**Aula 2**

SFR – Registrador de Função Especial

Configura o pino para saber qual a função dele. Utiliza um byte para isto.

**Utilizando o SFR para entradas digitais**

Programando bit a bit

TRIS***~~familia~~***bits.TRIS***~~familianumero~~***

Exemplo: TRIS***C***bits.TRIS***C2***

LAT***C***bits.LAT***C6***

PORT***C***bits.***RC7***

Família de Porta

A,B,C,D,E.

Número da família

De acordo com o pino do no Datasheet.

Sempre que houver um R no pino, ele pode ser configurado para ser entrada digital.

Programando a família, byte a byte

TRIS***~~familia~~*** ->configura todos os pinos da Família de porta.

Exemplo: TRISC = 0b00001001; 0b = número binário

TRISC = 9; número decimal (mesmo número que o de cima)

**Configura os pinos como pinos digitais**

TRIS = 0, saída (output).

TRIS= 1, entrada (input).

**Escreve na porta digital**

LAT = 0, saída em nível baixo.

LAT = 1, saída em nível alto.

**Leitura na porta digital**

PORT = 0, entrada com nível baixo.

PORT = 1, entrada com nível alto.

Variável Booleana

Se for = 0, é falso.

Se for ≠ 0, é verdadeiro.

Sendo assim, quando for apertar um botão, não é necessário fazer ***if(botao == 1)*** .

**#define**

Transforma o comando em uma variável mais fácil de entender.

Sintaxe

#define NOME COMANDO

Exemplo:

#define Led\_Verde LATCbits.LATC2

Pegadinha dos Operadores em C

/ -> divisão

% -> resto de divisão inteira

Em divisão, só existe resto de divisão caso os números sejam inteiros. Caso o número seja FLOAT, o resto da divisão fica no próprio número.

Para dividir como float o número deve ter ponto, simbolizando a vírgula.

Exemplo: 20.0/6

Operadores Lógicos

Operadores para o número. Qualquer numero que não for zero é verdadeiro.

&& -> AND

|| -> OR

! -> NOT

Operadores bit a bit. Pega bit a bit do número.

& -> AND bom para fazer máscara.

| -> OR

^ -> XOR

~ -> NOT

>> -> Deslocamento à direita a>>2 preenche os primeiros números com 0

<< ->Deslocamento à esquerda a<<3 preenche os últimos números com 0

Case Sintaxe

switch (VARIAVEL)

{

case (NUMERO1) :

.........

.........

break;

case (NUMERO2) :

.........

.........

break;

default :

.........

.........

break;

}

Os números são:

0b-> Binário

0x->Hexadecimal

Function Sintaxe

TIPO DA FUNÇÃO NOME DA FUNÇÃO (TIPO DA VARIÁVEL NOME DA VARIÁVEL , )

{

//Bloco de variáveis;

//Bloco da função;

return NUMERO COM O MESMO TIPO DA FUNÇÃO

}

Quando o tipo da função é do tipo VOID, não é necessário retornar nada.

Para chamar a função, basta colocar ela com seu código com suas variáveis.

Se a função ficar encima da função MAIN, é somente chamar-la. Caso não, basta colocar o include dela, colocando a primeira linha dela. (TIPO DA FUNÇÃO NOME DA FUNÇÃO (TIPO DA VARIÁVEL NOME DA VARIÁVEL , )

**Aula 3**

Display de 7 Segmentos

Catodo Comum

Lógica digital positiva.

5 Volts para ligar o led.

Terra comum.

Anodo Comum

Lógica digital negativa.

0,8 Volts para ligar o led.

Fonte comum.

Bibliotecas

Para bibliotecas do próprio copilador:  
#include <nome\_da\_biblioteca.h>

Para bibliotecas feitas pelo o usuário:

#include “nome\_da\_biblioteca.h”

Toda Biblioteca criada pelo usuário possui uma extensão “.c” e .”h” onde, “.c” possui o descritivo da função e o “.h” possui a chamada da função (protótipo).

O nome do “.c” e “.h” tem que possuir o mesmo nome.

Nunca se modifica o “.c”, sempre se modifica o “.h”. Por isto há um ***#define*** para o ***TRIS***.

Debounce

Utilizado para tirar o bounce (quicar) do Butão.

Criar um contador para borda ou criar um ***while*** que não faz nada quando o botão está ligado.

Delay

#include <delays.h> -> Delay Functions

São 2 tipos:

Por NOP (No Operation Program) -> Não processa nada enquanto o tempo necessário não chegar.

Por interrupção -> Enquanto processa, o tempo é contado e quando o tempo for feito, ele interrompe o que

TCY -> Tempo de Ciclo de Instrução

Para PIC18, são feito 4 instruções por ciclo.

Então para PIC18 e Clock 48 Megahertz :

1 TCY = 83,33 ns

Biblioteca <delays.h>

Obs: sempre que for colocar (0) como parâmetro, ele é o maior valor (256).

Para utilizar o delay, basta fazer regras de três.

1 TCY --------- 83,33 ns

X ---------- tempo necessário

Exemplo:

1 TCY ------83,33 ns

X ------- 200 ms

200.000.000/83,33 = 2.400.000 1TCY = 240.000 10TCY = 24.000 100TCY = 2.400 1KTCY =   
240 10KTCY

Delay10KTCYx(240) = 200ms

**Aula 4**

Variáveis vetoriais

char vetor01 -> 1 byte -> 8 bits

int vetor01 -> 2byte -> 16 bits

Lembrando que

char vetor01 [2] -> 2 byte -> 16 bits

vetor01 [x], sendo x a posição do vetor. Ele varia de 0 a 15 posições.

Strings

Se utiliza um vetor com os números da tabela ASCII para poder escrever.

char nome\_empresa [15] = {83,69,77,80,82,65,0};

Em C, se for escrito as letras entre aspas, o compilador transforma nos números da tabela ASCII.

char nome\_empresa [15] = {‘S’,’E’,’M’,’P’,R’,’A’,’\0’};

Como o vetor acima possui 15 posições, com o caráter nulo, 14 letras podem ser escritos.

Alguns exemplos que podem ser encontrados na apostila na página 105, tabela 12, de caracteres especiais.

‘\n’ -> nova linha.

‘\t’ -> tabulação.

Outra maneira de escrever é:

char nome\_empresa [15] = “SEMPRA”;

Neste exemplo, ele coloca o char nulo no final.

char nome\_empresa [] = “SEMPRA”;

Caso seja digitado desta maneira, o próprio compilador conta o número de posições necessárias para alocar estas letras, porém não há espaços vazios.

Modificadores RAM e ROM

char string[30] = “guarda na sram”;

ram char string1[30] = “guarda na sram”; (mesmo que o de cima, sendo assim pode ser ocultado)

rom char string2[30] = “guarda na flash”;

#include <string.h> -> Memory and String Manipulation Functions

Conferir quais os parâmetros na ajuda do programa.

Somente é STRING quando for vetor.

**char \*** -> apontador de endereço de memória.

Os comandos tem como parâmetro -> (string de destino, string de origem)

strcpy -> copiar na RAM

strcpy***pgm*** -> copiar na ROM

strcpy***pgm2ram*** -> copiar da ROM para RAM

strcpy***ram2pgm*** -> copiar da RAM para ROM

strcpy -> copia o string em outro.

Copia 1 string em outra e se não possuir espaço você perde o controle da string.

strcat -> cocatenar strings.

Concatena 2 strings e se não possuir espaço você perde o controle da string.

strcmp -> comparar strings.

Compara por ordem alfabética, devido aos números da tabela ASCII.

#include <stdlib.h> -> Data Conversion Functions

itoa -> int to ASCII

atoi -> ASCII to int

Variáveis globais e locais

Quando utilizamos a variável ROM, ela pode ser criada em duas posições, global, a qual todas funções podem acessar, e local, a qual apenas pode ser acessada apenas na função que ela foi definida.

Quando utilizamos a variável local, seu endereço se torna dinâmico, sendo assim, podendo ser apagado quando a rotina não estiver dentro daquela função. Para ela não ser apagada e não perder o endereço de memória, deve-se colocar o ***static*** antes, assim mantendo este endereço.

Exemplo: static char string[2];

**Aula 5**

Conversor Analógico / Digital

Além do erro do sensor, há o erro do conversor também.

Exemplo: Um medidor de temperatura de 0° a 100°C sendo convertido a um conversor de 1 byte.

Sendo assim o range do sensor é 100 (100°C - 0°C).

Sendo assim, o conversor possuiria 256 combinações.

100°C ----- 255

0°C ------ 0

N ------- X

Sendo assim o erro do conversor seria 100/255 = 0,3921 °C / bit .

Etapas de Conversão

Condicionamento do Sinal

Como o PIC foi alimentado com tensões de 0 a 5V, tensões maiores que esta não podem ser colocadas no pino, pois se não os pinos queimam. Pessoas também colocam um filtro para melhorar o sinal.

Entrada do sinal

Como conversores são muito caros, os microcontroladores costumam ter apenas um conversor analógico/digital. Para não apenas possuir um pino de conversor, um multiplexador é utilizado. Por causa disto que o PIC possui vários pinos de conversão.

Por apenas curiosidade de Hardware, antes do conversor, o sinal é guardado, para não ser perdido, para depois.

Freqüência de amostragem

Número de amostras em um segundo (Hz).

Critério de Naiste

Para ter uma freqüência boa de amostragem, ela tem que ser pelo menos 2 vezes maior que a freqüência do sistema.

f amostragem > 2\* f sistema

**Regra usual**

f amostragem ~ 10\*f sistema

Quanto maior forem os números de sensores, menor será a f amostragem para cada um. Sendo assim, lembrar de dividir a f amostragem pelo o número de amostras por sensor.

Se todos

Período de Amostragem (s)

Tempo entre duas amostras.

T =1/f

Uma forma de medir é a cada amostra convertida mandar ligar e desligar o LED. Com a ajuda de um osciloscópio, ao medir a onda no led, você tem o tempo real das amostras.

Resolução

Qual a menor medida que ele consegue fazer.

-Menor intervalo de sinal analógico que o CAD é capaz de detectar.

8 Bits - > valores de 0 a 255

10 Bits -> valores de 0 a 1.023

16 Bits -> valores de 0 e 65.535

Exemplo: Conversor de 5V em 10Bits

Resol ----- 1 um binária

5V ----- 1023

Resolução = 5V/1023 ~ 4,88 mV

5V ----- 100% 1023 ---- 100%

4,88 mv ----- x% 1 ----- x%

10 Bits -> Erro X% = 1/1023 \* 100% = 0,097% ~ 0,1%

8 Bits -> Erro X% = 1/255 \* 100% = 0,39% ~ 0,4%

Sendo assim:

Resolução = (Vrefmaior - Vrefmenor)/ (2^bits - 1)

Vvolts = Vbin \* Resolução

Vvolts = Tensão no pino.

Vbin = Tensão binária, de acordo com o conversor A/D.

Para mudar a referencia maior e menor, basta utilizar os pinos AN3 (Vref maior) e AN2 (Vref menor) para inserir as novas referencias e utilizar os bits VCFG1:VCFG0 para configurar-los.

Freqüência do Conversor A/D

f conversor A/D = f microcontrolador / X

Pag 273 do Datasheet -> mostra o mínimo de divisões para se utilizar a freqüência do microcontrolador.

f AD = f osc/64 = 750 kHz

T AD = Período de Conversão de 1 Bit

T AD = 1/f AD =1,33 microsegundo

T A = Periodo de Amostragem

T A = T acq + T cnv + T rep

T acq = Tempo de aquisição.

T acq\_min = 2,45 micro segundos

T cnv = Tempo de conversão.

T cnv\_min = 11 \* T AD

T rep = Tempo de Repouso.

T rep\_min = 3 \* T AD

T A\_min = 2,45 + 11\*T AD + 3\*T AD ~= 16\*T AD ~= 21,28 microsegundos

Freqüência máxima de amostragem do Conversor do PIC

F A\_max = 1/T A\_min = 47 kHz

Então para ler numa freqüência tão alta, tem que ter um esquema de comunicação com uma memória externa, tal como um PC.

Os registradores de pinos analógicos são ADCON (Pag 266 do Datasheet)

São os SFR para as entradas analógicas e das referências do Conversor.

A biblioteca do conversor do próprio compilador é a #include <adc.h>.

(Ajuda do programa -> PIC18F Peripheral Library Help Document)

OpenADC -> Liga e configura o conversor.

SetChanADC -> Escolhe o canal (do multiplex) para ser convertido.

SlChanConvADC -> Escolhe o canal (do multiplex) para ser convertido e começa a conversão.

ConvertADC -> Inicia a conversão do canal que foi escolhido anteriormente.

BusyADC -> Retorna o estado da conversão.

ReadADC -> Ler o ultimo valor registrado no conversor.

CloseADC -> Desliga o conversor.

**Aula 6**

\_asm \_endasm

Utilizado para escrever em Assemble em C.

#pragma code #pragma code

Utilizado para escrever em um endereço específico de memória.

#pragma interrupt -> define qual o endereço para a interrupção de alta prioridade.

#pragma interruptlow -> define qual o endereço para a interrupção de baixa prioridade.

Em C, definição de endereço se faz através de funções. Então ao chamar a função, você está entregando o endereço de memória para o microcontrolador.

20 Fontes de interrupções

16 internas e 4 externas

Na placa, conferir o PORTB, pois neste PIC todas as interrupções. Eles estão embaixo do display de 7 segmentos, na parte inferior. Se for utilizar interrupção, somente retirar este display.

Interrupções Internas

Timer interno

Conversão AD

Recebimento de Dados

Registradores de Interrupção para as interrupções internas

PIR -> Registro (Flags)

PIE -> Enable

IPR -> Prioridade

Interrupções Externas

-3 Pinos distintos e mais 4 Pinos de Keyboard Interruption que contam como 1.

3 Pinos de interrupção distintos através de bordas (INT 0:2)

KB -> Keyboard Interruption -> Qualquer um dos botões que apertar causa interrupção (KBI 0:3)

Na placa, apenas o botão\_boot possui interrupção externa.

INTCON -> registrador de interrupção para a PORTB

-Flag -> Indica interrupção. INTCONbits.RBIF = ;

-Enable -> Habilita a interrupção. INTCONbits.RBIE = 1;

-Priority -> Configura como alta ou baixa prioridade. INTCON2bits.RBIP = 0;

GIEH -> Habilita ou desabilita TODAS as interrupções. Se habilitada, as interrupções de alta prioridade estão habilitadas

INTCONbits.GIEH =;

GIEL - > Habilita ou desabilita as interrupções de baixa prioridade.

INTCONbits.GIEL =;

(Mesmo bit que os de cima, porém o nome muda se o IPEN = 0)

GIE -> Habilita ou desabilita TODAS as interrupções.

PEIE -> Habilita ou desabilita TODAS as interrupções periféricas.

RCON -> Indica a ordem em que as interrupções ocorrem

Se estiver zerado, as interrupções vêem na ordem de chegada.

RCONbits.IPEN=1;

Todos os pinos do PORTB possuem resistor de PULL UP interno (uma fonte ligada ao pino e ao apertar, joga 0 na entrada).

Ele esta na INTCON2.

RBPU -> Habilita o Resistor de PULL UP. (Ele é em lógica negativa)

INTCON2bits.RBPU=1; (desabilita TODOS os resistores de PULL UP)

Interrupções não podem demorar muito, porque se não podem atrasar muito o programa principal, a não ser que seja o desejado.

A errata do Chip do PIC, somente permite zerar o flag de interrupção se ler a PORTB antes.

Timer

#include <timers.h>

O clock que entra no timer já é a freqüência da CPU/4, além do prescaler.

Nesta placa, o maior tempo de sem prescaler é de 5,5 ms.

O prescaler pode dividir o clock da CPU em até 256.

Reg\_TMR0 = overflow – [(tempo desejado \* f int)/prescaler]

Reg\_TMR0 = o tempo que você tem que carregar inicialmente no timer para o tempo ficar correto.

Os registradores dos timers são:

T0CON -> Configuração do Timer0

TMR0H -> Armazena o valor do timer com os bits mais significativos

TMR0L -> Armazena o valor do timer com os bits menos significativos

//faltando item

INTCON2bits.TMR0IP -> prioridade

INTCONbits.TMR0IE -> enable

INTCONbits.TMR0IF -> flag