Módulo de Temporização/PWM

Wu Shin-Ting, 1o. Semestre de 2012

Um módulo TPM *x* é um sistema de tempo com 3, mas poderiam ser até 8, canais. As funções de temporização são baseadas em um contador de 16 *bits* com um pré-escalador de 1 a 128, conforme mostra Figura 1.

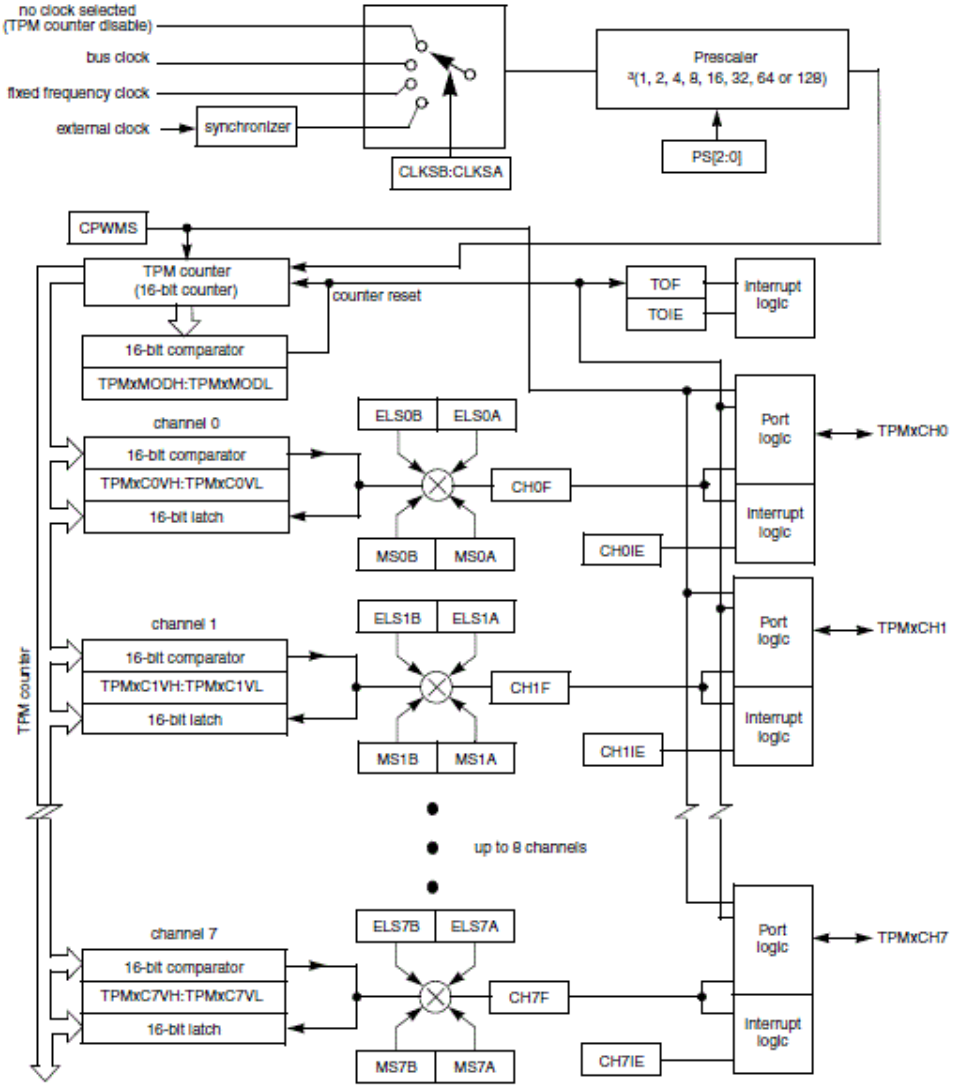


Figura 1: Diagrama de bloco de um módulo TPM.

Cada canal *n* suporta quatro modos de operação:

1. **captura de entrada:** quando detectada uma transição na entrada, o módulo copia o valor do registrador de contagem de tempo (*TPM counter*) para o registrador de valor do canal (TPMxCnV) .
2. **saída por comparação:** quando o valor do registrador de contagem de tempo (*TPM counter*) fique igual ao valor do registrador de valor do canal (TPMxCnV), o módulo pode setar, limpar, inverter ou ignorar o valor no pino correspondente.
3. **PWM alinhado a borda:** o valor do registrador de módulo (TPMxMOD) incrementado de 1 e o valor do registrador de valor do canal (TPMxCnV) correspondem,

respectivamente, ao período e ao ciclo de trabalho (largura do pulso) do sinal de saída PWM.

4. **PWM alinhado ao centro:** o período e a metade da largura do pulso do sinal de saída PWM correspondem, respectivamente, a 2 vezes o valor do registrador de módulo (TPMxMOD) e ao valor do registrador de valor do canal (TPMxCnV). O contador de tempo incrementa até atingir o valor contido no registrador de módulo para então decrementar até voltar para zero.

Os modos de operação dos canais podem ser distintos. Há um *bit* de controle que permite configurar, simultaneamente, todos os canais de um módulo em modo de operação PWM alinhado ao centro.

Há um registrador de estado e de controle para cada módulo *x*, TPMxSC. Conforme mostra Figura 2, controla-se através dos *bits* 4 e 3 deste registrador (CLKSB:CLKSA) a fonte de relógio (base de tempo), e dos *bits* 2, 1 e 0 (PS[2:0]) o fator de pré-escala. O *bit* TOF é sempre setado quando a contagem atinge o valor do registrador de módulo (TPMxMOD). E se o *bit* TOIE for também setado, gera-se uma interrupção.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 19-6. TPM Status and Control Register (TPMxSC)

Table 19-2. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is completed, the sequence is reset so TOF remains set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow. 1 TPM counter has overflowed.
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling). 1 TOF interrupts enabled.
5 CPWMS	Center-aligned PWM select. This read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.
4–3 CLKS[B:A]	Clock source selection bits. As shown in Table 19-3, this 2-bit field is used to disable the TPM counter or select one of three clock sources to TPM counter and counter prescaler.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of eight division factors for the TPM clock as shown in Table 19-4. This prescaler is located after any clock synchronization or clock selection so it affects the clock selected to drive the TPM counter. The new prescale factor affects the selected clock on the next bus clock cycle after the new value is updated into the register bits.

Figura 2: TPMxSC

Cada canal n tem registrador de estado e de controle próprio, TPMxCnSC, para controlar o seu modo de operação, conforme detalha a Figura 3.

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0							
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 19-11. TPM Channel n Status and Control Register (TPMxCnSC)

Table 19-5. TPMxCnSC Field Descriptions

Field	Description
7 CHnF	Channel n flag. When channel n is an input capture channel, this read/write bit is set when an active edge occurs on the channel n input. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF is not set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when this bit is set and channel n interrupt is enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while this bit is set and then writing a logic 0 to it. If another interrupt request occurs before the clearing sequence is completed CHnF remains set. This is done so a CHnF interrupt request is not lost due to clearing a previous CHnF. Reset clears this bit. Writing a logic 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n. 1 Input capture or output compare event on channel n.
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears this bit. 0 Channel n interrupt requests disabled (use for software polling). 1 Channel n interrupt requests enabled.
5 MSnB	Mode select B for TPM channel n. When CPWMS is cleared, setting the MSnB bit configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 19-6.
4 MSnA	Mode select A for TPM channel n. When CPWMS and MSnB are cleared, the MSnA bit configures TPM channel n for input capture mode or output compare mode. Refer to Table 19-6 for a summary of channel mode and setup controls. **Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 19-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that is driven in response to an output compare match, or select the polarity of the PWM output. If ELSnB and ELSnA bits are cleared, the channel pin is not controlled by TPM. This configuration can be used by software compare only, because it does not require the use of a pin for the channel.

Figura 3: TPMxCnSC.

Com estes elementos é possível gerar pulsos periódicos com o módulo TPM operando no modo **saída por comparação**. De fato, *Processor Expert* classifica todos os três canais do módulo TPM1 (TPM10:TPM11:TPM12) e do módulo TPM2 (TPM20:TPM21:TPM22) como fontes de interrupções periódicas.

Se escolhessemos TPM22 como fonte de interrupções periódicas a cada 125ms, o *Processor Expert* geraria um código equivalente ao seguinte trecho de código:

```
void TI1_Init(void)
{
  /* TPM2SC: TOF=0,TOIE=0,CPWMS=0,CLKSB=0,CLKSA=0,PS2=0,PS1=0,PS0=0 */
  TPM2SC = (byte)(0x00U);            /* Stop HW; disable overflow interrupt and set prescaler to 0 */
  /* TPM2C2SC: CH2F=0,CH2IE=1,MS2B=0,MS2A=1,ELS2B=0,ELS2A=0,??=0,??=0 */
  TPM2C2SC = (byte) (0x50U);         /* Set output compare mode and enable compare interrupt */
  TPM2MOD = (TPM2C2V = (word)(0xFFFEU));    /* Initialize appropriate value to the compare/modulo/reload register */
  /* TPM2CNTH: BIT15=0,BIT14=0,BIT13=0,BIT12=0,BIT11=0,BIT10=0,BIT9=0,BIT8=0 */
  TPM2CNTH=(byte)(0x00U);          /* Reset HW Counter */
  /* TPM2SC: TOF=0,TOIE=0,CPWMS=0,CLKSB=0,CLKSA=1,PS2=0,PS1=1,PS0=1 */
  TPM2SC=(byte)(0x0BU);            /* Set prescaler and run counter */
}
```

Pelo valor atribuído ao registrador TPM2SC na última instrução, podemos inferir que a base de tempo do barramento, *bus clock*, é 64Hz, que dividido por 8 resulta em um período de 125ms. Observe ainda que o *bit* CH2IE foi setado. Sendo o modo de operação **saída por comparação**, uma interrupção será gerada quando o valor do registrador de contagem de tempo (*TPM counter*) for igual ao valor do registrador de valor do canal (TPM2C2V) e a rotina de tratamento de interrupção **TI1_OnInterrupt**(**void**), no arquivo events.c e gerada pelo *Processor Expert*, será invocada.

Se escolhessemos TPM10 como fonte de interrupções periódicas a cada 16s, o *Processor Expert* geraria um código equivalente ao seguinte trecho de código:

```
void TI1_Init(void)
{
  /* TPM1SC: TOF=0,TOIE=0,CPWMS=0,CLKSB=0,CLKSA=0,PS2=0,PS1=0,PS0=0 */
  TPM1SC=(byte) (0x00U);           /* Stop HW; disable overflow interrupt and set prescaler to 0 */
  /* TPM1C0SC: CH0F=0,CH0IE=1,MS0B=0,MS0A=1,ELS0B=0,ELS0A=0,??=0,??=0 */
  TPM1C0SC=(byte)(0x50U);          /* Set output compare mode and enable compare interrupt */
 TPM2MOD = (TPM2C2V = (word)(0xFFFEU));       /* Initialize appropriate value to the compare/modulo/reload register */
  /* TPM1CNTH: BIT15=0,BIT14=0,BIT13=0,BIT12=0,BIT11=0,BIT10=0,BIT9=0,BIT8=0 */
  TPM1CNTH=(byte)(0x00U);          /* Reset HW Counter */
  /* TPM1SC: TOF=0,TOIE=0,CPWMS=0,CLKSB=1,CLKSA=0,PS2=0,PS1=1,PS0=0 */
  TPM1SC=(byte)(0x12U);           /* Set prescaler and run counter */
}
```

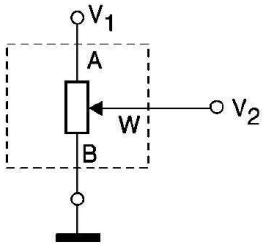
Neste segundo exemplo, o *Processor Expert* selecionou como fonte de base de tempo um relógio de frequência fixa e utilizou como fator de pré-escala 4 para gerar interrupções periódicas desejadas.

Observe que o *Processor Expert* manteve o valor do registrador TPMxMOD em ambos os exemplos e alterou a frequência das interrupções através da base de tempo e do fator de pré-escala. Figura 1 nos mostra que estas duas variáveis são globais, afetando as frequências de todos os canais do módulo TPMx. O que aconteceria com a frequencia de interrupções periódicas se alterássemos o valor do registrador para metade (0x7FFE) ou para um quarto deste valor (0x3FFE)?

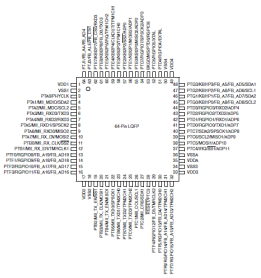
Roteiro para os itens 2 e 3 do Experimento 3: monitorar o valor de tensão do potenciômetro

(1) Projeto de *Hardware:* como ligar o potenciômetro, esquematizado na Figura 4.(a), ao microcontrolador, cuja pinagem é apresentada na Figura 4.(b). Pela explicação dada na aula, utilizaremos o canal ADCH 10. Este canal é acessível através do pino 38. Onde vocês ligariam V1 e V2 do potenciômetro para que a entrada do pino 38 não fique flutuando e que tenha uma variação de tensão entre 3.3V a 0V?

Para o item 3 do experimento precisa-se ligar ainda um *led*. Felizmente, no Experimento 2 foi ligado um no *bit* 0 da porta E (pino 55).



(a) Potenciômetro



(b) MCF51CN128 64 pinos

Figura 4: *Hardware*

(2) Projeto de *Software:* O fluxo de controle é orientado a dois tipos de interrupções: (a) interrupções periódicas de um sistema de tempo para amostrar o valor (analógico) do potenciômetro, e (b) interrupções quando completa a conversão de um valor analógico em um valor digital.

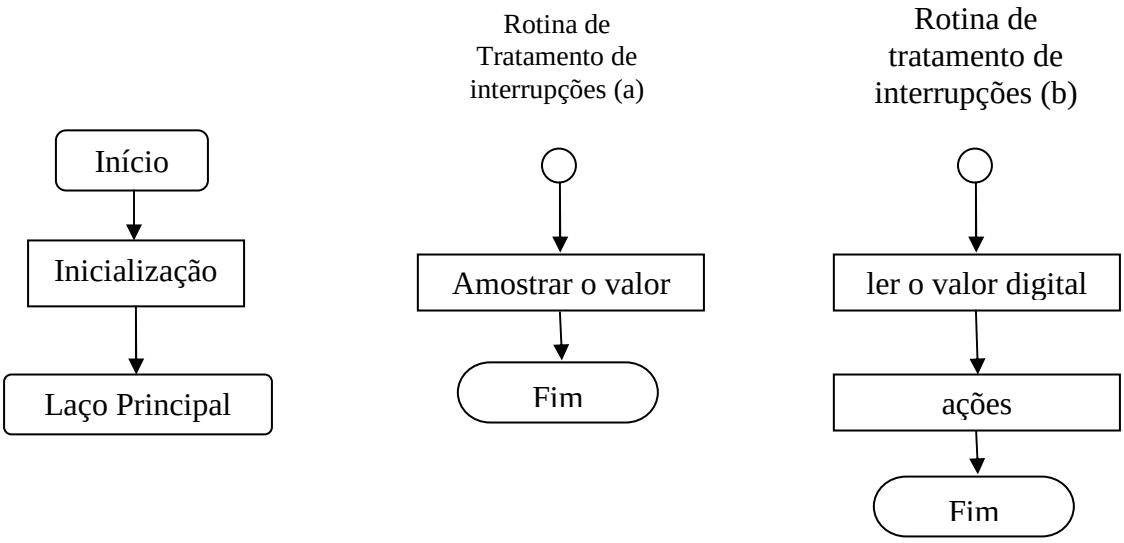


Figura 5: Fluxo de Controle por *software*.

O *Processor Expert* nos ajuda a inicializar os periféricos conforme os requisitos de projeto (utilizar o canal 0 do módulo TPM2 e amostragem a cada 125 ms). Ele gera ainda um arquivo events.c que inclui as duas rotinas de tratamento de interrupções: **TI1_OnInterrupt**(**void**) e **AD1_OnEnd**(**void**). Basta incluir nestas rotinas de tratamento as instruções apropriadas. Na primeira seria a amostragem dos valores e na segunda rotina as instruções conforme os requisitos de projeto: no item 2, verificar o valor convertido no modo de depuração; e no item 3, deve-se alterar a frequência das piscadas do *led* conforme o valor do potenciômetro. Com base no que vocês fizeram no experimento 2 (a frequência das piscadas controlada pelo canal 0 do módulo TPM1) e no que foi apresentado neste documento, em quais registradores, TPMxMOD, TPMxCnV e/ou TPM1SC, vocês poderiam atuar para alterar a frequência ds piscadas do *led* ligado no pino 55 durante o tempo de execução do programa? Como?