Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

MINICURSO

Microcontrolador PIC 16F877

Alison Lins de Lima

Fábio Montenegro Pontes

Jonathan B. da Silva

Rafael Tavares Coutinho

Thiago Pinto

João Pessoa, 23 de Maio de 2010

Plano de Curso

1. **Evolução da Tecnologia e Surgimento dos microcontroladores;**
2. **Teoria sobre Microcontrolador**

    2.1 - O que é um microcontrolador e onde se aplica;

    2.2 - Diagrama de Blocos básico de um microcontrolador;

    2.3 - Arquitetura e filosofia;

3. **Memórias**

    3.1 - RAM

    3.2 - ROM

4. **Tipos de PIC (quanto à memória de programa)**
5. **PC e a Pilha**
6. **Interrupções**
7. **Polarização do Pic16F877**

    7.1 – Alimentação

    7.2 – Circuitos Osciladores

8. **Programação**

    8.1 – Assembler

    8.2 - C

9. **Comandos Básicos**
10. **Estrutura básica de um código em C**
11. **Práticas**

    11.1 – acionando um led

    11.2 – pisca – pisca com led e variando a frequência;

    11.3 – Contador com Display de sete segmentos;

    - Verificar a diferença do código para o ânodo comum e o cátodo comum;

    11.4 – Escrevendo em um LCD de 16x2;

    11.5 – Usando um botão, para acionar um led;

ANEXOS

1. Evolução da Tecnologia e Surgimento dos microcontroladores;

2. Teoria sobre Microcontrolador

2.1. O que é um microcontrolador e onde podemos encontrá-lo.

Def 1.: Podemos definir um microcontrolador com sendo um computador simplificado em um único circuito integrado, utilizado no controle de processos lógicos. (Entende-se como processo lógico o acionamento de leds, displays sete seguimentos, lcds, relés, sensores ...)

Def 2.: São circuitos integrados dotados internamente de um ULA (unidade lógica e aritmética), uma unidade de controle e conjunto de periféricos (ex.: contadores, temporizadores, portas I/O, memórias).

Aplicações:

Indústria Automobilística
- i) Injeção Eletrônica;
- ii) Fechaduras Eletrônicas;

Empresas de Segurança
- i) Alarmes;
- ii) Sensores de movimentação;
- iii) Cercas Elétricas;

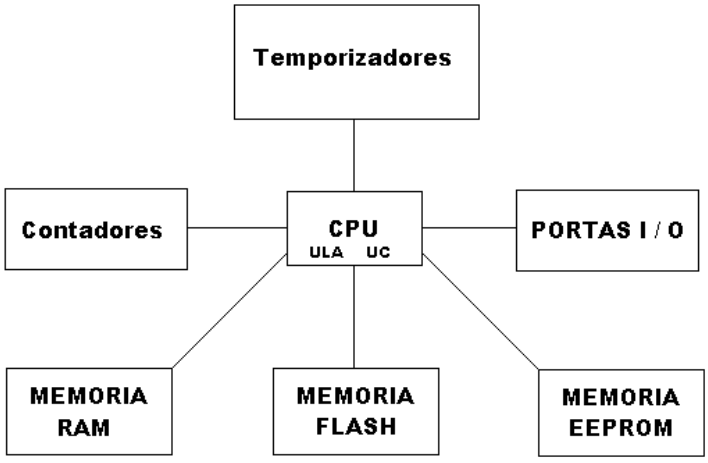
Indústria de Telefonia
- i) Celulares
- ii) Binas
- iii) Centrais Telefônicas

Robótica
- i) Robôs para competição e de uso geral

Brinquedos
- i) Bonecos
- ii) Aeromodelismo

Saúde
- i) Equipamentos Médicos Hospitalares

2.2. Diagrama de Blocos básico de um microcontrolador



2.3. Arquitetura e filosofia

Def.: São as partes que constitui o microcontrolador e como as mesmas estão interligadas.

Von – Neuman (CISC) - Primeiras arquiteturas pensadas para computadores.

Harvard (RISC)

As arquiteturas se diferenciam pela ligação da CPU com as memórias de dados e a de programa, na arquitetura de Harvard, a CPU esta conectada as memórias por barramentos separados o que permite o uso do conjunto de instruções (RISC), já na arquitetura de Von – Neumann, a CPU esta conectada as memórias pelo mesmo barramento, isso demanda o uso de conjunto de instruções complexas (CISC).

Obs.: Arquitetura do PIC 16F877, encontra-se nos anexos.

3. Memórias (VER ANEXO II)

3.1 – RAM

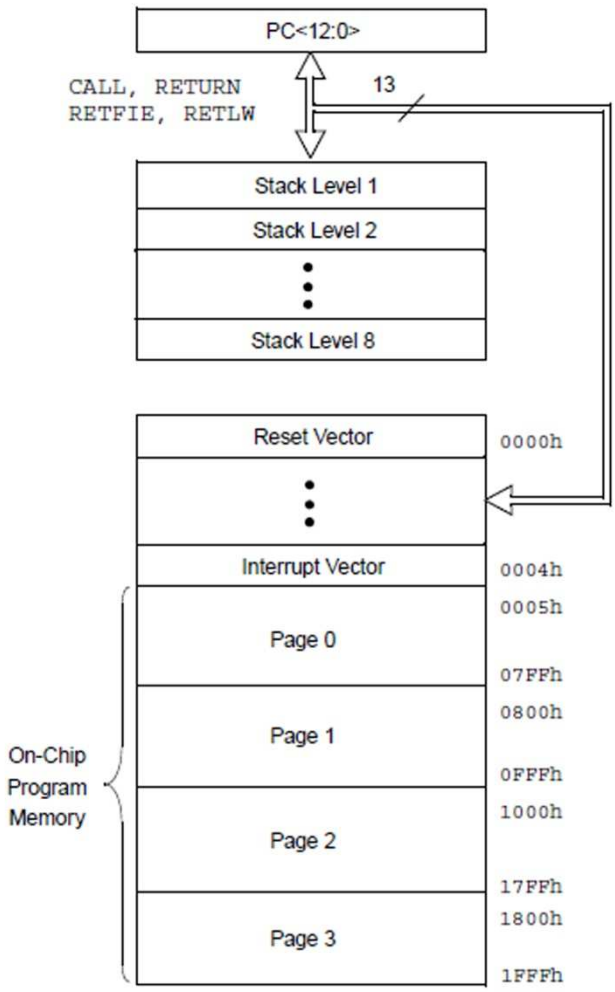
As memórias RAM são memória de uso temporário, voláteis os dados só permanecem na memória enquanto está ligada, quando a memória é desligada todos os dados são perdidos.

3.2 - ROM

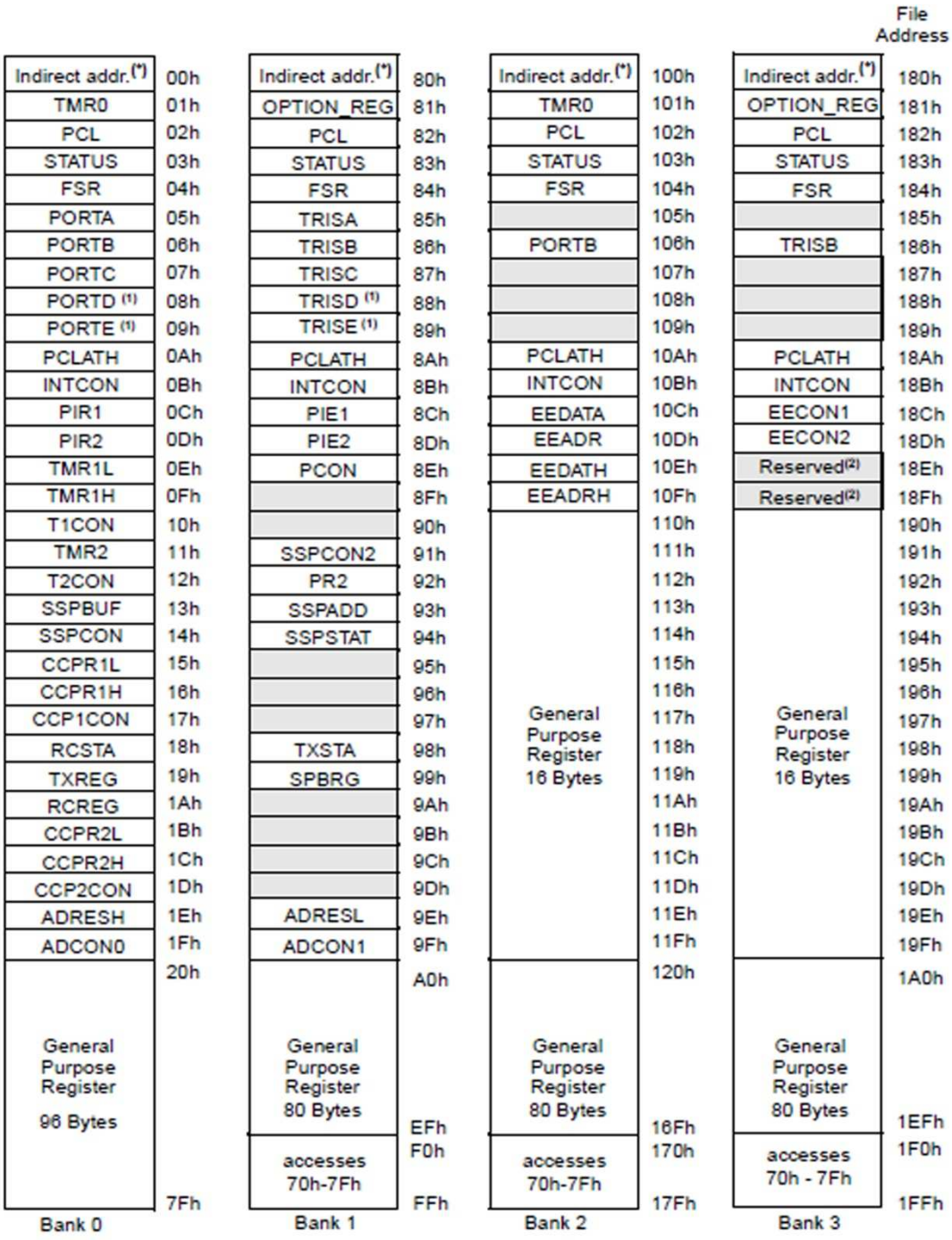
São memória de gravação permanente, mesmo quando estão desligadas os dados permanecem guardados.

Principais Características das memórias de programa e dados do PIC 16F877:

Memória de Programa (Flash)
8k x 14 bits
Memória de dados (Ram)
368 bytes
Memória de dados (EEprom)
256 bytes



Memória de dados possui registradores de propósitos gerais (GFR) e de propósitos especiais (SFR).



4. Classificação dos PICs (quanto à memória de programa)

PIC 16**C**711 C – PROM, grava apenas uma vez.

PIC 16**CL**7XX CL – EPROM, grava por eletricidade e apaga por luz ultravioleta.

PIC 16**F**877 F – FLASH, grava e apaga por eletricidade.

5. PC e a Pilha

PC é um registrador que aponta para o endereço da próxima instrução a ser executada.

O PC tem 13bits, com isso pode endereçar 8k de memória.

$$\mathbf{2^{13} = 8192\ \ endereços}$$

$$\mathbf{2^{13} = \frac{8192}{1024} = 8k}$$

OBS.: A partir do PIC 16f84 o PC já possuía 13 bits, mesmo tendo uma memória de dados de 1k x 14 bits.

A pilha é um registrador de oito níveis, no qual são guardados os endereços de retorno das rotinas de interrupções.

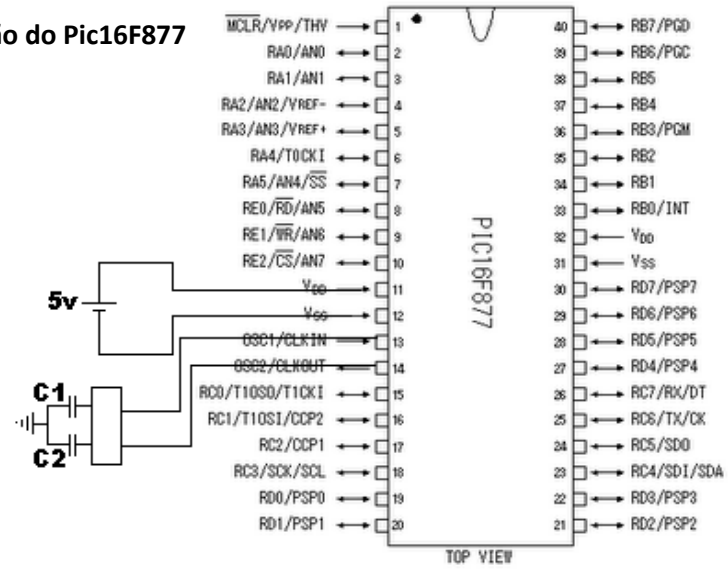
endereço 3
endereço 2
endereço 1

"Quando acontece a primeira interrupção, o endereço armazenado em PC vai para o primeiro nível da pilha, se por acaso acontece uma segunda interrupção então novamente o endereço que esta dentro do PC ocupa o primeiro nível da pilha e o endereço que antes ocupava o primeiro nível desce para o segundo, quando a interrupção 2 acaba, o endereço que esta no primeiro nível, é posto novamente no PC, que retorna para a posição em que havia parado na primeira interrupção, daí o endereço 1 volta para o primeiro nível da pilhar e daí por diante..."

6. Interrupções

É uma parada de emergência na execução de uma rotina, que ocorre por um fato interno ou externo e resulta na chamada de uma sub-rotina que começa ser executada e quando esta acaba o microcontrolador volta a executar a rotina principal.

7. Polarização do Pic16F877



C1 = C2 27pF

8. Programação

8.1 – Assembler

É uma linguagem de baixo nível ou linguagem de máquina.
Vantagem: Código com o menor tamanho que se pode escrever;
Desvantagem: Cada microcontrolador tem seu conjunto de instruções; (**Condição Péssima**)

8.2 – C

É uma linguagem de alto nível mais fácil de aprender em relação ao assembler.

Vantagem: facilidade no aprendizado e portável para outros microcontroladores;
Desvantagem: código maior do que se o mesmo fosse programado em assembler.

9. Comandos Básicos de:

Entrada

Tratamento a BIT	Tratamento a PORTA
input (PIN_D0)	input _D()
input (PIN_D0)	input _D()

Saída

Tratamento a BIT	Tratamento a PORTA
output_low(PIN_D0)	output_D(valor)
output_high(PIN_D0)	output_D(valor)

Loop

for(i=0; i>0; i++)	for(; ;) loop infinito
while(condição)	while(1) ou while(true) loop infinito
do{ } while(codição)	
switch / case	

Condicional

if (condição){ }else{}

Atraso

delay_ms(1000)	atraso em milisegundos
delay_us(1000)	atraso em microsegundos

10. Estrutura Básica de um programa em C

```
#include <16f877.h>                          // inclusão da Biblioteca do microcontrolador
#fuses HS,NOWDT,NOPUT,NOBROWNOUT,NOLVP       // Palavras de configurações  do microcontrolador
#use delay(clock=20000000)                   // Valor do clock

void main (){                                // Corpo principal do programa
        comando 1                            //
        comando 2                            // Comandos
        comando 3                            //
}
```

11. Práticas

Acionando um led

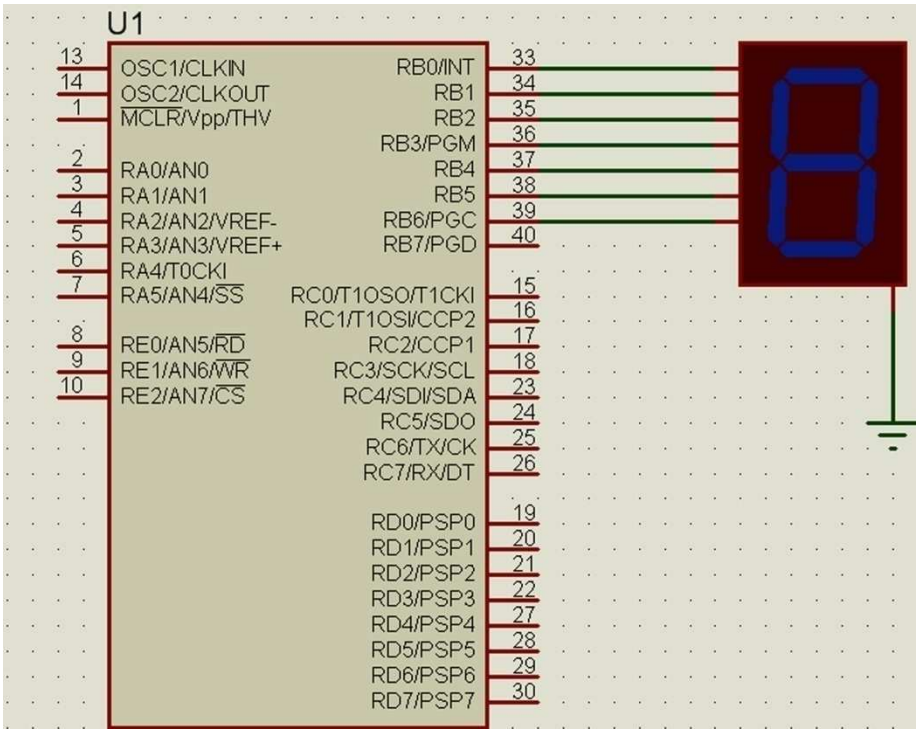
```
#include <16f877.h>
#fuses HS,NOWDT,NOPUT,NOBROWNOUT,NOLVP
#use delay(clock=20000000)
void main (){
        output_high(PIN_d0);
}
```

Pisca – pisca com led e variando a frequência

```
#include <16f877.h>
#fuses HS,NOWDT,NOPUT,NOBROWNOUT,NOLVP
#use delay(clock=20000000)
void main (){
        while(1){
                output_high(PIN_d0);
                delay_ms(1000);
                output_low(PIN_d0);
                delay_ms(1000);
        }
}
```

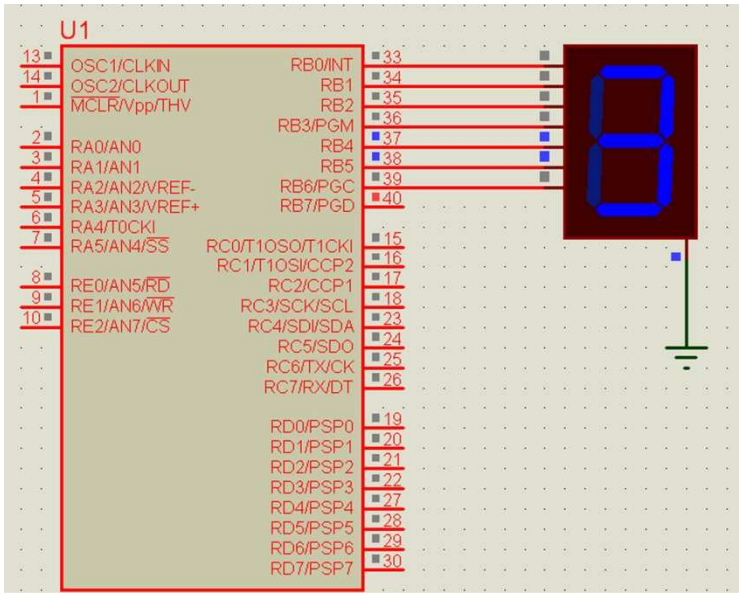
Contador com Display de sete segmentos versão 1

```
#include <16f877.h>
#fuses hs, NOLVP, NOWDT, NOPUT, NOPROTECT, NOBROWNOUT
#Use delay(clock = 20000000)
void main(){
  while(1){
    output_b(0b0000110);   // 1
    delay_ms(1000);
    output_b(0b1011011);   // 2
    delay_ms(1000);
    output_b(0b1001111);   // 3
    delay_ms(1000);
    output_b(0b1100110);   // 4
    delay_ms(1000);
    output_b(0b1101101);   // 5
    delay_ms(1000);
    output_b(5);
    delay_ms(1000);
    output_b(6);
    delay_ms(1000);
    output_b(7);
    delay_ms(1000);
  }
}
```

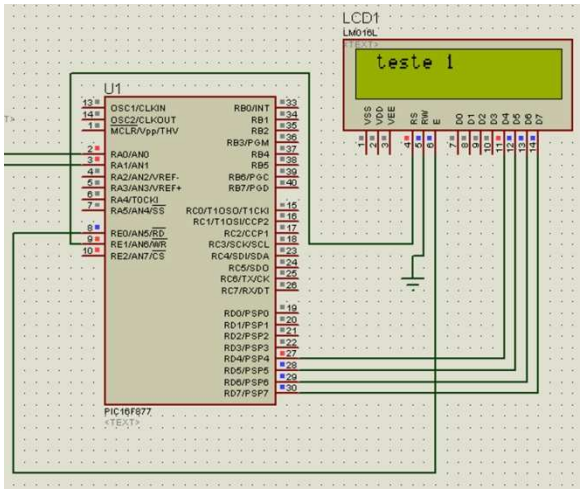


Contador com Display de sete segmentos versão 2

```
#include <16f877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=16000000)
byte const tabela[]={
        0b10111111,
        0b10000110,
        0b11011011,
        0b11001111,
        0b11100110,
        0b11101101,
        0b11111101,
        0b10000111,
        0b11111111,
        0b11100111
};
Void main(){
        int t=500,valor = 0;
        while(true){
                valor ++;
                if(valor>9)
                        valor = 0;
                output_b (tabela[valor]); // apresenta o valor
    delay_ms(t);
        }
}
```



Escrevendo em um LCD de 16x2;




```
#include <16f877.h>
#include <regs_16f87x.h>
#fuses HS,NOWDT,NOPUT,NOBROWNOUT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <bib_LCD_16x2_all.c>
void main (){
        ini_lcd_16x2();
         while(true){
                printf(exibe_lcd,"\f PET ENGENHARIA  \n ELETRICA  IFPB");
                delay_ms(3000);

                printf(exibe_lcd,"\f    OFERECE    \n O MELHOR CURSO");
                delay_ms(3000);

                printf(exibe_lcd,"\f      DE      \nMICROCONTROLADOR");
                 delay_ms(3000);
          }
}
```

Usando um botão, para acionar um led;

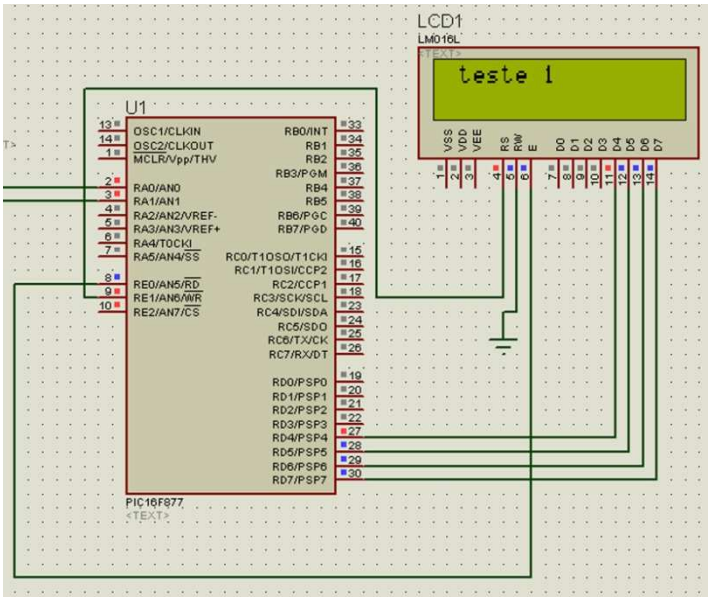
```
#include <16f877.h>
#fuses hs,nolvp,nobrownout,nowrt
#use delay(clock=20000000)
void main(){
  while(true){
    if(!input(pin_a0)) {       // se a0 for pressionado acende o led
      output_high(pin_d0);
    }
    if(!input(pin_a1)){        // se a0 for pressionado desliga o led
      output_low(pin_d0);
    }
  }
}
```

Usando um sensor para acionar um led;

O código é semelhante ao anterior, a mudança acontece apenas no hardware, ao invés de usarmos um botão na entrada iremos utilizar um sensor.

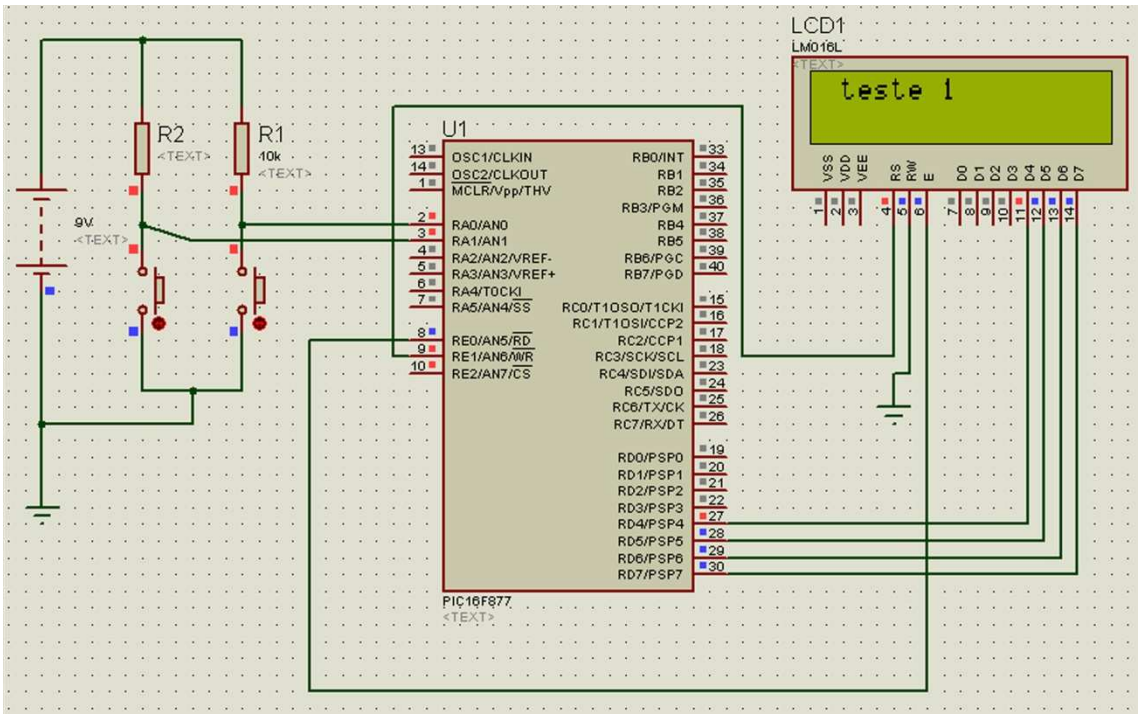
Escrevendo uma variável no LCD. (variável inteira)

```
#include <16f877.h>
#include <regs_16f87x.h>
#fuses HS,NOWDT,NOPUT,NOBROWNOUT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <bib_LCD_16x2_all.c>
int valor;
void main (){
        ini_lcd_16x2();
         valor = 0;
        while(true){
                printf(exibe_lcd,"\f teste %d",valor);
                 delay_ms(100);
                 valor++;
        }

}
```



Imprimindo uma variável no LCD, onde essa variável é o valor da porta.

```
#include <16f877.h>
#include <regs_16f87x.h>
#fuses HS,NOWDT,NOPUT,NOBROWNOUT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <bib_LCD_16x2_all.c>
int valor;
void main (){
        ini_lcd_16x2();
         valor = 0;
        while(true){
                valor = input(pin_a0);
                 if(input(pin_a0)) {       // se a0 for pressionado acende o led
                        output_high(pin_e2);
                 }

                printf(exibe_lcd,"\f teste %d",valor);
                delay_ms(100);
        }
}
```



ENTRADA E SAÍDA

Chave	LED
TRM0	0
TRM1	1
CH0	2
INT	3
CH1	4

Obs.: os LEDs e as chaves são baixo ativos.

Modo Padrão (mais simples)

```
#include <16F871.H
#fuses HS,NOWDT,PUT,NOBROWNOUT,NOLVP
#use delay(clock=4000000)

void main (void){
  for(;;){            //loop infinito
    if(!input(PIN_A4))   output_low(PIN_D0);
    else          output_high(PIN_D0);

    if(!input(PIN_C0))   output_low(PIN_D1);
    else          output_high(PIN_D1);

    if(!input(PIN_C1))   output_low(PIN_D2);
    else          output_high(PIN_D2);

    if(!input(PIN_B0))   output_low(PIN_D3);
    else          output_high(PIN_D3);

    if(!input(PIN_E2))   output_low(PIN_D4);
    else          output_high(PIN_D4);
  }
}
```

Modo Direto

```
#include <16F871.H>     // arquivo de definições do microcontrolador usado
#fuses HS,NOWDT,PUT,NOBROWNOUT,NOLVP    // bits de configuração

#use delay(clock=4000000)   // informa ao sistema o frequência de clock, para temporização

//    Endereços dos portais
#byte   PORTA=0x05
#byte   PORTB=0x06
#byte   PORTC=0x07
#byte   PORTD=0x08
#byte   PORTE=0x09
```

```
//   Entradas
#bit   CH_TMR0 = PORTA.4
#bit   CH_TMR1 = PORTC.0
#bit   CH_0 = PORTC.1
#bit   CH_INT = PORTB.0
#bit   CH_1 = PORTE.2

//    Saídas
#bit   LED0 = PORTD.0
#bit   LED1 = PORTD.1
#bit   LED2 = PORTD.2
#bit   LED3 = PORTD.3
#bit   LED4 = PORTD.4

void main (void)
{
  set_tris_d(0xE0);      // configuração direcional: os 5 LSbits do portal D são saídas

  for(;;){              //loop infinito
    if(!CH_TMR0)   LED0 = 0;
    else        LED0 = 1;

    if(!CH_TMR1)   LED1 = false;
    else        LED1 = true;

    if(!CH_0)      LED2 = false;
    else        LED2 = true;

    if(!CH_INT)      LED3 = false;
    else        LED3 = true;

    if(!CH_1)      LED4 = false;
    else        LED4 = true;
  }
}
```

Modo Fixo

```
#include <16F871.H>      // arquivo de definições do microcontrolador usado
#fuses HS,NOWDT,PUT,NOBROWNOUT,NOLVP    // bits de configuração

#use   fixed_io(d_outputs = PIN_D0,PIN_D1,PIN_D2,PIN_D3,PIN_D4) // saídas da porta D. Os
demais são entradas

#use delay(clock=4000000)   // informa ao sistema o frequência de clock, para temporização

//   Entradas
#DEFINE   CH_TMR0   PIN_A4
#DEFINE   CH_TMR1    PIN_C0
#DEFINE   CH_0      PIN_C1
#DEFINE   CH_INT     PIN_B0
#DEFINE   CH_1     PIN_E2
```

```
//   Saídas

#DEFINE   LED0     PIN_D0
#DEFINE   LED1     PIN_D1
#DEFINE   LED2     PIN_D2
#DEFINE   LED3     PIN_D3
#DEFINE   LED4     PIN_D4

void main (void)
{
  for(;;){            //loop infinito
    if(!input(CH_TMR0))   output_low(LED0);
    else          output_high(LED0);

    if(!input(CH_TMR1))   output_low(LED1);
    else          output_high(LED1);

    if(!input(CH_0))     output_low(LED2);
    else          output_high(LED2);

    if(!input(CH_INT))   output_low(LED3);
    else          output_high(LED3);

    if(!input(CH_1))     output_low(LED4);
    else          output_high(LED4);
  }
}
```

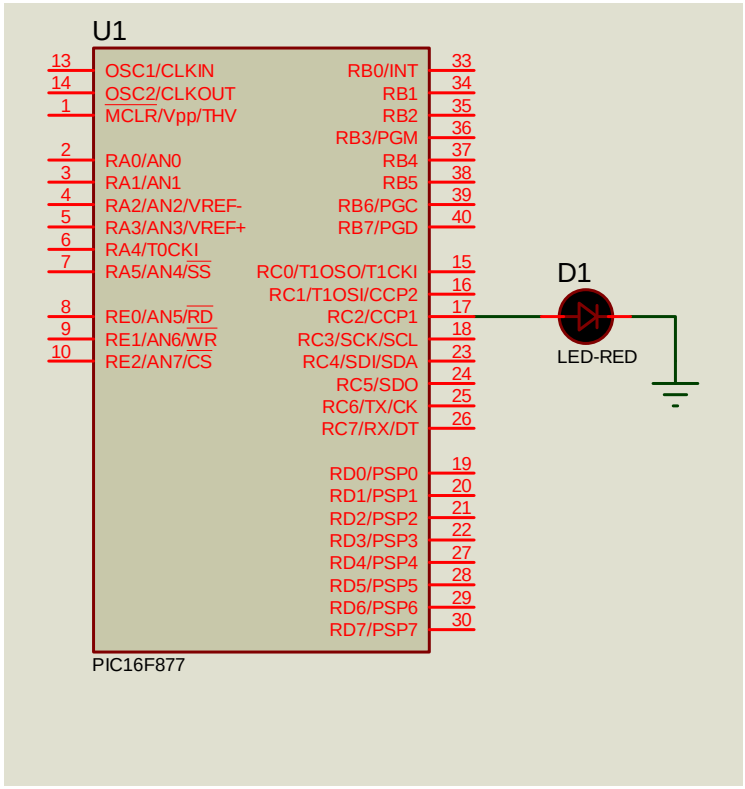
Modo Rápido

```
#include <16F871.H
#fuses HS,NOWDT,PUT,NOBROWNOUT,NOLVP

#use fast_io(A)       // acesso ao portal A feito pelo modo rápido
#use fast_io(B)       // acesso ao portal B feito pelo modo rápido
#use fast_io(C)       // acesso ao portal C feito pelo modo rápido
#use fast_io(D)       // acesso ao portal D feito pelo modo rápido
#use fast_io(E)       // acesso ao portal E feito pelo modo rápido

#use delay(clock=4000000)


//   Entradas
#DEFINE   CH_TMR0   PIN_A4
#DEFINE   CH_TMR1    PIN_C0
#DEFINE   CH_0     PIN_C1
#DEFINE   CH_INT     PIN_B0
#DEFINE   CH_1     PIN_E2

//   Saídas
#DEFINE   LED0     PIN_D0
#DEFINE   LED1     PIN_D1
```

```
#DEFINE   LED2     PIN_D2
#DEFINE   LED3     PIN_D3
#DEFINE   LED4     PIN_D4

void main (void)
{
  set_tris_d(0xE0);     // configuração direcional: os 5 LSbits do portal D são saídas

  for(;;){
    if(!input(CH_TMR0))   output_low(LED0);
    else          output_high(LED0);

    if(!input(CH_TMR1))   output_low(LED1);
    else          output_high(LED1);

    if(!input(CH_0))     output_low(LED2);
    else          output_high(LED2);

    if(!input(CH_INT))   output_low(LED3);
    else          output_high(LED3);

    if(!input(CH_1))     output_low(LED4);
    else          output_high(LED4);
  }
}
```

Controlando o brilho do LED com o Módulo PWM

```
#include<16F877a.h>
#device adc=10
#fuses   HS,NOWDT,NOLVP,NOBROWNOUT
#use delay(clock=20000000)

void main (void){

  long int ciclo = 0;

  setup_timer_2 (T2_DIV_BY_1,0xff,1);     // 4 khz
  setup_ccp1 (ccp_pwm);
  set_pwm1_duty (0);
  while(1){                          //loop infinito
    for(ciclo = 0;ciclo<=500;ciclo+=10){
      set_pwm1_duty(ciclo);
      delay_ms(70);
    }
    for(ciclo = 500;ciclo>=0;ciclo-=10){
      set_pwm1_duty(ciclo);
      delay_ms(70);
    }
  }
}
```

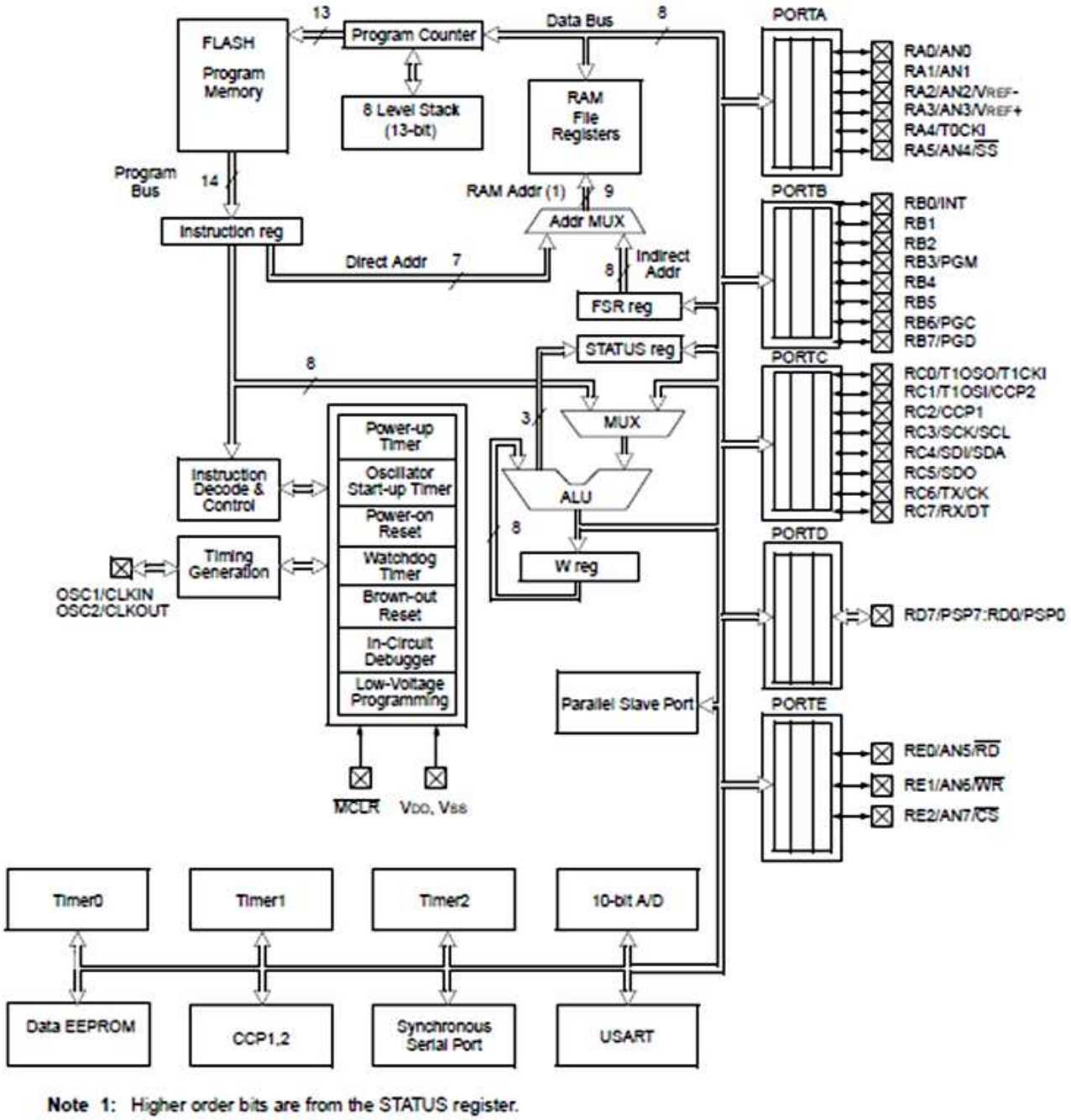


ANEXOS

ANEXO I

Arquitetura do PIC16F877

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

ANEXO II

PIC 16F84

ORGANIZAÇÃO E CARACTERÍSTCAS DAS MEMÓRIAS DE PROGRAMA E DE DADOS

MEMÓRIA DE PROGRAMA : (EEPROM FLASH 1k x 14)

VETOR RESET: 00H
VETOR INTERRUPÇÃO: 04H

A PILHA E O PC

MEMÓRIA DE DADOS: (RAM 68x8)** SFR e GFR (EEPROM 64x8)

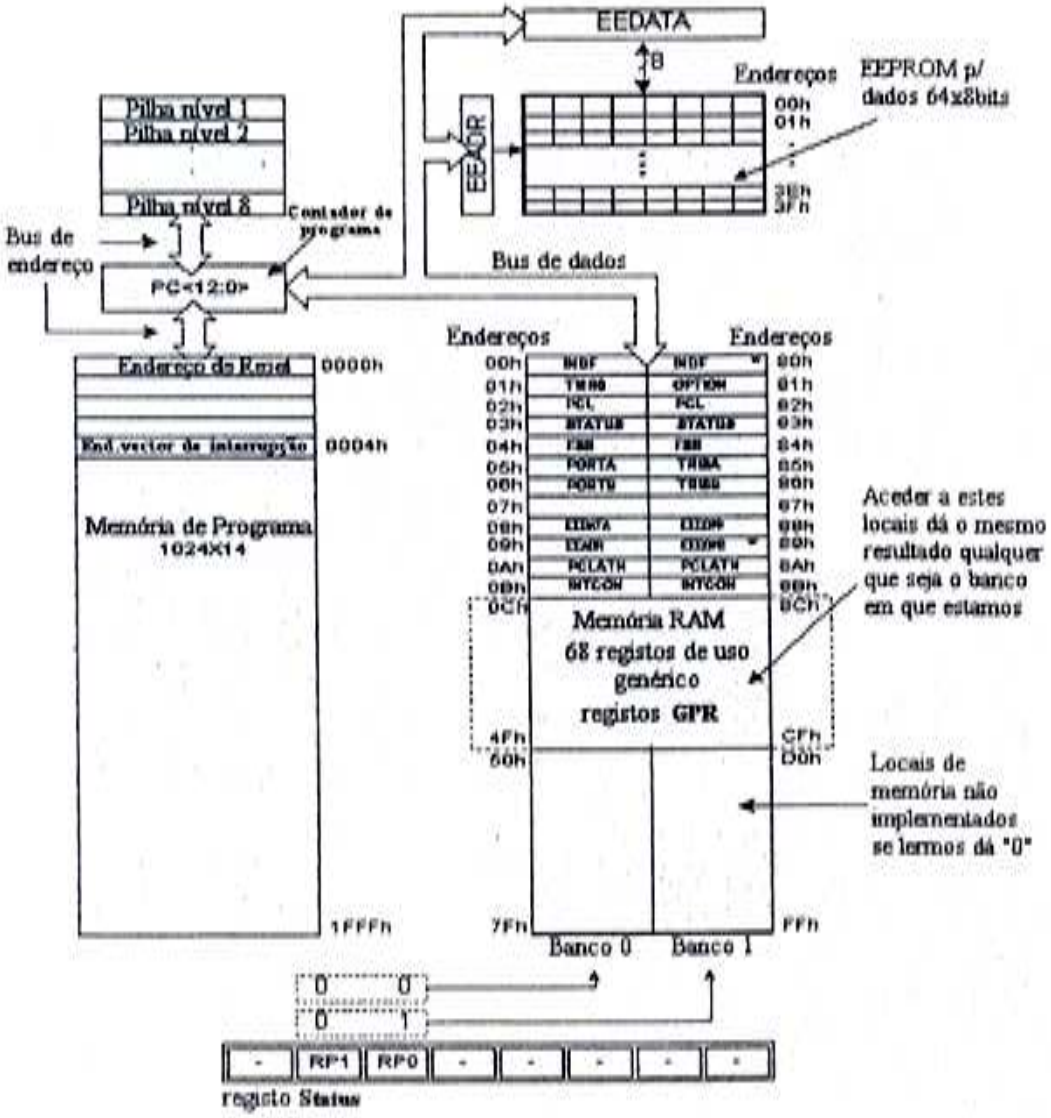
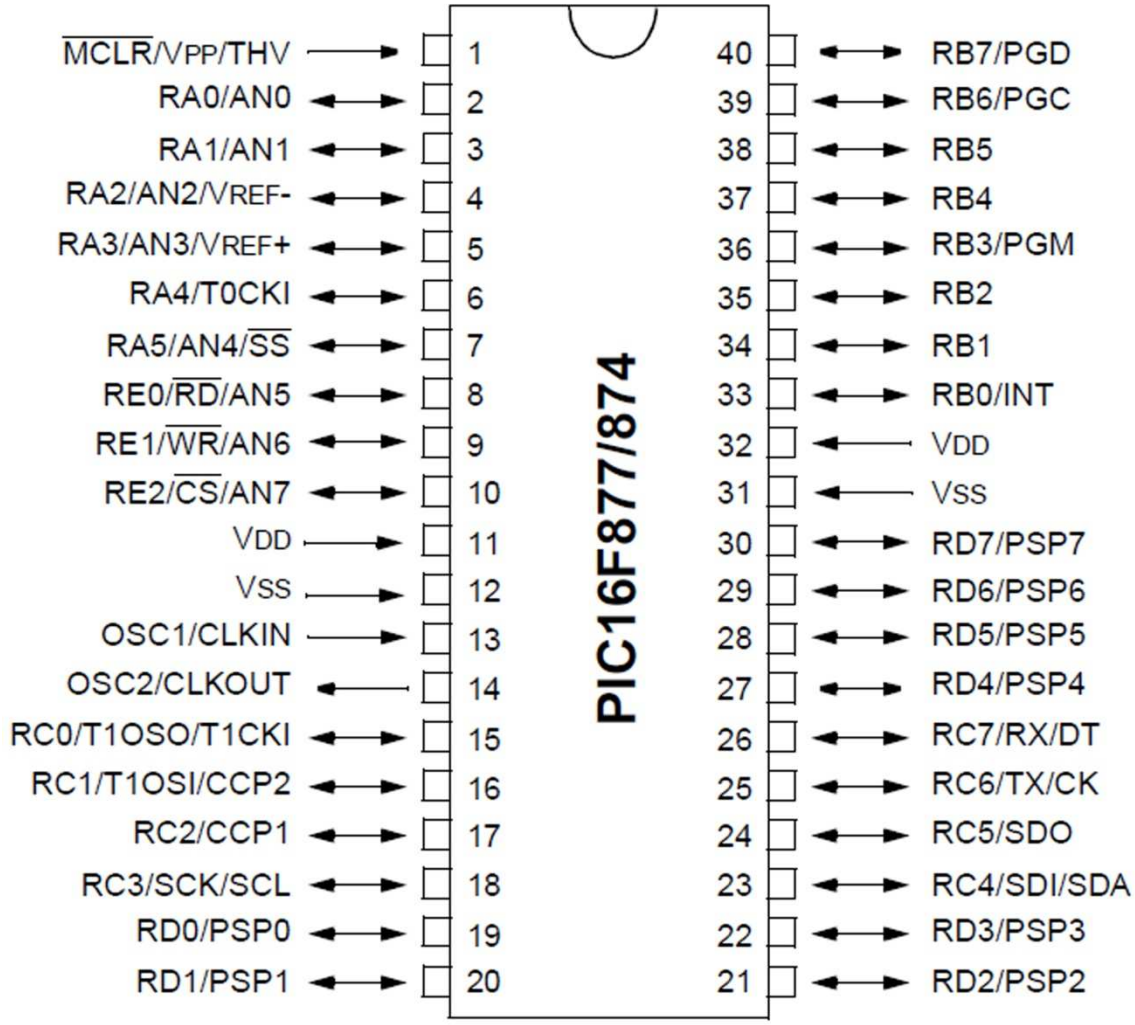


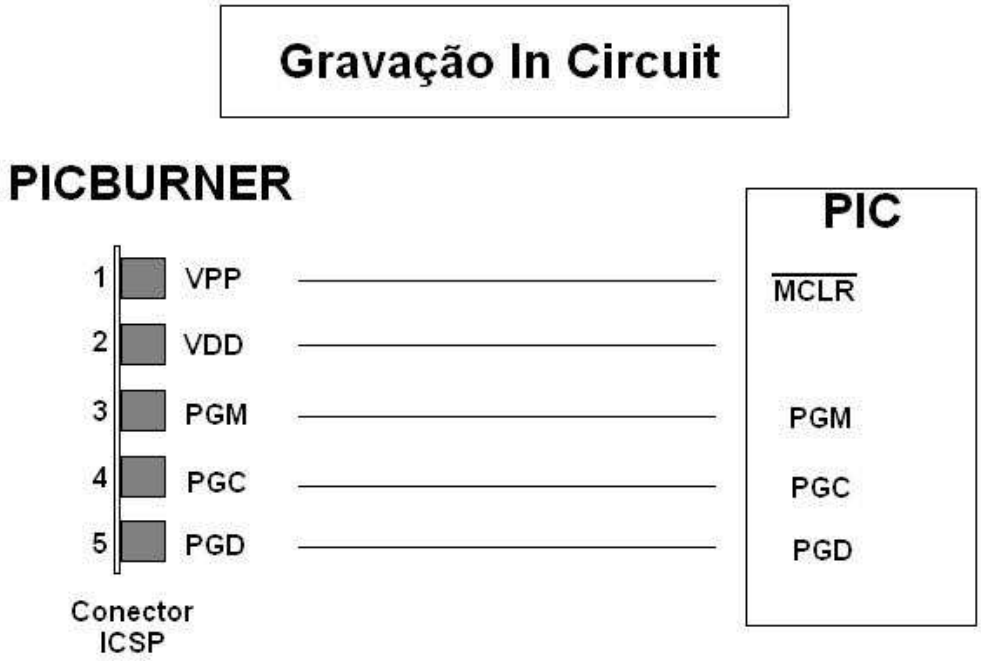
FIG.9-ORGANIZAÇÃO DA MEMÓRIA NO MICROCONTROLADOR PIC16F84

ANEXO III

PINAGEM 16F877



ANEXO IV



Alison Lins