MICROCONTROLADORES – LABORATÓRIO 03

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 3º Laboratório da disciplina Microcontroladores. Para a realização das três experiências deste laboratório foi utilizado um *protoboard* com alguns periféricos associados ao microcontrolador dsPIC30F4011 tais como, LCD 16x2, Teclado PS/2, LED infravermelho, botões e ventoinha; um gravador de PIC, os *softwares MPLAB C30* e *PICKit2*; osciloscópio e gerador de função. O principal conceito abordado neste laboratório foi Input Capture.

Palavras Chave: Microcontrolador, dsPIC30F4011, interrupção, temporizador, input capture.

Abstract: This article aims to demonstrate the results obtained in experiments on the 3rd Microcontrollers Laboratory discipline. For the realization of the three experiments of this lab used a breadboard with some peripherals associated with the dsPIC30F4011 microcontroller such as 16x2 LCD, PS / 2, Infrared LED, buttons and fan; recorder PIC, MPLAB C30 and software PICKit2; oscilloscope and function generator. The main concept was addressed in this lab Input Capture.

Keywords: Microcontroller, dsPIC30F4011, interruption, timer, input capture.

1 INTRODUÇÃO

Em sistemas onde envolva a captura de dados referente à frequência (ou período) e largura de pulso, é utilizado o módulo *Input Capture*. Este módulo é utilizado para capturar valores de tempo de umas das bases de tempo selecionáveis de um evento no pino de entrada do kit dsPIC30F Genios.

Atuando juntamente com os conceitos de *Input Capture*, Interrupção e Temporizador, é possível criar sistemas que interajam com o ambiente externo ao do kit, possibilitando sistemas maiores e complexos.

Os sistemas mencionados no parágrafo anterior são representados pelas experiências realizadas neste laboratório. Estas são:

- Experiência 1: selecionar a velocidade da ventoinha utilizando o botão de interrupção (INT0) para escolher o modo de velocidade: veloz, médio e lento. Exibir no LCD 16x2 as rotações por minuto (RPM) de cada velocidade;

- Experiência 2: apresentar um menu no LCD 16x2, de forma vertical, com opções para controlar a velocidade da ventoinha, alternar o *buzzer* de dois em dois segundo e desliga-lo, e deslocar uma mensagem.

- Experiência 3: utilizando o sensor de ultrassom para medir distâncias, exibindo as distâncias no LCD 16x2 e ligar e desligar o *buzzer* a depender da distância.

2 INPUT CAPTURE

O dsPIC30F4011 possui 4 entradas de captura: IC1, IC2, IC7 e IC8, eles são funcionalmente idênticos, em que um ‘x’ em ICx indica a entrada de captura.

O módulo *Input Capture* tem a tarefa de obter o valor atual do contador do timer (TMRx) em um evento de entrada, para fazer medições de frequência, períodos ou pulsos.

Este módulo tem vários modos de operação selecionáveis com o registrador ICxCON (bits de controle ICM<2:0>).

Os modos de operação incluem:

- Valor do *Timer* em toda borda de descida da entrada aplicada no pino ICx;
- Valor do *Timer* em toda borda de subida da entrada aplicada no pino ICx;
- Valor do *Timer* em toda quarta borda de descida da entrada aplicada no pino ICx;
- Valor do *Timer* em toda 16ª borda de descida da entrada aplicada no pino ICx;
- Valor do *Timer* em toda borda de subida e descida da entrada no pino ICx.

São características do módulo o modo evento de captura simples, seleção modo *Timer* 2 e *Timer* 3 e Interrupção por evento de captura de entrada. O módulo captura de entrada tem um *buffer* FIFO.

Os bits do registrador ICxCON podem ser vistos nas tabelas 1.

ICxCON: Input Capture x Control Register					
Bit	**Nome**	**Descrição**	**Bit**	**Nome**	**Descrição**
2-0	ICM<2:0>	**Input Capture Mode Select bits** 111 = Input Capture functions as interrupt pin only, when device is in Sleep or Idle mode (Rising edge detect only, all other control bits are not applicable.) 110 = Unused (module disabled) 101 = Capture mode, every 16th rising edge 100 = Capture mode, every 4th rising edge 011 = Capture mode, every rising edge 010 = Capture mode, every falling edge 001 = Capture mode, every edge (rising and falling) (ICI<1:0> does not control interrupt generation for this mode.) 000 = Input capture module turned off	7	ICTMR	**Input Capture Timer Select bits** 1 = TMR2 contents are captured on capture event 0 = TMR3 contents are captured on capture event
3	ICBNE	**Input Capture Buffer Empty Status (Read Only) bit** 1 = Input capture buffer is not empty, at least one more capture value can be read 0 = Input capture buffer is empty	12-8	-	**Não Implementado. Ler como 0**
4	ICOV	**Input Capture Overflow Status Flag (Read Only) bit** **1 = Input capture overflow occurred** **0 = No input capture overflow occurred**	13	ICSIDL	**Input Capture Module Stop in Idle Control bit** 1 = Input capture module will halt in CPU Idle mode 0 = Input capture module will continue to operate in CPU Idle mode
6-5	ICI<1:0>	**Select Number of Captures per Interrupt bits** 11 = Interrupt on every fourth capture event 10 = Interrupt on every third capture event 01 = Interrupt on every second capture event 00 = Interrupt on every capture event	15-14	-	**Não Implementado. Ler como 0**

Tabela 1 – Bits do registrador ICxCON [1].

3 CONCEPÇÃO DAS EXPERIÊNCIAS

Neste laboratório foram implementadas quatro experiências que utilizam *Input Capture*, interrupções externas, temporizadores e temporizadores com interrupção.

3.1 Experiência 1

A primeira experiência consistia em controlar a velocidade da ventoinha de três modos distintos: lento, médio e rápido. O sistema iniciaria parado. Pressionando uma tecla ligada ao pino de interrupção externa INT0, a velocidade mudaria para veloz (*duty cycle* de 80%), pressionando novamente, mudaria para média (50%) e, pressionando uma terceira vez, mudaria para lenta (30%), repetindo o ciclo em novos pressionamentos. O *duty cycle* selecionado seria mostrado no LCD, bem como o número de rotações por minuto (RPM) e o estado, isto é, "VELOZ", "MÉDIA" ou "LENTA". A frequência do sinal no pino de saída para a ventoinha foi de 1000 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*), ao próximo estado (utilizada na rotina da interrupção externa 0) e é calculado o valor do RPM utilizando o *input capture*.

Foi feito o uso dos temporizadores *timer1*, *timer2*, *timer3* o *timer4*, 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 (0,001s), inverso da frequência 1000 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:

$$PR1 = T * \frac{(10 - duty)}{10} \quad (1)$$

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:

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

O ciclo continua indefinidamente. O valor de *duty* também é enviado ao LCD.

O *timer2* foi utilizado para variar a entrada do *Input Capture* 1 (IC1) tornando possível o calculo do número de rotações da ventoinha. O *timer3* foi utilizado juntamente com o IC1. E o *timer4* ficou responsável por calcular o número de rotações por segundo (RPS).

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.

O *Input Capture* 1, teve em sua configuração a captura nas bordas de subida e descida e utilizou o *timer3*.

Para a primeira criatividade, foi escolhido para ativar o *buzzer* quando as rotações por minuto atingisse um valor crítico. Na segunda criatividade, foi apresentado no LCD

16x2 um gráfico de barras indicando o percentual de rotação da ventoinha.

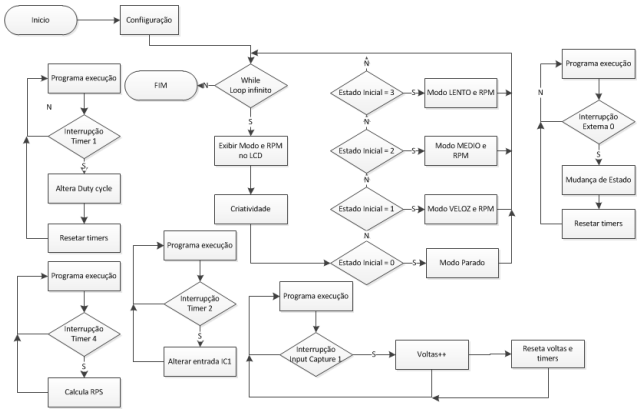

Figura 1 – Fluxograma da experiência 1

3.2 Experiência 2

Na segunda experiência o objetivo da questão era projetar e implementar um sistema com um menu para controlar vários periféricos do kit. O menu foi controlado pelo teclado PS/2 e será apresentado no display LCD do kit. O menu, que deve ser de forma vertical, é apresentado a seguir:

CONTROLE DE PERIFÉRICOS

a) Ventoinha

1) *Duty cycle* 100%

2) *Duty cycle* 50%

3) *Duty cycle* 30%

b) *Buzzer*

1) Liga e desliga em intervalos de 2 seg.

2) Desliga

c) Mensagem

O menu tem três periféricos a serem controlados, são eles: Ventoinha, *Buzzer* e Mensagem (Aparecendo no LCD). Como meio de navegação optou-se por utilizar as teclas 8, 5 e 2 do teclado numérico, as funções de cada tecla estão representadas na tabela 2.

Tecla	**Função**
8 Numérico	Navega para a opção superior a atual
5 Numérico	Seleciona a opção atual
2 Numérico	Navega para a opção inferior a atual

Tabela 2. Função das Teclas.

O programa principal do sistema contém as configurações necessárias para o funcionamento do sistema, as configurações usadas foram: Utilização do RC14 e RC13 como entradas - para o funcionamento do teclado -, habilitação das interrupções dos *timers* 1 e 2/3 e configuração dos mesmos.

O *timer* 1 foi utilizado para o *buzzer* ligar e desligar em intervalos de 2 segundos, o *timer* 2/3 foi utilizado para controlar o *duty cycle* da ventoinha, com valores calculados em outros laboratórios para configurar os registradores dos *timers*. Uma variável auxiliar armazena a velocidade desejada da ventoinha entre LENTA, MEDIA e VELOZ - *Duty cycle* de 30%, 50% e 100% respectivamente.

A configuração do LCD é feita pela biblioteca fornecida pelo professor com alterações de alunos do período 2012-2. O código fornecido pelo professor para controlar o teclado PS/2 foi encapsulado em uma biblioteca facilitando a configuração do mesmo. Para o teclado PS2 utilizou-se o *Change Notification* como interrupção.

Além disso, foi acrescentada uma variável que armazena um apontador para uma função a ser chamada quando o teclado interrompe, facilitando a utilização do usuário que basta codificar uma função como se fosse uma interrupção e configurá-la chamando a função setKBDFunc(). A função codificada para ser utilizada como uma interrupção do teclado foi a trataTecla() que possui o comportamento descrito pelo fluxograma da figura 2.

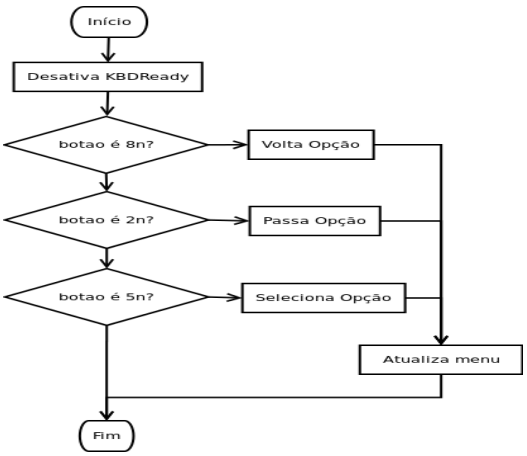

Figura 2. Fluxograma para a função trataTecla().

A função trataTecla() verifica o valor do KBDCode e caso seja o 8, 5 ou 2 numéricos ele realiza a ação

correspondente de acordo com a tecla pressionada. Caso tenha sido o 8 ou 2 numéricos o programa muda a opção atual para a superior ou inferior e atualiza o menu no display LCD. Caso tenha sido o 5 numérico então verifica a opção atual e realiza a ação correspondente, as ações correspondentes podem ser vistas pelo menu exigido na questão.

Para as criatividades optou-se por melhorar o sistema de navegação do menu que não estava muito eficiente, primeiramente pensamos em implementar uma tecla para voltar ao menu anterior - e assim poder corrigir um erro de pressionar de tecla - e logo após pensamos em como agilizar a seleção dos menus.

A implementação da opção de voltar utilizou a tecla *BackSpace*. Na função trataTecla() verificamos se a tecla pressionada corresponde ao *BackSpace* e em caso afirmativo o programa volta ao primeiro menu.
Para melhorar a navegação do menu resolvemos utilizar um sistema de escolhas diferentes. As teclas a serem pressionadas se corresponderiam a cada opção dos menus. Caso o menu atual tenha 3 opções, 3 teclas seriam usadas. As teclas utilizadas foram as letras Q, A e X do teclado.

3.3 Experiência 3

A terceira experiência consistia em controlar um sensor de ultrassom, que possibilitava a medição de distância entre ele e um anteparo. Além de capturar a saída do sensor, com um valor referente à distância, o kit do dsPIC30F4011 devia ligar o *buzzer* em intervalos alternados cada vez menores, até que em distâncias medidas menores que 50 centímetros este apitasse intermitentemente.

O sensor de ultrassom funciona da seguinte maneira: à medida que a distância ao anteparo (logo, a distância medida) aumenta, a largura do pulso do sensor também aumenta. Inicialmente, para realizar testes com o sensor, usou-se um osciloscópio para estudar o sinal de saída do sensor, sem a preocupação de capturá-lo pelo kit do microcontrolador.

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

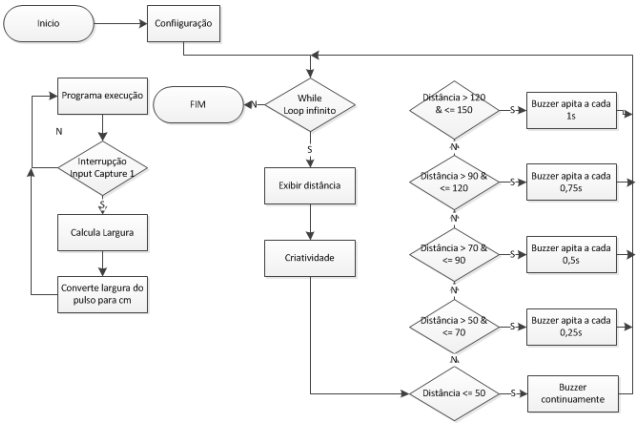

Figura 3. Fluxograma para a experiência 3.

Uma vez que se calculou o valor da largura de pulso, utilizando o *Input Capture* 1, basta comparar o valor obtido pelo PIC com os apresentados pelo osciloscópio e, levando em consideração a distância medida com uma trena, determinar a distância que cada largura de pulso representa. Para um resultado ainda mais preciso, utilizou-se o *datasheet* do sensor, o qual informa que para cada polegada (2,54cm), há um acréscimo de 147 microsegundos na largura do pulso [3].

A próxima etapa é fazer com que o *buzzer* apite em intervalos cada vez menores, a partir de 1,5 metros de distância medida até 50 centímetros (onde ele deve apitar intermitentemente).

Ao entrar na interrupção do *buzzer*, o programa verifica se ele está ligado, alternando seu estado. Ao entrar novamente, o estado é alternado e assim sucessivamente. Além deste comportamento "liga-desliga", o *buzzer* deve apitar em tempos diferentes, de acordo com a distância medida. Para isso, basta variar o valor de PRX, que é o limite da contagem do *timer* associado ao *buzzer*. O valor até o qual deve ser feita a contagem (logo, o valor de PRX) é facilmente encontrado.

A primeira criatividade consistiu na criação de um sistema de acompanhamento visual da distância do anteparo ao sensor. Com ele, é possível realizar o monitoramento de forma facilitada, não só analisando numericamente à distância. Imaginando que o sistema pode ser usado como, por exemplo, um sensor de ré, o usuário iria querer visualizar a distância da forma mais intuitiva possível, o que motivou a criação deste sistema. Quando um obstáculo se aproxima do sensor o número de 'traços' aumenta indicando ao usuário que um obstáculo está cada vez mais próximo, caso contrário, o número de 'traços' diminui, como ilustrado na figura 4.

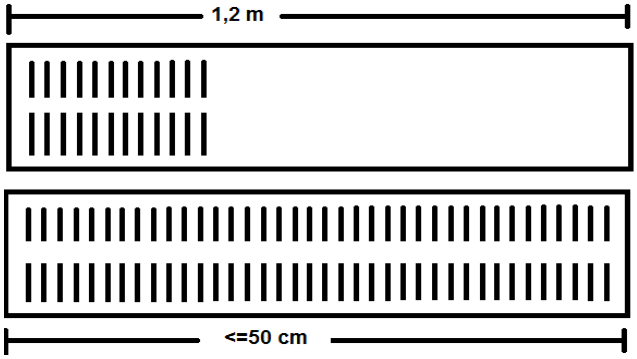

Figura 4. Representação gráfica da criatividade da experiência 3.

A segunda criatividade tinha como objetivo permitir ao usuário alterar a distância mínima que o sensor seria acionado intermitentemente.

4 CONCLUSÃO

Os experimentos realizados neste laboratório mostraram que o uso do periférico de *Input Capture* é uma poderosa ferramenta para a programação em microcontroladores. Usando tal ferramenta, podem-se realizar tarefas à medida que ocorre algum evento no sinal de entrada (Input capture).

A utilização do *Input Capture* permitiu um aumento no número de aplicações do microcontrolador. Com tais recursos, foi possível utilizar periféricos como o *encoder* infravermelho e *clock* externo, além de permitir a criação de ondas quadradas com pulsos modulados, medir a frequência de sinais na entrada, e largura de pulsos.

O *Input Capture* apresentou algumas dificuldades inicialmente na aplicação dos experimentos causando inconsistências. Mas a partir dessas dificuldades iniciais conseguiu-se entender perfeitamente todo o seu funcionamento possibilitando a implementação de todos os experimentos propostos pelo professor e suas criatividades.

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

5 REFERÊNCIA BIBLIOGRÁFICA

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

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

[3] LV-MaxSonar®-EZ1™ High Performance Sonar Range Finder. Datasheet.