de 100\*(12/32) = 37,5 Hz, uma leitura a cada 27 ms.

Funcionou bem: **timer1\_cont** = 0, 1, ..., 31, 0, 1, ..., interrupção em 100 Hz

|  |  |  |  |
| --- | --- | --- | --- |
| 0-ADC Start | 8) +Ler, ADC start\* | 16-ADC Start | 24) +Ler, ADC start\* |
| 1) Ler, ADC start | 9) Ler, ADC start | 17) Ler, ADC start | 25) Ler, ADC start |
| 2) +Ler, ADC start\* | 10) +Ler, ADC start\* | 18) +Ler, ADC start\* | 26) +Ler, ADC start\* |
| 3) Ler, ADC start | 11) Ler, ADC start | 19) Ler, ADC start | 27) Ler, ADC start |
| 4) +Ler, ADC start\* | 12) +Ler, Canal1(VCAR)\* | 20) +Ler, ADC start\* | 28) +Ler, Canal 2(VCAP)\* |
| 5) Ler, ADC start | 13) ADC Start | 21) Ler, ADC start | 29) ADC Start |
| 6) +Ler, ADC start\* | 14) Ler, ADC start | 22) +Ler, ADC start\* | 30) Ler, ADC start |
| 7) Ler, ADC start | 15) +Ler, Canal 0 | 23) Ler, ADC start | 31) +Ler, Canal 0 |

\* indica a fase para tirar a média na leitura do teclado.

**sw\_val** = última leitura do teclado

**vcar\_val** = última leitura da tensão (12 V) gerada pelo carro e

**vcap\_val** = última leitura da tensão sobre o supercapacitor.

**------------------- TIMER 2 -----------------------------------**

É responsável por atualizar o LCD. E envia um nibble para o LCD a cada interrupção. No teste com LCD funcionou com frequência de interrupção de até 25.000 Hz. Acima disso começou a dar problemas. São 42 interrupções para cada linha

|  |  |  |
| --- | --- | --- |
| Freq Timer 2 | Atualizar uma linha (42 int) | Instr do AVR |
| 1.000 Hz | 42 ms | 16.000 |
| 5.000 Hz | 8,4 ms | 3.200 |
| 10.000 Hz | 4,2 ms | 1.600 |

Por enquanto, está sendo usado 1.000 Hz.

Podemos tentar com a freq. = 5.000 Hz. Isto significa que se consegue atualizar uma linha do LCD entre duas interrupções do Timer 1, que é de 100 Hz (10 ms). Entretanto, gera uma grande quantidade de interrupções próximas. Pesar esse ponto em consideração.

**Timer 2: Provocar interrupção numa dada freq (fOC2A)**

CS22:0 = 100B 🡪 Configurar prescaler = 64 🡪 16 MHz / 64 = 250 kHz.

WGM12:0 = 010B 🡪 Modo 2 CTC na coincidência com OCR1A;

Se for 1000 Hz (10 mseg) 🡪

Se for 10.000 Hz (10 mseg) 🡪

Se for 20.000 Hz (10 mseg) 🡪

#define FREQ\_T2 1000 //Freq de interrupção do timer 2

...

OCR2A = (25000/(FREQ\_T2/10))-1;

Gabarito para configurar os registradores do TC2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **TCCR2A** | COM2A1 | COM2A0 | COM2B1 | COM2B0 | - | - | WGM21 | WGM20 |
| 0 | 0 | 0 | 0 |  |  | 1 | 0 |
| **TCCR2B** | FOC2A | FOC2B | - | - | WGM22 | CS22 | CS21 | CS20 |
| 0 | 0 |  |  | 0 | 1 | 0 | 0 |
| **TIMSK2** | - | - | - | - | - | OCIE2B | OCIE2A | TOIE2 |
|  |  |  |  |  | 0 | 1/0 | 0 |
| **TIFR2** | - | - | - | - | - | OCF2B | OCF2A | TOV2 |
|  |  |  |  |  | 0 | 0 | 0 |
| **ASSR** | - | EXCLK | AS2 | TCN2UB | OCR2AUB | OCR2BUB | TCR2AUB | TCR2BUB |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **GTCCR** | TMS | - | - | - | - | - | PSRASY | PSRSYNC |
| 0 |  |  |  |  |  | 0 | 0 |