UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA

FATECS – FACULDADE DE TECNOLOGIA

E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

FLÁVIA ROSA DE SOUSA PEREIRA

MONITORAÇÃO DE NÍVEL SONORO

BRASÍLIA/DF

1º SEMESTRE DE 2008

FLÁVIA ROSA DE SOUSA PEREIRA

MONITORAÇÃO DE NÍVEL SONORO

Monografia apresentada ao Curso de Engenharia da Computação, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Prof. M.C. Claudio Penedo

BRASÍLIA/DF

1º SEMESTRE DE 2008

Resumo

Existem ambientes nos quais é necessário que haja silêncio. Um exemplo muito próximo ao meio acadêmico é a biblioteca, onde é comum se ouvir conversas em alto tom. Hoje as bibliotecas, disponibilizam funcionários que ficam observando o comportamento das pessoas e solicitando silêncio quando necessário. O problema dessa solução é que se reserva uma determinada quantidade de funcionários para esta tarefa. Com a utilização do sistema de monitoração sonoro aqui proposto a quantidade de funcionários destinados a esta tarefa poderia ser reduzida. Ele é capaz de identificar um alto nível sonoro e encaminhar uma mensagem solicitando silêncio ao local ruidoso monitorado. Caso a solicitação não seja atendida, o dispositivo de mesa encaminha uma mensagem ao dispositivo central. O dispositivo central tem o objetivo de apenas identificar o local onde há o alto nível sonoro.

Palavras-chaves: Nível sonoro, sensor sonoro, PIC16F877A

Abstract

There are environments in which there must be silence. An example close to academia is the library, where it is common to hear conversations in high tone. Today the libraries provide employees who are observing the behaviour of people and asking silence when it is necessary. The problem is that this solution will reserve a certain amount of employees for this task. With the use of noise monitoring system proposed here, the number of employees for this task could be reduced. It is able to identify a high noise level and to forward a message requesting silence the noisy place monitored. If the request is not answered, the table device forward a message to the central device. The central device has goal of only identify the place where there is a high noise level.

Keywords: Level sound, sensor sound, PIC16F877A

Agradecimentos

Sou muito grata a todos que me apoiaram na realização deste projeto.

Agradeço a Deus, por não me deixar cair, por saber colocar as pessoas certas na hora certa em minha vida.

Aos meus pais, sem eles eu não estaria terminando este curso.

À minha irmã, que sempre me apoiou.

À professora Maria Marony, que quando eu não sabia para onde ir, deu-me excelentes dicas.

Ao professor Claudio Penedo pelo profissionalismo e dedicação. Sempre me apoiando no necessário para a conclusão deste trabalho.

E em especial, agradeço ao meu marido, Daniel, que é a minha fortaleza. Ele soube me dar paz e carinho nos grandes momentos de dificuldades.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Equações

Glossário

ABNT – Associação Brasileira de Normas Técnicas

ASCII -American Standard Code for Information Interchange

CKint – Clock interno

CKext – Clock externo

CM ou ***Tcy*** – Ciclo de máquina

FIFO – Fisrt In First Out

I^2C – Inter- Integred Circuit

USART – Universal Synchronous Asynchronous Receiver Transmitter

XT – Oscilador cristal

Lista de Símbolos

V – Unidade de tensão elétrica, Volt

Ω – Unidade de resistência, ohms

s – Unidade de tempo, segundo

F – Unidade de capacitância, Farad

Hz – Unidade de freqüência, Hertz

1 INTRODUÇÃO

A sociedade tem a demanda de um maior controle de ruído sonoro em alguns ambientes, a exemplo das bibliotecas. Isso porque muitas pessoas não respeitam esta necessidade e é muito comum ouvir os funcionários das bibliotecas pedindo silêncio. Assim, por um período, o silêncio é respeitado. Às vezes, ocorre que não há funcionários da biblioteca por perto e a conversa em nível mais alto atrapalha quem está estudando.

Enquanto o funcionário da biblioteca não chega, o barulho persiste e a pessoa que veio ler ou estudar perde a concentração e até mesmo pode ocasionar estresse.

Este trabalho tem, então, o objetivo de monitorar o nível sonoro, enviando uma mensagem de silêncio às pessoas que estejam provocando um nível maior de ruído. Essas pessoas recebem a mensagem solicitando silêncio sem que haja constrangimento das mesmas. Porém, não se pode deixar de levar em consideração que tal grupo possa ignorar a mensagem. Desta forma, este trabalho contempla a implementação de um dispositivo central, que informa em qual mesa tem-se nível sonoro alto. Este dispositivo central é monitorado por um funcionário responsável pelo setor.

O projeto basicamente constitui na implementação de dois dispositivos, a saber:

a. Dispositivo de mesa: equipamento que ficará em cada mesa monitorando o nível sonoro;
b. Dispositivo central: equipamento que receberá a informação vinda de cada dispositivo de mesa indicando a sua origem.

São desenvolvidos neste projeto, além dos hardwares descritos anteriormente, o software de reconhecimento de nível sonoro alto e a comunicação entre o dispositivo de mesa e o dispositivo central.

A aplicação deste projeto em ambientes como bibliotecas e enfermarias pode ser bastante proveitosa. Pois em uma ambiente como de uma biblioteca é muito comum haver grupos de estudantes discutindo assuntos em alto nível sonoro, o que atrapalha a concentração de quem está estudando neste ambiente. Estudar em um

ambiente com muito barulho pode acarretar muitos problemas, dentre os quais: a perda de concentração, a fadiga, a irritabilidade e o estresse.

Em muitos hospitais é possível constatar que suas enfermarias estão sempre cheias de pessoas conversando, o que atrapalha o repouso dos enfermos. No caso de uma UTI neonatal, por exemplo, onde há uma criança pré-matura que esteja ganhando peso ou terminando o desenvolvimento de seus órgãos, o barulho excessivo pode ocasionar uma redução da capacidade auditiva em tal criança.

O controle de nível sonoro é uma atividade importante em diversos ambientes. No entanto, no mercado não há mecanismo que faça tal controle. E este trabalho objetiva suprir esta demanda de forma inovadora.

1.1 Estrutura do Trabalho

Os capítulos deste trabalho serão distribuídos da seguinte forma:

No Capítulo 1 é feita a introdução do trabalho.

No Capítulo 2 são apresentados os conceitos das comunicações I^2C (*Inter-Integreted Circuit*) e USART (*Universal Synchronous Asynchronous Receiver Transmitter*) entre microcontroladores.

No Capítulo 3 é feita uma descrição do projeto proposto, mostrando tanto a topologia utilizada como os detalhes da construção do protótipo, do desenvolvimento do software e de seu funcionamento geral. Também são apresentados o ambiente de desenvolvimento e de simulação de programa que facilitam a realização do projeto.

No Capítulo 4 há a conclusão de projeto, apresentando os resultados obtidos, as dificuldades encontradas e as sugestões para trabalhos futuros.

2 COMUNICAÇÃO ENTRE MICROCONTROLADORES

Neste capítulo são abordados conceitos teóricos sobre as comunicações I^2C e USART.

2.1 Comunicação I^2C^{TM}

O I^2C (circuito inter-integrado) é um protocolo de comunicação serial desenvolvido pela Philips para realizar comunicação entre circuitos integrados (CI) com uma menor quantidade de pinos. O que era utilizado, inicialmente, para comunicação entre os componentes internos de sons de carro ou televisão, hoje, possui uma grande variedade de aplicações, inclusive para a comunicação de microcontroladores. Esse tipo de protocolo utiliza o conceito de *Master*/*Slave*, onde o *master* é responsável pelo controle da comunicação. (SOUZA, 2003).

A comunicação é feita em duas vias: Serial Clock - SCL e Serial Data - SDA. O Clock é a linha de relógio controlada pelo *master*, o que faz com que somente o *master* possua controle sobre a comunicação. O Data é a linha de dados bidirecional por onde trafegam os dados transmitidos tanto pelo *master* como pelo *slave*. (LOPES, 2001).

O *slave* deve esperar a descida do SCL para enviar dados, pois é apenas na descida do clock que a comunicação é feita. A utilização de resistores *pull-up* nos pinos SCL e SDA é obrigatória para garantir o estado ocioso quando não houver comunicação. Estes resistores podem variar de $1K\Omega$ a $10K\Omega$ de acordo com a capacitância do barramento, a quantidade de dispositivos conectados e o comprimento do fio condutor. (ZANCO, 2006).

Para efetuar a comunicação entre as vias SDA e SCL existem as condições de *start*, *stop*, *re-start* e *acknowledge* que são indicadas na Figura 2.1. (MICROCHIP, 2001).

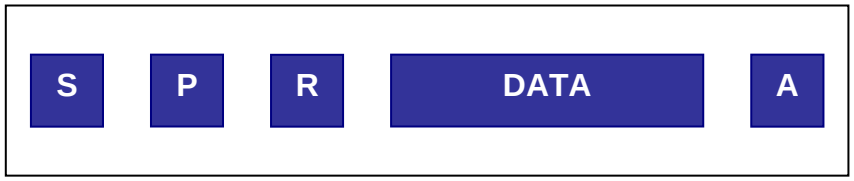


Figura 2.1 – Seqüência de envio das condições de *Start*, *Stop*, *Re-start* e *Acknowledge*.

Para o início da transmissão, deve ocorrer a condição de *start*, conforme ilustrado na Figura 2.2, onde o SDA vai para nível baixo (0) e o SCL fica em nível alto (1). (MICROCHIP, 2001).

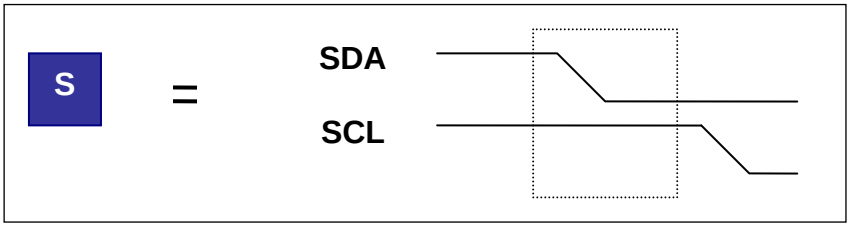


Figura 2.2 – Condição de *Start*.

Uma condição de *start* obriga a uma condição de *stop* para que não haja travamento do sistema. Essa condição *stop* ocorre quando o SDA vai para nível alto (1) e o SCL fica em nível alto (1). (MICROCHIP, 2001). Na Figura 2.3 é ilustrada como a condição de *stop* acontece.

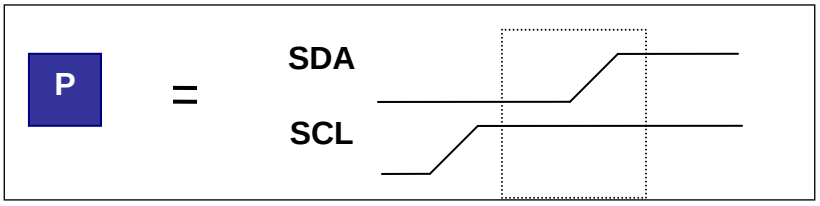


Figura 2.3 – Condição de *Stop*.

Já a condição *re-start* indica que o transmissor deseja enviar mais dados, porém não deseja realizá-lo na mesma linha. Pode-se dizer que o *re-start* é uma condição de *stop* seguido de uma condição de *start*, conforme apresentado na Figura 2.4. (MICROCHIP, 2001).

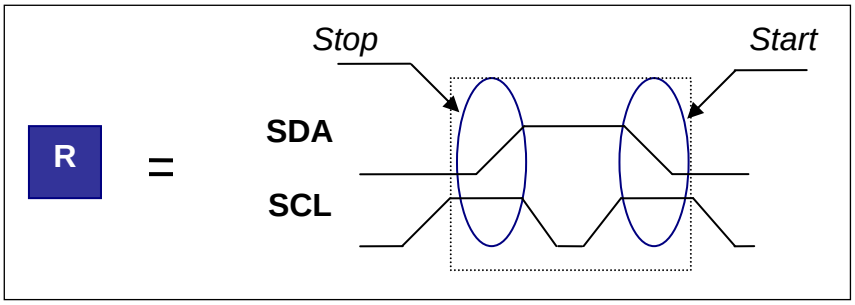


Figura 2.4 – Condição de *Re-Start*.

Na Figura 2.5 é ilustrada a transmissão de dados. Os bits da linda SDA são lidos quando o clock estiver em nível alto. (MICROCHIP, 2001).

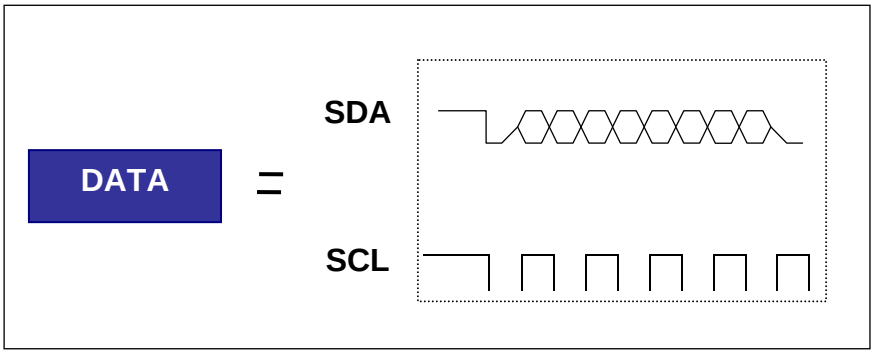


Figura 2.5 – Transmissão de dados.

Para finalizar a transmissão é necessário que ocorra a confirmação de recebimento da transmissão de dados. Essa confirmação acontece no 9º pulso de clock e quando SDA está em nível baixo. Caso o SDA esteja em nível alto, significa que não houve confirmação, que é chamado de não *acknowledge*. (MICROCHIP, 2001). Na Figura 2.6 é apresentado o exemplo de confirmação do *acknowledge*.

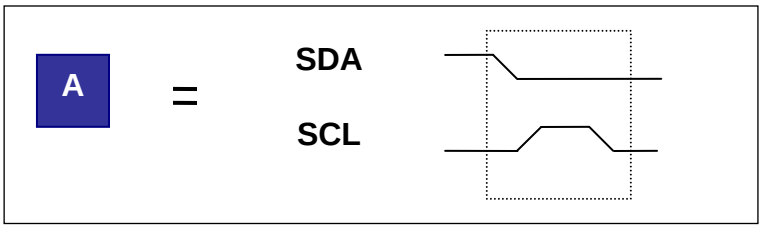


Figura 2.6 – Confirmação de *acknowledge*.

Na Figura 2.7 é apresentado um exemplo da situação do não *acknowledge*.

Figura 2.7 – Não há confirmação de *acknowledge*.

Dessa forma, quando um dado é recebido sem erros o receptor responde com um *acknowledge*. Caso contrário, o receptor responde com um não *acknowledge*.

2.1.1 Registradores envolvidos no I^2C

O protocolo I^2C opera com seis registradores de controle que são (MICROCHIP, 2001):

1. Registrador de controle 1 – SSPCON1
2. Registrador de controle 2 – SSPCON2
3. Registrador de estado – SSPSTAT
4. Registrador de endereço – SSPAD
5. Registrador de deslocamento – SSPSR. Não acessível diretamente
6. Buffer - SSPBUF

O protocolo I^2C permite a configuração de endereçamento com 7 ou 10 bits, sendo que, com 7 bits a quantidade de endereços é limitada a 128, já com 10 bits essa quantidade passa para 1024.

2.1.2 Configurando um *slave*

Para configurar um *slave* inicia-se configurando os pinos SCL e SDA como entradas colocando os bits do TRISC<3:4> em 1. Em seguida ajustam-se os bits de SSPCON<SSPM3:SSPM0> em 0110. Desta forma, o PIC16F877A fica configurado como *slave* com endereçamento de 7 bits. O ajuste do bit SSPSTAT<SMP> é necessário para habilitar um sistema interno de tratamento das bordas quando em freqüência de 400KHz. Para freqüência de 100KHz esse bit é setado. A comunicação é habilitada setando o bit SSPCON<SSPEN>. (SOUZA, 2003)

Assim que a comunicação é ativada o microcontrolador passa a aguardar a ocorrência de uma condição de *start*. Após a condição de *start* e a recepção dos primeiros 8 bits é feita a comparação do registrador SSPSR<7:1> com o valor existente no registrador SSPADD. O bit SSPSR<0> define se a comunicação é de leitura ou escrita, sendo que zero significa escrita e um significa leitura. (SOUZA, 2003)

2.1.2.1 Recepção no *slave*

Ao iniciar a recepção do *slave* a condição de *start* é reconhecida e o bit SSPSTAT<S> é setado indicando que o bit *start* foi detectado. Os 7 bits de endereço são enviados junto com o bit R/W=0. Automaticamente os bits SSPSTAT<R/W> e SSPSTAT<D/A> tornam-se zero indicando que se trata de uma escrita e que os bits recebidos representam um endereço. Os bits de endereço são deslocados para o registrador SSPSR<7:1> onde são comparados aos valores existentes no registrador SSPADD. Caso os valores não sejam os mesmos, a comunicação do lado *slave* é interrompida e fica esperando a ocorrência de um novo *start*. Porém se

houve coincidência, o valor do registrador SSPSR é carregado para o registrador SSPBUF e o bit SSPSTAT<BF> fica em um indicando que o valor foi transferido para o registrador SSPBUF, o pino SDA se transforma em nível baixo (saída) e fica aguardando o 9º pulso de clock. Assim que o SDA volta a ser entrada, o *flag* de interrupção PIR1<SSPIF> é colocado em nível alto permitindo que seja feita a leitura do SSPBUF, forçando novamente SSPSTAT<BF>=0. Na Figura 2.8 é ilustrada a seqüência para envio de endereço. (ZANCO, 2006)

Figura 2.8 – Seqüência de envio do endereço.

Logo após a confirmação do endereçamento, inicia-se a recepção de dados. O clock envia oito novos pulsos, o bit SSPTAT<D/A>=1 e o bit SSPSTAT<BF>=1 indicam novamente que o dado foi enviado para SSPBUF. O pino SDA impõe um nível baixo para responder *acknowledge* igual a zero no 9º pulso do clock e em seguida retorna ao nível alto, e mais uma vez o *flag* de interrupção PIR1<SSPIF> é colocado em nível alto indicando que o dado pode ser tratado. (SOUZA, 2003)

O envio de dados tem a mesma seqüência apresentada acima até que ocorra uma condição de *stop*. Ocorrendo esta condição, o bit SSPSTAT<P> é setado indicando que detectou com bit de *stop*. (SOUZA, 2003).

Caso ocorra uma condição de *re-start*, o *slave* fica esperando um novo byte de endereço e o bit R/W. (MICROCHIP, 2001) Segue abaixo, na Figura 2.9, a seqüência completa de envio de bits para a transmissão de dados.

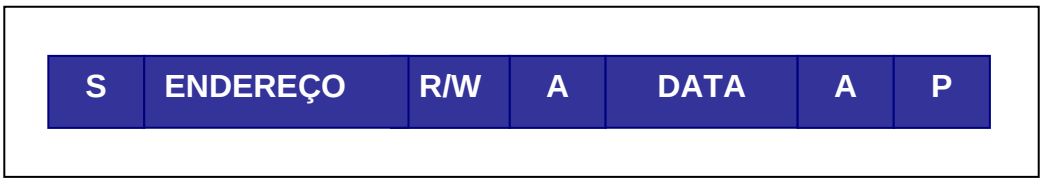


Figura 2.9 – Seqüência de envio de bits para transmissão de dados.

2.1.2.2 Transmissão no *slave*

No início da transmissão, o procedimento é o mesmo que o da recepção. Os pinos SCL e SDA são configurados como entrada, a condição de *start* é reconhecida e o bit SSPSTAT<S> é setado significando que o bit *start* foi detectado. Os 7 bits de endereço são enviados pelo *master* junto com o bit R/W=1 este último bit indica que se trata de uma leitura dos bits recebidos. E então, o bit SSPSTAT<R/W> passa a ser um, indicando que trata de uma leitura e o bit SSPSTAT<D/A> fica em zero para confirmar que um endereço foi recebido. Estes valores dos bits de endereço são deslocados para o registrador SSPSR<7:1> onde são comparados aos valores existentes no registrador SSPADD. Caso os valores não sejam os mesmos, a comunicação do lado *slave* é interrompida e fica esperando a ocorrência de um novo *start*. Se houver coincidência, o valor do registrador SSPSR é carregado para o registrador SSPBUF e o bit SSPSTAT<BF> fica em um, indicando que o valor foi transferido para o registrador SSPBUF. O pino SDA se transforma em nível baixo e fica aguardando o 9º pulso de clock. Em seguida o SDA volta a ser entrada e o *flag* de interrupção PIR1<SSPIF> é colocado em nível alto permitindo que seja feita a leitura do SSPBUF, forçando novamente SSPSTAT<BF>=0. (SOUZA, 2003)

O sistema automaticamente altera o bit SSPCON<CKP> para zero impondo nível baixo em SCL. Com isso, o *master* fica impedido de enviar sinal de clock enquanto o *slave* não estiver preparado para enviar dados. O *flag* de interrupção PIR1<SSPIF> é setado para que seja escrito o valor a ser transmitido em SSPBUF, o que faz o bit SSPSTAT<BF> ser igual a um. Então, o bit SSPCON<CKP> passa a nível alto e o SCL volta a ser entrada. Com a liberação do clock, o byte existente em SSPBUF é transmitido. Ao enviar o 9º bit em nível alto (*acknowledge*) significa que a transmissão encerrou, neste momento a condição de *stop* é reconhecida ao setar o bit SSPSTAT<P>, finalizando a comunicação. Porém, se o 9º bit estiver em nível baixo, o sistema desabilita o clock para que o BUFFER seja carregado e continue todo o procedimento de envio novamente. A transmissão de dados só se encerra quando o *acknowledge* for igual a um. (ZANCO, 2006)

2.1.3 Configurando um *master*

O *master* é responsável por todo o controle da comunicação. Ele gera e monitora a condição de *start*, a condição de *stop*, a condição *re-start*, o *acknowledge* e o clock. Para iniciar a comunicação, os pinos SCL e SDA devem estar configurados como entrada através dos bits TRISC<3:4> e para habilitar o *master* os bits SSPCON<SSPM3:SSPM0> devem ser configurados em 1000. Assim como no *slave*, o bit SSPSTAT<SMP> é responsável por habilitar o tratamento das bordas para a freqüência de 400KHz. A comunicação é habilitada ao setar o bit SSPCON<SSPEN>. (ZANCO, 2006)

2.1.3.1 Recepção no *master*

Conforme já mencionado anteriormente, os pinos SCL e SDA estão configurados como entrada. Ativa-se o bit SSPCON2<SEN> para que a condição de *start* seja gerada automaticamente. O endereço do hardware que se deseja comunicar é armazenado no registrador SSPBUF fazendo com que o *flag* de interrupção PIR1<SSPIF> seja setado e o bit SSPCON2<SEN> torne-se zero. Logo após a escrita do endereço no registrador SSPBUF são setados de forma automática os bits SSPSTAT<BF> e SSPSTAT<R/W>, este último é transmitido com os sete bits de endereço. Ao terminar a transmissão, o bit SSPSTAT<BF> passa a ser zero e o pino SDA aguarda um *acknowledge* no 9º pulso do clock. (SOUZA, 2003)

Quando o SSPCON2<RCEN> for igual 1 o *master* passa a trabalhar no modo de recepção. Um byte de dados é recebido pelo *master* nos oito pulsos de clock seguintes. O valor contido no bit SSPSTAT<D/A> indica o recebimento do byte de dado (1) ou endereço (0). O dado recebido é encaminhado para o registrador SSPBUF e o bit SSPSTAT<BF> é setado. Sempre que um byte é recebido um bit de confirmação (*acknowledge*) é transmitido. Para responder o *acknowledge*, o valor desejado deve ser colocado no bit SSPCON2<ACKDT>, se o valor for um o *slave* pára de transmitir ao *master*, caso contrário o *master* continua recebendo dados do

slave. O bit SSPCON2<ACKEN> gera condição de continuar a transmissão enquanto seu valor for igual a um. Após o *acknowledge,* os bits SSPCON<ACKEN> e PIR1<SSPIF> estão com zero e um, respectivamente, gerando outra interrupção. Caso o *acknowledge* seja igual a zero, inicia-se todo o processo para envio de um novo byte. Caso o *acknowledge* seja um, é gerada uma condição de *stop*. Após a conclusão da condição de *stop*, o bit SSPCON2<PEN> passa a ser zero e o bit PIR1<SSPIF> passa a ser um. (ZANCO, 2006)

2.1.3.2 Transmissão no *master*

Após a condição de *start*, já mencionada anteriormente, o *flag* da interrupção é ativado com os bits PIR1<SSPIF>=1 e SSPSTAT<SEN>=0. O endereço é escrito no registrador SSPBUF, onde o bit R/W é igual a zero. Conseqüentemente os bits SSPSTAT<BF> e SSPSTAT<R/W> tem valor um. Logo após a transmissão, o bit SSPSTAT<BF> passa a ser zero e o pino SDA aguarda a transmissão do 9º pulso de clock com o *acknowleged* igual à zero. O valor é então transferido para o bit SSPCON2<ACKSTAT>. (ZANCO, 2006)

O dado a ser transmitido é colocado no registrador SSPBUF com isso os bits SSPSTAT<BF> e SSPSTAT<R/W> passam a nível lógico alto. Essa transmissão ocorre através de oito pulsos do clock, levando em consideração que não há nenhum outro dispositivo solicitando pausa. Com o término da transmissão, o SSPSTAT<BF> fica em nível baixo e o SDA aguarda a confirmação do *acknowledge*. Este valor de *acknowledge* é passado para SSPCON2<ACKSTAT>. Neste momento uma nova interrupção pode ser gerada, pois o bit PIR1<SSPIF> está em nível lógico alto e o bit SSPSTAT<R/W> está em nível lógico baixo. Caso ocorra um novo envio de dado, este é armazenado em SSPBUF seguindo todo o procedimento citado acima. A transmissão só é interrompida quando o bit SSPCON2<PEN> for igual a um, gerando a condição de *stop*, ao final desta condição o bit SSPCON2<PEN> passa para o nível lógico baixo e o bit PIR1<SSPIF> fica em nível lógico alto. (SOUSA, 2003)

O protocolo I^2C é amplamente utilizado na comunicação de microcontroladores com memórias seriais. Contudo, o I^2C não foi utilizado neste

projeto devido à dificuldade de estabelecer a comunicação entre microcontroladores Como solução para esta comunicação, foi utilizada a comunicação USART.

2.2 Comunicação USART

A comunicação USART que significa transmissor receptor universal síncrono e assíncrono é uma comunicação serial que possui dois modos de funcionamento: síncrono e assíncrono. O modo assíncrono será abordado neste projeto. (SOUZA,2003)

2.2.1 Modo Assíncrono

No modo de comunicação USART assíncrona a transmissão é feita de maneira full-duplex possuindo duas vias de dados, uma para recepção e outra para transmissão de dados. Desta forma, o microcontrolador pode transmitir e receber dados simultaneamente. (ZANCO, 2006)

Neste tipo de comunicação, não há uma via de sincronismo (clock), a sincronização é feita pela via de dados através da taxa de transferência que também é chamada de *Baud Rate*. O *Baud Rate* define o período em que o bit permanece na linha de transmissão, ou seja, a quantidade de bits que pode ser transmitida por segundo (bps). No PIC16F877A há um gerador de *baud rate* (BRG) cujo valor para a transmissão assíncrona é calculado a partir da Equação 2.1 e depende do valor da freqüência de clock (Fosc). (ZANCO, 2006)

$$SPBRG = \frac{Fosc}{MxBaudRate} - 1 \qquad (2.1)$$

Se o *baud rate* for de baixa velocidade o valor da constante M é igual a 64, e se for de alta velocidade o valor da constante M é igual a 16. Os valores de SPBRG são mostrados na Tabela 2.1 para *baud rate* de baixa velocidade e com a freqüência de clock igual 4MHz.(MICROCHIP,2007)

Tabela 2.1 – Transmissão Assíncrona para Fosc=4MHz de baixa velocidade

Baud Rate Nominal (bps)	*Baud Rate* Real (bps)	% ERRO	SPBRG (decimal)
110	-	-	-
300	300,48	0,16	207
1200	1201,92	0,16	51
2400	2403,85	0,16	25
4800	4807,69	0,16	12
9600	8928,57	-6,99	6
19200	20833,33	8,51	2
38400	31250,00	-18,62	1
57600	62500,00	8,51	0
115200	-	-	-
Máxima	62500,00	-	0
Mínima	244,14	-	255

O valor do *baud rate* deve ser igual tanto na transmissão como na recepção para que haja o sincronismo. Com o sincronismo padronizado, a transmissão de dados é iniciada com o envio do *start* bit, que é a transição do nível lógico um para o nível lógico zero durando um período de bit. Logo em seguida, o transmissor envia os bits de dados, que pode ser de 4, 5, 6, 7 ou 8 bits, iniciando pelo bit menos significativo. A transmissão é finalizada com o envio do *stop* bit que possui nível lógico igual a um. (ZANCO,2006). Na Figura 2.10 é ilustrado o envio do *start* bit, dados e o *stop* bit.

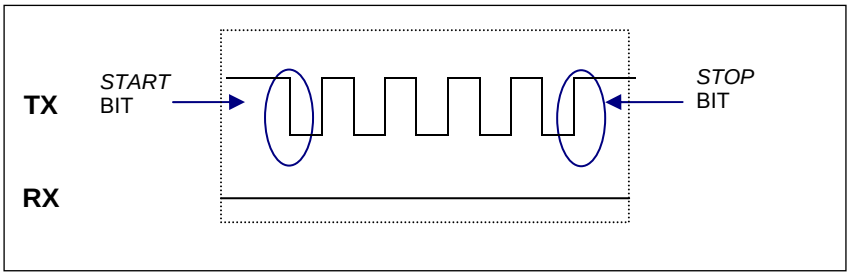


Figura 2.10 – Envio do *start* bit, dados e *stop* bit

As vias de transmissão e recepção devem estar com o mesmo formato de dados e em nível lógico alto.

2.2.1.1 Registradores envolvidos na comunicação USART

Os registradores envolvidos na comunicação USART assíncrona são (MICROCHIP, 2001):

1. Registrador de transmissão – TXSTA
2. Registrador de recepção – RCSTA
3. Buffer de transmissão – TXREG
4. Buffer de recepção – RCREG
5. Registrador de deslocamento de transmissão – TSR. Não acessível diretamente
6. Registrador de deslocamento de recepção – RSR. Não acessível diretamente

2.2.1.2 Configurando a transmissão

Para iniciar a transmissão de dados na comunicação USART assíncrona o primeiro passo a ser tomado é a configuração dos pinos RC6 (TX) e RC7(RX) como

entrada. O bit RCSTA<SPEN> habilita a comunicação USART quando setado e o bit TXSTA<TXEN> habita a transmissão de dados.(SOUZA, 2005)

2.2.1.3 Transmissão

Com a habilitação da transmissão, todo dado carregado no registrador TXREG é transferido, automaticamente, ao registrador TSR, de onde o dado é enviado pela porta serial. Assim, TXREG fica vazio e liberado para nova transmissão. Esse buffer de saída é duplo, podendo haver um byte em TXREG e outro em TSR. (SOUZA, 2005)

Toda vez que a transmissão é completada o bit TXSTA <TRMT> é setado e uma nova transmissão pode ser iniciada. (ZANCO, 2006)

2.2.1.4 Configurando a recepção

A configuração da recepção de dados inicia da mesma forma que a configuração da transmissão. Os pinos RC6 (TX) e RC7(RX) são configurados como entrada. O bit RCSTA<SPEN> habilita a comunicação USART quando setado e o bit RCSTA <CREN> é setado para habilitar a recepção de dados.(SOUZA, 2005).

2.2.1.5 Recepção

O dado recebido é armazenado no registrador interno RSR. Quando esse registrador esta cheio, o dado é enviado para o registrador RCREG, que possui dois níveis de pilha do tipo *FIFO* (o primeiro a entrar é o primeiro a sair). (SOUZA, 2005)

Dessa forma, é possível armazenar no RCREG dois bytes e começar a receber um terceiro em RSR. Toda vez que um byte armazenado em RSR é transferido para RCREG o bit PIR1<RCIF> é setado. Este bit é limpo

automaticamente quando o registrador RCREG estiver vazio, porém, se chegar um *stop* bit do terceiro byte antes do RCREG ter sido lido, este byte é perdido e o bit RSTA<OERR> é setado, indicando que houve um transbordo. (ZANCO, 2005)

Assim, que o bit PIR1<RCIF> for limpo, automaticamente, significa que a recepção foi concluída.

3 IMPLEMENTAÇÃO DO PROJETO

Neste capítulo é apresentado o desenvolvimento da parte de hardware e de software para a implementação da monitoração de nível sonoro. O dispositivo da mesa é responsável pela detecção do nível sonoro, pela solicitação de silêncio à mesa e pelo envio de mensagem ao dispositivo central, informando que o nível sonoro foi excedido. O outro dispositivo é o dispositivo central, responsável por receber as mensagens do dispositivo de mesa e informar ao funcionário responsável que tal mesa está com nível sonoro alto. Na Figura 3.1 é ilustrada a topologia do projeto tendo um dispositivo central e dois dispositivos de mesa.

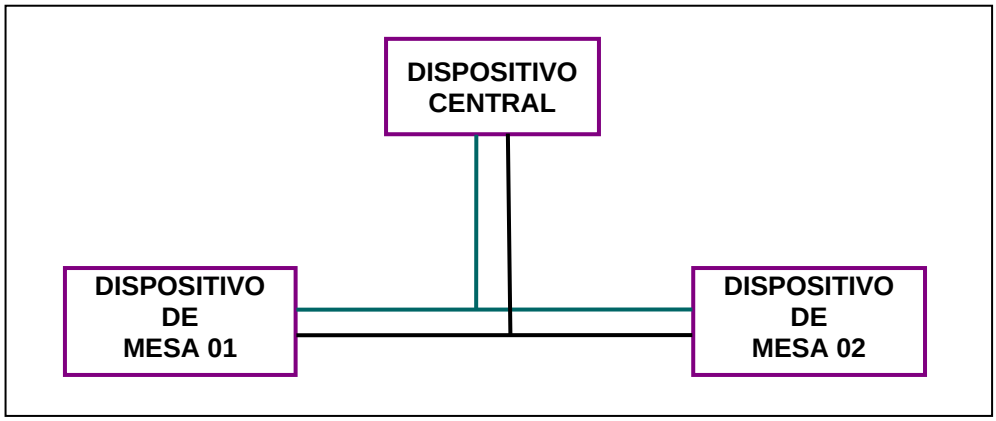


Figura 3.1 – Topologia do projeto

Para efeito de demonstração, são implementados dois dispositivos de mesa ligados, por cabo, ao dispositivo central. O dispositivo central pode suportar até 20 dispositivos de mesa ligados a ele.

3.1 Dificuldades com o I^2C

As configurações dos dispositivos *master* e *slave* foram feitas conforme descrito no Capítulo 2. Porém a comunicação não funcionou. Utilizando um osciloscópio, foi possível observar que o *master* gera a condição de *start*, os pulsos de clock e o endereço do *slave*. Este endereço foi o 10100010, conforme é apresentado na Figura 3.2. Aparentemente o *slave* reconhece o endereço como sendo o dele, pois responde com o *acknowledge* igual a zero, mas o *master* envia uma seqüência de dados desconhecida. Com isso, a comunicação se perde, a cada tentativa de comunicação o mesmo fato se repete.

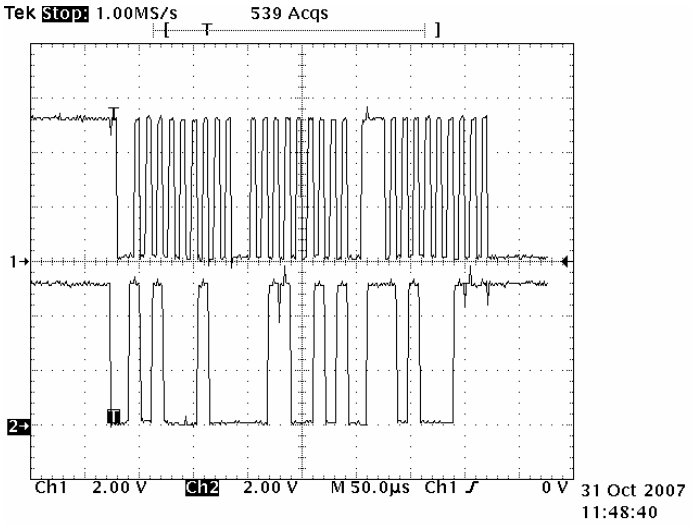


Figura 3.2 – Condição de *start* e envio de endereço no barramento I^2C feitos pelo *master*

Em outros momentos, com outras tentativas de configuração em que o PIC 16F877A foi programado apenas com a comunicação I^2C, o *master* fica enviando o endereço repetidamente sem a confirmação do *acknowledge* pelo *slave*.

Devido a todas essas situações o protocolo I^2C foi inviabilizado neste projeto. Como solução para a comunicação entre os dispositivos foi aplicada a comunicação USART.

3.2 Funcionamento do hardware

Para a realização deste projeto foram utilizados sensores sonoros, microcontroladores PIC16F877A e display's. O dispositivo de mesa é responsável por detectar o nível sonoro do ambiente e alertar o usuário e o dispositivo central. O dispositivo central é responsável por receber as mensagens dos dispositivo de mesa e informar ao funcionário responsável que determinada mesa está com nível sonoro alto. Na Figura 3.3 é ilustrado o diagrama do projeto.

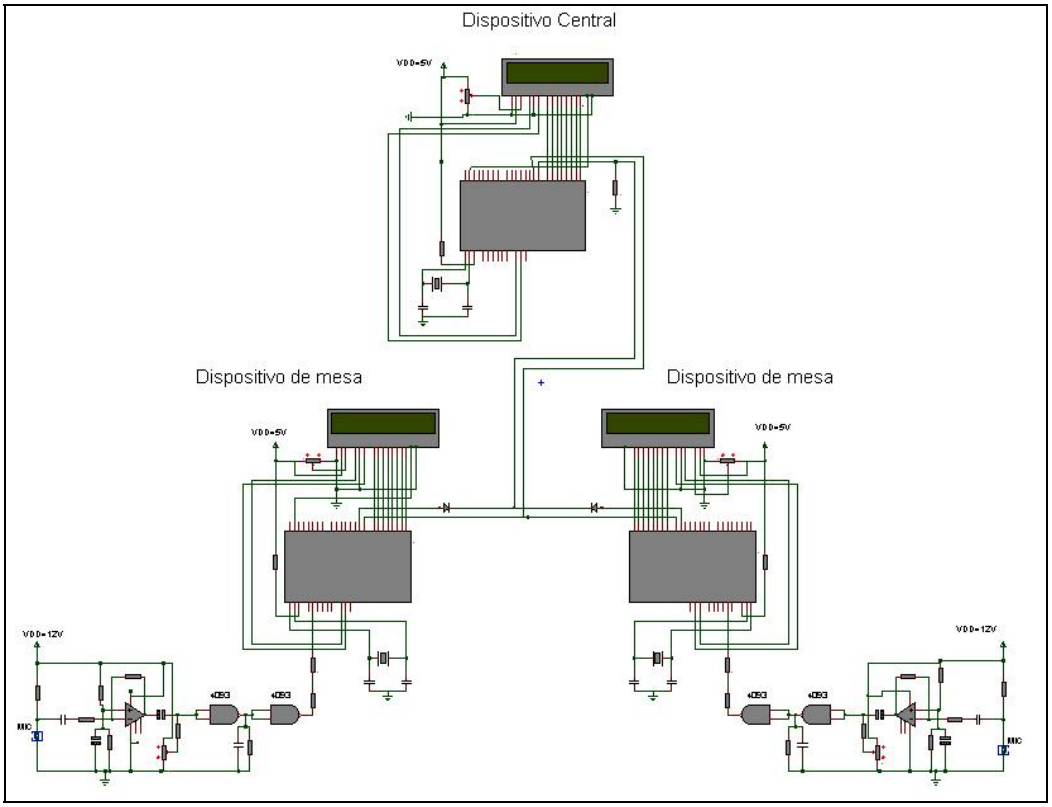


Figura 3.3 – Diagarama do projeto

3.2.1 Dispositivo de mesa

Cada dispositivo de mesa possui um sensor sonoro que capta o som e o envia ao microcontrolador, que reconhece o nível sonoro alto e, o microcontrolador

por sua vez, envia uma solicitação de silêncio através do display implementado em cada dispositivo de mesa, de forma a alertar o usuário. Na segunda ocorrência de uma detecção de nível sonoro alto pelo dispositivo de mesa, este envia uma mensagem ao dispositivo central.

Na Figura 3.4 é apresentado o esquema elétrico do dispositivo de mesa onde podem ser vistos o display, o microcontrolador e o circuito do sensor sonoro. O microcontrolador utilizado foi o PIC16F877A, por ser um componente comum e pela facilidade de aquisição.

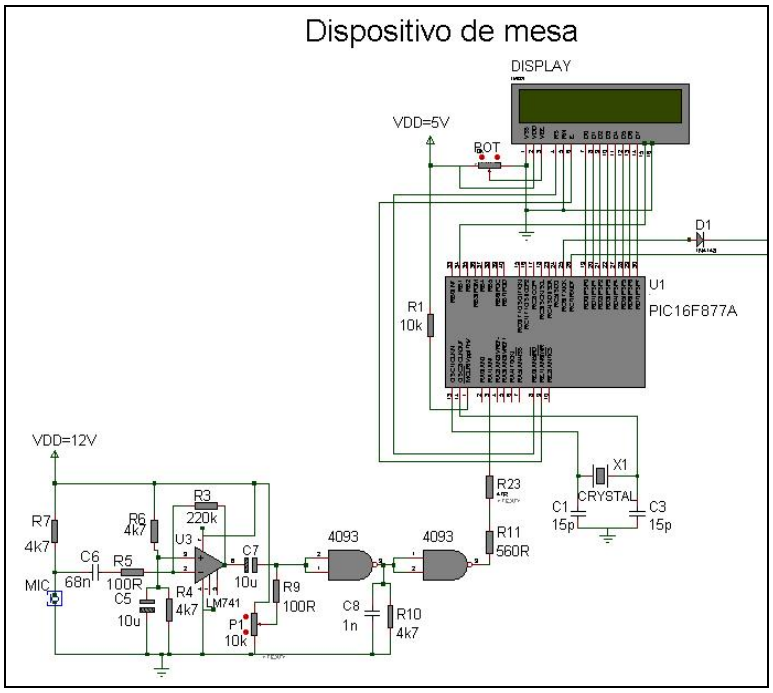


Figura 3.4 – Diagrama do dispositivo de mesa

O circuito do sensor sonoro é alimentado com uma tensão de 12V. Este circuito é constituído de um microfone de eletreto alimentado pelo resistor R7. O sinal de áudio captado pelo microfone é aplicado na entrada inversora do amplificador operacional LM741 através do capacitor C6. O divisor de tensão composto por R6 e R4 gera uma tensão equivalente à metade da tensão de alimentação, ou seja, 6V para polarizar a entrada não inversora do amplificador operacional. O capacitor C5 estabiliza o valor dessa tensão. Depois que o sinal de áudio é amplificado com o ganho de R3/R5, este é disponibilizado no capacitor C7 e

em seguida, aplicado em duas portas *NAND* para o que se torne um sinal digital. A saída da segunda porta *NAND*, quando em nível alto, apresenta uma tensão de 12V. (DETETIVE, 2001) Como a tensão máxima na entrada do microcontrolador é de 5V, são utilizados dois resistores, R11 e R23, para reduzir a tensão. Na Figura 3.5 é ilustrado o dispositivo de mesa que utilizado para implementação do projeto.

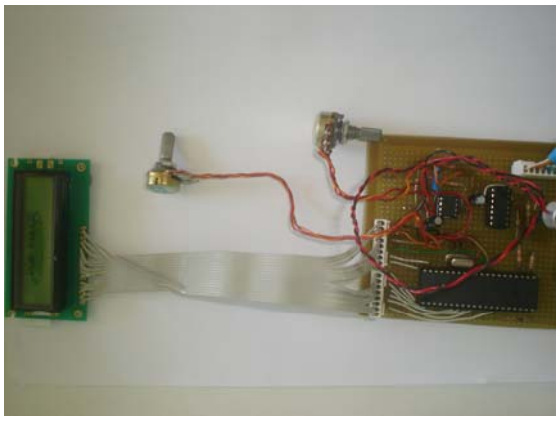


Figura 3.5 – Dispositivo de mesa

O pino 03 do microcontrolador PIC16F877A corresponde à entrada digital PORTA RA1, com a chegada de nível alto neste pino o microcontrolador envia a seguinte mensagem ao display: ”Por favor, Silêncio!!”. Caso o sinal de entrada no microcontrolador esteja em nível baixo, o PIC16F877A fica aguardando o nível alto. Na Figura 3.6 é apresentada a mensagem que aparece no display quando o dispositivo de mesa detecta nível de ruído elevado.

Figura 3.6 – Mensagem aparecendo no display: Por Favor Silencio!!

No caso de recorrência de nível alto na entrada do microcontrolador do dispositivo de mesa é, então, enviado um sinal para o dispositivo central através do pino 25 (PORTC RC6), indicando que há nível sonoro alto em sua mesa.

3.2.2 Dispositivo central

O dispositivo central é composto por um display e um microcontrolador PIC16F877A. O sinal que chega ao dispositivo central é recebido por seu microcontrolador pelo pino 26 (PORTC RC7). Na Figura 3.7 é ilustrado o diagrama do dispositivo central.

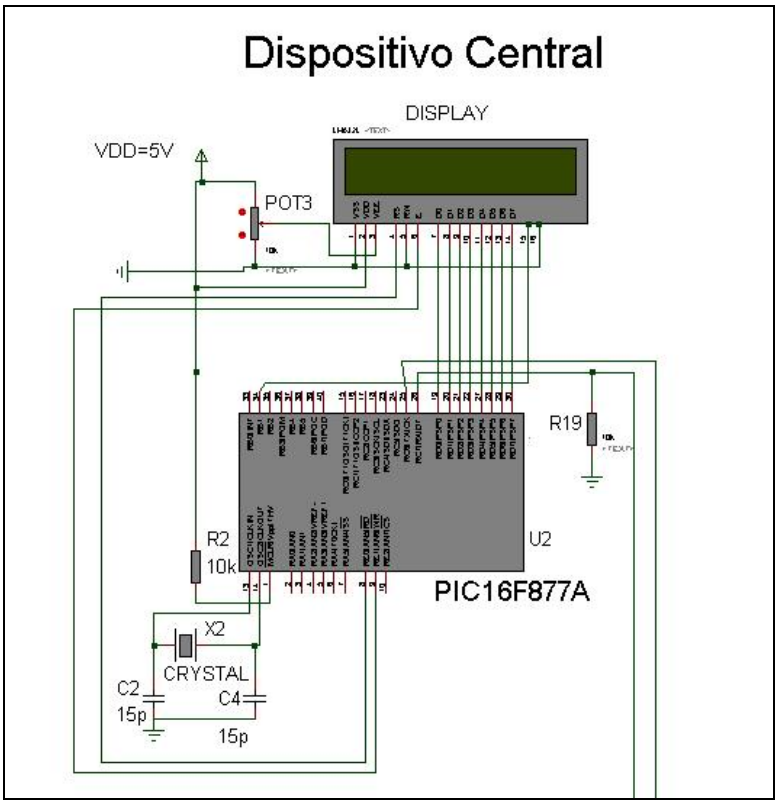


Figura 3.7 – Diagrama do circuito da central

O PIC16F877A do dispositivo central identifica qual dispositivo de mesa originou o sinal, e com isso envia uma mensagem ao display indicando a mesa onde

o nível foi excedido. Na Figura 3.8 é ilustrado o dispositivo central utilizado para implementação do projeto.

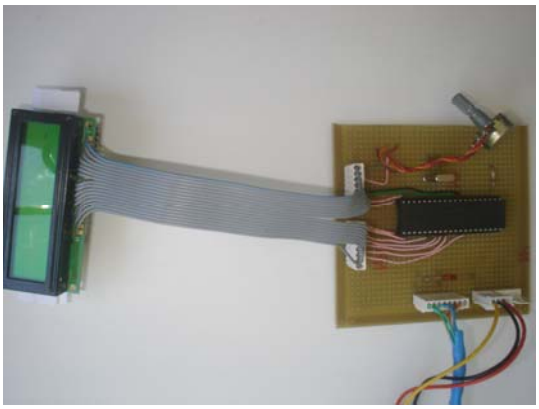


Figura 3.8 – Dispositivo Central

Na Figura 3.9 são apresentadas as mensagens escritas no display do dispositivo central quando este recebe um sinal do dispositivo de mesa 1 (A) ou do dispositivo de mesa 2 (B).

(A) Mesa: 1 (B) Mesa: 2

Figura 3.9 – Dispositivo central a origem do ruído.

3.2.3 Simulação

Para executar a simulação o projeto foi instalado em um ambiente com baixo nível de ruído. Os dois dispositivos de mesa ficaram a 5 metros de distância do dispositivo central. Os dispositivos de mesa estão devidamente identificados como Dispositivo mesa 1 e Dispositivo mesa 2 conforme estão ilustrados na Figura 3.10.

(A) Mesa 1 (B) Mesa 2

Figura 3.10 – Identificação dos dispositivos de mesa.

Nos dispositivos de mesa é possível regular o contraste do display e alterar a sensibilidade dos sensores através de seus potenciômetros. Esses potenciômetros estão disponibilizados conforme é apresentado na Figura 3.11e são chamado de reguladores.

Figura 3.11 – Reguladores de contraste do display e sensibilidade do sensor

Quando ligados, os dispositivos de mesa captam o som ambiente. O ajuste de sensibilidade faz com que ele detecte sons mais ou menos intensos, ou seja, quanto mais sensível ele estiver mais facilmente detecta o som no ambiente e vice-versa. O dispositivo central envia, periodicamente, mensagens aos dispositivos de mesa. Essas mensagens são interpretadas pelos dispositivos de mesa como uma pergunta sobre o estado do nível sonoro no ambiente naquele momento. Essas mensagens possuem o formato do código ASCII, sendo a letra U para o dispositivo mesa 1; e a letra P para o dispositivo mesa 2. O dispositivo central está devidamente identificado conforme é ilustrada na Figura 3.12.

Figura 3.12 – Identificação do dispositivo central

Com o auxílio de um osciloscópio é possível observar a troca de mensagens entre os dispositivos de mesa e o dispositivo central, quando da recorrência de nível alto. Na Figura 3.13 é ilustrada a resposta do dispositivo de mesa 1 que responde à pergunta do dispositivo central com o envio da letra B do código ASCII. A linha superior apresenta as perguntas do dispositivo central aos dispositivos de mesa, já a linha inferior apresenta a resposta do dispositivo mesa 1 ao dispositivo central.

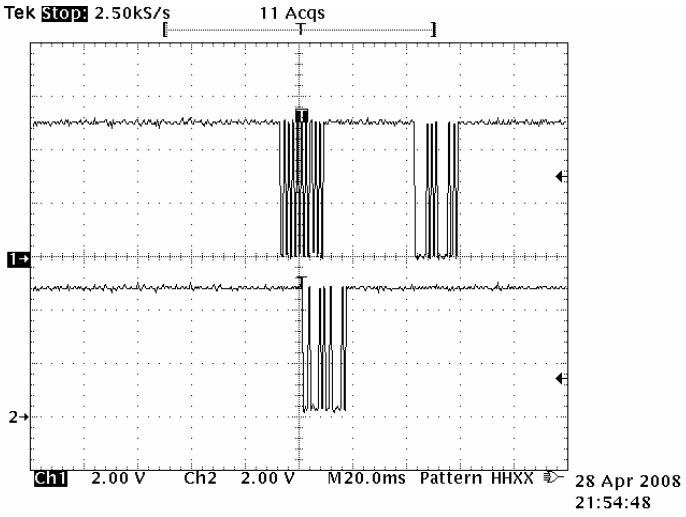


Figura 3.13 – Troca de mensagens entre o dispositivo central e o dispositivo mesa 1

Na Figura 3.14 é ilustrada a comunicação entre o dispositivo central e o dispositivo mesa 2, quer responde com o envio da letra C do código ASCII. A linha superior apresenta as perguntas do dispositivo central e a linha inferior a resposta do dispositivo mesa 2 ao dispositivo central.

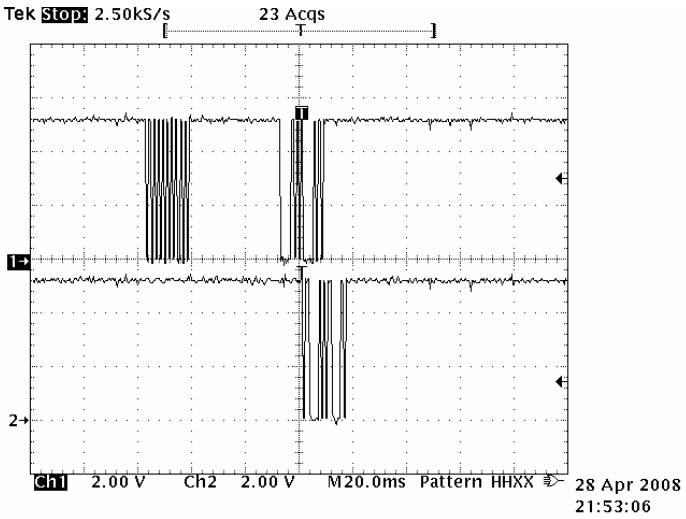


Figura 3.14 – Troca de mensagens entre o dispositivo central e o dispositivo mesa 2

3.3 Funcionamento do Software

Para a programação as ferramentas utilizadas são a linguagem Assembler e a comunicação USART. O ambiente de desenvolvimento do microcontrolador é o MPLAB IDE® que é um software com editor de texto para a linguagem assembler. O simulador utilizado para verificar erros de programação não detectados pelo ambiente MPLAB é o Pic Simulator IDE.

3.3.1 MPLAB IDE® - Interface de desenvolvimento

Para a gravação do programa-fonte no microcontrolador é utilizado o ambiente de desenvolvimento chamado de MPLAB IDE®, versão 7.60 que foi criado pela Microchip. Este ambiente de desenvolvimento permite criar programa em assembly, efetuar a simulação e o debug do programa e gravar programas no microcontrolador. (ZANCO, 2005) Na Figura 3.15 é ilustrado o ambiente do MPLAB.

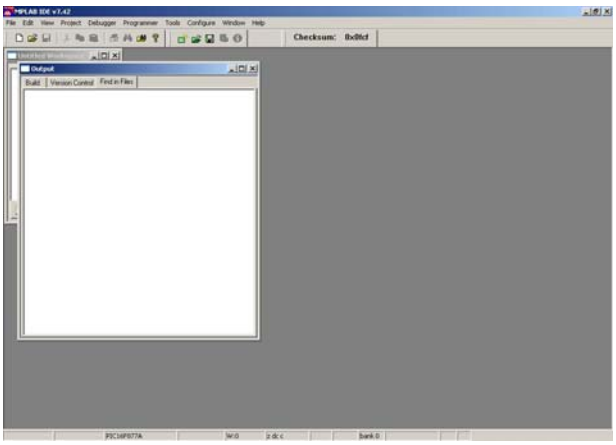


Figura 3.15 – Ambiente MPLAB

O MPLAB possui uma linguagem de programação relativamente fácil de utilizar e é, também, um compilador que converte códigos fonte em instruções de

máquina. Ele possui um conjunto de ferramentas para as mais diversas aplicações, e apresenta um ambiente rico para a visualização de dados devido à sua capacidade gráfica. Além disso, é um ambiente de desenvolvimento totalmente gratuito. (MICROCHIP, 2000)

3.3.2 PIC Simulator IDE

O PIC Simulator IDE é uma ferramenta extremamente útil para verificar possíveis erros em um programa. Esse aplicativo simula a execução do programa, mostrando como é realizado cada procedimento dentro do microcontrolador. Na Figura 3.16 é possível visualizar a tela do PIC Simulator, onde são apresentadas várias informações sobre a execução do programa, tais como as instruções, o registrador W, os registradores de funções especiais e os registradores de propósito geral.

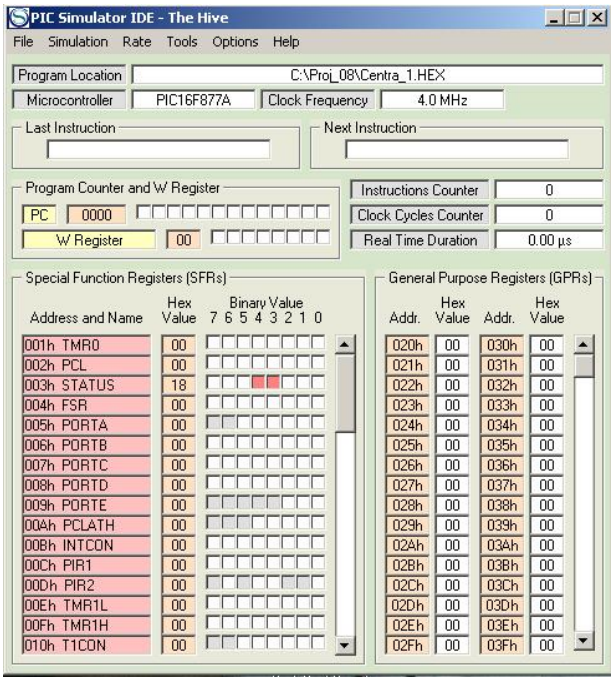


Figura 3.16 – PIC Simulator IDE

O PIC Simulator IDE possui suporte a vários microcontroladores, dentre eles o PIC16F877A. (OSHON SOFTWARE PROJECT,2007). Há muitas ferramentas importantes como a visualização dos pinos quando acionados, como é apresentado na Figura 3.17.

Figura 3.17 – Visualização dos pinos do microcontrolador PIC16F877A

Na Figura 3.18 é apresentada a visualização do display fornecida pelo PIC Simulator.

Figura 3.18 – Módulo display

Na Figura 3.19 é ilustrada a visualização da memória de programa, o que facilita a identificação de erros. (OSHON SOFTWARE PROJECT, 2007).

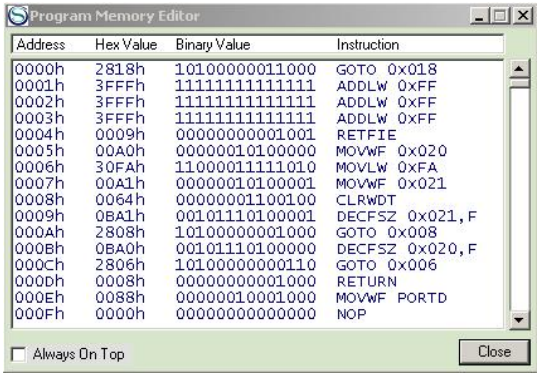


Figura 3.19 – Visualização da memória de programa

Esse aplicativo pode ser encontrado na internet gratuitamente. (OSHON SOFTWARE PROJECT, 2007)

3.3.3 Software do dispositivo de mesa

Este tópico descreve a implementação do programa utilizado no microcontrolador do dispositivo de mesa. Na Figura 3.20 é ilustrado seu fluxograma.

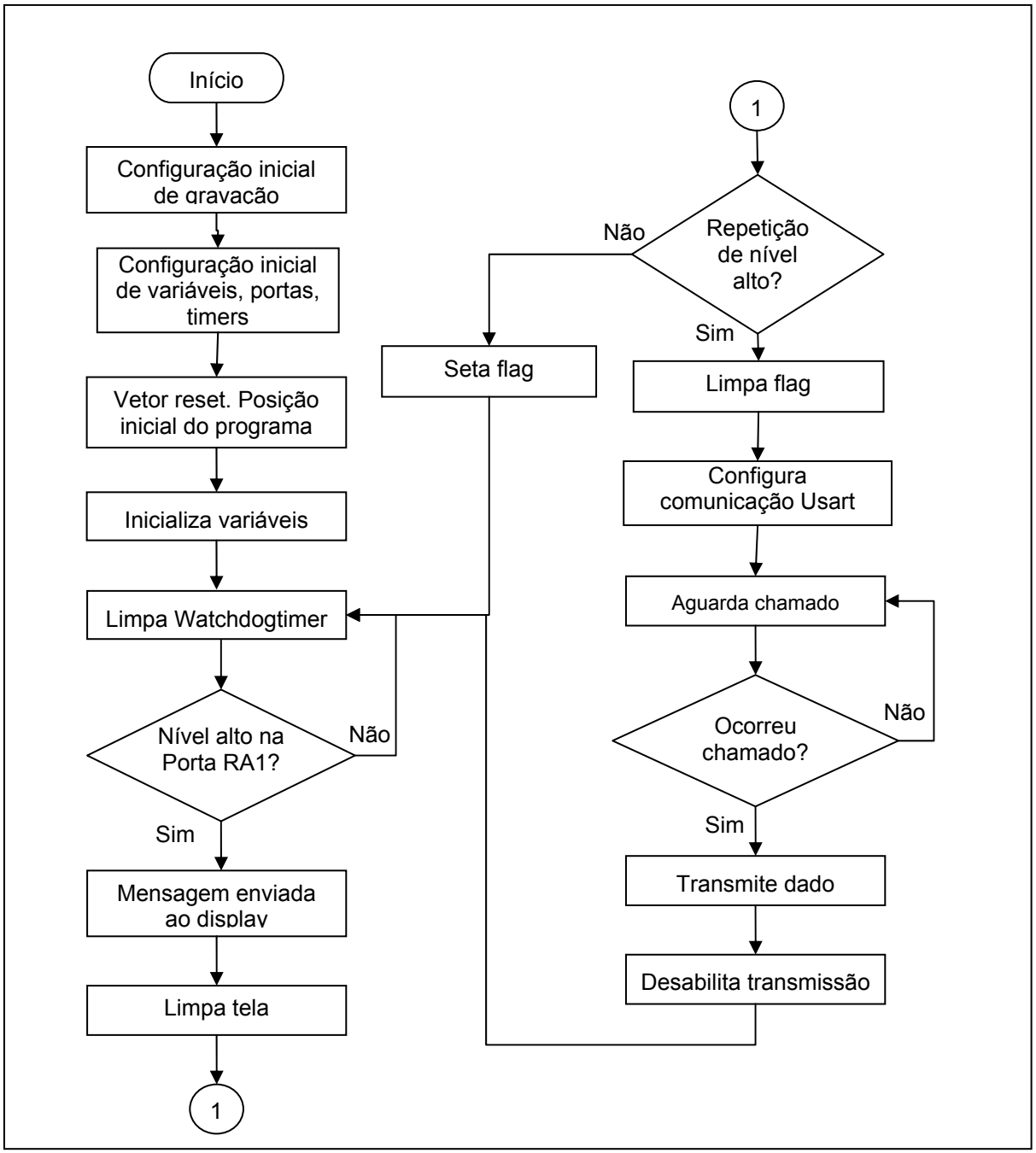


Figura 3.20 – Fluxograma da programação do dispositivo mesa

O programa tem início com a configuração inicial de gravação do PIC16F877A e com as configurações de variáveis, portas e timers. O programa é posicionado em seu endereço de reset onde inicializa as variáveis e em seguida entra na rotina para verificar o sinal no pino RA1. Esta rotina fica aguardando nível alto para mandar mensagem ao display, se houver repetição deste nível alto o programa configura a comunicação USART e aguarda o chamado do dispositivo

central. Na ocorrência do chamado o programa responde com a transmissão do dado correspondente ao número do dispositivo de mesa. Em seguida a transmissão é desabilitada e o programa volta a monitorar o pino RA1.

Ao iniciar a programação do dispositivo de mesa é necessário configurar a gravação do PIC16F877A logo nas primeiras linhas. Essa configuração inicia-se com a diretiva __CONFIG onde são configurados os seguintes itens (MICROCHIP,2001):

- Proteção da memória de programa – desprotegida
- Memória EEPROM protegida contra leitura
- DEBUG desabilitado
- programação em baixa tensão desabilitada
- *Watchdog Timer* desativado
- Oscilador de cristal (XT) com $4MHz$ de freqüência.

As linhas de comando a seguir apresentam como é feita a configuração para a gravação do PIC16F877A.

```
      __CONFIG _CP_OFF & _CPD_OFF & _DEBUG_OFF & _LVP_OFF &
_WRT_OFF & _BODEN_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC
```

Além disso, é necessário incluir o arquivo P16F877A.INC para que os nomes das variáveis definidas pela Microchip possam ser usados sem a necessidade de reescrevê-las. Essa inclusão é feita utilizando o # INCLUDE <P16F877A.INC> no software. Para facilitar a comutação entre BANK0 e BANK1, eles são definidos através do registrador STATUS<RP1:RP0>, quando o bit STATUS<RP0> está em nível alto seleciona BANK1 caso esteja em nível baixo o BANK0 é selecionado. (SOUZA, 2003). A seleção dos bancos é apresentada abaixo.

```
#DEFINE     BANK1       BSF   STATUS,RP0
#DEFINE     BANK0       BCF   STATUS,RP0
```

Para definir a porta por onde o sinal do sensor entra no PIC16F877A, basta dar um nome à entrada para facilitar as chamadas da PORTA, o nome escolhido é SINAL_SENSOR que está diretamente relacionado à PORTA RA1.

A PORTD<D7:D0> é configurada como saída de dados para o display. E as PORTE RE0 e PORTE RE1 são configuradas como RS (comando ou dado) e *Enable* (leitura ou escrita no display), respectivamente. Para acender o led do *backlight* do display é definida como saída a PORTB RB1. (SOUZA, 2003). Todas as definições para a utilização de display são realizadas da seguinte forma:

```
#DEFINE     DISPLAY          PORTD
#DEFINE     RS               PORTE,0
#DEFINE     ENABLE           PORTE,1
#DEFINE     BACK             PORTB,1
```

Ainda é preciso definir constantes que têm seu uso para definição de *flags* e para a construção de delay utilizando os contadores TEMPO0 e TEMPO1. Essas constantes são definidas a partir da diretiva CBLOCK que aloca endereços na memória de programa. A definição dessas constantes é feita conforme segue.

```
CBLOCK      0X20
            TEMPO1
            TEMPO0
            FLAG
ENDC
```

A partir das definições de constantes é possível criar uma rotina para contagem do tempo. A rotina DELAY_MS está apresentada a seguir possui um atraso de 1ms que pode ser multiplicado por um valor passado em work (w). (SOUZA, 2003)

```
DELAY_MS
        MOVWF       TEMPO1
        MOVLW       .256
        MOVWF       TEMPO0
        CLRWDT
        DECFSZ      TEMPO0,F
        GOTO $-2
        DECFSZ      TEMPO1,F
        GOTO $-6
RETURN
```

De posse da rotina DELAY_MS, é criada uma rotina para a escrita de caracteres no display. Nesta rotina, chamada de ESCREVE, é utilizada não só a rotina de atraso, mas também o ENABLE e a constante DISPLAY. Segue a implementação da rotina.

```
ESCREVE
        MOVWF       DISPLAY
        NOP
        BSF         ENABLE
        MOVLW       .1
        CALL        DELAY_MS
        GOTO        $+1
        BCF         ENABLE
        MOVLW       .1
        CALL        DELAY_MS
RETURN
```

Dois *flags* foram criados para facilitar o controle da transmissão de dados. O *flag* ACESO é usado para iniciar a transmissão de dados. O *flag* TESTE é utilizado para iniciar a recepção do chamado. A definição destes *flags* ocorre da seguinte forma.

```
#DEFINE     ACESO       FLAG,0
#DEFINE     TESTE       FLAG,1
```

Após definir todas as entradas e saídas, a diretiva ORG é usada para indicar o vetor de reset do programa. Este vetor reset está localizado no endereço 0x00. (MICROCHIP, 2001). A diretiva ORG aponta para o endereço 0x00 e logo em seguida o programa é desviado para a rotina CONF.

Porém, há um erro no arquivo *linker scripts*, do 16f877a.lkr, conforme é ilustrado na Figura 3.21. Este arquivo é responsável pela divisão da memória de programa.

```
MPLINK 4.11, Linker

Copyright (c) 2006 Microchip Technology Inc.

Error - section '.org_0' can not fit the absolute section. Section '.org_0'

start=0x00000000, length=0x0000021a

Errors    : 1
```

Figura 3.21 – Erro ao iniciar o programa na posição 0x00 da memória de programa

Para solucionar o problema alterou-se o arquivo *linker script*, deixando as primeiras linhas do codepage e do section em comentários, e acrescentando os endereços de 0x00 a 0x04 na segunda linha do codepage (page0). As modificações estão apresentadas na Figura 3.22. Interessante mencionar que no endereço da memória de programa 0x04 fica o vetor de interrupção.

```
//CODEPAGE   NAME=vectors  START=0x0000    END=0x0004
PROTECTED
CODEPAGE   NAME=page0    START=0x0000     END=0x07FF
.
.
.
//SECTION    NAME=STARTUP  ROM=vectors    // Reset and interrupt vectors
```

Figura 3.22 – Correção no arquivo 16f877a.lkr

Apesar dessas alterações continuou ocorrendo erro. Então, a solução final foi retirar esse arquivo que não alterou o funcionamento deste programa.

Depois de solucionado o problema no vetor de reset, o programa vai para a função chamada CONF onde terão todas as configurações dos registradores. Todas as PORT's são limpas e os registradores TRISA, TRISB e TRISC são configurados como entrada, exceto TRISB RB1, TRISC RC6 e RC7 que são configurados como saída. O registrador TRISD é, também, configurado como saída, pois os dados que são enviados para o display saem dele. No registrador PORTE, são usadas apenas as portas RE0 e RE1 configuradas como saída, nelas estão o RS e o *Enable* do display. (SOUZA, 2003). Como não é usada a interrupção de Timer0, o registrador INTCON tem todas as suas interrupções desabilitadas. (MICROCHIP, 2000). Os *flags* TESTE e ACESO são limpos no início do programa. Essas configurações são realizadas na rotina CONF que está escrita logo a seguir.

```
CONF
        CLRF        PORTA
        CLRF        PORTB
        CLRF        PORTC
        CLRF        PORTD
        CLRF        PORTE
        BANK1
        MOVLW       B'11111111'
        MOVWF       TRISA
        MOVLW       B'11111001'
        MOVWF       TRISB
        CLRF        PORTB
        MOVLW       B'10000000'
        MOVWF       TRISC
        MOVLW       B'00000000'
        MOVWF       TRISD
        MOVLW       B'00000100'
        MOVWF       TRISE
        MOVLW       B'11011111'
        MOVWF       OPTION_REG
        MOVLW       B'00000000'
        MOVWF       INTCON
        MOVLW       B'00000111'
        MOVWF       ADCON1
        BCF         TESTE
        BCF         ACESO
```

Após a configuração o programa segue para a rotina principal, MAIN, que encaminha para a rotina de LEITURA que, por sua vez, verifica o pino RA1 do PIC16F877A e retorna a rotina principal. Caso a rotina LEITURA tenha verificado nível alto na entrada do microcontrolador o programa é desviado para a rotina TRATA_SINAL, para que se escreva no display. Mais uma vez o programa volta a rotina principal e apaga a mensagem no display através da rotina LIMPA_TELA. Ainda na rotina principal é feita a verificação do *flag* TESTE, se estiver limpo ele é setado, caso contrário o programa desvia para a rotina RX para fazer a recepção de dados via USART. Ao retornar à rotina principal o programa verifica se o *flag* ACESO está setado. Nessa situação, a rotina TX é chamada para que haja o envio de dados através da comunicação USART e logo após a transmissão volta para o início da rotina principal, caso o *flag* ACESO não esteja setado o programa volta diretamente ao início do programa principal. A rotina principal, MAIN, é apresentada a seguir.

```
MAIN
        CLRWDT
        GOTO        LEITURA
MAIN2

        CALL        TRATA_SINAL
        NOP
        CALL        LIMPA_TELA
        NOP
        BTFSS       TESTE
        GOTO        $+2
        GOTO        $+3
        BSF         TESTE
        GOTO        MAIN
        BCF         TESTE
        CALL        RX
        NOP
        BTFSS       ACESO
        GOTO        MAIN
        GOTO        $+1
        CALL        TX
        GOTO        MAIN
```

A verificação de nível alto no microcontrolador PIC16F877A é feita pela rotina LEITURA. Essa rotina limpa o *watchdog timer* para que não haja estouro de tempo,

e verifica se há nível alto na entrada Sinal_Sensor. Se não há sinal o programa vai para a rotina principal MAIN. Caso haja nível alto a rotina MAIN2 é chamada. A rotina LEITURA pode ser visualizada logo abaixo.

```
LEITURA
        CLRWDT
        BTFSS       SINAL_SENSOR
        GOTO        MAIN
        GOTO        MAIN2
```

Como mostrado na rotina principal a próxima rotina a ser chamada é a TRATA_SINAL. Nela, é chamada a rotina para iniciar o display, e em seguida, o *backlight* do display é aceso, o cursor fica na endereço 0x83 do display para iniciar a escrita. A expressão “POR FAVOR” é escrita no display na primeira linha. Logo após, o cursor se posiciona no endereço 0xC3 e escreve a palavra “Silêncio!!” na segunda linha. Cerca de 1,25 segundos após a escrita no display, o *backlight* é apagado e o programa volta a rotina principal.

```
TRATA_SINAL
        CALL INICIALIZACAO_DISPLAY
        BSF    BACK
        BCF    RS
        MOVLW       0x3C
        CALL   ESCREVE
        MOVLW       .1
        CALL   DELAY_MS
        MOVLW       0X83
        CALL   ESCREVE
        BSF    RS
        MOVLW       A'P'
        CALL   ESCREVE
        MOVLW       A'O'
        CALL   ESCREVE
        MOVLW       A'R'
        CALL   ESCREVE
        MOVLW       A' '
        CALL   ESCREVE
        MOVLW       A'F'
        CALL   ESCREVE
        MOVLW       A'A'
        CALL   ESCREVE
        MOVLW       A'V'
        CALL   ESCREVE
        MOVLW       A'O'
        CALL   ESCREVE
        MOVLW       A'R'
        CALL   ESCREVE
        MOVLW       A' '
        CALL   ESCREVE
        BCF            RS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       0X3C
        CALL   ESCREVE
        MOVLW       .1
        CALL   DELAY_MS
        BSF            RS
        MOVLW       A'S'
        CALL   ESCREVE
        MOVLW       A'I'
        CALL   ESCREVE
        MOVLW       A'L'
        CALL   ESCREVE
        MOVLW       A'E'
        CALL   ESCREVE
        MOVLW       A'N'
        CALL   ESCREVE
        MOVLW       A'C'
        CALL   ESCREVE
        MOVLW       A'I'
        CALL   ESCREVE
        MOVLW       A'O'
        CALL   ESCREVE
        MOVLW       A'!'
        CALL   ESCREVE
        MOVLW       A'!'
        CALL   ESCREVE
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        MOVLW       .250
        CALL   DELAY_MS
        BCF    BACK
        RETURN
```

A rotina LIMPA_TELA tem a função de limpar a tela do display. Fazendo com que a mensagem escrita no display não permaneça na tela durante muito tempo. Como pode ser notado na rotina abaixo, o comando apaga a tela é enviado ao display.

```
LIMPA_TELA
        BCF         RS
        MOVLW       B'00000001'
        CALL        ESCREVE
        RETURN
```

Conforme mencionado na rotina principal, se o *flag* TESTE estiver setado o programa passa para a rotina RX responsável pela recepção USART. Esta rotina inicia ativando a baixa velocidade de transferência com um *baud rate* de 1200bps. E então, é ativada a transmissão assíncrona através do bit TXSTA<SYNC>. A porta serial e a recepção são habilitadas setando os bits RCSTA<SPEN> e RCSTA<CREN> respectivamente. O bit PIR1<RCIF> é testado para verificar se a recepção foi completada. Em seguida, o bit RCSTA<FERR> verifica se ocorreu erro na recepção; caso positivo a recepção é reiniciada, se não o dado disponibilizado no registrador RCREG é enviado ao work (W) e comparado com o seu identificador. Se o dado não corresponder, o programa volta ao inicio da recepção aguardando novo dado; se o dado corresponder, a recepção é desabilitada e o *flag* ACESO é setado. Em seguida o programa volta para a rotina principal. A rotina RX é apresentada a seguir.

```
RX
        BANK1
        BCF         TXSTA,BRGH
        MOVLW       .51
        MOVWF       SPBRG
        BCF         TXSTA,SYNC
        BANK0
        BSF         RCSTA,SPEN
        BSF         RCSTA,CREN
        BTFSS       PIR1,RCIF
        GOTO        $-1
        BTFSC       RCSTA,FERR
        GOTO        ER_RECEPCAO
        MOVF        RCREG,W
        XORLW       A'P'
        BTFSS       STATUS,Z
        GOTO        RX
        BCF         RCSTA,CREN
        BSF         ACESO
        RETURN
```

Ao retornar à rotina principal o *flag* ACESO é testado. Caso este *flag* esteja limpo o programa volta ao início da rotina principal. Porém, se este *flag* estiver setado o programa passa para a rotina TX onde é iniciada a transmissão do dado para o dispositivo central. Novamente é ativada a baixa velocidade de transferência com uma taxa de transferência de 1200bps, a transmissão assíncrona através do bit TXSTA<SYNC>. A porta serial é habilitada setando o bit RCSTA<SPEN>. E para habilitar a transmissão de dado o bit TXSTA<TXEN> é setado e o dado é colocado no registrador TXREG. O fim da transmissão é verificado através do bit TXSTA<TRMT>. Se o bit for igual a zero a rotina aguarda o fim da transmissão. Se o bit for igual a um significa que a transmissão do dado terminou. A transmissão é desabilitada limpando o bit TXEN. O *flag* ACESO também é limpo e o programa retorna a rotina principal. Os passos da rotina TX são apresentados a seguir.

```
TX
        BANK1
        BCF         TXSTA,BRGH
        MOVLW       .51
        MOVWF       SPBRG
        BCF         TXSTA,SYNC
        BANK0
        BSF         RCSTA,SPEN
        BANK1
        BSF         TXSTA,TXEN
        BANK0
        MOVLW       A'C'
        MOVWF       TXREG
        BANK1
        BTFSS       TXSTA,TRMT
        GOTO        $-1
        BANK1
        BCF         TXSTA,TXEN
        BCF         ACESO
        RETURN
```

A rotina apresentada a seguir mostra como o display é configurado. A comunicação usa as 8 vias de dados e com o cursor deslocado à direita, porém sem que este apareça. (SOUZA, 2003). A inicialização do display nas linhas de comando é descrita a seguir:

```
INICIALIZACAO_DISPLAY
        BCF         RS
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       .5
        CALL        DELAY_MS
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       B'00111100'
        CALL        ESCREVE
        MOVLW       B'00000001'
        CALL        ESCREVE
        MOVLW       .1
        CALL        DELAY_MS
        MOVLW       B'00001100'
        CALL        ESCREVE
        MOVLW       B'000000111'
        CALL        ESCREVE
        MOVLW       B'00000110'
        CALL        ESCREVE
        BSF         RS
RETURN
```

3.3.4 Software do dispositivo central

Este tópico descreve a implementação do programa utilizado no microcontrolador do dispositivo central. Na Figura 3.23 é ilustrado seu fluxograma.

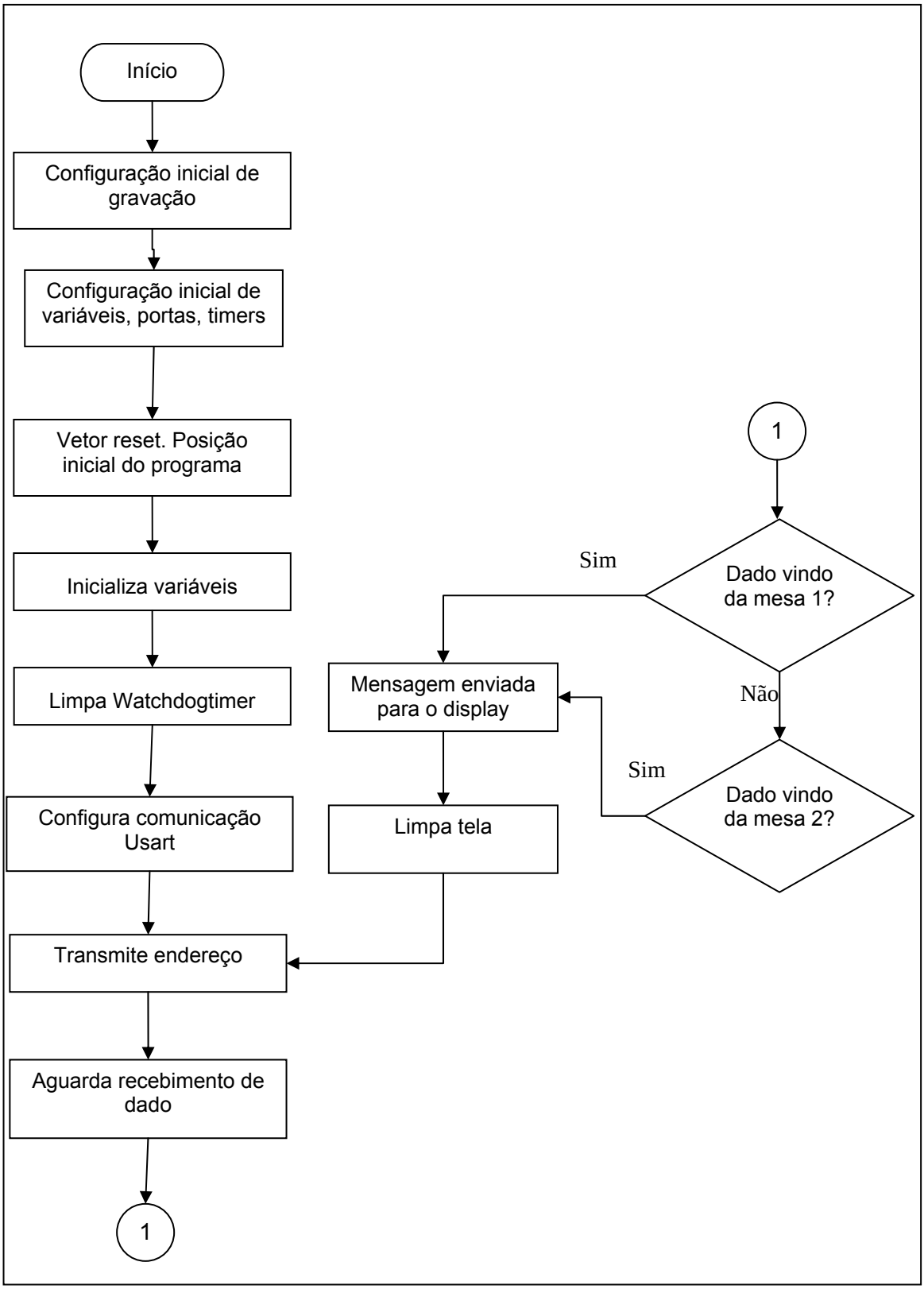


Figura 3.23 – Fluxograma da programação do dispositivo central

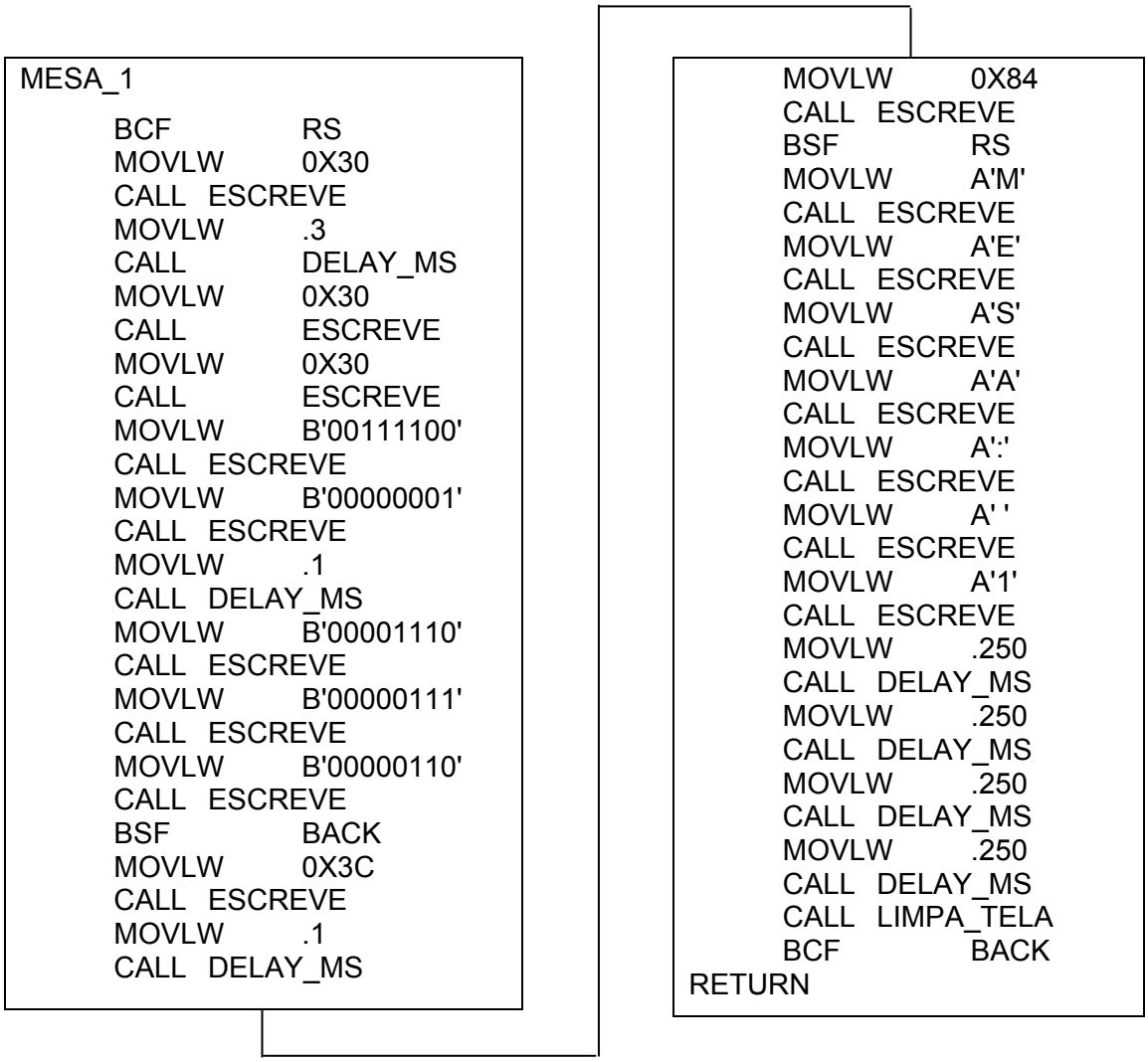
Para a realização do programa do dispositivo central as configurações iniciais são as mesmas que a do programa do dispositivo mesa. Vai desde a configuração da diretiva __CONFIG até a configuração da função CONF, com exceção da PORT RA1 e dos *flags* ACESO e TESTE que não são usados neste programa.

A rotina principal do dispositivo central é a rotina TX. Nesta rotina, ocorre a inicialização da comunicação USART no dispositivo central ativando a baixa velocidade de transferência (1200bps). Então, é ativada a transmissão assíncrona através do bit SYNC localizado no registrador TXSTA. A porta serial e a recepção são habilitadas através dos bits RCSTA<SPEN> e RCSTA<CREN>, ao habilitar o bit TXSTA<TXEN> a transmissão é ativada. Com ativação da transmissão, o dado U (utilizando o código ASCII) é carregado no registrador TXREG e enviado. Após 40ms o outro dado, P, é carregado no registrador TXREG e enviado também.

Com o envio dos endereços, o programa fica aguardando na recepção dos dados vindos dos dispositivos de mesa. O bit PIR1<RCIF> é testado para verificar se a recepção foi completada. Se o bit PIR1<RCIF> for igual a um, o dado que está no registrador RCREG é carregado no work (W) e testado se a resposta veio da mesa 1. Caso positivo, o programa chama a rotina MESA_1. Caso contrário o programa aguarda o término da recepção do novo dado e verifica se a resposta veio da mesa 2. Se a resposta for sim o programa chama a rotina MESA_2, Se não o programa retorna a rotina principal, MAIN. Os detalhes da rotina TX são vistos a seguir.

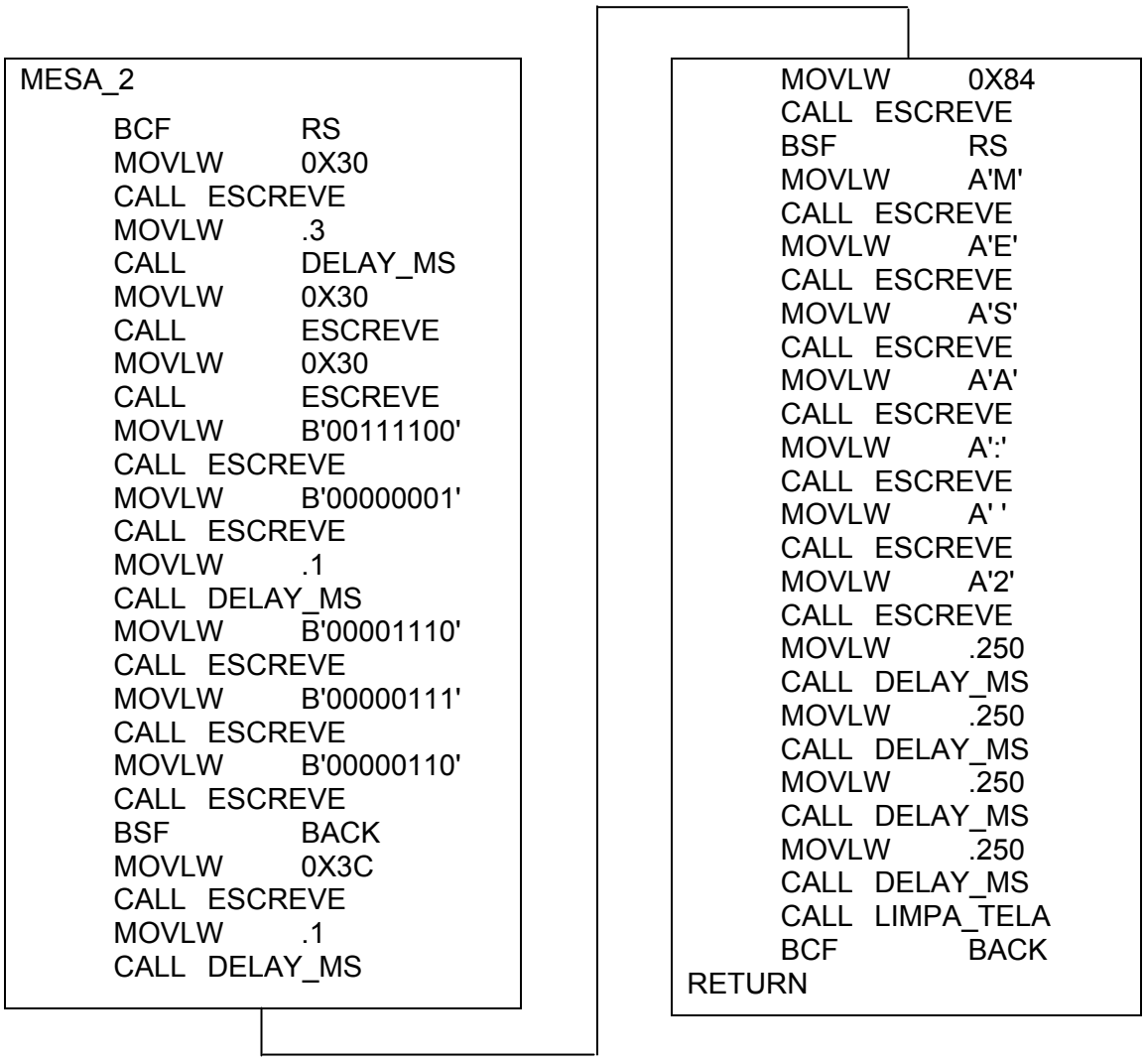
```
TX
        BANK1
        BCF         TXSTA,BRGH
        MOVLW       .51
        MOVWF       SPBRG
        BCF         TXSTA,SYNC
        BANK0
        BSF         RCSTA,SPEN
        BSF         RCSTA,CREN
        BANK1
        BSF         TXSTA,TXEN
        BANK0
        MOVLW       A'U'
        MOVWF       TXREG
        MOVLW       .40
        CALL        DELAY_MS
        MOVLW       A'P'
        MOVWF       TXREG
        BTFSS       PIR1,RCIF
        GOTO        $-1
        MOVF        RCREG,W
        XORLW       A'B'
        BTFSS       STATUS,Z
        GOTO        $+3
        GOTO        $+1
        CALL        MESA_1
        BTFSS       PIR1,RCIF
        GOTO        $-1
        MOVF        RCREG,W
        XORLW       A'C'
        BTFSS       STATUS,Z
        GOTO        $+3
        GOTO        $+1
        CALL        MESA_2
        BANK1
        BTFSS       TXSTA,TRMT
        GOTO        $-1
RETURN
```

Quando o programa confirma que o dado veio do dispositivo da mesa 1 a rotina MESA_1 é chamada. Esta rotina é iniciada com a configuração do display da mesma forma que foi configurado o display do dispositivo de mesa. O *backlight* é aceso e o display escreve “MESA: 1” em sua tela. Passado um segundo, a rotina LIMPA_TELA é chamada para limpar a tela e o *backlight* é apagado. A rotina é descrita a seguir.



```
MESA_1
        BCF         RS
        MOVLW       0X30
        CALL  ESCREVE
        MOVLW       .3
        CALL        DELAY_MS
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       B'00111100'
        CALL  ESCREVE
        MOVLW       B'00000001'
        CALL  ESCREVE
        MOVLW       .1
        CALL  DELAY_MS
        MOVLW       B'00001110'
        CALL  ESCREVE
        MOVLW       B'00000111'
        CALL  ESCREVE
        MOVLW       B'00000110'
        CALL  ESCREVE
        BSF         BACK
        MOVLW       0X3C
        CALL  ESCREVE
        MOVLW       .1
        CALL  DELAY_MS
        MOVLW       0X84
        CALL  ESCREVE
        BSF         RS
        MOVLW       A'M'
        CALL  ESCREVE
        MOVLW       A'E'
        CALL  ESCREVE
        MOVLW       A'S'
        CALL  ESCREVE
        MOVLW       A'A'
        CALL  ESCREVE
        MOVLW       A':'
        CALL  ESCREVE
        MOVLW       A' '
        CALL  ESCREVE
        MOVLW       A'1'
        CALL  ESCREVE
        MOVLW       .250
        CALL  DELAY_MS
        MOVLW       .250
        CALL  DELAY_MS
        MOVLW       .250
        CALL  DELAY_MS
        MOVLW       .250
        CALL  DELAY_MS
        CALL  LIMPA_TELA
        BCF         BACK
RETURN
```

E quando o programa verifica que o dado veio do dispositivo da mesa 2, a rotina MESA_2 é chamada para que o display escreva em sua tela “MESA 2”. Com as mesmas configurações feitas na rotina MESA_1. E novamente a rotina LIMPA_TELA é chamada para que a tela do display seja limpa. Isso pode ser confirmado com a rotina ilustrada a seguir.



```
MESA_2
        BCF         RS
        MOVLW       0X30
        CALL  ESCREVE
        MOVLW       .3
        CALL        DELAY_MS
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       0X30
        CALL        ESCREVE
        MOVLW       B'00111100'
        CALL  ESCREVE
        MOVLW       B'00000001'
        CALL  ESCREVE
        MOVLW       .1
        CALL  DELAY_MS
        MOVLW       B'00001110'
        CALL  ESCREVE
        MOVLW       B'00000111'
        CALL  ESCREVE
        MOVLW       B'00000110'
        CALL  ESCREVE
        BSF         BACK
        MOVLW       0X3C
        CALL  ESCREVE
        MOVLW       .1
        CALL  DELAY_MS
        MOVLW       0X84
        CALL  ESCREVE
        BSF         RS
        MOVLW       A'M'
        CALL  ESCREVE
        MOVLW       A'E'
        CALL  ESCREVE
        MOVLW       A'S'
        CALL  ESCREVE
        MOVLW       A'A'
        CALL  ESCREVE
        MOVLW       A':'
        CALL  ESCREVE
        MOVLW       A' '
        CALL  ESCREVE
        MOVLW       A'2'
        CALL  ESCREVE
        MOVLW       .250
        CALL  DELAY_MS
        MOVLW       .250
        CALL  DELAY_MS
        MOVLW       .250
        CALL  DELAY_MS
        MOVLW       .250
        CALL  DELAY_MS
        CALL  LIMPA_TELA
        BCF         BACK
RETURN
```

Assim, o programa do dispositivo central continua enviando os dados para os dispositivos de mesa e sempre aguardando uma resposta.

4. CONCLUSÃO

Objetivou-se neste trabalho a implementação de um sistema de monitoração sonora, que detectasse o excesso de ruído em ambientes onde é necessário que haja silêncio. Para tal pesquisou-se sobre sensores sonoros, microcontroladores, comunicação entre microcontroladores e a linguagem de programação Assembler.

Algumas dificuldades foram encontradas no decorrer do projeto, tais como a definição do sensor, a programação do microcontrolador e a comunicação entre os dispositivos.

Inicialmente optou-se por um sensor que trabalhava com tensão de 5 volts. Tensão esta idêntica à tensão de trabalho do microcontrolador. Mas este sensor não possuía sensibilidade suficiente para a detecção do som. Então, a solução foi substituir este sensor por outro, que trabalhava com tensão de 12 volts. Como essa tensão é maior que a tensão de trabalho do microcontrolador, foi necessário utilizar resistores para interligar o circuito do sensor ao microcontrolador.

Quanto à programação, a dificuldade surgiu no arquivo linker, que faz a divisão da memória de programa do PIC16F877A. Este arquivo não estava compilando corretamente quando se utilizava o vetor de reset no endereço 0x00. Aparentemente, o problema foi solucionado mudando o endereço de inicialização do programa para a posição 0x05. Posteriormente, o problema voltou a acontecer e após várias pesquisas a solução foi descartar este arquivo. Solução esta que não provocou queda de desempenho do projeto.

Com relação a comunicação entre os dispositivos, foi escolhida, inicialmente, a comunicação I^2C, mas esta não funcionou como o esperado. Foram feitas várias tentativas, tais como: utilizar interrupções, testar apenas a comunicação I^2C, trocas de faixas de endereço e tentativas de comunicação por broadcasting. Contudo, nenhuma dessas opções levou ao funcionamento da comunicação I^2C. A solução encontrada para viabilizar o funcionamento do projeto foi utilizar a comunicação USART, que é bem mais simples.

Após as correções dos problemas relatados anteriormente, o circuito funcionou devidamente. O nível sonoro é detectado pelo dispositivo de mesa e uma mensagem aparece em seu display solicitando silêncio. Quando isso ocorre pela

segunda vez, o dispositivo de mesa informa ao dispositivo central, que escreve uma mensagem em seu display para informar ao funcionário responsável pelo local.

Dessa forma, todo o circuito está funcionando corretamente.

Este projeto foi desenvolvido de forma a proporcionar o menor custo possível. Para tanto foram utilizados componentes de baixo custo e que atendessem aos requisitos necessários. Dessa forma, o custo estimado do hardware do projeto ficou em torno de R$ 160,00. Caso se deseje fazer uma ampliação no número de dispositivos de mesa pode-se estimar um custo de R$ 50,00 por dispositivo.

Como sugestões para projetos futuros ficam as seguintes opções:

a) Implementar uma rede de sensores sonoros controlada por apenas um microcontrolador.

b) Acoplar uma câmera ao dispositivo de mesa. Ao detectar nível sonoro alto o dispositivo de mesa enviaria as imagens para o dispositivo central.

c) Implementar a comunicação sem fio entre o dispositivo de mesa e o dispositivo central.

d) Enviar através dos dispositivos de mesa o áudio captado pelos sensores para o dispositivo central.

REFERÊNCIAS BIBLIOGRÁFICAS

DETETIVE. **Projeto de escuta**. [Home page]. 2001. Disponível em <http://www.geocities.com/projeto periferia5/voz.htm>. Acessado em 22 de maio de 2007

LOPES, Ana Cristina. **Microprocessadores e Aplicações: Acetatos de apoio às aulas teóricas. Departamento de Engenharia Eletrocténica**. [Home page]. 2001. Disponível em <http://www.orion.ipt.pt/.>. Acessado em 08 de outubro de 2007.

MICROCHIP. **I^2C^{TM} Master Mode: Overview and Use of the PICmicro® MSSP I^2C Interface with a 24xx01x EEPROM.** [Home page]. 2001. Disponível em <http://ww1. microchip.com/downloads/en/devicedoc/i2c.pdf >. Acessado em 12 de outubro de 2007.

MICROCHIP. **PIC16F87XA - Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers.** [Home page]. 2000. Disponível em <http:///ww1.microchip.com/ downloads/en/DeviceDoc/39582b.pdf >. Acessado em 03 de março 2007.

OSHON SOFTWARE PROJECT. **PIC Simulator IDE** [Home page]. 2007. Disponível em <http://www.oshonsoft.com/pic.html>. Acessado em 22 de fevereiro de 2007

PEREIRA, Fábio. **Microcontroladores PIC: Técnicas Avançadas**. 3ª edição. 2004

SOUZA, David José de; LAVÍNIA, Nicolas César. **Conectando o PIC 16F877A: Recursos Avançados**. 1ª edição. 2003

ZANCO, Wagner da Silva. **Microcontroladores PIC16F628A/648A: Uma abordagem prática e objetiva**.1ª edição. 2005.

ZANCO, Wagner da Silva. **Microcontroladores PIC: Técnicas de Software e Hardware para Projetos de Circuitos Eletrônicos Com Base no PIC16F877A**. 1ª edição. 2006

APÊNDICE – CÓDIGO FONTE DO SOFTWARE

Dispositivo de mesa

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                      CONFIGURAÇÕES PARA GRAVAÇÃO                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 __CONFIG _CP_OFF & _CPD_OFF & _DEBUG_OFF & _LVP_OFF & _WRT_OFF &
_BODEN_OFF & _PWRTE_ON & _WDT_ON & _XT_OSC

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                          DEFINIÇÃO DAS VARIÁVEIS                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTE BLOCO DE VARIÁVEIS ESTÁ LOCALIZADO LOGO NO INÍCIO DO BANCO 0

      CBLOCK      0X20        ; POSIÇÃO INICIAL DA RAM. ATRIBUIÇÃO DIRETA DE
                              ; ENDEREÇO A CADA VARIÁVEL
            TEMPO1            ; CONTADOR P/ DELAY. POSIÇÃO 0X21
            TEMPO0            ; CONTADORES P/ DELAY. POSIÇÃO 0X22
            FLAG              ; FLAG DE USO GERAL. POSIÇÃO 0X23
            DADO              ; REGISTRADOR DE ENVIO DE DADO PELO USART
      ENDC
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                DEFINIÇÃO DAS VARIÁVEIS INTERNAS DO PIC                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      #INCLUDE <P16F877A.INC>       ; MICROCONTROLADOR UTILIZADO
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                      DEFINIÇÃO DOS BANCOS DE RAM                      *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;OS PSEUDOS-COMANDOS "BANK0" E "BANK1" AJUDAM A COMUTAR ENTRE OS BANCOS DE
MEMÓRIA.

#DEFINE     BANK1 BSF   STATUS,RP0  ; SELECIONA BANK1 DA MEMORIA RAM
#DEFINE     BANK0 BCF   STATUS,RP0  ; SELECIONA BANK0 DA MEMORIA RAM

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   VETOR DE RESET DO MICROCONTROLADOR                  *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
;POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

      ORG   0X00              ; ENDEREÇO DO VETOR DE RESET
      GOTO  CONF              ; PULA PARA CONF

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                   DECLARAÇÃO DOS FLAGs DE SOFTWARE                    *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;A DEFINIÇÃO DE FLAGs AJUDA NA PROGRAMAÇÃO E ECONOMIZA MEMÓRIA RAM.

#DEFINE     ACESO FLAG,1            ; 1 -> INICIA COMUNICAÇÃO USART
                                    ; 0 -> VOLTA AO PROGRAMA PRINCIPAL
#DEFINE     TESTE FLAG,2            ; 1 -> REINCIDENCIA DE NÍVEL ALTO
                                    ; 0 -> PRIMEIRA OCORRÊNCIA DE NÍVEL ALTO
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                ENTRADA                                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;A ENTRADA ASSOCIADA A UM NOME PARA FACILITAR A PROGRAMAÇÃO E
;FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE     SINAL_SENSOR      PORTA, 1    ;ESTADO DA ENTRADA SINAL_SENSOR NA
                                    ; PORTA RA1, PINO 3 NO PIC
                                    ;1 -> NÍVEL ALTO
                                    ;0 -> NÍVEL NORMAL
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                                SAÍDAS                                 *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;AS SAÍDAS ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;FUTURAS ALTERAÇÕES DO HARDWARE.

#DEFINE     BACK        PORTB,1     ;BACKLIGHT DO DISPLAY
                                    ; 1-> ACENDE
                                    ; 0-> APAGA
#DEFINE     DISPLAY     PORTD       ; BARRAMENTO DE DADOS DO DISPLAY
#DEFINE     RS          PORTE,0     ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                    ; 1 -> DADO
                                    ; 0 -> COMANDO

#DEFINE     ENABLE      PORTE,1     ; SINAL DE ENABLE P/ DISPLAY
                                    ; ATIVO NA BORDA DE DESCIDA
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
; *                    ROTINA DE DELAY (DE 1MS ATÉ 256MS)                *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;ESTA É UMA ROTINA DE DELAY VARIÁVEL, COM DURAÇÃO DE 1MS X O VALOR PASSADO
;EM WORK (W).

DELAY_MS
      MOVWF       TEMPO1      ; CARREGA TEMPO1 (UNIDADES DE MS)
      MOVLW       .250
      MOVWF       TEMPO0      ; CARREGA TEMPO0 (P/ CONTAR 1MS)

      CLRWDT                  ; LIMPA WDT (PERDE TEMPO)
      DECFSZ      TEMPO0,F    ; FIM DE TEMPO0 ?
      GOTO        $-2         ; NÃO - VOLTA 2 INSTRUÇÕES
                              ; SIM - PASSOU-SE 1MS
      DECFSZ      TEMPO1,F    ; FIM DE TEMPO1 ?
      GOTO        $-6         ; NÃO - VOLTA 6 INSTRUÇÕES
                              ; SIM
RETURN                        ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              ROTINA DE ESCRITA DE UM CARACTER NO DISPLAY            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA ENVIA UM CARACTER PARA O MÓDULO DE LCD. O CARACTER A SER
; ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.

ESCREVE
      MOVWF DISPLAY               ; ATUALIZA DISPLAY (PORTD)
      NOP                         ; PERDE 1US PARA ESTABILIZAÇÃO
      BSF   ENABLE                ; ENVIA UM PULSO DE ENABLE AO DISPLAY
      MOVLW .1
      CALL  DELAY_MS
      GOTO  $+1
      BCF   ENABLE
      MOVLW .1
      CALL  DELAY_MS          ; DELAY DE 1MS
RETURN                            ; RETORNA

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *            CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; NESTA ROTINA SÃO INICIALIZADAS AS PORTAS DE I/O DO MICROCONTROLADOR E AS
```

```
; CONFIGURAÇÕES DOS REGISTRADORES ESPECIAIS (SFR). A ROTINA INICIALIZA A
; MÁQUINA E AGUARDA O ESTOURO DO WDT.

CONF
      CLRF  PORTA            ; GARANTE AS SAÍDAS EM ZERO
      CLRF  PORTB
      CLRF  PORTC
      CLRF  PORTD
      CLRF  PORTE
      BANK1                  ; SELECIONA BANCO 1 DA RAM
      MOVLW B'11111111'
      MOVWF TRISA            ; CONFIGURA PORTA COMO ENTRADA
      MOVLW B'11111001'      ;CONFIGURA PORTB COMO ENTRADA
      MOVWF TRISB            ;EXCETO PORTB RB1 E RB:SAÍDA
      CLRF  PORTB
      MOVLW B'10000000'
      MOVWF TRISC            ; CONFIGURA PORTC COMO ENTRADA

      MOVLW B'00000000'
      MOVWF TRISD            ; CONFIGURA PORTD COMO SAÍDA
      MOVLW B'00000100'
      MOVWF TRISE            ; CONFIGURA RE0 E RE1 COMO SAÍDA
      MOVLW B'11011111'
      MOVWF OPTION_REG       ; CONFIGURA O REGISTRADOR OPTION
                             ; COMO OS PULL-UPs DA PORTB DESABILITADOS
                             ; INTER. NA BORDA DE SUBIDA DO RB0
                             ; O TMR0 SERÁ INCREMENTADO EXTERNAMENTE
                             ; PELA MUDANÇA DO PINO RA4/T0CKI
                             ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                             ; WDT - 1:128 SERÃO NECESSÁRIOS 128
                             ; CICLOS DE MAQ. P/ O PROGRAMA SER SETADO
                             ; TIMER - 1:1
      MOVLW B'11000000'      ; HABILITADA AS INTERRUPÇÕES GIE(CHAVE GERAL
                             ; DAS INTERRUPÇÕES) E PEIE (INTERRUPÇÕES DE
      MOVWF INTCON           ;PERIFÉRICOS).

      MOVLW B'00001000'      ; HABILITA A INTERRUPÇÃO SSPIF (INTERRUP.
                             ; COMUNICAÇÃO PARALELA SÍNCRONA).
      MOVWF PIR1
      MOVLW B'00011000'      ; HABILITA AS INTERRUPÇÃO TXIE(HABILI.
      MOVWF PIE1             ; INTERRUPÇÃO) E SSPIE(HABILI. INTERRUPÇÃO
```

```
                              ;NA PORTA SERIAL SSP).
      MOVLW B'00000111'       ; CONFIGURA PORTA E PORTE COMO DIGITAL
      MOVWF ADCON1            ; E TENSÃO DE REFERÊNCIA vDD(+5V)
                              ; E vSS
      BANK0                   ; SELECIONA BANCO 0 DA RAM
      BCF   TESTE             ; LIMPEZA DOS FLAGS
      BCF   ACESO

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                            ROTINA PRINCIPAL                           *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

MAIN
      CLRWDT
      GOTO  LEITURA
MAIN2
      CALL  TRATA_SINAL
      NOP
      CALL  LIMPA_TELA
      NOP
      BTFSS TESTE
      GOTO  $+2
      GOTO  $+3
      BSF   TESTE
      GOTO  MAIN
      BCF   TESTE
      CALL  RX
      NOP
      BTFSS ACESO
      GOTO  MAIN
      GOTO  $+1
      CALL  TX
      GOTO  MAIN

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *               ROTINA DE ESCRITA DA TELA PRINCIPAL                     *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ESCREVE A TELA PRINCIPAL DO PROGRAMA AS PALAVRAS
;  " POR FAVOR " QUE INTERCALA COM A PALAVRAS "SILÊNCIO!!"

TRATA_SINAL
```

```
CALL  INICIALIZACAO_DISPLAY
BSF   BACK
BCF   RS                ; SELECIONA DISPLAY PARA COMANDOS
MOVLW 0X3C              ; ESCREVE COMANDO PARA
CALL  ESCREVE           ; LIMPAR A TELA
MOVLW .1
CALL  DELAY_MS          ; DELAY DE 1MS
MOVLW 0X83              ; COMANDO PARA POSICIONAR O CURSOR
CALL  ESCREVE           ; LINHA 0 / COLUNA 1
BSF   RS                ; SELECIONA O DISPLAY P/ DADOS
                        ; COMANDOS PARA ESCREVER
                        ; "POR FAVOR"
MOVLW A'P'
CALL  ESCREVE
MOVLW A'O'
CALL  ESCREVE
MOVLW A'R'
CALL  ESCREVE
MOVLW A' '
CALL  ESCREVE
MOVLW A'F'
CALL  ESCREVE
MOVLW A'A'
CALL  ESCREVE
MOVLW A'V'
CALL  ESCREVE
MOVLW A'O'
CALL  ESCREVE
MOVLW A'R'
CALL  ESCREVE
MOVLW A' '
CALL  ESCREVE
MOVLW .250
CALL  DELAY_MS
MOVLW .250
CALL  DELAY_MS
MOVLW .250
CALL  DELAY_MS
MOVLW .250
CALL  DELAY_MS
BCF   RS
```

```
      MOVLW 0XC3                    ; COMANDO PARA POSICIONAR O CURSOR
      CALL  ESCREVE                 ; LINHA 1 / COLUNA 1
      MOVLW .1
      CALL  DELAY_MS
      BSF   RS                      ; SELECIONA O DISPLAY P/ DADOS
                                    ; COMANDOS PARA ESCREVER
                                    ; DE "SILENCIO!!"
      MOVLW A'S'
      CALL  ESCREVE
      MOVLW A'I'
      CALL  ESCREVE
      MOVLW A'L'
      CALL  ESCREVE
      MOVLW A'E'
      CALL  ESCREVE
      MOVLW A'N'
      CALL  ESCREVE
      MOVLW A'C'
      CALL  ESCREVE
      MOVLW A'I'
      CALL  ESCREVE
      MOVLW A'O'
      CALL  ESCREVE
      MOVLW A'!'
      CALL  ESCREVE
      MOVLW A'!'
      CALL  ESCREVE
      MOVLW .250
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      BCF   BACK
RETURN


;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                            COMUNICAÇÃO USART                            *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
RX
      BANK1                   ;ALTERA PARA O BANCO 1
      BCF   TXSTA,BRGH        ;ATIVA BAIXA VELOCIDADE DE TRANSFERENCIA
      MOVLW .51               ;CONFIGURA BAUD RATE PARA 4800 bps
      MOVWF SPBRG
      BCF   TXSTA,SYNC        ;ATIVA COMUNICAÇÃO ASSINCRONA
      BANK0                   ;ALTERA PARA O BANCO 0
      BSF   RCSTA,SPEN        ;HABILITA A PORTA SERIAL
      BSF   RCSTA,CREN        ;HABILITA RECEPÇÃO
      BTFSS PIR1,RCIF         ;RECEPÇÃO COMPLETADA?
      GOTO  $-1               ;AGUARDA FIM DA RECEPÇÃO
      BTFSC RCSTA,FERR
      GOTO  ER_RECEPCAO
      MOVF  RCREG,W
      XORLW A'U'              ; VERIFICA SE O ENDEREÇO RECEBIDO É DA MESA 1,
                              ; SE ENDEREÇO FOR IGUAL A ‘U’ OU DA MESA 2 SE
                              ;O ENDEREÇO FOR IGUAL A ‘P’
      BTFSS STATUS,Z
      GOTO  RX
      BCF   RCSTA,CREN
      BSF   ACESO
RETURN

ER_RECEPCAO
      BCF   RCSTA,CREN
      GOTO  RX

TX
      BANK1                   ;ALTERA PARA O BANCO 1
      BCF   TXSTA,BRGH        ;ATIVA BAIXA VELOCIDADE DE TRANSFERENCIA
      MOVLW .51               ;CONFIGURA BAUD RATE PARA 4800 bps
      MOVWF SPBRG
      BCF   TXSTA,SYNC        ;ATIVA COMUNICAÇÃO ASSINCRONA
      BANK0                   ;ALTERA PARA O BANCO 0
      BSF   RCSTA,SPEN        ;HABILITA A PORTA SERIAL
      MOVLW A'B'              ;CARREGA NO REGISTRADOR DE TRANSMISSÃO A
                              ;LETRA ‘B’ PARA A MESA 1 E CARREGA A LETRA
      MOVWF TXREG             ;’C’ PARA A MESA 2.
      BANK1
      BSF   TXSTA,TXEN        ;HABILITA TRANSMISSÃO
      BANK0
```

```
      MOVLW A'B'              ;CARREGA NO REGISTRADOR DE TRANSMISSÃO O
                              ; LETRA ‘B’ PARA A MESA 1 E CARREGA A LETRA
      MOVWF TXREG             ; ;’C’ PARA A MESA 2.
      BANK1
      BTFSS TXSTA,TRMT        ;AGUARDA TERMINO DA TRANSMISSÃO
      GOTO  $-1               ;NÃO
      BANK1
      BCF   TXSTA,TXEN        ;DESABILITA TRANSMISSÃO
      BCF   ACESO
RETURN

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                            LEITURA DA PORTA RA0                         *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ROTINA FICA VERIFICANDO SE HÁ SINAL ALTO NA ENTRADA

LEITURA
      CLRWDT                        ; LIMPA WATCHDOG TIMER
      BTFSS SINAL_SENSOR      ; VERIFICA SE A PORTA RA1 ESTÁ COM NÍVEL 1.
      GOTO  MAIN
      GOTO  MAIN2             ; SIM - PULA PARA TRATA_SINAL

; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *              CONFIGURAÇÕES INICIAIS DO DISPLAY                        *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; ESTA ROTINA INICIALIZA O DISPLAY P/ COMUNICAÇÃO DE 8 VIAS, DISPLAY PARA 2
; LINHAS, CURSOR APAGADO E DESLOCAMENTO DO CURSOR À DIREITA.

INICIALIZACAO_DISPLAY
      BCF   RS                ; SELECIONA O DISPLAY P/ COMANDOS
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW .3
      CALL  DELAY_MS          ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW B'00111100'       ; ESCREVE COMANDO PARA
      CALL  ESCREVE           ; INTERFACE DE 8 VIAS DE DADOS
      MOVLW B'00000001'       ; ESCREVE COMANDO PARA
```

```
      CALL  ESCREVE           ; LIMPAR TODO O DISPLAY
      MOVLW .1
      CALL  DELAY_MS          ; DELAY DE 1MS
      MOVLW B'00001100'       ; ESCREVE COMANDO PARA
      CALL  ESCREVE           ; LIGAR O DISPLAY SEM CURSOR
      MOVLW B'000000111'
      CALL  ESCREVE
      MOVLW B'00000110'       ; ESCREVE COMANDO PARA INCREM.
      CALL  ESCREVE           ; AUTOMÁTICO À DIREITA
      BSF   RS                ;SELECIONA O DISPLAY PARA DADOS
RETURN

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                                LIMPA TELA                               *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

LIMPA_TELA
      BCF   RS                ; SELECIONA O DISPLAY P/ COMANDOS
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW B'00000001'       ; ESCREVE COMANDO PARA
      CALL  ESCREVE           ; LIMPAR TODO O DISPLAY
RETURN

; * * *  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *                              FIM DO PROGRAMA                            *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

      END                     ; FIM DO PROGRAMA
```

Dispositivo Central

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                            ARQUIVO DE DEFINIÇÃO                        *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      #INCLUDE <P16F877A.INC>       ; MICROCONTROLADOR UTILIZADO
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                           CONFIGURAÇÕES PARA GRAVAÇÃO                  *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 __CONFIG _CP_OFF & _CPD_OFF & _DEBUG_OFF & _LVP_OFF & _WRT_OFF &
_BODEN_OFF & _PWRTE_ON & _WDT_ON & _XT_OSC

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                            PAGINACAO DA MEMORIA                        *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;COMANDOS PARA ALTERACAO DE PAGINA DE MEMORIA

#DEFINE     BANK0 BCF         STATUS,RP0
#DEFINE     BANK1 BSF         STATUS,RP0
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                            VARIÁVEIS UTILIZADAS                        *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; BLOCO DE VARIÁVEIS LOCALIZADO NO INÍCIO DO BANCO 0

      CBLOCK      0X20        ; POSIÇÃO INICIAL DA RAM. ATRIBUIÇÃO DIRETA DE
                              ; ENDEREÇO A CADA VARIÁVEL
            TEMPO1            ; CONTADOR P/ DELAY. POSIÇÃO 0X21
            TEMPO0            ; CONTADORES P/ DELAY. POSIÇÃO 0X22

      ENDC

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                       VETOR DE RESET DO MICROCONTROLADOR               *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; POSIÇÃO INICIAL PARA EXECUÇÃO DO PROGRAMA

      ORG   0x0000            ; ENDEREÇO DO VETOR DE RESET
      GOTO  CONF              ; PULA PARA CONF
```

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                                SAÍDAS                                 *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  AS SAÍDAS ESTÃO ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
;  FUTURAS ALTERAÇÕES.

#DEFINE     BACK        PORTB,1     ;BACKLIGHT DO DISPLAY
                                    ; 1-> ACENDE
                                    ; 0-> APAGA
#DEFINE     DISPLAY     PORTD       ; BARRAMENTO DE DADOS DO DISPLAY
#DEFINE     RS          PORTE,0     ; INDICA P/ O DISPLAY UM DADO OU COMANDO
                                    ; 1 -> DADO
                                    ; 0 -> COMANDO

#DEFINE     ENABLE      PORTE,1     ; SINAL DE ENABLE P/ DISPLAY
                                    ; ATIVO NA BORDA DE DESCIDA

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                     ROTINA DE DELAY (DE 1MS ATÉ 256MS)                *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  DELAY VARIÁVEL COM DURAÇÃO DE 1MS X O VALOR PASSADO EM WORK (W).

DELAY_MS
      MOVWF TEMPO1                  ; CARREGA TEMPO1 (UNIDADES DE MS)
      MOVLW .250
      MOVWF TEMPO0                  ; CARREGA TEMPO0 (P/ CONTAR 1MS)
      CLRWDT                        ; LIMPA WDT (PERDE TEMPO)
      DECFSZ      TEMPO0,F          ; FIM DE TEMPO0 ?
      GOTO  $-2                     ; NÃO - VOLTA 2 INSTRUÇÕES
                                    ; SIM - PASSOU-SE 1MS
      DECFSZ      TEMPO1,F          ; FIM DE TEMPO1 ?
      GOTO  $-6                     ; NÃO - VOLTA 6 INSTRUÇÕES
                                    ; SIM
RETURN                              ; RETORNA

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*          ROTINA DE ESCRITA DE UM CARACTER NO DISPLAY                  *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ESTA ROTINA ENVIA UM CARACTER PARA O MÓDULO DE LCD. O CARACTER A SER
;  ESCRITO DEVE SER COLOCADO EM WORK (W) ANTES DE CHAMAR A ROTINA.
```

```
ESCREVE
      MOVWF DISPLAY                ; ATUALIZA DISPLAY (PORTD)
      NOP                          ; PERDE 1US PARA ESTABILIZAÇÃO
      BSF   ENABLE                 ; ENVIA UM PULSO DE ENABLE AO DISPLAY
      MOVLW .1
      CALL  DELAY_MS
      GOTO  $+1
      BCF   ENABLE
      MOVLW .1
      CALL  DELAY_MS               ; DELAY DE 1MS
RETURN                             ; RETORNA

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                CONFIGURAÇÕES INICIAIS DE HARDWARE E SOFTWARE        *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

CONF
      BANK1
      MOVLW B'11111111'
      MOVWF TRISA            ; CONFIGURA PORTA COMO ENTRADA
      MOVLW B'11111101'      ;CONFIGURA PORTB COMO ENTRADA
      MOVWF TRISB            ; EXCETO PORTB RB1: SAÍDA
      MOVLW B'10000000'
      MOVWF TRISC            ; CONFIGURA PORTC COMO ENTRADA
      MOVLW B'00000000'
      MOVWF TRISD            ; CONFIGURA PORTD COMO SAÍDA
      MOVLW B'00000100'
      MOVWF TRISE            ; CONFIGURA RE0 E RE1 COMO SAÍDA
      MOVLW B'11011111'
      MOVWF OPTION_REG       ; CONFIGURA O REGISTRADOR OPTION
                             ; COMO OS PULL-UPs DESABILITAD
                             ; INTER. NA BORDA DE SUBIDA DO RB0
                             ; O TMR0 SERÁ INCREMENTADO EXTERNAMENTE
                             ; PELA MUDANÇA DO PINO RA4/T0CKI
                             ; TIMER0 INCREM. PELO CICLO DE MÁQUINA
                             ; WDT   - 1:128 SERÃO NECESSÁRIOS 128
                             ; CICLOS DE MAQ. P/ O PROGRAMA SER SETADO
                             ; TIMER - 1:1
      MOVLW B'00000000'      ; CHAVE GERAL DE INTERRUPCAO DESLIGADA
      MOVWF INTCON
      MOVLW B'00000111'      ; CONFIGURA PORTA E PORTE COMO DIGITAL
```

```
      MOVWF ADCON1            ; E TENSÃO DE REFERÊNCIA vDD(+5V)
                               E vSS
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                      COMUNICAÇÃO USART: TX E RX                       *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

TX
      BANK1                   ;ALTERA PARA O BANCO 1
      BCF   TXSTA,BRGH        ;ATIVA BAIXA VELOCIDADE DE TRANSFERENCIA
      MOVLW .51               ;CONFIGURA BAUD RATE PARA 4800 bps
      MOVWF SPBRG
      BCF   TXSTA,SYNC        ;ATIVA COMUNICAÇÃO ASSINCRONA
      BANK0                   ;ALTERA PARA O BANCO 0
      BSF   RCSTA,SPEN        ;HABILITA A PORTA SERIAL
      BSF   RCSTA,CREN        ;HABILITA RECEPÇÃO
      BANK1
      BSF   TXSTA,TXEN        ;HABILITA TRANSMISSÃO
      BANK0
      MOVLW A'U'        ;CARREGA NO REGISTRADOR DE TRANSMISSÃO O ENDEREÇO
      MOVWF TXREG             ;TRANSMITE O ENDEREÇO
      MOVLW A'U'        ;CARREGA NO REGISTRADOR DE TRANSMISSÃO O ENDEREÇO
      MOVWF TXREG             ;TRANSMITE O ENDEREÇO
      MOVLW .40
      CALL  DELAY_MS
      MOVLW A'P'        ;CARREGA NO REGISTRADOR DE TRANSMISSÃO O ENDEREÇO
      MOVWF TXREG             ;TRANSMITE O ENDEREÇO
      MOVLW A'P'        ;CARREGA NO REGISTRADOR DE TRANSMISSÃO O ENDEREÇO
      MOVWF TXREG             ;TRANSMITE O ENDEREÇO
      MOVLW .2
      BTFSS PIR1,RCIF         ;RECEPÇÃO COMPLETADA?
      GOTO  $-1               ;AGUARDA FIM DA RECEPÇÃO
      MOVF  RCREG,W           ;CARREGA O WORK COM O DADO RECEBIDO EM RCREG
      XORLW A'B'
      BTFSS STATUS,Z          ;MESA 1 RESPONDEU?
      GOTO  $+3               ;VERIFICA FIM DA TRANSMISSÃO.
      GOTO  $+1
      CALL  MESA_1            ;ESCREVE NO DISPLAY
      BTFSS PIR1,RCIF         ;RECEPÇÃO COMPLETADA?
      GOTO  $-1               ;AGUARDA FIM DA RECEPÇÃO
      MOVF  RCREG,W
      XORLW A'C'
```

```
      BTFSS STATUS,Z
      GOTO  $+3
      GOTO  $+1
      CALL  MESA_2
      BANK1
      BTFSS TXSTA,TRMT        ;AGUARDA TERMINO DA TRANSMISSÃO
      GOTO  $-1

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                      ESCRITA NO DISPLAY MESA: 1                       *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

MESA_1
      BCF   RS                ; SELECIONA DISPLAY PARA COMANDOS
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW .3
      CALL  DELAY_MS          ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE           ; INICIALIZAÇÃO
      MOVLW B'00111100'       ; ESCREVE COMANDO PARA
      CALL  ESCREVE           ; INTERFACE DE 8 VIAS DE DADOS
      MOVLW B'00000001'       ; ESCREVE COMANDO PARA
      CALL  ESCREVE           ; LIMPAR TODO O DISPLAY
      MOVLW .1
      CALL  DELAY_MS          ; DELAY DE 1MS
      MOVLW B'00001110'       ; ESCREVE COMANDO PARA
      CALL  ESCREVE           ; LIGAR O DISPLAY SEM CURSOR PISCANTE
      MOVLW B'00000111'
      CALL  ESCREVE
      MOVLW B'00000110'       ; ESCREVE COMANDO PARA INCREM.
      CALL  ESCREVE           ; AUTOMÁTICO À DIREITA
      BSF   BACK
      MOVLW 0X3C              ;ESCREVE COMANDO PARA
      CALL  ESCREVE           ; LIMPAR A TELA
      MOVLW .1
      CALL  DELAY_MS          ; DELAY DE 1MS
      MOVLW 0X84              ; COMANDO PARA POSICIONAR O CURSOR
      CALL  ESCREVE           ; LINHA 0 / COLUNA 1
```

```
      BSF   RS              ; SELECIONA O DISPLAY P/ DADOS
                            ; COMANDOS PARA ESCREVER
                            ; "MESA: 1"
      MOVLW A'M'
      CALL  ESCREVE
      MOVLW A'E'
      CALL  ESCREVE
      MOVLW A'S'
      CALL  ESCREVE
      MOVLW A'A'
      CALL  ESCREVE
      MOVLW A':'
      CALL  ESCREVE
      MOVLW A' '
      CALL  ESCREVE
      MOVLW A'1'
      CALL  ESCREVE
      MOVLW .250
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      CALL  LIMPA_TELA
      BCF   BACK
RETURN

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                      ESCRITA NO DISPLAY MESA: 2                       *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

MESA_2
      BCF   RS              ; SELECIONA DISPLAY PARA COMANDOS
      MOVLW 0X30            ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE         ; INICIALIZAÇÃO
      MOVLW .3
      CALL  DELAY_MS        ; DELAY DE 3MS (EXIGIDO PELO DISPLAY)
      MOVLW 0X30            ; ESCREVE COMANDO 0X30 PARA
      CALL  ESCREVE         ; INICIALIZAÇÃO
```

```
MOVLW 0X30              ; ESCREVE COMANDO 0X30 PARA
CALL  ESCREVE           ; INICIALIZAÇÃO
MOVLW B'00111100'       ; ESCREVE COMANDO PARA
CALL  ESCREVE           ; INTERFACE DE 8 VIAS DE DADOS
MOVLW B'00000001'       ; ESCREVE COMANDO PARA
CALL  ESCREVE           ; LIMPAR TODO O DISPLAY
MOVLW .1
CALL  DELAY_MS          ; DELAY DE 1MS
MOVLW B'00001110'       ; ESCREVE COMANDO PARA
CALL  ESCREVE           ; LIGAR O DISPLAY SEM CURSOR PISCANTE
MOVLW B'00000111'
CALL  ESCREVE
MOVLW B'00000110'       ; ESCREVE COMANDO PARA INCREM.
CALL  ESCREVE           ; AUTOMÁTICO À DIREITA
BSF   BACK
MOVLW 0X3C              ; ESCREVE COMANDO PARA
CALL  ESCREVE           ; LIMPAR A TELA
MOVLW .1
CALL  DELAY_MS          ; DELAY DE 1MS
MOVLW 0X84              ; COMANDO PARA POSICIONAR O CURSOR
CALL  ESCREVE           ; LINHA 0 / COLUNA 1
BSF   RS                ; SELECIONA O DISPLAY P/ DADOS
                        ; COMANDOS PARA ESCREVER
                        ; "MESA: 2"
MOVLW A'M'
CALL  ESCREVE
MOVLW A'E'
CALL  ESCREVE
MOVLW A'S'
CALL  ESCREVE
MOVLW A'A'
CALL  ESCREVE
MOVLW A':'
CALL  ESCREVE
MOVLW A' '
CALL  ESCREVE
MOVLW A'2'
CALL  ESCREVE
MOVLW .250
CALL  DELAY_MS
MOVLW .250
```

```
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      MOVLW .250
      CALL  DELAY_MS
      CALL  LIMPA_TELA
      BCF   BACK
RETURN
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                                LIMPA TELA                              *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

LIMPA_TELA
      BCF         RS              ; SELECIONA O DISPLAY P/ COMANDOS
      MOVLW B'00000001'           ; ESCREVE COMANDO PARA
      CALL  ESCREVE               ; LIMPAR TODO O DISPLAY
RETURN
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*                            FIM DO PROGRAMA                             *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      END
```

ANEXO – PIC 16F877A

O PIC 16F877A possui uma série de recursos que possibilita vários tipos de aplicações. Segue abaixo, uma lista com as suas características mais importantes.

1. Via de programação com 14 bits e 35 instruções;
2. 33 portas configuráveis como entrada ou saída;
3. 15 interrupções;
4. Memória de programação E^2PROM;
5. Memória de programa com 8Kwords;
6. Memória E^2PROM com 256 bytes;
7. Memória RAM com 368 bytes;
8. Três timers, sendo 2 de 8 bits e 1 de 16 bits;
9. Comunicações seriais SPI, I^2C e USART;
10. Conversores analógicos e comparadores analógicos;
11. Dois módulos CCP: Capture, Compare e PWM;
12. Programação in-circuit;
13. Power-on Reset interno (POR);
14. Brown-out Reset interno (BOR).

Uma vantagem que a família PIC apresenta é o fato de que todos os modelos possuírem um set de instruções muito parecido, o que facilita a mudança ou atualização de modelos. Outro ponto importante é que as características básicas também costumam ser mantidas, por exemplo, há uma preocupação em manter a pinagem, dos microcontroladores de mesmo nível, o mais idêntica possível, facilitando a troca por modelos mais novos, principalmente quando o circuito já está concebido.

Pinagem e sua descrição

A pinagem deste microcontrolador pode ser vista na Figura 1, onde são apresentados os 40 pinos e suas respectivas funções. Pode-se observar também,

que a maioria dos pinos possui mais de uma função, dependendo da configuração e do programa alguns pinos podem utilizar mais de uma função durante a operação.

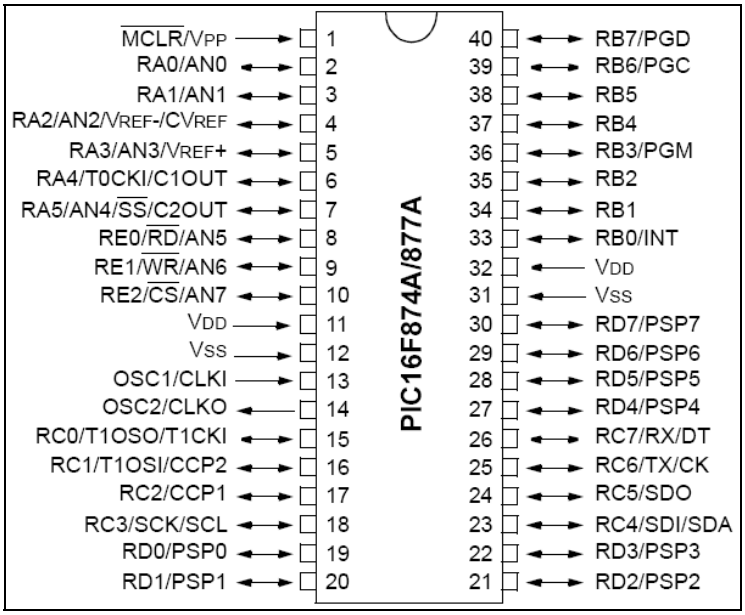


Figura 1 – Pinagem do PIC 16F877A

Logo a seguir, a Tabela 1, apresenta uma pena descrição sobre cada pino do PIC 16F877A para melhor compreensão de funções.

Tabela 1 – Nomenclatura dos pinos: PIC 16F877A

Pino	Nº Pino	Descrição
OSC1/CLKIN	13	Oscilador cristal ou entrada de osciladores externos
OSC2/CLKOUT	14	Saída para oscilador cristal
MCLR/Vpp	1	Master Clear (reset). O microcontrolador funciona quando este pino está em nível alto
RA1 AN0	2	Entrada e saída digital Entrada analógica 0
RA1 AN1	3	Entrada e saída digital Entrada analógica 1

RA2 AN2 V_{REF}-/CV_{REF}	4	Entrada e saída digital Entrada analógica 2 Tensão negativa de referência analógica
RA3 AN3 V_{REF+}	5	Entrada e saída digital Entrada analógica 3 Tensão positiva de referência analógica
RA4 T0CKI C1OUT	6	Entrada e saída digital. Open-drain quando configurado como saída Entrada externa do contador TMR0 Saída do comparador 1
RA5 AN4 SS C2OUT	7	Entrada e saída digital Entrada analógica 3 *Slave* para a comunicação SPI Saída do comparador 2
RB0 INT	33	Entrada e saída digital Interrupção externa
RB1	34	Entrada e saída digital
RB2	35	Entrada e saída digital
RB3 PGM	36	Entrada e saída digital Entrada para programação de baixa tensão
RB4	37	Entrada e saída digital
RB5	38	Entrada e saída digital
RB6 PGC	39	Entrada e saída digital Clock de programação serial ou pino de in-circuit debugger
RB7 PGD	40	Entrada e saída digital Dado de programação serial ou pino de in-circuit debugger

RC0 T1OSO T1CKI	15	Entrada e saída digital Saída do oscilador externo para TMR1 Entrada de incremento para TMR1
RC1 T1OSI CCP2	16	Entrada e saída digital Entrada do oscilador externo para TMR1 Entrada do Capture2 ou Saídas para Compare2/PWM2
RC2 CCP1	17	Entrada e saída digital Entrada do Capture1 ou Saídas para Compare1/PWM1
RC3 SCK SCL	18	Entrada e saída digital Entrada/Saída do clock para comunicação SPI Entrada/Saída do clock para comunicação I^2C
RC4 SDI SDA	23	Entrada e saída digital Entrada de dados para comunicação SPI Entrada/Saída de dados para comunicação I^2C
RC5 SDO	24	Entrada e saída digital Saída de dados para comunicação SPI
RC6 TX CK	25	Entrada e saída digital Transmissão para comunicação assíncrona USART Clock para comunicação síncrona USART
RC7 RX DT	26	Entrada e saída digital Recepção para comunicação assíncrona USART Dados para comunicação síncrona USART
RD0 PSP0	19	Entrada e saída digital Comunicação paralela dado 0
RD1 PSP1	20	Entrada e saída digital Comunicação paralela, dado 1

RD2 PSP2	21	Entrada e saída digital Comunicação paralela, dado 2
RD3 PSP3	22	Entrada e saída digital Comunicação paralela, dado 3
RD4 PSP4	27	Entrada e saída digital Comunicação paralela, dado 4
RD5 PSP5	28	Entrada e saída digital Comunicação paralela, dado 5
RD6 PSP6	29	Entrada e saída digital Comunicação paralela, dado 6
RD7 PSP7	30	Entrada e saída digital Comunicação paralela, dado 7
RE0 RD AN5	8	Entrada e saída digital Controle de leitura da comunicação paralela Entrada analógica
RE1 WR AN6	9	Entrada e saída digital Controle de escrita da comunicação paralela Entrada analógica
RE2 CS AN7	10	Entrada e saída digital Habilitação externa para comunicação paralela Entrada analógica
V_{SS}	12, 31	GND
V_{DD}	11, 32	Alimentação positiva

Ciclos de máquina

Na maioria dos modelos da linha PIC segue a Equação 1, onde o clock interno é equivale a ¼ do clock externo:

$$CK\,\mathrm{int} = \frac{CKext}{4} \qquad (1)$$

Assim, se o clock externo for de 4MHz, por exemplo, o clock interno será de 1MHz. Dessa forma, a duração de um ciclo de máquina (CM ou Tcy) será de 1µs. Veja Equação 2.

$$Tcy = \frac{1}{CK\,\mathrm{int}} \qquad (2)$$

O clock é dividido devido ao funcionamento interno do processador. São necessárias várias operações para que uma instrução seja executada. Essas operações são executadas em cada subciclo de máquina (Q1, Q2, Q3 e Q4).

O contador de programa, que aponta para a próxima instrução a ser executada, é incrementado sempre no início de Q1, a instrução será executada durante o período de Q1 a Q4. Isso é suficiente para que a ULA troque informações com a memória de dados e o registrador W quando necessário. Ao final do período Q4 a próxima instrução é buscada na memória.

Todo esse processo também é conhecido como Pipeline, e, considerando o clock interno de 4 MHz, quase todas as instruções podem ser executadas em apenas 1µs. Na Figura 2 é ilustrada esse conceito.

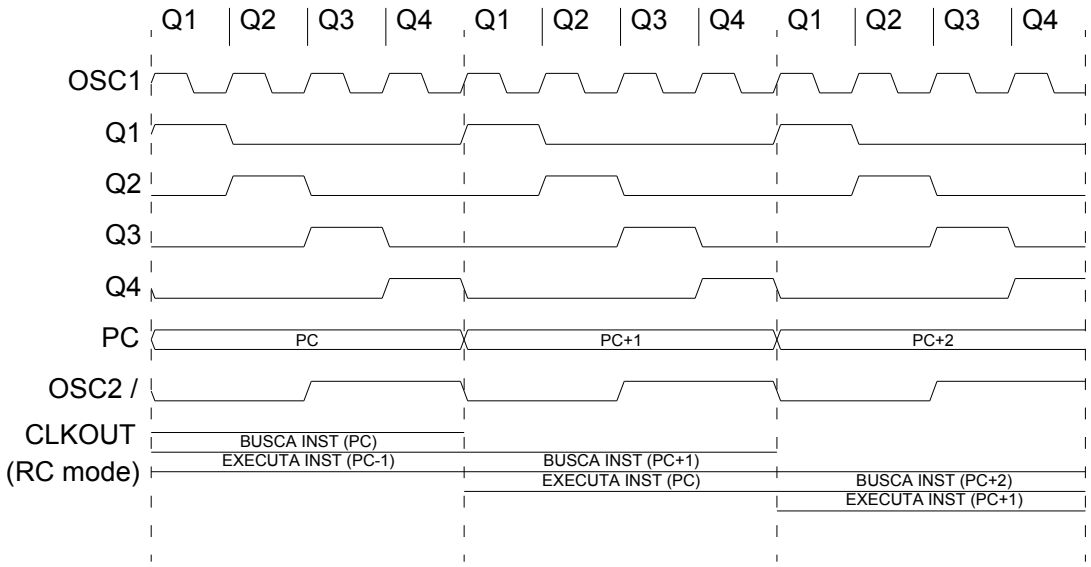


Figura 2 – Clock externo/clock interno no PIC 16F877A

Diagrama de Blocos do PIC 16F877A

No diagrama de blocos do PIC 16F877A mostra o funcionamento interno do microcontrolador. Este diagrama mostra como está posicionado a ULA, os registradores, as memórias e os periféricos. Pode-se verificar, pela Figura 3, que a ULA está próxima ao centro, ela é ligada diretamente ao registrador W e ao barramento de dados de 8 bits. Logo, acima da ULA está o registrador Status. Na parte superior, encontram-se as memórias de programa (FLASH) e de dados (RAM), já a memória de dados não-volátil (EEPROM) está localizada no canto inferior esquerdo. Do lado direito estão os PORTs, de PORTA a PORTE, com a indicação das funções de cada pino. Os demais periféricos estão na parte inferior, inclusive a interface de comunicação serial SPI/I^2C, os conversores Analógicos/Digitais (A/D) e os módulos CCP (Compare, Capture e PWM).

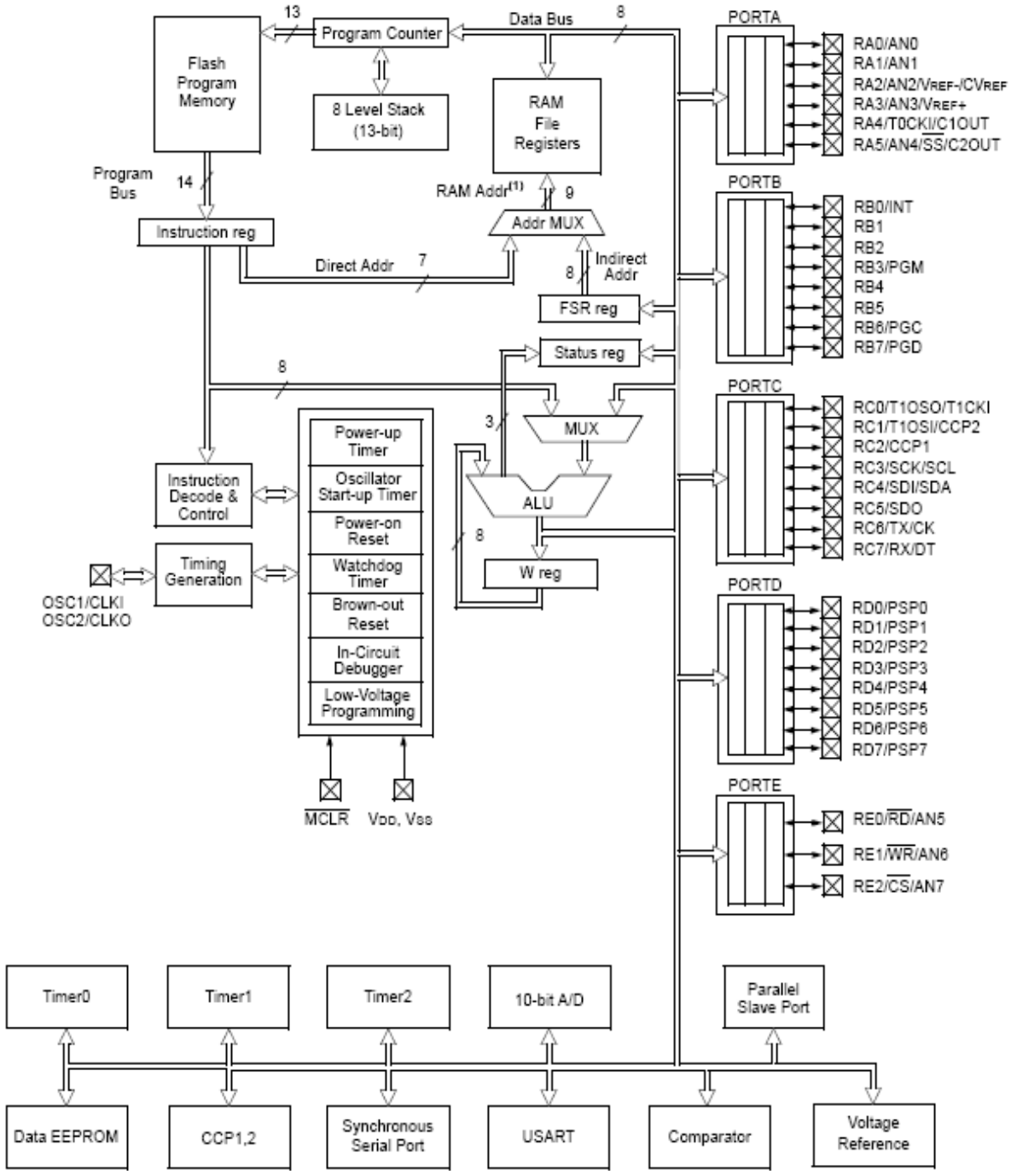


Figura 3 – Diagrama de blocos do PIC 16F877A

Memória

Como visto no diagrama de blocos, esse microcontrolador possui três memórias independentes, memória de programa, chamada de FLASH, a memória

de dados, mais conhecida como memória RAM e memória de dados não volátil EEPROM.

A memória de programa serve para armazenar o programa que vai controlar as funcionalidades do microcontrolador. Ela é do tipo FLASH com 14 bits, pode ser gravada e regravada eletronicamente, sendo sua escrita bastante rápida. No PIC 16F877A a memória de programa possui 14,3Kbytes.

A memória RAM é a memória que o microcontrolador usa para guardar informações ou variáveis durante a execução do programa, quando o microcontrolador é desligado ou reiniciado, os dados nessa memória são perdidos. É nesta memória de dados que se encontram os registradores de funções especiais e os registradores de uso geral. Esta memória do PIC 16F877A possui 368 bytes.

Os registradores de funções especiais servem para a configuração de várias funções e são utilizados por todos os periféricos. Esses registradores compõem uma região da memória RAM e podem ser lidos e escritos pelo usuário e também pelo hardware.

Os registradores de uso geral servem para o armazenamento de variáveis definidas pelo usuário e que devem ser lidas ou escritas pelo programa. Esses registradores compõem os 368 bytes de memória RAM.

Para facilitar acesso aos registradores, a Microchip dividiu a memória de dados em quatro partes chamadas de bank0, bank1, bank2 e bank3. Cada registrador está localizado em um dos bancos acima citado com os seus respectivos endereços de memória. Para utilizar esses registradores na programação do microcontrolador, deve inicialmente selecionar o banco ao qual pertença. Na Figura 4, mostra a divisão dos bancos e o posicionamento de cada registrador para melhor visualização.

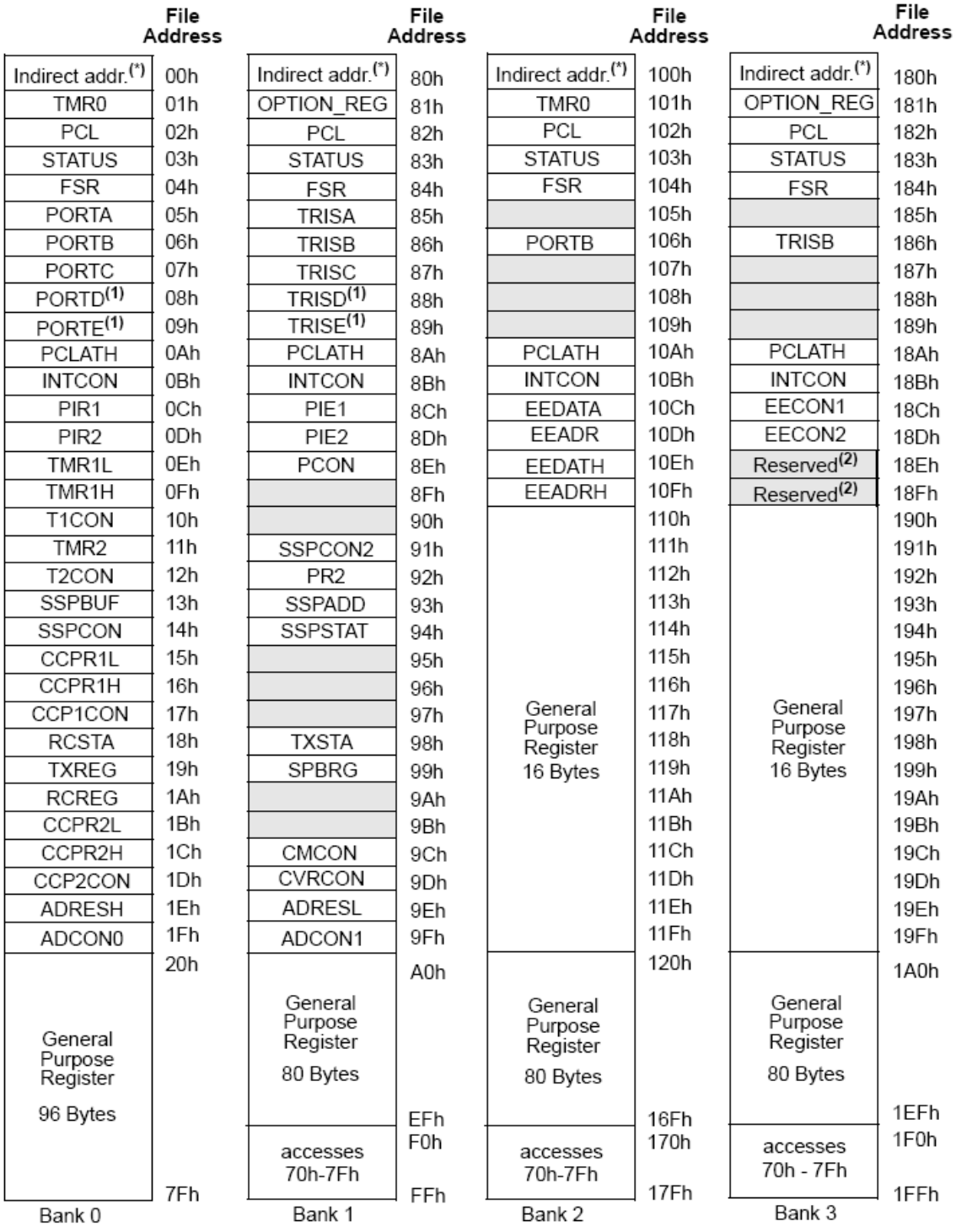


Figura 4 – Mapa da memória de dados

A memória EEPROM funciona exatamente como a memória RAM, com a diferença que os dados armazenados nessa memória não se perdem quando o microcontrolador é desligado ou reiniciado. A capacidade da memória EEPROM do PIC 16F877A é de 256 bytes.

Registradores

Os registradores são os responsáveis pela execução do programa. Cada registrador possui uma função definida quer seja para selecionar um banco, habilitar interrupções, selecionar pinos como entrada ou saída, selecionar portas analógicas ou digitais, selecionar tipo de comunicação, entre outras. Entretanto será abordados os registradores usados neste projeto.

Registrador Status

Este registrador possui as funções aritméticas da ULA, reset de estados e é através deste registrador que é possível selecionar o banco para memória de dados. Veja na Figura 5 o posicionamentos dos 8 bits do registrador Status.

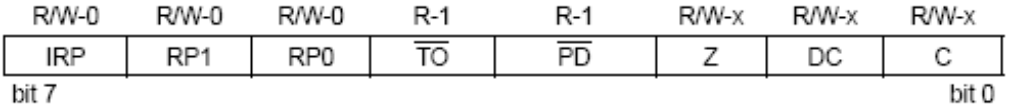


Figura 5 – Registrador Status

Bit 7 IRP (Register Bank Selector): Seletor de banco de memória de dado por endereçamento indireto

1 = Bank 2, 3

0 = Bank 0,1

Bit 6-5 RP1:RP0 (Register Bank Selector): Seletor de banco de memória de dado por endereçamento indireto

11 = Bank 3

10 = Bank 2

01 = Bank 1

00 = Bank 0

Bit 4 /TO: Timer-out

1 = Depois do power-up ou foram executadas instruções CLRWDT ou SLEEP

0 = Estouro do WatchDog Timer (WDT)

Bit 3 /PD: Power-Down

1 = Ocorreu um power-up ou CLRWDT

0 = Ocorreu SLEEP

Bit 2 Z: bit Zero

1 = Resultado aritmético igual a zero

0 = Resultado aritmético diferente de zero

Bit 1 DC: Digital Carry/borrow

1 = Ocorreu estouro (carry) na operação da ULA, ultrapassou os 4 bits menos significativos

0 = Não ocasionou em estouro

Bit 0 C: Carry/borrow

1 = Ocorreu estouro (carry) na operação da ULA, ultrapassou dos bits mais significativos

0 = Não ocasionou em estouro

Registrador OPTION_REG

O registrador OPTION_REG configura os *pull-ups* da PORTB, a interrupção externa através do pino RA4, possui vários controles de bits para a interrupção TMR0 e configura a aplicação do prescaler. Esta configuração do prescaler é necessária para que o programa não trave quer seja por uma falha ou quer seja por um loop infinito. O temporizador WatchDog Timer (WDT) é o temporizador

responsável pelo reset do microcontrolador de acordo com a configuração dos bits do registrador OPTION_REG<3:0>.

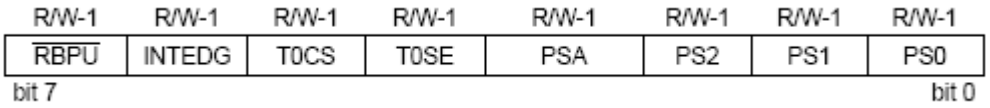


Figura 6 – Registrador Status

PSA Configura a aplicação do prescaler

1 = Prescaler será aplicado em WDT

0 = Prescaler será aplicado em TMR0

PS2:PS0 Configuração do valor do prescaler para WDT

011 1:8

110 1:16

101 1:32

110 1:64

111 1:128

Os três últimos bits PS2, PS1 e PS0 definem quantos ciclos de máquina serão necessário para que haja estouro do WDT. Por exemplo, configurados os bits como 111 significa que após 128 ciclos de máquinas haverá o estouro do WDT.

Registradores TRIS e PORT

Estes registradores TRIS são responsáveis por configurar os pinos correspondentes aos registradores PORT do PIC 16F877A como entrada e saída. Eles estarão configurados como entrada quando seus bits forem iguais a 1 e estarão configurados como saída quando seus bits estiverem em 0. Existem 5 tipos de registradores TRIS que são TRISA, TRISB, TRISC, TRISD e TRISCE que

correspondem a 5 tipos de registradores PORT que são respectivamente PORTA com 6 bits, PORTB com 8bits, PORTC com 8 bits, PORTD com 8 bits e PORTE com 3bits.

Na PORTD estão os 8 bits para a comunicação paralela do PIC 16F877A, por exemplo a implementação de um display seria através destes pinos. Porém o controle de envio de dados, leitura e escrita é através da PORTE onde estão os pinos RS(Register Select) para comando ou dado e o E (Enable) que habilita a escrita da no display.

Registrador INTCON

Este registrador habilita as interrupções de TMR0. Não usados neste trabalho, porém é necessário configurar como desabilitadas.

Registradores ADCON0 e ADCON1

Para a conversão de entrada analógica ou digital é necessário configurar estes registradores. No registrador ADCON0, é possível selecionar o canal a ser utilizado e habilitar a conversão analógica / digital. Já no registrador ADCON1 é possível configurar os pinos analógico ou digital. Veja, na Figura 7, como é feita a configuração pinos através dos bits ADCON1<PCFG3:PCFG0>.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A
0001	A	A	A	A	VREF+	A	A	A
0010	D	D	D	A	A	A	A	A
0011	D	D	D	A	VREF+	A	A	A
0100	D	D	D	D	A	D	A	A
0101	D	D	D	D	VREF+	D	A	A
011x	D	D	D	D	D	D	D	D
1000	A	A	A	A	VREF+	VREF-	A	A
1001	D	D	A	A	A	A	A	A
1010	D	D	A	A	VREF+	A	A	A
1011	D	D	A	A	VREF+	VREF-	A	A
1100	D	D	D	A	VREF+	VREF-	A	A
1101	D	D	D	D	VREF+	VREF-	A	A
1110	D	D	D	D	D	D	D	A
1111	D	D	D	D	VREF+	VREF-	D	A

Figura 7 – Configuração A/D dos pinos

Interrupções

As interrupções geram desvio do programa para o mesmo vetor de interrupção. Essas interrupções podem ser divididas em convencionais e de periféricos. As convencionais são: Timer 0, Externa, Mudança de estado. Já as de periféricos existem 12 tipos de interrupções entre elas: Porta paralela, Conversores A/D, Transmissão USART, Recepção USART, Comunicação serial do tipo SPI e I^2C.

Características elétricas

As características elétricas são extremamente importantes para melhor dimensionamento do projeto. Nas Tabelas 2, 3 e 4 estão as especificações elétricas do PIC 16F877A.

Tabela 2 – Característica de Temperatura

Situação	**Temperatura em ºC**
Temperatura de trabalho	-55ºC a +125ºC
Temperatura de armazenamento	-65ºC a +150ºC

Tabela 3 – Característica de Tensão

Pinos	**Tensão máxima (V)**
Trabalho	4.0 a 5.5
V_{DD}	-0.3 a 7.5
MCRL	0 a 14
RA4	0 a 8.5
Demais pinos	-0.3 a V_{DD} + 0.3

Tabela 4 – Característica de corrente nos pinos

Pinos	**Corrente máxima (mA)**
V_{SS} saída	300
V_{DD} entrada	250
Entrada de pino quando V_{SS}	25
Saída de pino quando V_{DD}	25
Entrada de PORTA, PORTB e PORTE combinados	200
Saída de PORTA, PORTB e PORTE combinados	200
Entrada de PORTC e PORTD combinados	200
Saída de PORTC e PORTD combinados	200