



Universidade Estadual de Campinas  
Faculdade de Engenharia Elétrica  
e de Computação



---

# EA871 - Laboratório de Programação Básica de Sistemas Digitais

Roteiro 13: TPM – Input Capture e Output Compare

---

Autores: Fernando Teodoro de Cillo e Rafael Silva Cirino

RA:

197029

223730

Campinas  
Julho de 2022

## Introdução:

O intuito deste experimento é realizar o controle preciso dos instantes de captura dos sinais de entrada (*Input Capture*) e de mudança dos estados dos sinais de saída (*Output Compare*) via módulos TPMx.

## Experimento:

1

**1.a** O período registrado foi de 0.3994s, como mostra a figura 1. Este valor está muito próximo do período de 0.3999s, esperado pela configuração, mostrada na figura 2. O cálculo do período esperado é mostrado na equação (1).

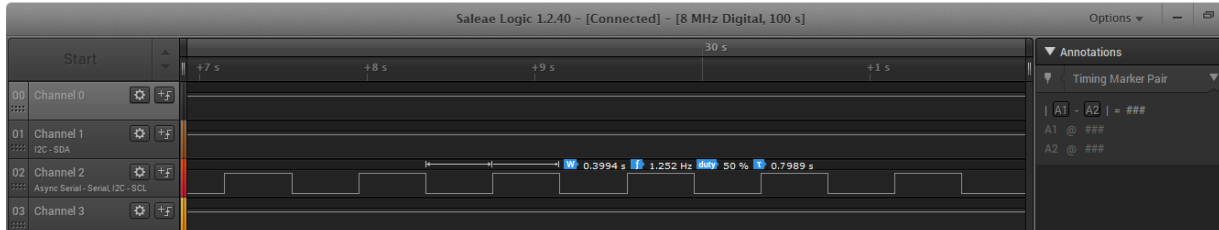


Figura 1: Analisador lógico utilizado para registrar o período

```
// Inicializar o par (botoeira NMI, canal TPM0CH2) como ICOC
TPM_initSwitchNMIChannelTPM0CH2(65535, 0b111); // (65535*128)/20971520 = 0.3999
```

Figura 2: Configuração do TPM

$$T = \frac{65535 \cdot 128}{20971520} \approx 0.3999s \quad (1)$$

**1.b** Considerando os 3 períodos de onda no canal 1 temos  $t = 3 \approx 1.19s$ . No pino 4 é amostrado o valor 1.43s que, ao subtrair 1.19, retorna 0.23, que não corresponde ao esperado. Esperava-se que a diferença fosse um período  $T = 0.3994s$ , que seria o tempo necessário para a saída responder ao acionamento da botoeira.

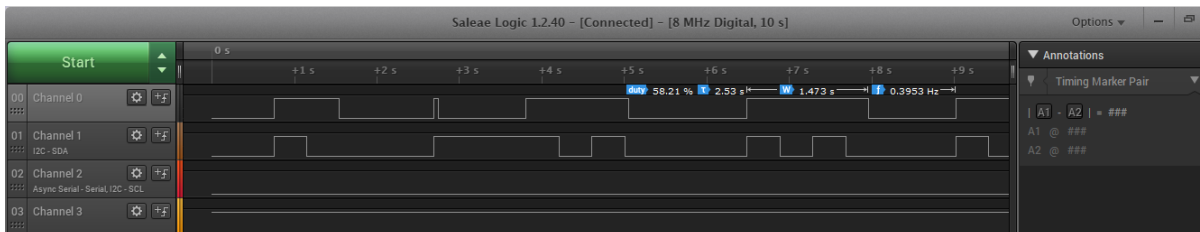


Figura 3: Analisador lógico no pino 4 do header H5

**1.c** A instrução `TPM_CnSC_REG(TPM[x], b) |=(mode<<2)` obedece a tabela 4, retirada do manual. Segundo a tabela, na função `TPM_initChIC` a instrução limpa o estado desse bit, e em `TPM_initChOC` ativa para funcionar quando houver alteração no contador LPTPM.

| TPMx_CnSC field descriptions |   |
|------------------------------|---|
| Field                        | Description   |
| 31-8<br>Reserved             | This field is reserved.<br>This read-only field is reserved and always has the value 0.   |
| 7<br>CHF                     | Channel Flag<br>Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect.<br>If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF.<br>0 No channel event has occurred.<br>1 A channel event has occurred. |
| 6<br>CHIE                    | Channel Interrupt Enable<br>Enables channel interrupts.<br>0 Disable channel interrupts.<br>1 Enable channel interrupts.  |
| 5<br>MSB                     | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.   |
| 4<br>MSA                     | Channel Mode Select<br>Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.   |
| 3<br>ELSB                    | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.  |
| 2<br>ELSA                    | Edge or Level Select<br>The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.  |
| 1<br>Reserved                | This field is reserved.<br>This read-only field is reserved and always has the value 0.   |
| 0<br>DMA                     | DMA Enable<br>Enables DMA transfers for the channel.<br>0 Disable DMA transfers.<br>1 Enable DMA transfers.   |

Figura 4: Tabela de descrição da instrução TPM\_CnSC

6 A função `ComputaIntervaloTempo` segue a equação (2), como mostra o código da figura 5.

$$\text{ciclos} = \begin{cases} M + \frac{(MOD - T1) + T2}{MOD}, & \text{se } T1 \geq T2 \\ M + \frac{(T2 - T1)}{MOD}, & \text{se } T1 < T2 \end{cases} \quad (2)$$

```

void ComputaIntervaloTempo (unsigned short T1, unsigned short T2, unsigned int counter, float *tempo){
    int mod;
    float ciclos;

    mod = TPM0_MOD;

    if (T1 >= T2){
        ciclos = counter + ((mod - T1) + T2)/mod;
    } else {
        ciclos = counter + (T2 - T1)/mod;
    }

    if (T1 > T2){
        *tempo = (((mod + T2 - T1)/mod) + (ciclos - 1)) * 0.2;
    } else {
        *tempo = (((T2 - T1)/mod) + ciclos) * 0.2;
    }
}

```

Figura 5: Função `ComputaIntervaloTempo`

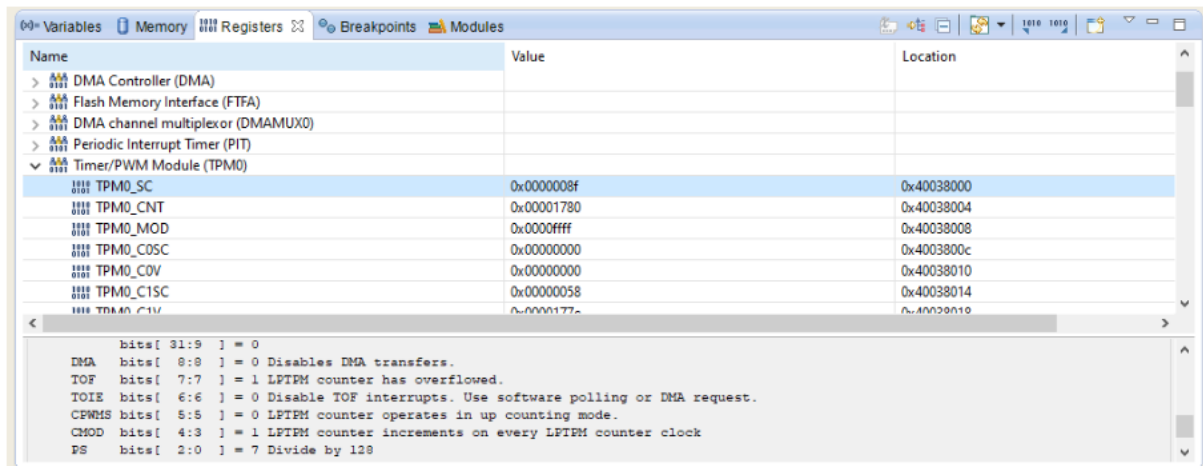


Figura 6: Teste de unidade da função ComputaIntervaloTempo, com CPWMS=0

7 O teste do *led* azul é mostrado no seguinte *gif*: <https://drive.google.com/file/d/11tzWuW1JijiYvbahddVB8hamiF3Wt-rU/view?usp=sharing>

## 10

### INICIO

#### MAIN:

- inicializa LCD e TPM
- endereço do bitmap do cronometro
- cria variável que armazena o estado do sistema
- lê o estado

- inicializa a variável string[8] (char)
- inicializa a variável valor[3] (char curto sem sinal)
- inicializa a variável seg (inteiro)
- inicializa a variável tempo (ponto flutuante)

#### laço:

- lê estado
- escolha (estado)
  - caso ESPERA
  - caso CONTAGEM
  - caso LEITURA
    - passa
    - caso REALIMENTACAO
      - lê segundos
      - mostra segundos no visor do LCD
      - muda para o estado CONTAGEM

caso INTERVALO

lê os valores capturados

computa o tempo total com as eqs. (1) e (2)

mostra o resultado e o bitmap no LCD

muda para o estado LEITURA

caso RESETA\_VISOR

apaga o valor da tensão no LCD

apaga o bitmap

muda para o estado ESPERA

fimescolha

FIM

12

A máquina de estados segue a sequência da figura 7, que mostra a transição entre os estados.

| Name         | Value      | Location   |
|--------------|------------|------------|
| write_bitmap | 0x20002fec | 0x20002fec |
| estado       | ESPERA     | 0x20002fef |
| string       | 0x20002fe4 | 0x20002fe4 |
| valor        | 0x20002fdc | 0x20002fdc |
| seg          | 4          | 0x20002fd8 |
| tempo        | 4.0        | 0x20002fd4 |

(a) estado ESPERA

| Name   | Value    | Location   |
|--------|----------|------------|
| novo   | CONTAGEM | 0x20002f8f |
| estado | 0        | 0x1ffff010 |
| estado | 0        | 0x1ffff010 |
| estado | 0        | 0x1ffff010 |
| estado | 0        | 0x1ffff010 |
| estado | 0        | 0x1ffff010 |

(b) estado CONTAGEM

| Name   | Value         | Location   |
|--------|---------------|------------|
| novo   | REALIMENTACAO | 0x20002f8f |
| estado | 1             | 0x1ffff010 |
| estado | 1             | 0x1ffff010 |
| estado | 1             | 0x1ffff010 |
| estado | 1             | 0x1ffff010 |
| estado | 1             | 0x1ffff010 |

(c) estado REALIMENTACAO

| Name   | Value     | Location   |
|--------|-----------|------------|
| novo   | INTERVALO | 0x20002f8f |
| estado | 1         | 0x1ffff010 |
| estado | 1         | 0x1ffff010 |
| estado | 1         | 0x1ffff010 |
| estado | 1         | 0x1ffff010 |
| estado | 1         | 0x1ffff010 |

(d) estado INTERVALO

| Name   | Value   | Location   |
|--------|---------|------------|
| novo   | LEITURA | 0x20002fc7 |
| estado | 3       | 0x1ffff010 |
| estado | 3       | 0x1ffff010 |
| estado | 3       | 0x1ffff010 |
| estado | 3       | 0x1ffff010 |
| estado | 3       | 0x1ffff010 |

(e) estado LEITURA

| Name   | Value        | Location   |
|--------|--------------|------------|
| novo   | RESETA_VISOR | 0x20002f8f |
| estado | 4            | 0x1ffff010 |
| estado | 4            | 0x1ffff010 |
| estado | 4            | 0x1ffff010 |
| estado | 4            | 0x1ffff010 |
| estado | 4            | 0x1ffff010 |

(f) estado RESETA\_VISOR

| Name   | Value  | Location   |
|--------|--------|------------|
| novo   | ESPERA | 0x20002fc7 |
| estado | 5      | 0x1ffff010 |
| estado | 5      | 0x1ffff010 |
| estado | 5      | 0x1ffff010 |
| estado | 5      | 0x1ffff010 |
| estado | 5      | 0x1ffff010 |

(g) estado ESPERA

Figura 7: Máquina de estados

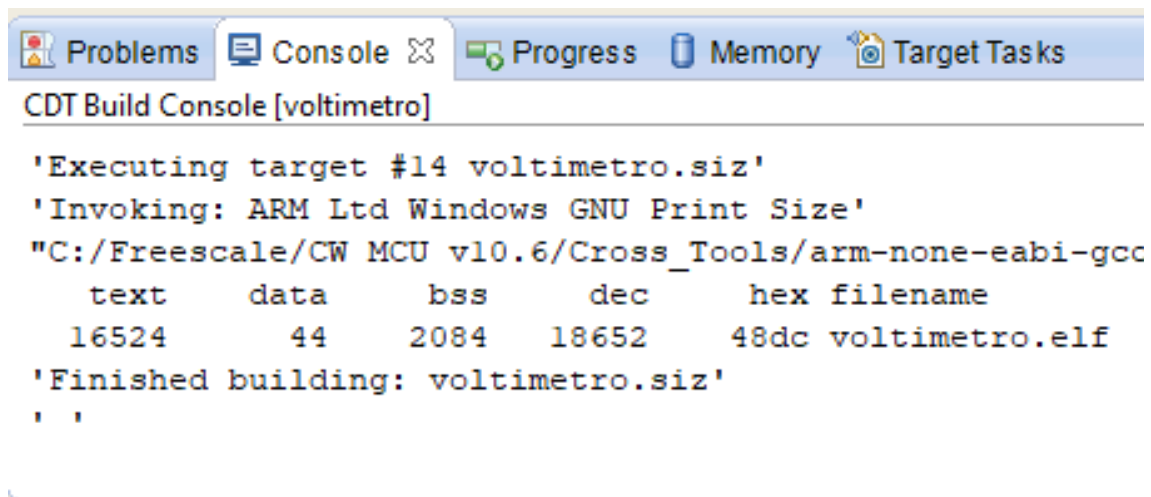
## 14

Tabela 1: Testes funcionais dos tempos medidos usando o cronômetro

|   | Placa    |      | Tempo medido | Erro   |
|---|----------|------|--------------|--------|
|   | segundos | TPM0 |              |        |
| 1 | 4        | 4.2  | 4.24         | -0.94% |
| 2 | 8        | 8    | 7.97         | 0.38%  |
| 3 | 3        | 3.2  | 3.25         | -1.54% |
| 4 | 14       | 14   | 14.03        | 0.21%  |
| 5 | 26       | 26.6 | 25.57        | 0.11%  |
| 6 | 140      | 140  | 139          | 0.72%  |
|   |          |      | média        | -0.25% |

## 15

Na figura 9 temos o *print size* obtido no experimento 13, comparando com a figura 8, referente ao experimento 12, podemos perceber que há uma diminuição em dois valores: *.txt*, que corresponde ao tamanho das instruções e dados constantes armazenados na memória flash, e *.data*, que se refere a valores com inicialização armazenada na memória RAM. O tamanho de *.bss*, variáveis sem inicialização armazenadas na memória RAM, aumentou. Essas mudanças são esperadas, porque o projeto *calculadora* exigia uma quantidade de variáveis e de funções executáveis muito grande.



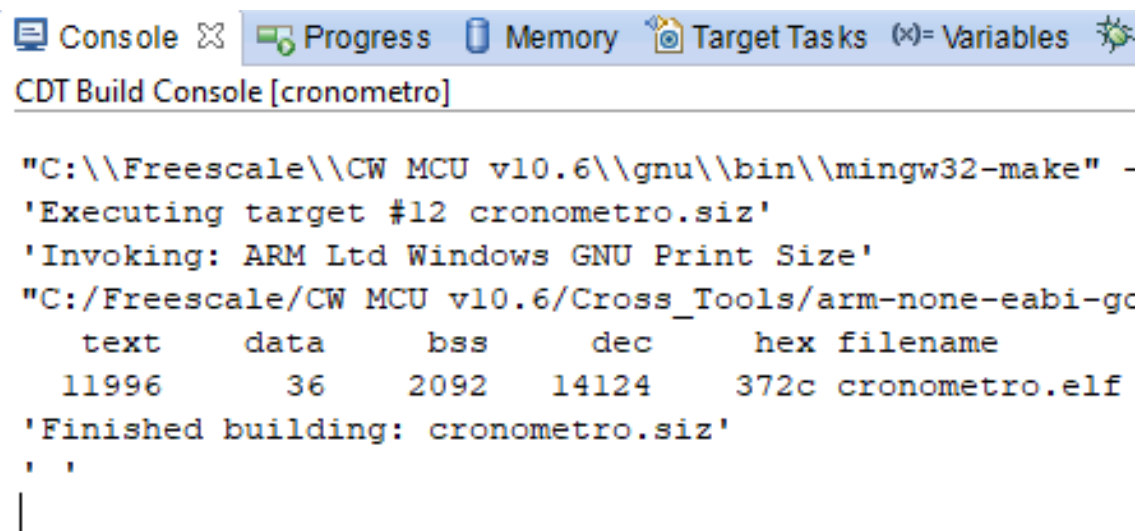
```

CDT Build Console [voltimetro]

'Executing target #14 voltimetro.siz'
'Invoking: ARM Ltd Windows GNU Print Size'
"C:/Freescale/CW MCU v10.6/Cross_Tools/arm-none-eabi-gcc
    text    data    bss    dec    hex filename
  16524     44   2084  18652  48dc voltimetro.elf
'Finished building: voltimetro.siz'
, ,

```

Figura 8: *Print size* do projeto *voltimetro*



```
"C:\\Freescale\\CW MCU v10.6\\gnu\\bin\\mingw32-make" -  
'Executing target #12 cronometro.siz'  
'Invoking: ARM Ltd Windows GNU Print Size'  
"C:/Freescale/CW MCU v10.6/Cross_Tools/arm-none-eabi-gc  
  text    data    bss     dec     hex filename  
 11996     36    2092   14124   372c cronometro.elf  
'Finished building: cronometro.siz'  
, ,  
|
```

Figura 9: *Print size* do projeto cronometro