DEPARTAMENTO DE ELECTROTECNIA

MICROCONTROLADORES

João Paulo Baptista

Curso de Engenharia Electrotécnica
Electrónica e Computadores

Sistemas Digitais / 2001

MICROCONTROLADORES

1 Arquitectura Base de um Computador

O projecto de qualquer computador pode ser conseguido através da implementação de 3 blocos funcionais, interligados entre si, como se representa na fig.1.

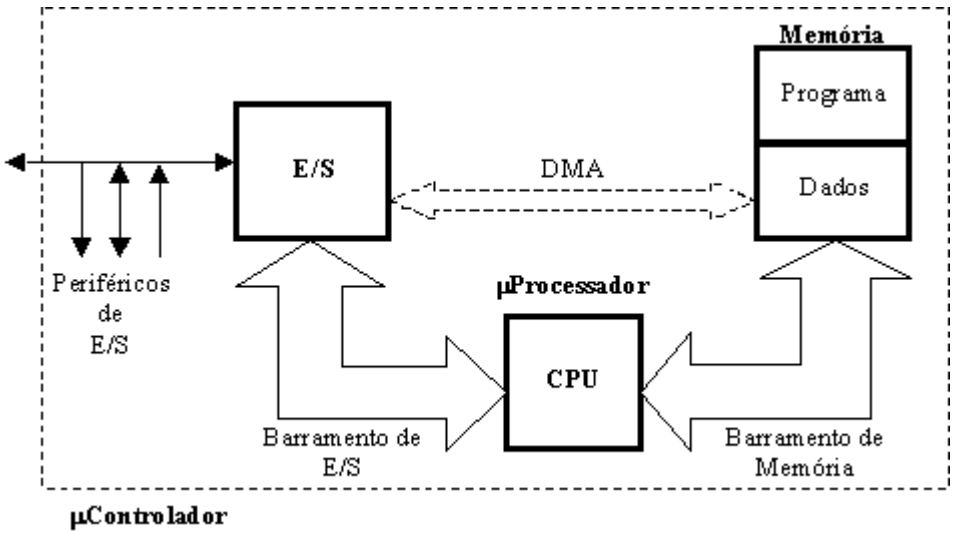


Figura 1 – Arquitectura base de um computador

1.1 CPU

O CPU (Unidade de Processamento Central) é o responsável pela correcta execução das instruções de um programa, efectuando as transferências de dados e as operações lógicas e aritméticas implícitas nessas instruções. Este CPU é constituído por:

- **Unidade de Controlo**, que gera todos os sinais necessários à execução de cada uma das instruções incluídas no programa.
- **Processador**
  - **ALU** – Unidade Lógica e Aritmética, onde se encontram implementadas as unidades que permitem efectuar as operações lógicas e aritméticas;
  - **Registos**, utilizados para o armazenamento temporário de operandos e resultados de operações;
- **Controlo de Barramentos**, permitindo gerir o tráfego de informação para o acesso aos dispositivos exteriores.

Num computador, as funções do CPU são desempenhadas normalmente por um microprocessador (p. ex. Z80).

1.2 Memória

Este bloco, constituído por dispositivos de memória (ROM, EPROM, E^2PROM, FLASH EPROM, RAM, NVRAM, etc.) permite o armazenamento de instruções de programa e dados e pode ser logicamente dividida em:

- **Memória de Programa**, onde são guardadas as instruções a executar pelo CPU;
- **Memória de Dados**, onde são guardados todos os dados necessários ao processamento ou provenientes do processamento.

1.3 E/S (Entradas/Saídas)

Este bloco do computador permite a sua interligação com dispositivos periféricos para entrada ou saída de dados (teclado, monitor, impressora, sensores, conversores A/D ou D/A, etc.). Fisicamente é constituído por hardware que permite a transmissão/recepção de dados e, em alguns casos, a sua formatação.

1.4 Barramento de Memória e Barramento de E/S

Embora logicamente diferentes, normalmente estes 2 barramentos não são distintos fisicamente. São constituídos por condutores que permitem a transferência da informação entre o CPU e a memória ou os dispositivos de E/S. A constituição destes barramentos é a seguinte:

- **Barramento de Endereços**, unidireccional, onde o CPU coloca o endereço de memória ou do dispositivo de onde quer ler ou onde quer escrever (p. ex. no Z80 é constituído pelas linhas A0..A15);
- **Barramento de Dados**, bidireccional, onde circulam os dados resultantes da comunicação de ou para o CPU (p. ex. linhas D0..D7 no Z80);
- **Barramento de Controlo**, bidireccional, que contém os sinais necessários para uma correcta implementação do protocolo de comunicação (p. ex. no Z80 as linhas WR, RD, MREQ, IOREQ, etc.).

1.5 Barramento de DMA (Direct Memory Access)

Na maior parte dos computadores, a transferência de dados entre os dispositivos de E/S e a memória pode ser feita sem a intervenção directa do CPU, libertando-o assim para outras tarefas que podem ser feitas em paralelo. Esta transferência de informação, que utiliza controladores específicos, é feita através do barramento de acesso directo à memória.

1.6 MicroControladores

O aumento da capacidade de integração, permitiu aos fabricantes a inserção, num único circuito integrado, dos diferentes componentes que implementam um computador. Assim, hoje em dia é possível encontrar uma infinidade de dispositivos que contém CPU, memória, portos de entrada/saída de dados e periféricos específicos para determinadas aplicações (conversores A/D e D/A, portos de comunicação série, geradores de PWM, temporizadores e contadores, dispositivos de watchdog, etc.). A maior parte das vezes estes μControladores disponibilizam os barramentos para o exterior permitindo assim a sua expansão através da

adição de circuitos periféricos. Na fig. 1 está indicada a constituição destes dispositivos de processamento, dos quais iremos estudar em pormenor 2 exemplos típicos.

2 µControlador PIC16F84A

Trata-se de um micro controlador desenvolvido pela Microchip (www.microchip.com), cuja implementação é conseguida num integrado de 18 pinos como se mostra na fig. 2.

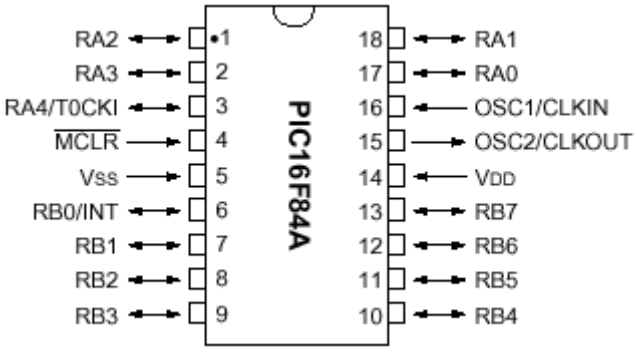


Figura 2 – O µC PIC16F84A

Neste circuito são disponibilizados para o exterior 2 portos de entrada/saída de dados – RB0..RB7 e RA0..RA4 o que permite a utilização de 13 bits de I/O, programáveis individualmente como entrada ou saída.

Através de 5 destes bits de I/O é possível aceder do exterior ao sistema de interrupções do micro controlador (RB0 e RB4..RB7).

Dado que estes circuitos possuem já um oscilador interno, a referência de oscilação é conseguida através da ligação de um cristal ou de uma malha RC nos pinos OSC1 e OSC2. Isto vai determinar, como será visto mais à frente, a velocidade de funcionamento do µC. A entrada CLKIN permite ainda que o sinal de relógio do µC seja fornecido a partir de um oscilador externo.

Uma tensão de 0V aplicada à entrada MCLR permite colocar o µC em estado de *reset*. Logo que esta tensão comute para VDD, o µC sai da situação de *reset*, começando a execução do programa a partir da instrução que se encontra no endereço 0 (vector de *reset*). Os valores com que os diferentes registos são carregados nesta operação serão apresentados nos parágrafos seguintes.

O bit RA4 pode funcionar alternativamente como entrada de um sinal de relógio para um contador crescente de 8 bits que se encontra integrado no µC. O funcionamento deste contador / temporizador (TMR0) será objecto de um estudo mais aprofundado no parágrafo 4.

2.1 Arquitectura do PIC16F84

A fig. 3 apresenta um diagrama de blocos que traduz a arquitectura deste μC.

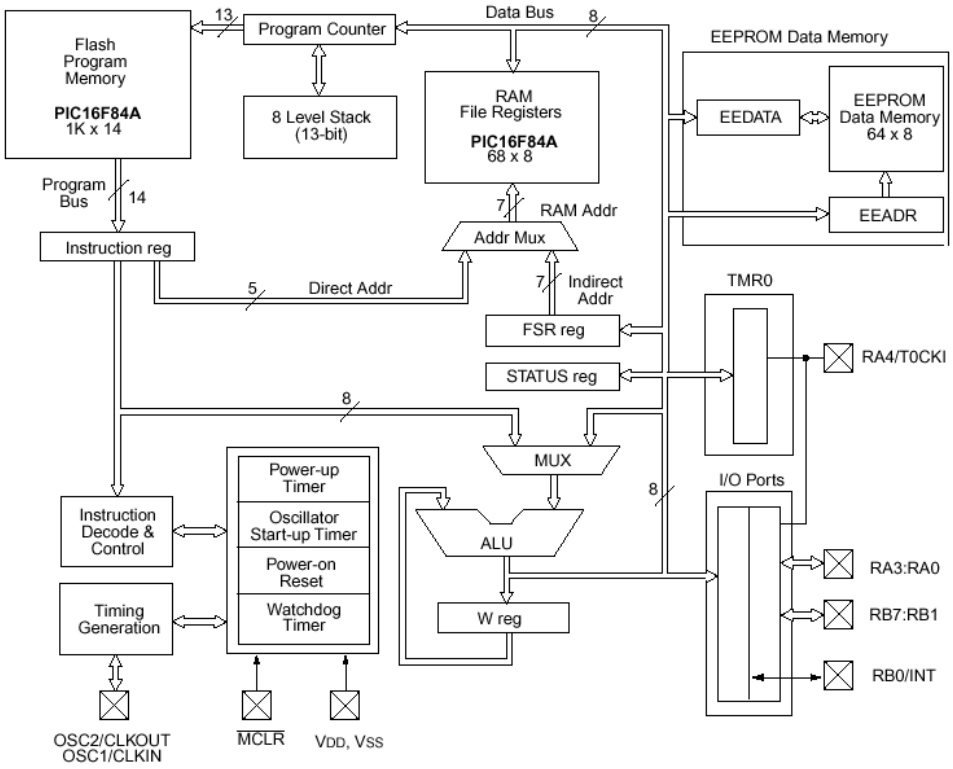


Figure 3 – Arquitectura do PIC16F84A

Neste diagrama de blocos podem ser identificados os diferentes componentes de um computador como foram apresentados no parágrafo 1. A constituição de cada um dos componentes será a seguinte:

- CPU
  - Processador – ALU, reg W, reg STATUS, reg FSR
  - Unidade de Controlo – reg de instruções, descodificador de instruções, contador de programa, stack
- Memória
  - Memória de programa E^2PROM (1K x 14 bits)
  - Memória de dados RAM (68 x 8 bits)
- Barramentos
  - barramentos internos, independentes, um para a memória de programa (14 bits de largura) e um para a memória de dados (8 bits de largura)
- Periféricos
  - portos de E/S (I/O)
  - 1 temporizador / contador de 8 bits (TMR0)
  - Memória de dados E2PROM (64 x 8 bits)
  - Temporizador 'Watchdog'

2.1.1 Arquitectura de 'Harvard'

Uma das particularidades deste μC tem a ver com o tipo de arquitectura específica normalmente denominada de arquitectura de Harvard. Neste tipo de arquitectura, a memória de programa e a memória de dados estão fisicamente separadas o mesmo acontecendo com os respectivos barramentos de interligação com o CPU. Dado que o acesso a estas memórias é feito através de barramentos diferentes, é possível a utilização da memória de dados ao mesmo tempo que é feita a pesquisa de uma instrução da memória de programa.
Ao contrário dos CPUs que utilizam uma arquitectura de Von-Newman, em que aquela distinção física não é verificada e o ciclo de funcionamento é feito através do sequenciamento de operações de 'fetch' e de execução, nos CPUs com arquitectura de Harvard a execução da instrução actual pode ser feita em simultâneo com o 'fetch' da próxima instrução. Este tipo de operação, que se esquematiza no diagrama temporal apresentado na fig. 4, permite obter uma velocidade de processamento que, teoricamente, será o dobro da obtida com um CPU com arquitectura Von-Newman.

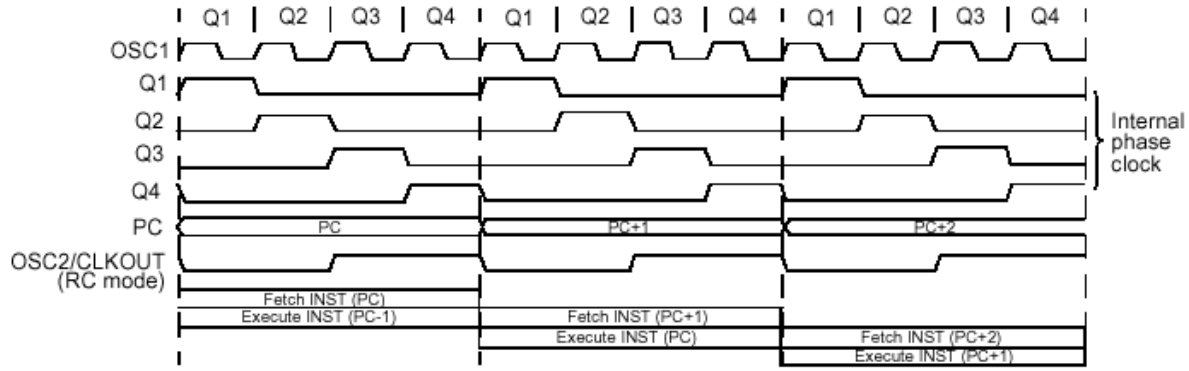


Figure 4 – Execução com sobreposição

Claro que só teoricamente é que o aumento de velocidade é de 2 vezes. Com determinado tipo de instruções, todas aquelas que provocam uma alteração do contador de programa (Program Counter), esta sobreposição do 'fetch' e da execução falha, havendo necessidade de efectuar um novo 'fetch'. Este problema é apresentado na fig. 5.

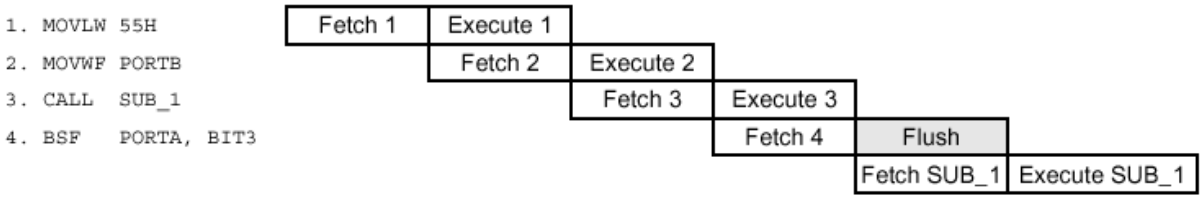


Figura 5 – Execução de instruções que alteram o PC

Quando está a ser executada a instrução 3 está em simultâneo a ser feito o 'fetch' da instrução 4. Dado que o resultado da execução de 3 é um 'salto' para uma instrução que não é a 4, a instrução pesquisada tem de ser rejeitada para dar origem à pesquisa de uma nova instrução. Podemos ver aqui que a instrução CALL (assim como outras instruções) demora o dobro do tempo a ser executada.

2.2 Memória de Programa

O PIC16F84 dispõe de um contador de programa com 13 bits capaz de endereçar 8K posições de memória de 14 bits. Neste μC só se encontra implementado 1K com endereços de 0000h a 03FFh. A estrutura desta memória é apresentada na fig. 6.
A estrutura de stack apresenta um máximo de 8 níveis o que impõe uma limitação em termos de chamadas a subrotinas dentro de outras subrotinas.
O vector de reset corresponde à posição de memória 0000h enquanto que o vector de interrupção se encontra atribuído à posição 0004h.

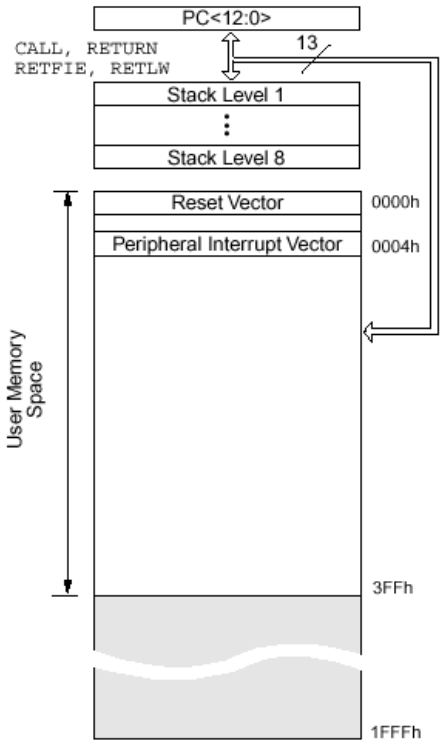


Figura 6 – Memória de programa

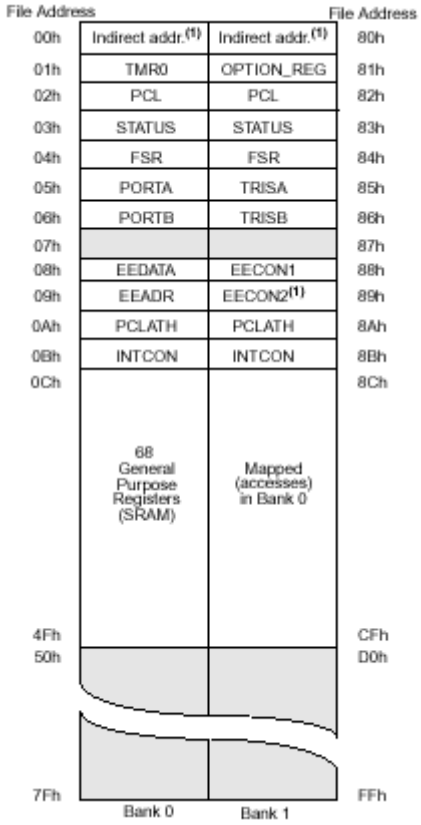


Figura 7 – Memória de dados

2.3 Memória de Dados

A memória de dados do 16F84 encontra-se dividida em duas áreas, conforme se pode ver na fig. 7. A primeira é a área correspondente aos Registos de Função Especial (SFR) enquanto que a segunda diz respeito aos Registos de Uso Geral. Os SFRs são utilizados para o controlo do μC.

Uma das particularidades desta memória é a sua divisão em dois Bancos (Banco 0 e Banco 1) sendo o acesso a cada um destes bancos controlado a partir do bit 5 do registo STATUS (RP0). De notar que, no que diz respeito aos registos de uso geral, o banco 1 destes registos encontra-se mapeado no banco 0 o que quer dizer que qualquer acesso ao banco 1 é feito na realidade sobre o banco 0.

2.3.1 Registos de Uso Geral

Cada registo de uso geral tem uma largura de 8 bits e pode ser acedido directamente ou indirectamente através do registo FSR (ver mais à frente).
Como foi dito anteriormente, os endereços do banco 1 encontram-se mapeados no banco 0. Por exemplo, o endereçamento da posição de memória 0Ch ou 8Ch acede ao mesmo registo de uso geral.

2.3.2 Registos de Função Especial

Os Registos de Função Especial (Fig.7 e Tabela1) são utilizados pelo CPU e pelos periféricos para o controlo do μC. Estes registos são implementados em RAM estática.

Os SFRs que fazem parte do núcleo do μC serão analisados nesta secção enquanto que os SFRs ligados aos dispositivos periféricos serão descritos em secções posteriores.

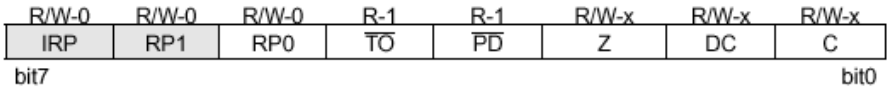
Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA (4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB (5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.

Tabela 1 – Registos de função especial (SFR)

2.3.2.1 Registo STATUS

O registo STATUS contém as flags ligadas à ALU, as flags de RESET e os bits de selecção do banco da memória de dados.



R = bit de leitura
W = bit de escrita
U = não implementado
-n = valor de reset

Bit 7	**IRP** – Bit de selecção de banco (endereçamento indirecto) Este bit não é utilizado no PIC16F84(A)
Bit 6-5	**RP1:RP0** – Bit de selecção de banco (endereçamento directo) 00 = Banco 0 (00h – 7Fh) 01 = Banco 1 (80h – FFh) No PIC16F84(A) só é utilizado o RP0
Bit 4	**$\overline{\text{TO}}$** – Bit de time-out 1 = a seguir a um power-up, a uma instrução CLRWDT, a uma instrução SLEEP 0 = ocorreu um time-out do WDT
Bit 3	**$\overline{\text{PD}}$** – Bit de power-down 1 = a seguir a um power-up ou a uma instrução CLWDT 0 = depois da execução de uma instrução de sleep
Bit 2	**Z** – flag de zero 1 = o resultado de uma operação lógica ou aritmética é zero 0 = o resultado de uma operação lógica ou aritmética é diferente de zero
Bit 1	**DC** – Digit carry / borrow 1 = carry existente do bit 3 para o bit 4 do resultado 0 = não existe carry do bit 3 para o bit 4 do resultado
Bit 0	**C** – Bit de carry / borrow 1 = existe carry do bit mais significativo do resultado 0 = não existe carry do bit mais significativo do resultado

Figura 8 – Registo STATUS (endereço 03h, 83h)

2.3.2.2 Registo OPTION_REG

Trata-se de um registo que permite escrita e leitura e que contém vários bits de controlo para a configuração do divisor de frequência (prescaler) do TMR0 / WDT, da interrupção externa INT, do TMR0 e das polarizações (pull-ups) do PORTB.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = bit de leitura
W = bit de escrita
U = não implementado
-n = valor de reset

Bit 7 **$\overline{\text{RBPU}}$** – Activação dos pull-ups internos do PORTB
1 = pull-ups internos desactivados
0 = pull-ups internos activados

Bit 6 **INTEDG** – Selecção do flanco de activação da interrupção INT
0 = interrupção no flanco descendente do sinal em RB0/INT
1 = interrupção no flanco ascendente do sinal em RB0/INT

Bit 5 **T0CS** – Selecção da fonte do sinal para o TMR0
1 = transição no pino RA4/T0CKI
0 = ciclo interno de instrução

Bit 4 **T0SE** – Selecção do flanco do sinal que activa o TMR0
1 = incrementa no flanco descendente do sinal em RA4/T0CKI
0 = incrementa no flanco ascendente do sinal em RA4/T0CKI

Bit 3 **PSA** – Bit de atribuição do prescaler (divisor de frequência)
1 = prescaler atribuído ao WDT
0 = prescaler atribuído ao TMR0

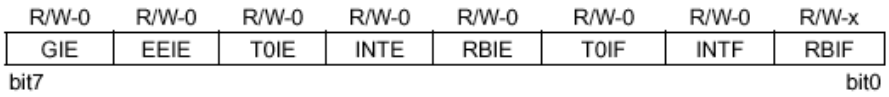
Bit 2-0 **PS2: PS0** – Programação do prescaler (divisor de frequência)

Bit Value	TMR0 Rate	WDT Rate
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

Figura 9 – Registo OPTION-REG (endereço 81h)

2.3.2.3 Registo INTCON

O registo INTCON contém os diferentes bits que permitem configurar o sistema de interrupções (enable /disable) e as flags de pedido de interrupção. Este registo pode ser lido ou escrito.



R = bit de leitura
W = bit de escrita
U = não implementado
-n = valor de reset

Bit 7	**GIE** – Enable / Disable global das interrupções 1 = Permite todas as interrupções que não se encontram mascaradas 0 = Desactiva todas as interrupções
Bit 6	**EEIE** – Bit de autorização da interrupção de fim de escrita na E^2PROM de dados 1 = permite a interrupção EE 0 = desactiva a interrupção EE
Bit 5	**T0IE** – Bit de autorização da interrupção do TMR0 1 = permite a interrupção do TMR0 0 = desactiva a interrupção do TMR0
Bit 4	**INTE** – Bit de autorização da interrupção INT 1 = permite a interrupção INT 0 = desactiva a interrupção INT
Bit 3	**RBIE** – Bit de autorização da interrupção gerada pela alteração de RB4:RB7 1 = permite a interrupção de RB4:RB7 0 = desactiva a interrupção de RB4:RB7
Bit 2	**T0IF** – Flag de pedido de interrupção do TMR0 (quando existe 'overflow') 1 = 'overflow' do TMR0 (é necessário limpar a flag por software) 0 = não há 'overflow' do TMR0
Bit 1	**INTF** – Flag de pedido de interrupção externa INT 1 = há um pedido de interrupção (é necessário limpar a flag por software) 0 = não existe pedido de interrupção INT
Bit 0	**RBIF** – Flag de pedido de interrupção por alteração de RB4:RB7 1 = há um pedido de interrupção (é necessário limpar a flag por software) 0 = não existe pedido de interrupção RB

Figura 10 – Registo INTCON (endereço 0Bh, 8Bh)

2.4 Registos PCL e PCLATH

O contador de programa (PC) especifica o endereço da instrução a ser pesquisada ('fetch') da memória de programa para ser executada. No PIC16F84 o PC tem uma largura de 13 bits, estando os 8 bits menos significativos colocados no registo PCL. Este registo pode ser lido ou escrito por software. O byte mais significativo do PC é designado por PCH e contém os bits 12:8 do contador de programa. Este registo não pode ser escrito ou lido directamente. Todas as operações que seja necessário efectuar sobre ele são feitas indirectamente sobre o registo PCLATH.

2.4.1 STACK

A estrutura de stack implementada no PIC16F84 permite a ocorrência de combinações de até 8 chamadas de subrotinas ou interrupções, permitindo armazenar os respectivos endereços de retorno. De notar que a utilização desta estrutura é da exclusiva responsabilidade da unidade de controlo do μC, não havendo por isso qualquer instrução de software que a permita manipular.

2.5 Endereçamento Indirecto: registos INDF e FSR

O registo INDF (endereço 00h / 80H) não é um registo físico. O acesso a este registo (leitura ou escrita) acede realmente ao registo cujo endereço se encontra no registo FSR. Este último registo é o ponteiro para as operações com endereçamento indirecto.

3 Portos de I/O

Alguns dos pinos dos portos de I/O apresentam funções alternativas para a utilização com alguns periféricos do μC. Geralmente, quando essas funções são utilizadas, o respectivo bit não pode ser utilizado como I/O.

3.1 Registos PORTA e TRISA

O registo PORTA implementa um porto de I/O bidireccional com 5 bits, estando ligado aos pinos exteriores do μC, RA0:RA4. O correspondente registo onde se define, para cada bit, se se trata de entrada ou saída de dados é o TRISA.

Colocando um bit do registo TRISA a 1 permite programar o respectivo bit do PORTA como entrada de dados já que, como se pode ver nas figs. 11 e 12, o correspondente andar de saída fica em estado de alta impedância. A escrita de um 0 num bit do registo TRISA programa o respectivo bit do PORTA como saída de dados (o estado do andar de saída vai depender do valor que se escreve na 'Data Latch'). De notar que, após uma operação de reset, qualquer um dos bits de I/O fica programado como entrada de dados.

A leitura do registo PORTA permite fazer a leitura do estado dos pinos exteriores enquanto que uma operação de escrita sobre o mesmo registo coloca os valores de escrita nas 'Data Latch' de saída.

O bit RA4 apresenta uma multiplexagem com o sinal de entrada para o TMR0. Como entrada de dados este bit apresenta uma entrada do tipo Schmitt Trigger e como saída é do tipo dreno aberto.

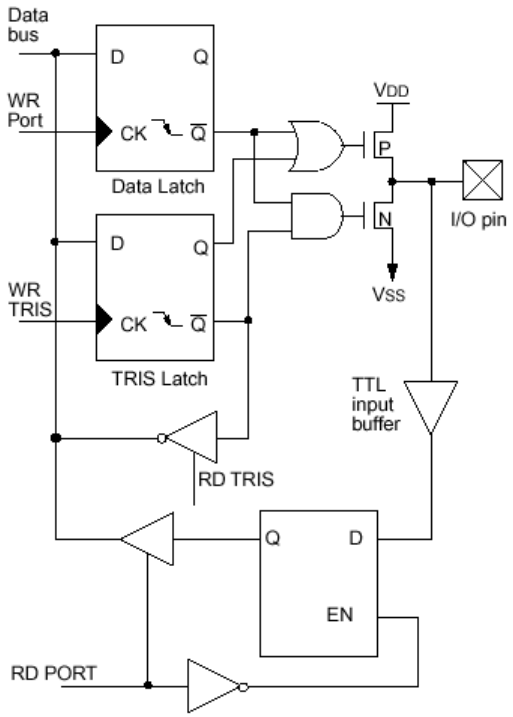


Figura 11 – RA3:RA0

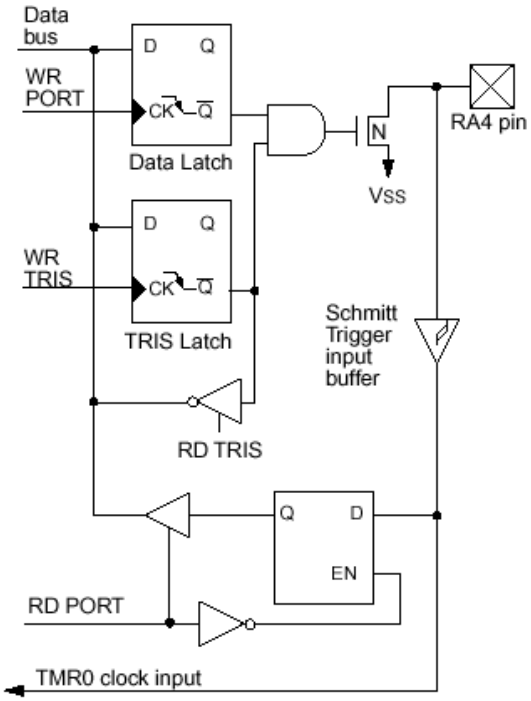


Figura 12 – RA4

Exemplo de programação do PORTA com RA0:RA1 como saída de dados e RA2:RA4 como entrada de dados:

```
clrf    PORTA           ;Escreve 0s nas 'Data Latch'
bsf     STATUS,RP0      ;Muda para o banco 1
movlw   b'00011100'     ;Palavra de programação
movwf   TRISA           ;para ser colocada no TRISA
bcf     STATUS,RP0      ;Regressa ao banco 0
```

Características de corrente para o PORTA:

- Corrente máxima fornecida por cada pino …….. I_{OH}max = 20mA
- Corrente máxima absorvida por cada pino ……..…... I_{OL}max = 25mA
- Corrente máxima total fornecida pelo PORTA ……..… ΣI_{OH}max = 50mA
- Corrente máxima total absorvida pelo PORTA ……..… ΣI_{OL}max = 80mA

3.2 Registos PORTB e TRISB

PORTB é um porto bidireccional de 8 bits que, como acontece com o PORTA, dispõe também de um registo TRISB para a configuração dos bits como entrada ou saída de dados. Nas figuras 13 e 14 representam-se os diagramas de blocos dos bits que compõe este porto.

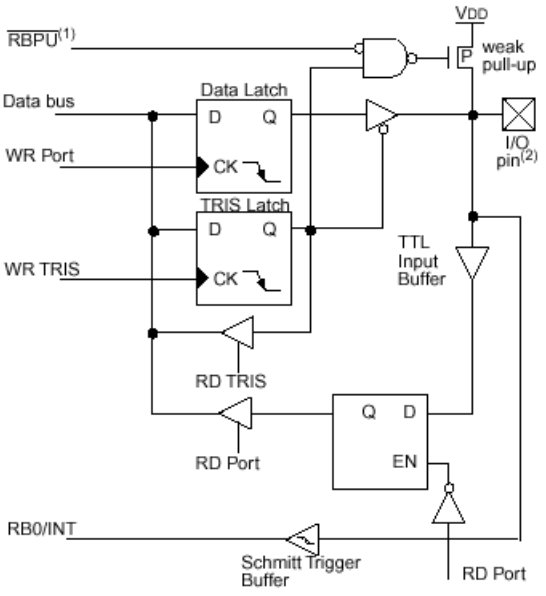


Figura 13 – RB3:RB0

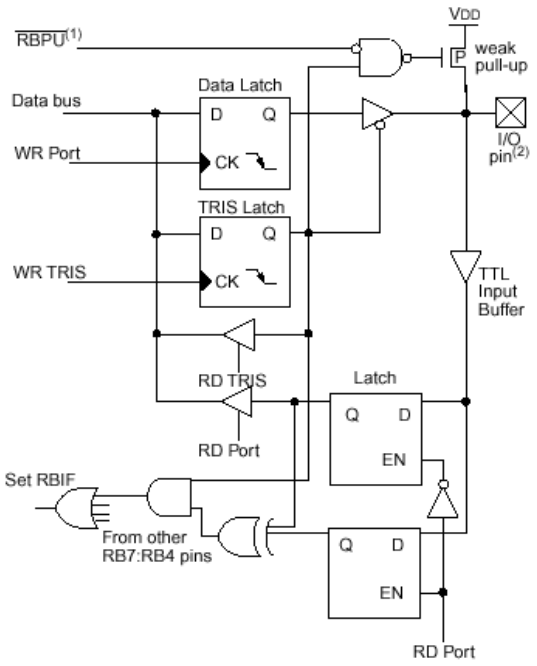


Figura 14 – RB7:RB4

Cada um dos bits do PORTB dispõe de um pull-up interno que é controlado pelo bit 7 do registo OPTION ($\overline{RBPU}$). Quando este bit é igual a 0, todos os pull-ups estão activados. A configuração de um bit como saída de dados provoca automaticamente a desactivação do respectivo pull-up. Quando é feito o power-on reset, os pull-ups são desactivados.

Quatro dos bits do PORTB, RB7:RB4, permitem gerar uma interrupção sempre que há uma mudança de estado em qualquer um deles, desde que estejam programados como entrada de dados. Os bits de entrada são permanentemente comparados com o último valor lido do porto. Desde que seja encontrada uma diferença em qualquer um desses bits, é activada a flag RBIF de pedido de interrupção. Na rotina de serviço da interrupção deve ser feita uma leitura do porto de maneira a anular a situação de diferença de bits, permitindo assim fazer o clear da flag por software.

Características de corrente para o PORTB:

- Corrente máxima fornecida por cada pino …….. $I_{OH}max = 20mA$
- Corrente máxima absorvida por cada pino ……..…... $I_{OL}max = 25mA$
- Corrente máxima total fornecida pelo PORTB ……...… $\Sigma I_{OH}max = 100mA$
- Corrente máxima total absorvida pelo PORTB ……...… $\Sigma I_{OL}max = 150mA$

4 Temporizador / Contador TMR0

O periférico TMR0 apresenta as seguintes características principais:

- Funções de temporizador ou contador de 8 bits
- Possibilidade de leitura e de escrita do registo
- Possibilidade de selecção de relógio interno ou externo
- Selecção do flanco do sinal externo que incrementa o contador
- Divisor de frequência de 8 bits programável por software
- Permite gerar uma interrupção quando se verificar uma situação de overflow (passagem de FFh a 00h)

A figura seguinte apresenta um diagrama simplificado deste periférico.

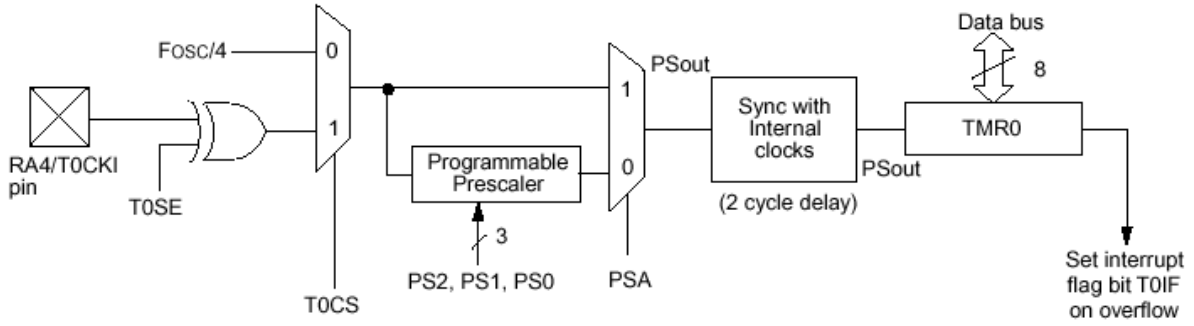


Figura 15 – Diagrama de blocos do TIMER0

4.1 Funcionamento

O Timer0 pode funcionar como temporizador ou como contador. A selecção do modo de funcionamento é feita através do bit T0CS (Timer0 Clock Sourse, bit 5 do OPTION_REG).

No modo de temporizador, T0CS=0, o registo TMR0 incrementa em cada ciclo de instrução desde que não seja utilizado o Prescaler (divisor de frequência). Se, por software, se efectuar uma escrita do registo TMR0, o incremento será inibido durante 2 ciclos de instrução.
Por exemplo, se o relógio do µC funcionar a 4MHz, e se não for utilizado o Prescaler, o incremento verifica-se de 1 em 1µS.

No modo de contador, T0CS=1, o registo TMR0 incrementa em cada flanco ascendente ou descendente do sinal presente no pino exterior RA4/T0CKI. O flanco que provoca o incremento é definido pelo bit T0SE (Timer0 Source Edge, bit 4 do OPTION_REG). Se T0SE=0 o flanco activo é o ascendente. Neste modo de funcionamento, o sinal proveniente do exterior deve obedecer a determinados requisitos que lhe permite uma sincronização com o relógio interno do µC.

4.2 Prescaler

O prescaler não é mais que um divisor de frequência programável e é constituído por um contador de 8 bits associado a um multiplexer de 8 ÷1, como se mostra na figura 16. Este prescaler é partilhado pelo Timer0 e pelo Watchdog Timer, não podendo contudo ser usado

em simultâneo pelos dois periféricos. Sendo assim, a atribuição do prescaler ao Timer0 significa que o Watchdog Timer funciona sem prescaler, e vice-versa.

O valor do prescaler não pode ser lido nem alterado por uma operação de escrita.

O bit PSA (PreScaler Assignement, bit 3 do OPTION_REG) permite fazer a atribuição do Prescaler ao Timer0 (PSA=0) ou ao Watchdog (PSA=1).

Os bits PS2:PS0 definem a taxa de divisão de frequência que será aplicada ao sinal de entrada. Quando o Prescaler está atribuído ao Timer0, podem ser utilizadas as taxas de 1:2, 1:4, …, 1:256. Para o caso do Watchdog as taxas possíveis são 1:1, 1:2, …, 1:128. Estes 3 bits encontram-se mapeados nos 3 bits menos significativos do OPTION_REG.

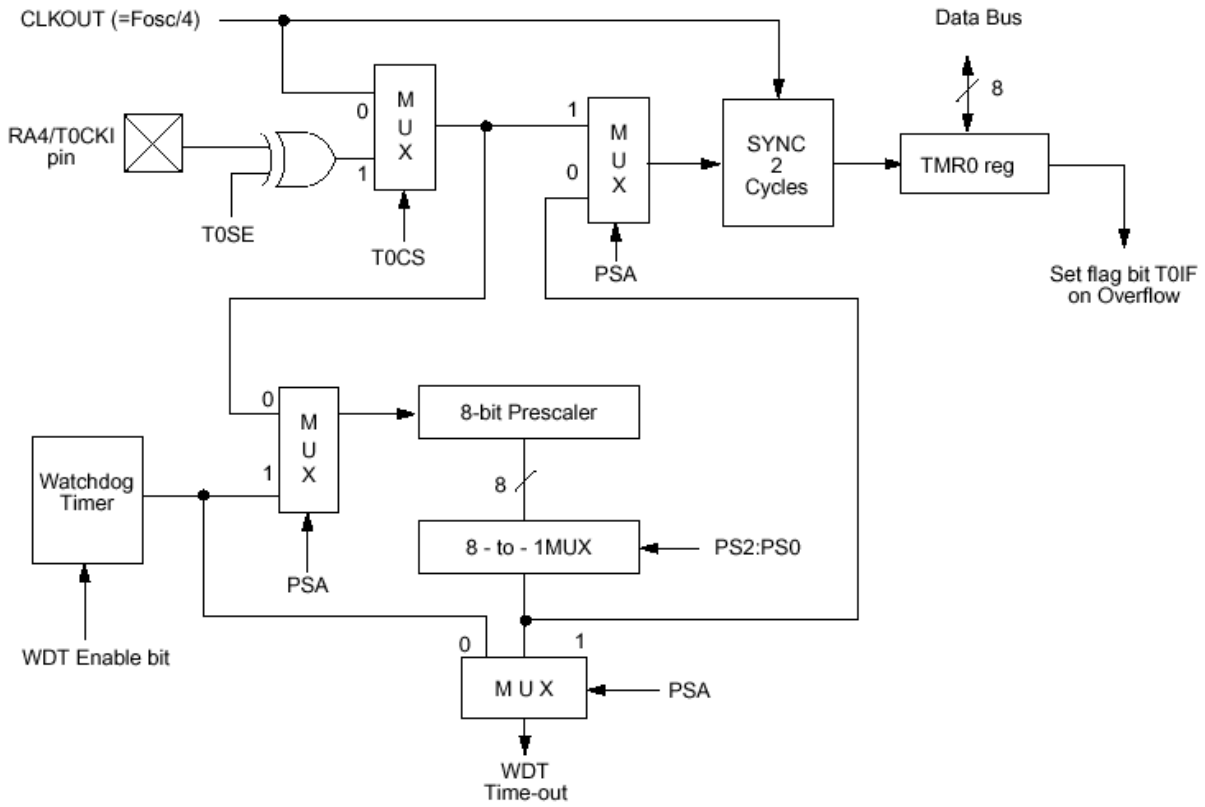


Figura 16 – Prescaler / Postscaler para TIMER0 e Watchdog

Quando o Prescaler estiver ligado ao Timer0, qualquer instrução que implemente uma escrita no registo TMR0 provoca um reset do Prescaler. No caso de estar ligado ao Watchdog, a instrução CLRWDT também provoca o reset do Prescaler.

4.3 Interrupção do Timer0

Sempre que o incremento do registo TMR0 provocar a passagem de FFh para 00h (overflow) é activada a flag de interrupção T0IF (Timer0 Interrupt Flag, bit 2 do registo INCON). Se a respectiva interrupção estiver autorizada, como veremos mais à frente, é desencadeado um processo de interrupção do μC. Esta flag deve ser colocada em 0 por software.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

Tabela 2 – Registos associados ao TIMER0

5 Memória de Dados EEPROM

Trata-se de um conjunto de 64 bytes implementados em memória não volátil do tipo EEPROM. Estas posições de memória podem ser lidas ou escritas durante o funcionamento normal do μC, dispondo do seu próprio espaço de endereçamento. O acesso a esta zona de memória é feita indirectamente através de 4 SFRs (registos de função especial):

- EECON1
- EECON2
- EEDATA
- EEADR

O registo EEDATA recebe os 8 bits lidos de um determinado endereço ou contém os 8 bits destinados a ser escritos num determinado endereço. Este endereço deve estar escrito no registo EEADR. Dado que só dispomos de 64 bytes. Os endereços podem variar entre 00h e 3Fh.

A escrita de um byte provoca o automático apagamento da respectiva posição de memória antes que o novo byte seja escrito (erase before write).

Esta memória de dados permite um número bastante elevado de operações de apagamento/escrita, tipicamente de 10 milhões de operações, sendo estas operações controladas por um temporizador interno ao dispositivo de memória. O tempo destas operações pode variar com a tensão de alimentação, com a temperatura e de μC para μC.

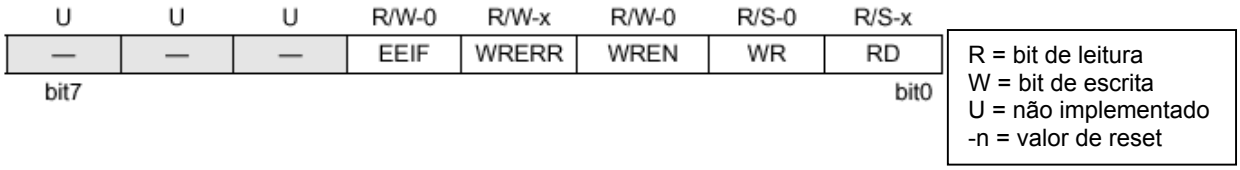
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2								---- ----	---- ----

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends upon condition. Shaded cells are not used by data EEPROM.

Tabela 3 – Registos associados à memória de dados E^2PROM

5.1 Leitura da memória de dados EEPROM

Para ler uma posição de memória basta para isso colocar em EEADR o endereço pretendido e colocar em 1 o bit RD do registo EECON1 (bit 0). O resultado da leitura fica disponível no registo EEDATA no próximo ciclo de instrução.



Bit 7:5	**Não implementados –** Lidos como '0'
Bit 4	**EEIF** – Flag de interrupção da operação de escrita na EEPROM 1 = operação de escrita terminada (deve ser colocada em 0 por software) 0 = a operação de escrita não está completa ou não começou
Bit 3	**WRERR** – Flag de erro da EEPROM 1 = a operação de escrita foi terminada prematuramente 0 = operação de escrita terminada correctamente
Bit 2	**WREN** – Bit de autorização de escrita na EEPROM 1 = operação de escrita autorizada 0 = operação de escrita não autorizada
Bit 1	**WR** – Bit de controlo de escrita 1 = inicia uma operação de escrita. Este bit é colocado em 0 pelo hardware no fim da operação de escrita 0 = o ciclo de escrita na EEPROM está completo
Bit 0	**RD** – Bit de controlo de leitura 1 = inicia uma operação de leitura. Este bit é colocado em 0 pelo hardware no fim da operação de leitura 0 = operação de leitura não iniciada

Figura 17 – Registo EECON1

Exemplo de leitura:

```
movlw   endereço        ;Endereço em EEADR
movwf   EEADR
bsf     STATUS,RP0      ;Comuta para banco 1
bsf     EECON1,RD       ;Activa bit de leitura
bcf     STATUS,RP0      ;Retorna ao banco 0
movf    EEDATA,W        ;Lê dados de EEDATA
```

5.2 Escrita na memória de dados EEPROM

O procedimento de escrita é iniciado com o preenchimento do registo de endereço e do registo de dados. A seguir deve ser iniciada uma sequência específica de instruções que permitem a escrita real do byte na posição de memória endereçada por EEADR.

Exemplo de escrita:

```
bsf     STATUS,RP0      ;Comuta para banco 1
bcf     INTCON,GIE      ;Desactiva interrupções
bsf     EECON1,WREN     ;Autoriza escrita
movlw   55h             ;Inicia processo de escrita
movwf   EECON2
movlw   0AAh
movwf   EECON2
bsf     EECON1,WR       ;Ordem de escrita
bsf     INTCON,GIE      ;Autoriza interrupções
bcf     EECON1,WREN     ;Inibe escrita
bcf     STATUS,RP0      Comuta para banco 0
```

A operação de escrita não será iniciada se a sequência mostrada acima não for executada (escrever 55h em EECON2, escrever AAh em EECON2, activar o bit WR). Dado que a sequência de escrita não deve ser interrompida, as interrupções, se activas, devem ser desactivadas.

Logo que a operação de escrita estiver terminada, o bit WR é colocado a 0 pelo hardware e o bit EEIF é colocado a 1 para sinalizar o fim da operação. Este bit, que pode ser utilizado pelo sistema de interrupções do μC, deve ser colocado a 0 por software.

Dado que o bit WREN (WRite ENable) funciona como uma segurança para o processo de escrita, prevenindo que, por um funcionamento deficiente, haja escritas esporádicas na memória, deve ser colocado a zero logo a seguir a ter dado a ordem de escrita.

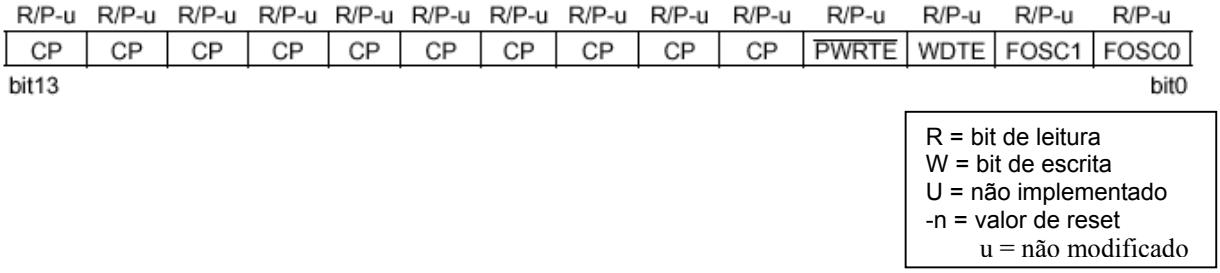
6 Atributos específicos do CPU

O que permite distinguir um μC de outros processadores é o conjunto de circuitos internos que lhe permitem lidar com aplicações de tempo real. O PIC16F84 dispõe de um conjunto de atributos que permitem maximizar a fiabilidade do sistema onde está inserido, minimizar os custos através da possibilidade de eliminação de componentes externos, e uma poupança de energia através de modos de funcionamento específicos. Estes atributos são:

- Selecção do oscilador
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscilator Start-up Timer (OST)
- Interrupções
- Watchdog Timer (WDT)
- SLEEP
- Protecção do código
- Identificação
- Programação série no circuito (In-circuit serial programming)

6.1 Bits de configuração

Os bits de configuração podem ser programados de maneira a seleccionar diferentes tipos de configuração do micro controlador. Estes bits estão mapeados na posição de memória 2007h e só podem ser acedidos durante a programação.



Bit 13:4	**CP –** Bits de protecção do código 1 = protecção de código desligada 0 = memória de código protegida (o código não pode ser copiado)
Bit 3	**$\overline{\text{PWRTE}}$** – Bit de activação do temporizador de power-up 1 = temporizador não activado 0 = temporizador activado
Bit 2	**WDTE** – Bit de activação do watchdog 1 = watchdog não activado 0 = watchdog activado
Bit 1:0	**FOSC1:FOSC0** – Bits de selecção do tipo de oscilador 11 = oscilador RC 10 = oscilador HS 01 = oscilador XT 00 = oscilador LP

Figura 18 – Palavra de Configuração

6.2 Configurações do oscilador

O modo de funcionamento do oscilador do PIC16F84A pode ser programado em 4 modos diferentes, utilizando-se para isso os bits FOSC1 e FOSC0 da palavra de configuração.

- LP - Cristal 'low power'
- XT – Cristal
- HS – Cristal 'high speed'
- RC – Malha RC (resistência + condensador)

Caso seja utilizado um cristal, este deve ser ligado aos pinos OSC1 e OSC2 de acordo com a figura seguinte. Na tabela anexa à figura indica-se o tipo de programação a efectuar de acordo com a frequência de trabalho pretendida, dando-se ainda uma ideia dos valores a utilizar para os condensadores C1 e C2.

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz 200 kHz	68 - 100 pF 15 - 33 pF	68 - 100 pF 15 - 33 pF
XT	100 kHz 2 MHz 4 MHz	100 - 150 pF 15 - 33 pF 15 - 33 pF	100 - 150 pF 15 - 33 pF 15 - 33 pF
HS	4 MHz 10 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

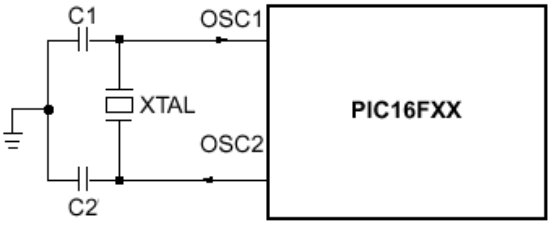


Figura 19 – Oscilador com cristal

Nos modos de funcionamento XT, LP ou HS, o μC aceita um sinal de relógio externo como se mostra na fig. 20.

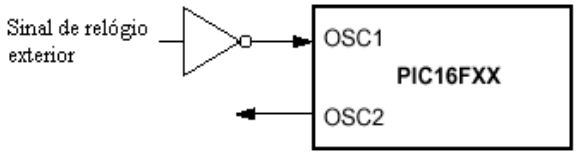


Figura 20 – Oscilador externo

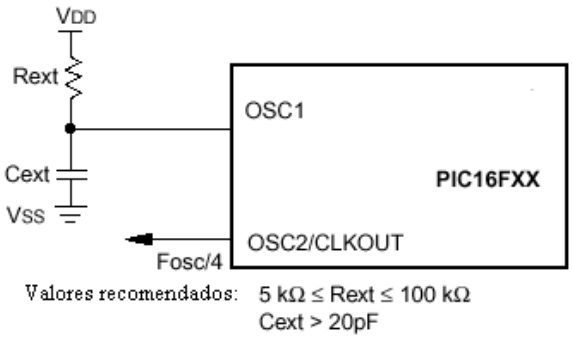


Figura 21 – Oscilador RC

Para aplicações em que não são necessárias temporizações precisas, a opção do oscilador controlado por uma malha RC é a solução mais barata, como se mostra na fig. 21. Neste caso, a frequência de oscilação é uma função da tensão de alimentação, dos valores de R e C e da temperatura. Mais ainda, esta frequência pode variar de um μC para outro μC devido a variações de parâmetros internos.

6.3 Reset

O PIC16F84A permite vários tipos de reset:

- Power-on reset (POR)
- Reset pelo pino MCLR durante o funcionamento normal do μC
- Reset pelo pino MCLR quando o μC se encontra em SLEEP
- Reset provocado pelo WDT durante o funcionamento normal do μC
- Reset provocado pelo WDT quando o μC se encontra em SLEEP

Os valores que são colocados nos diferentes registos internos após estas operações de reset podem ser consultados em tabelas que são apresentadas nas folhas de especificação (datasheet) do micro controlador.

6.4 Power-on Reset (POR)

Sempre que o hardware interno do μC detecta uma subida da tensão de alimentação, mais precisamente quando essa tensão se encontra entre 1.2V e 1.7V, é gerado um impulso de RESET desde que o pino exterior MCLR esteja ligado directamente (ou através de uma resistência) a V_{DD}. Isto permite eliminar os componentes externos que implementam a malha RC normalmente necessária para provocar o reset do μC quando é ligada a alimentação. De notar que, para um correcto funcionamento deste sistema de reset, é necessário que o tempo de subida da tensão de alimentação seja inferior a um limite mínimo que consta das especificações eléctricas do μC (cerca de 0.05V/ms).

No caso em que o tempo de subida da tensão de alimentação não obedeça às características definidas anteriormente, será necessária a utilização de uma malha RC externa para provocar o power-on reset do μC, com se representa na figura ao lado. Para que a queda de tensão na resistência R não seja superior a 0.2V, o valor de R deve ser inferior a 40KΩ. O valor de R1, para protecção da entrada, deverá estar entre 100Ω e 1KΩ. O díodo D permite a descarga do condensador quando se verifica o power-down.

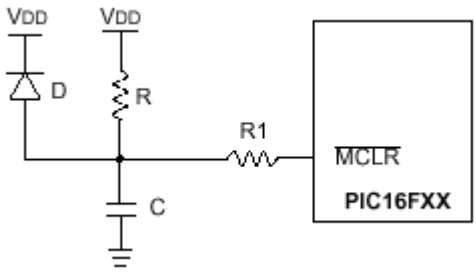


Figura 22 – Malha de Reset exterior

6.5 Temporizador de Power-up (PWRT)

O temporizador de Power-up permite uma temporização fixa de cerca de 72 ms (T_{PWRT}) a partir da situação de POR (ver figuras 23, 24, 25 e 26). Este temporizador, feito a partir de um oscilador RC interno, permite manter o μC em situação de reset durante aquele tempo, permitindo assim que a tensão de alimentação atinja uma situação de estabilidade antes do μC entrar em funcionamento.
A utilização deste temporizador pode ou não ser permitida através do bit PWRTE da palavra de configuração (ver parágrafo 6.1).

6.6 Temporizador para início do oscilador (OST)

Este temporizador permite um atraso na entrada em funcionamento do μC de 1024 ciclos de relógio (T_{OST}), a partir do final da temporização do PWRT (ver figuras 23, 24, 25 e 26). Isto permite uma melhor estabilização do oscilador interno depois de uma operação de entrada em funcionamento.

Este temporizador só existe nos modos de funcionamento XT, LP e HS.

Quando o tempo de subida da tensão de alimentação é muito grande, é possível que no final das duas temporizações, $T_{PWRT} + T_{OST}$, ainda não tenha sido conseguida a situação de estabilidade, isto é, se se verificar ainda que $V_{DD} < V_{DDmin}$. Neste caso é necessário utilizar a malha externa RC para aumentar o tempo do power-on reset do μC.

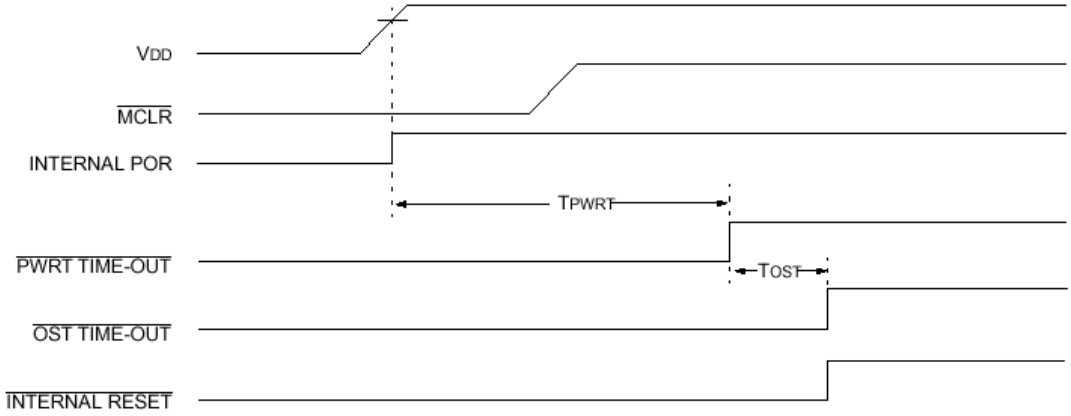


Figura 23 – Sequência de Reset (MCLR não ligado a V_{DD}): caso 1

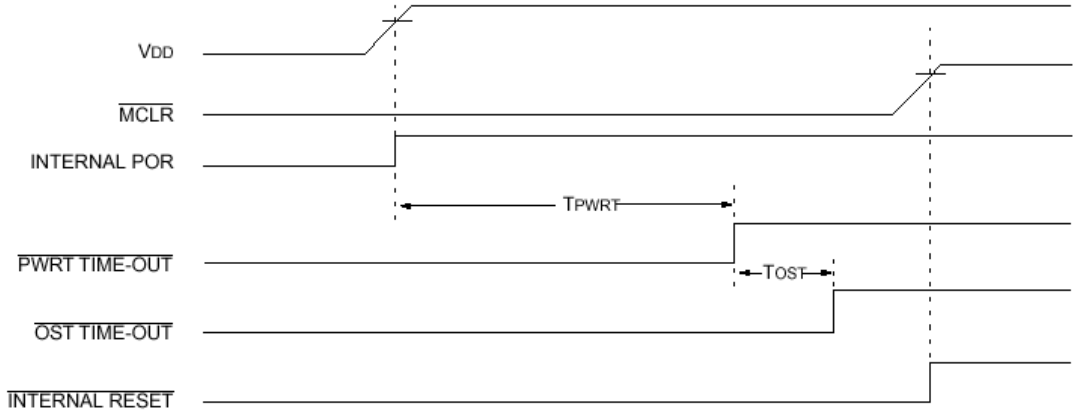


Figura 24 - Sequência de Reset (MCLR não ligado a V_{DD}): caso 2

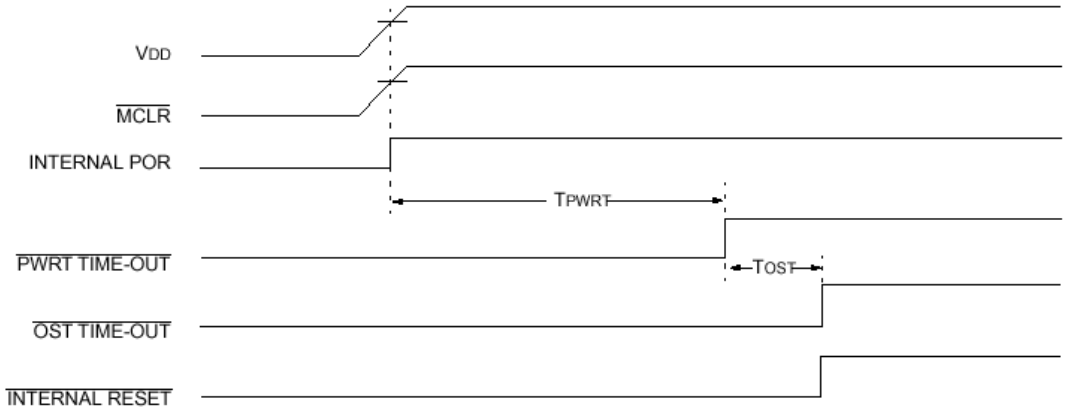
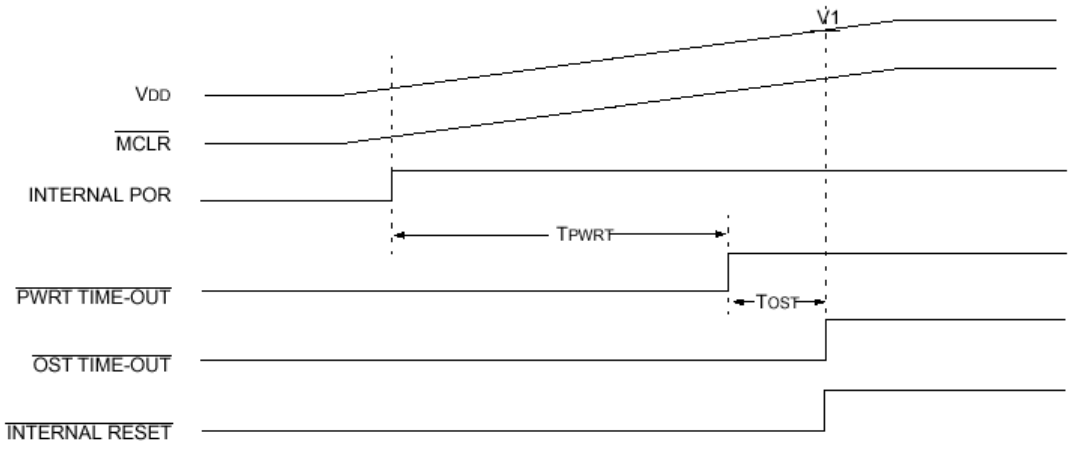


Figura 25 - Sequência de Reset (MCLR ligado a V_{DD}): V_{DD} com tempo de subida baixo



Quando V_{DD} tem um tempo de subida alto, é possível que T_{PWRT} + T_{OST} terminem antes de V_{DD} ter atinjido o seu valor final. Neste exemplo, o reset será correcto se V1 ≥ V_{DD}min.

Figura 26 - Sequência de Reset (MCLR ligado a V_{DD}): V_{DD} com tempo de subida alto

6.7 Bits de estado de Power-down (TO/PD)

A tabela seguinte mostra os valores que poderão ser encontrados nos bits TO e PD do registo STATUS depois de uma operação de Reset ou de retoma de funcionamento depois de uma operação de SLEEP.

TO	PD	Condição
1	1	Power-on Reset ou reset por MCLR durante o funcionamento normal
0	1	Reset provocado pelo WDT durante funcionamento normal
1	0	Reset por MCLR durante SLEEP ou saída de SLEEP provocada por interrupção
0	0	Saída de SLEEP por WDT

Tabela 4 – Bits de estado e seu significado

6.8 Interrupções

O PIC16F84A dispõe de um único nível de interrupção com 4 fontes de interrupção:

- Interrupção externa através do pino RB0/INT
- Interrupção provocada pelo overflow do TMR0
- Interrupção provocada pela mudança de estado do PORTB (RB7:RB4)
- Interrupção provocada pelo fim da operação de escrita na E^2PROM de dados

O registo de controlo das interrupções (INTCON) contém as diferentes flags de pedido de interrupção e os bits que permitem activar ou desactivar individualmente ou na globalidade as diferentes interrupções.

A lógica que permite o desencadeamento do processo de interrupção está representada na figura seguinte.

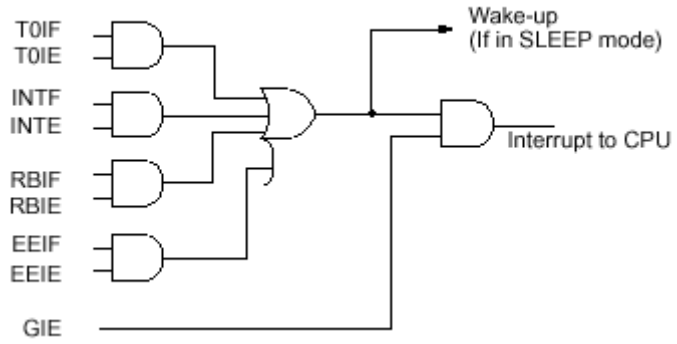


Figure 27 – Lógica associada às interrupções

Na situação de reset, todos os bits correspondentes à autorização individual ou global das interrupções encontram-se desactivados (T0IE = INTE = RBIE = EEIE = GIE = 0).

A rotina de serviço de qualquer interrupção deve terminar sempre com a instrução RETFIE (RETurn From Interrupt Enable) que, além de fazer o sistema retornar ao programa no local onde se verificou a interrupção, coloca também GIE = 1 permitindo de novo o acesso dos periféricos às interrupções.

Quando o μC responde a um pedido de interrupção, o bit GIE é colocado em 0 para não permitir mais nenhuma interrupção, o endereço de retorno é colocado na stack e o PC é carregado com o valor 0004h que é o vector de interrupção para este μC. Será a partir desta posição da memória de programa que deverá ser escrita a rotina de serviço da interrupção. Nesta rotina de serviço, caso haja mais do que uma interrupção autorizada, será necessário ler as flags de pedido de interrupção para se poder determinar qual o periférico que requisitou os serviços do sistema de interrupções. As flags de interrupção activadas devem ser colocadas em 0 por software de maneira a que, quando o μC sair da rotina de serviço, não seja interrompido novamente.

Para as interrupções externas, o tempo de latência será de 3 a 4 ciclos de instrução, dependendo do exacto momento em que foi efectuado o pedido.

De notar ainda que as diferentes flags de pedido de interrupção podem ser activadas independentemente de as respectivas interrupções estarem ou não autorizadas ou do estado do bit GIE.

6.8.1 Interrupção INT

A interrupção externa INT é activada por flanco:

- ascendente se INTEDG = 1 (OPTIN_REG.6)
- descendente se INTEDG = 0

Sempre que um flanco válido aparece no pino RB0/INT a flag INTF é activada. Esta flag deve ser desactivada por software.
A autorização desta interrupção é feita através do bit INTE (INTCON.4).

A interrupção INT pode 'acordar' o μC do estado de SLEEP se a interrupção estiver autorizada. O estado do bit GIE decide qual o procedimento a ter depois de 'acordar' (ver 6.11).

6.8.2 Interrupção do TMR0

O overflow do TMR0 (FFh → 00h) coloca em 1 a flag T0IF (INTCON.2). A Autorização desta interrupção é feita à custa do bit T0IE (INTCON.5).

6.8.3 Interrupção do PORTB

A mudança de estado em qualquer um dos bits RB7:4 coloca em 1 a flag RBIF (INTCON.0). A autorização da interrupção pode ser feita através do bit RBIE (INTCON.3).

6.8.4 Interrupção da EEPROM de Dados

Quando a EEPROM termina o processo interno de escrita, activa a flag EEIF (EECON1.4). A autorização da interrupção é feita através do bit EEIE (INTCON.6).

6.9 Temporizador Watchdog (WDT)

O temporizador Watchdog é constituído por um oscilador interno RC, não necessitando de quaisquer outros componentes externos. Este oscilador é totalmente independente do oscilador do μC. Isto quer dizer que, mesmo que o oscilador principal não funcione (por exemplo no caso do μC se encontrar no estado de SLEEP), o WDT continuará em funcionamento.
O diagrama de blocos deste periférico pode ser visto na figura seguinte.

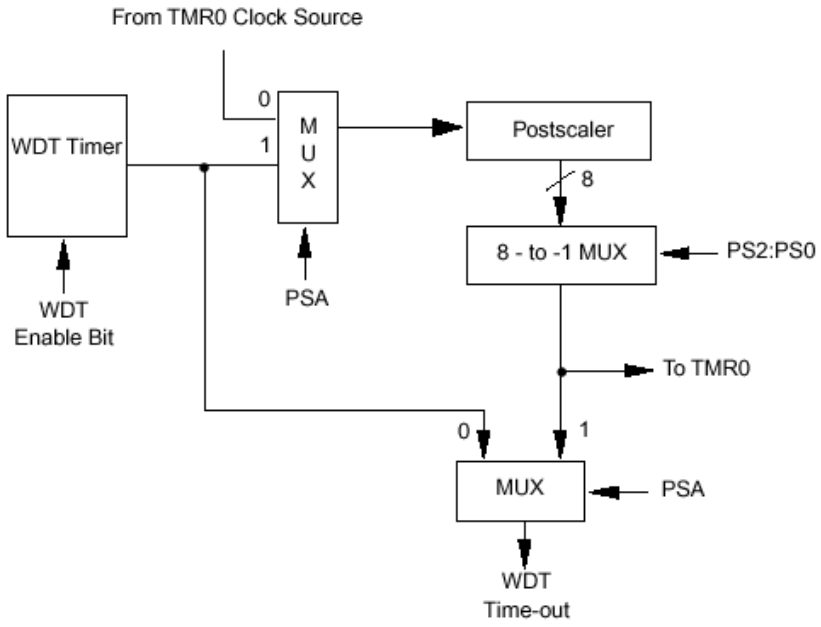


Figura 28 – Diagrama de blocos do sistema WatchDog

Durante o funcionamento normal, um timeout deste temporizador provoca um RESET do μC. Se o μC se encontrar em SLEEP, o timeout do WDT provoca o 'acordar' do μC, continuando o programa na instrução seguinte à instrução de SLEEP.

A activação / desactivação do WDT é feita através do bit WDTE da palavra de configuração.

6.9.1 Período do WDT

Este temporizador tem um período nominal de 18 ms (sem prescaler). Este período pode sofrer variações pois está dependente de alguns parâmetros como a tensão de alimentação, a temperatura e outros parâmetros inerentes ao próprio μC. Se forem necessários períodos superiores, pode ser utilizado o prescaler com uma taxa de divisão de 1 até 128. Desta maneira podemos conseguir períodos de timeout até cerca de 2.3 segundos.

As instruções CLRWDT e SLEEP fazem o reset do tempo de watchdog e do postscaler se este estiver atribuído ao WDT.

6.9.2 Uma utilização possível para o WDT

Um dos problemas que se pode por no funcionamento de um μC é a alteração súbita do PC devida a diversos tipos de fenómenos, como por exemplo ruído eléctrico. Esta alteração não controlada por programa faz com que, a partir desse momento, as tarefas desempenhadas pelo μC nada tenham a ver com o programa que foi desenhado pelo utilizador. Só uma operação de RESET poderá repor o normal funcionamento do sistema.

A utilização do WDT pode remediar este tipo de situações, desde que se encontre activado. O programador deverá espalhar criteriosamente pelo programa instruções CLRWDT de maneira a que nunca se verifique o timeout do temporizador em qualquer situação de funcionamento normal. Logo que se verifique uma falha, é provável que essas instruções não sejam executadas provocando assim um timeout do WDT com o consecutivo RESET do μC.

6.10 Modo de funcionamento Power-down (SLEEP)

O PIC16F84 pode ser colocado em power-down (SLEEP) podendo mais tarde ser de novo colocado em funcionamento normal (Wake-up).

6.10.1 SLEEP

A instrução SLEEP é utilizada para fazer com que o μC entre em power-down. Quando isto acontece, é feito o reset do WDT (se estiver activo), os bits PD e TO são colocados respectivamente em 0 e 1 e é desactivado o oscilador interno. Os portos de I/O mantém o estado que tinham antes da execução da instrução SLEEP.
Para que o consumo de energia seja minimizado, todos os pinos I/O devem estar a VSS ou a VDD, devendo-se garantir ainda que nenhum circuito exterior esteja a receber corrente do μC. Os pinos programados como alta impedância (entrada) devem ter pull-up ou pull-down exterior para se evitarem flutuações na entrada.

6.10.2 Wake-up

O μC pode sair do modo de SLEEP através de um dos acontecimentos seguintes:

- Reset exterior por activação do pino MCLR
- Timeout do WDT se estiver activo
- Interrupções RB0/INT, RB7:4 ou fim de escrita na EEPROM

O TMR0 não pode gerar interrupção pois o seu funcionamento depende do oscilador interno que, em power-down, se encontra parado.

No primeiro caso, reset por MCLR, provoca o RESET do μC. Os dois casos restantes são considerados como uma continuação da execução do programa. Os bits TO e PD permitem determinar a causa do reset do μC.

Enquanto está a ser executada a instrução SLEEP, a próxima instrução está a ser pesquisada (PC + 1). Para que o μC 'acorde' a partir de uma interrupção é necessário que esta esteja autorizada. A saída de sleep acontece independentemente do estado do bit GIE. Se GIE estiver a 0, o programa continua na instrução seguinte ao SLEEP. Se GIE = 1 então o μC executa a instrução seguinte ao SLEEP e salta para o vector de interrupção (0004h) onde se encontra a rotina de serviço da interrupção. Caso não haja interesse na instrução seguinte ao SLEEP, essa instrução poderá ser um NOP. Na figura seguinte mostra-se o diagrama temporal do processo descrito.

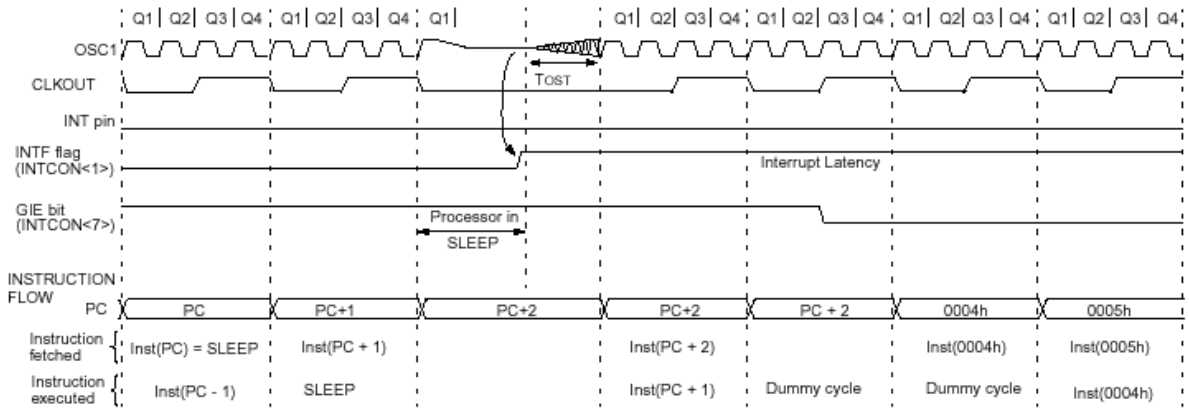


Figura 29 – Wake-up por interrupção

6.11 Protecção de código

Quando se faz a programação do μC, é possível programar os bits de protecção de código. Se isto acontecer não será mais possível fazer a leitura do código gravado. Só uma nova gravação poderá desactivar a protecção. Caso se opte por código desprotegido então em qualquer altura esse código pode ser lido quer para comparação quer para efectuar cópias do μC.

6.12 Identificador

As posições de memória 2000h – 2004h são usadas para guardar um identificador do produto (nº de série, checksum, etc.). Estas posições de memória só podem ser acedidas durante o processo de gravação.

6.13 Programação série in-circuit

Estes μCs podem ser programados na placa onde estão inseridos, através de uma comunicação série. Isto permite que as placas sejam totalmente montadas (incluindo o próprio μC) sendo posteriormente efectuada a gravação do código no μC. Este processo utiliza unicamente 2 linhas para sinal de relógio e para dados e mais três linhas para a tensão de alimentação e para a tensão de programação. A Microchip fornece o protocolo e todas as especificações técnicas necessárias para este tipo de programação.

7 Tabela de Instruções

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	**f, d**	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	**f, d**	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	**f**	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	**-**	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	**f, d**	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	**f, d**	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	**f, d**	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	**f, d**	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	**f, d**	Increment f, Skip if 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	**f, d**	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	**f, d**	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	**f**	Move W to f	1	00 0000 lfff ffff			
NOP	**-**	No Operation	1	00 0000 0xx0 0000			
RLF	**f, d**	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	**f, d**	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	**f, d**	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	**f, d**	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	**f, d**	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	**f, b**	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	**f, b**	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	**f, b**	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	**f, b**	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	**k**	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	**k**	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	**k**	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	**-**	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	**k**	Go to address	2	10 1kkk kkkk kkkk			
IORLW	**k**	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	**k**	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	**-**	Return from interrupt	2	00 0000 0000 1001			
RETLW	**k**	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	**-**	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	**-**	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	**k**	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	**k**	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	