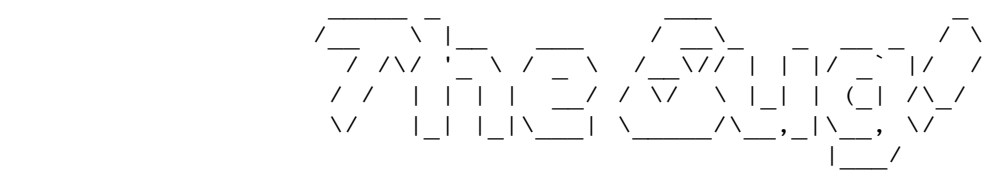
[--- The Bug! Magazine



[M . A . G . A . Z . I . N . E]

[Numero 0x01 <---> Edicao 0x01 <---> Artigo 0x08]

.> 05 de Marco de 2006,
.> The Bug! Magazine < staff [at] thebugmagazine [dot] org >

Introdução a família de microntroladores PIC

por Rafael Silva < rafaelsilva [at] rfdslabs [dot] com [dot] br >

1. Introdução

A partir deste artigo você vai estar apto a operar qualquer microcontrolador da família PIC da MICROCHIP. Este texto tenta explicar funcionalidades básicas dos microcontroladores PIC como, memórias, arquiteturas, formato das instruções, registradores etc... Lets Go...

É no ano de 1969 que uma equipa de engenheiros japoneses pertencentes à companhia BUSICOM chega aos Estados Unidos com a encomenda de alguns circuitos integrados para calculadoras a serem implementados segundo os seus projectos. A proposta foi entregue à INTEL e Marcian Hoff foi o responsável pela sua concretização. Como ele tinha tido experiência de trabalho com um computador (PC) PDP8, lembrou-se de apresentar uma solução substancialmente diferente em vez da construção sugerida. Esta solução pressupunha que a função do circuito integrado seria determinada por um programa nele armazenado. Isso significava que a configuração deveria ser mais simples, mas também era preciso muito mais memória que no caso do projecto proposto pelos engenheiros japoneses. Depois de algum tempo, embora os engenheiros japoneses tenham tentado encontrar uma solução mais fácil, a ideia de Marcian venceu e o primeiro microprocessador nasceu. Ao transformar esta ideia num produto concreto, Frederico Faggin foi de uma grande utilidade para a INTEL. Ele transferiu-se para a INTEL e, em somente 9 meses, teve sucesso na criação de um produto real a partir da sua primeira concepção. Em 1971, a INTEL adquiriu os direitos sobre a venda deste bloco integral. Primeiro eles compraram a licença à companhia BUSICOM que não tinha a mínima percepção do tesouro que possuía. Neste mesmo ano, apareceu no mercado um microprocessador designado por 4004. Este foi o primeiro microprocessador de 4 bits e tinha a velocidade de 6 000 operações por segundo. Não muito tempo depois, a companhia Americana CTC pediu à INTEL e à Texas Instruments um microprocessador de 8 bits para usar em terminais. Mesmo apesar de a CTC acabar por desistir desta ideia, tanto a Intel como a Texas Instruments continuaram a trabalhar no microprocessador e, em Abril de 1972, os primeiros microprocessadores de 8 bits apareceram no mercado

com o nome de 8008. Este podia endereçar 16KB de memória, possuía 45 instruções e tinha a velocidade de 300 000 operações por segundo. Esse microprocessador foi o pioneiro de todos os microprocessadores actuais. A Intel continuou com o desenvolvimento do produto e, em Abril de 1974 pôs cá fora um processador de 8 bits com o nome de 8080 com a capacidade de endereçar 64KB de memória, com 75 instruções e com preços a começarem em $360.

Uma outra companhia Americana, a Motorola, apercebeu-se rapidamente do que estava a acontecer e, assim, pôs no mercado um novo microprocessador de 8 bits, o 6800. O construtor chefe foi Chuck Peddle e além do microprocessador propriamente dito, a Motorola foi a primeira companhia a fabricar outros periféricos como os 6820 e 6850. Nesta altura, muitas companhias já se tinham apercebido da enorme importância dos microprocessadores e começaram a introduzir os seus próprios desenvolvimentos. Chuck Peddle deixa a Motorola para entrar para a MOS Technology e continua a trabalhar intensivamente no desenvolvimento dos microprocessadores.

Em 1975, na exposição WESCON nos Estados Unidos, ocorreu um acontecimento crítico na história dos microprocessadores. A MOS Technology anunciou que ia pôr no mercado microprocessadores 6501 e 6502 ao preço de $25 cada e que podia satisfazer de imediato todas as encomendas. Isto pareceu tão sensacional que muitos pensaram tratar-se de uma espécie de vigarice, considerando que os competidores vendiam o 8080 e o 6800 a $179 cada. Para responder a este competidor, tanto a Intel como a Motorola baixaram os seus preços por microprocessador para $69,95 logo no primeiro dia da exposição. Rapidamente a Motorola pôs uma acção em tribunal contra a MOS Technology e contra Chuck Peddle por violação dos direitos de autor por copiarem ao copiarem o 6800. A MOS Technology deixou de fabricar o 6501, mas continuou com o 6502. O 6502 é um microprocessador de 8 bits com 56 instruções e uma capacidade de endereçamento de 64KB de memória. Devido ao seu baixo custo, o 6502 torna-se muito popular e, assim, é instalado em computadores como KIM-1, Apple I, Apple II, Atari, Comodore, Acorn, Oric, Galeb, Orao, Ultra e muitos outros. Cedo aparecem vários fabricantes do 6502 (Rockwell, Sznertek, GTE, NCR, Ricoh e Comodore adquiriram a MOS Technology) que, no auge da sua prosperidade, chegou a vender microprocessadores à razão de 15 milhões por ano!

Contudo, os outros não baixaram os braços. Frederico Faggin deixa a Intel e funda a Zilog Inc.

Em 1976, a Zilog anuncia o Z80. Durante a concepção deste microprocessador, Faggin toma uma decisão crítica. Sabendo que tinha sido já desenvolvida uma enorme quantidade de programas para o 8080, Faggin conclui que muitos vão permanecer fieis a este microprocessador por causa das grandes despesas que adviriam das alterações a todos estes programas. Assim, ele decide que o novo microprocessador deve ser compatível com o 8080, ou seja, deve ser capaz de executar todos os programas que já tenham sido escritos para o 8080. Além destas características, outras características adicionais foram introduzidas, de tal modo que o Z80 se tornou um microprocessador muito potente no seu tempo. Ele podia endereçar directamente 64KB de memória, tinha 176 instruções, um grande número de registos, uma opção para refrescamento de memória RAM dinâmica, uma única alimentação, maior velocidade de funcionamento, etc. O Z80 tornou-se um grande sucesso e toda a gente se transferiu do 8080 para o Z80.

Pode dizer-se que o Z80 se constituiu sem sombra de dúvida como o microprocessador de 8 bits com maior sucesso no seu tempo. Além da Zilog, outros novos fabricantes como Mostek, NEC, SHARP e SGS apareceram. O Z80 foi o coração de muitos computadores como o Spectrum, Partner, TRS703, Z-3 e Galaxy,

que foram aqui usados.

Em 1976, a Intel apareceu com uma versão melhorada do microprocessador de 8 bits e designada por 8085. Contudo, o Z80 era tão superior a este que, bem depressa, a Intel perdeu a batalha. Ainda que mais alguns microprocessadores tenham aparecido no mercado (6809, 2650, SC/MP etc.), já tudo estava então decidido. Já não havia mais grandes melhorias a introduzir pelos fabricantes que fundamentassem a troca por um novo microprocessador, assim, o 6502 e o Z80, acompanhados pelo 6800, mantiveram-se como os mais representativos microprocessadores de 8 bits desse tempo. “Texto retirado do livro the PIC MICROCONTROLER”

2. Introdução à família de microcontroladores PIC

Um microcontrolador se caracteriza por incorporar no mesmo encapsulamento um microprocessador, memória de programa e dados e vários periféricos como temporizadores, “*watchdog timers*”, comunicação serial, conversores Analógico/Digital, geradores de PWM, etc, fazendo com que o hardware final fique extremamente complexo.

A *Microchip* é uma empresa precursora no uso de tecnologia *RISC* (*Reduced Instruction Set Computer*) em microcontroladores. Diferente da arquitetura Von Neumann, a estrutura *RISC* é baseada em barramentos independentes para dados e para programa, e tem como característica fundamental os tamanhos diferenciados (por ex: no PIC16C55X o barramento de dados é de 8 bits, enquanto o de programa é de 14 bits) o que significa que uma instrução está ”empacotada” em uma única palavra de programa (no caso do PIC16C55X de 14 bits), que além de conter o *opcode* (instrução) contém os operandos (dados para execução da instrução).

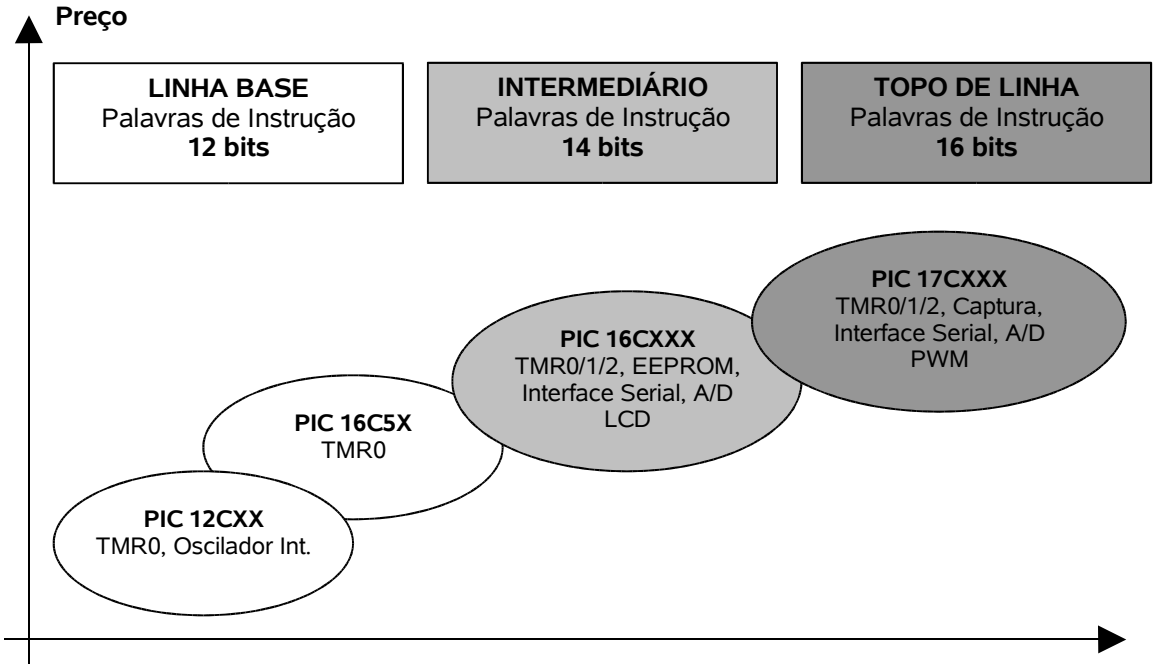


Figura 1 – Evolução dos Microcontroladores PIC

A *Microchip* oferece 4 (quatro) famílias de microcontroladores de 8 bits no barramento de dados que se adaptam aos mais variados projetos. Estas famílias são:

- PIC12CXXX: Linha Compacta.
- PIC16C5X/PIC16C55X: Linha Base.
- PIC16CXX: Linha Intermediária.
- PIC17CXX: Topo de Linha.

Todas as famílias oferecem diversas opções de memória de programa: *OTP* (*One Time Programable*) e *EPROM* (*Erasable and Programmable Read Only Memory* – utilizada para desenvolvimento). Além disso, oferecem opções de baixa tensão e diversos tipos de circuito osciladores, assim como várias opções de baixa tensão e diversos tipos de circuito osciladores, assim como várias opções de encapsulamento. Alguns componentes estão disponíveis em *ROM* (mascarados) e *EEPROM/FLASH* (reprogramáveis).

3. Arquitetura do microcontrolador

O alto desempenho da família de microcontroladores PIC pode ser atribuído as seguintes características de arquitetura RISC:

- Mapa de Registradores versátil
- Todas as instruções com palavras simples
- Palavra de instrução Longa
- Arquitetura de instruções em “Pipeline”
- Instruções de apenas um ciclo de máquina
- Conjunto de instruções reduzido
- Conjunto de instruções ortogonal (simétrico)

A arquitetura von-Neumman tradicional utiliza o mesmo barramento para fazer a busca a instruções na memória de programa e para acessar (escrever ou ler) a memória de dados.

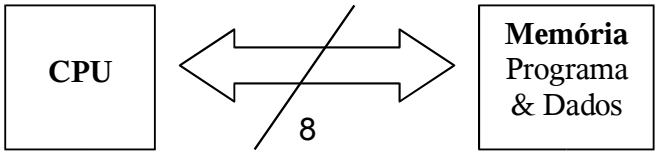


Figura 2 - Arquitetura Von-Neumman

A arquitetura do microcontrolador utiliza dois barramentos de endereços distintos para acessar instruções e dados.

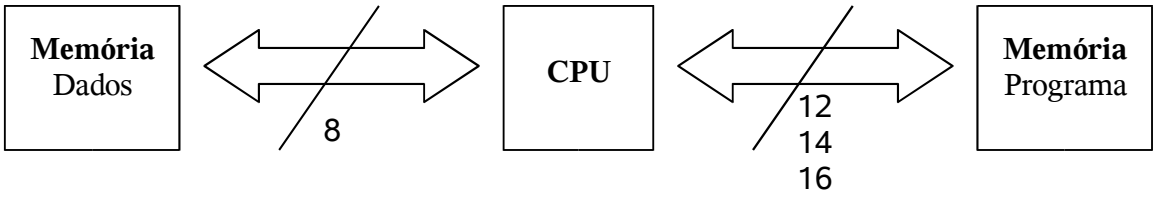


Figura 3 – Arquitetura do microcontrolador PIC

4. Formato das instruções no microcontrolador PIC

A entrada de clock (pino OSC1/CLKIN) é internamente dividida por quatro para gerar quatro clocks em quadratura sem sobreposição, nomeados Q1, Q2, Q3, e Q4. Internamente, o Contador de Programa (PC) é incrementado em Q1 e a instrução é retirada da memória de programa e colocada no registrador de instruções em Q4. Ela é decodificada e executada no ciclo seguinte de Q1 até Q4.

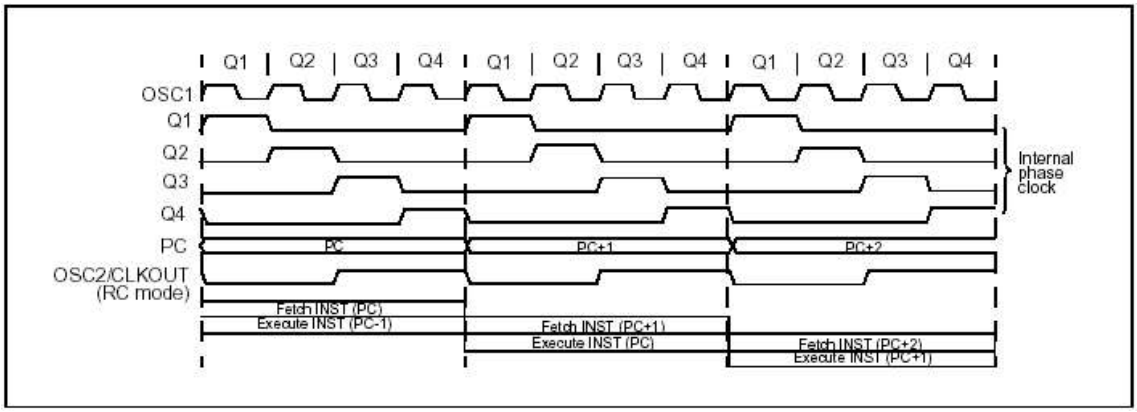


Figura 4 – Ciclo de instruções do microcontrolador

Um ciclo de instruções consiste de quatro ciclos Q (Q1, Q2, Q3, Q4). A busca e execução da instrução são feitas em linha, de tal forma que a busca leva um ciclo de instrução e a execução leva outro ciclo. Contudo, devido a característica de "*pipeline*", cada instrução é executada efetivamente em um ciclo, pois simultaneamente ocorrem a execução de uma instrução e a busca a instrução seguinte. Se uma instrução causa a alteração do Contador de programa então dois ciclos são necessários para completar a instrução.

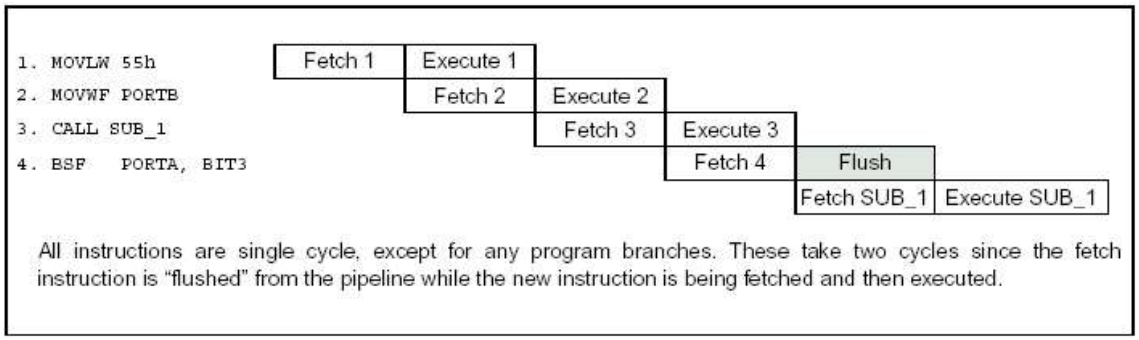


Figura 5 – Pipeline no microcontrolador PIC

A arquitetura em "*pipeline*" sobrepõe busca e execução, tornando a execução de instruções possível de se realizar em um único ciclo de máquina. Qualquer instrução de desvio (tais como GOTO, CALL, ou escrever no PC) leva dois ciclos de máquina.

A arquitetura com barramentos separados para instruções e dados permitem larguras diferentes, com isso o barramento de instruções é otimizado para uma palavra de comprimento única. O número de bits do barramento de instruções depende de quantas instruções são implementadas e do número de registradores disponíveis em cada família de microcontroladores.

5. Mapa de registradores

Todas as instruções aritiméticas e booleanas são feitas através do registrador de trabalho W. O destino da operação pode ser o próprio registrador W ou um dos registradores disponíveis no microcontrolador, dependendo unicamente da instrução executada.

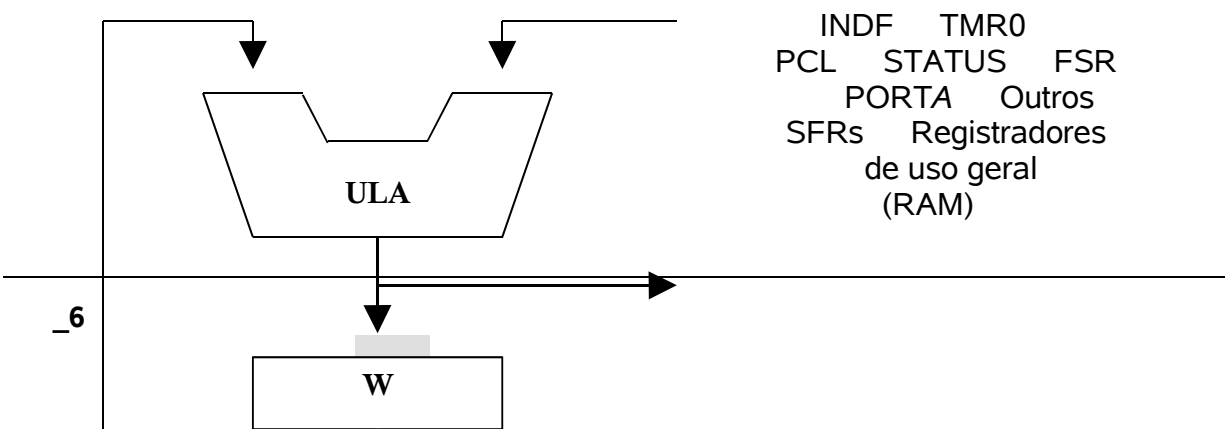


Figura 6 – Mapa de registradores

6. Interrupções

- Múltiplas fontes de interrupções interna/externa
- Prioridade de interrupção setada por software
- Habilitação global e individual das interrupções
- Diversas interrupções despertam o processador do estado de dormência (SLEEP)
- Tempo de latência da interrupção é fixada em 3 ciclos de instrução

7. Família 16F84A

Além das características gerais da arquitetura dos microcontroladores PIC vistas até agora, existem outros aspectos peculiares aos membros de cada família. Como a diversidade de componentes é muito grande, vamos analisar as características dos componentes da família 16F84A.

7.1. Arquitetura de Hardware da família PIC16F84A

Na figura abaixo é apresentada de forma simplificada a arquitetura interna da família PICF84. Ela é baseada em registradores, com o barramento de memória de dados separado do barramento de memória de programa (arquitetura RISC). Este conceito permite ter um conjunto de instruções simples, mas extremamente poderoso que enfatiza as operações bit, byte e de registradores.

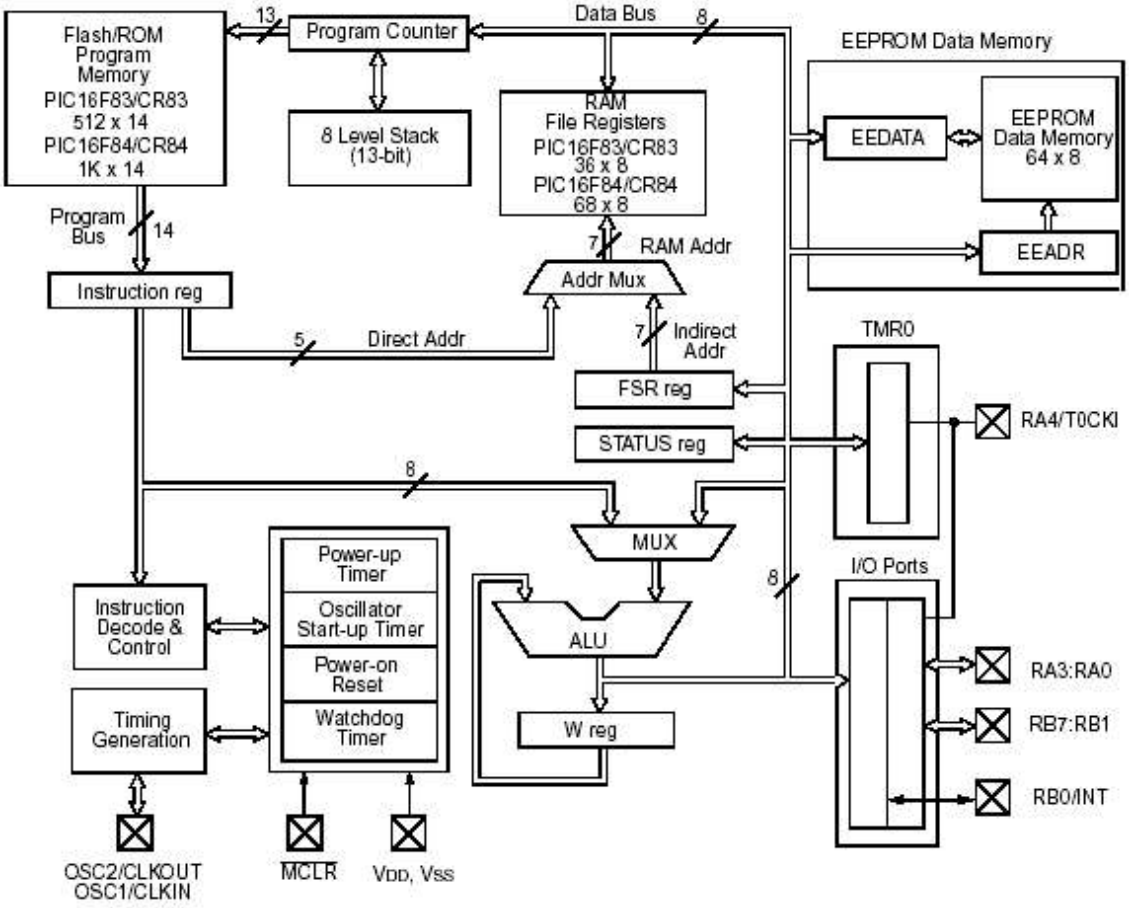


Figura 7 – Arquitetura de Hardware da família PIC18F8X

7.2. Principais membros da família

Device	Program Memory (words)	Data RAM (bytes)	Data EEPROM (bytes)	Max. Freq. (MHz)
PIC16F83	512 Flash	36	64	10
PIC16F84	1K Flash	68	64	10
PIC16CR83	512 ROM	36	64	10
PIC16CR84	1K ROM	68	64	10

Diversas freqüências de operação e empacotamento estão disponíveis. Dependendo da aplicação e dos requisitos de produção existe uma opção de

dispositivo mais apropriado para ser selecionado. Para maiores detalhes consulte o datasheet do componente que você utilizar.

7.3. Características Gerais

- Apenas 35 palavras de instrução para aprender
- Todas instruções com um ciclo exceto para desvios que levam dois ciclos
- Velocidade de operação: DC até 20 Mhz de clock
- Instruções com 14 bits de largura
- Barramento de dados de 8 bits
- 16 registradores de funções especiais de hardware
- Pilha com 8 níveis de profundidade
- Modos de endereçamento direto, indireto e relativo para dados e instruções.
- Capacidade de interrupção

7.4. Características dos periféricos

- 13 pinos de I/O individualmente configurados
- Temporizador/Contador de 8 bits com 8 bits de “pré-escala”
- Power-On Reset (POR)
- Temporizador Watch-Dog (WDT) com oscilador próprio para operações seguras
- Proteção de Código Programável
- Modo SLEEP para diminuição de consumo de energia.
- Opções de oscilador selecionável:
    - RC – oscilador RC de baixo custo
    - XT – cristal padrão
    - HS – Cristal de alta velocidade
    - LP – Cristal de baixa freqüência (redução de consumo)
- Programação Serial “in-circuit” (através de dois pinos)
- 4 bytes de identificação (ID) programáveis pelo usuário

7.5. Organização da Memória de Programa

A família PIC16F84A tem um contador de programa (PC) de 13 bits capaz de endereçar até 8K x 14 bits de memória de programas. O vetor de RESET está

localizado no endereço 0000h e o vetor de interrupção no endereço 0004h. Outra característica importante a ser salientada é a impossibilidade de se ler diretamente da memória de programa. A maneira com que isso é feito na arquitetura do PIC é utilizando a instrução “RETLW k”, que será visto mais adiante na apresentação do conjunto de instruções.

7.6. Organização da Memória de Dados

A memória de dados é composta de registradores e RAM para uso geral. Os registradores são divididos em 2 grupos funcionais: Registradores de Funções Especiais (32 endereços iniciais de cada banco) e Registradores de Uso Geral (endereços restantes de cada banco). Entre os registradores de funções especiais estão: o registrador TMR0, o contador de programa (PC), o registrador de Status (STATUS), os registradores I/O (ports) e o registrador de seleção (FSR). Além disso, os registradores de funções especiais são usados para controlar a configuração dos ports de I/O e as opções de pré-escala. Os registradores de uso geral são usados para dados e controle de informação sob comando das instruções.

7.7. Registradores de Funções Especiais

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 (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	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB	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

Figura 8 – Mapa de registradores especiais

7.7.1.INDF : Registrador de Endereçamento Indireto (Indirect Data Addressing Register)

O registrador INDF não é um registrador fisicamente alocado e é utilizado em conjunto com o registrador FSR para realizar endereçamento indireto. Qualquer instrução usando o registrador INDF acessa um dado cujo endereço está contido no registrador FSR.

7.7.2.TMR0: Relógio de Tempo Real (Real Time Clock/Counter)

O conteúdo desse registrador é sucessivamente incrementado utilizando um “clock”, que tanto pode ser interno (derivado do sistema de “clock” de

microcontrolador) quanto externo (aplicado ao pino RA4/T0CKI). Pode-se utilizar uma pré-divisão interna que permite ampliar (multiplicar) a contagem.

7.7.3.PC(PCL e PCLATH): Contador de Programa (Program Counter)

O registrador PC é responsável pela geração do endereço de busca à instruções na memória de programa. Normalmente é incrementado de uma unidade após a execução de uma instrução, com exceção das instruções que utilizavam desvios como GOTO e CALL. Como o PC tem 13 bits de largura, o byte menos significativo vem do registrador PCL, que é um registrador de escrita e leitura e os 5 bits mais significativos estão armazenados no registrador PCLATH.

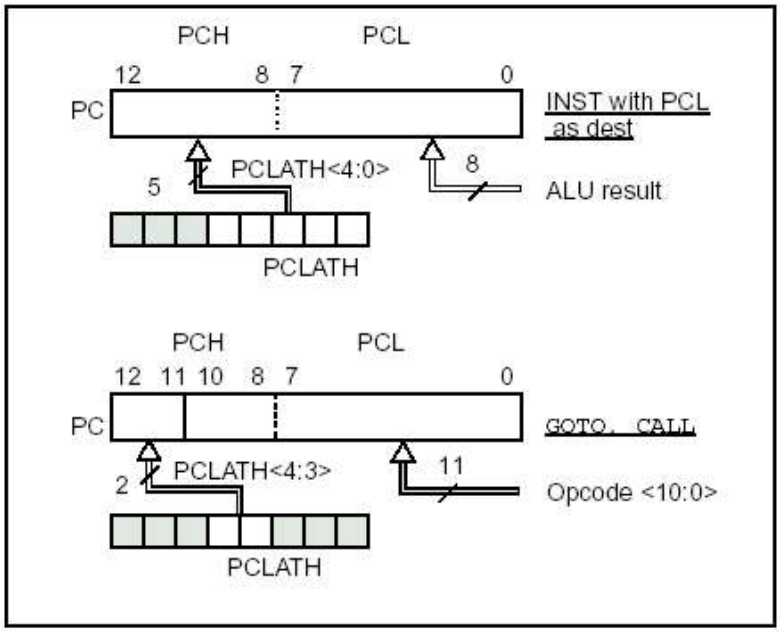


Figura 9 – Contador de programa

O PC tem todos os bits resetados durante o Reset. Durante a execução do programa ele é auto-incrementado juntamente com a execução da instrução, a menos que o resultado da instrução o altere. Uma rotina em software com GOTO computado é acompanhada da adição de um deslocamento (offset) ao contador de programas (ADDWF PCL).

7.7.4.STACK: Pilha (Stack)

A família PIC 16F84A implementa uma estrutura de pilha de oito níveis de profundidade por 13 bits da lergura. A área de pilha não faz parte da memória de programa, nem da memória de dados e também não pode ser lida ou escrita. O empilhamento é feito quando é chamada uma subrotina utilizando a instrução CALL. O desempilhamento é feito utilizando-se a instrução RETLW. Dados não podem ser

empilhados, prática muito utilizada em outras famílias de microcontroladores que não possuem arquitetura RISC.

7.7.5.STATUS

É o registrador que possui os flags que indicam resultados de operações da unidade lógica e aritmética (ULA), a condição de RESET e o bit de pré-seleção do banco de memória de dados.

Os bits:
C,
D
Z (bits 0, 1 e 2)
indicam o estado de uma operação aritmética na ULA.
Os bits:
P
D# (bit 3)
TO# (bit 4)
indicam o estado de Reset.
Os bits:
TO#
PD#
são somente para leitura, não são possíveis de serem alterados por software. Quando na apresentação do sistema de Reset e do Conjunto de Instruções iremos detalhar mais esse registrador.

7.7.6.OPTION

O registrador OPTION tem 6 bits de largura e contém vários bits de controle para configurar a pré-escala que será utilizada pelo TMR0 ou WDT. Nele também é definidos o valor da pré-escala e o tipo de contagem. Após um Reset, todos os bits são configurados com “1”.

7.7.7.Registrador de Trabalho

É o registrador mais utilizado, pois toda a transferência de dados é feita através dele. Além disso, todas as operações aritméticas e booleanas o utilizam como sendo um dos operandos. Ele é parecido com o acumulador de outras famílias de microcontroladores. Este registrador não é endereçável.

7.7.8.Registrador de Seleção (File Select Register)

Na família PIC 16F84X os Bits 0 a 6 selecionam um dos 128 registradores de uso geral disponíveis no modo de endereçamento indireto. O mapa de registradores de uso geral pode ser acessado tanto diretamente quanto indiretamente através do registrador de seleção (FSR). Uma instrução utilizando o registrador INDF vai operar com o endereço que está contido no registrador de seleção (FSR). Isto é um endereçamento indireto.

Por exemplo:

- Registrador 05 contém o valor 10h
- Registrador 06 contém o valor 0Ah
- Carregue o valor 05 no registrador FSR
- Uma leitura ao registrador INDF retornará o valor 10h
- Incrementa o valor do registrador FSR de um (FSR = 06)
- Uma leitura ao registrador INDF retornará o valor 0Ah

7.7.9.Registradores de I/O

Os Registradores de I/O podem ser escritos ou lidos sobre o controle do programa como qualquer outro registrador. Na condição Reset todos os I/Os são definidos como entrada, assim como todos os registradores de controle de I/O (TRISA e TRIS) são configurados com 1.

7.7.10.PORT*A*

Esse registrador tem correspondência direta com os pinos RA4:RA0 do microcontrolador. Ele possui somente 5 bits (RA0 a RA4); os demais 3 bits não são implementados e são lidos como zero. Cada bit desse PORT pode ser individualmente configurado como entrada ou saída. O pino RA4/T0CK1 é uma entrada Schimitt Trigger e uma saída com dreno aberto. O port RA4 é multiplexado com a entrada de clock T0Ck1. Todos os outros pinos de PORT*A* tem níveis de entrada Schimmit Trigger e drivers de saída CMOS completos.

7.7.11.16.9.2. PORTB

Esse registrador tem correspondência direta com os pinos RB7:RB0 do microcontrolador. Ele tem 8 bits de largura e é bidirecional. Cada bit desse PORT pode ser individualmente configurado como entrada ou saída. Cada pino PORTB tem um *pull-up* interno.

Um bit de controle RBPU# (OPTION <7>) pode ligar os pull-ups. O pull-up é automaticamente desligado quando os pinos são configurados como saídas. Os pull-ups também são desabilitados no Power-on Reset. Quatro pinos do PORTB, RB7:RB4 tem uma característica de interrupção na mudança de estado. Apenas os pinos configurados com entrada podem causar esta interrupçaõ. Os pinos de entrada (RB7:RB4) são comparados com o valor antigo armazenado no latch, na última leitura do PORTB.

7.7.12.TRISA e TRISB

São os registradores referentes à configuração dos pinos de I/O da PORTA e PORTB. Escrever 1's nos registradores TRISA e TRISB fazem dos bits entradas, colocando o driver de saída em alta impedância. Escrever 0's nesses registradores fazem dos bits saídas, colocando o conteúdo da saída nos latches dos PORTs correspondentes. Após um Reset, todos os registradores são configurados com 1's, ou seja, todos os pinos de I/O configurados como entrada.

7.8. Interface de I/O

Todos os pinos de I/Os podem ser usados tanto como entrada quanto saída. A sua direção é definida pelos registradores de direção TRISA e TRISB. Cada bit desses registradores corresponde a um pino de I/O que, quando setado, corresponde à entrada e, quando resetado, corresponde à saída. Por exemplo, se quisermos setar o PORTB com o nibble menos significativo como entrada e o mais significativo como saída, então o valor a ser escrito em TRISB será 00001111 em binário, ou "0F" em hexadecimal.

Nota-se que o dado de saída se mantém armazenado em um flip-flop independente do pino de I/O estar configurado como entrada ou saída. Esse dado se mantém nessa condição até que um novo dado seja escrito. Para o processo de leitura, o dado tem que estar estabilizado e o pino configurado para entrada pois ela não é armazenada em flip-flops.

Devido a essas características, o programador deve ter alguns cuidados quando usar instruções do tipo "read and write modified" como BSF ou BCF, que lêem o PORT de I/O, executam a aoperação no bit e escrevem o resultado no PORT de I/O. Por exemplo, uma instrução BSF no pino 5 do PORTB irá provocar a leitura dos 8 pinos de I/O do PORTB. Então, a CPU irá executar a operação de setar o bit 5 e o resultado será escrito no PORTB. Se outro pino de I/O for usado como bidirecional, e nesse instante estiver configurado para entrada o sinal presente será lido e reescrito sobre o dado que estivesse previamente escrito no latch de saída do pino de I/º Se o pino estiver configurado como entrada não haverá nenhum problema, mas se ele estiver configurado para saída, o dado no latch de saída pode ser desconhecido. O que o programador deve ter em mente é nunca provocar um "curto-circuito" nos pinos de I/O . Por exemplo, se no pino de I/O estiver conectado a GND e no latch de saída estiver setado para "1", com o pino e I/O configuradao para saída, uma alta corrente irá circular, o que danificará o componente.

7.9. Temporizador

O módulo temporizador/contador (TMR0) tem as seguintes características:

- Temporizador/contador de 8 bits
- Pode-se utilizá-lo para leitura e escrita
- Pré-escala de 8 bits programável por software
- Seleção de clock interno ou externo
- Seleção do tipo de borda para clock externo

- Interrupção no estouro de FFH para 00H

O modo temporizador é selecionado fazendo o bit T0CS = 0 (Timer 0 Chip Select – OPTION <5>). No modo temporizador, o registrador TMR0 será incrementado a cada ciclo de máquina (sem pré-escala). Se ocorrer uma escrita em TMR0, o incremento será inibido pelos dois ciclos de máquina seguintes. O modo contador é selecionado fazendo o bit T0CS = 1. Nesse modo, TMR0 será incrementado a cada borda de subida ou descida do pino RA4/T0CK1. A borda responsável pelo incremento é determinada pelo bit T0SE (Timer 0 Select Edge – OPTION <4>). O bit T0SE = 0 seleciona borda de subida e T0SE seleciona borda de descida.

A pré-escala é compartilhada entre o módulo TMR0 e o Watch-Dog Time (WDT). A pré-escala é controlada no software pelo bit de controle PSA (PreScaler Assignmet – OPTION <3>). Quando o bit PSA= 1, a pré-escala fica sob controle do TMR0 e a pré-escala do WDT é selecionada para 1:1. A pré-escala é setada por software através dos bits PS2:PS0 (OPTION <2:0>) com valores de seleção que variam de 1:2, 1:4, ..., 1:256. O uso da pré-escala pode ser para o WDT ou para o TMR0, mas nunca simultâneo. Nada impede que na primeira fase do programa se utiliza a pré-escala para o WDT e na segunda fase para o TMR0. O WDT tem oscilador próprio capaz de gerar um tempo 18ms de período sem a pré-escala. Com pré-escala de 1:128 o tempo sobe para 2,5 segundos. A interrupção do Timer 0 é gerada quando o conteúdo do registrador TMR0 passar de FFH para 00H. Esse estouro faz T0IF =1. A interrupção pode ser mascarada fazendo T0IE = 0 (Timer 0 Interrupt Enable – INTCO<5>). O bit T0IF deve ser resetado por software na rotina de serviço da interrupção antes de reabilitar a interrupção. A interrupção do Timer 0 não pode acordar o processador do mod SLEEP porque o timer é desligado antes de entrar no estado de dormência.

7.10.Interrupções

As interrupções disponíveis nas famílias PIC 16/17 variam de componente para componente, dependendo das características de hardware implementadas. As famílias PIC 16/17 possuem 3 fontes de interrupção implementadas. São elas:

- Interrupção Externa RB0/INT
- Interrupção Temporizador TMR0 (estouro na contagem)
- Interrupção por mudança no PortB(pinos RB7:RB4)

O registrador de controle de interrupção (INTCON) armazena os pedidos individuais de interrupção em seus bits. Ele também contém bits de habilitação individual e global de interrupções. O bit de hebilitação global de interrupções, GIE (INTCON <7>) habilita (se setado) todas as interrupções não mascaradas ou desabilita (se resetado) todas as interrupções. As interrupções individuais podem ser desabilitadas através do seu correspondente bit de habilitação no registrado INTCON. O bit GIE é resetado durante o Reset.

A instrução “Retorno da Interrupção”, RETFIE, sai da rotina para desabilitar futuras interrupções, o endereço de retorno é colocado na pilha e o registrador PC é carregado com 0004H. Uma vez na rotina de serviço da interrupção a(s) fonte(s) de

interrupção podem ser determinadas pelo teste das flags correspondente no registrador INTCON. O bit da interrupção atendida deve ser resetado no software antes de reabilitar as interrupções para evitar interrupções recursivas.

7.11.Conjuntos de Instruções de Software

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	

Figura 10 – Instruções do microcontrolador PIC16F8X

8. Prática

8.1. Introdução

A Microchip disponibiliza ferramentas para desenvolvimento de aplicações para seus microcontroladores PIC. Dentre estas ferramentas podemos citar o MPLAB e o PICStart. O MPLAB é um IDE que fornece uma integração entre editor de texto, compiladores e microcontroladores.

Podemos também, utilizar outras ferramentas de desenvolvimento, na Internet encontramos diversas delas. Vamos utilizar nesta experiência a ferramenta *P16Pro Programmer for MicroChip microcontrollers*, ela disponibiliza além do software, um gravador de microcontroladores PIC (anexo 01). Através do software P16pro e do circuito de gravação, pode-se carregar um programa escrito em hexadecimal no microcontrolador. Diversos parâmetros podem ser configurados no software para programar diferentes versões de microcontroladores PIC. Utilize o manual da ferramenta para maiores detalhes da utilização do software P16pro.

8.2. Gerar uma onda Quadrada em uma porta do PIC

Vamos programar o PICF84A para gerar uma onda quadrada. Abra um editor de texto e escreva o código abaixo, não esqueça de respeitar as tabulações antes de cada comando e seus operandos.

```
#include       "p16f84.inc"
       LIST P=16F84

count equ 0x0C

;inicializacao
;programa principal
INI    BSF 03h,5;
       MOVLW 00;
       MOVWF 06;
       BCF 03h,5;

LOOP BSF 06,0
       CALL DELAY
       BCF 06,0
       CALL DELAY
       GOTO LOOP

DELAY
       MOVLW 9fh
       MOVWF 1A
       ;MOVWF 1B
```

```
DELAYAUX
        ;DECFSZ 1B
        ;GOTO DELAYAUX

        DECFSZ 1A
        GOTO DELAYAUX

        RETLW 00

        END
```

Em seguida, transcreva o código para hexadecimal, para isso, submeta o arquivo .ASM que você acabou de criar para o programa MPASM302.exe. Agora vamos gravar o código hexadecimal no PIC:

a. Conecte o gravador de microcontroladores na porta paralela do PC.
b. Posicione o PIC no gravador, verificar posição do chanfro.
c. Ligue a fonte do gravador.
d. Abra o programa P16pro.
e. Selecione o PIC16F84A: Device (F3).
f. Carregue o programa gravado na memória do microcontrolador: Read (F6).
g. Apague o conteúdo da memória: Erase (F9).
h. Verifique se a memória está vazia: Blank Check (F7).
i. Abra o programa escrito em hexadecimal: OpenProg (F1).
j. Selecione o oscilador XT: Fuses (F2).
k. Grave o programa escrito em hexadecimal no microcontrolador: Program (F4).
l. Verifique a gravação e a opção do oscilador: Read (F6).
m. Desligue a fonte do gravador.
n. Retire o PIC do gravador, tome cuidado com os pinos.

Para verificar o funcionamento do programa, vamos implementar um circuito de teste (figura 11). Este circuito deve ligar a porta RB*0* do microcontrolador a um led. Deve-se também ligar um oscilador ao PIC, ele será à base dos seus ciclos de instrução (ver manual). O oscilador será um cristal, com freqüência e configuração de montagem especificada no manual do PICF84A [2].

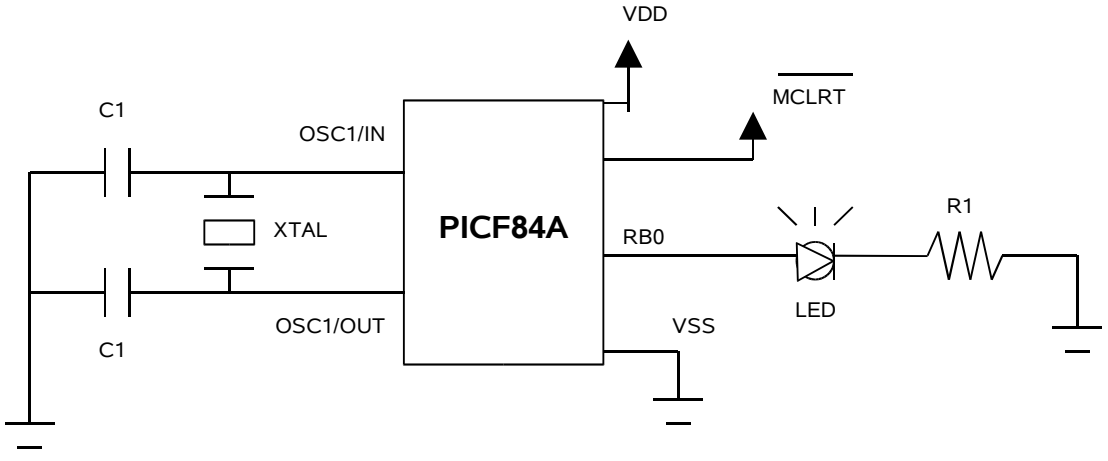


Figura 11 – Circuito de teste

Configurações do Circuito [2]	
C1	15 - 33 pF
XTAL	4 - 10 MHz
R1	1KΩ

9. Bibliografia

- [1] Microchip PIC16F8X Datasheet – www.microchip.com
- [2] Projetos de Hardware e Software utilizando Microcontroladores PIC – Edmur Canazian - 1999
- [3] P16Pro Programmer for MicroChip microcontrollers – Short Manual
- [4] The PIC Microcontroler Book