Aula prática N.º 8

Objetivos

- Programação e utilização de *timers*.
- Utilização das técnicas de *polling* e de interrupção para detetar a ocorrência de um evento e efetuar o consequente processamento.

Introdução

Timers são dispositivos periféricos de grande utilidade em aplicações baseadas em microcontroladores permitindo, por exemplo, a geração de eventos de interrupção periódicos ou a geração de sinais PWM (*Pulse Width Modulation*) com *duty-cycle* variável. O seu funcionamento baseia-se na contagem de ciclos de relógio de um sinal com frequência conhecida. O PIC32 disponibiliza 5 *timers*, T1 a T5, que podem ser usados para a geração periódica de eventos de interrupção ou como base de tempo para a geração de sinais PWM. Esta última funcionalidade está reservada aos timers T2 e T3 e é implementada recorrendo ainda a um módulo designado pelo fabricante por *Output Compare Module*.

No PIC32MX795F512H (versão usada na placa DETPIC32), os *timers* T2 a T5 são do tipo B e o T1 é do tipo A. A principal diferença entre o *timer* de tipo A e os de tipo B reside no módulo *prescaler* (pré-divisor) que apenas permite, no de tipo A, a divisão por 1, 8, 64 ou 256. Nos de tipo B a constante de divisão pode ser 1, 2, 4, 8, 16, 32, 64 ou 256. Os *timers* do tipo B podem ser agrupados dois a dois implementando, desse modo, um *timer* de 32 bits. A Figura 1 apresenta o diagrama de blocos simplificado de um *timer* tipo B do PIC32.

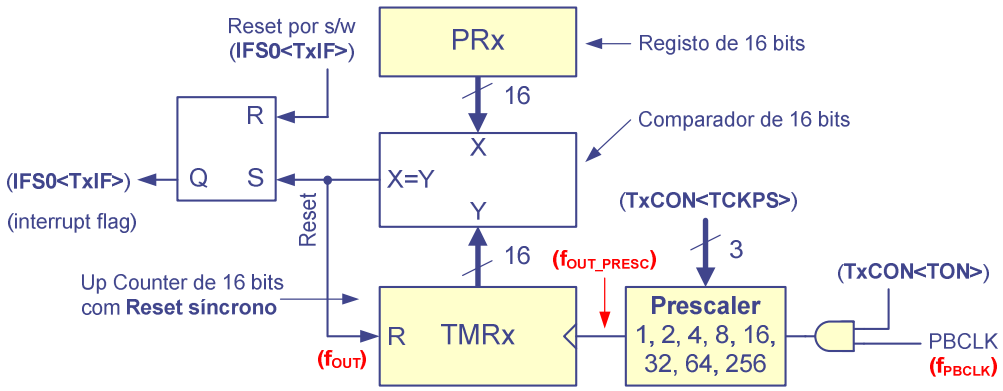

Figura 1. Diagrama de blocos simplificado de um timer tipo B.

Nesta visão simplificada, a fonte de relógio para os *timers* é apenas o *Peripheral Bus Clock* (**`PBCLK`**) que, na placa DETPIC32, está configurado para ter uma frequência igual a metade da frequência do sistema, isto é, f_{PBCLK} = 20 MHz (**`FREQ/2`**, ou **`PBCLK`** em C).

Cálculo das constantes para geração de um evento periódico

O módulo de pré-divisão (*prescaler*) faz uma divisão da frequência $\mathbf{f_{PBCLK}}$ por uma constante configurável nos 3 bits **`<TCKPS>`** do registo **`TxCON`**[1] (designada mais à frente por $\mathbf{K_{PRESCALER}}$), para os *timers* T2 a T5, ou nos 2 bits **`TCKPS`** do registo **`T1CON`**, para o *timer* T1. Por exemplo, no *timer* tipo A, se **`<TCKPS>`** for configurado com o valor 3, a que corresponde uma constante de divisão de 256, o valor de f_{OUT_PRESC} obtido é:

$$f_{OUT\_PRESC} = \frac{f_{PBCLK}}{256} = \frac{f_{PBCLK}}{K_{prescaler}}$$

[1] A letra "x" deve ser substituída pelo número do *timer* (2 a 5). Para informação completa sobre o modelo de programação, deve ser consultado o manual do fabricante "PIC32 Family Reference Manual, Section 14 – Timers".

Conhecida a frequência do sinal à saída do *prescaler*, pode determinar-se a frequência do sinal gerado pelo *timer*, do seguinte modo:

$$f_{OUT} = \frac{f_{OUT\_PRESC}}{PRx + 1}$$

em que **PRx** é o valor da constante de 16 bits armazenada num dos registos **PR1** a **PR5** (*timers* T1 a T5).

Exemplo: determinar o valor de **PR2** e da constante de divisão do *prescaler* de modo a que o *timer* T2 gere eventos de "fim de contagem" a uma frequência de 10 Hz (i.e. a cada 100 ms).

Se o *prescaler* for configurado com o valor 1, então $f_{OUT\_PRESC} = f_{PBCLK}$ = 20 MHz e **PR2** fica:

$$PR2 = \left(\frac{20 \times 10^6}{10}\right) - 1 = 1.999.999$$ **(em C, PR2=PBCLK/10 - 1;)**

Ora, uma vez que o registo **PR2** é de 16 bits, o valor máximo da constante de divisão é **65535** (2^{16}-1), pelo que a solução anterior é impossível. Será então necessário configurar o módulo *prescaler* para baixar a frequência do sinal à entrada do contador do *timer*, de modo a tornar possível a divisão usando uma constante de 16 bits.

$$f_{OUT} = \frac{\left(f_{PBCLK} / K_{PRESCALER}\right)}{(PR2 + 1)}$$

Usando para **PR2** o valor máximo possível (**65535**), podemos determinar o valor mínimo para a constante de divisão do *prescaler* como:

$$K_{PRESCALER} = \left\lceil \frac{f_{PBCLK}}{((65535 + 1) \times f_{OUT})} \right\rceil = \lceil 30.51 \rceil = 31$$

Se, por exemplo, se usar uma constante de divisão de 32 (os valores possíveis seriam 32, 64 ou 256), f_{OUT_PRESC} = 20 MHz / 32 = 625 KHz. Refazendo o cálculo para o valor de **PR2** obtém-se:

$$PR2 = \left(\frac{625 \times 10^3}{10}\right) - 1 = 62499$$

valor que já é possível armazenar num registo de 16 bits.

A obtenção de um evento com a mesma frequência no *timer* T1 obrigaria à utilização de uma constante de divisão de 64, uma vez que o valor 32 não está disponível nesse *timer* (tipo A).

Configuração do *timer*

A programação dos *timers* envolve: i) configuração da constante de divisão do *prescaler* (registo **TxCON**, bits **<TCKPS>**), ii) configuração da constante de divisão **PRx**., iii) ativação do timer (registo **TxCON**, bit **<TON>**). A sequência para a configuração do *timer* T2 com os parâmetros do exemplo anterior é:

```
T2CONbits.TCKPS = 5; // 1:32 prescaler (i.e. fout_presc = 625 KHz)
PR2 = 62499;         // Fout = 20MHz / (32 * (62499 + 1)) = 10 Hz
TMR2 = 0;            // Clear timer T2 count register
T2CONbits.TON = 1;   // Enable timer T2 (must be the last command of the
                     //  timer configuration sequence)
```

O campo TCKPS (Timer Clock Prescaler Select) tem 3 bits:

Binário (TCKPS)	Valor decimal	Fator de divisão (prescaler)
000	0	1:1
001	1	1:2
010	2	1:4
011	3	1:8
100	4	1:16
101	5	1:32
110	6	1:64
111	7	1:256

Configuração do *timer* para gerar interrupções

Se se pretender que o *timer* gere interrupções é necessário, para além da configuração-base apresentada no ponto anterior, configurar o sistema de interrupções na parte respeitante ao *timer* ou *timers* que estão a ser usados, nomeadamente, prioridade (registo **`IPCn`**, bits **`<TxIP>`**), *enable* das interrupções geradas pelo *timer* pretendido (registo **`IEC0`**, bits **`<TxIE>`**) e *reset* inicial do bit **`<TxIF>`** (registo **`IFS0`**)[2]. Para o *timer* T2, a sequência de comandos que configura o sistema de interrupções fica então:

```
IPC2bits.T2IP = 2;    // Interrupt priority (must be in range [1..6])
IEC0bits.T2IE = 1;    // Enable timer T2 interrupts
IFS0bits.T2IF = 0;    // Reset timer T2 interrupt flag
```

TABLE 7-1: INTERRUPT IRQ, VECTOR AND BIT LOCATION

Interrupt Source(1)	IRQ Number	Vector Number	Interrupt Bit Location			
			Flag	Enable	Priority	Sub-Priority
Highest Natural Order Priority						
CT – Core Timer Interrupt	0	0	IFS0<0>	IEC0<0>	IPC0<4:2>	IPC0<1:0>
CS0 – Core Software Interrupt 0	1	1	IFS0<1>	IEC0<1>	IPC0<12:10>	IPC0<9:8>
CS1 – Core Software Interrupt 1	2	2	IFS0<2>	IEC0<2>	IPC0<20:18>	IPC0<17:16>
INT0 – External Interrupt 0	3	3	IFS0<3>	IEC0<3>	IPC0<28:26>	IPC0<25:24>
T1 – Timer1	4	4	IFS0<4>	IEC0<4>	IPC1<4:2>	IPC1<1:0>
IC1 – Input Capture 1	5	5	IFS0<5>	IEC0<5>	IPC1<12:10>	IPC1<9:8>
OC1 – Output Compare 1	6	6	IFS0<6>	IEC0<6>	IPC1<20:18>	IPC1<17:16>
INT1 – External Interrupt 1	7	7	IFS0<7>	IEC0<7>	IPC1<28:26>	IPC1<25:24>
T2 – Timer2	8	8	IFS0<8>	IEC0<8>	IPC2<4:2>	IPC2<1:0>

[2] Para saber quais os registos que deve configurar para um *timer* em particular deve consultar o manual do fabricante "PIC32, Family Reference Manual Section 14-Timers", e o "PIC32MX5XX/6XX/7XX, Family Data Sheet", Pág. 74 a 76 (ambos disponíveis no site da UC).

1º. Escolher um valor de prescaler em que o valor de PRx caiba num registo de 16 bits:

$PRx \leq 65535_{(2^{16})}$

$$PRx = \left(\frac{f_{PBCLK}}{K_{Prescaler} \times f_{out}} \right) - 1$$

f_{out} ↑ frequência desejada à saída

Testes:

Prescaler = 1:

```plaintext
PRx = (20.000.000 / (1 * 2)) - 1 = 9.999.999 ❌
```

Prescaler = 8:

```plaintext
PRx = (20.000.000 / (8 * 2)) - 1 = 1.249.999 ❌
```

Prescaler = 64:

```plaintext
PRx = (20.000.000 / (64 * 2)) - 1 = 156.249 ❌
```

Prescaler = 256:

```plaintext
PRx = (20.000.000 / (256 * 2)) - 1 = 39.062 ✅
```

Trabalho a realizar

Parte I

1. Calcule as constantes relevantes e configure o *timer* T3, de modo a gerar eventos com uma frequência de 2 Hz. Em ciclo infinito, faça *polling* do bit de fim de contagem **T3IF** (**IFS0<T3IF>**) e envie para o ecrã o caracter **'.'** sempre que esse bit fique ativo:

```
int main(void)
{
   // Configure Timer T3 (2 Hz with interrupts disabled)
   while(1)
   {
      // Wait while T3IF = 0
      // Reset T3IF
      putChar('.');
   }
   return 0;
}
```

2. Substitua o atendimento por *polling* por atendimento por interrupção, configurando o *timer* T3 para gerar interrupções à frequência de 2 Hz.

```
int main(void)
{
   // Configure Timer T3 with interrupts enabled
   EnableInterrupts();
   while(1)
   {
      IdleMode()[3]; // CPU enters Idle mode (CPU is halted,
                     // but peripherals continue to operate)
   }
   return 0;
}

void _int_(VECTOR) isr_T3(void)   // Replace VECTOR by the timer T3
                                  //  vector number
{
   putChar('.');
   // Reset T3 interrupt flag
}
```

3. Altere o programa anterior de modo a que o *system call* **putChar()** seja evocado com uma frequência de 1 Hz (como poderá facilmente verificar não é possível obter diretamente, através do timer, a frequência de 1 Hz; uma solução será chamar o *system call* a cada 2 interrupções).
4. O objetivo deste exercício é fazer a configuração do sistema de interrupções e dos *timers* T1 e T3: o *timer* T1 a gerar interrupções à frequência de 5 Hz e o *timer* T3 a gerar interrupções à frequência de 25 Hz.
   a) Determine as constantes relevantes para que o *timer* T1 (tipo A) gere eventos de interrupção a cada 200 ms (5 Hz) e o *timer* T3 (tipo B) gere eventos de interrupção a cada 40 ms (25 Hz).
   b) Escreva o programa principal com todas as configurações necessárias e as Rotinas de Serviço à Interrupção dos *timers* T1 e T3. Nas rotinas de serviço à interrupção deve apenas imprimir um caracter: '1' na RSI do *timer* T1 e '3' na RSI do *timer* T3.

[3] Macro **IdleMode()** definida no ficheiro **detpic32.h**. Consulte o anexo no final do guião.

1) Cálculo do valor das variáveis do 1º exercício

$$f_{out} = \frac{f_{PBCLK}}{K(PR_x + 1)} \rightarrow K \geq \frac{f_{PBCLK}}{f_{out}(1 + PR_x)} \Leftrightarrow K \geq \frac{20 \times 10^6}{2 \times 65535} = 152$$

$PR_x \leq 65535$

$K = 256$

$$PR_x = \frac{f_{PBCLK}}{K \cdot out} - 1$$

$$= \frac{20 \cdot 10^6}{256 \cdot 2} - 1$$

$$= 39062.5 - 1$$

$$= 39061.5$$

≈ 39061 ou ≈ 39062

TCKPS	K
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	256