

INTRODUÇÃO AO USO DE PIC's ATRAVÉS DE LINGUAGEM BASIC

Escrito por : Eng. Antonio Paulo Hawk

Quantas e quantas vezes você quis fazer um projeto de eletrônica com um PIC ?

E quantas vezes você percebeu que não tinha nada que o ajudasse a começar o seu aprendizado ?

E quantas vezes você procurou ajuda nos Fóruns de Eletrônica, onde invariavelmente lia que "você tinha de aprender o Assembler e usar o MPLAB" , ou pior ainda, "você tem de aprender a Linguagem C" ?

Pois é, eu sei que isso acabava sempre te desestimulando, afinal, para alguém que apenas quer usar um PIC como se fosse um simples CI igual aos outros, aprender Assembler é algo terrível, ainda mais por conta própria !

E convenhamos, para alguém que está iniciando, ver uma listagem de um programa em "C" é ainda pior, pois aparentemente não existe lógica nenhuma na sintaxe do programa, pois os programadores costumam minimizar os comandos !!!

Digo aparentemente, claro, pois para quem está acostumado com o "C" , nada é mais coerente do que esta linguagem.

Porém, para simples "amadores" como nós, deveria existir um processo mais simples de usar os PIC's em nossos projetos. **E existe !!!!!!!!!!!!!** Sim, fique feliz meu amigo !!!

Após pesquisar um pouco na Internet, descobri uma ferramenta maravilhosa, com um enorme poder, fácil de usar, com versão demo disponível, e um baixo custo se resolvermos adquirir oficialmente o programa. Chama-se PIC SIMULATOR IDE !

Afinal, o que é esse **PIC SIMULATOR IDE** ou abreviado por **PSI** ?

Esse programa na verdade é um ambiente quase completo de desenvolvimento, faltando apenas as funções de um GRAVADOR DE PIC.

O **PSI** (vamos abreviar assim o nome do programa para facilitar) compreende as seguintes funções :

1. Compilador BASIC
2. Simulador de programa
3. Simulador de alguns Hardwares básicos para uso de PIC
4. Montador Assembly in-line
5. Debugador de programa

Ou seja, podemos fazer o nosso programa usando comandos em linguagem BASIC que são muito intuitivos, e como todos sabem, BASIC é a linguagem mais fácil de se aprender.

Melhor ainda, podemos fazer o programa, acrescentar os hardwares existentes no PSI, compilar o programa, e ver o que acontece com as entradas e saídas de todos os hardwares, durante a execução do programa em BASIC !!!!!!!

Antes de falar mais sobre o PSI (Abreviatura do PIC SIMULATOR IDE) , vou primeiro falar sobre o funcionamento básico de um PIC, e de que maneira fazemos a sua famosa "Configuração do Hardware de um PIC" .

Depois, explicarei sobre o PSI, suas maravilhosas funcionalidades, e darei alguns exemplos de projetos e programas em BASIC e sua simulação de funcionamento.

Aos mais apressados, sugiro que não pulem nenhuma parte deste tutorial, pois você pode não ter os resultados esperados se não tiver todo o conhecimento que será ensinado aos poucos durante este tutorial, portanto, meus amigos, LEIAM BEM, e se tiverem dúvida, LEIAM NOVAMENTE !!!

CONHECENDO MAIS PROFUNDAMENTE UM PIC

Afinal, o que é um PIC, e porque toda a industria de equipamentos usa esses componentes ?

Resumindo muito , um PIC é um microprocessador que possui em seu hardware um monte de periféricos, tais como memória Flash, memória EEPROM, memória RAM, portas de entrada e de saída, circuito gerador de PWM, conversores de tensão Analógico - Digitais, portas de comunicação de vários tipos e protocolos inclusive USB, osciladores, ufa..... já dá para ter uma idéia do poder que esta pequena maravilha possui !

Embora tenha tudo isso dentro, o PIC geralmente não possui mais do que 40 Pinos !!!!
Sim, isso mesmo, existem PICS que tem desde 6 pinos até 40 pinos ou mais, onde temos um montão de hardwares diferentes já integrados dentro dele !!!!!

Vamos aqui fazer uma comparação :

O seu computador onde você está lendo este tutorial é um excelente exemplo para começar a entender as diferenças.

O coração de seu PC é composto por uma placa mãe , que aceita uma variedade enorme de processadores, e permite acrescentar também a memória RAM para uso dos programas.

Vamos olhar primeiro o processador do seu PC : ele pode ter quase 1000 pinos !!!!!!!
Mesmo assim, ele é só o Microprocessador, mais nada, todo o restante do hardware que permite ele operar, que contém a memória RAM, a Eeprom com o Bios, o gerenciador de memória RAM, os slots de expansão para colocarmos placas de I/O, etc, tudo está na Placa Mãe (MOTHERBOARD) !!

Quanto custa o seu Processador que está em seu PC ? Digamos que varia desde uns R\$ 70,00 até mais de R\$ 1.500,00 !!!!! E, lembre-se, ele não inclui nenhum periférico dentro dele mesmo !!!!!!! Só serve para processar, mas é totalmente dependente da Motherboard para ser útil de alguma maneira !

Agora, olhe para um PIC ele não tem nem de longe o poder de processamento de um processador atual, aliás quase todos os processadores que eram usados em um PC nos anos 90 já eram mais potentes que os mais potentes PICs que temos hoje.

Mas

O PIC tem algo que quase nenhum outro processador tem : UM MONTÃO DE HARDWARE PRONTO DENTRO DELE !

Podemos fazer maravilhas com um simples PIC e alguns componentes simples colocados juntos !

E o melhor, o custo de um PIC vai desde uns R\$ 4,00 até uns R\$ 100,00 (alguns podem custar mais, mas é raro) !!!!

Para concluir esta comparação, vou dar um exemplo de um PIC muito usado hoje em dia, que é o **PIC 16F877A** . Vamos ver o que temos dentro dele :

- Possui 40 pinos
- Memória de programa de 14.3 Kbytes comportando até 8.192 instruções
- Memória RAM de 368 bytes
- Memória EEPROM de 256 bytes
- até 33 pinos para I/O
- até 8 conversores A/D de 10 bits
- 2 módulos PWM com resolução até 10 bits
- Interface para comunicação serial SPI
- Interface para comunicação serial IIC
- USART para comunicação serial comum
- 1 Timer de 8 bits
- 2 Timers de 16 bits
- 2 comparadores analógicos
- Circuito de Watchdog
- Clock de 0 até 20 MHz

Este componente pode ser encontrado no mercado nacional com preços em torno de **R\$ 15,00** !!!!

Agora, como que é possível usar tudo isto ao mesmo tempo se só temos 40 pinos ????

A resposta é bem simples: **NÃO É POSSÍVEL !**

Temos de escolher exatamente quais os módulos que iremos usar em nossa aplicação, por exemplo, se quisermos usar os conversores A/D com leituras entre 2 tensões de

referência, teremos de usar no máximo 6 conversores !!!! 2 deles perdemos para fazer as tensões de referência Vref- e Vref+ .

Da mesma maneira, se quisermos utilizar 33 pinos como portas de Entrada / Saída Digitais, não teremos mais nenhum conversor A/D, nem as interfaces de comunicação seriais.

Pelo exposto acima, você já deve ter percebido que existem limites definidos no hardware que dizem o quê que podemos usar ao mesmo tempo.

E como que dizemos ao PIC como queremos que ele funcione, por exemplo, que ele tenha 1 porta de 8 bits de saída, e uma porta de entrada com 5 bits ; que ele tenha 4 portas analógicas que irão converter tensões existentes dentro de uma faixa de tensões que definiremos nos pinos VREF- e VREF+ ; que vamos usar um cristal oscilador de 20 MHz, e que vamos usar a USART para nos comunicarmos com um PC ?????

A este procedimento é que chamamos de "Configuração de Hardware de um PIC".

Quando escrevemos um programa, consideramos que o PIC já está configurado para ter todo o hardware que queremos usar.

E como que o PIC é configurado ? Algumas funções são configuradas quando gravamos o nosso programa nele !!!

Sempre que vamos gravar o nosso programa, são gravados no PIC os comandos de configuração especiais de hardware, assim quando ele é alimentado , ANTES DE RODAR O PROGRAMA ARMAZENADO DENTRO DELE, esses comandos são executados e preparam o hardware para funcionar de acordo com o nosso programa !

Estes comandos de inicialização são encontrados nos Datasheets dos PIC's, e o nome que o fabricante deu a eles é **CONFIGURATION WORDS**.

Os PICS podem ter várias Configuration Words, a maioria deles tem apenas uma.

Além destas Configuration Words, existem outros registros que instruem o hardware a funcionar como queremos, mas estes registros estão acessíveis ao nosso programa, portanto podemos mandar trabalhar da maneira que queremos, e podendo mudar esta maneira conforme nossa vontade ao longo do programa.

Por exemplo, configurar uma porta de I/O para que alguns pinos sejam saídas, e outros pinos sejam entradas, e configurar para que alguns dos pinos de entrada gerem interrupção, isto também faz parte do que chamamos de "Configuração de Hardware de um PIC" .

A minha experiência em ensinar o uso dos PIC's mostrou que é justamente neste procedimento acima onde os iniciantes tem maior dificuldade de compreender o "porquê" das coisas, mas isto se deve principalmente ao fato de existirem disponíveis na Internet muitos exemplos de programas em Assembler, que para fazer isto tudo abusam das comutações de Bancos dos PIC's, e realmente complica muito o iniciante que

acaba sem entender o motivo de tantos comandos seguidos mudando os bancos dos Pícs.

Por hora, o que importa aos iniciantes é que conheçam muito bem *o que que o hardware existente em um PIC pode fazer*, apenas isto, *não se preocupem em COMO FAZER ISTO*, pois é justamente aí que entra o **PIC SIMULATOR IDE** e sua linguagem BASIC !

Recomendo que peguem um Datasheet do PIC 16F877A, do 12F629 e do 16F628A, e conheçam em termos gerais o funcionamento de cada módulo de hardware do PIC.

Este conhecimento é fundamental para que continue este tutorial, pois logo iremos programar o PIC com a linguagem BASIC, e você deverá saber pelo menos o que que os comandos de configuração estão fazendo no hardware do PIC.

Você vai reparar que a Microchip explica como funciona o PIC, e como programar o hardware dele, **sem que** para isso seja necessária a linguagem Assembler !!!!!!!

Portanto, agora você já pode ficar mais sossegado, pois vai aprender a usar o PIC sem grandes dificuldades e logo ficará muito satisfeito em ver os seus programas funcionarem !

CONSIDERAÇÕES INICIAIS PARA O PIC SIMULATOR IDE

Você pode baixar a versão demo do programa, indo até a página de download, e selecionando o programa PIC SIMULATOR IDE. Esta versão permite você aprender o uso do programa, mas tem 2 limitações importantes :

- Somente podemos selecionar alguns modelos de PIC
- Permite apenas 50 linhas no programa em BASIC

Entre os modelos que podemos usar nesta versão, citamos o 16F84 e o 16F877. Sugerimos a você fazer o download destes datasheets, pois nosso tutorial vai utilizar estes dois modelos. Seguem os links para o download :

PIC SIMULATOR IDE : <http://www.oshonsoft.com/downloads.html>

PIC 16F84 : <http://ww1.microchip.com/downloads/en/DeviceDoc/30430c.pdf>

PIC 16F877 : <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

Se você possui o **Windows Vista**, após a instalação, vá até o executável do programa que deve estar em **C:\Program Files\PIC Simulator IDE\picsimulatoride.exe** e clique com o botão direito do mouse, escolha PROPRIEDADES, escolha a aba COMPATIBILIDADE, e clique em **Executar este programa como Administrador**, e clique em **OK** . Pronto, seus problemas acabaram.

Com tudo baixado e instalado, vamos começar o nosso tutorial.

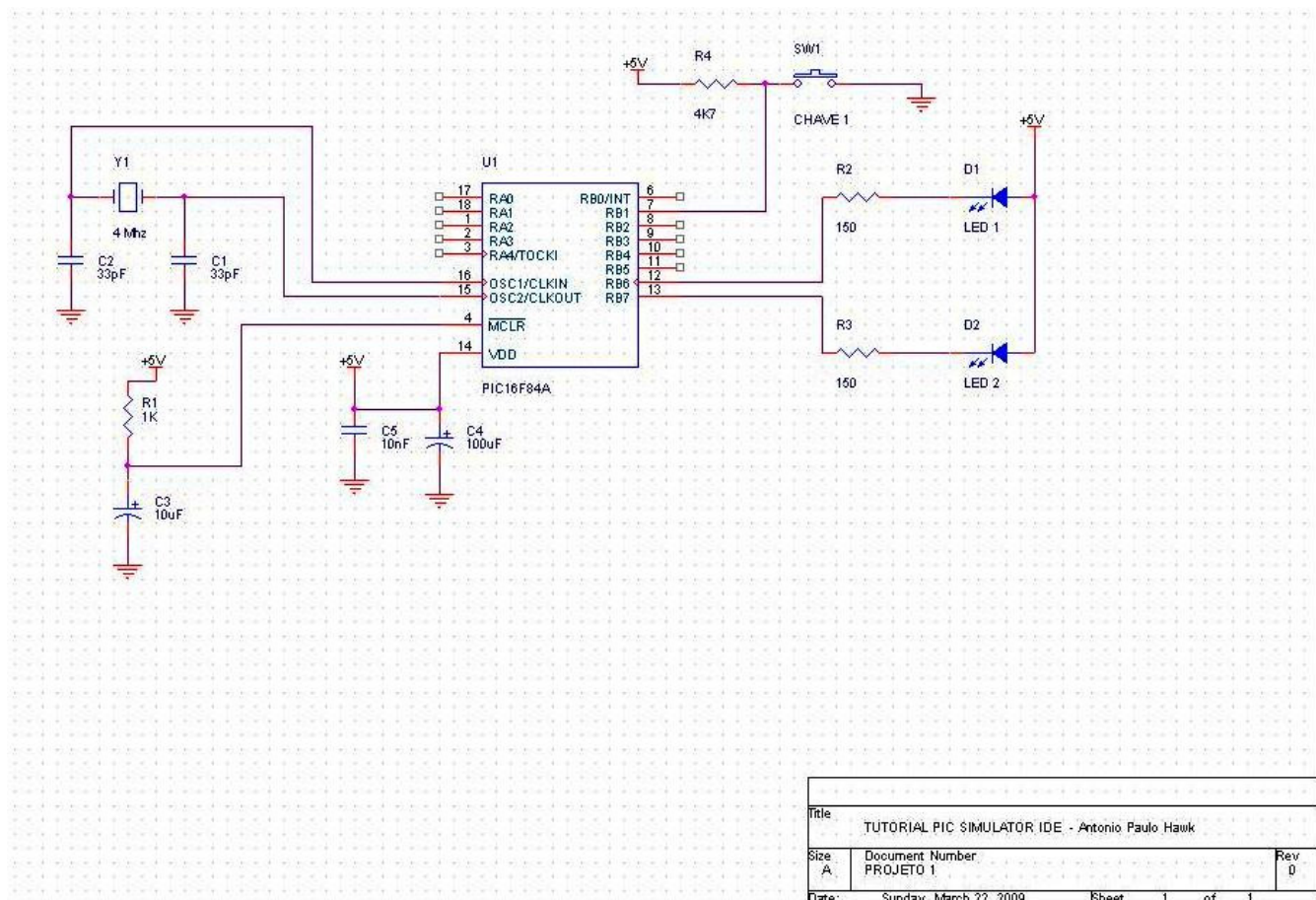
USANDO O PIC SIMULATOR IDE EM PROJETOS REAIS

Para iniciar a parte prática, vamos inicialmente fazer um projeto bem simples, usando o PIC16F84A (não se preocupe, embora vamos fazer tudo no PSI como PIC16F84, na prática você vai usar o PIC16F84A, o software é o mesmo, você pode gravar o seu programa em qualquer um desses dois modelos de PIC.).

PROJETO 1 : PISCA-PISCA SIMPLES

Neste projeto, queremos simplesmente implementar um pisca-pisca simples, com 2 leds, de maneira que quando um led acende, o outro apaga, e vice- versa. Embora tenhamos colocado uma chave, vamos usá-la apenas no próximo projeto. Basta apenas ligar a alimentação e pronto, os leds saem piscando, a uma velocidade de 2 hertz, ou melhor, cada led pisca 2 vezes por segundo.

Segue o circuito para esta aplicação :



IMPORTANTÍSSIMO : LIGUE TAMBÉM O PINO 5 DO PIC16F84A AO TERRA !!!

É um circuito muito simples, que funciona com uma simples fonte de +5 Volts com corrente de 50 mA.

Vamos apenas deixar claro a função dos componentes acima :

- R1 e C3 são responsáveis pelo RESET inicial do PIC.
- C1, C2 e Y1 são o circuito do oscilador a cristal , de 4 MHz.
- C4 e C5 são para filtragem e desacoplamento da alimentação.
- R2 e R3 são limitadores de corrente para cerca de 20 mA .

O importante é saber, a partir de nosso esquema, o que que vamos informar ao PSI sobre o nosso circuito :

- tipo do oscilador é XT com frequência de 4 MHz. Caso você tenha alguma dúvida, pode padronizar o seguinte : Para cristais entre 1 e 4 MHz, sempre use XT, e para frequências maiores, use HS.
- Usamos a porta B pino RB1 (7) como entrada (INPUT).
- Usamos a porta B pinos RB6 (12) e RB7 (13) como saída (OUTPUT).
- Ah, o mais importante de tudo : usamos o PIC16F84.

Portanto, para podermos ver a simulação de nosso projeto, precisamos abrir as janelas do Microprocessador, a barra de Leds, e pronto. Vamos fazer tudo do início, para que você possa fazer sem nenhum problema.

CONFIGURAÇÃO INICIAL DO PIC SIMULATOR IDE

Vamos configurar agora o PSI para que você possa usar em vários projetos deste tutorial. Recomendo que você não mexa nas outras opções enquanto não estiver bem preparado no uso deste programa.

Clique em OPTIONS , SELECT MICROCONTROLLER e escolha o PIC 16F84 e clique em OK.

Clique em OPTIONS, CHANGE CLOCK FREQUENCY, e digite a frequência de nosso cristal, que no caso é de 4 MHz. Digite 4 e clique em OK.

Agora, vamos fazer a configuração dos bits. Para cada processador, existe um registro de configuração inicial, que é onde configuramos o hardware do PIC. Para o 16F84, temos apenas 4 itens a configurar. Vou explicar elas brevemente e sugiro que você não se preocupe muito com isto.

1. CODE PROTECTION - Aqui escolhemos se permitimos que o programa gravado seja lido por um gravador, ou não. Imagine que você não queira que ninguém consiga copiar o seu programa, basta selecionar ON e pronto, será impossível alguém copiar o seu PIC.
2. POWER-UP TIMER - É um sistema de RESET inicial, que mantém o PIC inoperante durante alguns milisegundos mesmo após a alimentação ser ligada. É uma prevenção, que garante que o restante do nosso circuito esteja totalmente funcional quando o PIC começar a executar o programa. Para isso, basta selecionar ENABLED.
3. WATCHDOG TIMER - É um sistema que permite fazer um RESET automático em caso de nosso programa estar "perdido" em algum loop infinito, ou tenha ocorrido alguma falha drástica no hardware. Funciona da seguinte maneira, nós

programamos um período, por exemplo, 50 milissegundos, e em nosso programa, nós temos de ficar "zerando" esse contador antes que esse prazo termine. Quando zeramos o contador, temos de zerar novamente em menos de 50 milissegundos, e assim por diante. Caso não ocorra esse zeramento, seja qual for o motivo, ocorre um RESET, e o programa vai reiniciar do zero novamente. É uma maneira de garantir que o PIC não fique parado sem fazer nada, ou, pior ainda, se ocorrer alguma coisa que nós não previmos no programa e ele sair do nosso controle, ocorrerá esse RESET. Se quiser ativar esta função, basta selecionar ENABLED. Mas já aviso que para os iniciantes, esta função deve estar desligada.

4. OSCILLATOR SELECTION - É aqui que definimos qual o tipo de oscilador que vamos usar. Como já disse acima, devemos selecionar XT.

Resumindo, para este nosso projeto, vamos deixar assim :

- CODE PROTECTION - ON
- POWER-UP TIMER - ENABLED
- WATCHDOG TIMER - DISABLED
- OSCILLATOR SELECTION - XT

Configure como está acima descrito, e em seguida clique APPLY e logo após GENERATE BASIC CODE. Isto fará com que a CONFIGURATION WORD seja automaticamente declarada em nosso programa BASIC, sem que tenhamos de nos preocupar ! Fácil não ?

Após isto, clique em CLOSE para fechar esta janela. Só para conferir, abra a janela do compilador BASIC para ver o comando colocado na linha 1 : clique em TOOLS, e depois em BASIC COMPILER, e abrirá uma janela com as linhas de nosso programa em BASIC, claro que no início teremos apenas esta primeira linha - DEFINE CONF_WORD = 0X0001 . Após isto, feche a janela do compilador, para terminarmos as configurações do PSI.

Agora, clique novamente em OPTIONS e deixe selecionada as seguintes opções :

- COMPACT MICROCONTROLLER VIEW
- BASIC PROGRAM TRACKING
- SHOW RAM MEMORY USAGE INFORMATION
- USE VOLTAGE FOR ANALOG INPUTS
- CONTINUOUS ANALOG INPUT SLIDER UPDATE

Vou dar uma rápida explicada nestes settings que fizemos acima :

COMPACT MICROCONTROLLER VIEW - Quando pedirmos para o PSI mostrar o PIC para que possamos interagir e ver os valores lógicos (e analógicos) nos pinos do PIC, aparecerá uma tela mais compacta, que ocupa menos espaço na tela do seu computador.

BASIC PROGRAM TRACKING - Quando rodarmos o programa em BASIC, poderemos acompanhar o que ocorre , instrução por instrução. Assim fica mais fácil de acompanhar o que que nosso programa está fazendo, e assim podemos achar qualquer erro no programa muito mais rápido.

SHOW RAM MEMORY USAGE INFORMATION - Quando compilamos o nosso programa em BASIC sem nenhum erro, ao final será apresentada uma tela que contém estatísticas sobre o uso da memória RAM e da memória de programa, assim sempre sabemos o quanto ainda temos disponível para uso.

USE VOLTAGE FOR ANALOG INPUTS - Quando usarmos algum sinal analógico no conversor A/D , podemos informar ao PSI qual o valor que será lido no pino que queremos usar. Para isso, existem 2 maneiras, a primeira é colocar diretamente um valor entre 0 e 1023, que será o valor que será lido, ou colocar uma tensão nesse pino, cujo valor será entre 0 e 5 Volts. Como na grande maioria dos casos de simulação usamos tensões aplicadas nas entradas, é melhor que possamos escrever direto qual a tensão que está aplicada no pino. Mas lembre-se, o resultado da leitura sempre será um número entre 0 e 1023, pois quase todos os PIC's possuem conversor A/D de 10 bits.

CONTINUOUS ANALOG INPUT SLIDER UPDATE - Em vez de digitarmos diretamente a tensão, aparecerá uma barra pequena para que possamos escolher o valor de tensão. é como se fosse um potenciômetro retilíneo (slider), e que torna mais fácil a simulação de nosso programa, pois podemos variar a tensão de entrada a qualquer momento mesmo com o programa rodando , assim podemos ver imediatamente os efeitos da mudança nos valores.

Para terminar estas configurações iniciais, clique novamente em OPTIONS, e certifique-se que estão selecionadas as seguintes opções :

- SAVE POSITIONS
- SAVE ALWAYS ON TOP

Agora pode fechar a janela principal do programa PSI, pois ele vai manter esta configuração sempre que você iniciar novamente o programa.

Quando você for simular um programa, existirão muitas janelas abertas ao mesmo tempo, e se você resolver fechar a janela principal do PSI, as janelas abertas serão memorizadas pelo programa.

Quando você abrir novamente o PSI, todas as janelas voltarão a se abrir sozinhas nos mesmos lugares em sua tela. Assim você pode imediatamente continuar de onde parou.

Isto encerra a configuração básica do PSI. A seguir vou mostrar como vamos acrescentar o nosso "hardware" nos pinos do PIC para fazermos nossa simulação.

PREPARANDO A SIMULAÇÃO ADICIONANDO "HARDWARE"

Como vimos acima em nosso primeiro projeto, que é um simples pisca-pisca, precisamos acrescentar dois LED's e um botão , que serão ligados nos pinos do PIC.

Como é apenas um botão de entrada, não precisamos colocar nenhum hardware de entrada, pois a própria tela onde vemos o PIC possui um local para selecionarmos qual o valor lógico em qualquer pino de entrada do PIC.

Mas, para visualizarmos os LEDS, temos de acrescentar um hardware, que o PSI já possui pré-montado no programa : 8 LEDS juntos, os quais podemos selecionar quais as cores que queremos, e em que pinos do PIC eles serão ligados. Vamos lá :

Clique em Tools, e em seguida escolha **8X Led Board**, e aparecerá uma janela contendo 8 LEDS . Posicione esta janela onde você achar mais conveniente.

Como funciona esta janela de Leds ?

Você tem 8 Leds, e pode ligar este LED a qualquer pino das portas de I/O do PIC, e quando esse pino tiver o estado lógico 1 (tensão de +5 Volts) , o LED correspondente a este pino irá acender.

LEMBRE-SE DE QUE NO PSI O LED SÓ ACENDE QUANDO A SAIDA DO PINO FOR NIVEL ALTO (1) !!!!!

Agora, vamos acrescentar a janela que tem o "desenho" dos pinos do PIC, pois é nela que podemos mudar o valor das entradas. Vamos lá :

Clique em Tools, e escolha **Microcontroller View**. Aparecerá uma pequena janela com os pinos do PIC, e com mais duas informações em cada pino. Uma das informações é o valor lógico da entrada, que no caso do nosso PIC 16F84 é tudo digital, e portanto o valor de cada pino de I/O só pode ser **OFF** (nível 0) ou **ON** (nível 1) .

Repare que ao lado dessa informação existe um pequeno botão escrito "**T**" (do inglês Toogle - mudar), que ao ser clicado com o mouse muda o valor de OFF para ON para OFF e assim sucessivamente, assim podemos escolher qual o valor lógico em cada pino.

Apenas a título de informação, se este PIC tivesse entradas Analógicas, apareceria nelas em vez de OFF ou ON o valor da tensão aplicada no pino, e em vez de estar escrito "T" no botão estaria escrito "A" , que ao ser clicado permite aparecer um SLIDER para selecionarmos o valor entre 0 e 5 volts .

Repare que os pinos do PIC possuem também o nome que o fabricante deu a esse pino, por exemplo, no pino 1 está escrito **RA2**, que significa **PORTA A , BIT2** .

Alguns pinos podem possuir dupla função nessa tela, por exemplo, o pino 6 está escrito **INT/RB0**, pois de acordo com o que queremos programar no PIC, este pino pode servir como uma simples porta de I/O normal, ou pode servir como uma entrada de INTERRUPÇÃO.

Neste tutorial não vou usar interrupção, pois acredito que está um pouco fora do objetivo que é o de permitir o uso imediato dos PIC's para quem nunca usou microprocessador.

Agora, vamos lembrar o que definimos no nosso projeto de pisca- pisca :

Usamos a porta B pinos RB6 (12) e RB7 (13) como saída (OUTPUT).

Usamos a porta B pino RB1 (7) como entrada (INPUT).

Portanto, temos de ligar 2 LEDs de nossa janela de LEDS aos pinos 12 e 13 do PIC.

Veja como é fácil :

1. Clique no primeiro LED no quadrado entre o desenho redondo do LED e o retângulo com a cor verde. Aparece uma pequena janela que nos permite escolher qual a porta que iremos usar, e qual o Bit desta porta. No nosso caso, vamos ligar ao pino 12 do PIC, que é chamado de RB6, significando PORT B, bit 6. Para definir isto, selecione PORTB e clique em seguida no retângulo com o número 6, e a seguir clique em SELECT, Pronto, vai aparecer na tela dos LEDs a informação PORTB,6 .
2. Agora, repare no retângulo na cor verde que existe ao lado direito da informação do pino do LED PORTB,6 . Ele significa que quando o Led for acender, ele vai ficar com a cor Verde. Vamos mudar para a cor Vermelha : Clique no retângulo verde, aparecerá uma janela pedindo para que escolhamos a cor, clique em RED, e a seguir clique em Select, pronto, agora aparecerá a cor Vermelha dentro do retângulo.
3. Defina agora o segundo LED com a cor AZUL (BLUE) e ligue ele ao pino 13 do PIC (RB7) . Deve aparecer a informação PORTB,7 e o retângulo ao lado deve estar com a cor Azul.

Pronto, agora poderemos ver a simulação de nosso programa, vendo o piscar dos dois Leds com as suas respectivas cores.

ESCREVENDO NOSSO PRIMEIRO PROGRAMA

Neste momento você deve ter 3 janelas abertas, sendo uma a principal , que é a do PSI, e as outras são o painel de 8 LEDs e a visão dos pinos do PIC (MICROCONTROLLER VIEW).

Vamos abrir a janela do Compilador Basic, para que possamos escrever nosso programa.

Clique em Tools, e selecione BASIC COMPILER.

Se aparecer uma janela com a linha 0001 sem nada mais , temos de colocar novamente a CONFIGURATION WORD. Para isso, clique em Options, Configuration Bits, veja se todas as opções estão corretas, clique em APPLY, e clique em Generate Basic Code, a seguir clique em Close.

Veja na janela do Compilador que agora aparece a linha 0001 com a instrução Define CONF_WORD = 0X0001. Agora só falta escrevermos o nosso programa.

LISTAGEM DO PROGRAMA :

IMPORTANTE : *RECOMENDO SEMPRE QUE VOCE DIGITE O PROGRAMA LINHA A LINHA, POIS É A MELHOR MANEIRA DE FIXAR A SINTAXE USADA NOS PROGRAMAS EM BASIC !*

O programa completo é este aqui, copie ele na janela do Basic Compiler :

```

Define CONF_WORD = 0x0001
Define CLOCK_FREQUENCY = 4
TRISA = 0x00
TRISB = 00000001b
Dim led1 As Bit
Dim led2 As Bit
inicio:
led1 = 0
led2 = 1
Gosub saida
WaitMs 10
led1 = 1
led2 = 0
Gosub saida
WaitMs 10
Goto inicio
End
saida:
PORTB.6 = led1
PORTB.7 = led2
Return

```

A instrução da linha 0002 define a frequência do cristal usado.

Inicialmente, temos de configurar as portas PORTA e PORTB para que funcionem como queremos, de acordo com o nosso esquema eletrônico apresentado no início.

Não usaremos aqui o PORTA, portanto vamos inicializar todos os pinos como SAIDA, só para não deixarmos nenhuma porta de entrada flutuante.

Usamos o PORTB, portanto vamos inicializar o bit 1 como entrada, e todos os outros como saída.

No PSI, temos uma instrução específica para inicializar um PORT , usando a instrução TRISA ou TRISB conforme a porta que queremos. Como podem existir PICS de 5 portas, as letras podem ir de A até E, por exemplo, TRISE.

No PSI, podemos usar notação hexadecimal ou binária, e para os PICs suportados, no caso do comando TRISA se colocarmos um bit em 0 define a porta como SAIDA, e o bit em 1 define como ENTRADA.

Na linha 0003 o comando TRISA 0x00 usei a notação hexadecimal (repare o x !) , e isto faz todos os bits do PORT A serem SAIDA.

Já na linha 0004, usei notação binária (reparem a letra b !) , assim fica mais fácil visualizar que eu especifiquei que os bits 0 até 6 (RB0 até RB6) sejam saída, e apenas o bit 7 (RB7) seja entrada. Simples não é ?

Com isto já definimos tudo o que precisamos sobre o hardware de nosso programa, agora o restante são comandos do Basic do PSI, e caso você tenha dúvidas sobre a linguagem Basic, recomendo verificar o manual da linguagem do PSI, que está diretamente nesta página :

<http://www.oshonsoft.com/picbasiccompiler.html>

Claro que está em Inglês ... mas existe um excelente tutorial em espanhol, e você poderá aprender bastante com ele, está nesta página :

http://www.ucontrol.com.ar/wiki/index.php/PIC_BASIC_%28PSI%29

Vou apenas comentar algumas instruções que podem ser um pouco mais complexas para quem está iniciando.

As linhas 0005 e 0006 servem para criar as variáveis chamadas **led1** e **led2**, que eu usarei para acender (led1 = 1) ou para apagar (led1 = 0) os LEDs no simulador.

A instrução WAITMS 10 que é usada nas linhas 0011 e 0015 são comandos que fazem o programa esperar 10 milissegundos . Uso este valor para que na SIMULAÇÃO você possa ver os Leds acendendo e apagando. Mas este valor serve apenas para você fazer a simulação. Você pode usar qualquer valor entre 1 e 65535

No programa final a ser gravado no PIC, você terá de mudar estas instruções, pois o tempo correto é de 250 milissegundos (lembre, nossos Leds tem de piscar 2 vezes por segundo !!!!), ou seja, os Leds são alternados 4 vezes por segundo, o que faz eles piscarem (acenderem) 2 vezes por segundo.

LEMBRE-SE DE SEMPRE MUDAR OS TEMPOS DAS INSTRUÇÕES WAITMS QUANDO FOR RODAR A SIMULAÇÃO, USE SEMPRE VALORES BAIXOS. SÓ QUANDO TUDO ESTIVER PRONTO PARA GRAVAR VOCÊ VOLTA A COLOCAR OS VALORES ORIGINAIS OK ?

Outro detalhe muito importante : no nosso projeto original , se você ver o esquema que coloquei acima, verá que o LED só acende quando a saída do PIC for 0, mas na simulação do PSI o LED só acende quando a saída é 1 !!!!!!!!!!!!!!!

Ou seja, a lógica das portas de saída tem de ser trocadas também antes de você gravar o PIC em definitivo. No nosso programa deste exemplo, bastaria inverter o estado das variáveis **led1** e **led2**.

Outra instrução interessante é a que muda o estado dos pinos de saída, fazendo os Leds acenderem ou apagarem. Veja a linha 0019 e a linha 0020.

```
PORTB.6 = led1  
PORTB.7 = led2
```

A primeira instrução muda o estado **apenas** do bit 6 da porta B , ou seja, o pino RB6 recebe o valor da variável **led1**.

Similarmente, a próxima instrução faz o pino RB7 receber o valor de [led2](#).
Agora , uma dica importantíssima sobre o uso do Basic do PSI :

Sempre escreva as suas sub-rotinas **APÓS** o final do programa, sinalizado pelo comando END !!!!

Repare que eu uso uma sub-rotina para mudar as saídas dos Leds, e ela está escrita imediatamente após o final do programa. Veja as linhas 0017 e 0018 !

Bom, o restante dos comandos são bem básicos, e não cabe a mim ensinar a linguagem, deixo para você ler e praticar através dos dois Links que eu citei acima.

Agora, apenas escreva o programa reproduzindo fielmente a listagem apresentada acima.

FINALMENTE - A SIMULAÇÃO !!

Neste instante, você deve ter as seguintes janelas abertas : PIC SIMULATOR IDE (o programa principal) , MICROCONTROLLER VIEW , 8X LEDS , e a última que é a do BASIC COMPILER.

Na janela do BASIC COMPILER, clique em **TOOLS**, aparecerão 3 opções.

1. **COMPILE** - Esta opção apenas compila o nosso programa, e ao final mostra um resumo dos recursos que estamos usando do nosso PIC. Se seu programa não tiver nenhum erro , será apresentada uma janela com as estatísticas. O que mais importa no momento é a FLASH usage : mostra o quanto que estamos usando da memória de programa de nosso PIC.
2. **COMPILE & ASSEMBLE** - Além da compilação, faz a transformação para o código de máquina.
3. **COMPILE & ASSEMBLE & LOAD** - Esta é a opção preferida, pois ela já faz todo o trabalho e já carrega o programa na memória do PSI, para que possamos fazer a simulação. Ao final é apresentada uma janela por poucos segundos, com um resumo que mostra quantas posições da memória Flash de programa estaremos usando. Também gera ao mesmo tempo os arquivos .ASM e .HEX contendo o código do programa na linguagem Assembler, e o código objeto prontinho para ser gravado no PIC.

Clique na terceira, **COMPILE & ASSEMBLE & LOAD**, e verá ao final , na janela que dura pouco tempo, que nosso programa ocupa apenas 77 bytes de um total de 1024 bytes que este PIC possui !!!!

Nada mau para um programa escrito em com linguagem de alto nível.

Agora, vamos configurar o PSI para rodar nossa simulação :

Na janela principal do PSI, clique em **RATE**, e selecione a última opção, **ULTIMATE**.
Lembre-se de sempre usar esta opção, as outras realmente não tem muito uso.....

Agora, o grande momento, vamos ver o nosso programa funcionando !

Na janela principal do PSI, clique em **SIMULATION**, e escolha a primeira opção , **START**, e fique vendo as coisas acontecerem nas janelas !!!!

Inicialmente, veja na janela 8X LEDS , verá que os dois primeiros Leds estarão piscando alternadamente !!!!! E cada um numa cor diferente !

Agora, repare na janela MICROCONTROLLER VIEW, verá que os pinos com os nomes de RB7 e RB8 ficam alternando ON e OFF. Na verdade , quando um deles está ON, o LED correspondente a ele se acende !

E por último, repare na janela com o programa em BASIC, repare que a instrução que será executada fica na cor VERMELHA, e assim poderemos sempre saber como que o nosso programa está rodando. Claro, é muito rápido, mas podemos mudar a maneira de fazer a simulação, podemos fazer rodar passo a passo, assim poderemos executar as instruções manualmente e vermos os resultados no PSI.

Repare na janela principal do programa : temos muitas coisas para observar, e uma das mais importantes ficam na parte intermediária do lado direito : [as janelas Clock Cycles Counter e Real Time Duration](#).

Elas nos mostram respectivamente quantos ciclos de clock se passaram até o momento, e qual o tempo em microssegundos que demoraria para chegar até a instrução a ser executada.

Muita gente fala que o BASIC é lento.... vamos mudar a simulação e começar de novo rodando passo a passo, para acompanharmos essas informações sobre os tempos envolvidos.

Na janela principal, clique em **OPTIONS**, e escolha **RESET SIMULATION STATISTICS**, isto faz os contadores voltarem a zero.

Agora, clique em **RATE**, e selecione **STEP BY STEP**.

E por último, clique em **SIMULATION**, selecione **START**.

Aparecerá nessa mesma janela uma nova opção, embaixo do menu principal de comandos , ao lado esquerdo, escrito **Run To Next BASIC Statement**.

Clique nessa opção e veja o que acontece !

Repare que na janela do BASIC, a instrução na linha 0003 ficou na cor vermelha, ou seja, o programa já rodou até a instrução anterior a ela, e ela será a próxima a ser executada !!!

Veja agora na janela principal do PSI os contadores : 20 ciclos de clock, totalizando 5 microssegundos.

Clique novamente na opção **Run To Next BASIC Statement**, e repare que agora a o programa parou na linha 0004, e executou 28 ciclos de clock , num tempo total de 7 microssegundos.

Vá clicando até a linha **0011 WaitMs 10** ficar com a cor vermelha, e pare por um momento. Repare que já temos um LED aceso, e o tempo necessário foi de apenas 24 microssegundos, que demoraram 96 ciclos de clock para serem executados. Quem é que falou mesmo que o BASIC é lento ??????????????????????????????

Falei para parar aqui porque se você clicar para prosseguir a simulação, vai perceber o porque que usamos tempos "falsos" na instrução WaitMs , pois no nosso caso, ela manda esperar 10 milissegundos, o que significa 10.000 microssegundos !!!

Isso demora bastante para ser executado pelo programa do simulador, e depende também da velocidade de seu PC. Agora que você já sabe, clique para continuar essa instrução, e repare no tempo que vai demorar para ser concluída essa instrução. Imagine se fossem os tempos REAIS, que usaremos apenas quando formos compilar para gravar o programa no PIC.

Agora, veja novamente os contadores : tempo total 10037 microssegundos.

Antes dessa instrução ser executada, o tempo era de 24 microssegundos, agora é de 10037 microssegundos. O que significa na verdade que essa instrução não é tão precisa assim, pois ela demorou exatos 10013 microssegundos. Resumindo, em milissegundos, o nosso programa ficou parado por 10,013 milissegundos. Embora o erro seja bem pequeno, ele existe, e se seu programa exige uma temporização perfeita, você deve considerar isto, e sempre se oriente pelo contador Real Time Duration. Apenas como informação, uma boa parte do erro é causada pela duração da instrução que chama a sub-rotina, e pela instrução que devolve o controle para o programa principal.

Dito isto, agora clique em **RATE**, e volte para **ULTIMATE**, e veja a sua simulação voltar a rodar.

Para terminar a simulação, clique em **SIMULATION** e escolha **STOP**.

Pronto, fácil não ??????????????

Com isto você já está apto a fazer pequenos programas e logo estará dominando esta poderosa ferramenta.

Vamos agora passar ao nosso próximo programa, que usará o botão que existe em nosso projeto.

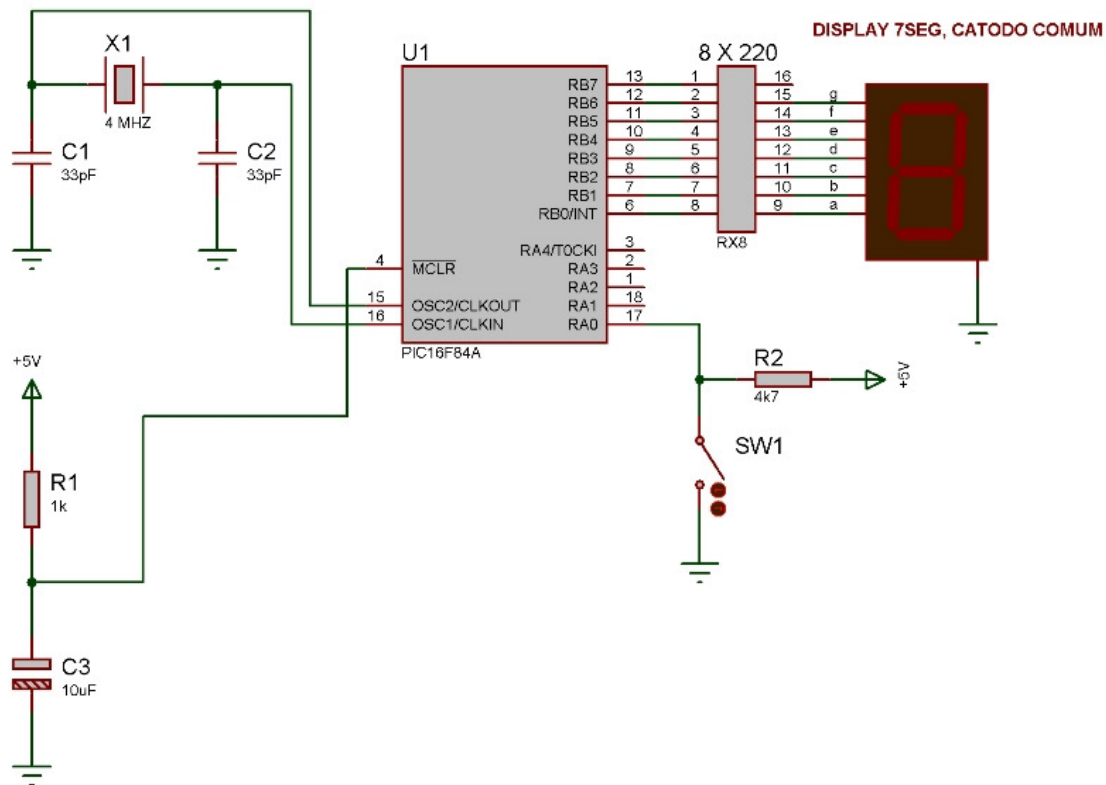
PROJETO 2 - USANDO DISPLAY 7 SEG.

Usar um display de 7 segmentos é muito fácil, basta criarmos uma tabela que indica quais os segmento que devem ficar acesos para cada dígito.

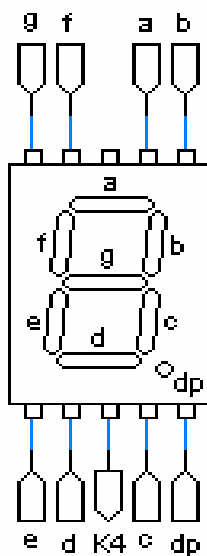
Vamos aqui implementar um contador , que aumenta um numero a cada meio segundo, e assim indefinidamente.

Como dica , lembre-se de que cada saída do PIC permite fornecer uma corrente de 20 mA com segurança, mas existe uma limitação para a corrente total fornecida por um PIC. Como segurança, vamos usar aqui 200 mA, mas você pode verificar esse valor exato no Datasheet do PIC.

CIRCUITO DA APLICAÇÃO



IMPORTANTE - LIGAR O PINO 14 AO +5V E O PINO 5 AO TERRA (GND)
O display de 7 segmentos está ligado conforme a identificação dos segmentos abaixo :



Listagem do programa :

```
0001 Define CONF_WORD = 0x0001
0002 Define CLOCK_FREQUENCY = 4
0003 TRISA = 0xff 'todos os pinos como entrada
0004 TRISB = 0x00 'todos os pinos como saída
0005 Dim digito As Byte 'os números que vamos mostrar, de 0 a 9
0006 Dim saída As Byte 'padrão de segmentos dos números 0 a 9
0007 Dim chave As Bit
0008 PORTB = 0xff 'acende todos os 7 seg.
0009 WaitMs 30 'espera 2 segundos ( 2000 ms ) e apaga
0010 PORTB = 0x00
0011 espera:
0012 chave = PORTA.0
0013 If chave = 1 Then Goto espera
0014 sempre: 'chave sw1 foi pressionada
0015 For digito = 0 To 9
0016 saída = Lookup(0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f),
digito
0017 PORTB = saída
0018 WaitMs 10 'espera meio segundo ( 500 ms ) e vai para o próximo
0019 Next digito
0020 Goto sempre
0021 End
```

Antes de simular, aqui temos um pequeno truque. Se você reparar o esquema elétrico, vai reparar que no nosso circuito real, o pino RA0 da PORTA está ligado ao +5V através do resistor de Pull-Up de 4k7, e portanto o nível neste pino, assim que ligarmos o circuito, é nível lógico 1 ; apenas quando apertamos a chave é que o nível lógico desse pino irá momentaneamente para 0.

Já no Pic Simulator IDE, o estado padrão que é iniciada os pinos do PIC é nível 0, o que pode fazer que nossa simulação simplesmente rode direto sem esperar que a chave SW1 seja acionada.

Para resolver este problema, existe uma maneira de configurarmos os níveis nas portas ANTES de rodarmos a simulação. Para que isto funcione, temos de mudar mais um setting no programa.

Na janela principal do Pic Simulator IDE, clique em **OPTIONS**, e deixe clicada a opção **Preserve Input States On Simulation Start**.

Pronto, agora basta você ir na janela Microprocessor View e mudar o estado do pino RA0 para ON, e então pode rodar o simulador.

Antes, vamos ver como configurar o display de 7 segmentos no PSI.

Na janela principal do PSI, clique em Tools, e em seguida 7-Segment Leds Display Panel . Vai aparecer uma nova janela, que mostra 4 displays tipo LED de 7 segmentos.

Agora precisamos fazer a configuração do display. Para isso, escolha o primeiro display e clique em SETUP.

Temos de configurar qual pino do PIC que é ligado a cada segmento. O padrão que eu uso é sempre este : segmento a - PORTX0 , segmento b - PORTX1 e assim por diante. Claro que não existe o PORTX, aqui você tem de escolher qual dos ports do PIC você vai usar. No nosso projeto, usamos o PORTB, portanto você tem de ir em cada um dos segmentos e selecionar o PORT e o bit do port que você quer usar. Exemplo : segmento a - usar PORTB pino 0.

Repita o procedimento para todos os 7 segmentos. Agora existe um botão nessa janela chamado LED COLOR, clique e escolha a cor que você quer que seu display acenda. Por ultimo, embaixo da tabela dos segmentos existe um botão para você selecionar se este display estará ativo ou não na nossa simulação, escolhendo respectivamente DISPLAY ALWAYS ENABLED ou DISPLAY DISABLE ou se quiser especificar um determinado pino do PIC para ligar ou não esse display. No nosso caso o display estará sempre ligado, então escolha ALWAYS ENABLED.

Por último, antes de simular, repare numa instrução nova : **LookUp**

Esta instrução faz uma consulta em uma tabela de vários valores, e retorna o valor que está indicado na posição passada na função. Esta tabela pode ter até 256 elementos, sendo que o primeiro elemento na tabela significa posição 0 e o ultimo elemento significa posição 255. Vamos ver o exemplo de nosso programa :

```
saída = LookUp(0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f), digito
```

Repare que nossa tabela possui 10 elementos, então o valor que pode ser passado para a consulta pode ir de 0 até 9. O valor a ser pesquisado é passado na variável **digito**, e o valor correspondente na tabela é atribuído à variável **saída**.

Para que estamos usando isto ? Para sabermos quais os segmentos que devem ser acesos para cada número que vamos mostrar.

Por exemplo, para o número 0 o valor retornado por Lookup é 0X3F, que em binário é 00111111 ou seja, acende os segmentos a,b,c,d,e,f .

Já para o número 7 retorna 0x07, em binário 00000111, segmentos a,b,c.

Assim fica fácil né ?

Agora, vamos rodar o simulador.

Primeiro, na janela do BASIC COMPILER, digite o programa mostrado acima, salve como PROJETO2.BAS, e utilize a opção TOOLS / Compile & Assemble & Load. Se der tudo certo, pode rodar a simulação, na velocidade ULTIMATE.

Deixe aberta as janelas mostrando o DISPLAY DE7 SEGMENTOS e a janela MICROCONTROLLER VIEW, certificando-se de que o pino RA0 esteja em ON.

Na janela principal do PSI, clique em RATE e escolha Ultimate.

A seguir, clique em SIMULATION e escolha START.

Veja que o LED vai acender todos os segmentos, depois vai apagar e ficar esperando você mudar o estado do pino RA0. Assim que você mudar, a contagem vai começar de 0 até 9 e assim indefinidamente.

Lembre-se de que os valores que estão nas funções WAITMS são apenas válidos para a simulação.

Se você for gravar o PIC e montar o circuito, lembre-se de trocar pelos valores corretos que estão indicados nos comentários !!!!

PROJETO 3 - USANDO 2 DISPLAYS 7 SEG.

Agora faremos algo bem mais sofisticado, e para isso precisaremos usar recursos bem avançados de programação.

Queremos mostrar valores de 2 dígitos, isto é, indo de 00 até 99, e para isso precisamos de 2 displays. Se formos manter o mesmo esquema, precisaremos trocar de PIC, pois apenas para os displays precisaremos de 14 pinos, ou quase 2 PORTs inteiros de saída. Nosso PIC utilizado até agora é o PIC16F84, que possui um total de 13 pinos de I/O, o que não permite que seja usado pois faltam pinos. Ou trocamos o PIC por outro maior, ou temos de "inventar" algo....

Para isso que existe uma técnica chamada de **Multiplexação**.

Ela permite que possamos ligar vários displays em paralelo, e usamos um dos pinos do PIC para controlar qual o display que queremos acender e mostrar. Por exemplo, no nosso mesmo esquema, ficaríamos com 7 pinos para os segmentos de ambos os displays, e mais 2 pinos que selecionam qual dos displays que mostram o valor. Se fizermos isso rapidamente, a pelo menos 30 vezes por segundo, nosso olho vai ter a impressão de que ambos os displays estarão acesos ao mesmo tempo, mas na verdade o que estamos fazendo é isto : ligar um display, mostrar um valor nesse display, desligar o display, em seguida ligamos o outro display, mostramos o valor nesse outro display, apagamos o display, e começamos tudo de novo. Se isto for feito muito rápido, teremos a impressão de que os dois displays estarão sempre acesos.

Para facilitar a temporização, usaremos uma **INTERRUPÇÃO** programada pelo TIMER do PIC.

Aconselho que você estude bem esta técnica, pois ela permitirá você trabalhar em BASIC usando interrupções de qualquer tipo, sejam internas ou geradas por um hardware externo.

COMO CALCULAR E GERAR INTERRUPÇÃO DE TEMPOS EM TEMPOS.

Dentro de um PIC existe alguns TIMERS, que podem ser utilizados para gerar a temporização para nossos programas.

O mais comum é utilizar o TIMER0, existente em praticamente qualquer PIC. O que precisamos saber é que ele é um contador de 8 bits.

Veja bem no Datasheet do PIC16F84 as várias maneiras de funcionamento desse timer. Mas, para ser usado como temporizador de interrupção, a maneira é sempre a que iremos ensinar aqui.

Lembrando do princípio da multiplexação, se fizermos cada display acender pelo menos 30 vezes por segundo vamos ter a ilusão de termos os displays sempre acesos. Vamos definir então que queremos acender no mínimo 80 vezes por segundo. E isto significa gerar uma interrupção a cada 12,5 milissegundos CERTO ?

Então precisamos gerar uma interrupção a cada 12,5 milissegundos. E sabemos que estamos usando um cristal de 4 MHz, o que gera um clock interno de 1 MHz.

Quando queremos gerar uma interrupção a cada 12,5 mseg, queremos que esse contador estoure a contagem de 8 bits (de 0 até 255, ou 256 no total) a cada 12,5 mseg. Ou seja, o TIMER0 tem de contar os ciclos de clock que ele recebe na entrada, e só estourar a contagem a cada 12,5 mseg.

Mas, temos um problema : quantos ciclos de clock ocorrem a cada 12,5 mseg ???
Oras, se temos uma frequência na entrada de 1Mhz, temos 1 milhão de ciclos em um segundo, basta multiplicar por 0,025 para sabermos quantos ciclos ocorrem. A resposta é esta : 12.500 ciclos.

Oras, temos um problema, pois se nosso contador só pode contar 256 ciclos antes de gerar uma interrupção, temos de dividir a frequência de entrada, de modo que para uma entrada de 12.500 ciclos de clock seja feita apenas 1 contagem até 256.

Para fazer esta divisão, é que existe o PRESCALER !

O PIC16F84 possui um PRESCALER selecionável entre os valores de 2,4,8,16,32,64,128 e 256.

Apenas para fazer uma conta rápida, a única maneira de dividir os 12.500 ciclos de clock para caber em 256 é ($12.500 / 256$) = 42,82 ou seja, temos de selecionar OBRIGATORIAMENTE o PRESCALER de 64, POIS É O VALOR IMEDIATAMENTE ACIMA DE 42

Então, agora sabemos que estamos dividindo os clocks na entrada por 64, ou seja, quando tivermos 12.500 ciclos na entrada, o nosso contador vai atingir a contagem de 195

Como o resultado não deu um número inteiro, não vamos conseguir gerar este valor exato de 12,5 mseg.

E como que faremos o estouro do contador em 256 , se vamos ter apenas uma contagem até 195 ?????

Simples, em vez de começarmos a contagem de zero, vamos começar ela de $256 - 195 = 61$!!!!!

Ou seja, precisamos programar o nosso PIC para usar uma contagem inicial de 61 e um PRESCALER de 64 . Pronto !!!!!

Vamos ver exatamente qual o período em que vamos ter as interrupções ????

Vamos ter 195 contagens de 64 ciclos, o que dá $195 \times 64 = 12480$ ciclos , ou seja, vamos gerar uma interrupção a cada 12,48 mseg . Para a nossa finalidade, que não exige precisão, está ótimo !!!!

RESUMO DA TÉCNICA :

- 1 - Saber qual o clock interno do processador (XTAL / 4)
- 2 - Calcular a contagem inicial do TIMER, e programar o fator de PRESCALER.

Exemplo : timer gerando interrupção a cada 12,5 milisegundos, com cristal oscilador de 4 MHz.

- Clock interno = $4 \text{ MHz} / 4 = 1 \text{ MHz}$, ou 1000000 ciclos de clock
- Queremos uma interrupção a cada 12.500 clocks. Como é muito maior de 256, usaremos o prescaler que permite gerar resultado menor do que 256, no caso usaremos PRESCALER = 64 pois $12.500 / 64 = 195,65....$
- Vamos ter uma interrupção a cada 195 contagens de 64 ciclos, e portanto usaremos a contagem inicial de $(256 - 195) = 61$

Resultado : PRESCALER de 64 e contagem inicial (pré-carga) de 61

COMO PROGRAMAR A INTERRUPÇÃO DO TIMER0 EM BASIC

Usaremos as palavras reservadas do BASIC :

```
OPTION_REG.T0CS = 0 'usar clock interno
OPTION_REG.PSA = 0 'prescaler ligado no TIMER0
OPTION_REG.PS2 = 1 'valor prescaler = 64
OPTION_REG.PS1 = 0
OPTION_REG.PS0 = 1
TMR0 = 0x3d 'contagem inicial do TIMER0 de 61
```

Com os comando acima, tudo já está programado, só falta ligar as interrupções, para isso usamos a seqüência abaixo :

```
INTCON.T0IE = 1 'Habilita as interrupções do TIMER0
INTCON.GIE = 1 'Habilita todas as interrupções não mascaradas
ENABLE
```

Pronto, neste momento você já está tendo as interrupções , agora só falta você ter a rotina de tratamento delas certo ?

Vou apresentar aqui um esqueleto para ser usado nas interrupções de TIMER0 :

```
On Interrupt ' Rotina de tratamento de interrupção
SAVE SYSTEM ' salva o contexto, use sempre
```

TMR0= 0x3d ' coloca de novo a contagem inicial
INTCON.T0IF = 0 ' Habilita novas interrupções do TIMER0

Pronto, agora você já sabe todas as instruções envolvidas para usar as interrupções geradas pelo TIMER0.

O TIMER1 pode ser usado igualmente da mesma maneira que o TIMER0 , apenas lembrando que ele é um contador de 16 bits, atingindo a contagem de 65536.

T1CON.TMR1CS = 0	'usar o clock interno
T1CON.T1CKPS0 = 0	'fator do prescaler do timer1, bit0
T1CON.T1CKPS1 = 0	'fator do prescaler do timer1, bit1
TMR1H = 0xf8	'contagem inicial do TIMER1, byte alto
TMR1L = 0x30	'contagem inicial do TIMER1, byte baixo
	'apenas para elucidar, a contagem inicial é 0xf830 ou 63536
T1CON.TMR1ON = 1	'habilita a contagem do TIMER1
PIE1.TMR1IE = 1	'habilita a interrupção do TIMER1
INTCON.PEIE = 1	'habilita todas as interrupções não mascaradas
Enable	

O TIMER2 é um timer um pouco diferente, pois ele tem o recarregamento da contagem inicial automática, e é usado também para as instruções de PWM. Evite usar este timer !

Apenas como um exemplo de um projeto bem complicado com os Pícs : já vi um projeto em que era usado o Timer0 para o Watchdog, o Timer2 para a temporização de base de tempo de 1 milissegundo, e ainda usava o TIMER1 como contador para ler a frequência gerada por um circuito de fonte chaveada, sendo que as leituras de tensão efetuadas pelos conversores A/D tinham de se bem rápidas e também geravam interrupção quando eram concluídas.... para complicar ainda mais, era usada a comunicação serial com um PC, e também era usada a comunicação serial do tipo IIC para armazenar os dados numa memória externa do tipo FLASH EEPROM..... e a gota d'água era que para acompanhar as leituras feitas pelos conversores A/D em tempo real, era usada uma memória SRAM de 32K por 8 bits, e o protocolo de escrita / leitura com a geração do sinal de R/W foi feito usando os pinos de I/O do PIC. Quando a leitura terminava, os dados na memória SRAM eram processados, normalizados, e armazenados na memória serial FLASH, que é muito mais lenta que a SRAM , mas não perde os dados quando o circuito era desligado.

Tudo isto foi feito por 2 engenheiros, usando `Assembler`, que demorou cerca de 4 meses para ser escrito e debugado. E depois de tudo, a empresa quis que o projeto fosse refeito para um PIC mais barato..... que demorou mais 1 mês para ser feito.

Qual foi a economia que a empresa teve nesse projeto, trocando os PICS ?
Segundo os engenheiros, toda a economia na produção não chegava a R\$ 6,00

Qual a quantidade de produtos vendidos nesse projeto ?
Até o momento, 140 peças, com um lucro bruto de mais de R\$ 1.200,00 por peça.
Se analisarmos isto, a economia de R\$ 6,00 é RIDÍCULA, e o custo do mês adicionais para ser obtida passou em mais de 10 vezes a economia obtida de $140 \times 6,00 = \text{R\$ } 840,00$!!!!!!!!!!!

Bom, isto serve apenas para mostrar que quando temos de trabalhar para os outros, temos de engolir uns sapos ENORMES , mesmo sabendo que estamos certos !
Mas vocês concordam que o projeto feito acima é um exemplo magistral de uso de PIC, não é ?

Claro que um projeto desse pode ser escrito em Basic do PSI, mas pelo porte ENORME do projeto, eu recomendo a linguagem MikroBasic, que é um compilador bem mais poderoso que o PSI, mas bem mais complicado também, e sem o fantástico Simulador que temos disponível no PSI.

Lembre-se de que o nosso objetivo aqui neste pequeno tutorial é ensinar o uso de PICs, bem como a sua programação usando Basic. Para o aprendizado inicial, nada supera o PSI. Quando você completar este tutorial, estará apto para migrar ao MikroBasic, mas apenas se se envolver em um grande projeto de nível profissional.

Sei que isto tudo parece um pouco complicado, mas depois que você faz uma vez o seu próprio programa em BASIC você vai entender, decorar e se acostumar a fazer tudo com o PSI !!!!

Agora, uma vez mostradas as técnicas que vamos utilizar na multiplexação, vamos montar o circuito abaixo, e verificarmos o seu funcionamento no simulador.

Antes, apenas para lembrar, usando a multiplexação teremos um baixo consumo de corrente, pois a qualquer momento apenas UM DISPLAY estará aceso, não importe quantos "parecem" que estão acesos.

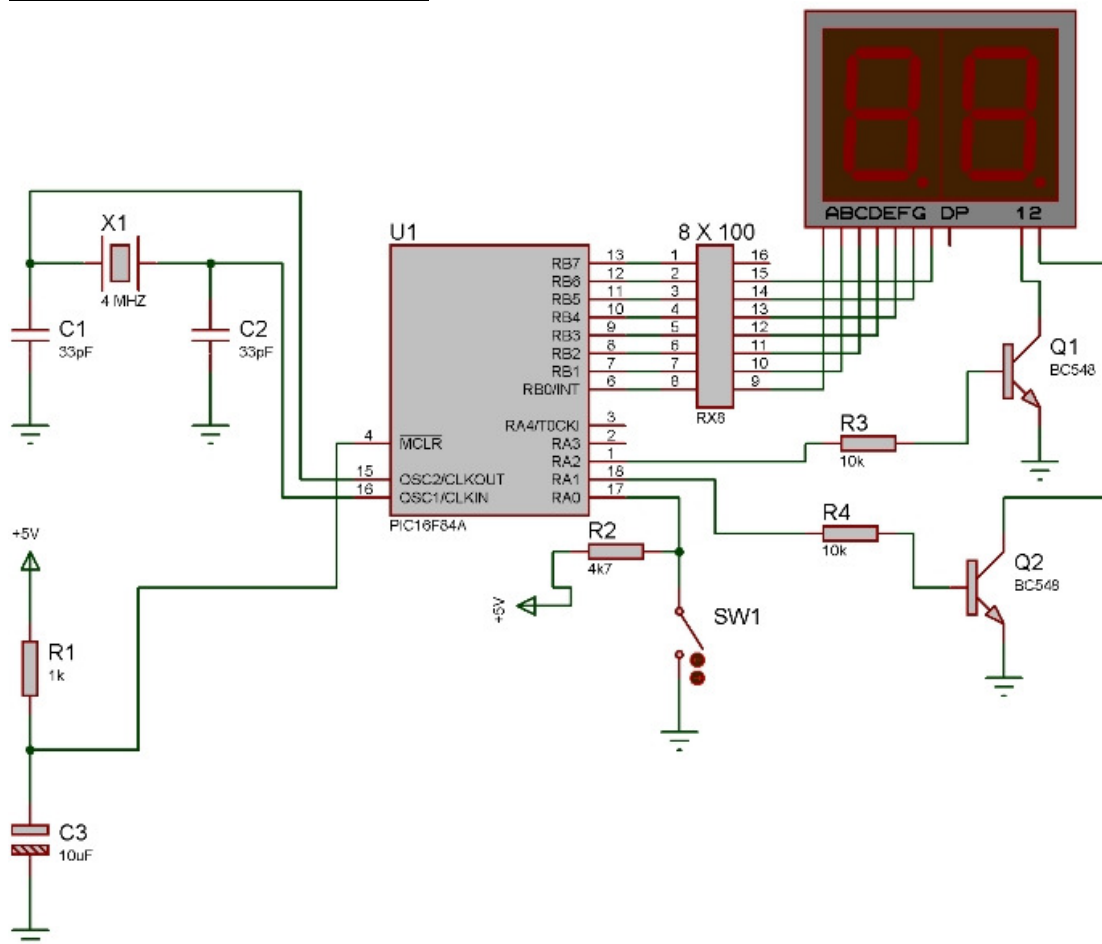
E também usando a multiplexação poderemos usar vários displays em um PIC simples, no nosso caso do PIC16F84 poderíamos usar sem nenhum problema até 4 displays e ainda teríamos pinos de entrada livres.

Um último comentário sobre esta técnica de temporização usando TIMER0 :

Esta mesma técnica permite gerar bases de tempo extremamente precisas, e podemos usar estes princípios para termos um relógio, um escalador de eventos (um sistema que realiza várias tarefas em sequências programadas) , e até mesmo medir eventos como frequência, contador de pulsos, períodos, etc.

Siga meu conselho : domine bem esta técnica de interrupção em Basic com o Timer0 !

ESQUEMA DO PROJETO 3:



IMPORTANTE - LIGAR O PINO 14 AO +5V E O PINO 5 AO TERRA (GND)

No esquema acima, estou usando um display duplo, que tem apenas os pinos mostrados acima. Você pode usar 2 displays comuns do tipo catodo comum, ligando os segmentos a - g em paralelo, e ligando cada um dos catodos ao seu transistor correspondente.

Como agora vamos multiplexar o display, é melhor aumentarmos a corrente de cada segmento, trocando o array RX5 por um de 8 x 100 ohms. Se você não conseguir encontrar esse array, use 8 resistores separados mesmo.

Agora, temos de usar os transistores para o chaveamento dos displays, pois lembre-se de que a corrente máxima em um pino do PIC é de cerca de 20 mA. Com todos os segmentos acesos, a corrente no transistor pode chegar a 160 mA.

Como estamos gerando uma interrupção a cada 12,5 mseg , cada display ficará ligado por 12,5 mseg, e desligado a cada 12,5 , depois liga de novo e assim sucessivamente. Isto significa que cada display acenderá 40 vezes a cada segundo, o que vai permitir a visualização de ambos os displays sem nenhum problema.

Se você acha que o brilho do display está fraco, pode trocar os resistores de 100 ohms por resistores de 47 ohms, e nesse caso estaremos usando a limitação de corrente existente no PIC. Não recomendo isso em projetos reais pois pode encurtar a vida útil do PIC.

Lembre-se de que se usar displays de cor diferente, temos de recalculamos os resistores de 100 ohms, por exemplo se usar display branco a tensão é de 3 volts, portanto o resistor tem de ser menor, pois teremos apenas cerca de 1.8 Volts sobre ele, em vez de 3.3 volts no caso dos displays vermelhos.

LISTAGEM DO PROGRAMA DO PROJETO 3 :

```
Define CONF_WORD = 0x0001
Define CLOCK_FREQUENCY = 4
AllDigital 'prepara o uso do PIC como DIGITAL apenas
TRISA = 11001b 'Prepara o pino RA0 como entrada e RA1 e RA2 como saída
TRISB = 0x00 'todos os pinos como saída pois é a porta do display
```

```
Dim contagem As Byte 'faz a contagem de 0 até 99
Dim digito As Byte 'os números que vamos mostrar, de 0 a 9 para lookup
Dim dezena As Byte 'dezena corrente
Dim unidade As Byte 'unidade corrente
Dim mask As Byte 'mascara corrente que veio de lookup
Dim dezmask As Byte 'mascara da dezena
Dim unimask As Byte 'mascara da unidade
Dim chave As Bit
Dim phase As Byte 'controla sequência mostrada na interrupção
Symbol enabledezena = PORTA.2
Symbol enableunidade = PORTA.1
```

```
enabledezena = False
enableunidade = False
unimask = 0
dezmask = 0
phase = 1
```

```
OPTION_REG.T0CS = 0 'usar clock interno
OPTION_REG.PSA = 0 'prescaler ligado no TIMER0
OPTION_REG.PS2 = 1 'valor prescaler = 64
OPTION_REG.PS1 = 0
OPTION_REG.PS0 = 1
TMR0 = 0x3d 'contagem inicial do TIMER0
INTCON.T0IE = 1 'Habilita as interrupções do TIMER0
INTCON.GIE = 1 'Habilita todas as interrupções não mascaradas
Enable 'agora sim as interrupções já estão acontecendo
```

```
digito = 10 'vamos acender padrão de espera no display
Gosub getmask 'pega a equivalência da tabela
unimask = mask 'prepara mostrar na unidade
dezmask = mask 'e prepara para mostrar na dezena
'automaticamente a rotina de interrupção vai mostrando os dígitos
```

```
espera: 'e agora só esperamos apertar a chave sw1
chave = PORTA.0
```

If chave = 1 Then Goto espera 'espera chave ser apertada

loop: 'finalmente a chave foi apertada

For contagem = 0 To 99

dezena = contagem / 10 'pegamos a dezena

unidade = contagem - (10 * dezena) 'calculamos a unidade

digito = dezena 'agora pegamos o equivalente da dezena para mostrar

Gosub getmask 'consultamos a tabela que retorna sempre em mask

dezmask = mask 'dezena já está prontinha para ser mostrada

digito = unidade 'pegamos agora o equivalente da unidade

Gosub getmask 'fazemos a mesma coisa para a unidade

unimask = mask 'unidade também já está prontinha

WaitMs 50 'agora esperamos 0,2 segundos (200 mseg)

Next contagem 'e continuamos o contador

Goto loop

End

On Interrupt 'Rotina de interrupção

Save System 'SEMPRE SALVAR ESTADO DO SISTEMA

If phase = 1 Then Gosub mostradezena 'ver aonde estamos , dezena

If phase = 2 Then Gosub mostraunidade 'ou unidade

phase = phase + 1 'preparar o próximo estado

If phase = 3 Then phase = 1

TMR0 = 0x3d 'contagem inicial do TIMER0 novamente

INTCON.T0IF = 0 'habilita novamente as interrupções do TIMER0

Resume

getmask: 'Rotina que pega a equivalência para acender os dígitos

mask = LookUp(0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x49),

digito

Return

mostradezena: 'Rotina que mostra a dezena e mantém ela acesa

enabledezena = False 'desliga os displays, já que não sabemos qual estava ligado

enableunidade = False

PORTB = dezmask 'apresenta a saída para o display

enabledezena = True 'e liga o display da dezena

Return

mostraunidade: 'Rotina que mostra a unidade e mantém ela acesa

enabledezena = False

enableunidade = False

PORTB = unimask

enableunidade = True 'agora liga o display da unidade

Return

Lembro aqui mais uma vez que os valores colocados em WAITMS podem ser alterados em função da velocidade de seu computador.

Mas quando você for gravar o PIC para o projeto real, use os valores REAIS para poder visualizar a contagem. Eles estão sempre no comentário !

Este programa foi baseado no exemplo fornecido pelo próprio manual do PIC SIMULATOR IDE, que é originalmente de 4 dígitos, e eu alterei para apenas 2 dígitos. Você pode modificar facilmente para quantos dígitos você precisar, limitado apenas pelo tipo de PIC.

Repare que eu mudei um pouco a função LookUp , pois eu acrescentei um novo equivalente, agora a função retorna 11 valores, sendo que os primeiros de 0 a 9 são os segmentos corretos para mostrar os números 0 a 9, e o último retorna apenas os 3 segmentos horizontais , que eu mostro no início do programa enquanto aguardo pressionar a tecla. Para obter esse desenho, basta chamar LookUp com o valor de entrada 10 .

Como os resultados dos cálculos dos dígitos de dezena e de unidade sempre vão retornar valores de 0 a 9, os valores mostrados sempre serão os corretos.

Outra dica importante : Se você estiver usando outra interrupção além da do TIMER0, teremos de modificar a nossa rotina de interrupção para fazer um teste para descobrirmos QUAL interrupção que foi gerada.... teremos de testar os flags internos dos registros de interrupção e adequar o programa para tratar cada interrupção conforme o hardware que a gerou. Isso é algo mais complicado, e pela minha própria experiência só é usado por profissionais. Lembro mais uma vez que este pequeno curso é para iniciantes em PIC !

Por último, sem querer arrumar briga com o pessoal que programa em Assembler

- 1. Nosso programa completo usou 30,8 % da capacidade do PIC, ou melhor, apenas 316 program words, de um total de 1024 que este PIC tem disponível. Nada mau né ?*
- 2. Repare a facilidade com que podemos implementar interrupções no Basic. Fica muito mais fácil depurar, e podemos ver a simulação quase REAL de nosso projeto.*
- 3. Repare que você pode acompanhar a contagem do TMR0 na tela principal do PSI, e quando ele estourar a contagem você tem também o contador de tempo de programa. Se você anotar esse contador de tempo, esperar um novo estouro do TMR0 e ver o novo contador, irá ver que as interrupções estão sendo geradas mesmo em cerca de 12,5 milissegundos.*
- 4. Olhando a listagem do nosso programa em Assembler (.lst) , vemos que este compilador possui uma ótima eficiência, pois gera um código bem enxuto.*

Agora, vamos à diversão.

SIMULANDO O PROGRAMA

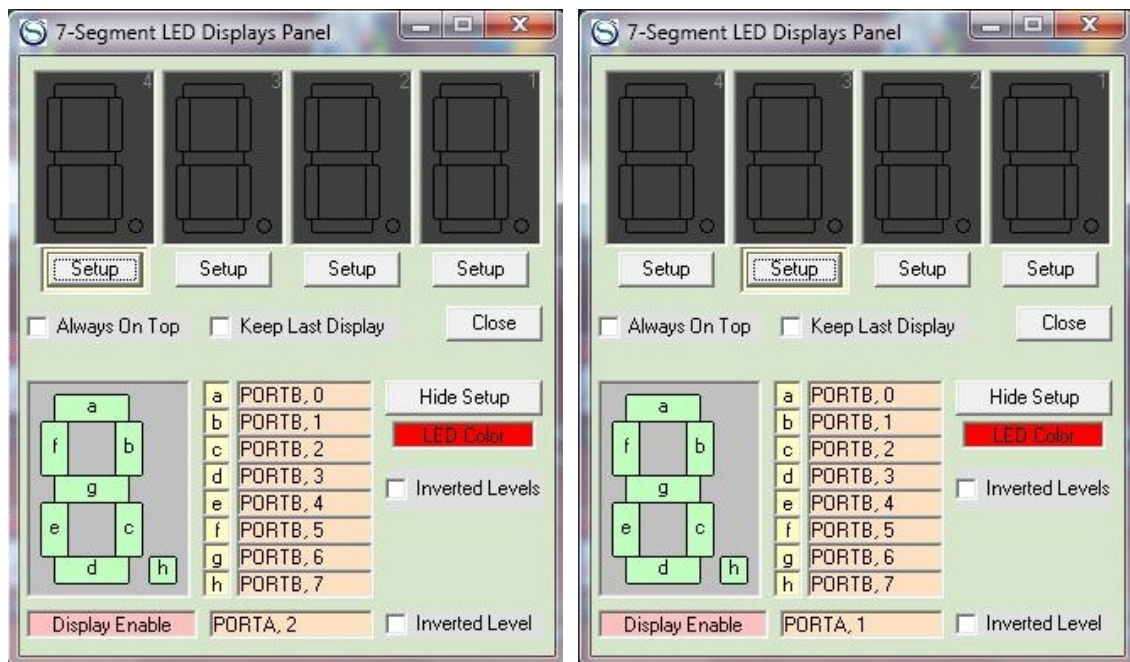
Copie o programa acima na janela do Basic do PIC Simulator IDE.

Vamos mostrar na tela apenas a janela do Microprocessador, e a janela do Display de Led de 7 segmentos.

Prepare novamente os dois displays, sendo que ambos terão os mesmos segmentos ligados no PORTB.

A diferença é apenas no pino que habilita o display.

O display da dezena é habilitado pelo pino PORTA RA2, e o da unidade pelo pino PORTA RA1. Veja a figura abaixo, respectivamente a dezena e a unidade :



Compile o programa digitado, usando a mesma opção Compile & Assemble & Load

Da mesma maneira que o projeto anterior, antes de rodar a simulação teremos de deixar o pino RA0 ligado (ON).

Agora, clique em SIMULATION e START . Para melhorar a velocidade da execução, minimize a janela do BASIC.

Repare como que vão acender os LEDS dos dois dígitos

Fique olhando o TMR0 na janela principal do PSI, ele fica sempre incrementando. Cada vez que ele chegar a FF, vai acontecer uma interrupção, e o display aceso vai apagar e o outro vai acender

Não se preocupe com a demora nessa execução, espere até verificar a troca dos displays.

A seguir, a parte interessante : mude o estado do pino RA0 na janela do Microprocessador, e repare como que o programa vai acender sequencialmente os displays.

Lembre -se de que no projeto real cada display acende e apaga 40 vezes por segundo !

Agora, repare no valor do TMR0, cada vez que o display aceso é trocado, o timer é recarregado com o valor 0X3D, e a seguir é incrementado até atingir FF e estourar, causando uma interrupção que faz a troca do display a permanecer aceso.

Pronto, agora você já pode montar vários projetos que usam displays de 7 segmentos, não é o bicho de 7 cabeças que a turma imagina que seja, graças ao Basic !

PROJETO 4 - MEDINDO TENSÕES E MOSTRANDO EM DISPLAY DE LEDS

Agora, você já está mais confiante em seus conhecimentos, e já pode fazer um projeto mais ousado. Vamos deixar de lado o nosso PIC16F84A, pois ele não tem os conversores A/D, nem memória RAM suficiente para coisas mais complexas.

Para nossos novos projetos, vamos usar um PIC 16F877A, pois ele tem não apenas os conversores A/D, mas também tem mais memória RAM, tem os protocolos de comunicação serial para usarmos memórias seriais tipo Flash, e também tem a comunicação serial assíncrona (UART) para que possamos nos comunicar com um PC !

Depois, vamos evoluir ainda mais o nosso projeto, alterando ele para mostrar os resultados em um display LCD , e posteriormente armazenando as leituras em uma memória do tipo Serial Flash que usa o protocolo serial, com SDA e SDCLOCK.

Por último, iremos fazer a última evolução e implementaremos também a comunicação serial com um PC, para que as leituras gravadas na memória flash possam ser enviadas ao PC e capturadas em um arquivo tipo texto, que pode ser exportado para uma planilha tipo Excel e os gráficos sejam visualizados na tela.

Para isto usaremos o programa HyperTerminal, presente em todos os Windows.

Depois disto, você estará apto a fazer qualquer projeto para PIC em seus hobbies e necessidades caseiras.

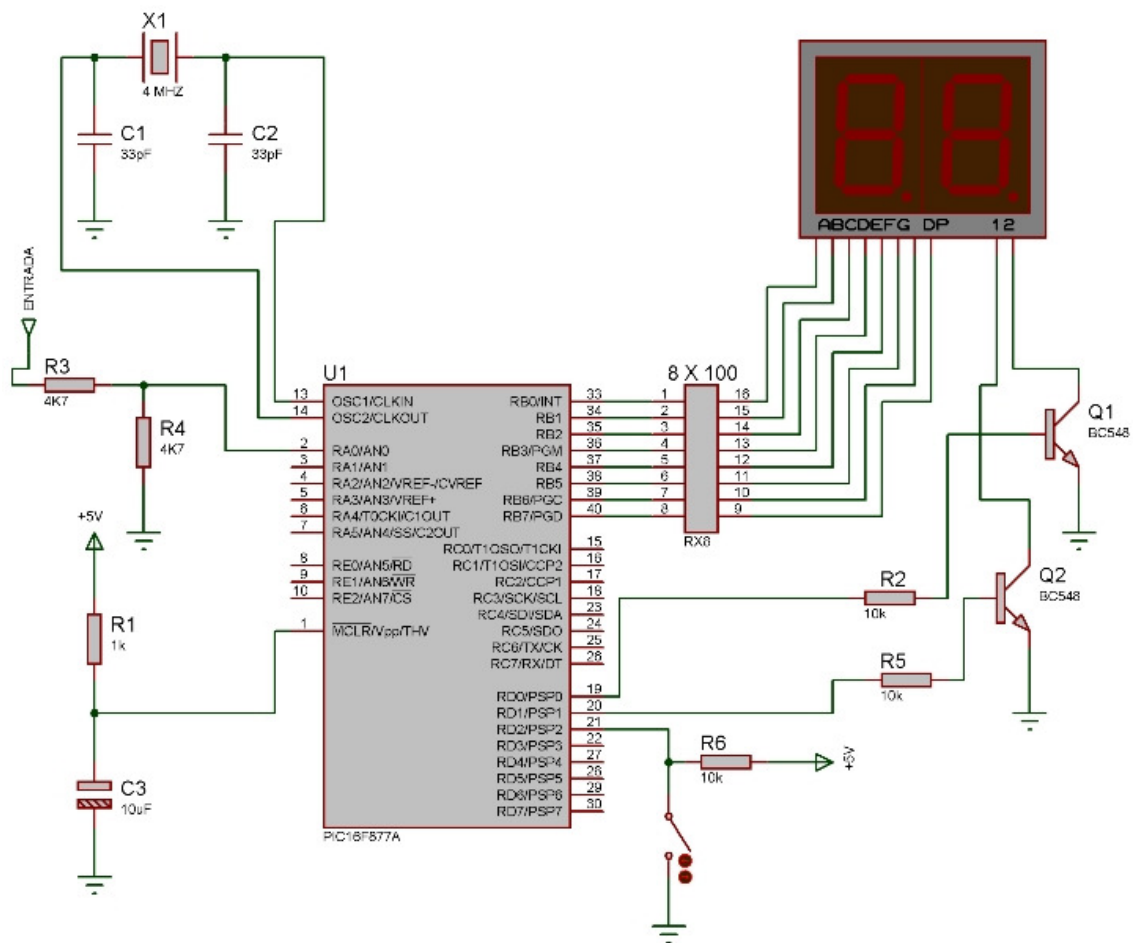
Nada mal para um pequeno tutorial de PIC com Basic !!!!

Bom, vamos ao nosso projeto :

Vamos utilizar um circuito simples, com 2 displays de LED 7 segmentos, e vamos medir a tensão de um ponto que pode variar de 0 a 9,9 volts, e nosso mostrador vai mostrar as tensões de 0 até 9,9 Volts. Assim teremos a oportunidade de aprender como medir tensões acima dos 5 volts que é o limite dos Pics, e também como mostrar valores com casas decimais. Parece fácil, mas lembre-se de que o PIC SIMULATOR IDE usa apenas aritmética com números inteiros, portanto teremos de usar alguns truques para mostrar as medidas com casa decimais.

Isso ocorre muito nos projetos reais do dia a dia, portanto estude bem estes truques.

ESQUEMA DO PROJETO 4 :



IMPORTANTE : LIGAR O PINO 32 AO +5V E O PINO 31 AO TERRA (GND)

Analisando o circuito, percebe-se que foi feita uma alteração, pois agora ligamos o pino do símbolo decimal do display. É que temos de mostrar valores decimais, indo de 0.0 até 9.9 Volts.

Para podermos medir as tensões de 0 até 9.9 volts, sem dar nenhum problema ao PIC, temos de usar um divisor resistivo, e com 2 resistores de 4k7 temos um divisor por 2. Se no pino descrito como ENTRADA colocarmos uma tensão de 8 volts, na entrada do PIC teremos 4 Volts.

Outra coisa que devemos aprender sobre os PICs que possuem entradas analógicas, é que podemos escolher várias opções de uso, por exemplo, podemos ter 8 entradas analógicas, todas elas medindo a tensão de 0 até 5 volts. Mas também podemos mudar a tensão de referência tanto mínima como máxima, por exemplo, se configurarmos o PIC para usar tensões de referência mínima de 1 volt e máxima de 2 volts, iremos ter apenas 6 entradas analógicas, e todas elas vão medir tensões entre 1 e 2 volts, sendo que para 1 Volt teremos a leitura de 0 e para 2 volts teremos a leitura de 1023.

Porque temos as medidas sempre entre 0 e 1023 ? É porque a resolução do conversor A/D é de 10 bits, por isso que os resultados vão até 1023 apenas.

Esta escolha de como que vamos medir as entradas, as tensões de referência mínima e máxima, quantas entradas analógicas e quantas digitais vamos ter no PORTA, estão descritos numa tabela do PIC que descreve todas as possibilidades do registrador ADCON1 . Verifique o datasheet para você entender todas as configurações.

Podemos neste mesmo registrador escolher qual a velocidade de nossas conversões A/D, pois podemos selecionar o clock interno do conversor entre metade da frequência de clock e 1/64 da frequência de clock.

Geralmente, nos usos "caseiros" dos PICs, sempre usamos conversão de tensão entre 0 e 5 volts, e usamos a metade da frequência de clock para as conversões.

Quanto ao problema de escala, e de fazer contas com números inteiros apenas e obter resultados decimais, entra a nossa imaginação .

Por exemplo, se tenho um circuito que mede tensões indo de 0 até 9.9 Volts, temos um problema grave pois o PIC suporta no máximo 5 volts nas entradas A/D. Então temos sempre de usar divisores, ou conversores.

Resumindo, quando temos na entrada do circuito 9.9 volts, o nosso conversor A/D irá receber 5 Volts na sua entrada e irá fornecer uma leitura de 1023.

Quando tivermos 0 Volts na entrada do circuito, teremos também 0Volts na entrada do conversor e teremos uma leitura de 0.

Lembra-se da velha regra de 3 ??? Pois é, 9.9 está para 1023 assim como Xvolts estão para Y na leitura.

Daqui temos que $Leitura = X * 1023 / 9.9$ onde X é a tensão na entrada do circuito.

Desta maneira, sabemos sempre qual a leitura que teremos para qualquer tensão de entrada.

Mas, o importante é já convertermos essa leitura para uma escala mais usável. Concorde que obter uma leitura de 800 não nos diz nada sem termos de fazer um monte de conta ?

Então, nesta hora que entra a nossa imaginação (melhor dizendo, EXPERIÊNCIA) !

Se conseguirmos converter de alguma maneira o resultado da leitura do conversor em um número que vai de 0 até 99, fica muito mais fácil mostrarmos os resultados no display , pois teremos apenas de separar a parte da dezena e a parte da unidade !

Então, temos de fazer uma divisão para converter os valores entre 0 e 1023 para valores entre 0 e 99. Ou seja, é uma simples regra de 3 novamente !

Veja só : $leitura\ final = leitura\ do\ conversor\ A/D * 99 / 1023$!!!!

Desta forma, sempre teremos valores entre 0 e 99 , pois 1023 é o maior valor que pode ser lido pelo nosso conversor A/D . Simples não ?

Mas nem tudo são rosas lembre-se de que o PSI aceita no máximo variáveis do tipo WORD, e os valores nela podem ir de 0 até 65535 .

Suponha que a saída do conversor A/D seja , por exemplo, 1000 , a conta $1000 * 99$ feita no PSI dá resultado errado pois 99.000 é maior do que 65535. Nossa conta iria para o brejo !

Então, aqui usamos outros truques : repare que tanto o número 99 como o número 1023 são divisíveis por 3 !!!!! Se fizermos isto em ambos, não mudaremos o resultado da nossa conta, mas ela vai caber sem nenhum problema em nossa variável do tipo WORD !

Vejamos : $\text{Leitura final} = \text{leitura do conversor A/D} * 33 / 341$!!! AGORA SIM

Esta conta acima sempre vai caber nas variáveis, sem dar erro !!!!

Agora, é só separarmos a parte das dezenas , para ser mostrada no primeiro dígito da esquerda, e a parte das unidades, para ser mostrada no segundo dígito , que é o dígito da direita.

Para isso, usamos mais um truque de matemática.

$\text{dezenas} = \text{leitura final} / 10$

$\text{unidades} = \text{leitura final} - (10 * \text{dezenas})$

Pronto, simples não é ? Agora é só chamarmos de novo a função LookUp e pronto !

Mas, temos outro problema... mesmo que separemos a dezena e as unidades deste resultado, como estamos trabalhando com variáveis WORD , teremos o resultado em variáveis também WORD . E para nosso azar

A função LookUp suporta apenas variável do tipo Byte !

Portanto, temos de descobrir uma maneira de transformar dados de uma variável tipo WORD para uma variável tipo BYTE .

Claro que para isso é necessário que o valor armazenado na variável tipo WORD seja menor que 256 senão teremos resultados totalmente errados.

A maneira de fazer isso no PSI é esta mostrada abaixo.

Suponha que temos o seguinte trecho de programa :

```
Dim leitura as Word
Dim leituranova as Byte
leituranova = leitura.LB
```

Esta função .LB colocada ao final de uma variável do tipo WORD pega a parte baixa (primeiro byte ou byte menos significativo) da variável escolhida e copia ela na nova variável !!

Simples né ? Mas confesso que demorei um tempão para fazer isso e estava na minha cara o tempo todo !

Por último, temos o truque de ligar apenas o ponto decimal.

Como você já sabe, nossa rotina de interrupção fica mostrando num momento um dos displays e em outro momento outro display.

O display que queremos deixar o ponto decimal ligado é o display das dezenas, assim sempre teremos o resultado com uma casa decimal.

Para isso, na rotina que liga o display das dezenas, após enviarmos os segmentos que queremos acender, damos um comando para ligar apenas o bit correspondente ao ponto decimal, que no nosso caso é o bit 7 da porta B, ou PORTAB.7 . Usamos o seguinte comando para isso :

High PORTB.7

Desta maneira não alteramos o valor dos outros 7 bits já presentes na porta de saída.

Pronto, acho que está bem explicado o funcionamento do programa, e com estes novos truques você já sabe como apresentar os valores lidos em seus projetos, inclusive como fazer as contas matemáticas sem estourar as variáveis.

Se você está achando isto complicado, imagine fazer as contas e conversões em Assembler !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Não se preocupe se isto tudo é muita informação para você no momento, apenas estude o programa, e faça a simulação passo a passo, olhando os resultados das variáveis, e você vai entender sem dificuldade. Claro que a melhor maneira de aprender é você escrever seus próprios programas !

Lembre-se de que muitos comandos de I/O serial no PIC SIMULATOR IDE trabalham apenas com variáveis tipo Byte, portanto você terá de utilizar esse truque de conversão de variáveis de WORD para BYTE e até de BYTE para WORD.

Estude os exemplos disto descritos no Manual do PSI. Assim você poderá utilizar diversos periféricos que podemos ligar aos PICs, como por exemplo leitor/gravador de cartão de memória tipo SD, leitor/gravador de PEN DRIVES, interface para cartão SIMM de telefonia, e interface para leitor de GPS !!!

Estes periféricos são disponíveis no Brasil, e não tem custo proibitivo !

Uma empresa que eu sempre acho novidades é a Tato do Brasil : www.tato.ind.br

Olhe de tempos em tempos para saber das novidades !

Agora, vamos ao nosso programa :

LISTAGEM DO PROGRAMA DO PROJETO 4 :

Define CONF_WORD = 0x3f71

Define CLOCK_FREQUENCY = 4

AllDigital 'prepara o uso do PIC como DIGITAL apenas

TRISD = 11111100b 'Pinos RD0 e RD1 como saída, e RD2 como entrada

TRISB = 00000000b 'todos os pinos como saída pois é a porta do display

ADCON1 = 0x00 'usar pino RA0 como entrada analógica

Dim contagem As Word

Dim temp As Byte

Dim digito As Byte 'os números que vamos mostrar, de 0 a 9 para lookup

Dim dezena As Byte 'dezena corrente

Dim unidade As Byte 'unidade corrente

Dim mask As Byte 'mascara corrente que veio de lookup

Dim dezmask As Byte 'mascara da dezena

Dim unimask As Byte 'mascara da unidade

Dim chave As Bit

Dim phase As Byte 'controla sequência mostrada na interrupção

Symbol enabledezena = PORTD.1

Symbol enableunidade = PORTD.0

enabledezena = False

enableunidade = False

unimask = 0

dezmask = 0

phase = 1

OPTION_REG.T0CS = 0 'usar clock interno

OPTION_REG.PSA = 0 'prescaler ligado no TIMER0

OPTION_REG.PS2 = 1 'valor prescaler = 64

OPTION_REG.PS1 = 0

OPTION_REG.PS0 = 1

TMR0 = 0x3d 'contagem inicial do TIMER0

INTCON.T0IE = 1 'Habilita as interrupções do TIMER0

INTCON.GIE = 1 'Habilita todas as interrupções não mascaradas

Enable 'agora sim as interrupções já estão acontecendo

digito = 10 'vamos acender padrão de espera no display

Gosub getmask 'pega a equivalência da tabela

unimask = mask 'prepara mostrar na unidade

dezmask = mask 'e prepara para mostrar na dezena

'automaticamente a rotina de interrupção vai mostrando os dígitos

espera: 'e agora só esperamos apertar a chave sw1

chave = PORTD.2

If chave = 1 Then Goto espera 'espera chave apertada

loop: 'finalmente a chave foi apertada

Adcin 0, contagem 'lê a entrada analógica e guarda leitura em contagem

'agora, temos de acertar a conversão, pois leitura vai de 0 a 1023
'e no nosso circuito 1023 tem de corresponder a 9,9 volts na entrada
'do circuito, ou 5 volts na entrada RA0.
'para facilitar, faremos contagem variar de 0 até 99 para a tensão
'de 0 até 9,9 volts

contagem = (contagem * 33) / 341

'na verdade a conta exata era (contagem * 99) / 1023
'mas o resultado de contagem * 99 pode ultrapassar 65535, que é
'o limite da variável tipo WORD . Portanto temos de tentar sempre
'usar a matemática para fazer as contas serem sempre menores.
'como tanto 99 como 1023 são divisíveis por 3, simplificamos !
'agora o resultado de contagem * 33 sempre será menor do que 65535 !
'desta maneira sempre teremos contagem entre 0 e 99, e nossa
'conversão de escala está pronta !
'agora, para não termos problemas com o programa, vamos
'converter a variável que é WORD para uma que é byte

temp = contagem.LB 'como o valor é menor do que 255, basta pegar
 'o byte LOW da variável.

dezena = temp / 10 'pegamos a dezena
unidade = temp - (10 * dezena) 'calculamos a unidade
digito = dezena 'agora pegamos o equivalente da dezena para mostrar
Gosub getmask 'consultamos a tabela que retorna sempre em mask
dezmask = mask 'dezena já está prontinha para ser mostrada
digito = unidade 'pegamos agora o equivalente da unidade
Gosub getmask 'fazemos a mesma coisa para a unidade
unimask = mask 'unidade também já está prontinha
WaitMs 50 'agora esperamos 0,2 segundos (200 mseg)

Goto loop

End

On Interrupt 'Rotina de interrupção

Save System 'SEMPRE SALVAR ESTADO DO SISTEMA

If phase = 1 Then Gosub mostradezena 'ver aonde estamos , dezena ou unidade

If phase = 2 Then Gosub mostraunidade

phase = phase + 1 'preparar o próximo estado

If phase = 3 Then phase = 1

TMR0 = 0x3d 'contagem inicial do TIMER0 novamente

INTCON.T0IF = 0 'habilita novamente as interrupções do TIMER0

Resume

getmask: 'pega a equivalência para acender os dígitos

```
mask = LookUp(0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x49),
digito
Return
```

```
mostradezena: 'mostra a dezena e mantém ela acesa
enabledezena = False
enableunidade = False
PORTB = dezmask
High PORTB.7
enabledezena = True
Return
```

```
mostraunidade: 'mostra a unidade e mantém ela acesa
enabledezena = False
enableunidade = False
PORTB = unimask
enableunidade = True
Return
```

Agora, copie este programa para a janela do compilador BASIC, e salve ele.
Compile o programa da mesma maneira que você já fez as outras vezes.

Você terá de modificar novamente o Display LED de 7 SEGMENTOS pois mudamos os bits que controlam o acendimento de cada display.

Antes de rodar o simulador, lembre-se de setar o pino RD2 (ON) na janela do Microcontrolador.

Agora, quando você for rodar o simulador, ele vai rodar bem mais devagar do que as últimas simulações, pois este PIC é bem mais complicado de simular.

Quando você mudar o estado de RD2, a velocidade vai ser bem maior, e você irá ver os valores serem mostrados alternadamente em ambos os displays.

Inicialmente irá aparecer a leitura 0.0 .

Agora, mude o valor da tensão na entrada RA0 , e aguarde o resultado ser mostrado nos displays.

Por exemplo, coloque o valor de 2,53 , e obterá o valor nos displays de 5.0 .

Se você se lembrar de nosso esquema elétrico, quando temos uma entrada no PIC de 2,53, significa que temos o dobro da tensão na entrada do nosso projeto, portanto teremos uma tensão na entrada de 5,06 volts , que será mostrada arredondada como 5.0 volts !

Brinque com o simulador, veja como o programa funciona, e quando quiser gravar o seu PIC para fazer experiências reais, lembre-se de mudar os valores dos comandos WAITMS para os valores reais !!!!!

PROJETO 5 - USANDO DISPLAY LCD E MEMÓRIAS FLASH SERIAIS