Projeto de Sistemas Microcontrolados

Aula 04 - Estudo de microcontroladores PIC – Parte II

Apresentação

Nesta aula você terá informações complementares sobre os microcontroladores PIC das famílias 16F, especificamente, verá como são organizadas suas memórias de dados e aprenderá sobre dois blocos funcionais comuns a praticamente todos os PICs: o temporizador Timer 0 e duas portas de entrada/saída de dados, o PORTA e o PORTB. Além disso, terá uma pequena introdução dos blocos periféricos que permitem fazer comunicação paralela, comunicação serial, captura, comparação e geração de sinais modulados em largura de pulsos e, finalizando, verá como são tratadas as interrupções de programa.

Objetivos

Ao final desta aula você será capaz de:

- Compreender como estão organizadas as memórias de dados (volátil e não volátil) dos microcontroladores PICs e analisar blocos funcionais comuns às famílias 16F e 18F, tais como o temporizador Timer 0 e os PORTA e PORTB, usados para entrada e saída de dados.
- Descrever sobre os blocos periféricos dos microcontroladores PIC que permitem fazer transferências paralelas e seriais de dados, conversão analógica/digital e a captura, comparação e geração PWM.
- Compreender como são tratadas as interrupções de programa interpretando devidamente pedidos de interrupção e definindo algoritmos e rotinas de atendimento a estas interrupções.

Introdução

Na aula anterior, você conheceu detalhes das características gerais dos microcontroladores 16F84, 16F628 e 16F877, estudou suas arquiteturas internas, teve informações referenciais sobre seus principais blocos funcionais constituintes e teve detalhes de como é a organização da memória de programa. Nesta aula, você terá informações sobre a organização das memórias de dados, presentes nas famílias de processadores PIC, e conhecerá mais detalhes sobre seus temporizadores, suas interfaces periféricas, que registros internos estão relacionados a essas interfaces e como funciona o sistema de interrupção de programa relacionado a cada bloco periférico.

Organização da memória de dados

A memória de dados de um microcontrolador PIC das famílias 16F e, também, da 18F se constitui de uma memória volátil, tipo RAM, e uma memória não volátil, do tipo EEPROM.

A memória de dados do tipo RAM é usada para fixar duas classes de registradores: uma constituída por registradores designados por FSR (Special Function Register - Registradores de função especial), os quais são definidos pelo fabricante e usados para manipulação e inicialização de blocos construtivos internos do PIC e outra, que é reservada para os GPR (General Purpose Register – Registradores de Propósito Geral), a serem definidos e manipulados livremente pelo usuário. As variáveis de dados, criadas e manipuladas por nossos programas aplicativos, estão nesta segunda classe de registradores e têm, para cada microcontrolador, áreas específicas, como poderemos ver logo a frente, onde podem ser armazenadas.

Por ser não volátil, a memória EEPROM é usada para armazenamento de dados (como constantes ou tabelas de dados fixas) cujos valores não podem ou não devem ser “perdidos” após um desligamento do microcontrolador e que, necessariamente, devem estar presentes após seu religamento ou inicialização.

A quantidade de *bytes* dessa memória é muito pequena, varia de microcontrolador para microcontrolador, podendo chegar, por exemplo, a um banco de memória com 256 *bytes*, como no caso do 16F877 e dos PICs da família 18F.

Embora não volátil, a memória EEPROM de dados permite que as informações sejam não apenas lidas, mas também escritas durante a execução normal de um programa. Os processos de leitura e escrita nessa memória seguem, no entanto, rituais próprios e envolvem o uso de alguns registradores de propósito especial: o EEADR (que registra o endereço da EEPROM que será lido ou onde será escrito um dado), o EEDATA (que registra o dado lido ou a ser escrito na EEPROM) e outros, a serem tratados durante a programação, que permitem, através de alguns bits especiais, iniciar o processo de leitura ou escrita (como é o caso do EECON1) ou indicar o fim de uma leitura ou escrita (como é o caso do INTCON). Esses processos são diferenciados de uma leitura ou escrita na memória RAM, porque as EEPROMs exigem maiores tempos de acessos do que as RAMs. Ou seja, as primeiras não conseguem responder no mesmo ritmo de processamento do programa, como fazem estas últimas.

A memória RAM, ao contrário da memória EEPROM, é subdividida em bancos de dados, enumerados a partir de 0, onde cada banco ocupa, no máximo, 128 *bytes*. No caso dos PICs da família 16F estudados, o 16F84, embora disponibilize menos de 128 *bytes*, tem sua memória RAM mapeada ocupando dois bancos (no caso, o banco 0 e o banco 1), como mostrado na Figura 1, e o 16F628, embora disponibilize menos de 256 *bytes*, tem sua memória RAM mapeada ocupando quatro bancos (no caso, do banco 0 ao banco 3), como mostrado na Figura 2, sendo que apenas os dois primeiros são efetivamente utilizados. Já o 16F877, cuja RAM apresenta mais de 300 *bytes*, ocupa efetivamente os quatro bancos, como mostrado na Figura 3.

As finalidades usuais dos vários registradores de Propósitos Especiais, mostrados nas figuras 1 a 3 e específicos a cada microcontrolador, serão vistas durante a semana de desenvolvimento de projetos, provavelmente, a última.

Um endereço importante, a ser memorizado nas organizações de memória apresentadas nas figuras 1 a 3, é o endereço 0Ch para o 16F84 e o endereço 20h para os 16F628 e 16F877. É a partir desses endereços que o usuário pode armazenar suas variáveis de programa.

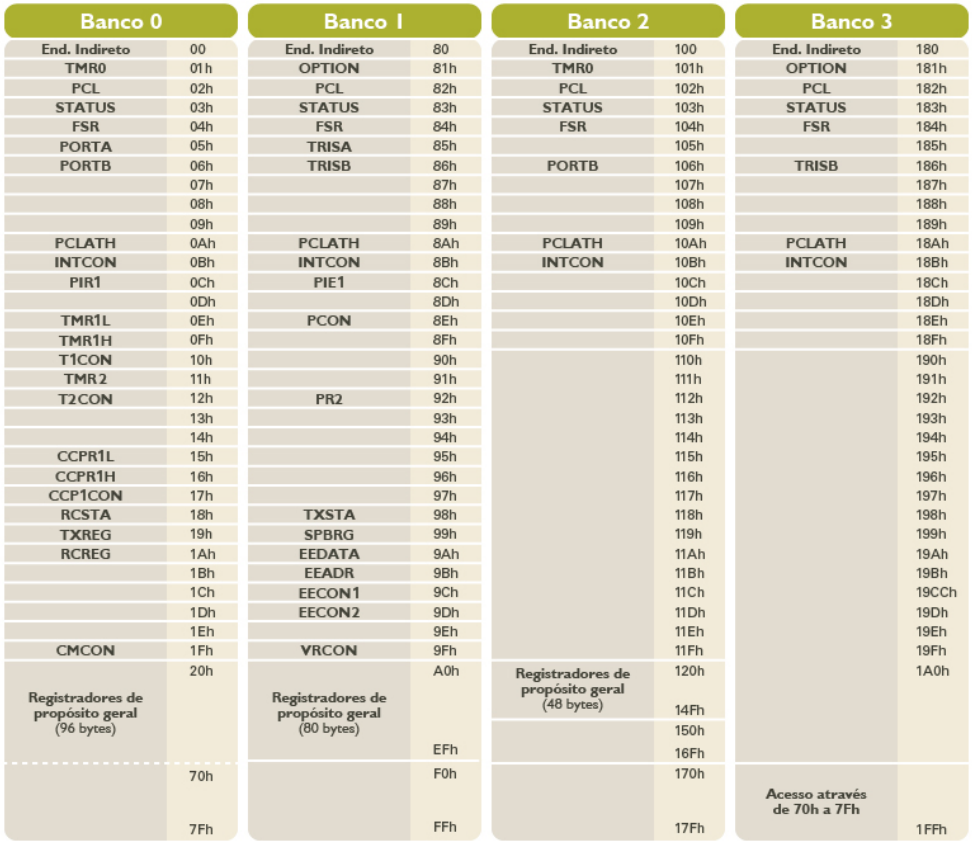
Certa dificuldade nesse tipo de organização de memória por bancos é que para ler ou escrever em um registro ou em uma dada posição de memória deve-se estar posicionado no respectivo banco. Para evitar constantes mudanças de banco, a Microchip espelha, em todos os bancos, seus principais registros e algumas áreas de uso do usuário.

Durante as tarefas práticas de programação, será mostrado como se deve proceder para garantir um posicionamento correto de banco.

Figura 01 - Organização da memória RAM de dados do 16F84.

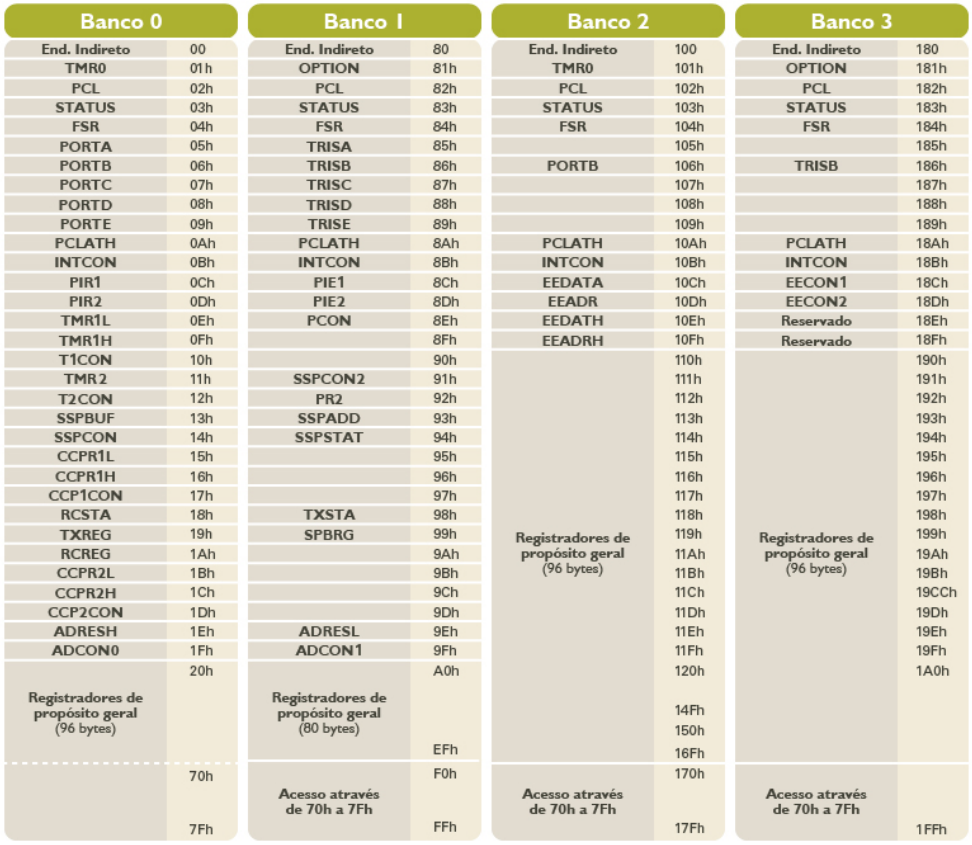
Endereço	Banco 0	Banco 1	Endereço
00	End. Indireto	End. Indireto	80
01	TMR0	OPTIUM_REG	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORTA	TRISA	85
06	PORTB	TRISB	86
07	Reservado	Reservado	87
08	EEDATA	EECON1	88
09	EEADR	EECON2	89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C 4F	68 RPGs	Mapeamento idêntico ao banco 0	8C CF
50 7F	Área não disponível	Área não disponível	D0 FF

Figura 02 - Organização da memória RAM de dados do 16F628.



Banco 0		Banco 1		Banco 2		Banco 3	
End. Indireto	00	End. Indireto	80	End. Indireto	100	End. Indireto	180
TMR0	01h	OPTION	81h	TMR0	101h	OPTION	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
	07h		87h		107h		187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch		10Ch		18Ch
	0Dh		8Dh		10Dh		18Dh
TMR1L	0Eh	PCON	8Eh		10Eh		18Eh
TMR1H	0Fh		8Fh		10Fh		18Fh
T1CON	10h		90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
	13h		93h		113h		193h
	14h		94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah	EEDATA	9Ah		11Ah		19Ah
	1Bh	EEADR	9Bh		11Bh		19Bh
	1Ch	EECON1	9Ch		11Ch		19CCh
	1Dh	EECON2	9Dh		11Dh		19Dh
	1Eh		9Eh		11Eh		19Eh
CMCON	1Fh	VRCON	9Fh		11Fh		19Fh
Registradores de propósito geral (96 bytes)	20h	Registradores de propósito geral (80 bytes)	A0h	Registradores de propósito geral (48 bytes)	120h		1A0h
			EFh		14Fh		
					150h		
					16Fh		
	70h		F0h		170h	Acesso através de 70h a 7Fh	
	7Fh		FFh		17Fh		1FFh

Figura 03 - Organização da memória RAM de dados do 16F877.



Banco 0		Banco 1		Banco 2		Banco 3	
End. Indireto	00	End. Indireto	80	End. Indireto	100	End. Indireto	180
TMR0	01h	OPTION	81h	TMR0	101h	OPTION	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD	08h	TRISD	88h		108h		188h
PORTE	09h	TRISE	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reservado	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reservado	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h	Registradores de propósito geral (96 bytes)	119h	Registradores de propósito geral (96 bytes)	199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19CCh
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
Registradores de propósito geral (96 bytes)	20h	Registradores de propósito geral (80 bytes)	A0h		120h		1A0h
			EFh		14Fh 150h 16Fh		
	70h		F0h		170h		
	7Fh	Acesso através de 70h a 7Fh	FFh	Acesso através de 70h a 7Fh	17Fh	Acesso através de 70h a 7Fh	1FFh

Atividade 01

1. Observe os mapeamentos da memória de dados dos três microcontroladores apresentados e determine quantas posições de memória contínua podem ser usadas na RAM para armazenamento de variáveis.

Análise de blocos funcionais comuns às famílias 16F e 18F

Nos dois itens seguintes, serão estudados detalhes construtivos e de funcionamento do Timer 0 e das portas A e B, comuns a todos os microcontroladores da família PIC 16F e 18F. Durante todo o restante deste texto, os FSRs usados na manipulação dos blocos funcionais em análise serão referenciados.

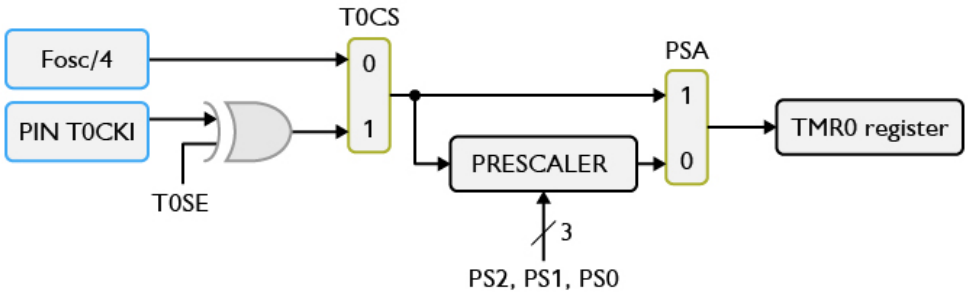
O Timer 0

O registrador Timer 0, referenciado por TMR0, é um temporizador/contador de 8 *bits* cujo conteúdo é continuamente incrementado, podendo, no entanto, ter programável a frequência e o valor de início de contagem.

Se, por exemplo, escrevermos nele o valor 10, após um tempo de 4 ciclos de relógio, o conteúdo do TMR0 começa a ser incrementado de 1, com frequência constante e independente da execução do resto do programa. Uma vez atingido o valor 255, será zerado automaticamente retornando, então, a contagem a 0 e não ao valor inicialmente imposto.

Na Figura 4 está representada a estrutura interna do PIC que determina o funcionamento do registro TMR0.

Figura 04 - Estrutura interna do PIC que determina o funcionamento do registro TMR0.



O bloco Fosc/4 e T0CKI representam as duas possíveis fontes de sinal para o contador TMR0. Fosc/4 é um sinal gerado internamente no PIC a partir do circuito de *clock* e dividida por quatro, enquanto T0CKI é um sinal gerado de um eventual circuito externo e aplicado ao pino 3 (T0CKI).

Os blocos T0CS e PSA são dois comutadores de sinal, os quais selecionam o sinal de entrada com base no valor dos *bits* T0CS e PSA do registrador OPTION.

A presença da porta XOR na entrada TOCKI do PIC permite que o *bit* TOSE do OPTION defina se o TMR0 será incrementado na descida (TOSE=1) ou na subida do pulso (TOSE=0) do sinal externo aplicado.

O bloco PRESCALER é um divisor programável de 8 *bits* que permite modificar a frequência do sinal aplicado ao TMR0 e, assim, poder aumentar o tempo de contagem do TMR0 com relação a sua cadência básica.

Por exemplo, na Figura 5, admitindo TOCS = 0 e PSA = 1, o TMR0 contará numa frequência Fosc/4 e, na Figura 6, admitindo TOCS = 0 e PSA = 0, o TMR0 contará numa frequência Fosc/4 dividida ainda por um valor imposto pelo Prescaler, de acordo com os valores dos *bits* PS2, PS1 e PS0 do registrador OPTION_REG (ou OPTION simplesmente), segundo a Tabela 1.

Figura 05 - Funcionamento do TMR0 com T0CS = 0 e PSA = 1.

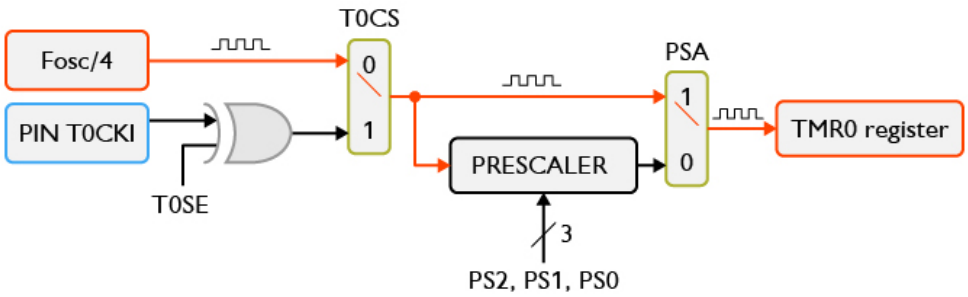
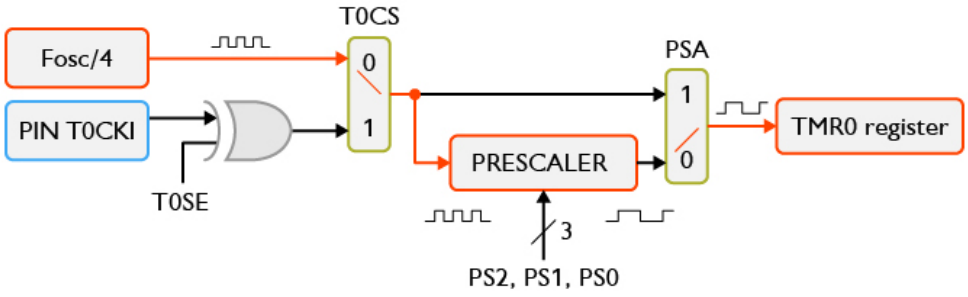


Figura 06 - Funcionamento do TMR0 com T0CS = 0 e PSA = 0.



O Prescaler é um contador de 8 *bits* e está disponível como um *prescaler* para o TMR0 e como um *postscaler* para o Watchdog (referenciado por WDT). Por simplicidade, é sempre referido como um *prescaler*. O uso do *prescaler* é excludente. Isso significa que enquanto estiver sendo usado com o TMR0, não poderá ser usado com o WDT ou vice-versa. Tal característica, certamente, limita a versatilidade do PIC.

PS2, PS1,PS0	TMR0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Tabela 1 – Configuração do Prescaler para o TMR0 e para o Watchdog (WDT)

A definição do uso do *prescaler,* se com o TMR0 ou com o WDT, é feita por *software*, atuando-se no *bit* PSA do registrador OPTION (se 0, o *prescaler* será aplicado ao TMR0). De forma idêntica, a definição do tempo de *prescaler* para o TMR0 ou para o WDT é feita por *software*, atuando-se nos *bits* PS0, PS1 e PS2 do registrador OPTION, de acordo com a Tabela 1 já referenciada.

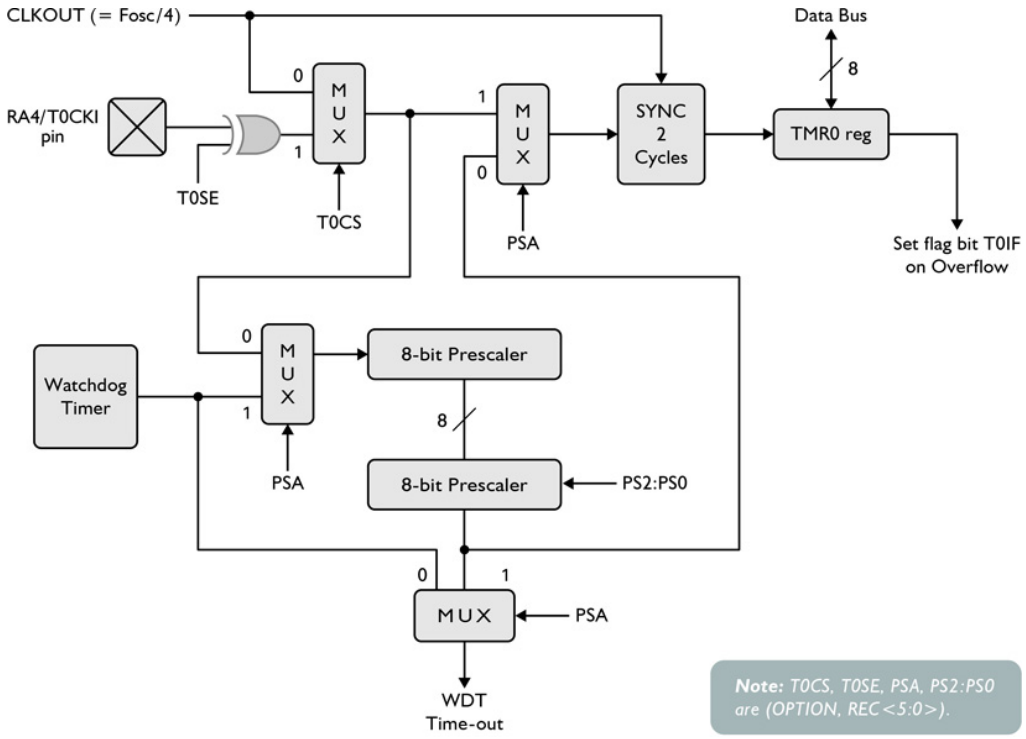
Sem *prescaler*, o TMR0 fica configurado na escala de 1:1 e "estoura" sua contagem máxima, por exemplo, em aproximadamente 72 ms para uma frequência de *clock* de 4 MHz.

Sem *prescaler*, o WDT é incrementado automaticamente e "estoura" a cada 18 ms para uma mesma frequência de *clock* de 4 MHz.

Um diagrama completo do sistema TMR0/WDT/PRESCALER pode ser visto na Figura 7.

Nas linhas 17 e 18 da Tabela 1 (Características gerais dos microprocessadores 16F84, 16F286 e 16F877) apresentada na aula 3 (revise), são especificados mais dois temporizadores, além do TMR0, que podem ser usados em aplicações gerais que exijam contagem de tempo, sendo um de 8 *bits*, o Timer 2 (que conta como o TMR0 até 0FFh) e outro de 16 *bits* (o Timer 1 que conta até 0FFFFh). Esses temporizadores se apresentam muito relacionados, ao contrário do TMR0, com aplicações que envolvem unidades periféricas, as quais já foram referenciadas e que estudaremos em algumas aplicações. Uma coisa é certa: para utilizá-los, torna-se necessário também um estudo mais detalhado, como o que foi feito neste item com o TMR0.

Figura 07 - Diagrama completo do sistema TMR0/WDT/PRESCALER.



As Portas A e B

Os PIC 16F84 e 16F628 dispõem de um total de 13 linhas de E/S organizadas em duas portas A e B, denominadas de PORTA e PORTB, comuns, também, ao 16F877 o qual disponibiliza mais três portas: a PORTC, a PORTD e a PORTE, todas de 8 *bits*.

A Porta A dispõe de 5 linhas configuráveis tanto para entrada como para saída de dados, identificadas por RA0 a RA4.

A Porta B dispõe de 8 linhas também configuráveis tanto para entrada como para saída de dados, identificadas por RB0 a RB7. Alguns desses pinos são também multiplexados com alguma outra função periférica alternativa desses dispositivos.

Para controle das linhas de E/S das portas A e B, o PIC dispõe de quatro registros internos que são: TRISA e PORTA, para a porta A, e TRISB e PORTB, para a porta B.

Os registros TRISA e TRISB determinam o funcionamento de uma linha como entrada ou como saída e os registros PORTA e PORTB armazenam valores de saída ou de entrada da porta. Todos os *bits* contidos nos registros mencionados correspondem a uma única linha de E/S. Ou seja, o *bit* 0 dos registradores PORTA e TRISA correspondem à linha RA0, o *bit* 1 à linha RA1 e assim por diante.

Se o *bit* 0 do registrador TRISA for colocado em zero, a linha RA0 será configurada como linha de saída e o valor do *bit* 0 do registro PORTA determinará o seu estado lógico. Da mesma forma, se o *bit* 0 do registrador TRISA for colocado em um, a linha RA0 estará configurada como linha de entrada e o valor lógico presente na linha externa RA0 se refletirá no *bit* 0 do registrador PORTA.

Após o acionamento do Reset, todos os pinos de E/S são configurados como saída e assumem valor “0”.

Todas as portas, embora pareçam ter características idênticas, apresentam algumas diferençazinhas que, se não assimiladas, poderão levar o circuito a um mau funcionamento.

Essas diferenças podem ser destacadas por meio dos esquemáticos apresentados nas figuras: 8a (representativa dos pinos RA0, RA1, RA2 e RA3); 8b (representativa do pino RA4); 9a (representativa dos pinos RB0, RB1, RB2 e RB3) e 9b (representativa dos pinos RB4, RB5, RB6 e RB7).

Figura 08 - (a) Esquemático: pinos RA0, RA1, RA2 e RA3; (b) Esquemático: pino RA4.

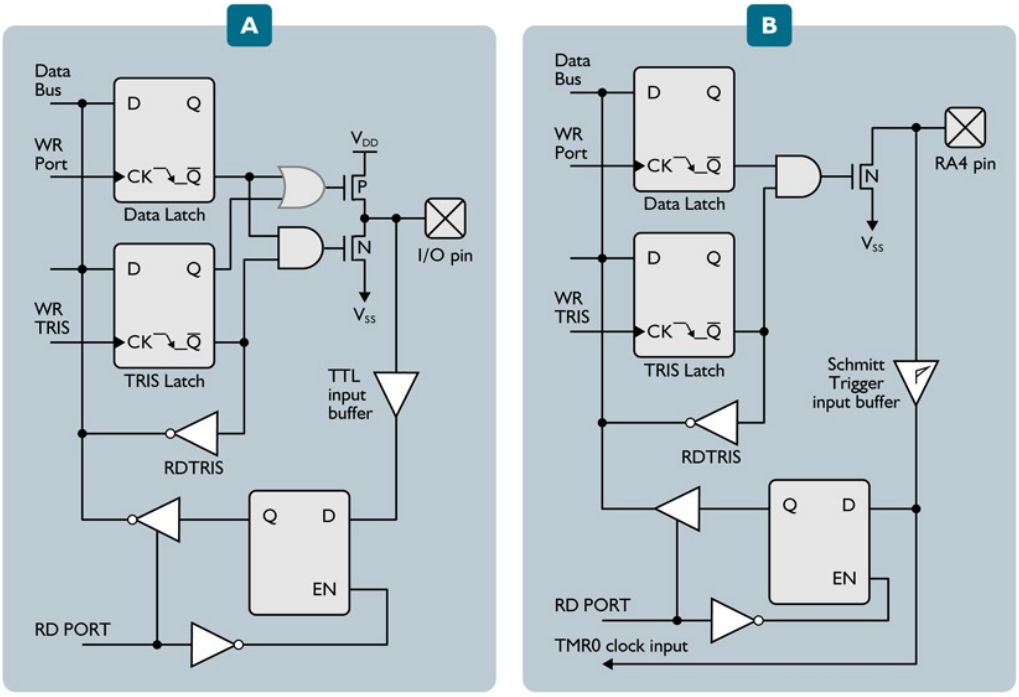
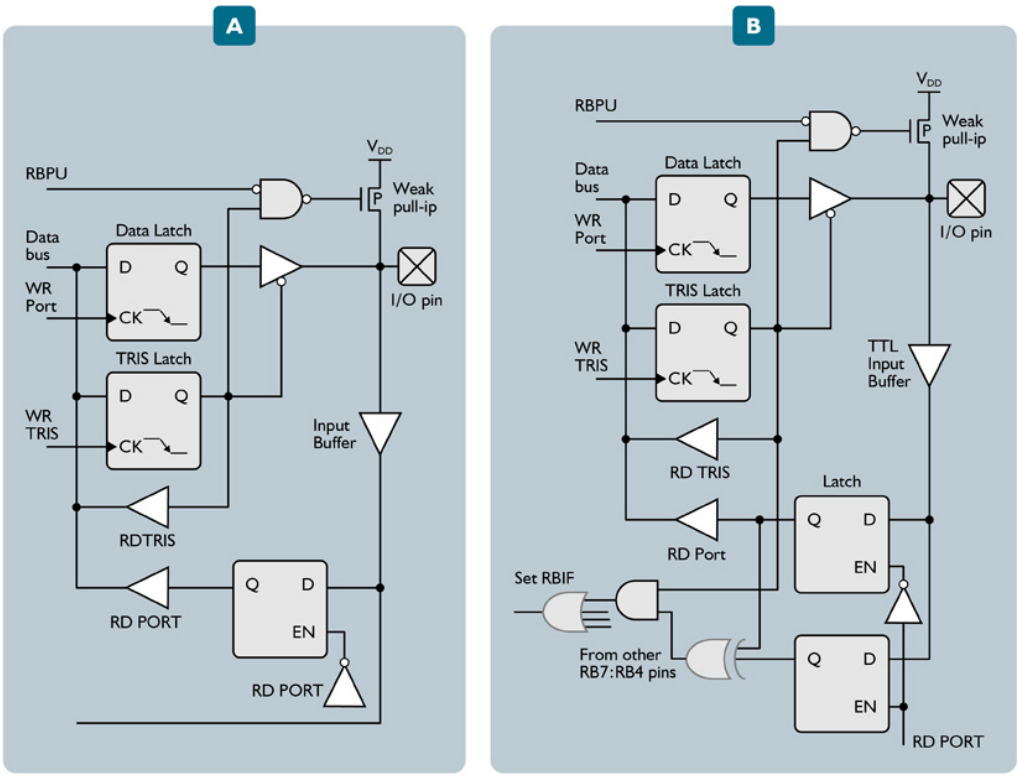


Figura 09 - Esquemático: pinos RB0, RB1, RB2 e RB3; (b) Esquemático: pinos RB4, RB5, RB6 e RB7.



Atividade 02

1. Observe, atentamente, os esquemáticos das portas A e B do 16F84, apresentados nas Figuras 8a, 8b, 9a e 9b, destacando as diferenças encontradas entre os esquemáticos mostrados nas Figuras 8b, 9a e 9b com relação ao apresentado na Figura 8a. O que caracteriza ou em que influencia cada uma delas no funcionamento de um PIC com essas características?.

   Veja a folha de especificação do 16F84 (*datasheet*) disponíveis no site da Microchip e, por meio deles, busque essas respostas.

Uma rápida introdução às principais unidades periféricas dos PICs 16F

Já na aula 3, foi colocado que alguns PICs agregam unidades que possibilitam fazer:

- Comunicação paralela de dados.
- Comunicação serial de dados em vários padrões.
- Conversão de sinais analógicos para digitais.
- Captura, comparação e geração de sinais digitais.

A partir deste momento, daremos uma breve explicação do que é cada uma dessas funcionalidades e quais os blocos e os principias registros envolvidos em sua realização. Como nem todos os PICs têm todas as funcionalidades presentes, tomaremos como referência para essa análise o 16F877.

Comunicação paralela de dados

A grande maioria dos microcontroladores PIC apresenta como opção para comunicação de dados paralelos uma unidade PSP (*Paralell Slave Port* – porta escrava paralela). A PSP consiste em uma porta paralela bidirecional de dados de 8 *bits* (envio e recepção). A denominação escrava prende-se ao fato de que o controle da comunicação pela PSP é feito, normalmente, pelo periférico externo envolvido na comunicação dos dados ou, em última instância, por um *software* aplicativo dedicado, técnica pouco utilizada, a não ser que o PIC seja dedicado a fazer única e exclusivamente o monitoramento da comunicação paralela de/para outro dispositivo.

Quando habilitado para operar com uma porta PSP controlada externamente por outro dispositivo (o que é feito através do *bit* PSPMODE do registrador TRISE), os 8 *bits* do PORTD passam a operar como uma porta paralela da largura de um *byte*,

sob controle dos pinos RE0 (RD – *read*), RE1 (WR – *write*) e RE2 (CS – *chip select*), ativos em nível baixo.

Quando operando no modo PSP, um dispositivo mestre externo pode ler dados através do PORT D (CS = 0, RD = 0 e WR = 1) e escrever dados através do PORTD (CS = 0, WR = 0 e RD = 1). Essas operações podem ser sincronizadas através de *flags* de *status* ou por pedidos de interrupção de programa, se habilitados.

Comunicação serial de dados

Normalmente, os microcontroladores PIC, como o 16F877, apresentam duas unidades periféricas de comunicação serial de dados: a USART e a SSP.

A USART é um Transmissor e Receptor Universal Síncrono e Assíncrono, que permite realizar comunicações seriais síncronas ou assíncronas, embora seu uso mais frequente seja em comunicações assíncronas, usando, neste caso, o padrão de comunicação conhecido como RS 232 C ou EIA 232 C.

No padrão RS 232 C, a linha de comunicação permanece em silêncio (nível alto) até que um *bit* de partida (*start bit*) vai baixo, indicando o início de uma transmissão de um dado de 7 ou 8 *bits*. Após a transmissão do dado, pode ser enviado ou não um *bit* de paridade e, finalizando, é enviado um *bit* de parada (*stop bit*). Durante as práticas, trataremos mais sobre esse padrão de comunicação, já que é o mais simples meio de comunicação de dados entre um transmissor e um receptor de dados.

Uma operação com a USART pode envolver vários registradores: os TXSTA e RXSTA, que controlam a configuração e as operações do transmissor; o SPBRG (*Baud Rate Generator Register*), responsável pelas configurações e pela definição da taxa de transmissão de dados; e os TXREG e RXREG, que armazenam, respectivamente, o dado a ser transmitido ou o dado recebido numa transmissão.

As operações de transmissão e de recepção serial de dados também podem ser sincronizadas por meio de *flags* de *status* ou por pedidos de interrupção de programa, se habilitados, e os pinos designados para essas transmissões são o RC6

(TX/CK) e o RC7 (RX/DT). As designações TX e RX são para transmissões assíncronas e CK e DT são para transmissões síncronas.

A SSP pode operar em dois modos de comunicação: o I2C (*Inter Integrated Communication*) e o SPI (*Serial Peripheral Interface*). O I*2*C é, frequentemente, utilizado para comunicação entre o microcontrolador e chips periféricos de memória e de relógio de tempo real (RTC – *Real Time clock*) que exigem um controle de comunicação mais seguro entre periféricos mestres e escravos e o SPI, por sua facilidade operacional, é frequentemente utilizado em aplicações que exigem controles de transmissão mais simples sob controle único de um mestre e um único escravo.

Captura, comparação e geração de PWM

Esses três processos são executados por um módulo chamado de CCP (Capture/Compare/PWM). No caso do 16F877, existem dois módulos CCP (CCP1 e CCP2) de funcionalidades idênticas. Doravante, para expressar características dos dois módulos, faremos referência a um módulo genérico CCPx.

Um módulo CCPx permite que se façam medições precisas de períodos de sinais digitais (usando sua função de capture) bem como que sejam gerados sinais de largura de pulso programável (sinais modulados por largura de pulsos ou PWM – *Pulse Width Modulation*).

Em essência, um CCPx nos modos capture/compare se constitui de três blocos, sendo um temporizador (que é o Timer 1), um circuito lógico de captura/comparação e um conjunto de dois registradores: o CCPRxH e o CCPRxL, utilizados para armazenar o valor em um processo de captura ou de comparação.

No modo captura, a largura de um pulso, o período de um sinal ou a sua frequência podem ser facilmente determinados após duas capturas sucessivas de subida ou de descida do sinal em análise.

No modo comparação, o valor do Timer 1 é, constantemente, comparado com o valor armazenado em CCPRxH e CCPRxL. Ao se verificar uma igualdade, um evento de comparação é disparado.

Nos dois casos, de uma captura ou de uma comparação, é possível a geração, se habilitada, de uma interrupção de programa.

No modo PWM, o temporizador usado é o Timer 2, concatenado com dois *bits* extras para uma resolução de 10 *bits*. Nesse modo, um registrador adicional é também usado, o PR2. Através de duas comparações é possível definir a largura do pulso de saída (definida pela comparação do Timer 2 com um valor de ciclo de trabalho – *duty cycle*, definido pelos valores em CCPRxH e CCPRxL) e o período do sinal gerado pela comparação do Timer 2 com o valor em PR2.

Conversão Analógica/Digital

Um conversor analógico digital ou simplesmente ADC ou A/D tem como função converter níveis de tensão analógica (0 V a 5 V, por exemplo) para um valor digital que o represente. Quanto maior a quantidade de *bits* usada na conversão, melhor será a aproximação representativa do valor analógico nesse caso, diz-se: melhor será sua resolução. Por exemplo, para converter uma tensão analógica que varia de 0 V a 5 V usando 8 *bits* teremos 256 valores digitais. Usando 10 *bits*, teremos 1024 valores digitais para representar tensões na mesma faixa de 0 V a 5 V. Ou seja, no segundo caso, a resolução usada, de 10 *bits*, é bem melhor que a usada na primeira (de 8 *bits*). A faixa de valores digitais, representativos da faixa analógica, variará de 0 a $2^n - 1$ (onde n é a quantidade de *bits* ou a resolução usada na conversão).

No caso do PIC 16F877, o conversor pode utilizar qualquer uma dessas duas resoluções, 8 ou 10 *bits*. Essa definição é programável pelo usuário.

Uma particularidade do conversor A/D presente no 16F877 é a quantidade de entradas analógicas disponíveis: no caso 8. Como só existe um conversor A/D, apenas uma entrada analógica pode ser amostrada por vez. Em regime, todas podem ser amostradas (uma por vez) e convertidas no seu equivalente digital.

No processo de conversão, uma entrada analógica é amostrada durante um tempo preestabelecido, tempo esse necessário para que um capacitor interno possa receber como carga final, o valor da tensão lida. Durante a carga do capacitor, um contador é disparado e conta até a carga do capacitor chegar ao valor da tensão presente na entrada analógica, momento em que o valor do contador,

representativo da tensão analógica lida, é armazenado ou transferido para outros registros envolvidos na conversão. Após esse processo de transferência, o capacitor passa por uma fase de descarga, o contador é zerado e pode-se iniciar outra amostra de tensão numa das 8 entradas analógicas (inclusive da mesma anteriormente lida, se assim for desejável).

Os tempos envolvidos numa conversão A/D dependem de vários fatores e são bem definidos pelo fabricante. Para se ter uma boa conversão, cabe ao projetista cumprir as determinações expostas nas folhas de especificação do PIC usado. Como diz o poeta, cada caso é um caso.

Diretamente associados ao conversor A/D, têm-se quatro registradores: o ADRESH e o ADRESL, que armazenam o resultado da conversão, e o ADCON0 e ADCON1, que controlam o processo de conversão.

Numa conversão A/D, algumas escolhas importantes devem preceder o processo de conversão como, por exemplo, que pinos serão usados como entradas analógicas, qual a fonte de *clock* para o contador usado pelo conversor e até mesmo que tensões de referência serão usadas durante a conversão.

Após essa pequena análise dos módulos periféricos do PIC, deu para perceber que utilizar qualquer um deles envolve especial atenção. O ideal, e é o que faremos, é deixar que maiores detalhes e procedimentos sejam trabalhados nas aplicações práticas. O assunto, logicamente, não se esgota por aqui, se desejar, você pode adquirir muitas outras informações adicionais através de pesquisas no Google ou em documentos fornecidos pelo fabricante no site <http://www.microchip.com>. Faça uma pesquisa no Google usando as palavras chaves “microprocessadores PIC”. Navegue por alguns. Procure relacionar com as informações fornecidas neste texto.

Para concluir esta aula, já que falamos por várias vezes em interrupção de programa, vamos ver como elas se processam e quais os registros envolvidos em suas configurações e uso.

Atividade 03

1. Se o bloco conversor A/D permite 8 entradas analógicas, que dispositivo possibilita essa seleção? Mostre, graficamente, um esquemático de como isso ocorre. Quantas linhas de seleção foram utilizadas em seu esquemático?

Tratamento e tipos de interrupção

O PIC, dependendo do modelo e das unidades internas e periféricas existentes, pode chegar a possuir um total de 14 interrupções diferentes. Entretanto, todas gerarão o desvio para o mesmo vetor de interrupção (endereço 004h).

A interpretação de qual interrupção ocorreu deve ser feita pelo programador por meio da análise dos *bits* sinalizadores de requisição de interrupção, presentes em três registradores especificamente reservados para tratamento de interrupções, o PIR1, o PIR2 e o INTCON.

Essas interrupções podem ser divididas em dois grupos: as convencionais (existentes em todos os PICs) e as de periféricos, específicas ao modelo.

Interrupções convencionais:

- Interrupção de timer0.
- Interrupção externa.
- Interrupção por mudança de estado nas portas RB4 a RB7.

Interrupções relacionadas a periféricos:

- Interrupção de escrita/leitura na porta paralela (PSP).
- Interrupção de conversão A/D completada.
- Interrupção de término de recepção pela USART.
- Interrupção de buffer de transmissão da USART esvaziado.
- Interrupção de dado lido ou enviado por comunicação serial (SPI e I2C).
- Interrupção causada pelo módulo CCP1.
- Interrupção do timer1.
- Interrupção do timer2.
- Interrupção de colisão em transmissão serial I2C.
- Interrupção causada pelo módulo CCP2.
- Interrupção de fim de escrita na EEPROM de dados ou também de fim de escrita na memória de programa (caso específico do 16F877).

Os mapas de requisição e habilitação das interrupções dos três microcontroladores estudados são apresentados na Figura 10 e na Figura 11. Em cada linha das figuras é dito o nome da interrupção (por exemplo, Estouro de TMR0), o *bit* responsável por habilitação (por exemplo, T0IF ou *bit* 2 do registrador INTCON) e o *bit* afetado ou "setado" quando ocorre a interrupção (por exemplo o TOIE ou *bit* 5 do registrador INTCON). Observe que, para uma interrupção ocorrer, o GIE deve ser ativado ou "setado" através do bit 7 do registrador INTCON e, para uma interrupção de periférico ocorrer, além do GIE, o PEIE, bit 6 do INTCON deve estar habilitado ou "setado".

Figura 10 - Mapa de requisição e habilitação das interrupções do PIC16F628 e do PIC16F84.

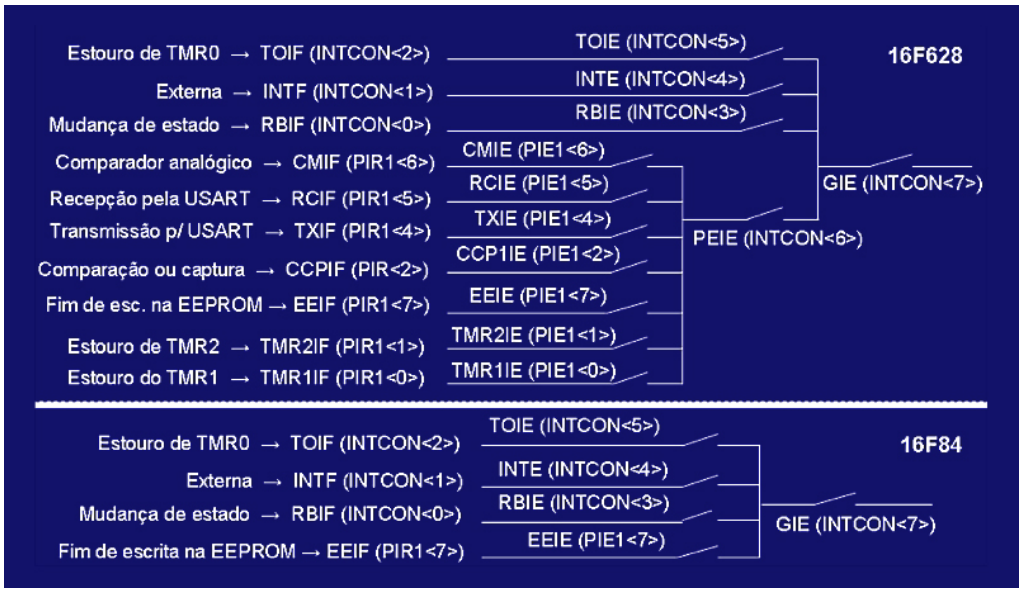
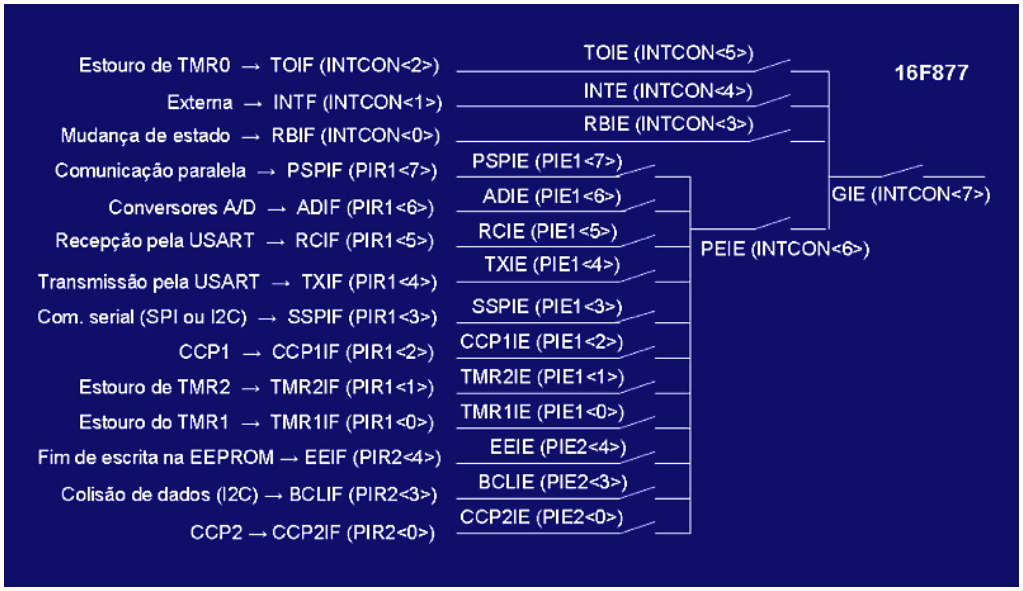


Figura 11 - Mapa de requisição e habilitação das interrupções do PIC 16F877.



Observações

- A chave GIE é, automaticamente, desligada antes de desviar para um vetor de interrupção, o que significa que duas interrupções não serão tratadas ao mesmo tempo, ou seja: uma interrupção não gerará um desvio caso outra interrupção já esteja sendo tratada.

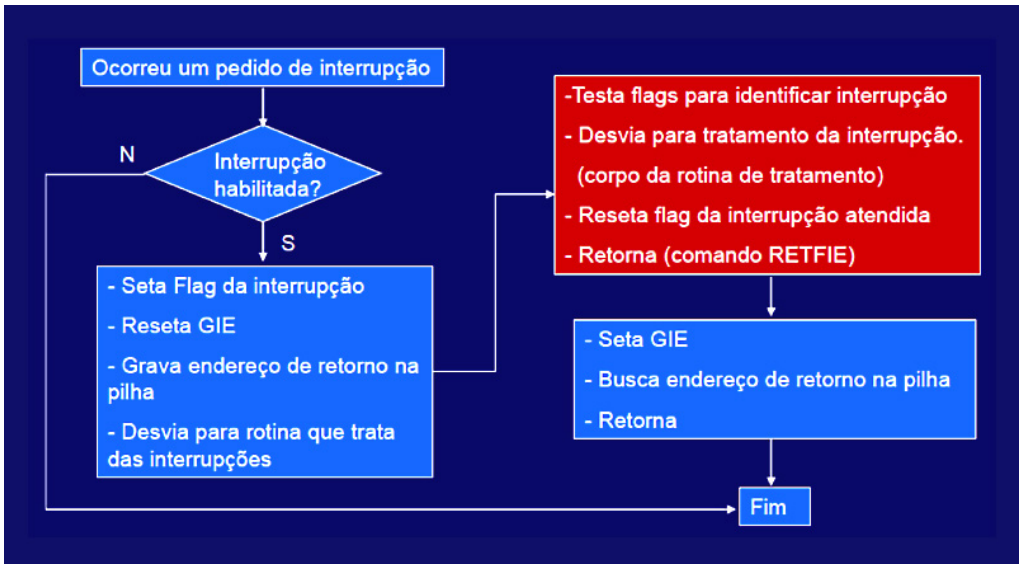
- Entretanto, o *flag* da segunda interrupção será marcado e, quando o tratamento da primeira terminar, o GIE será novamente ligado (através da instrução RETFIE) e o sistema voltará a ser desviado para o vetor de interrupção (devido ao *flag* da segunda interrupção).

- Como todas as interrupções desviam para o mesmo ponto, é necessário testar os *flags* de todas as interrupções ligadas para saber qual realmente ocorreu. Como mais de uma ação vinculada às interrupções pode acontecer ao mesmo tempo, a ordem dos testes é que determina a prioridade de tratamento.

- Excetos ADIF, TXIF e RCIF, os *flags* das interrupções não são automaticamente limpos ou “resetados” pelo sistema após o retorno da interrupção, cabe ao usuário efetuar esta operação no momento da saída da rotina de tratamento da interrupção.

O fluxograma no tratamento de uma interrupção é mostrado na Figura 12.

Figura 12 - Fluxograma no tratamento de uma interrupção dos PIC.



Atividade 04

1. Imagine que ocorreram três pedidos de interrupção, um de estouro de timer 0, um de final de escrita na EEPROM e um de recepção pela USART. Que *bits* sinalizam esses três pedidos de interrupção e que bits sinalizam se eles serão atendidos? Caso os três sejam atendidos, quem determina a prioridade de atendimento.

   Ao final do atendimento, que precaução deve ser tomada para que não se repita o atendimento do mesmo pedido de interrupção?

Resumo

Nesta aula você teve informações complementares sobre os microcontroladores PIC das famílias 16F, especificamente viu como são organizadas suas memórias de dados, aprendeu sobre dois blocos funcionais comuns a praticamente todos os PICs: o Timer 0 e as portas A e B usadas para entrada/saída de dados, teve uma pequena introdução dos blocos periféricos que permitem fazer comunicação paralela, comunicação serial, conversão A/D, captura, comparação e geração de sinais modulados em largura de pulsos e, finalizando, viu como são tratadas as interrupções de programa.

Autoavaliação

1. Explique como são organizadas as memórias de dados do 16F628.
2. Que dispositivo pode alterar a frequência de contagem do Timer0? Como ele atua e como é programado? Como atua no WDT?
3. Como um nível de tensão pode ser elevado numa porta que não apresenta estágio de saída ligado a Vcc? Em que pinos observou a presença de uma porta de E/S com essa característica?
4. O que é PWM? Como é gerado? Qual o bloco responsável pela sua geração?
5. Quantos tipos de comunicação serial são possíveis de se implementar num PIC 16f877?
6. Como se pode calcular a frequência de um sinal através do bloco capture?
7. O que entende por resolução de um conversor D/A?
8. Qual o fluxo de tratamento de uma rotina de interrupção de programa?
9. Quais as interrupções comuns a todos os PICs das famílias 16F e 18F?

Referências

BATES, P. Martin. **Programming 8 bit PIC microcontrollers in C with interactive hardware simulation**. Boston: Newnes. 2008.

MIYADAIRA, Alberto Noboru. **Microcontyroladores PIC 18**: aprenda a programar em Linguagem C. São Paulo: Editora Érica, 2009.

PEREIRA, Fábio. **PIC 18 Detalhado**: Hardware e Software. São Paulo: Editora Érica, 2010.

REESE, Robert B. **Microprocessors from Assembly Language to C using the Pic18fxx2**. Hingham, Massachusetts: Da Vinci Engineering Press, 2005.