MICROCONTROLADORES – LABORATÓRIO 02

Flávio Henrique A. dos Santos
flaviohenriqu@gmail.com
Cidade Universitária Profº. José Aloísio de Campos
Av. Marechal Rodon, s/n Jardim Rosa Elze
CEP - 49100-00
São Cristóvão - SE

Marsol Luz Araújo
marsollaraujo@gmail.com
Cidade Universitária Profº. José Aloísio de Campos
Av. Marechal Rodon, s/n Jardim Rosa Elze
CEP - 49100-00
São Cristóvão - SE

Resumo: Este artigo tem o intuito de demonstrar os resultados obtidos nas experiências realizadas no 2º Laboratório da disciplina Microcontroladores. Para a realização das quatro experiências deste laboratório foi utilizado um *protoboard* com alguns periféricos associados ao microcontrolador DSPIC30F4011 tais como, *displays* de 7 segmentos, botões e ventoinha; um gravador de PIC, os *softwares MPLAB C30* e *PICKit2*; osciloscópio e gerador de função. Os conceitos utilizados neste laboratório foram Interrupção e Temporizador.

Palavras Chave: Microcontrolador, DSPIC30F4011, interrupção, temporizador.

Abstract: This paper has the objective to show the results of experiences that were made during the second lab of the subject of Microcontrollers. To perform the 4 experiences from this lab were used a protoboard with some associated peripherals to the microcomputer DSPIC30F4011, such as displays of 7 segments, buttons and cooler; a PIC recorder, the MPLAB C30 e PICKit2 softwares; oscilloscope and a function generator. The concepts that were used in this laboratory were Interruption and Timers.

Keywords: Microcontroller, DSPIC30F4011, interruption, timer.

1 INTRODUÇÃO

À medida que os sistemas ficam mais complexos, são necessários métodos que viabilizem o processamento de informação e controle de entrada e saída de dados. No DSPIC30F4011 alguns destes métodos são a interrupção e o temporizador.

A interrupção permite ao sistema responder assincronamente a um evento e lidar com este enquanto outro programa está sendo executado. Um exemplo disto é a entrada manual usando um teclado de um forno microondas, em que enquanto está em pleno funcionamento o usuário pode alterar o tempo, tipo de preparo da comida ou cancelar o preparo.

O temporizador é um módulo que permite a contagem de tempo para "*clock* de tempo real" (RTC) ou para operar como temporizador/contador *free-running*. O DSPIC30F4011 dispõem de 5 módulos de *timer*, que podem ser usados de forma independente (contagem síncrona ou assíncrona de 16 bits) ou combinados (contagem de 32 bits).

As experiências realizadas neste laboratório misturam esses dois conceitos em sistemas reais, descritos a seguir:

- Experiência 1: selecionar a velocidade da ventoinha utilizando o botão de interrupção (INT0) para escolher modo de velocidade: lento (30% de *duty cycle*), médio (50% de *duty cycle*) e veloz (80% de *duty cycle*). Mostrar nos *displays* de 7 segmentos os valores do *duty cycle* de cada velocidade;

- Experiência 2: contar o número de voltas da ventoinha no modo de velocidade: lento (20% de *duty cycle*), médio (50% de *duty cycle*) e veloz (80% de *duty cycle*). Alterar o modo no botão de interrupção (INT0). Mostrar a quantidade de voltas de cada modo nos *displays* de 7 segmentos;

- Experiência 3: projetar um frequencímetro para sinais quadrados em nível *TTL*, que possa medir em Hz, KHz e MHz. Deve haver um botão para mudar

a faixa de medição, e valor da frequência deve ser mostrado nos *displays* de 7 segmentos;

- Experiência 4: alterar a velocidade do motor da ventoinha utilizando duas teclas (uma para escolher o valor de *PWM* e outra para confirmar). O *PWM* na faixa de 0% a 100%, e alterado em passos de 1%. O resultado aparece nos *displays* de 7 segmentos.

2 INTERRUPÇÃO

Uma interrupção é a ocorrência de uma condição que causa uma suspensão temporária de um programa enquanto que a condição é atendida por outro programa.

Um sistema de interrupção dá a ilusão de estar fazendo muitas instruções simultaneamente. A CPU não pode executar mais que uma instrução por vez, mas pode interromper a execução de um programa temporariamente, e executar outro, e depois retorna ao programa original.

O DSPIC30F4011 tem 30 fontes de interrupção, estas são habilitadas, priorizadas e controladas usando registradores de funções especiais. Estes registradores são: IFS, IEC, IPC, IPL e INTCON.

2.1 Registrador IFS

Existem 3 registradores IFS , cada um com 16 bits. Os *flags* são setados por seus respectivos periféricos ou sinais externos e são zerados via *software*. Se algum bit do registrador receber o valor 1, a interrupção requisitada já ocorreu; se fo 0, a interrupção requisitada não ocorreu.

O *flag* para o botão de interrupção INT0 pertence ao registrador IFS0, sendo o bit 0 deste.

2.2 Registrador IEC

Existem 3 registradores IEC, cada um com 16 bits. Todos os bits de controle de habilitação de interrupções são mantidos nestes três registradores. Estes bits de controle são usados para habilitar interrupções individualmente desde sinais periféricos ou externos.

Se algum bit do registrador receber o valor 1, a interrupção está habilitada; se fo 0, a interrupção não está habilitada.

O bit de habilitação do botão de interrupção INT0 é o bit 0 do registrador IEC0.

2.3 Registrador IPC

Existem 11 registradores IPC de 16 bits e 1 registrador de 8 bits. O nível de prioridade atribuído pelo usuário é associado com cada uma das interrupções e é mantido nestes doze registradores

A prioridade é "dita" através de 3 bits, 000 é a menor prioridade e 111 é a maior prioridade.

Para o botão INT0, a prioridade é setada através dos 3 primeiros bits do resgistrador IPC0.

2.4 Registrador IPL

Existe somente um registrador IPL, e este possui 4 bits. O nível de prioridade atual é armazenado nos bits IPL.

2.5 Registradores INTCON

São dois registradore de 15 bits, INTCON1 e INTCON2; eles funcionam como controles de interrupção global. O INTCON1 contém os *flags* de controle e *status* para as exceções do processador. O registrador INTCON2 controla o comportamento do sinal de requerimento de interrupção externa e o uso da AIVT (*Alternate Interrupt Vector Table*).

Para o botão de interrupção INT0, o bit 0 do registrador INTCON2 é responsável por "dizer" se a interrupção ocorre na borda negativa, o bit 0 recebe 1; ou na borda positiva, o bit 0 recebe 0.

3 TEMPORIZADORES

Na família de dispositivos DSPIC30F, há vários temporizadores de 16 bits, cujos nomes são *Timer*1, *Timer*2, *Timer*3, ..., etc. No DSPIC30F4011, utilizado neste laboratório, há 5 *timers*. Cada *timer* é um módulo temporizador/contador que contém os seguintes registradores de leitura/escrita: TMRx (Registrador de contagem do *timer* de 16 bits), PRx (Registrador de período de 16 bits associado com o *timer*), TxCON (Registrador de Controle de 16 bits associado com o *timer*), TxIE (bit de controle de habilitação de interrupção), TxIF (bit de estado do *flag* de interrupção) e TxIP<2:0> (bits de controle de Prioridade de Interrupção).

3.1 Registrador TMRx

O registrador TMRx, onde *x* pode ser de 1 a 5, está relacionado à contagem do *timer* de 16 bits. Este registrador armazena um valor (de até 16 bits) que é incrementado continuamente segundo as configurações do temporizador.

O temporizador pode ser configurado para incrementar o *timer* a cada pulso de *clock* interno ou após vários pulsos (de acordo com um conjunto de *prescalers* disponíveis). O incremento pode ocorrer também assincronamente, a partir

de um *clock* externo aplicado, o qual é sincronizado com a fase interna do *clock*.

3.2 Registrador TxCON

O registrador TxCON, onde *x* pode variar de 1 a 5, é um registrador de controle de 16 bits que configura o comportamento do *timer*. Este registrador pode controlar se o *clock* será externo ou interno; síncrono ou assíncrono; configurar o prescaler selecionável (1, 8, 64 ou 256); se será habilitado o modo *gated time accumulation*; e se habilita ou não a contagem do *timer*. As funções e seus respectivos bits podem ser consultados na tabela 1 (referente ao *timer* 1).

Tabela 1. Registrador T1CON

T1CON : Timer1 Control Register (Tipo A)		
Bit	**Nome**	**Descrição**
0	-	**Não Implementado. Ler como 0**
1	TCS	**Timer Clock Source Select bit** 1 = External clock from pin TxCK 0 = Internal clock (FOSC/4)
2	TSYNC	**Timer External Clock Input Synchronization Select bit** When TCS = 1: 1 = Synchronize external clock input 0 = Do not synchronize external clock input When TCS = 0: This bit is ignored. Read as '0'. Timer1 uses the internal clock when TCS = 0
3	-	**Não Implementado. Ler como 0**
5-4	TCKPS<1:0>	**Timer Input Clock Prescale Select bits** 11 = 1:256 prescale value 10 = 1:64 prescale value 01 = 1:8 prescale value 00 = 1:1 prescale value
Bit	**Nome**	**Descrição**
6	TGATE	**Timer Gated Time Accumulation Enable bit** 1 = Gated time accumulation enabled 0 = Gated time accumulation disabled (TCS must be set to '0' when TGATE = 1. Reads as '0' if TCS = 1)
7-12	-	**Não Implementado. Ler como 0**
13	TSIDL	**Stop in Idle Mode bit** 1 = Discontinue timer operation when device enters Idle mode 0 = Continue timer operation in Idle mode
14	-	**Não Implementado. Ler como 0**
15	TON	**Timer On Control bit** 1 = Starts the timer 0 = Stops the timer

3.3 Registrador PRx

O registrador PRx, onde *x* pode variar de 1 a 5, é um registrador de número de instruções associado ao *timer*. Este registrador armazena um valor pré-definido que será a fonte de interrupção do *timer* a ele associado.

Assim, toda vez que a contagem do TMRx casar com o valor do PRx, o *timer* será resetado e uma requisição de interrupção é gerada (TxIF) no registrador IFS. Após o reset do *timer*, a contagem continua, mesmo durante a rotina de interrupção.

O PR pode ser calculado por:

$$PR = T * \frac{Mips}{TCKPS} \quad (1)$$

Em que *T* é o período (em segundos); *Mips* são as instruções por segundo do microcontrolador (16 milhões para esse laboratório); e *TCKPS* é o prescaler que pode variar entre 1, 8, 64 e 256.

4 CONCEPÇÃO DAS EXPERIÊNCIAS

Neste laboratório foram implementadas quatro experiências que utilizam interrupção externa, temporizadores e temporizadores com interrupção.

4.1 Experiência 1

A primeira experiência consistia em controlar a velocidade da ventoinha de três modos distintos: lento, médio e rápido. O sistema iniciaria no modo lento (*duty cycle* de 30%). Pressionando uma tecla ligada ao pino de interrupção externa INT0, a velocidade mudaria para veloz (*duty cycle* de 80%) e, pressionando novamente, mudaria para média (50%), repetindo o ciclo em novos pressionamentos. O *duty cycle* selecionado seria mostrado nos *displays* de 7 segmentos. A frequência do sinal no pino de saída para a ventoinha foi de 200 Hz.

O fluxograma do programa pode ser conferido na Figura 1. Para solução deste problema foi utilizado Máquina de Estado Finito (FMS), um estado para cada modo de velocidade. Em cada possível caso (estado) ocorre à atualização de variáveis correspondentes ao novo *duty cycle* (utilizada na rotina do *timer1*) e ao próximo estado (utilizada na rotina da interrupção externa 0).

Foi feito o uso de apenas um temporizador, o *timer1* (16 bits), e sua função foi gerar o *duty cycle*. Quando em execução, este faz a atualização do registrador *PR1* através de uma equação envolvendo o período do sinal (0,005s), inverso da frequência 200 Hz do sinal, e a variável referente ao *duty*, e assim produz no pino da ventoinha um sinal correspondente a cada modo de velocidade requerido. No primeiro caso (pressionando uma primeira vez), a variável *duty* recebe 8, referindo-se a 80%. No segundo caso, *duty* recebe 5 (50%); e no terceiro, *duty* recebe 3 (30%). Se a saída da ventoinha está em alto, ela é desligada e o *PR1* recebe o valor complementar ao *duty cycle* (período em BAIXO), isto é:

$$PR1 = T * \frac{(10 - duty)}{10} \quad (2)$$

Por outro lado, quando o *timer* atingir esse *PR1*, a saída da ventoinha que estava em BAIXO, vai para ALTO e *PR1* recebe o período vezes o *duty cycle*, isto é:

$$PR1 = T * (10 - duty) \quad (3)$$

O ciclo continua indefinidamente. O valor de *duty* também é enviado aos *displays* (acrescentando o "0" para formar os valores 80, 50 e 30).

Em relação à interrupção externa, foi utilizada para alteração do modo de velocidade da ventoinha. Durante sua rotina, a variável correspondente ao estado atual é atualizada.

Para a criatividade, foi escolhido mostrar a frase "Speed c" se deslocando para a esquerda nos *displays* enquanto era pressionada uma tecla. A variável *c* podia ser 1 (para a primeira velocidade, ou seja, veloz), 2 para média e 3 para a lenta.

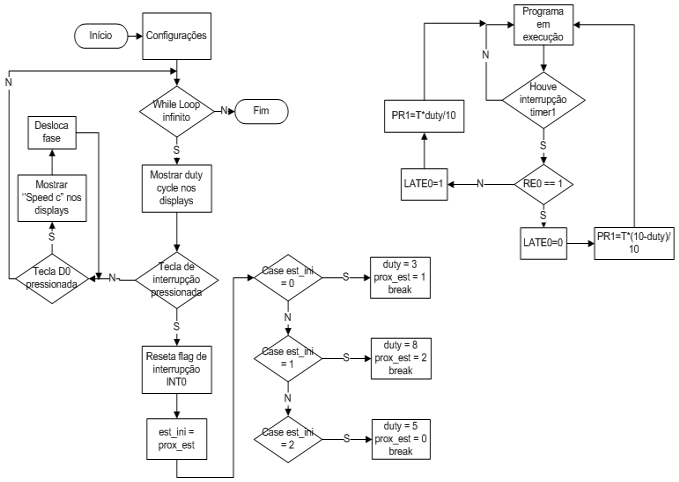

Figura 1. Fluxograma para experiência 1.

4.2 Experiência 2

A segunda experiência era basicamente igual à primeira, com a diferença de que a velocidade "lenta" era com o *duty cycle* de 20% e nos *displays* seria mostrado o número de rotações por segundo da ventoinha. E a frequência de operação agora é de 500 Hz.

Neste caso, foi necessário o uso de um *timer* com *clock* externo para receber os pulsos do *encoder* infravermelho; outro *timer* para contar cada segundo; e um terceiro com *clock* interno para enviar o sinal modulado para a ventoinha.

A ideia da experiência 2 pode ser conferida no fluxograma da Figura 2. Pela figura, nota-se que as instruções são semelhantes às da experiência 1.

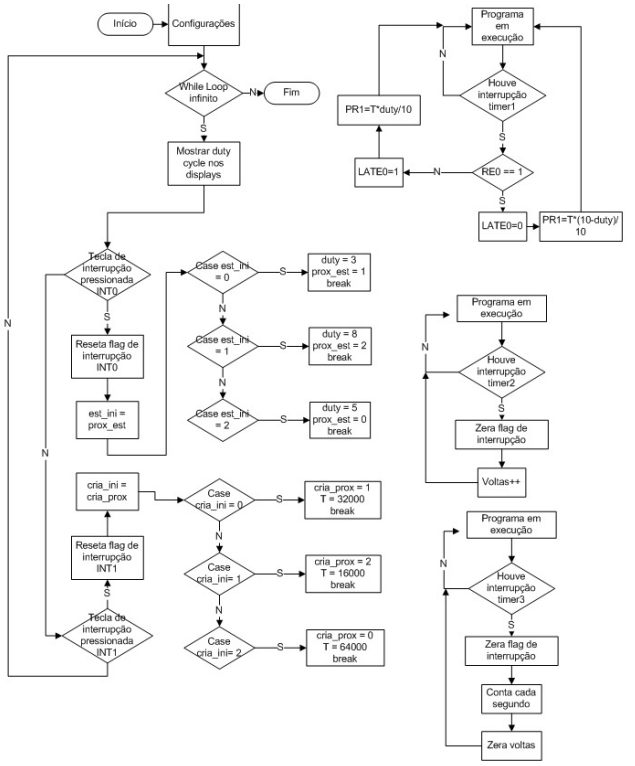

Figura 2. Fluxograma para a experiência 2.

A interrupção do *timer1* é idêntica à da questão anterior, para controlar o PWM enviado à ventoinha. Já a interrupção do *timer2* incrementa a variável *voltas* a cada sete pulsos do *encoder*. Este número se deve às sete hélices da ventoinha. Assim, setando o *PR2* para 7 e configurando o *timer*2 para utilizar o *clock* externo, cada vez que as hélices completavam uma volta, a interrupção ocorria, ou seja, *voltas* incrementava. Por fim, a cada 1 segundo (tempo controlado pelo *timer3*), ocorria outra interrupção (T3), na qual era armazenado *voltas*, isto é, era armazenado o número de voltas por segundo.

Para a criatividade, foi programada uma função que altera a freqüencia de operação do sinal para a metade (250 Hz) e para o dobro (1 kHz).

4.3 Experiência 3

A terceira experiência consistia em um frequencímetro para sinais quadrados a nível TTL, que poderia medir em Hz, KHz ou MHz. Para mudar as faixas de frequência, a tecla ligada ao pino de interrupção externa INT0 deveria ser acionada, e a frequência medida do sinal (vindo de um gerador de funções) deveria ser mostrada nos *displays*.

O fluxograma para a experiência 3 pode ser conferido na Figura 3.

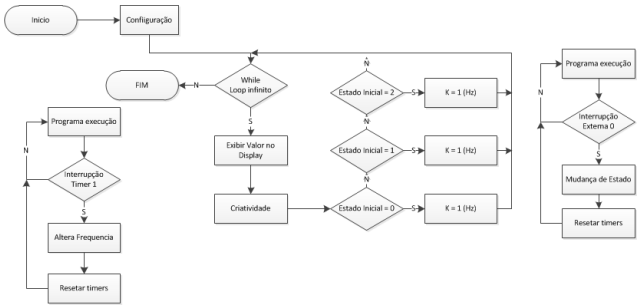

Figura 3. Fluxograma para a experiência 3.

O programa segue a mesma lógica da experiência anterior. Seleção dos estados através de uma máquina de estados finitos (FSM), um sinal externo funcionaria como *clock* externo que incrementaria as variáveis TMR2/3 (*timer*s concatenados para abranger a maior faixa de medição possível). Quando se passasse 1 segundo, ocorreria a interrupção do *timer* 1 e o valor de TMR3 com TMR2 seria armazenado na variável *freq*, ou seja, *freq* indicaria quantos pulsos o sinal deu por segundo. Pode ser observado pela Figura 3 que *freq* recebe TMR3 e o deslocam 16 bits, para então somar TMR2. Assim, é possível chegar a um valor de até 32 bits.

As faixas de medição eram mudadas com uma máquina de estados pela interrupção INT0. A variável de controle *K* receberia o fator de escala, 1 para Hz; 1000 para KHz e 1 000 000 para MHz, de modo que o valor mostrado nos *displays* seria *freq/K*. A frequência máxima medida foi de 5 MHz.

Para a criatividade, foi escolhido alterar a visualização da frequência para o modo de notação científica. A casa dos milhares nos *displays* seria preenchida com "E" – referente a exponencial, e os *leds* da porta E indicariam a potência de 10. Assim, se a frequência medida foi de 1000 Hz, ao se pressionar a tecla RF0 seria mostrado "1E3" nos display.

4.4 Experiência 4

A última experiência consistia em projetar um sistema no qual se poderia selecionar um PWM para controlar a velocidade da ventoinha pelo teclado matricial, desde que este estivesse habilitado. A habilitação deveria ocorrer mediante o pressionamento de uma tecla ligada ao pino de interrupção externa INT0, e dois botões para fazer a seleção do valor do *duty cycle*, um botão seleciona o valor e o outro confirma.

O fluxograma da experiência 4 pode ser visto na Figura 4. O PWM é obtido pelo mesmo princípio das experiências 1 e 2, ou seja, na interrupção do *timer* 1 controla-se o PR1 segundo as equações (2) e (3).

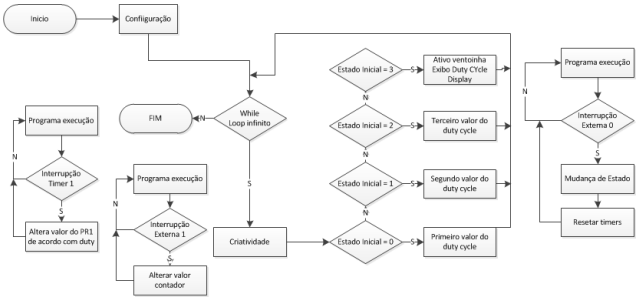

Figura 4. Fluxograma para a experiência 4.

Quando ocorre a interrupção externa, ou seja, quando se é pressionada a tecla ligada ao pino INT0, a máquina de estados finitos faz a mudança de estado iniciando a entrada de dados. Após selecionar o valor do *duty cycle* o último estado exibiu no display o valor do *duty* selecionado e ativa a ventoinha. O valor mínimo de *duty cycle* que fazia a ventoinha girar foi de 10%.

A criatividade para esta experiência consistia em calcular a tensão média de saída e mostrá-la enquanto a tecla RF0 estivesse pressionada. Para tal cálculo, foi utilizada a operação *floor* (disponível na biblioteca *math.h*), três variáveis flutuantes *a, b* e *c*, e duas inteiras *t1* e *t2*. Como o sinal de saída era de 5 volts, a tensão média seria 5 vezes o *duty cycle*. Por exemplo, se o *duty* digitado fosse 50 (%), a tensão média seria 2,5 volts. Logo, para mostrar tais valores foi feito o seguinte código:

```
// ----------------------------------------
c = (duty*0.05);        // Se duty for 50%, c
                        //recebe 2,5 V
a = floor(c);           // a recebe c
                        // truncado para baixo,
                        // ou seja, 2 volts
b = (c-a);              // b recebe a diferença
                        //entre c e a, nesse caso,
                        // 2,5 - 2 = 0,5 volts
b = 10*b;
t1 = a;
t2 = b;

// ----------------------------------------
```

Assim, *t1* receberia 2 e *t2* receberia o 5 (no caso do exemplo citado). Estas duas variáveis estavam contidas em um vetor *frase* que era mostrado nos *displays* de 7 se deslocando para a esquerda enquanto a tecla RF0 era pressionada.

5 CONCLUSÃO

A utilização dos temporizadores e interrupções permitiu um aumento no número de aplicações do microcontrolador. Com tais recursos, foi possível utilizar periféricos como um led infravermelho, um fotodiodo e *clock* externo, além de permitir a criação de ondas quadradas com pulsos modulados ou até medir a frequência de sinais na entrada.

As interrupções apresentaram uma lógica bem simples, mas aquelas envolvendo os temporizadores exigiram manobras de programação, já que os *timers* continuam contando mesmo durante a rotina de interrupção. Este fator foi a causa de inconsistências observadas durante o experimento, quando as ondas geradas eram observados no osciloscópio. Com a lógica descrita pelas equações (2) e (3), foi obtido o melhor resultado, havendo boa precisão nos *duty cycles*. O frequencímetro apresentava uma lógica mais simples e foi implementado sem muitos problemas, havendo apenas a limitação do próprio microcontrolador, que não alcançou frequências acima de 5 MHz.

No mais, as experiências funcionaram conforme solicitado e este laboratório abriu um leque de possibilidades para futuras aplicações.

6 REFERÊNCIA BIBLIOGRÁFICA

[Hidalgo, 2013.1] Hidalgo, A. R. 2013.1, Apostilas da disciplina Microcontroladores da Universidade Federal de Sergipe ministrada pelo Prof. Dr. Antônio Ramirez Hidalgo.

[Souza, 2008] Souza, V. A. Programação em C para o DSPIC: Fundamentos – São Paulo: Ensino Profissional, 2008.