UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Projeto de Formatura

Iluminação Pública Alimentada por Energia Solar

Orientador: Prof. Dr. Marcelo Godoy Simões

Eric Brandt Schonwald Mecatrônica

São Paulo, 18 de dezembro de 1998

9,5 (nove e cinco)

"O acaso compõe o melhor da sinfonia, pinta o melhor do quadro, constrói o mais difícil do monumento, inventa o instante da paixão e escolhe papel e barbante do pacote."
(Millôr Fernandes)

"Os homens são como tapetes. Às vezes precisam ser sacudidos"
(provérbio árabe)

"Deus fez o mundo em seis dias. No sétimo o diabo assumiu, e o exilou como subversivo"
(Millôr Fernandes)

Eric,

A ciência é como uma lâmpada: cada vez que se acende, ilumina a estrada humana em busca do progresso.

Abençoados são os homens que trazem consigo essa centelha divina, capazes de alavancar o mundo com o suor do seu trabalho.

Fico feliz por encontrar essa luz em você, meu amigo.

Alexandre Purcino Nogueira
São Paulo, 14 de dezembro de 1998

Passa uma borboleta por diante de mim
E pela primeira vez no Universo eu reparo
Que as borboletas não têm cor nem movimento,
Assim como as flores não têm perfume nem cor.
A cor é que tem cor nas asas da borboleta,
No movimento da borboleta o movimento é que se move,
O perfume é que tem perfume no perfume da flor.
A borboleta é apenas borboleta
E a flor é apenas flor.

Alberto Caeiro
(pseudônimo de Fernando Pessoa)

AGRADECIMENTOS

Como este é o primeiro livro que escrevo, aproveitarei a chance para me estender um pouco. Desta forma, meus agradecimentos:

Ao Prof. Dr. Marcelo Godoy Simões por ter sido meu orientador neste trabalho, bem como por ter me acompanhado por mais de dois anos pela Poli como orientador de iniciação científica e conselheiro sobre assuntos extra-acadêmicos.

A Nilson Noris Franceschetti, que muito me ajudou neste trabalho e sem quem não teria sido possível a sua conclusão.

Aos meus pais, que com seu esforço me permitiram chegar até aqui.

A Mary e Albert Dichi, que muito me ajudaram e apoiaram ao longo de minha vida, em todos os sentidos.

Aos meus tios Enrique, Fábio, Maria e Yeda e primos, que são a minha família.

Aos amigos, que sempre estiveram dispostos a me ouvir, me deram conselhos, contaram piadas, falaram besteiras e todos sempre demos boas risadas.

Ao Ariel, meu irmão, por ser um amigo com quem sempre posso contar.

E, não sejamos ingratos, meus agradecimentos a Alessandra Cristina Monteiro de Castro Trigo, futura arqueóloga, que entre milhares de outras coisas, me ensinou que na vida, às vezes é necessário deixar a razão de lado e agir com o coração.

A todos vocês, meus sinceros votos de sucesso e saúde.

Gostaria também de agradecer aos produtores de café por me ajudarem a chegar (acordado!) até o final deste trabalho e ao criador de Os Simpsons pelos momentos de distração.

Eric Brandt Schonwald
Dezembro de 1998

Índice

Lista de figuras

Lista de tabelas

Iluminação Pública Alimentada por Energia Solar

1. Introdução

Este trabalho visa o projeto eletrônico e implementação de um poste de iluminação pública a lâmpada fluorescente, totalmente independente da rede elétrica convencional. A sua fonte de energia será a energia solar, que durante o dia alimentará uma bateria automotiva. Além disso, serão programados os horários de acendimento e desligamento da lâmpada, através de comunicação serial entre o *hardware* do poste e um microcomputador. Doravante o projeto será chamado de **Poste Solar**.

2. Os módulos do projeto

Basicamente, o projeto pode ser dividido em quatro grandes módulos, a saber:

- *Hardware* e *software* que visam o armazenamento da energia solar, captada por células fotovoltaicas, em baterias;
- Desenvolvimento de circuito inversor (comercialmente chamado de reator) para acionamento eficiente da lâmpada fluorescente;
- Implementação de comunicação serial entre o *hardware* do poste e um microcomputador;
- Implementação de relógio de tempo real, com a finalidade de se determinar horários de ligamento e desligamento da lâmpada.

O "coração" do projeto é o microcontrolador PIC16C74, da **Microchip**, que está apresentado adiante neste trabalho.

A figura abaixo apresenta uma visão geral dos módulos do projeto.

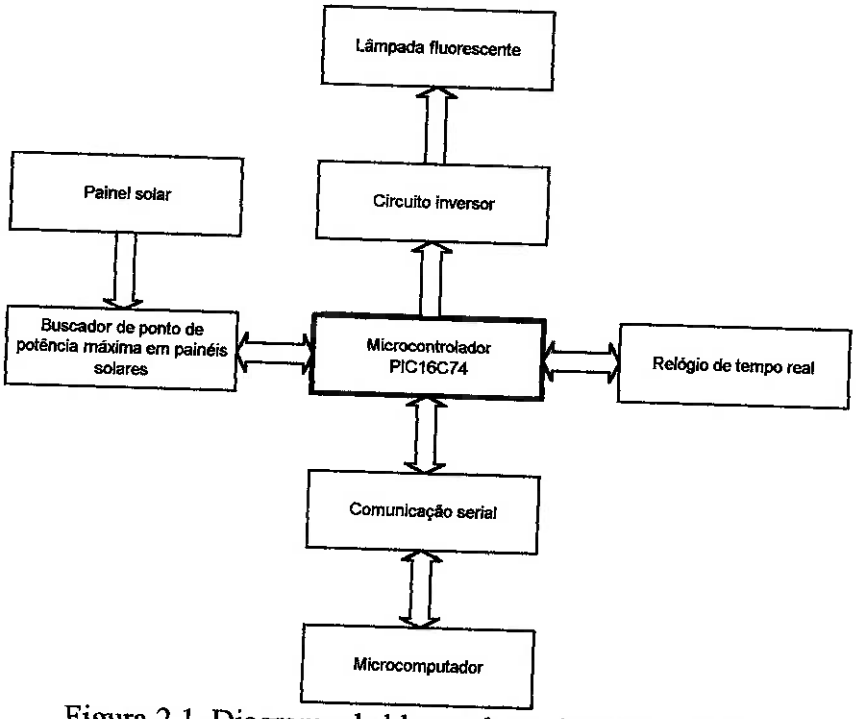


Figura 2.1. Diagrama de blocos do projeto Poste Solar.

3. O microcontrolador PIC16C74

Os microcontroladores da família PIC16C7X, da **Microchip**, são, conforme a definição do fabricante, **Microcontroladores CMOS de 8-bits com conversor analógico**. Esta família é composta pelos microcontrolador PIC16C71, PIC16C73 e PIC16C74, sendo este último o utilizado no projeto.

Esta família de microcontroladores RISC apresenta:

- apenas 35 instruções de programação;
- todas as instruções são de um ciclo, exceto para desvios no fluxo do programa, que são de dois ciclos;
- o clock do processador é ¼ da freqüência do cristal (que pode ser no máximo de 20MHz), ou seja, é possível se obter ciclos de instrução de 200ns;
- apresenta 11 tipos de interrupções;
- oito níveis de profundidade no *stack*;
- acesso direto, indireto e relativo;

- conversão A/D tendo como tensão de referência +5V, ou outra tensão em um pino específico;
- EPROM apagável por ultravioleta.

O microcontrolador utilizado neste projeto é o PIC16C74 (que apresenta 40 pinos), o maior da família, com as seguintes propriedades:

- dois pinos podem ser configurados como entrada de captura, saída de PWM (modulação da largura de pulso), ou saída de comparação;
- 4Kb de memória;
- três timers;
- interface de comunicação serial (SCI/USART);
- porta serial síncrona (SSP) com SPI e I^2C/ACCESS bus;
- watchdog timer;
- economia de energia no modo SLEEP;
- 33 pinos de I/O, distribuídos por 5 *ports*;
- 8 canais de A/D (entrada analógica).

Escolheu-se o microcontrolador PIC16C74 pois:

- Possui as funções necessárias para este projeto, não necessitando do uso de outros dispositivos. Entre elas, saída de PWM, três *timers* independentes com interrupção, comunicação serial e conversão A/D;
- É barato (cerca de US$ 15, dependendo da quantidade) e com representante comercial no Brasil;
- Possui alta velocidade de execução e grande confiabilidade.

Para este projeto será utilizado um cristal de 4MHz, o que implica em um **clock de 1MHz**, e **ciclos de 1μs**.

A programação do PIC se dá em Assembler RISC MPASM, com 35 instruções, e a compilação se dá através do *"MPASM Assembler"* da Microchip, além disso o fabricante também fornece o *"MPSIM Software Simulator"* no qual é possível simular a execução do programa. A gravação da EPROM se dá através do *"PRO-MATE Universal Programmer"*. Para se apagar a EPROM basta deixar a "janela" do *chip* exposta à luz ultravioleta por cerca de 15 minutos.

4. Lâmpada fluorescente – Circuito inversor

A fim de se otimizar o uso da energia solar acumulada nas baterias, utilizou-se um sistema eletrônico para a alimentação das lâmpadas fluorescentes, visto que este, segundo POSADAS em sua tese de mestrado, oferece um aumento global de eficiência de cerca de 30% sobre os sistemas convencionais utilizando reatores eletromagnéticos.

O projeto do inversor da lâmpada fluorescente seguiu o roteiro proposto por POSADAS. No Anexo B encontra-se o princípio de funcionamento da lâmpada fluorescente, do inversor push-pull utilizado, bem como o equacionamento para a determinação de seus componentes.

4.1. Cálculo do inversor da lâmpada fluorescente

Este capítulo segue o método proposto no Anexo B.

Para a tensão de operação utilizou-se uma bateria de carro, portanto $\mathbf{V_{cc}=12V}$.

A lâmpada utilizada é do modelo "TLT 20W/75 RS – Extra Luz do Dia" da Philips, com potência nominal de 20W. Como a proposta do circuito inversor utilizado é manter o mesmo fluxo luminoso para uma potência menor, economizando energia, adotou-se uma potência de cerca de 83% da nominal, desta forma, **P=16,6W**.

A resistência da lâmpada em regime foi determinada ligando-se a mesma à rede elétrica. Com o uso de um VARIAC, determinou-se a resistência da mesma quando ela

estivesse consumindo exatamente 20W. Obteve-se V_{60Hz}=59,4V e I_{60Hz}=0,377A.Desta forma a resistência da lâmpada é de **R=176,3Ω**.

A freqüência de operação adotada deve ser acima de 20kHz, a fim de estar fora do campo audível humano, mantendo desta forma uma operação silenciosa. Adotou-se **f_d=25kHz**.

Em relação ao fator de crista, o mesmo deve se encontrar entre 1,4 e 1,7. Adotou-se **F_c=1,55**.

Além disto, utilizou-se tensão de pico de $7{,}5 \cdot \sqrt{2}$ V nos filamentos da lâmpada e tensão de 450V em vazio para o enrolamento N_4, seguindo o mesmo critério adotado por POSADAS, que realizou medidas a fim de determinar estes valores.

Abaixo estão apresentados todos os valores obtidos no cálculo do inversor.

$V_{Np} = 18{,}85V$

$\frac{N_1}{N_5} = 1{,}77 \approx 2$

$\frac{N_4}{N_1} = 23{,}87 \approx 24$

$r_{eq} = 10{,}7\Omega$

$r = 0{,}316\Omega$

$Q = 5{,}732$

$c_e = 3{,}51 \cdot 10^{-6}$ F

$C_e = 6{,}135 \cdot 10^{-9}$ F

$c_{ep} = 3{,}4 \cdot 10^{-6}$ F

$c = 1{,}3795 \cdot 10^{-6}$ F

$C = 3{,}449 \cdot 10^{-7}$ F

$c_{eq} = 4{,}78 \cdot 10^{-6}$ F

$L = 8{,}446 \cdot 10^{-6}$ H

O transistor utilizado foi o TIP41C, com h_{FE} = 125. Desta forma:

$I = 1{,}38A$

$I_b = 0{,}03 \cdot I = 41{,}4 \cdot 10^{-3}$ A

$L_f = 1{,}22 \cdot 10^{-3}$ H

Adotando-se V_{ebmax} = 6V e sendo a tensão na junção base-emissor V_j = 0,6V, então:

$V_{N3p} = 6,6V$

$\frac{N_1}{N_3} = 2,85 \approx 3$

$R_3 = 558\Omega$

$R_1 = R_2 = 8,37\Omega$

A fim de se utilizar valores comerciais de componentes, realizou-se aproximações em seus valores, a saber:

$C = 330nF$

$C_e = 6,8nF$

$R_3 = 560\Omega$

$R_1 = R_2 = 8,2\Omega$

O indutor L_f e o transformador tiveram seu projeto realizado pela **Tecnotrafo**, cujos dados se encontram no Anexo A. As características do transformador são:

$\frac{N_1}{N_5} = 2$

$\frac{N_4}{N_1} = 24$

$\frac{N_1}{N_3} = 3$

$N_1 = N_2$

$N_5 = N_6$

$I = 1,38A$

$L_f = 8,45\mu H$

A figura abaixo apresenta o circuito do inversor, com os seus valores.

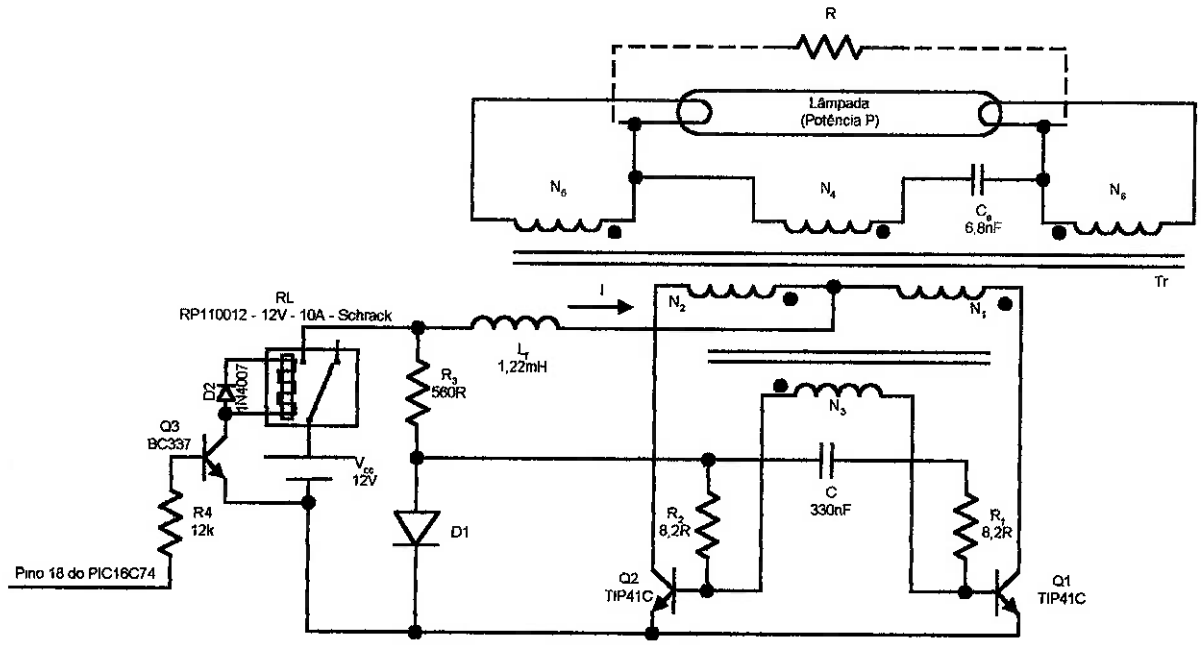


Figura 4.1. Circuito inversor utilizado.

O acendimento da lâmpada ocorre através de um sinal vindo do pino 18 do PIC16C74. Enquanto este sinal estiver em nível *low* a lâmpada se encontra desligada. Quando o sinal for para nível *high*, a lâmpada se acende. Esta é a função do relê RL no circuito.

5. Relógio de tempo real

Com a tecnologia existente atualmente, seria necessário um painel muito grande e caro para manter a lâmpada acesa durante a noite toda. Desta maneira, a fim de se determinar o horário de acendimento e desligamento da lâmpada fluorescente, é necessário se implementar um relógio de tempo real.

Poderia ter sido utilizado o próprio painel solar como detetor de dia/noite. Entretanto, dois fatores foram determinantes para a exclusão desta possibilidade. O primeiro é porque o uso de um detetor de luz faria com que a lâmpada se acendesse durante o dia, caso o mesmo estivesse nublado. O outro é que utilizando-se um relógio, é possível escolher quando e por quanto tempo a lâmpada se manterá acesa.

O microcontrolador PIC16C74 apresenta três *timers*, denominados de TMR0, TMR1 e TMR2.

Para se implementar o relógio de tempo real, utilizou-se o TMR1. Este *timer* apresenta a característica de poder apresentar operação assíncrona em relação ao microcontrolador, desta maneira não atrapalhando a função principal do mesmo, que é a busca do ponto de potência máxima do painel solar.

Para se utilizar o TMR1 em modo assíncrono é necessário o uso de um *clock* ou cristal externo, conectado aos pinos correspondentes. O TMR1 apresenta um contador de 16-bits. Quando ocorrer *overflow* do TMR1 pode ser habilitada a interrupção correspondente. Este contador apresenta *prescaler*[1] de 1, 2, 4 e 8.

O princípio de funcionamento do relógio é que a cada x segundos, ocorra uma interrupção no TMR1. A rotina de tratamento desta interrupção soma 1 a um contador implementado no programa. Desta maneira este contador seria um registro de múltiplos de x segundos. Toda vez que der meia-noite este contador zera.

Um dia tem 24 horas x 60 minutos x 60 segundos = 86400 segundos. A tabela abaixo apresenta a relação entre o intervalo x em que ocorre a interrupção de TMR1 e o valor que o contador terá à meia-noite.

x (segundos)	Contador à meia-noite (decimal)	Contador à meia-noite (hexadecimal)	Número de bits necessários para o contador
0,5	172800	2A300	18
1	86400	15180	17
2	43200	A8C0	16
4	21600	5460	15
8	10800	2A30	14

Tabela 5.1. Relação entre o intervalo de interrupção do TMR1 (x) e o número de bits do contador.

[1] *Prescaler* de 8 significa que a cada 8 bordas de subida no pino de entrada do cristal, será incrementado 1 no contador do mesmo.

Como o microcontrolador PIC16C74 trabalha com registradores de 8 bits, é interessante utilizar um contador que trabalhe com até 16 bits pois acima disto começa a se tornar complicado trabalhar com mais de dois registradores.

Como o relógio não necessita de uma precisão muito elevada, bem como para não sobrecarregar o microcontrolador com as interrupções do relógio, considerou-se que um *overflow* a cada 2 segundos seria suficiente, pois desta maneira também não se perderia a precisão do relógio após um grande intervalo de tempo. Desta forma o dia vai de 0000h até A8BFh.

Utilizando-se um cristal de 32.768 kHz, e um *prescaler* de 1, o TMR1 apresenta um *overflow* exatamente a cada 2 segundos. Isto ocorre porque o TMR1 apresenta uma interrupção a cada 2^{16} bordas de subida. Como 32768 é igual a 2^{15} e corresponde a 1s, então a cada $2 \cdot 2^{15}$, isto é, a cada 2 s, ocorre a interrupção.

O cristal utilizado é o modelo R38-32.768 kHz da **Raltron**. Este cristal apresenta uma variação de até 20 PPM (pulsos por milhão) em seu funcionamento. Desta forma, a cada um milhão de pulsos (que correspondem a 30,518 segundos), ele se atrasa (ou adianta) em até no máximo 20 pulsos ($6{,}1 \cdot 10^{-4}$ segundos). A tabela abaixo apresenta o erro máximo do cristal após vários períodos de tempo.

Período	Atraso (ou adiantamento) máximo do relógio (s)
1 minuto	$1{,}2 \cdot 10^{-3}$
1 hora	$7{,}2 \cdot 10^{-2}$
1 dia	1,73
1 mês	51,8
1 ano	621,7 s = 10 min

Tabela 5.2. Relação entre o período de tempo de funcionamento do relógio e o atraso (ou adiantamento) máximo do mesmo.

Pela tabela acima observa-se que o relógio atrasa até 51,8 s em um mês, correspondendo a 10 min em um ano. Considerando que o horário de funcionamento do poste não é algo que necessite precisão, é aceitável este erro, visto que o poste, devido

tanto às intempéries, como à lâmpada fluorescente e ao método de obtenção de energia através da luz solar, necessitará de uma manutenção que com certeza ocorrerá em períodos menores que um ano.

Basicamente, o *hardware* específico para a implementação do relógio é o cristal X2 e os capacitores C7 e C8, que podem ser observados no esquema do Anexo C.

A programação do módulo do relógio está detalhadamente descrita no Anexo D, que contém o programa final, englobando todos os módulos do projeto.

6. Comunicação serial com microcomputador

Para esta etapa devem ser desenvolvidos o *software* tanto do lado do microcontrolador PIC16C74 como do lado de um microcomputador PC.

As informações que devem ser trocadas são os horários de acendimento e desligamento da lâmpada, bem como o horário atual. Deve-se lembrar que todos os horários foram implementados em 2 bytes.

6.1. Comunicação serial do lado do microcomputador

O programa do lado do microcomputador foi desenvolvido em C, dada a sua facilidade em se manipular a porta serial do computador. Este programa será sobre sistema operacional DOS, pois como a manutenção do Poste Solar é feita em campo, quanto mais simples e barato for o computador utilizado, melhor.

Existem duas formas de se manipular a porta serial de um computador da família PC:

- através do *chip* 8250 UART, presente nos computadores (atualmente é utilizado uma versão mais nova do mesmo, o 16450 ou 16550), ou
- através das rotinas da ROM BIOS.

O segundo método é mais simples, mas também apresenta maiores limitações, pois permite velocidades até 9600 bauds e apenas o uso da COM1 ou COM2.

Entretanto, como a quantidade de dados transmitida é baixa, não há problema em se utilizar o segundo método. Além disto, a fim de se prevenir contra ruídos, recomenda-se velocidades baixas. Desta forma a velocidade de comunicação adotada será 1200 bauds, que é uma velocidade possível de ser utilizada com o PIC16C74 quando utilizando um cristal de 4MHz. Devido às características do microcontrolador utilizado, adota-se que:

A transmissão será em 1200 bauds por segundo, 8 bits, 1 start bit, 1 stop bit, sem paridade e sem *handshaking*.

Deve-se lembrar que o protocolo de comunicação serial dos microcomputadores da família PC é o RS-232C.

As características do programa que realiza a comunicação com o PIC são:

1. Leitura dos dados do PIC

Através do menu principal, escolhe-se a rotina que lê automaticamente os seis bytes que representam os 2 bytes do horário atual, o horário de ligamento e o horário de desligamento da lâmpada. Todos estes horários são dados em 16 bits, conforme se explicou no capítulo referente à implementação do relógio.

2. Envio dos dados ao PIC

Através do menu principal, escolhe-se a rotina que envia automaticamente os seis bytes já descritos acima. Em relação ao horário atual, pode-se utilizar automaticamente o horário do relógio do computador ou se inserir um outro horário qualquer. Após o envio é realizada uma leitura, a fim de se verificar se o PIC recebeu corretamente os dados enviados.

No Anexo E encontra-se o programa implementado do lado do computador, com todas as devidas explicações.

6.2. Comunicação serial do lado do PIC16C74

O microcontrolador utilizado apresenta dois módulos de comunicação serial, o SCI (*Serial Communication Interface*) e o SSP (*Synchronous Serial Port*).

Para comunicação com microcomputadores recomenda-se o uso do módulo SCI. Este módulo pode ser programado de maneira síncrona ou assíncrona. Será utilizada a comunicação assíncrona.

A comunicação assíncrona pode ser realizada em baixa velocidade e alta velocidade (dependendo do oscilador utilizado, esta pode chegar à velocidade de 1,25 Mbauds). Para se determinar a velocidade de comunicação utilizada, deve-se fornecer um valor X para o *Baud Rate Register* (SPBRG). As fórmulas para se determinar a velocidade da comunicação são:

Velocidade baixa	Velocidade alta
Baud rate = $F_{osc}/(64(X+1))$	Baud Rate = $F_{osc}/(16(X+1))$

Tabela 6.1. Fórmulas para determinação do valor X, para comunicação assíncrona.

Conforme foi comentado no item 6.1, a velocidade de comunicação será 1200 bauds. A freqüência do oscilador utilizado é F_{osc} = 4MHz. Desta maneira, a tabela abaixo apresenta o valor X encontrado, o valor arredondado utilizado, o valor real da velocidade e o erro em relação ao desejado.

	Velocidade baixa	Velocidade alta
Valor X encontrado	51,08	207,3
Valor X utilizado	51	207
Velocidade real (bauds)	1201,9	1201,9
Erro em relação a 2400 bauds (%)	0,16	0,16

Tabela 6.2. Valores a serem considerados na implementação da comunicação serial.

Pela tabela acima observa-se que para ambas as velocidades, o erro é o mesmo. então escolhe-se a velocidade alta pois nesta opção o microcontrolador realiza três leituras para cada bit, considerando o valor correto o valor que aparecer com maior freqüência. Resumindo, adota-se a opção de velocidade alta, com **X=207**.

O microcontrolador utilizado opera com nível TTL, ou seja, 0 e +5V. O padrão RS-232C, que é o protocolo de comunicação dos microcomputadores da família PC têm seus níveis lógicos em até –15 e +15V. Desta maneira, para "casar" os níveis lógicos das duas pontas da comunicação será utilizado o *chip* MAX232-N, que realiza automaticamente esta conversão. A figura abaixo apresenta a pinagem deste componente.

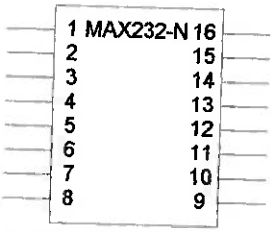


Figura 6.1. MAX 232

O MAX-232N, da *Texas Instruments*, possui dois "compatibilizadores" independentes de nível de tensão RS-232C para TTL, sendo denominados de canal 1 e 2. A fim de se compatibilizar os níveis TTL e RS-232C, o componente utiliza dois capacitores eletrolíticos, recomendando-se algo em torno de 1μF.

Pinos	Nome	Descrição
1/4	C_{1+} / C_{2+}	Pólo positivo dos capacitores eletrolíticos
2	V_{S+}	Saída positiva de tensão $V_{s+} = 2V_{cc}$-1.5 V
3/5	C_{1-} / C_{2-}	Pólo negativo dos capacitores eletrolíticos
6	V_{S-}	Saída negativa de tensão $V_{s-} = -2V_{cc}$+1.5 V
7/14	T2OUT/T1OUT	Saída nível RS-232C (Entrada da recepção do PC)
8/13	R2IN/R1IN	Entrada nível RS-232C (Saída da transmissão do PC)
9/12	R2OUT/R1OUT	Saída nível TTL (Entrada da recepção do PIC)
10/11	T2IN/T1IN	Entrada nível TTL (Saída da transmissão do PIC)
15	GND	Terra
16	V_{CC}	Alimentação 5V

Tabela 6.3. Pinagem do componente MAX232-N.

O *hardware* específico para a implementação da comunicação serial é o *chip* MAX232-N (U3) e o conector DB9F (CN1), que podem ser observados no esquema do Anexo C.

Note que no conector DB9F, que é o conector do cabo advindo do microcomputador, foi necessário curto-circuitar os pinos 4 e 6 (*Data Terminal Ready* e *Data Set Ready*, respectivamente) e os pinos 7 e 8 (*Request To Send* e *Clear To Send*), a fim de se "enganar" a BIOS do computador, que espera estes sinais a fim de poder enviar os dados.

A programação do módulo de comunicação serial do lado do PIC está detalhadamente descrita no Anexo D.

7. Energia solar

O quarto e último grande módulo do projeto envolve a obtenção de energia solar, o que é realizado utilizando-se um painel solar, um circuito de transferência de energia para a bateria e um *software* de controle deste circuito, que também será implementado no microcontrolador PIC16C74.

7.1. Painel solar

Teoricamente, a máxima densidade de potência solar na superfície do planeta Terra vale $\lambda = 1000W/m^2$. Este valor é obtido na linha do Equador, ao meio-dia e a 25°C, ao nível do mar. Entretanto é impossível se aproveitar totalmente esta energia pois as células solares comerciais tem eficiência de no máximo 25%. Entretanto células com esta eficiência são muito caras pois seu preço aumenta exponencialmente com a sua eficiência. Desta forma, normalmente são utilizadas células com eficiência entre 10 e 16%.

Devido à geometria do planeta, estima-se que a densidade de potência solar na cidade de São Paulo (que se encontra sobre o Trópico de Capricórnio, ou seja, ≈23° de latitude) seja de $\lambda=1000\cdot\cos(23°) = \mathbf{920W/m^2}$, conforme pode-se observar na figura abaixo.

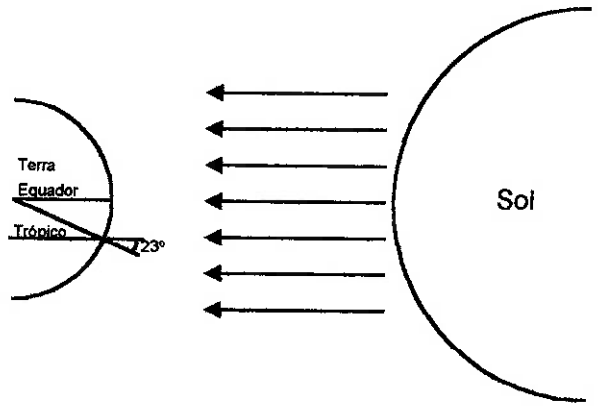


Figura 7.1. Representação esquemática da incidência solar no planeta Terra.

É um requisito de projeto que um dia de incidência seja suficiente para operar a lâmpada por dois dias. Por segurança, se a tensão da bateria estiver em menos de 11V (significando que a mesma está descarregada), a lâmpada não se acende, a fim de garantir que a bateria não se descarregue totalmente, possibilitando o seu recarregamento no dia seguinte. Para o projeto utilizou-se um painel disponível no departamento com as seguintes características:

Fabricante	Siemens
Modelo	M20
Potência máxima	20W
Corrente nominal	1,38A
Tensão nominal	14,5V
Corrente de curto circuito	1,60A
Tensão em aberto	18,0V
Dimensão da área coberta por células	52cmx31cm
Área das células	$\approx$1612cm^2

Tabela 7.1. Características do painel solar.

Estas características são para o painel a 25°C, ao meio-dia, na linha do Equador. Nesta situação a densidade de energia luminosa é máxima, correspondendo a 1000 W/m^2. Desta forma, pode-se concluir que a eficiência do painel é de aproximadamente 12,4%. Chama-se a atenção para este baixo valor pois o custo dos painéis solares cresce exponencialmente em relação à sua eficiência.

O item 7.2 apresenta o *hardware* implementado para a transferência de potência entre o painel e a bateria. Desta forma, é possível se adaptar o circuito para painéis que forneçam maior potência.

7.2. O circuito de transferência de energia

O projeto do circuito de transferência de energia entre o painel e a bateria e o seu *software* de controle foram denominados de **"Rastreador de Ponto de Potência Máxima em Painéis Solares" (RPPM)**, pois o sistema visa a melhor transferência de energia do painel solar para a bateria, visto que este ponto varia conforme a incidência luminosa, temperatura do painel e carga acoplada (características da bateria). No Anexo F encontra-se o princípio de funcionamento deste rastreador (recomenda-se sua leitura antes de prosseguir no texto).

No Anexo C encontra-se o esquema do circuito final. Os próximos sub-itens apresentarão cada detalhe considerado na confecção do mesmo.

7.2.1. Indutor que armazena energia

Como já foi apresentado, a tensão nominal do painel é maior que 12V, o que inviabiliza o uso de um conversor elevador de tensão (Boost). Desta forma, ao invés de simplesmente utilizar-se um indutor no circuito, utilizou-se um indutor-transformador, com a finalidade de se abaixar a tensão de entrada do circuito de transferência. Assim sendo, utilizou-se uma relação de espiras de 10:8. Uma característica importante que deve ser considerada é a indutância do enrolamento primário, que foi de 4,5mH.

Desta forma a configuração do circuito que representa o RPPM (Figura F.2), assume a seguinte forma:

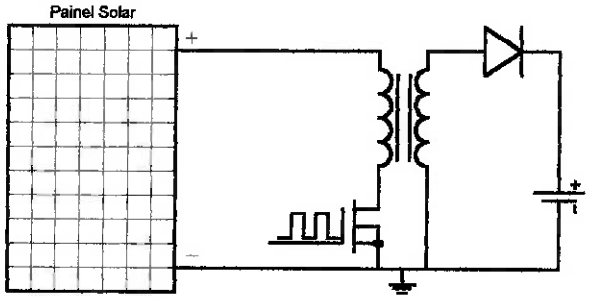


Figura 7.2. RPPM com indutor-transformador.

Abaixo encontram-se as características construtivas do trafo utilizado.

Núcleo	Tipo EE 55/20 Al = 250
Corrente máxima nominal	1,5A
Indutância do enrolamento primário	4,5mH
Relação de espiras	10:8
Número de espiras do primário	200
Número de espiras do secundário	160
Fio utilizado	AWG 22, fio duplo, primário e secundário enrolados juntos.

Tabela 7.2. Características construtivas do indutor-transformador.

Chama-se a atenção para o fato de ocorrer uma inversão do sinal entre o pino 17 do microcontrolador (saída do PWM) e a base dos MOSFETs Q1. Quando o sinal no microcontrolador está em nível *high*, o indutor está descarregando para a bateria. Por conseguinte, quando o sinal estiver em *low*, o mesmo está carregando.

7.2.2. Monitoramento da tensão da bateria

Como o funcionamento do *software* do RPPM baseia-se na verificação da potência que vai do painel para a bateria, é necessário saber-se a tensão e a corrente da mesma. O monitoramento da tensão é feito através do divisor resistivo R2, R3 e R4 do circuito, que pode ser visto no Anexo C.

A fim de se ter uma segurança para não se danificar o microcontrolador, admite-se que quando a tensão máxima da bateria for 14V, a tensão no pino 3 do microcontrolador será 5V. A fim de tornar o projeto "adaptável" a alguma alteração, utilizou-se a resistência R3, a fim de poder-se ter alguma tensão desejada entre R2 e R3.

Desta forma, a proporção entre R4/(R2+R3+R4) deve ser de 5/14. Definindo-se R4=30kΩ e R3=R4, sai que R2=24kΩ, que é um valor comercial de resistência.

7.2.3. Monitoramento da corrente de carga da bateria

A fim de se monitorar a corrente indo para a bateria, utilizou-se a resistência R1 de 50mΩ e o amplificador operacional U2. O princípio deste circuito é associar um valor de tensão em função da corrente que passa pelo resistor R1.

Definiu-se como a corrente máxima que carrega a bateria sendo 1,5A. Desta forma, a queda de tensão em R1 quando passar 1,5A será de 75mV.

A função do amplificador operacional é a de um amplificador não-inversor de tensão. Tem-se que um amplificador operacional alimentado com 5V, tem sua saída saturando em 3,5V. Desta forma calcula-se o ganho do amplificador sendo G=3,5V/7,5mV = 46,7 = 47. O ganho de um amplificador não-inversor é dado por G=(R10/R11)+1 (Anexo C). Definindo-se R11=1kΩ, o valor comercial mais próximo para R10 é de 47kΩ. Assim o ganho real deste amplificador é de G=48.

Entretanto, se o amplificador operacional tiver sua alimentação negativa (pino 4 de U2) aterrada, ocorre uma alteração no ganho para valores de saída próximos de 0V. Desta forma propôs-se alimentar o amplificador operacional com –5V. Para tanto utilizou-se um conversor com saída negativa, que utiliza um 555 como núcleo central, conforme pode ser observado no circuito do RPPM (U4). Este circuito, quando alimentado com +5V, tem como tensão de saída –3,4V.

Abaixo tem-se os dados coletados para o amplificador operacional U2 tendo sua alimentação negativa com 0V, -5V e –3,4V.

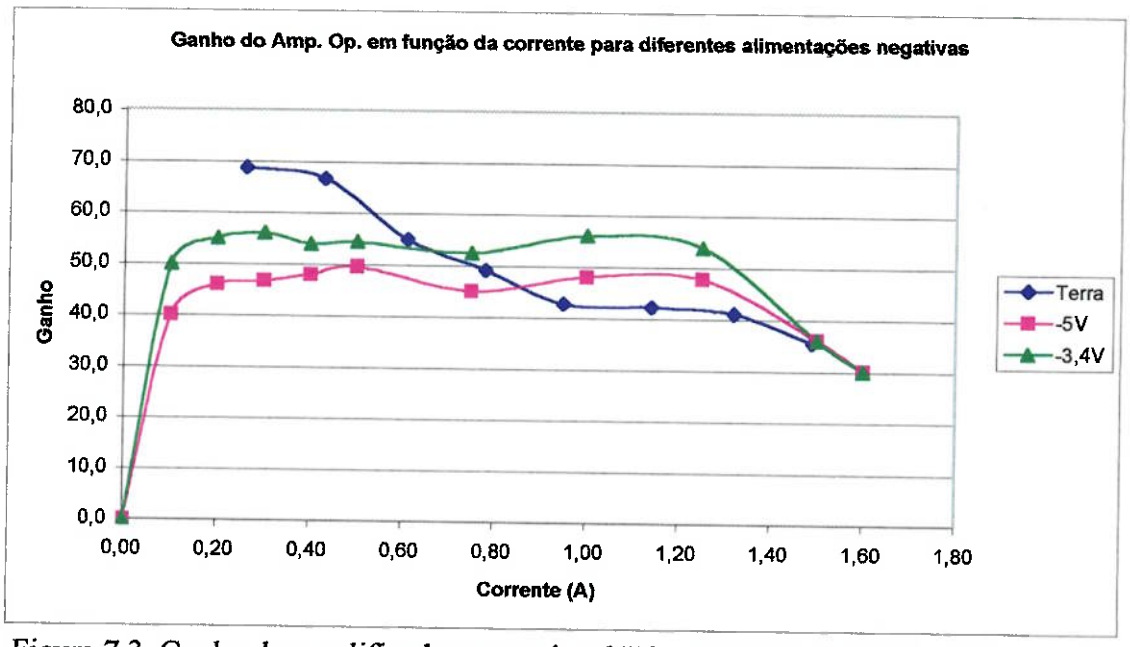


Figura 7.3. Ganho do amplificador operacional U2 em função da corrente de carga da bateria.

Pela figura acima, observa-se que o ganho real, ao invés de ser 48, é de aproximadamente 55. Isto não traz nenhuma conseqüência negativa para o projeto.

A figura abaixo mostra a curva de calibração da tensão de saída do amplificador operacional (entrada de "corrente" do microcontrolador) em função da corrente.

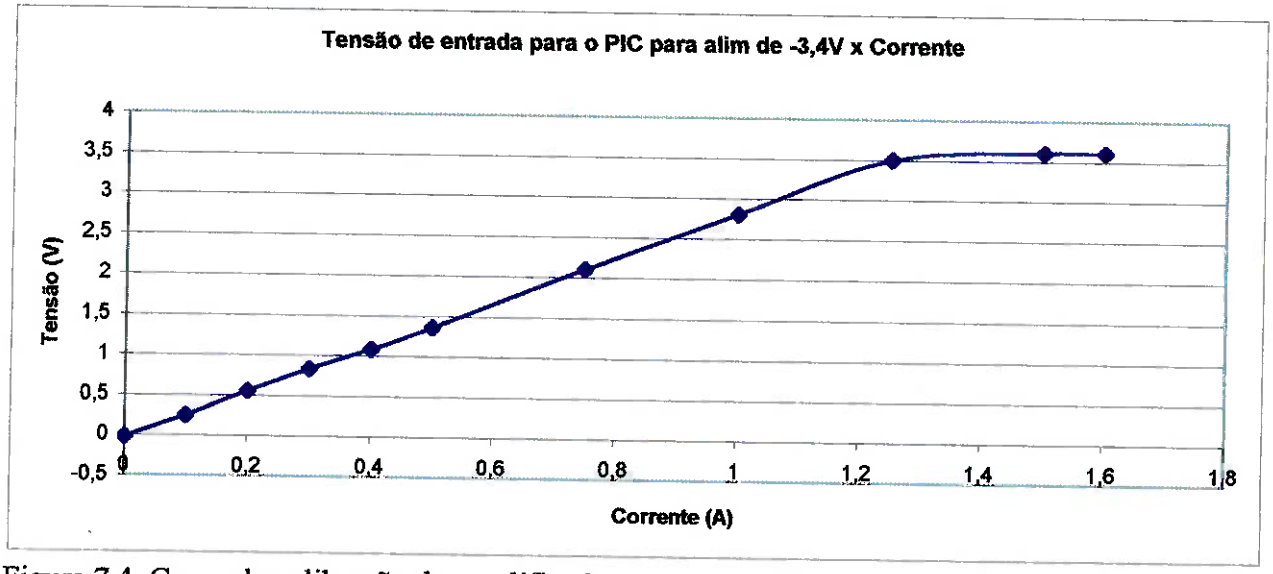


Figura 7.4. Curva de calibração do amplificador operacional U2 em função da corrente.

Observa-se que ocorre saturação para a corrente em 1,25A e não em 1,5A como foi projetado, entretanto, isto não traz nenhuma conseqüência negativa para o projeto

pois a potência máxima do painel é de 20W. Desta forma, supondo perda zero no RPPM, a máxima corrente de carga da bateria seria I=20W/12V=1,67A. Como há perdas da ordem de 30%, a máxima corrente será perto de 1,2A. As perdas e eficiência do RPPM serão discutidas em um item adiante. Por outro lado, basta variar o ganho de U2, para se poder trabalhar com correntes maiores.

7.3. O *software* do rastreador do ponto de potência máxima

Conforme foi explicado no Anexo F, existe um ponto de máxima potência, e este ponto varia com a incidência solar, a temperatura e a impedância da carga (no caso, uma bateria automotiva de 12V). O software implementado no microcontrolador PIC16C74, basicamente controla o sinal de PWM, base do MOSFET. A cada *x* milissegundos, o microcontrolador monitora a potência atual que está carregando a bateria, através do produto tensão vezes corrente, e, após comparar com a média das últimas 32 potências calculadas, aumenta ou diminui o tempo de condução do MOSFET. Ou seja, se no instante anterior o programa diminuiu o *duty-cycle*[2] do PWM e a potência atual é menor que a média, então neste instante ele aumentará o *duty-cycle*. O fluxograma abaixo mostra a idéia geral do *software* rastreador do ponto de potência máxima.

[2] Define-se *duty-cycle* como sendo a razão entre t_{high}/t_{total} do sinal de PWM.

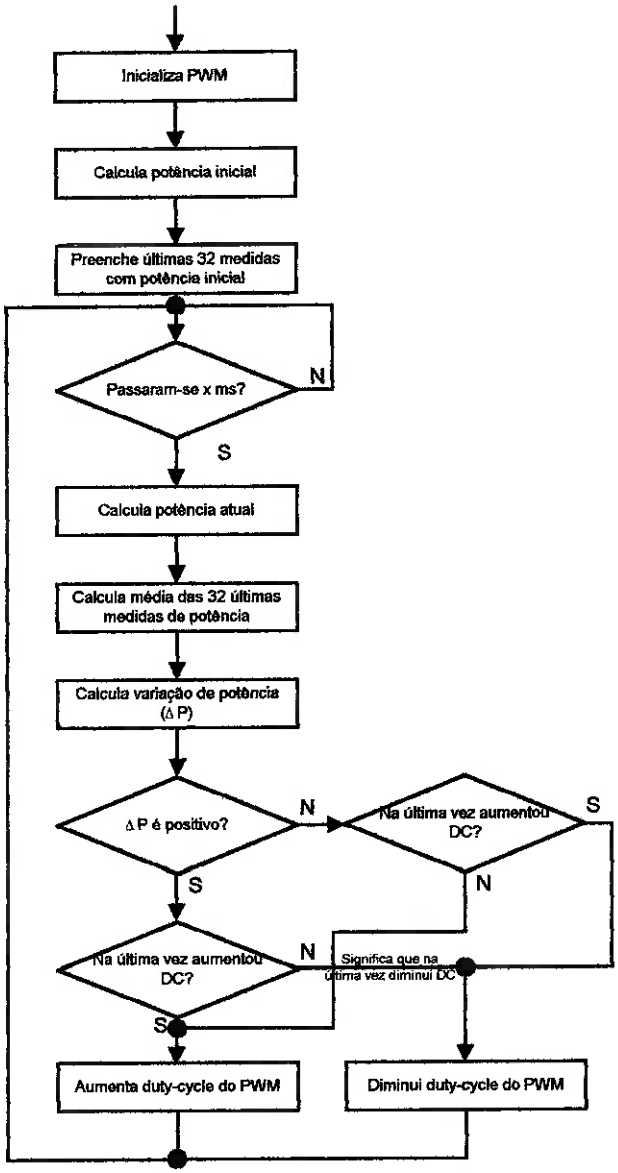


Figura 7.5. Fluxograma básico do RPPM.

Algumas considerações a serem feitas são o período do PWM e o período em que o programa monitorará a potência de carga e variará do *duty-cycle* do mesmo. Adotou-se o período do PWM como 32μs. Este valor é obtido através da fórmula:

$$T_{PWM} = (PR2 + 1) \cdot 4\, T_{OSC} \cdot (\text{Prescaler do TMR2})$$

Como o cristal utilizado é de 4MHz, então $T_{OSC} = 1/(4 \cdot 10^6)$ s. Utilizou-se o prescaler do TMR2 como 1. Então, para este caso, o valor de PR2, que é programável, vale 31.

O *duty-cycle* do PWM é dado por:

$$DC_{PWM} = (DC1)\cdot T_{OSC}\cdot(\text{Prescaler do TMR2})$$

Onde DC1 é programável e tem 10 bits. DC_{PWM} é dado em segundos e representa t_{high} do PWM. Dividindo DC_{PWM} por T_{PWM} tem-se o *duty-cycle* em porcentagem. Ou seja,

$$\frac{DC_{PWM(s)}}{T_{PWM}} = \frac{DC1}{4(PR2+1)}$$

$$DC_{PWM(\%)} = \frac{DC1}{128}$$

O período de monitoração de corrente e tensão, após algumas experiências em laboratório foi adotado como 65,5ms. Este período é ideal pois se ele for menor ocorre instabilidade do sistema e se ele for maior, a adaptação do programa a uma nova situação de iluminação é extremamente lenta. Este valor é baseado na interrupção do TMR0. Este *timer* é um contador de 8 bits e é incrementado a cada borda de subida do *clock*, que tem sua freqüência igual à $f_{osc}/4$, ou seja f_{clock} = 1MHz.

Adotou-se um prescaler de 32 para este *timer*, então ocorre uma interrupção a cada $32\cdot256\cdot(1\cdot10^{-6})$ = 8,192ms. O primeiro fator representa o prescaler, o segundo o *overflow* de 8 bits e o terceiro a freqüência do *clock*. Utilizou-se um contador de interrupções do TMR0 a fim de se atingir valores múltiplos de 8,192ms. A melhor operação, como já foi dito, foi obtida para este contador valendo 8, totalizando um período de amostragem de 65,536ms.

Como tanto a conversão A/D da corrente como da tensão são de 8 bits, então o produto de ambas será um valor de 16 bits. Assim sendo, a potência é formada por dois bytes. A fim de se armazenar as 32 últimas medidas (ou seja, 64 bytes), utilizou-se um vetor implementado através de endereçamento indireto de memória. Este vetor vai da posição 0xB0 até 0xF0. Onde as posições 0xB0, 0xB2, etc., guardam os bytes mais significativos de cada leitura e as posições 0xB1, 0xB3, etc., os menos significativos.

Os detalhes de programação se encontram no Anexo D, onde o *software* que integra o rastreador de potência, o relógio de tempo real e a comunicação serial está detalhadamente documentado.

8. Resultados experimentais sobre o módulo de captação de energia solar

8.1. Experiências em laboratório

Este item tem como finalidade apresentar o fato de que o Rastreador do Ponto de Potência Máxima realmente encontra o ponto de máxima potência. Para tanto construi-se um circuito em que se varia o PWM manualmente. Desta forma, levantou-se várias curvas sobre o funcionamento do painel, onde foi possível encontrar o ponto de máximo para cada situação. A partir disto, para as mesmas condições, colocou-se o microcontrolador rodando o algoritmo de busca. O ponto encontrado pelo mesmo também está plotado nestas curvas. Ressalta-se que estas experiências foram realizadas com uma fonte de tensão, pois é impossível simular de maneira controlada várias situações diferentes em um painel solar.

A figura abaixo apresenta o circuito de geração manual de PWM.

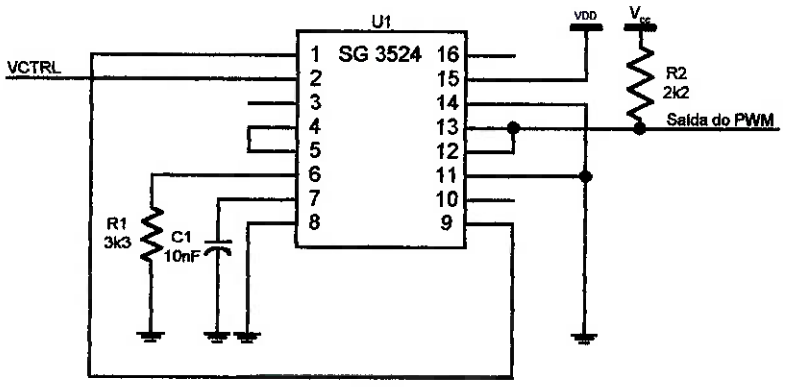


Figura 8.1. Circuito de variação manual de PWM.

No circuito acima, V_{CC} = 5V, V_{DD} = 12V e V_{CTRL} varia entre 1,5 e 3,5V aproximadamente, variando linearmente o *duty-cycle*. A saída do PWM entra diretamente no lugar do pino 17 do PIC16C74 no circuito apresentado no Anexo C. O período do PWM neste circuito é dado por R1·C1 = 33μs. Entretanto, experimentalmente, observou-se que este valor foi de 34μs. Isto é importante de se considerar pois os gráficos são dados em porcentagem de PWM. Lembra-se que o período do PWM advindo do microcontrolador é de 32μs.

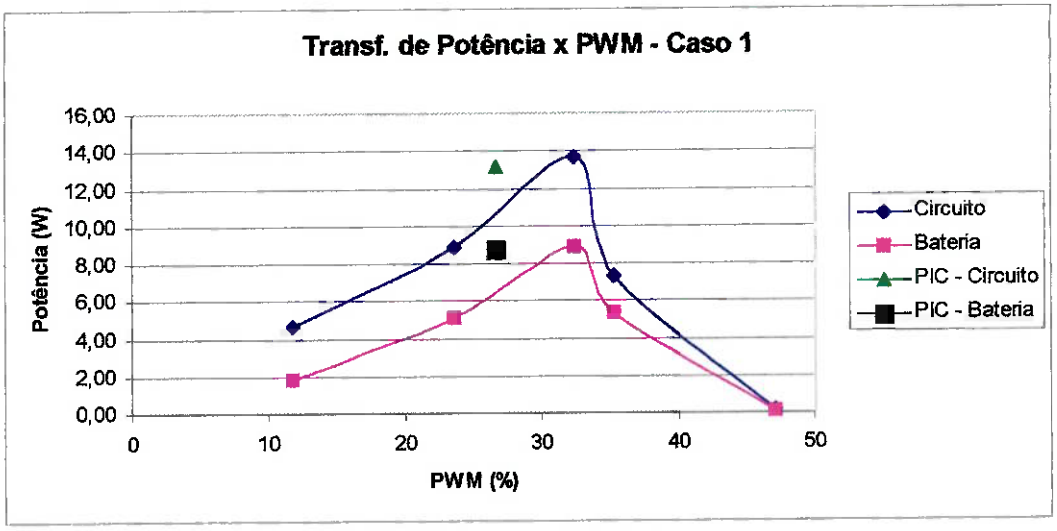
A tabela abaixo apresenta as medidas realizadas para seis casos de potência, que são a combinação de tensão de entrada em 7V, 10V e 13V e corrente de alimentação em 0,5A e 1,5A.

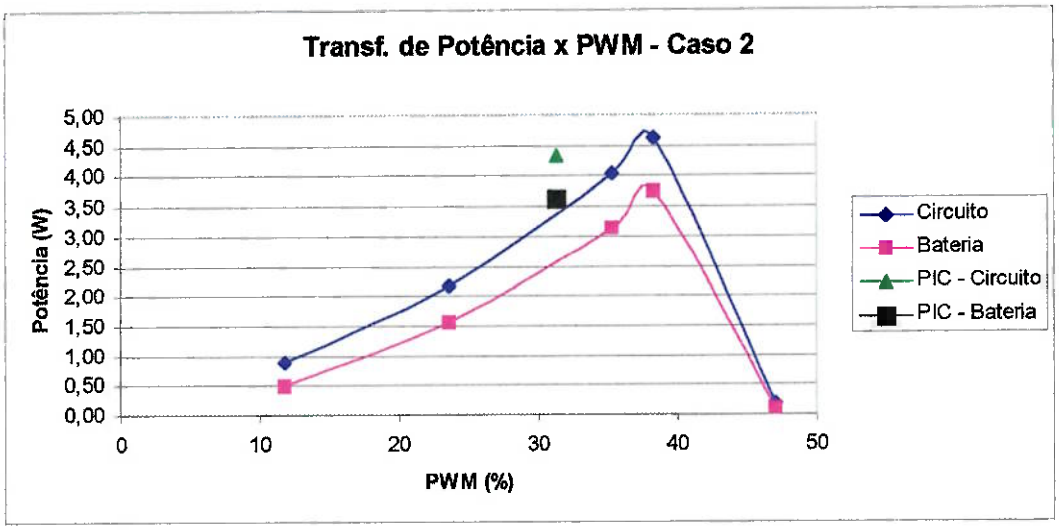
Medida no.	PWM (34 s) (s)	PWM (34 s) (%)	Tensão na fonte (V)	Circuito Tensão (V)	Circuito Corrente (A)	Circuito Potência (W)	Bateria Tensão (V)	Bateria Corrente (A)	Bateria Potência (W)	Perda na transf. de pot. (%)
Caso 1 - Tensão em aberto 10V, limitador de corrente em 1,5A										
1	4	11,76	4,0	3,12	1,500	4,68	12,05	0,15	1,81	61,38
2	8	23,53	6,9	5,94	1,497	8,89	12,10	0,42	5,08	42,85
max	11	32,35	9,9	9,19	1,490	13,69	12,17	0,73	8,88	35,12
3	12	35,29	10,0	9,08	0,809	7,35	12,12	0,44	5,33	27,40
4	16	47,06	10,0	9,94	0,020	0,20	12,04	0,01	0,12	39,44
PIC	8,5	26,56	9,8	8,88	1,480	13,14	12,17	0,72	8,76	33,33
Caso 2 - Tensão em aberto 10V, limitador de corrente em 0,5A										
1	4	11,76	2,4	1,73	0,505	0,87	12,02	0,04	0,48	44,97
2	8	23,53	4,9	4,30	0,503	2,16	12,03	0,13	1,56	27,69
3	12	35,29	8,7	8,02	0,503	4,03	12,06	0,26	3,14	22,27
max	13	38,24	9,9	9,27	0,499	4,63	12,07	0,31	3,74	19,11
4	16	47,06	10,0	9,90	0,020	0,20	12,00	0,01	0,12	39,39
PIC	10	31,25	9,8	8,80	0,492	4,33	12,08	0,30	3,62	16,30
Caso 3 - Tensão em aberto 7V, limitador de corrente em 1,5A										
1	4	11,76	3,5	3,06	1,508	4,61	12,05	0,15	1,81	60,83
2	8	23,53	6,4	5,97	1,508	9,00	12,01	0,43	5,16	42,64
max	8,5	25	6,9	6,52	1,505	9,81	12,12	0,48	5,82	40,71
3	12	35,29	7,0	6,96	0,020	0,14	12,04	0,01	0,12	13,51
PIC	6	18,75	7,0	6,54	1,478	9,67	12,12	0,47	5,70	41,07
Caso 4 - Tensão em aberto 7V, limitador de corrente em 0,5A										
1	4	11,76	1,9	1,79	0,516	0,92	12,00	0,04	0,48	48,03
2	8	23,53	4,5	4,35	0,513	2,23	12,02	0,13	1,56	29,98
max	11	32,35	7,0	6,81	0,499	3,40	12,04	0,22	2,65	22,05
3	12	35,29	7,0	6,95	0,210	1,46	12,00	0,01	0,12	91,78
PIC	8	25	6,8	6,50	0,486	3,16	12,07	0,20	2,41	23,58
Caso 5 - Tensão em aberto 13V, limitador de corrente em 1,5A										
1	4	11,76	3,9	2,97	1,510	4,48	11,86	0,14	1,61	64,03
2	8	23,53	6,9	5,98	1,500	8,97	11,93	0,43	5,11	43,08
3	12	35,29	11,3	10,36	1,500	15,54	12,02	0,84	10,11	34,95
max	13	38,24	12,1	11,70	1,480	17,32	12,12	0,95	11,53	33,44
4	16	47,06	13,0	12,83	0,130	1,67	11,92	0,13	1,53	8,52
PIC	11	34,38	13,0	11,90	1,500	17,85	12,16	0,98	11,87	33,51
Caso 6 - Tensão em aberto 13V, limitador de corrente em 0,5A										
1	4	11,76	2,2	1,80	0,490	0,88	11,96	0,04	0,48	45,76
2	8	23,53	4,8	4,43	0,490	2,17	11,97	0,14	1,64	24,45
3	12	35,29	8,5	8,08	0,499	4,03	12,00	0,26	3,17	21,43
max	15	44,12	12,3	11,21	0,494	5,54	12,04	0,37	4,45	19,56
4	16	47,06	13,0	12,88	0,110	1,42	11,98	0,11	1,33	6,14
5	20	58,82	13,0	12,94	0,020	0,26	11,95	0,02	0,24	7,65
PIC	14	43,75	12,5	12,10	0,500	6,05	12,06	0,41	4,94	18,27

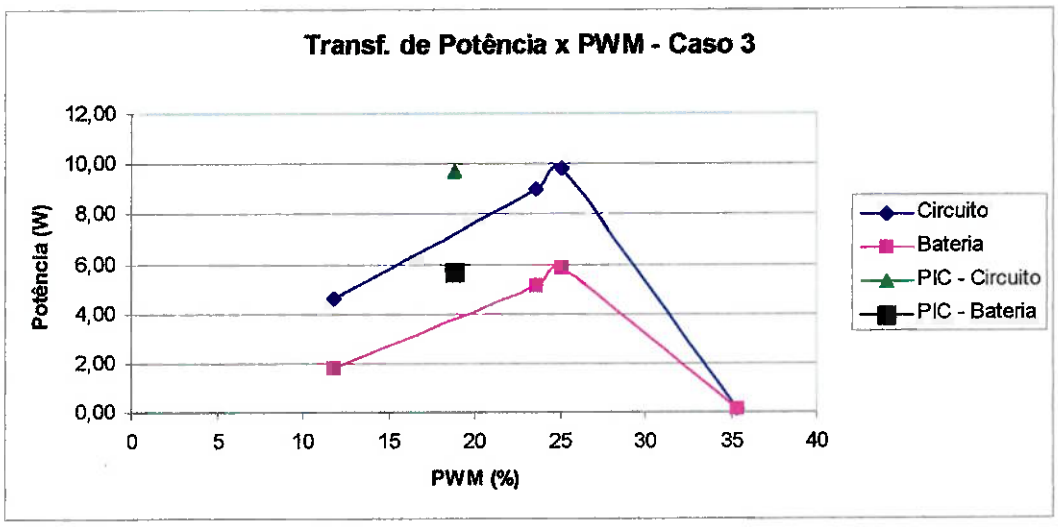
Tabela 8.1. Dados coletados para validação do algoritmo de busca.

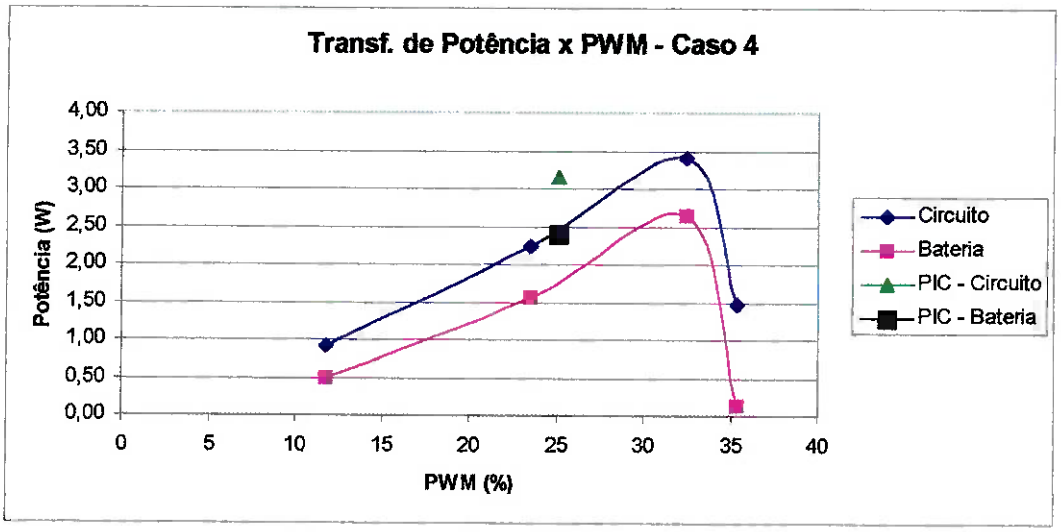
Obs.: O PWM do PIC é baseado em 32μs.

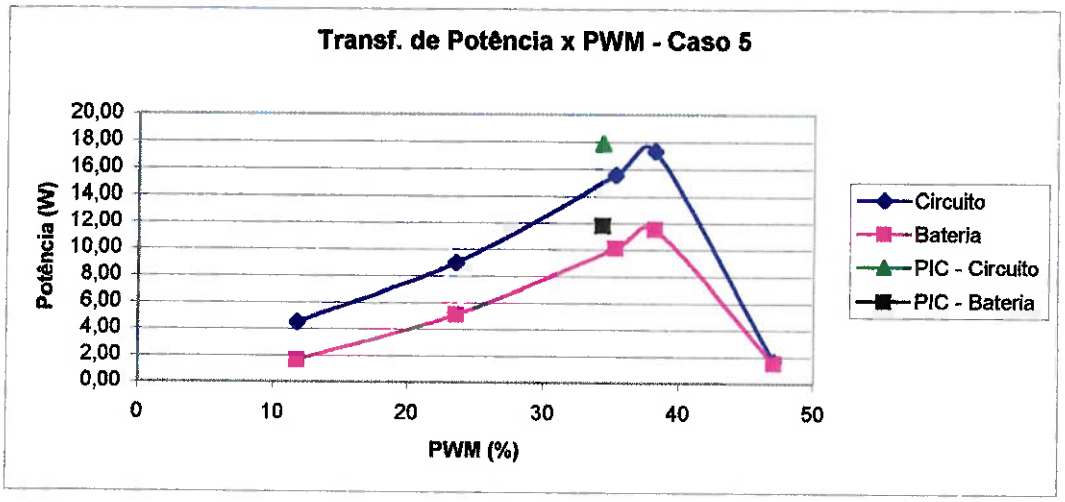
As figuras a seguir plotam as potências de entrada e de saída de cada caso, bem como o ponto que o PIC encontrou. Chama-se a atenção que embora o PIC não encontre o mesmo ponto em função da porcentagem de PWM, o valor de potência é o mesmo. Isto, por si só, já valida o algoritmo de busca. Pode-se entender esta diferença como os períodos dos dois PWM serem diferentes.


Transf. de Potência x PWM - Caso 1
Potência (W)
16,00
14,00
12,00
10,00
8,00
6,00
4,00
2,00
0,00
0
10
20
30
40
50
PWM (%)
Circuito
Bateria
PIC - Circuito
PIC - Bateria


Transf. de Potência x PWM - Caso 2
Potência (W)
5,00
4,50
4,00
3,50
3,00
2,50
2,00
1,50
1,00
0,50
0,00
0
10
20
30
40
50
PWM (%)
Circuito
Bateria
PIC - Circuito
PIC - Bateria


Transf. de Potência x PWM - Caso 3
Potência (W)
12,00
10,00
8,00
6,00
4,00
2,00
0,00
0
5
10
15
20
25
30
35
40
PWM (%)
Circuito
Bateria
PIC - Circuito
PIC - Bateria





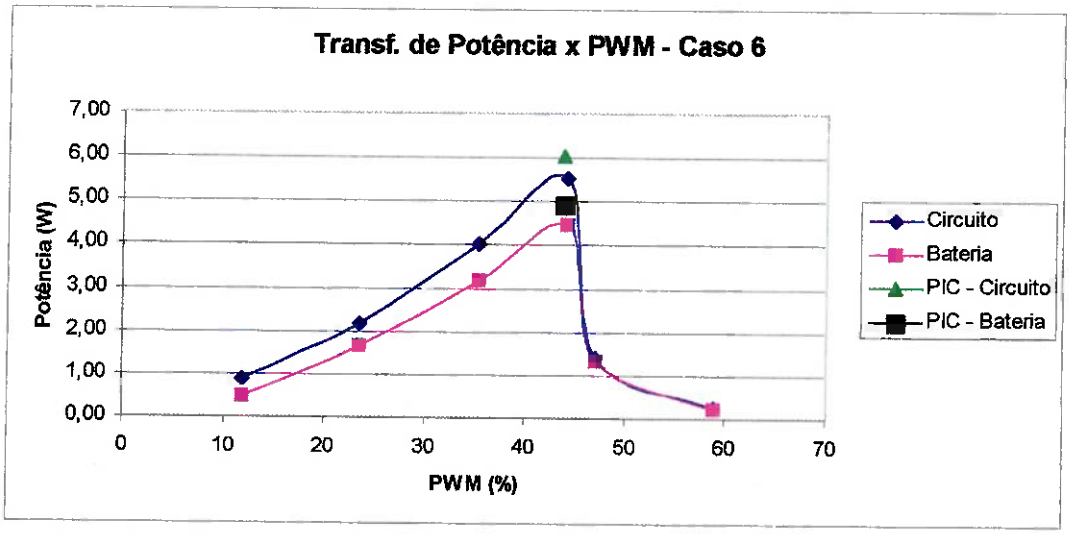


Figura 8.2. Comparação entre ponto máximo de potência obtido através de PWM manual e algoritmo de busca do microcontrolador.

É importante observar que a perda na transferência, no seu ponto máximo, é entre 16% e 40% (este valor seria ainda maior se não se estivesse utilizando dois MOSFETs em paralelo). Para diminuir este "desperdício", é necessário utilizar, por exemplo um MOSFET com menores perdas que o IRF740 (por exemplo o IRF540), ou um IGBT, sem que seja necessário correções consideráveis no circuito. Entretanto, estes componentes são caros e não são encontrados no mercado, devendo ser comprados sob encomenda de grandes quantidades. Embora isto fuja ao escopo deste trabalho, deve ser considerado em uma aplicação comercial do Poste Solar.

Outra característica importante do funcionamento do algoritmo é o seu limiar de operação, ou seja, existe um limite mínimo de tensão e corrente para qual o algoritmo opere corretamente. Abaixo pode-se observar o gráfico obtido empiricamente sobre os valores mínimos de operação do programa.

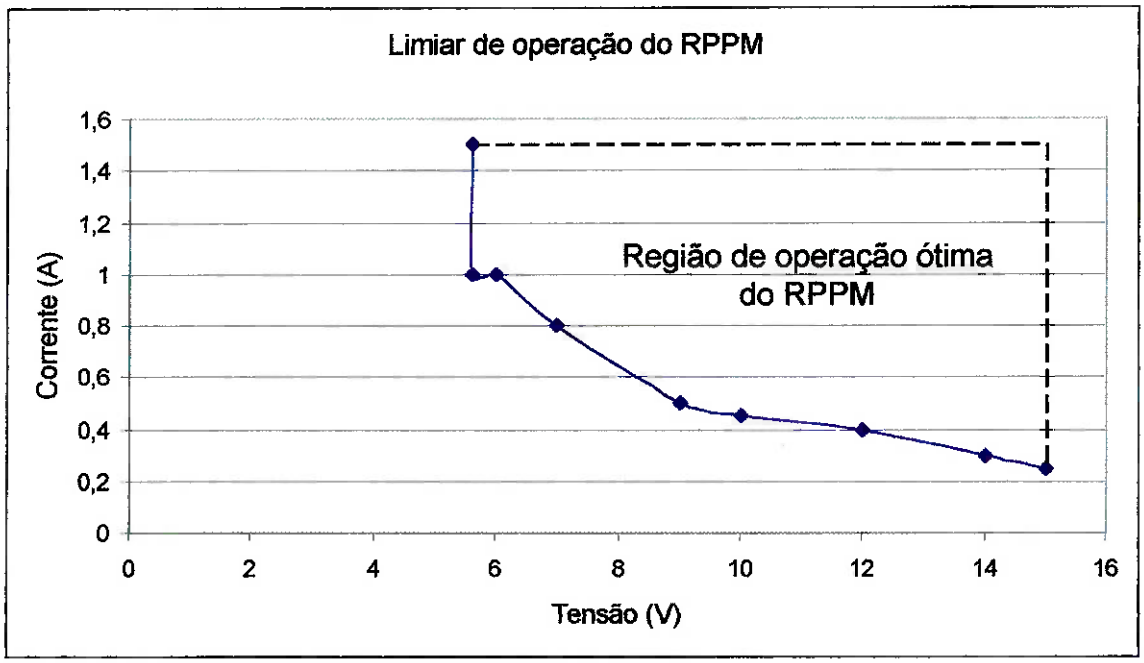


Figura 8.3. Região de operação ótima do RPPM em função da tensão e corrente advindas do painel solar.

Finalmente, constatou-se que o circuito, ao ser alimentado com 12V (uma bateria automotiva), consome 38mA, ou seja, seu **consumo** é de **456mW**.

8.2. Experiências em campo

Após a validação do algoritmo, realizou-se experiências em campo, medindo-se ao longo do dia o funcionamento do programa.

A fim de se estimar a potência solar incidente, utilizou-se uma célula fotovoltaica calibrada, onde 100mW/cm^2 (ou seja 1000W/m^2) correspondem a uma tensão de saída de 108mV. Considerando-se a eficiência do painel (12,4%) e sua área (1612cm^2), pode-se estimar a potência solar incidente no mesmo.

A tabela abaixo apresenta os dados coletados no dia 3/12/98, um dia em que se fez bastante sol (coincidentemente, este foi considerado o dia mais quente na cidade de São Paulo desde 1944, data em que iniciou-se este tipo de acompanhamento). Chama-se a atenção para o fato das medidas terem sido realizadas durante o horário de verão, ou seja, 13 horas do horário da tabela representa as 12 do horário geográfico.

Medida no.	Hora	PWM (s)	Painel			Bateria			Perda na transf. de pot. (%)	Célula de Aferição (mV)	Potência solar incidente (W/m2)	Potência teórica absorv. pelo painel (W)	Variação entre pot. teór. e observ. no painel (%)
			Tensão (V)	Corr. (A)	Pot. (W)	Tensã (V)	Corr. (A)	Pot. (W)					
1	10,5	11	12,0	1,3	15,60	12,25	0,88	10,78	30,90				
2	11,25	11	12,0	1,3	15,60	12,24	0,90	11,02	29,38				
3	11,75	11	11,5	1,3	14,95	12,24	0,90	11,02	26,31				
4	12,5	12	12,0	1,4	16,56	12,26	0,93	11,40	31,15				
5	14	11	12,5	1,4	17,50	12,30	0,88	10,82	38,15				
6	14,5	10	11,8	1,2	14,16	12,30	0,85	10,46	26,17	90	134,33	16,66	14,99
7	15	11	12,0	1,2	14,40	12,30	0,82	10,09	29,96	90	134,33	16,66	13,55
8	15,5	12	12,0	1,0	12,00	12,32	0,78	9,61	19,92	78	116,42	14,44	16,88
9	16	12	11,5	0,8	9,20	12,29	0,63	7,74	15,84	54	80,60	9,99	7,95
10	16,5	12	12,0	0,9	10,80	12,29	0,70	8,60	20,34	68	101,50	12,59	14,19
11	17	11	11,0	0,8	8,80	12,32	0,60	7,39	16,00	60	89,56	11,10	20,76
12	17,5	12	12,5	0,7	8,75	12,31	0,53	6,52	25,44	44	65,67	8,14	-7,45

Tabela 8.2. Dados coletados em campo. (Obs.: medidas realizadas durante horário de verão).

A partir dos dados coletados acima, foi possível traçar o gráfico da potência incidente no painel, potência absorvida pelo mesmo e potência de carga da bateria, conforme pode ser observado abaixo.

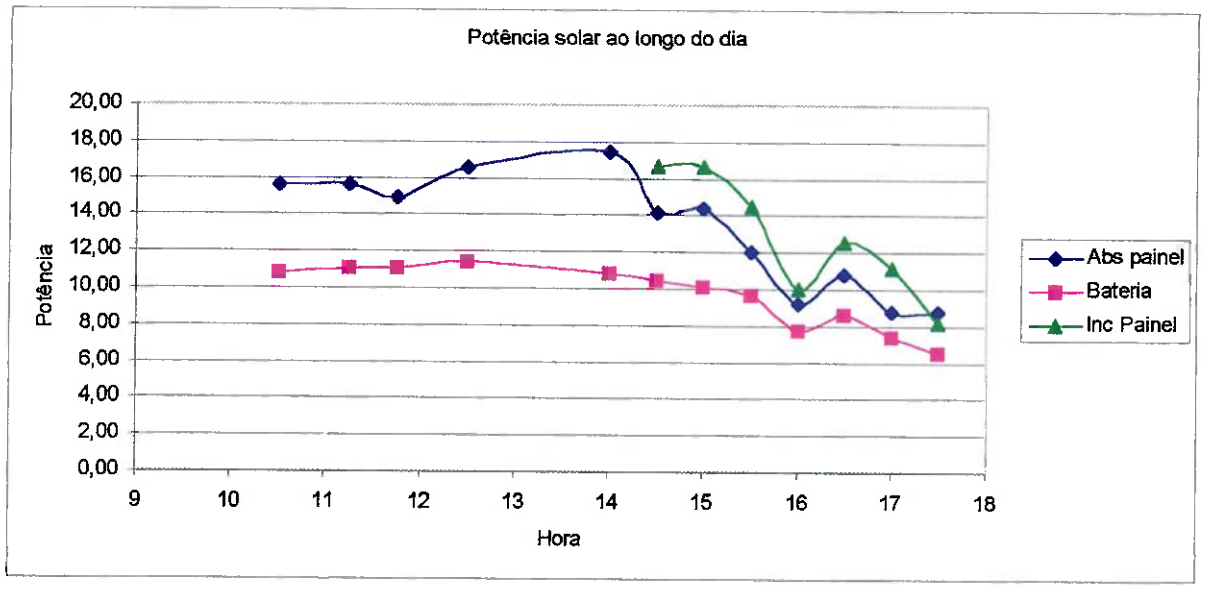


Figura 8.4. Potência incidente no painel solar (teórica), potência absorvida pelo mesmo e potência de carga da bateria. (Obs.: medidas realizadas durante horário de verão).

Pelo gráfico acima, percebe-se que em alguns momentos, não existe um acompanhamento entre a potência da bateria e do painel. Entretanto, deve-se salientar que embora a tensão e corrente na bateria sejam praticamente constantes num dado instante, ocorre uma oscilação nestas variáveis do painel, o que não era observado na fonte do laboratório. Isto provavelmente se deve à dinâmica do painel. Desta forma, pode-se concluir que em alguns pontos, não se observou corretamente o valor média de tensão e corrente do painel, visto que a curva do mesmo não acompanha o trajeto da curva de incidência teórica.

9. Bibliografia

BERLIN, Howard M. *Aplicações para o 555 com Experiências*. EDITELE – Ed. Técnica Eletrônica Ltda. São Paulo, 1976.

GOODWIN, Mark. *Serial Communications in C and C++*. Ed. MIS Press. 1ª Ed., USA, 1992.

MALVINO, Albert P. *Eletrônica*. Vols. 1 e 2. Ed. McGraw-Hill. São Paulo, 1986.

Microchip. *Embedded Control Handbook 1994/95*. USA, 1994.

Microchip. *MPASM Assembler User's Guide*. USA, 1994.

Microchip. *MPSIM Simulator User's Guide*. USA, 1994.

Microchip. *PIC16C7X 8-Bit CMOS Microcontrollers with Analog Converter*. USA, 1995.

POSADAS, Jorge A. S.. *Metodologia de Projeto de um Inversor Ressonante para Lâmpadas Fluorescentes Alimentado por Fonte de Corrente*. Dissertação de mestrado apresentada ao Departamento de Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo. São Paulo, 1998.

Raltron. Catálogo de cristais RALTRON 96. Florida, USA. 1996.

TAYLOR, Billy P. *C/C++ Programmer's Guide Using PC BIOS*. Ed. Microtrend Books, USA, 1993.

Universidade de São Paulo. Folder: *"Conservar Energia é Fácil"*. Programa Permanente para o Uso Eficiente da Energia Elétrica na USP, 1998. (Xerox no Anexo G).

SIMÕES, Marcelo Godoy. *Otimização de Transferência de Potência de Painéis Solares Utilizando Controle Inteligente*. Departamento de Engenharia Mecânica da Escola Politécnica da Universidade de São Paulo. São Paulo, 1996.

Anexo A. Lista de fabricantes e representantes

Microcontrolador PIC16C74

Microchip (Fabricante)
site na Internet: www.microchip.com

MAX-232N

Texas Instruments (Fabricante)
site na Internet: www.ti.com

Cristal de 32.768 kHz

Raltron (Fabricante)
site na Internet: www.raltron.com

Hastec (Representante)
R. Ferreira do Alentejo, 90
Tel.: 522-1799
São Paulo - SP

Projeto do transformador e indutor do circuito inversor

Tecnotrafo
R. Cidade de Bagdá, 554
Tel.: 5563-4303
São Paulo – SP

Siemens Painéis Solares

www.siemens.com
www.solarpv.com

Anexo B. Inversor push-pull (baseado na obra de POSADAS)

Uma característica importante das lâmpadas fluorescentes é a sua alta eficiência, quando comparadas às lâmpadas incandescentes. Isto pode ser observado na figura abaixo.

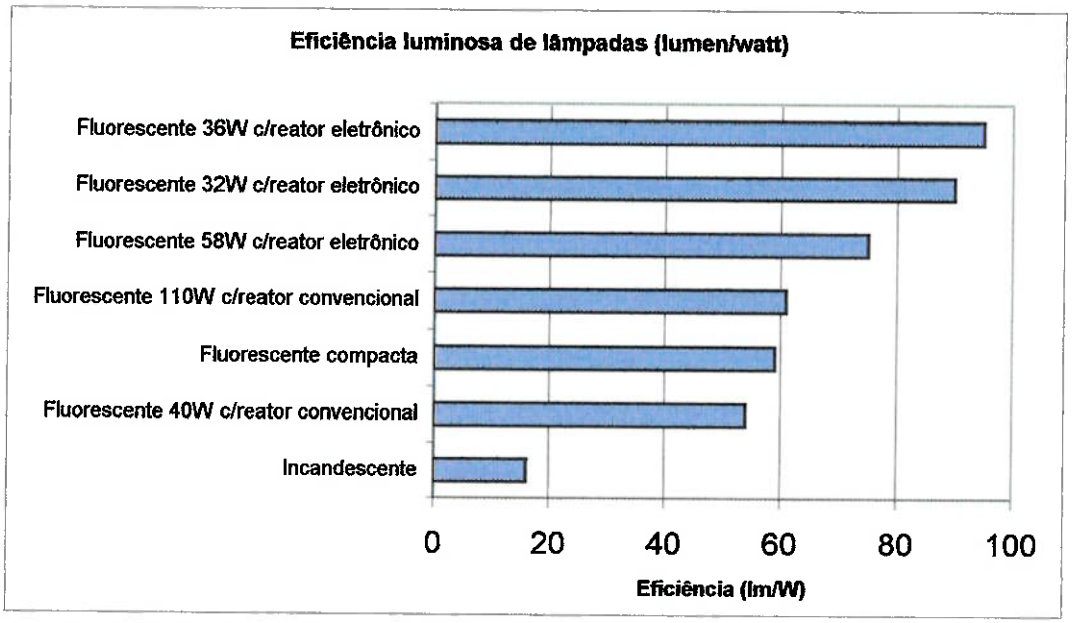


Figura B.1. Eficiência luminosa de vários tipos de lâmpadas.
Fonte: Programa Permanente para o Uso Eficiente da Energia Elétrica na USP.

A lâmpada fluorescente foi desenvolvida na década de 40. Seus quatro conectores, dois em cada extremidade são conectados por filamentos, que atingem temperaturas de 800 a 1100°C, mais baixas que as temperaturas dos filamentos das lâmpadas incandescentes. O revestimento destes filamentos é feito de um material que emite elétrons por efeito termo-iônico. Quando os filamentos estão quentes, esquenta-se o gás que preenche o interior da lâmpada, geralmente argônio. Este gás vaporiza o mercúrio que está dentro lâmpada. Os elétrons emitidos pelos filamentos, ao se chocarem com o mercúrio vaporizado, emitem uma radiação ultravioleta, que ao atingir o revestimento de pó branco (chamado genericamente de "fósforo"), interno ao vidro, emitem luz.

Desta maneira a lâmpada apresenta dois estágios de operação: partida (ignição) e descarga. No primeiro os filamentos são aquecidos e no segundo é mantido o regime de operação da lâmpada, sendo mantida a temperatura dos filamentos. A função do *starter* nos reatores convencionais é gerar um curto-circuito, que aquece os filamentos, permitindo a partida da lâmpada, conforme pode ser observado na figura abaixo.

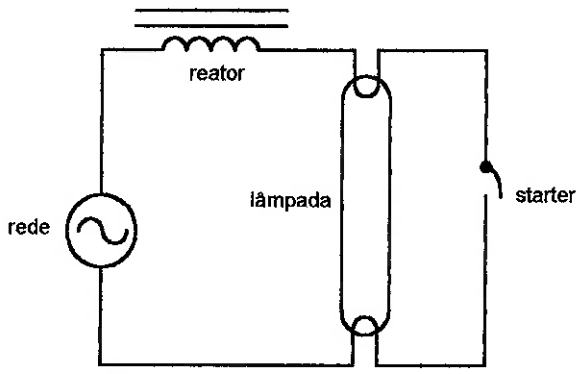


Figura B.2. Reator convencional com o uso de starter.

O circuito proposto por POSADAS, é um inversor ressonante alimentado por fonte de corrente de baixa tensão (por exemplo, uma bateria de carro). A figura abaixo apresenta este circuito.

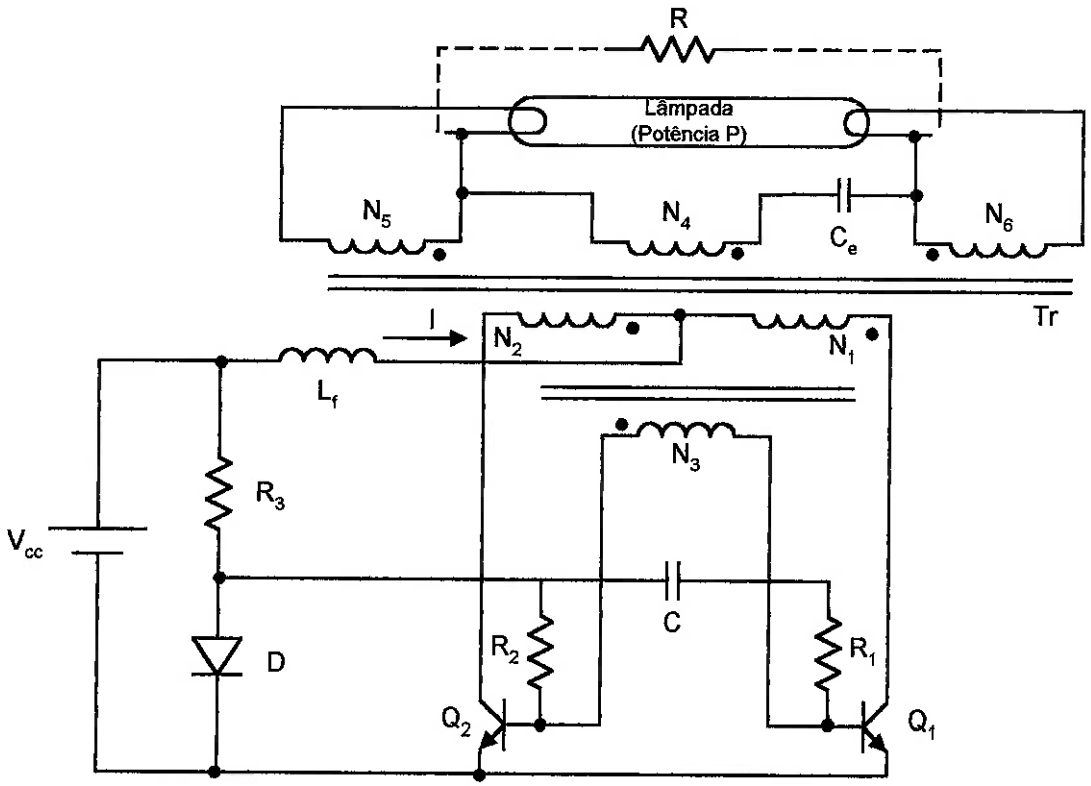


Figura B.3. Inversor ressonante alimentado por fonte de corrente, com estabilização capacitiva (C_e). Poderia também ser utilizado um indutor.

Este inversor é conhecido por inversor *push-pull*, sendo auto-oscilante e normalmente utilizado em sistemas automotivos, onde se dispõe de alimentação CC de baixa tensão.

Neste circuito, os transistores conduzem alternadamente com razão cíclica de 50% e são acionados através do enrolamento de base N_3, pelo circuito de base R_3, D, R_1 e R_2. A partida dispensa um circuito de inicialização, pois os ramos R_2-Q_2 e R_1-Q_1 nunca são perfeitamente simétricos. Desta forma, apenas um dos transistores conduzirá inicialmente, impondo corrente constante ao circuito através de um dos enrolamentos N_1 ou N_2 do primário do transformador Tr. O capacitor C e a indutância de magnetização do transformador determinam a freqüência de operação do circuito em vazio (sem lâmpada).

Os enrolamentos de filamento N_5e N_6 realizam a função de pré-aquecimento dos filamentos da lâmpada e, durante a partida, não haverá corrente no enrolamento de alta

tensão N_4, até que a temperatura dos filamentos alcance o valor adequado para a emissão dos elétrons. A freqüência de operação em carga (lâmpada acesa) é determinada pela resistência equivalente da lâmpada R, pela indutância de magnetização do transformador, pela capacitância C e pelo elemento de estabilização (indutor ou capacitor) da corrente na lâmpada. Neste projeto foi utilizado o capacitor C_e.

Para efeito de projeto, o circuito pode ser modelado conforme a figura abaixo.

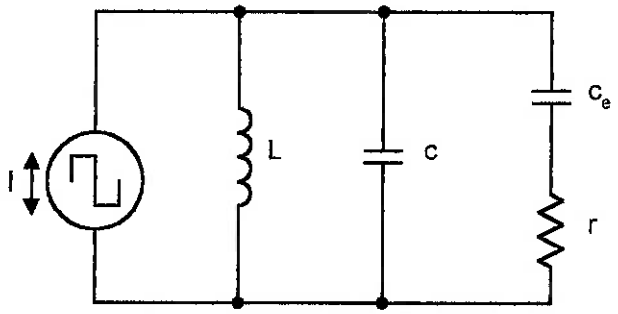


Figura B.4. Circuito equivalente do inversor ressonante alimentado por fonte de corrente com estabilização capacitiva.

sendo:

I: amplitude da corrente da fonte

L: indutância de magnetização do transformador Tr refletida para o enrolamento N_1 ou N_2

c: capacitância C refletida para o enrolamento N_1 ou N_2

c_e: capacitância de estabilização C_e refletida para o enrolamento N_1 ou N_2

r: resistência equivalente da lâmpada R refletida para o enrolamento N_1 ou N_2

Como este inversor apresenta alta eficiência, tem-se duas opções:

a) manter a potência nominal da lâmpada em 60Hz, aumentando-se o fluxo luminoso ou

b) manter o fluxo luminoso nominal, reduzindo-se a potência fornecida á lâmpada.

Para se utilizar o roteiro proposto por POSADAS, inicialmente devem ser conhecidos:

1. a tensão de alimentação (V_{cc});
2. a potência de operação da lâmpada (P);
3. a resistência da mesma quando em regime (R);
4. a freqüência de operação da lâmpada (f_d);
5. para estabilização capacitiva, o fator de crista (F_c), que é um fator que representa o quanto uma senóide está distorcida.

Abaixo se encontra o equacionamento apresentado por POSADAS em sua tese de mestrado.

A tensão de pico na tomada central do transformador, que é a tensão em N_1, é dada por:

$$V_{Np} = \frac{\pi}{2} \cdot V_{cc} \quad \text{Eq. B.1}$$

A resistência equivalente r_{eq} do modelo simplificado é dada por:

$$r_{eq} = \frac{(\pi \cdot V_{cc})^2}{8 \cdot P} \quad \text{Eq. B.2}$$

A resistência da lâmpada rebatida para o circuito principal, é relacionada com o número de espiras do transformador, sendo dada por:

$$r = \frac{R_{60}}{\left(\frac{N_4}{N_1}\right)^2} \quad \text{Eq. B.3}$$

O índice de mérito é dado por

$$Q = \sqrt{\frac{r_{eq}}{r} - 1} \quad \text{Eq. B.4}$$

Até este momento, estes cálculos se aplicam tanto para a estabilização indutiva como capacitiva. De agora em diante, será apresentada a formulação específica para estabilização capacitiva. Este tipo de estabilização foi eleito em detrimento à estabilização indutiva por ser mais fácil a utilização de capacitores (que são comerciais), do que indutores (que são feitos sob encomenda).

A capacitância de estabilização do modelo simplificado é dada por:

$$c_e = \frac{1}{2 \cdot \pi \cdot f_d \cdot r \cdot Q} \quad \text{Eq. B.5}$$

Refletindo-se para o rolamento N_4, tem-se que:

$$C_e = \frac{c_e}{\left(\frac{N_4}{N_1}\right)^2} \quad \text{Eq. B.6}$$

A capacitância equivalente do modelo simplificado, advinda da transformação série-paralelo, do ramo c_e – r da Figura B.4 é dada por:

$$c_{ep} = \frac{Q}{r_{eq} \cdot 2 \cdot \pi \cdot f_d} \quad \text{Eq. B.7}$$

A capacitância equivalente paralela às bobinas N_1 e N_2 é dada por:

$$c = \left(\frac{\pi}{F_c - \sqrt{2}} \cdot \sqrt{\frac{r}{8 \cdot r_{eq}} - 1} \right) \cdot c_{ep} \quad \text{Eq. B.8}$$

Desta forma, o valor do capacitor C é

$$C = \frac{c}{4} \quad \text{Eq. B.9}$$

Define-se como capacitância equivalente do modelo simplificado a associação em paralelo de c e c_{ep}, ou seja,

$$c_{eq} = c + c_{ep} \quad \text{Eq. B.10}$$

A indutância de magnetização L dos ramos N_1 e N_2 do transformador é dada por:

$$L = \frac{1}{c_{eq}} \cdot \frac{1}{(2 \cdot \pi \cdot f_d)^2 + \left(\frac{1}{2 \cdot r_{eq} \cdot c_{eq}}\right)^2} \quad \text{Eq. B.11}$$

O indutor L_f é um filtro para manter a corrente estável. Para seu dimensionamento é necessário inicialmente se estimar a variação de corrente ΔI.

A corrente I que passa pelo filtro L_f é:

$$I = \frac{\pi}{4} \cdot \sqrt{\frac{2 \cdot P}{r_{eq}}} \quad \text{Eq. B.12}$$

Adota-se um valor qualquer para ΔI (no projeto adotou-se 3% de I), e então a indutância L_f é:

$$L_f = 0{,}661 \cdot \frac{V_{cc}}{2 \cdot \pi \cdot f_d \cdot \Delta I} \quad \text{Eq. B.13}$$

A determinação dos resistores é tal que leve os transistores a trabalhar em corte e saturação.

A corrente de base I_b do transistor deve ser suficiente para levá-lo à saturação. Conforme fez POSADAS, adotando-se eficiência de 70% para o inversor, 75% do valor típico para o ganho h_{FE} do transistor e um coeficiente de segurança de 15% sobre o valor I_b calculado, tem-se:

$$I_b = 1{,}15 \cdot \frac{\frac{I}{0{,}7}}{0{,}75 \cdot h_{FE}} \quad \text{Eq. B.14}$$

Adotando-se o valor de pico do enrolamento N_3, dado por V_{N3p}, tem-se

$$V_{N_{3p}} = V_{ebmax} + V_j \quad \text{Eq. B.15}$$

Onde: V_{ebmax} é a máxima tensão reversa de emissor-base admissível

V_j é a tensão na junção base-emissor quando diretamente polarizada.

Desta forma as resistências são dadas por:

$$R_3 = \frac{V_{cc} + \left(\frac{V_{N_{3p}}}{\pi} - V_j \right)}{I_b} \quad \text{Eq. B.16}$$

$$R_1 = R_2 = 0{,}015 \cdot R_3 \quad \text{Eq. B.17}$$

As relações de espiras do transformador são dadas abaixo.

$$N_1 = N_2$$
$$N_5 = N_6$$
$$\frac{N_4}{N_1} = \frac{\text{tensão de pico em vazio do enrolamento } N_4}{V_{NP}} \quad \text{Eq. B.18}$$
$$\frac{N_1}{N_5} = \frac{V_{Np}}{\text{tensão de pico nos filamentos da lâmpada}}$$
$$\frac{N_1}{N_3} = \frac{V_{Np}}{V_{N_{3p}}}$$

A construção do transformador Tr deve obedecer aos seguintes critérios:

- o enrolamento de base N_3 deve ser o enrolamento mais próximo ao núcleo, devendo estar bem acoplado ao mesmo;
- os enrolamentos primários N_1 e N_2 são os próximos, devendo ser bifilares, a fim de evitar efeitos eletromagnéticos indesejáveis;
- os próximos enrolamentos devem ser os dos filamentos, N_5 e N_6;
- por último, o enrolamento mais externo deve ser o enrolamento secundário N_4.

Desta forma, estão totalmente apresentados os passos para o dimensionamento do circuito inversor da lâmpada fluorescente.

Anexo C. Circuito final

Nas próximas páginas encontra-se o circuito final do projeto e uma relação dos componentes utilizados e suas quantidades.

Abaixo tem-se uma descrição sucinta da função de cada pino utilizado do microcontrolador PIC16C74.

Pino no.	Nome do pino no manual	Função do pino no circuito
1	$\overline{MCLR}/V_{PP}$	Desligar o *reset*
2	RA0/AN0	Entrada analógica da leitura de corrente
3	RA1/AN1	Entrada analógica da leitura de tensão
11, 32	V_{DD}	Alimentação +5V
12, 31	V_{SS}	Terra
13	OSC1/CLKIN	Gerador de *clock*
14	OSC2/CLKOUT	Gerador de *clock*
15	RC0/T1OSO/T1CKI	*Clock* do TMR1
16	RC1/T1OSI/CCP2	*Clock* do TMR1
17	RC2/CCP1	Saída de PWM
18	RC3/SCK/SCL	Saída de controle do acendimento da lâmpada
25	RC6/TX/CK	Transmissão serial
26	RC7/RX/DT	Recepção serial

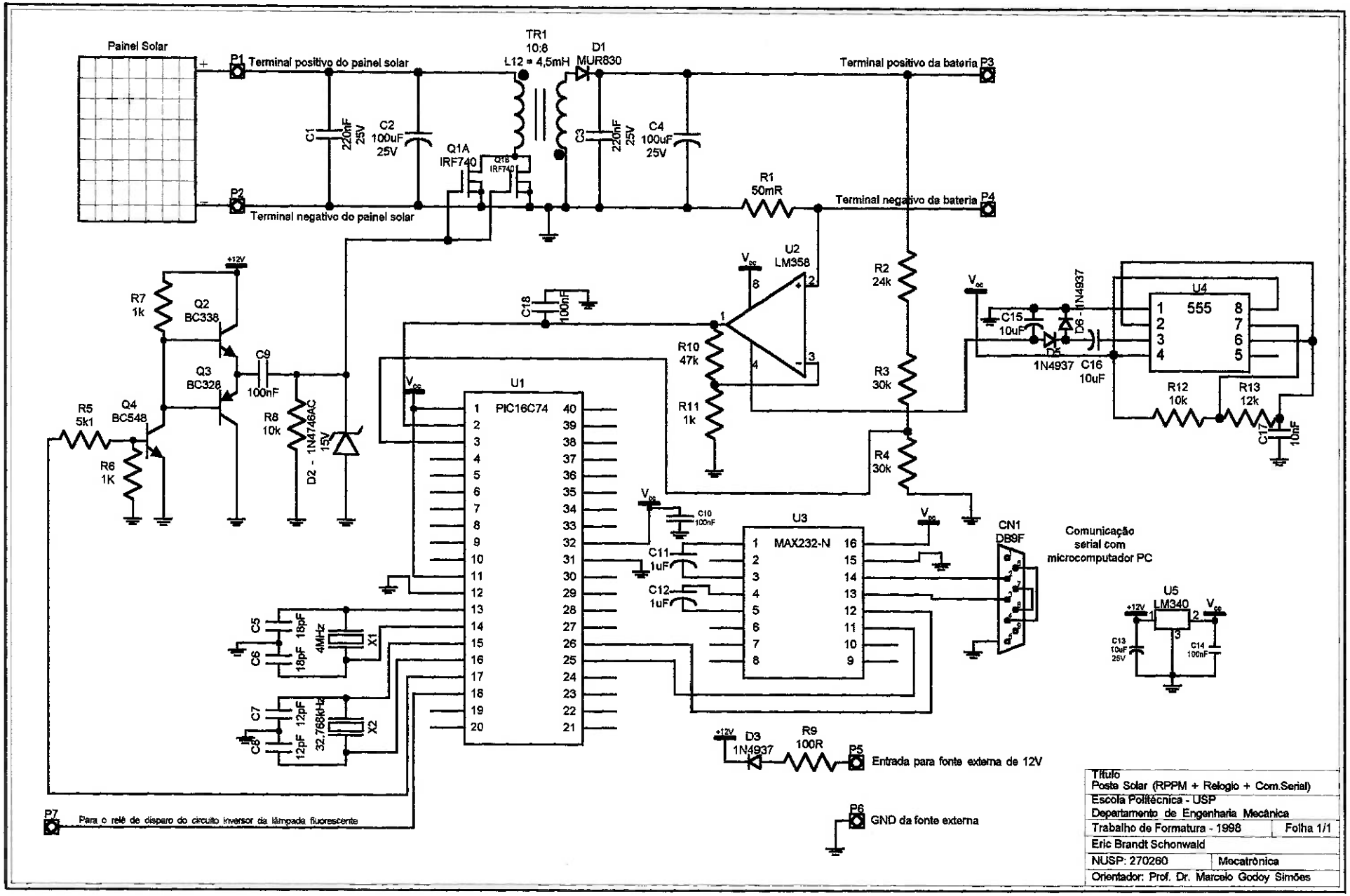
Tabela C.1. Descrição dos pinos utilizados do PIC16C74.

Finalmente, a tabela abaixo apresenta quais os componentes e suas quantidades utilizadas neste projeto.

Componente	Quantidade
Capacitor 12pF	2
Capacitor 18pF	2
Capacitor 10nF	1
Capacitor 100nF	4
Capacitor 220nF 25V	2
Capacitor 1μF	2
Capacitor 10μF	3
Capacitor 100μF 25V	2
Resistor 50mΩ	1
Resistor 100Ω	1
Resistor 1k	3
Resistor 5k1	1
Resistor 10k	2
Resistor 12k	1
Resistor 24k	1
Resistor 30k	2
Resistor 47k	1
Diodo de potência MUR830	1
Diodo zener 1N4746AC	1
Diodo 1N4937	3
MOSFET IRF740	2
Transistor BC338	1
Transistor BC328	1
Transistor BC548	1
Cristal 4MHz	1
Cristal 32.768kHz	1
Microcontrolador PIC16C74	1
LM358	1
MAX232-N	1
555	1
LM340	1
Transformador núcleo EE 55/20 Al=250	1
Fio AWG 22	

Tabela C.2. Relação dos componentes e suas quantidades.

Figura C.1. Circuito final do projeto Poste Solar (próxima página).


Painel Solar
P1 Terminal positivo do painel solar
P2 Terminal negativo do painel solar
TR1 10:6 L12 = 4,5mH
D1 MUR830
Terminal positivo da bateria P3
Terminal negativo da bateria P4
C1 220nF 25V
C2 100uF 25V
C4 100uF 25V
Q1A IRF740
Q1B IRF740
R1 50mR
U2 LM358
R2 24k
R3 30k
R4 30k
R10 47k
R11 1k
C18 100nF
+12V
R7 1k
Q2 BC338
Q3 BC328
C9 100nF
R5 5k1
Q4 BC548
R6 1K
R8 10k
D2 - 1N4746AC 15V
U1 PIC16C74
C5 18pF
C6 18pF
4MHz X1
C7 12pF
C8 12pF
32.768kHz X2
C10 100nF
C11 1uF
C12 1uF
U3 MAX232-N
CN1 DB9F
Comunicação serial com microcomputador PC
C15 10uF
D6 - 1N4937
D5 1N4937
C16 10uF
U4 555
R12 10k
R13 12k
C17 10nF
U5 LM340
C13 10uF 25V
C14 100nF
D3 1N4937
R9 100R
P5 Entrada para fonte externa de 12V
P6 GND da fonte externa
P7 Para o relé de disparo do circuito inversor da lâmpada fluorescente
Título
Posta Solar (RPPM + Relogio + Com.Serial)
Escola Politécnica - USP
Departamento de Engenharia Mecânica
Trabalho de Formatura - 1998
Folha 1/1
Eric Brandt Schonwald
NUSP: 270260
Mecatrônica
Orientador: Prof. Dr. Marcelo Godoy Simões

Anexo D. Codificação do *software* no microcontrolador PIC16C74

A fim de se entender melhor o programa final implementado no microcontrolador, o mesmo será dividido em três partes, que serão explicadas independentemente. Entretanto, como os três módulos estão baseados em interrupções, este módulo será apresentado primeiro.

D.1. Rotina de tratamento de interrupções

O fluxograma abaixo apresenta a rotina de tratamento de interrupções. TMR0 é o *timer* do RPPM (interrupção a cada 8,192ms). TMR1 é o *timer* do relógio (interrupção a cada 2s).

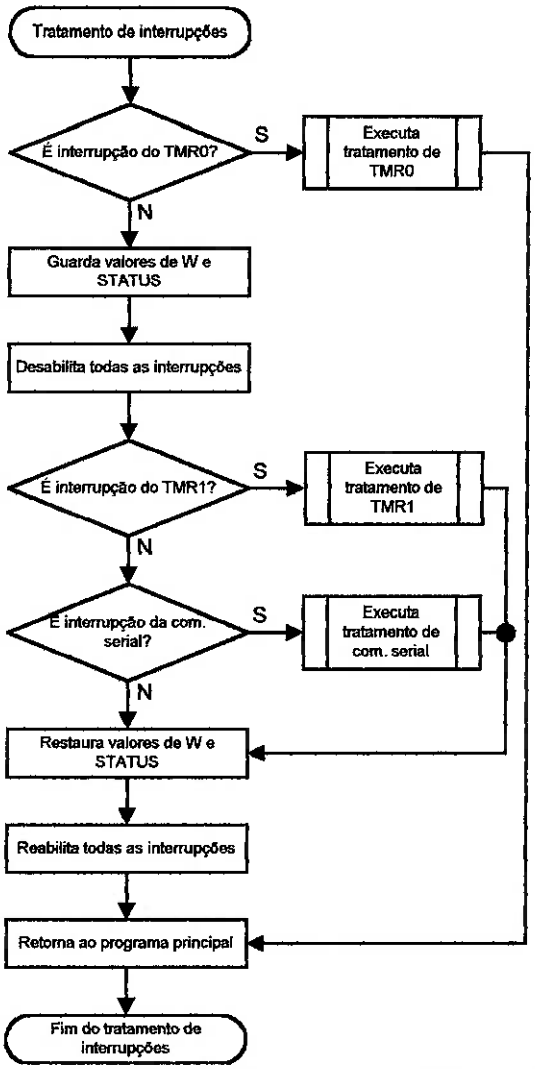


Figura D.1. Rotina de tratamento de interrupções

Neste fluxograma deve-se salientar que a rotina que trata TMR0 está em uma configuração um pouco diferente das demais pois esta interrupção ocorre com uma grande freqüência e o seu tratamento é bastante simples. Desta forma, se se interrompesse as outras interrupções, haveria dificuldades com a comunicação serial bem como um atraso do relógio, conforme observou-se empiricamente.

D.2. Relógio de tempo real

A codificação do relógio dentro do programa principal restringe-se apenas às configurações iniciais e portanto não será apresentada em forma de fluxograma. Por outro lado, a rotina de interrupção do TMR1 é que corresponde ao funcionamento do relógio propriamente dito. Seu fluxograma encontra-se abaixo.

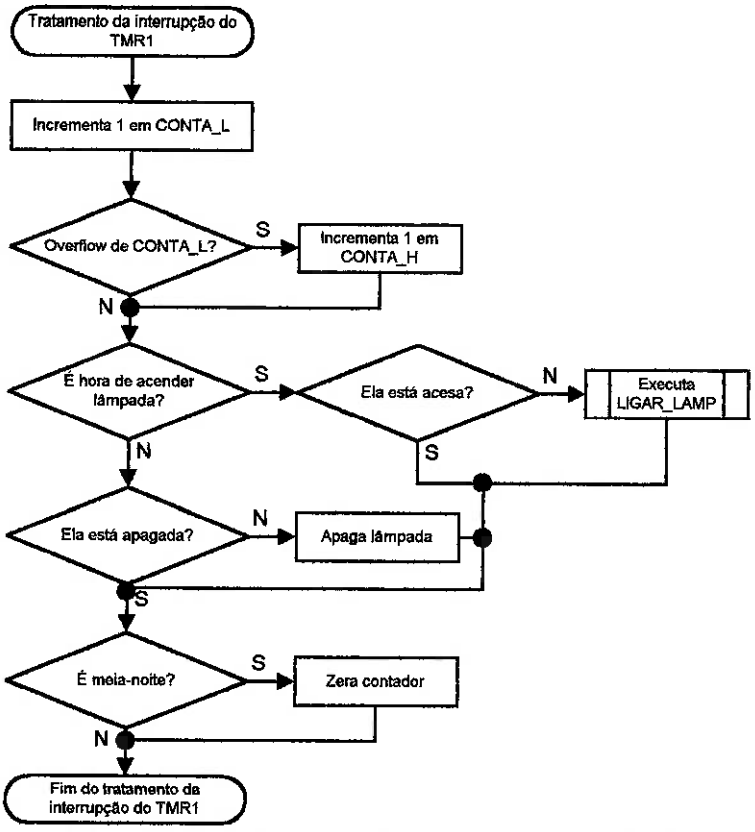


Figura D.2. Tratamento de interrupção do relógio

A rotina LIGAR_LAMP verifica se a tensão da bateria, no momento de acender a lâmpada é maior que V_MIN (utilizou-se V_MIN proporcional a 11V). Se for, a

lâmpada é acesa. Caso contrário, entende-se que a bateria está descarregada e então a lâmpada não será acesa, a fim de se evitar que a bateria se descarregue por completo. O fluxograma de LIGAR_LAMP se encontra abaixo.

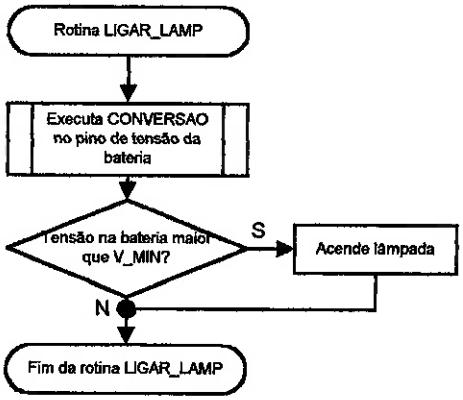


Figura D.3. Rotina de acendimento da lâmpada.

A respeito da rotina apresentada acima, o acendimento da lâmpada se dá através do bit 3 do PORTC do microcontrolador utilizado (pino 18). Quando PORTC<3> está em nível *low*, a lâmpada está apagada. Quando PORTC<3> está em nível *high*, acende-se a lâmpada. Isto ocorre através do chaveamento de um relê existente no circuito inversor da lâmpada fluorescente. A rotina CONVERSAO apresentada, será explicada na parte do RPPM. Esta rotina realiza a conversão A/D de um pino especificado antes de sua chamada.

V_MIN é definida como uma constante nas primeiras linhas do cabeçalho. Seu valor pode ser obtido através de simples regra de três. O divisor resistivo R2-R3-R4 do circuito foi dimensionado para uma tensão de 5V no pino de leitura de tensão da bateria quando a mesma tiver 14V. Quando ela tiver 11V, a tensão neste pino será de 3,93V. Na conversão A/D, 5V correspondem a FFh. Portanto 3,93V corresponderão a **C9h**. Valor que pode ser observado na listagem do programa.

A verificação de se é hora de acender a lâmpada é feita através da seguinte lógica: Se a hora atual é maior que a hora de ligar e menor que a hora de desligar então pode-se concluir que deve-se acender a lâmpada. Como o microcontrolador não possui

comandos de verificação de maior ou menor, criou-se uma macro[3] chamada SE_AMENORB. Este macro é chamada pelos parâmetros BYTE1, BYTE2 e LUGAR. Se BYTE1 < BYTE2, vai para lugar, senão segue para próxima linha. Observe abaixo a lógica desta macro.

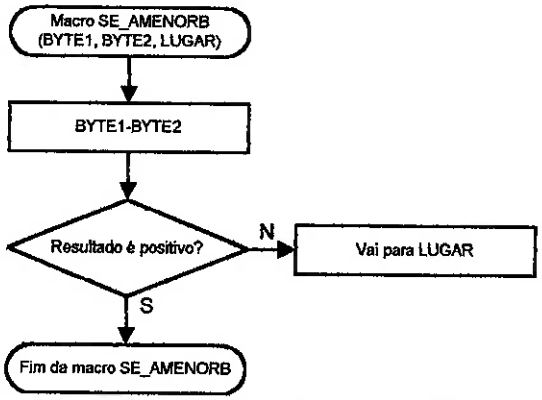


Figura D.4. Macro SE_AMENORB.

D.3. Comunicação serial

A fim de que o microcontrolador possa se comunicar com o computador, definiu-se dois códigos de operação, que o computador envia ao PIC após solicitação do usuário:

1. 02h significa que o microcontrolador irá receber os novos horários.
2. 04h significa que o microcontrolador deve enviar os horários para o computador.

Da mesma maneira que o relógio, toda a rotina da comunicação ocorre dentro da rotina de tratamento de interrupções, tendo no programa principal apenas as inicializações e configurações necessárias. Desta forma só é necessário se observar o fluxograma da interrupção de comunicação serial.

[3] Macro, na programação do PIC16C74 é uma seqüência de linhas de programação que são expandidas durante a compilação no lugar em que se chamou a macro. Isto é bastante útil para seqüências de linhas repetidas.

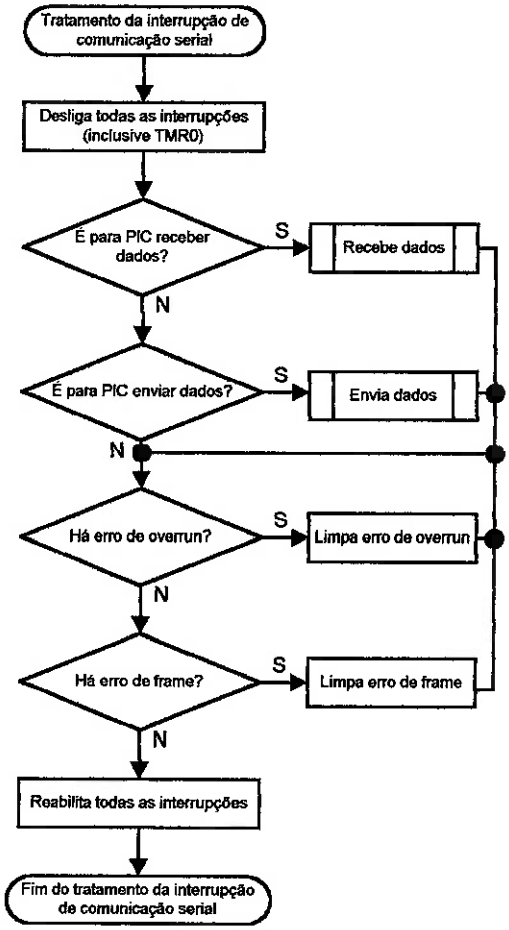


Figura D.5. Rotina de tratamento de interrupção da comunicação serial.

Abaixo pode-se encontrar as rotinas de envio e recepção de dados de forma detalhada. Percebe-se que elas são divididas em dois níveis. O primeiro indica com qual byte se está trabalhando e o segundo são as macros ENVIA e RECEBE que cuidam de maneira propriamente dita da comunicação.

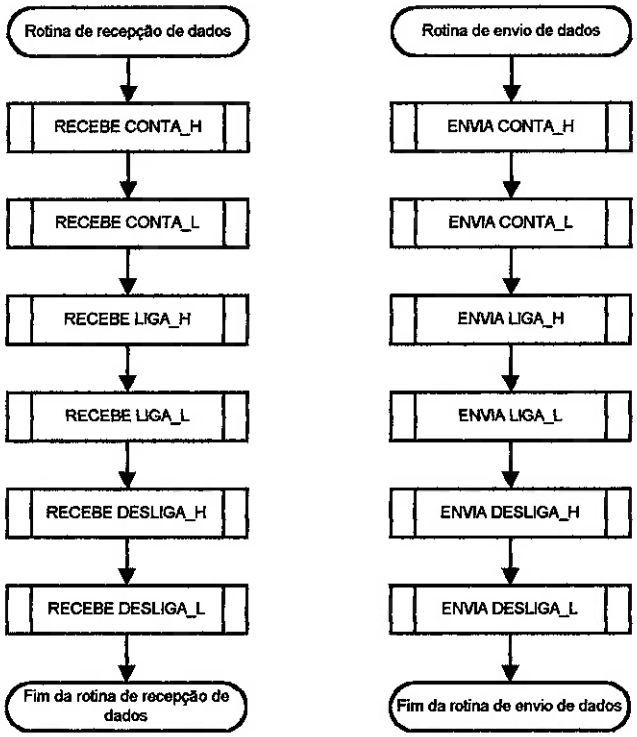


Figura D.6. Rotinas de envio e recepção de dados.

Abaixo tem-se as macros utilizadas na comunicação serial.

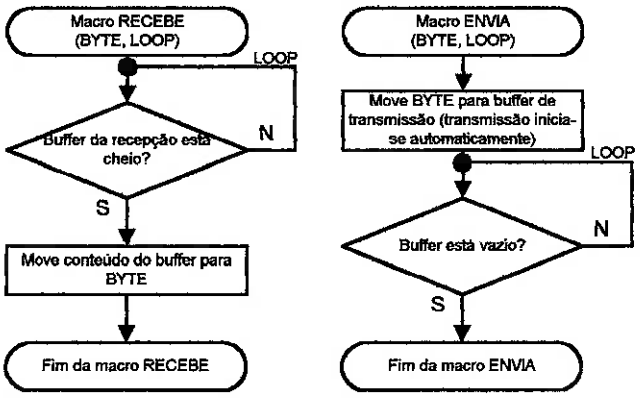


Figura D.7. Macros de comunicação serial.

Percebe-se que nas macros acima, é necessário passar o nome de um *label* para direcionar o loop de *polling*. Este nome deve ser um nome qualquer, não existente no programa e nunca deve se repetir. "Convencionou-se" utilizar os nomes AA,BB,CC, etc...

O *software* de comunicação do lado do computador encontra-se no Anexo E.

D.4. Rastreador do Ponto de Potência Máxima

Uma visão geral do *software* rastreador já foi apresentada no capítulo 7.3. Abaixo encontra-se este algoritmo de forma um pouco mais detalhada.

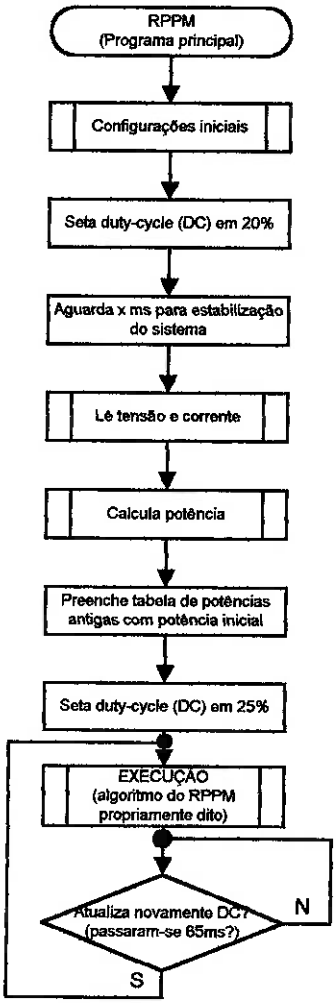


Figura D.8. Fluxograma básico do RPPM.

Como já foi dito anteriormente, a potência calculada é constituída de dois bytes. Para se armazenar os últimos 32 valores medidos (64 bytes) utilizou-se o recurso de endereçamento indireto, oferecido pelo PIC (registradores FSR e INDF). Desta forma utiliza-se os endereços de memória de 0xB0 a 0xEF, onde os endereços 0xB0, 0xB2, etc são os bytes mais significativas e os 0xB1, 0xB3, etc os menos significativos. O armazenamento na memória é rotativo, ou seja, as primeiras 32 medidas ocupam os 64 endereços, já a 33ª medida ocupará o lugar da primeira, a 34ª o da segunda, etc.

Para se saber se já é o momento de se atualizar o DC com uma nova medida (após 65,536ms) existe um contador de interrupções do TMR0. A cada interrupção (8,192ms), o contador é decrementado. Quando o contador for igual a zero, atribui-se novamente o valor 8 (valor da constante TOTAL_OVERS) a este contador e reinicia-se o processo EXECUCAO.

Abaixo expande-se EXECUCAO.

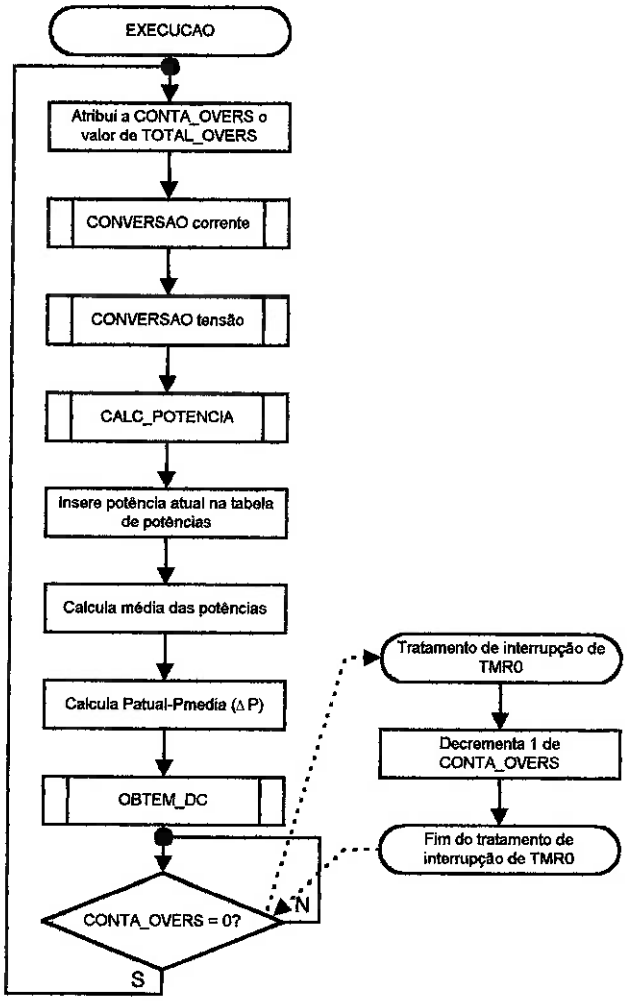


Figura D.9. Bloco principal do RPPM

Por fim , só resta esmiuçar as rotinas CONVERSAO (que executa a conversão A/D), o bloco CALC_POTENCIA (que realiza o produto tensão x corrente) e a rotina OBTEM_DC, conforme observa-se abaixo.

Para se chamar a rotina CONVERSAO, deve-se atribuir valores para a variável de trabalho do microcontrolador W, pois o pino desejado é obtido através de uma operação OR dentro da rotina. Para se definir o pino AN0 (leitura de corrente, pino 2), W=00h. Para pino AN1 (tensão, pino 3), W=08h.

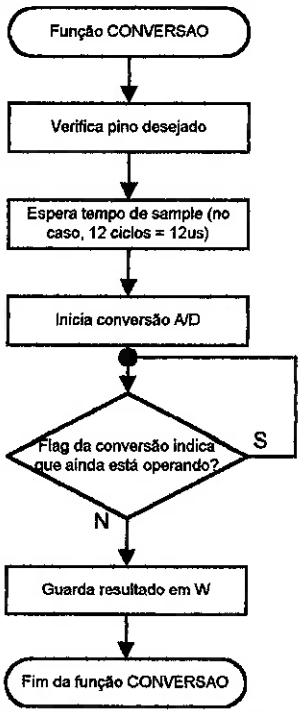


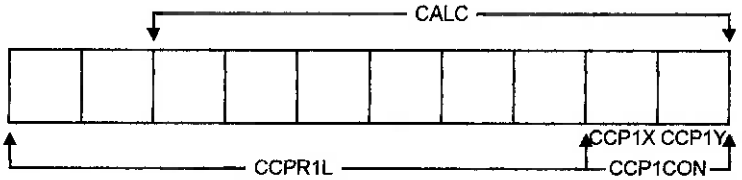
Figura D.10. Rotina de conversão A/D.

A rotina CALC_POTENCIA é puramente matemática e portanto não será apresentada aqui na forma de fluxograma. Entretanto pode-se ver esta rotina na listagem o fim deste capítulo.

Por fim, só resta apresentar o fluxograma da rotina OBTEM_DC, que observa a variação de potência e de PWM e decide se aumenta ou diminui o *duty-cycle* do mesmo. Esta rotina trabalha com duas variáveis: CALC e CALC2. CALC, no início possui o valor atual do *duty-cycle* e na saída possui o novo valor. Da variável CALC2, utiliza-se apenas o bit 0 para representar se a diferença de potência é positiva ou negativa.

Utilizou-se a variável CALC ao invés de diretamente o registrador CCPR1L (que guarda o valor do *duty-cycle*) pois o PWM do microcontrolador pode utilizar até

10bits, que são os 8 de CCPR1L e mais dois que existem no registrador CCP1CON, sendo que estes dois últimos representam os bits menos significativos. Decidiu-se utilizar os 10 bits a fim de se aumentar a precisão do algoritmo. Como o máximo *duty-cycle* possível para este projeto possui 128 divisões, conforme pode ser visto no item 7.3, então é possível guardar todos os bits significativos em um único registrador. Esta é a função da variável CALC. Além disto a função também utiliza a variável DELTA, que vale 1 e é o incremento do *duty-cycle*.



Com isto posto, abaixo encontra-se o fluxograma da função OBTEM_DC.

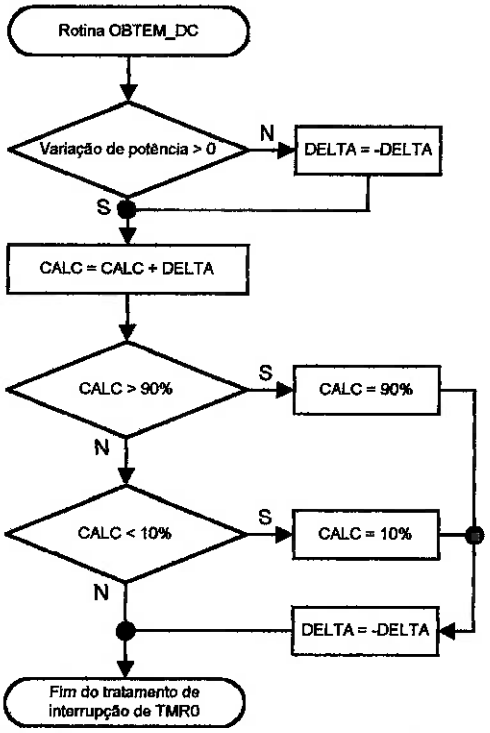


Figura D.11. Rotina que obtém valor do *duty-cycle*.

Para se fazer um número negativo, basta pegar o seu complemento e somar 1.

Para se determinar uma porcentagem de *duty-cycle* (10% e 90%), basta relembrar a fórmula apresentada no item 7.3.

$$DC_{PWM\,(\%)} = \frac{CALC}{128}$$

Desta forma para DC=10%, CALC = 12,8 = 00001101b e DC=90%, CALC = 115,2 = 01110011b, conforme pode se observar no programa.

Na próxima folha tem-se a listagem completa do programa.

```
;Universidade de Sao Paulo - Escola Politecnica - Depto. Eng. Mecanica
;Trabalho de Formatura - Projeto Poste Solar
;Eric Brandt Schonwald - NUSP: 270260
;Orientador: Marcelo Godoy Simoes
;Dezembro de 1998
;Arquivo: PSOLAR.ASM  -  Microcontrolador PIC16C74 da Microchip
;Compilado no MPASM 1.21

;Este programa tem as seguintes funcoes:
;   - Ratreador do Ponto de Potencia Maxima (RPPM)
;   - Relogio de Tempo Real
;   - Comunicacao Serial com Microcomputador PC
;
;Cristal primario 4MHz

;Rastreador do Ponto de Potencia Maxima (RPPM)
;   Interrupcao do TMR0 a cada 8,192ms
;   Atualizacao do PWM a cada 65,536ms
;   Media das ultimas 32 potencias medidas
;   Periodo do PWM: 32us

;Relogio de Tempo Real
;   Cristal secundario 32.768kHz
;   Interrupcao do TMR1 a cada 2s

;Comunicacao Serial
;   Comunicacao assincrona
;   1200 bauds por segundo, 8 bits, sem paridade
;   1 start bit, 1 stop bit, sem handshaking


LIST P=16C74            ;Microcontrolador utilizado
INCLUDE "P16CXX.INC"    ;Arquivo padrao de declaracoes

;*****************************************************************
;                   DECLARACAO DE CONSTANTES
;*****************************************************************

#DEFINE TOTAL_OVERS 8   ;Atualizacao do PWM a cada TOTAL_OVERS
                        ;   interrupcoes do TMR0
#DEFINE MAX_FSR 0xF0    ;Maximo endereco da tabela de potencias medi-
                        ;   das
#DEFINE V_MIN 0xC9      ;No minimo bateria com 11V para ligar a
lampada
                        ;FFh corresponde a 14V

;*****************************************************************
;                   DECLARACAO DE VARIAVEIS
;*****************************************************************


    RCIF equ 5

    W_TEMP      equ 0x60    ;guarda W para posterior recuperacao
    W_TEMP1     equ 0xE0
    STATUS_TEMP equ 0x61    ;guarda STATUS para posterior recuperacao

    CBLOCK  0x30

        ;Variaveis do RPPM
        CALC                ;contador na rotina principal
```

```
        CALC2               ;auxiliar nos somatorios em que um byte
                            ;extra eh preciso
        CALC3               ;contador nas subrotinas
        TENSAO_BAT, CORR_BAT  ;tensao e corrente na bateria apos A/D
        POT_H, POT_L        ;potencia
        MEDIA_H, MEDIA_L    ;media das potencias antigas
        DIF_H, DIF_L        ;diferenca entre pot atual e media
        CONTA_OVERS         ;conta overflows do TMR0
        DELTA               ;incremento do PWM
        GUARDA_FSR          ;ponteiro de endereco da tabela de
potencias

        ;Variaveis do relogio
        CONTA_H,CONTA_L     ;horario atual
        DIACHEIO_H, DIACHEIO_L  ;meia-noite
        LIGA_H, LIGA_L, DESLIGA_H, DESLIGA_L ;acendimento e
desligamento
                                ; da lampada

        ;Variaveis da comunicacao serial
        RECEBIDO            ;palavra de controle da comunicacao serial
        ;ENVIADO, CONTROLE
        ;CONTADOR, AUX1, AUTORIZADO
        ;LIGAPWM_H, LIGAPWM_L, DESLIGAPWM_H, DESLIGAPWM_L

        ;Variaveis para acendimento da lampada
        GUARDA_TENSAO   ;tensao na bateria no momento do acendimento
        TENSAO_MIN      ;tensao minima para acendimento

    ENDC

;*****************************************************************
;                               MACROS
;*****************************************************************

;---------------------------------------------------------------
;Macro de recepcao dos dados
;---------------------------------------------------------------
RECEBE MACRO BYTE, LOOP
LOOP    BTFSS PIR1, RCIF        ;Quando RCIF=1 significa que PIC
            GOTO LOOP           ;recebeu o byte. Desta forma,
       MOVF RCREG, W            ;guarda em BYTE.
       MOVWF BYTE
ENDM

;---------------------------------------------------------------
;Macro de envio dos dados
;---------------------------------------------------------------
ENVIA MACRO BYTE, LOOP
        MOVF BYTE, W
        MOVWF TXREG
        BSF STATUS, RP0         ;Banco 1
LOOP    BTFSS TXSTA, TRMT       ;Quando TRMT=1, PIC terminou de enviar
            GOTO LOOP
        BCF STATUS, RP0         ;Banco 0
ENDM

;---------------------------------------------------------------
;Macro que direciona o programa para LUGAR, se BYTE=VALOR
;---------------------------------------------------------------
SEIGLIT MACRO BYTE, VALOR, LUGAR
```

```
    MOVLW VALOR
    SUBWF BYTE, W
    BTFSC STATUS, Z
        GOTO LUGAR
    NOP
ENDM

;-------------------------------------------------------------------
;Macro que direciona o programa para LUGAR, se BYTE1 < BYTE2
;-------------------------------------------------------------------
SE_AMENORB MACRO BYTE1, BYTE2, LUGAR
    MOVF BYTE2, W
    SUBWF BYTE1, W      ;W=BYTE1-BYTE2
    BTFSS STATUS, C     ;Se resultado positivo(BYTE1>BYTE2) continua
        GOTO LUGAR      ;Se negativo, vai para LUGAR
    NOP
ENDM

;*******************************************************************
;                        INICIO DO PROGRAMA
;*******************************************************************
    ORG 0x000           ;Endereco inicial
    GOTO START          ;Inicia prograama em START

;*******************************************************************
;                    TRATAMENTO DE INTERRUPCAO
;*******************************************************************
    ORG 0x004           ;Endereco do ponteiro de interrupcoes

INTERRUPCOES
        BTFSC INTCON, T0IF  ;interrupcao do TMR0
            GOTO T0_OVFL

        MOVWF W_TEMP        ;bloco que armazena valores de W e STATUS
        SWAPF STATUS, W     ;a fim de evitar que voltem alterados para
        BCF STATUS, RP0     ;o programa principal
        MOVWF STATUS_TEMP

LOOP_INT
        BCF INTCON, GIE     ;bloco que garante que as interrupcoes
        BTFSC INTCON,GIE    ;realmente estao desabilitadas
        GOTO LOOP_INT

        BTFSC PIR1, TMR1IF  ;interrupcao do TMR1
            GOTO T1_OVFL

        BTFSC PIR1, RCIF    ;interrupcao de recepcao serial
            GOTO RECEBEU

ERROR1                      ;entra aqui se foi interrupcao por outro
        GOTO FIM_INT

;-------------------------------------------------------------------
;Interrupcao do TMR0  - Algoritmo de busca
;-------------------------------------------------------------------
T0_OVFL
        BCF INTCON, T0IF    ;zera flag da interrupcao
        DECF CONTA_OVERS, F ;decremente CONTA_OVERS
        RETFIE              ;retorna para programa principal

;-------------------------------------------------------------------
```

```
;Interrupcao do TMR1  -  Relogio
;-----------------------------------------------------------------
T1_OVFL
        BCF PIR1, TMR1IF     ;zera flag da interrupcao
        INCFSZ CONTA_L, F    ;soma 1 em CONTA_L. Se nao estourar (FF)
            GOTO LIGA_DESLIGA ;vai para LIGA_DESLIGA
        INCF CONTA_H, F      ;CONTA_L estourou. Soma 1 em CONTA_H

LIGA_DESLIGA          ;bloco que verifica se eh hora de ligar lampada
    ;Se CONTA_H>=LIGA_H E CONTA_L>=LIGA_L E CONTA_H<DESLIGA_H E
    ;   CONTA_L<DESLIGA_L entao verifica PORTC, 3 senao vai para
    ;   DESLIGAR
        SE_AMENORB CONTA_H, LIGA_H, DESLIGAR
        SE_AMENORB CONTA_L, LIGA_L, DESLIGAR
        SE_AMENORB DESLIGA_H, CONTA_H, DESLIGAR
        SE_AMENORB DESLIGA_L, CONTA_L, DESLIGAR
        BTFSS PORTC, 3  ;Se eh hora de ligar e esta desligada chama
            CALL LIGAR_LAMP      ;subrotina LIGAR_LAMP
        NOP
        GOTO FIM_INT_TMR1

DESLIGAR
        BTFSC PORTC, 3  ;Se nao eh hora de estar ligada e estiver
            BCF PORTC, 3     ;ligada, entao desliga
        GOTO FIM_INT_TMR1

FIM_INT_TMR1
        ;Se CONTA_H>=DIACHEIO_H E CONTA_L>=DIACHEIO_L entao passou
      ;de meia-noite, entao zera. Senao vai para FIM_INT
        SE_AMENORB CONTA_H, DIACHEIO_H, FIM_INT
        SE_AMENORB CONTA_L, DIACHEIO_L, FIM_INT
        CLRF CONTA_H         ;zera os contadores
        CLRF CONTA_L
        GOTO FIM_INT

;-----------------------------------------------------------------
;Interrupcao de Recepcao Serial
;-----------------------------------------------------------------
RECEBEU
        BSF STATUS, RP0      ;Banco 1
        BCF PIE1, RCIE       ;desabilita interrupcao recepcao
        BCF PIE1, TMR1IE     ;desabilita interrupcao do TMR1
        BCF INTCON, T0IE     ;desabilita interrupcao do TMR0
        BCF STATUS, RP0      ;Banco 0

        MOVF RCREG, W        ;Verifica qual a palavra de controle
        MOVWF RECEBIDO       ;recebeu do computador

        ;Se foi 02h siginifica que eh para receber novos horarios
        SEIGLIT RECEBIDO, B'00000010', RECEBE_HOR
        ;Se foi 04h siginifica que eh para enviar horarios para PC
        SEIGLIT RECEBIDO, B'00000100', ENVIA_HOR
        ;Se chegou ate aqui significa erro da comunicacao serial
        GOTO SER_ERRO

SER_ERRO
        GOTO FIM_RECEBE

RECEBE_HOR
    ;Bloco que recebe os seis bytes necessarios para o funcionamento
do
```

```
    ;relogio/lampada. Os nomes AA,BB,... sao quaisquer inexistentes
    ;no programa. Server para garantir loop de polling e nao podem se
    ;repetir.
        RECEBE CONTA_H, AA
        RECEBE CONTA_L, BB
        RECEBE LIGA_H, CC
      RECEBE LIGA_L, DD
        RECEBE DESLIGA_H, EE
        RECEBE DESLIGA_L, FF
        GOTO FIM_RECEBE

ENVIA_HOR
    ;Bloco que envia os seis bytes. Os nomes HH,II,... sao para
garantir
    ;loop de polling dentro da macro.
          ENVIA CONTA_H, HH
          ENVIA CONTA_L, II
          ENVIA LIGA_H, JJ
          ENVIA LIGA_L, KK
          ENVIA DESLIGA_H, LL
          ENVIA DESLIGA_L, MM
          GOTO FIM_RECEBE

FIM_RECEBE
        BTFSC RCSTA, OERR       ;Houve erro de overrun (sobreposicao)
?
            GOTO LIMPA_OERR
      BTFSC RCSTA, FERR        ;Houve erro de frame?
            GOTO LIMPA_FERR
        GOTO FIM_INT_SER        ;Se nao ha nenhum erro sai da
interrup.
LIMPA_OERR                      ;Limpa erro de overrun
        BCF RCSTA, CREN
        BSF RCSTA, CREN
        GOTO FIM_RECEBE
LIMPA_FERR                      ;Llimpa erro de frame
        MOVF RCREG, W
        GOTO FIM_RECEBE

FIM_INT_SER
        BSF STATUS, RP0         ;Banco 1
        BSF PIE1, RCIE          ;reabilita interrupcao de recepcao
        BSF PIE1, TMR1IE        ;reabilita interrupcao do TMR1
        BSF INTCON, T0IE        ;reabilita interrupcap do TMR0
        BCF STATUS, RP0         ;Banco 0
        GOTO FIM_INT

FIM_INT
        SWAPF STATUS_TEMP, W    ;recupera valores de W e STATUS
originais
        MOVWF STATUS
        SWAPF W_TEMP, F
        SWAPF W_TEMP, W
        RETFIE                  ;Sai das interrupcoes, habilitando
todas.

;*****************************************************************
;                        PROGRAMA PRINCIPAL
;*****************************************************************

;-----------------------------------------------------------------
```

```
;Inicializacoes
;-----------------------------------------------------------------------
START                       ;Inicio do programa principal

;Inicializacoes do TMR1
        CLRF STATUS         ;Banco 0
        BCF T1CON, TMR1ON   ;Desliga TMR1

        CLRF TMR1H          ;Zera TMR1
        CLRF TMR1L
        CLRF CONTA_H        ;Zera contador de interrupcoes
        CLRF CONTA_L
        CLRF LIGA_H
        CLRF LIGA_L
        CLRF DESLIGA_H
        CLRF DESLIGA_L
        MOVLW 0xA8          ;Um dia tem A8C0h vezes 2 segundos
        MOVWF DIACHEIO_H
        MOVLW 0xC0
        MOVWF DIACHEIO_L

        CLRF INTCON         ;Desabilita todas as interrupcoes
        CLRF PIR1           ;Nao ocorreu nehuma interrupcao

        BSF STATUS, RP0     ;Banco 1
        CLRF PIE1           ;Desabilita todas as interrupcoes

        MOVLW B'11110011'   ;PORTC como entrada para oscilador, saida
de
        MOVWF TRISC             ;PWM e pino da lampada

        BCF STATUS, RP0     ;Banco 0
        BCF PORTC, 3        ;PORTC<3>=1 - liga lampada
                            ;PORTC<3>=0 - desliga lampada
        CLRF PIR1           ;Nao ocorreu nenhuma interrupcao
        BSF INTCON, PEIE    ;Habilita todas as interrupcoes de
perifericos
                            ;   nao mascaradas
        BSF INTCON, GIE     ;Habilita todas as interrupcoes nao
mascaradas

        MOVLW B'00001110'   ;TMR1: Prescaler 1, Oscilador enabled,
        MOVWF T1CON         ;   Assincrono, Clock Externo no pino
RC0/TCKI
        BSF T1CON, TMR1ON   ;Liga TMR1

; Inicializacoes da comunicacao serial
        BSF STATUS, RP0     ;Banco 1
        MOVLW 0xCF          ;x=207d=CFh   B.R.=1200 bauds
      MOVWF SPBRG
        MOVLW B'00100110'   ;transm. assincrona, 8 bits
        MOVWF TXSTA         ;modo rapido, habilitada

        BCF STATUS, RP0     ;Banco 0
        MOVLW B'10010000'   ;recep. em 8 bits, habilitada
        MOVWF RCSTA         ;pinos RC6/7 como pinos de com. serial

        BSF STATUS, RP0     ;Banco 1
        BSF PIE1, TMR1IE    ;Habilita interrupcao do TMR1
        BSF PIE1, RCIE      ;Habilita interrupcao da com. serial
        BCF STATUS, RP0     ;Banco 0
```

```
; Inicializacoes do RPPM
        CLRF TMR0           ;zera TMR0
        BSF INTCON, T0IE    ;Habilita interrupcao TMR0
        CLRF CCPR1L         ;Zera saida de PWM
        MOVLW B'00001100'   ;Modo PWM
        MOVWF CCP1CON
      BSF STATUS, RP0     ;Banco 1
        MOVLW B'11010100'   ;prescaler 1:32; repeticao - 8,192ms
        MOVWF OPTION_REG
        CLRF ADCON1         ;portas A e E sao so analogicas
        MOVLW B'00011111'
        MOVWF PR2           ;32us - periodo do pulso PWM
        BCF STATUS, RP0     ;Banco 0
        MOVLW B'00000001'   ;DELTA = 1
        MOVWF DELTA
        MOVLW 0xB0          ;Tabela de potencias comeca em 0xB0
        MOVWF FSR

MED_INICIAIS
        BSF CCP1CON, CCP1X  ;PWM com 10 bits para aumentar precisao
        MOVLW B'00000110'   ;Duty-cycle = (CCPR1L + CCP1X + CCP1Y) /
128
        MOVWF CCPR1L        ;DC das medidas iniciais: 20%
        BSF T2CON, TMR2ON   ;aqui comecam os pulsos
        MOVLW D'12'         ;98,304ms de espera para estabilizar
sistema
        MOVWF CONTA_OVERS
ESTABILIZACAO               ;estabilizacao
        MOVF CONTA_OVERS, F
        BTFSS STATUS, Z
            GOTO ESTABILIZACAO

        CLRW                    ;pino AN0 - corrente
        CALL CONVERSAO
        MOVWF CORR_BAT
        MOVLW B'00001000'       ;pino AN1 - tensao
        CALL CONVERSAO
        MOVWF TENSAO_BAT
        CALL CALC_POTENCIA      ;calcula potencia

PREENCHE                        ;preenche tabela de potencias
        MOVF POT_H, W
        BSF STATUS, RP0         ;como os enderecos estao no banco 1
        MOVWF INDF              ;eh necessario seta RP0
        BCF STATUS, RP0
        INCF FSR, F             ;proximo endereco
      MOVF POT_L, W
        BSF STATUS, RP0
        MOVWF INDF
        BCF STATUS, RP0
        INCF FSR, F
        MOVF FSR, W         ;verifica se atingiu ultimo endereco da
tabela
        SUBLW MAX_FSR
        BTFSS STATUS, Z
            GOTO PREENCHE
        MOVLW 0xB0          ;retorna ao primeiro endereco da tabela
        MOVWF FSR
        BCF CCP1CON, CCP1X
        MOVLW B'00001000'
```

```
        MOVWF CCPR1L        ;DC inicial de 25%

;aqui comeca o loop principal do RPPM
EXECUCAO
        MOVLW TOTAL_OVERS
        MOVWF CONTA_OVERS
      CLRF CALC2
        CLRF MEDIA_H
        CLRF MEDIA_L
        CLRW                ;le pino AN0 - corrente
        CALL CONVERSAO
        MOVWF CORR_BAT
        MOVLW B'00001000'   ;le pino AN1 - tensao
        CALL CONVERSAO
        MOVWF TENSAO_BAT
        CALL CALC_POTENCIA  ;calcula potencia
        MOVF POT_H, W
        BSF STATUS, RP0
        MOVWF INDF          ;insere na tabela
        BCF STATUS, RP0
        INCF FSR, F
        MOVF POT_L, W
        BSF STATUS, RP0
        MOVWF INDF
        BCF STATUS, RP0
      INCF FSR, F
        MOVF FSR, W
        SUBLW MAX_FSR
        BTFSC STATUS, Z     ;se ultimo endereco da tabela
            GOTO ZERA_FSR   ;vai para ZERA_FSR
        GOTO FSR_OK         ;senao vai para FSR_OK
ZERA_FSR
        MOVLW 0xB0          ;retorna ao primeiro endereco da tabela
        MOVWF FSR
FSR_OK
        MOVF FSR, W
        MOVWF GUARDA_FSR    ;guarda posicao atual do ponteiro
        MOVLW 0xB0          ;primeiro endereco da tabela
        MOVWF FSR
        CLRF CALC2          ;zera variavel auxiliar (3o. byte)

TIRA_MEDIA_POT
        BSF STATUS, RP0
        MOVF INDF, W        ;obtem valor da tabela
      BCF STATUS, RP0
        ADDWF MEDIA_H, F    ;MEDIA_H = MEDIA_H + POT_TABELA_H
        BTFSC STATUS, C     ;se overflow, soma 1 em CALC2
            INCF CALC2, F
        INCF FSR, F         ;proximo endereco
        BSF STATUS, RP0
        MOVF INDF, W        ;obtem valor da tabela
        BCF STATUS, RP0
        ADDWF MEDIA_L, F    ;MEDIA_L = MEDIA_L + POT_TABELA_L
        BTFSC STATUS, C     ;se overflow, soma 1 em MEDIA_H
            INCF MEDIA_H, F
        BTFSC STATUS, Z
            INCF CALC2, F   ;se overflow, soma 1 em CALC2
        INCF FSR, F
        MOVF FSR, W
        SUBLW MAX_FSR
```

```
        BTFSS STATUS, Z     ;enquanto nao percorrer os 32 enderecos
volta
            GOTO TIRA_MEDIA_POT ;para o inicio do bloco

      RRF CALC2, F                    ;dividindo por 32...
        RRF MEDIA_H, F
        RRF MEDIA_L, F
        RRF CALC2, F
        RRF MEDIA_H, F
        RRF MEDIA_L, F
        RRF CALC2, F
        RRF MEDIA_H, F
        RRF MEDIA_L, F
        RRF CALC2, F
        RRF MEDIA_H, F
        RRF MEDIA_L, F
        RRF CALC2, F
        RRF MEDIA_H, F                  ;media calculada
        RRF MEDIA_L, F

;aqui inicia-se o ajuste do DC
        RLF CCPR1L, W           ;bloco que armazena PWM de 10 bits na
        MOVWF CALC              ;variavel CALC
      BCF CALC, 0
        RLF CALC, F
        BTFSC CCP1CON, CCP1X
            BSF CALC, 1
        BTFSC CCP1CON, CCP1Y
            BSF CALC, 0
        CLRF CALC2              ;zera CALC2
        MOVF MEDIA_H, W
        SUBWF POT_H, W
        MOVWF DIF_H
        BTFSS STATUS, C         ;se diferenca de potencia negativa,
            BSF CALC2, 0        ;seta CALC2<0>
        MOVF MEDIA_L, W
        SUBWF POT_L, W
        MOVWF DIF_L
        MOVLW 1
        BTFSS STATUS, C         ;se DIF_L negativo, subtrai 1 de DIF_H
            SUBWF DIF_H, F
        BTFSS STATUS, C         ;se DIF_H < 0, seta CALC2<0>
          BSF CALC2, 0
        CALL OBTEM_DC           ;chama OBTEM_DC
        MOVF GUARDA_FSR, W      ;restitui ponteiro da tabela
        MOVWF FSR
        MOVLW B'11001111'       ;bloco que restaura PWM 10 bits a
partir
        ANDWF CCP1CON, F        ;de CALC
        RRF CALC, F
        BTFSC STATUS, C
            BSF CCP1CON, CCP1Y
        RRF CALC, W
        BTFSC STATUS, C
            BSF CCP1CON, CCP1X
        ANDLW B'00011111'
        MOVWF CCPR1L

PARADA                          ;aguarda a passagem de 65,5ms em
        MOVF CONTA_OVERS, F     ;loop "infinito"
        BTFSC STATUS, Z
```

```
            GOTO EXECUCAO
      GOTO PARADA

;*****************************************************************
;                              FUNCOES
;*****************************************************************

;---------------------------------------------------------------
;Funcao de conversao A/D
;---------------------------------------------------------------
CONVERSAO
       IORLW B'10000001'     ;obtem qual pino e inicia sample
       MOVWF ADCON0
       MOVLW 4               ;sample-time
       MOVWF CALC3
ESPERA
       DECFSZ CALC3, F
           GOTO ESPERA
       BSF ADCON0, GO_DONE   ;inicia conversao
CONVERTENDO                  ;verifica se terminou conversao
      BTFSC ADCON0, GO_DONE
            GOTO CONVERTENDO
        BCF ADCON0, ADON      ;desliga conversao A/D
        MOVF ADRES, W         ;guarda resultado em W
        RETURN

;---------------------------------------------------------------
;Funcao que multiplica tensao por corrente
;---------------------------------------------------------------
CALC_POTENCIA
        CLRF POT_H
        CLRF POT_L
        MOVLW 8
        MOVWF CALC3
        MOVF TENSAO_BAT, W
        BCF STATUS, C
LOOP_MULT
        RRF CORR_BAT, F
        BTFSC STATUS, C
          BSF CORR_BAT, 7
        BTFSC STATUS, C
            ADDWF POT_H, F
        RRF POT_H, F
        RRF POT_L, F
        DECFSZ CALC3, F
            GOTO LOOP_MULT
        RETURN


;---------------------------------------------------------------
;Funcao que obtem novo duty-cycle
;---------------------------------------------------------------
OBTEM_DC
        BTFSS CALC2, 0          ;verifica se varicacao de potencia < 0
            GOTO SINAL_CERTO    ;se nao, vai para SINAL_CERTO
        COMF DELTA, F          ;se <0 entao DELTA = -DELTA
        INCF DELTA, F
SINAL_CERTO
      MOVF DELTA, W
        ADDWF CALC, F           ;CALC = CALC + DELTA
        MOVLW B'01110100'       ;Verifica se maior que 90%
```

```
        SUBWF CALC, W
        BTFSS STATUS, C
            GOTO ABAIXO_MAXIMO  ;se nao, vai para ABAIXO_MAXIMO
        MOVLW B'01110011'   ;caso contrario, PWM = 90%
        GOTO FORA_LIMITE
ABAIXO_MAXIMO
        MOVLW B'00001101'       ;Verifica se menor que 10%
        SUBWF CALC, W
        BTFSC STATUS, C
            RETURN              ;se nao, retorna com novo PWM em CALC
        MOVLW B'00001101'       ;se sim, PWM = 10%
FORA_LIMITE
        MOVWF CALC              ;guarda novo PWM em CALC
        COMF DELTA, F           ;DELTA = -DELTA
        INCF DELTA, F
        RETURN

;-----------------------------------------------------------------------
;Funcao que verifica tensao minima da bateria para ligar lampada
;-----------------------------------------------------------------------
LIGAR_LAMP
        MOVLW B'00001000'               ;pino AN1 - tensao
        CALL CONVERSAO
        MOVWF GUARDA_TENSAO
        MOVLW V_MIN
        MOVWF TENSAO_MIN
        SE_AMENORB GUARDA_TENSAO, TENSAO_MIN, FIM_LIGAR_LAMP
        BSF PORTC, 3        ;se tudo OK, liga
FIM_LIGAR_LAMP
        RETURN


END

;*****************************************************************
;                       FIM DO PROGRAMA PSOLAR.ASM
;*****************************************************************
```

Anexo E. Codificação do programa de comunicação serial do lado do microcomputador

O programa do lado do microcomputador, implementado em C, tem a função de permitir ao usuário programar os horários de acendimento e desligamento da lâmpada. O usuário entra os horários no formato **hh mm ss**. O programa converte este formato em dois bytes, que é o formato de horário com que o programa do lado do microcontrolador trabalha. O programa permite que o usuário escolha entre enviar o horário atual do relógio do computador ou um horário atual qualquer. Uma outra opção do programa é apenas receber os dados do microcontrolador, apresentando a diferença entre o horário atual do computador e o horário atual do computador. O fluxograma do programa principal pode ser visto abaixo.

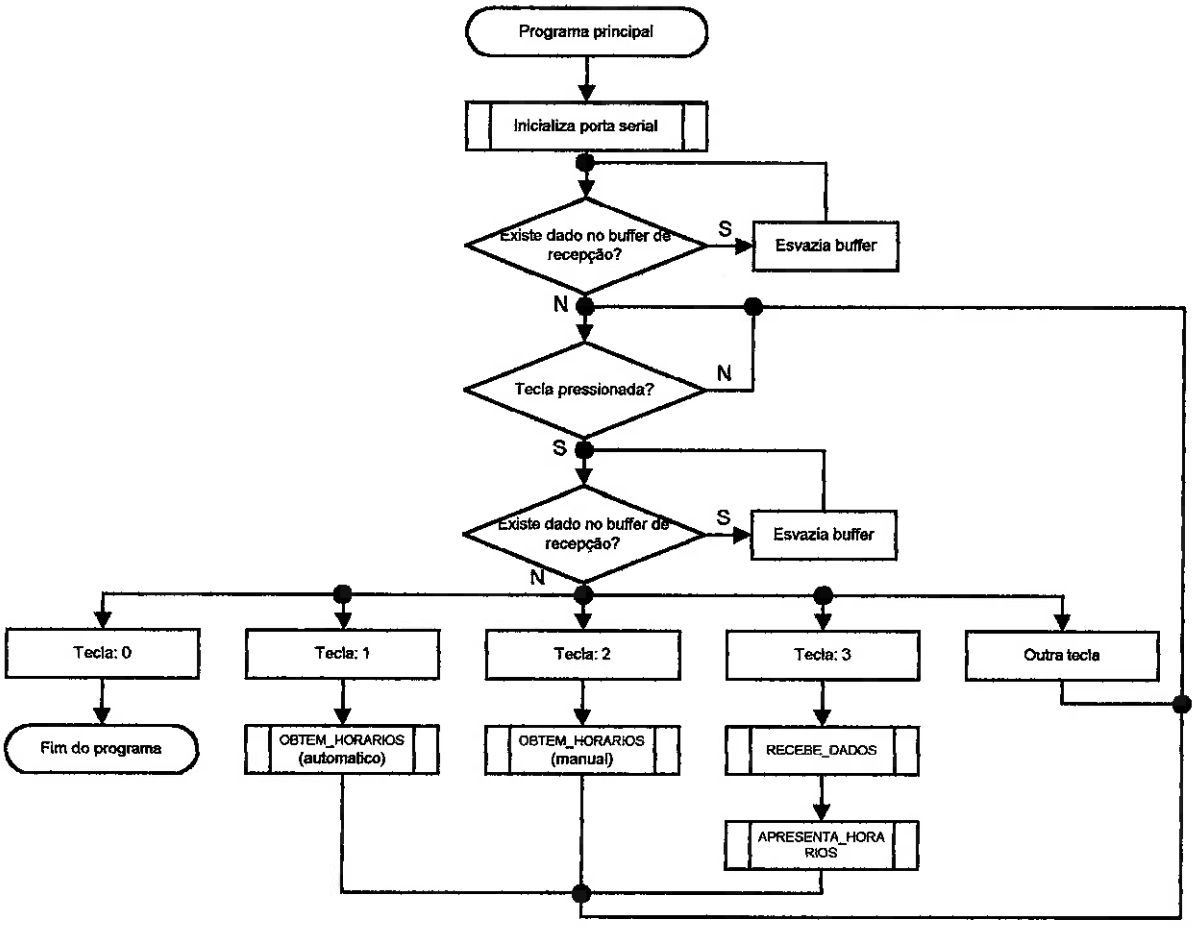


Figura E.1. Programa principal do lado do computador.

A função OBTEM_HORARIOS tem a função de obter os horários no formato usual (hh mm ss), convertê-los em dois bytes (formato do relógio implementado no microcontrolador), enviá-los ao microcontrolador, comparar os dados que o PIC recebeu com os que o programa enviou e validar ou não a transmissão efetuada. Abaixo tem-se o fluxograma desta função.

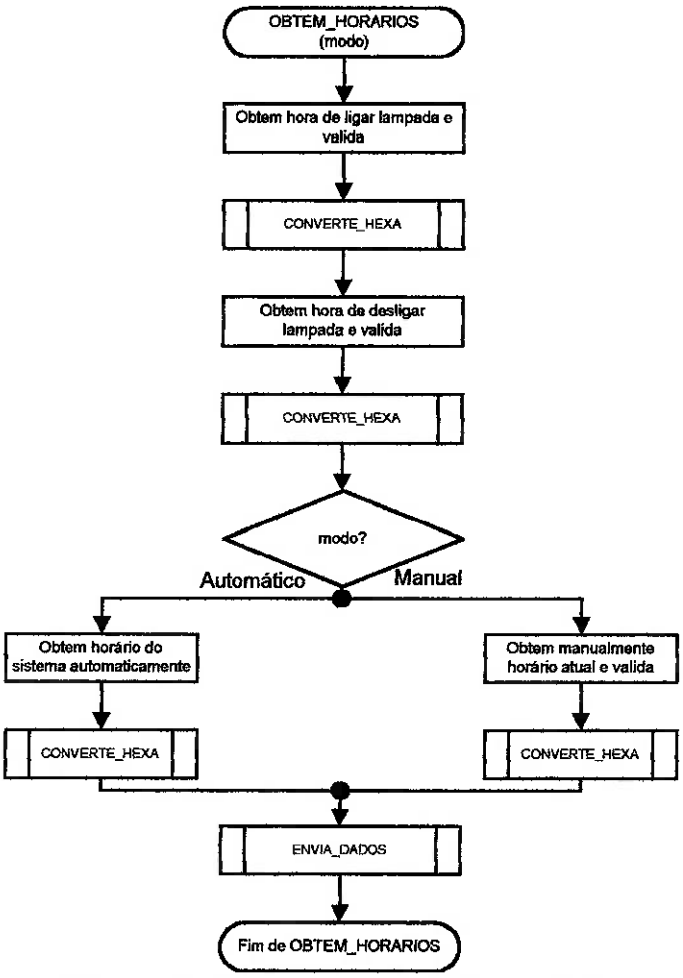


Figura E.2. Função OBTEM_HORARIOS.

A função CONVERTE_HEXA apenas converte os horários do formato normal para o de dois bytes e não será apresentada aqui. Entretanto, cabe salientar que para o envio dos dados, eles são armazenados num vetor chamado **dado** de 6 posições. As posições 0 e 1 representam os dois bytes do horário atual. As posições 2 e 3 o horário de acendimento e as 4 e 5 o horário de desligamento da lâmpada.

Abaixo expande-se a função ENVIA_DADOS.

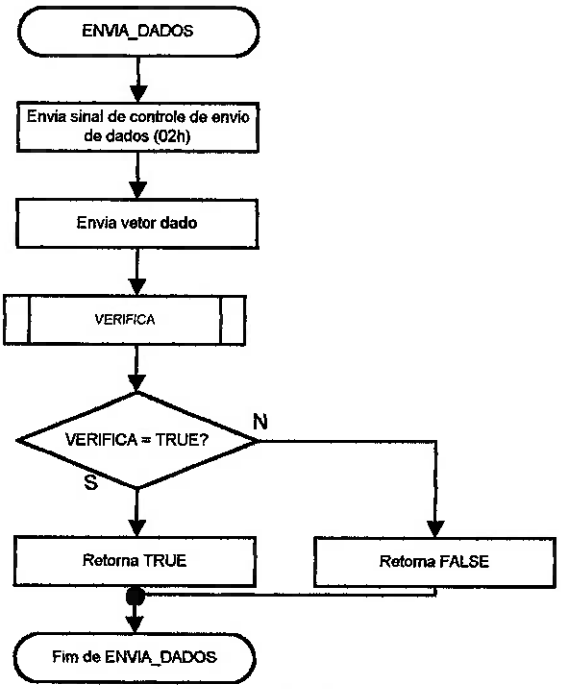


Figura E.3. Função ENVIA_DADOS.

A função que envia, efetivamente, byte a byte se chama-se put_serial. A mesma não será apresentada aqui pois apenas seta os registros de controle da comunicação serial. Uma explicação detalhada destes registros será apresentada no final deste anexo.

A função VERIFICA também não tem necessidade de ser apresentada, pois ela apenas recebe os dados do PIC num vetor de 6 posições chamada **Verdado** e o compara posição a posição com o vetor **dado**.

Voltando ao programa principal. A rotina RECEBE_DADOS recebe os dados do microcontrolador e os armazena para posterior apresentação na tela. Antes de se preparar para receber os dados esta função envia o byte de controle 04h para o microcontrolador, significando que o computador solicita os dados do PIC.

A rotina APRESENTA_HORARIOS chama a função CONVERTE_HORA que transforma os horários de dois bytes para o formato usual (hh mm ss) e os apresenta na tela.

E.1. Registradores da comunicação

A comunicação serial do lado do computador é feita sobre a interrupção 14h da BIOS. A linguagem C apresenta a função **int86**, que tem como parâmetros o número da interrupção e a passagem dos registradores daquela interrupção. A função retorna os valores nos mesmos registradores em que foram passado os parâmetros. Ou seja, sua chamada é da forma:

```
#include <dos.h>
union REGS regs;
int recebido;
regs.h.ah = 0x02;
regs.x.dx = 0;
int86(0x14, &regs, &regs);
recebido = regs.h.al;
```

O exemplo acima ilustra a recepção (AH = 0x02) na COM1 (DX = 0). O byte recebido se encontra no registrador AL.

As folhas em anexo apresentam os registradores utilizados pela interrupção 14h, que é a interrupção da BIOS de comunicação serial.

Após isto tem-se a listagem completa do programa.

The first of the ROM BIOS serial communications routines is an *Initialize Serial Port routine*. Figure 3.1 illustrates how this routine is called and any values that are returned by this routine.

INT 14H, Function 00H – Initialize Serial Port

Description:

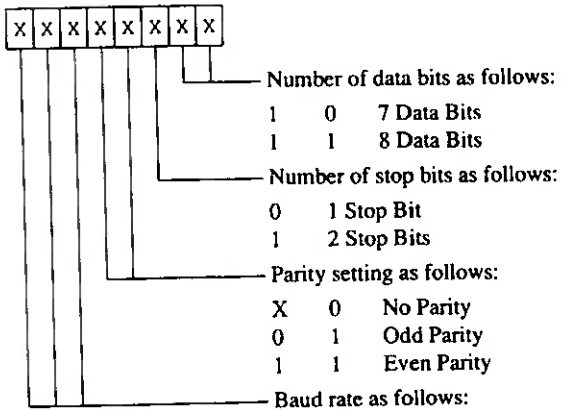
This ROM BIOS routine initializes a serial port's baud rate, number of data bits, parity setting, number of stop bits, and returns the serial port's current status.

Call With:

AH 00H (The function number.)

AL The serial port's initialization values as follows:

```
Bits 7 6 5 4 3 2 1 0
```



Number of data bits as follows:

1	0	7 Data Bits
1	1	8 Data Bits

Number of stop bits as follows:

0	1 Stop Bit
1	2 Stop Bits

Parity setting as follows:

X	0	No Parity
0	1	Odd Parity
1	1	Even Parity

Baud rate as follows:

0	0	0	110 Baud
0	0	1	150 Baud
0	1	0	300 Baud
0	1	1	600 Baud
1	0	0	1200 Baud
1	0	1	2400 Baud
1	1	0	4800 Baud
1	1	1	9600 Baud

DX 00H for COM1, 01H for COM2

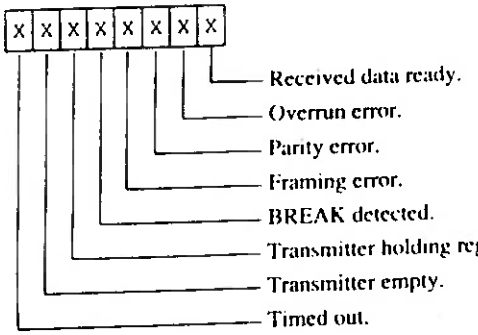
continued...

...from previous page

Returns:

AH The port status as follows:

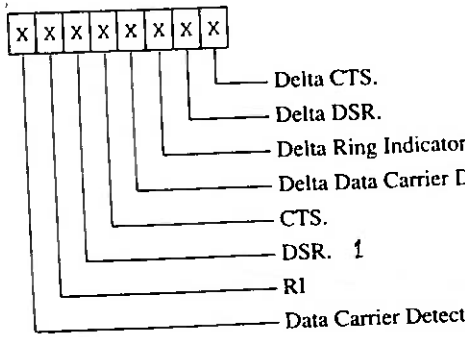
```
Bit 7 6 5 4 3 2 1 0
```



Received data ready.
Overrun error.
Parity error.
Framing error.
BREAK detected.
Transmitter holding register empty.
Transmitter empty.
Timed out.

AL The modem status as follows:

```
Bit 7 6 5 4 3 2 1 0
```



Delta CTS.
Delta DSR.
Delta Ring Indicator.
Delta Data Carrier Detect.
CTS.
DSR.
RI
Data Carrier Detect.

***Figure 3.1** ROM BIOS INT 14H, Function 00H.*

The second of the ROM BIOS serial communications routines is the *Write Character to Serial Port routine*. Figure 3.2 illustrates how this routine is called and any values that are returned by this routine.

INT 14H, Function 01H - Write Character to Serial Port

Description:

This ROM BIOS routine sends a character out a specified serial port and returns the port's current status.

Call With:

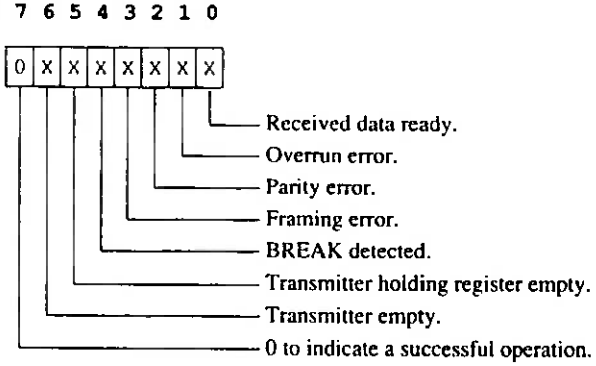
AH 01H (The function number.)

AL The character to be sent out the serial port.

DX 00H for COM1, 01H for COM2

If Successful, Returns:

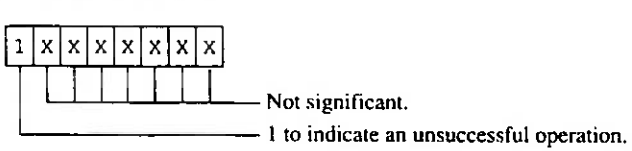
AH Indicates a successful operation and returns the serial port's current status as follows:



AL The character that was sent to the serial port.

If Not Successful, Returns:

AH Indicates an unsuccessful operation as follows:



AL The character that was sent to the serial port.

Figure 3.2 ROM BIOS INT 14, Function 01H.

The third of the ROM BIOS serial communications routines is the *Read Character from Serial Port routine*. Figure 3.3 illustrates how this routine is called and any values that are returned by the routine.

INT 14H, FUNCTION 02H - Read Character from Serial Port

Description:

This ROM BIOS routine reads a character from a specified serial port. In addition to returning the character, this routine indicates any errors that may have occurred.

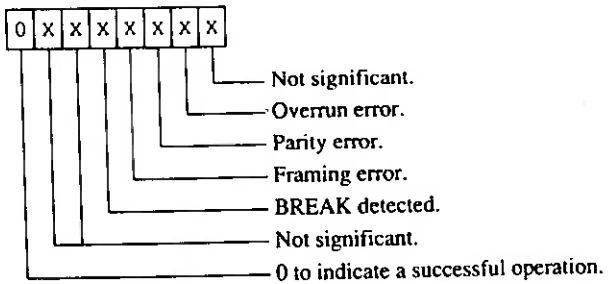
Call With:

AH 02H (The function number.)

DX 00H for COM1, 01H for COM2

If Successful, Returns:

AH Indicates a successful operation and returns the serial port's status as follows:



AL The character that was fetched from the serial port.

If Not Successful, Returns:

AH Indicates an unsuccessful operation as follows:

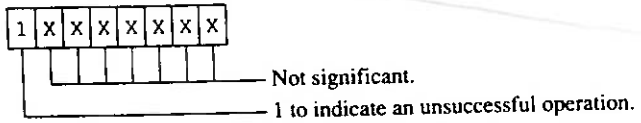


Figure 3.3 ROM BIOS INT 14, Function 02H.

The fourth and final ROM BIOS serial communications routine is the *Get Serial Port Status routine*. Figure 3.4 illustrates how this routine is called and the values that are returned by this routine.

INT 14, Function 03H - Get Serial Port Status

Description:

This ROM BIOS routine returns the serial port's current status.

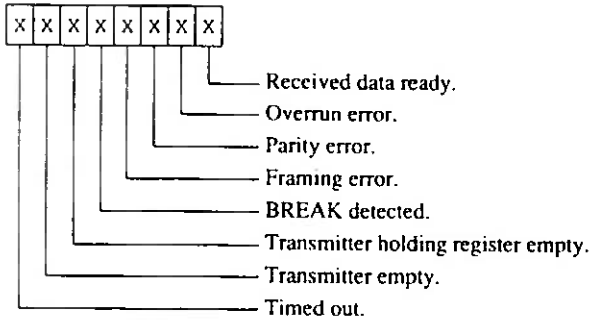
Call With:

AH 03H (The function number.)

DX 00H for COM1, 01H for COM2

If Successful, Returns:

AH The port status as follows:



AL The modem status as follows:

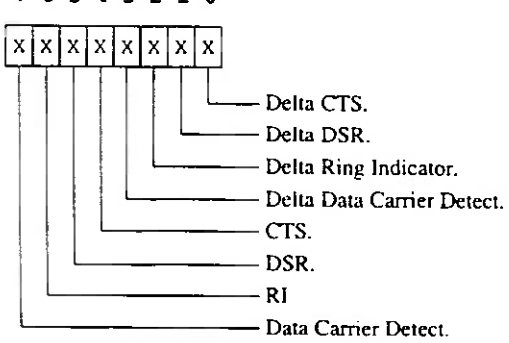


Figure 3.4 ROM BIOS INT 14, Function 03H.

```
/*
Universidade de Sao Paulo - Escola Politecnica - Depto. Eng. Mecanica
Trabalho de Formatura - Projeto Poste Solar
Eric Brandt Schonwald - NUSP: 270260
Orientador: Marcelo Godoy Simoes
Dezembro de 1998
Arquivo: PSOLAR.C
Compilado no TURBO C++ Versao 3.0 - Borland

A funcao deste programa e realizar a comunicacao serial com
microcontrolador PIC16C74, enviando e recebendo horarios de
acendimento,
desligamento e horario atual do computador.
*/


#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#include <math.h>


#define TRUE 1
#define FALSE 0

#define ATRASO 1  /*delay de 1ms entre um byte e outro*/

int dado[8];      /*guarda os valores dos horarios em formato de 2
bytes*/
int atual[2],liga[2],desliga[2];    /*horarios no formato hh mm ss*/
int PORT, portnumber;   /*numero da porta serial*/
int Verdado[8];   /*vetor de recepcao dos dados para verificacao*/

/* declaracao das funcoes*/
int open_port(int n);
void put_serial(int n, int c);
int get_serial(int n);
int in_ready(int n);
int envia_dados();
int recebe_dados(int recebido[8]);
void obtem_horarios(char modo);
void apresenta_horarios();
void corrige_hora();
void converte_hora(int MsbPIC, int LsbPIC, int *hora, int *minuto, int
*segundo);
void converte_hexa(int hora, int minuto, int segundo, int *MSByte, int
*LSByte);
int verifica();

/*PROGRAMA PRINCIPAL*/
void main(void)
{
      char opcao;
      /*menu de selecao da porta serial*/
      clrscr();
      printf("Programa de comunicacao com PIC - Escolha da porta
serial");
      printf("\n1  -  COM1");
      printf("\n2  -  COM2");
      printf("\n0  -  Sair");
```

```
      printf("\n\nOpcao:");

      PORT = 0;
      while (PORT == 0)
      {

            opcao = getch();
            switch (opcao) {
                  case '0':
                        exit(0);
                        break;
                  case '1':
                        PORT = 1;
                        portnumber = 0;
                        break;
                  case '2':
                        PORT = 2;
                        portnumber = 1;
                        break;
                  default:
                        printf("\nOpcao invalida");
            }
      }


      /* abre porta serial, verificando se cabo esta conectado*/
      while (open_port(portnumber)==FALSE)
  {
            clrscr();
            printf("\nCabo desconectado!!!");
            printf("\nReconecte-o e pressione qualquer tecla (0 para
sair).");
            opcao = getch();
            if (opcao == '0')
                  exit(0);
      }
      /*limpa buffer*/
      while (in_ready(portnumber))
      {
            putch(get_serial(portnumber));
      }

      /*menu principal do programa*/
      clrscr();
      printf("Programa de comunicacao com PIC");
      printf("\n1  -  Envia dados - Horario automatico");
      printf("\n2  -  Envia dados - Horario manual");
      printf("\n3  -  Recebe dados");
      printf("\n0  -  Sair");
      printf("\n\nOpcao:");


      while (TRUE)
      {
            if (kbhit())
            {
                  opcao = getch();
                  while (in_ready(portnumber))
                  {
                        putch(get_serial(portnumber));
                  }
```

```
                switch (opcao) {
                      case '0':                                    /*
exit on Alt-X */
                            exit(0);
                            break;
                      case '1':
                            printf("\n\nEnvio de dados para
microcontrolador. Horario automatico");
                            obtem_horarios('a');
                            break;
                      case '2':
                            printf("\n\nEnvio de dados para
microcontrolador. Horario manual");
                            obtem_horarios('m');
                            break;
                      case '3':
                            printf("\n\nRecepcao de dados do
microcontrolador");
                            if (recebe_dados(dado)==TRUE)
                                  apresenta_horarios();
                            break;
                      default:
                            printf("\nOpcao invalida");
                }
          }
     }
}

/*funcao que verifica se os dados enviados e recebidos sao iguais*/
int verifica()
{
     int i;

     printf("\nAguarde, verificando...");
     recebe_dados(Verdado);
     /*tolerancia de 8 segundos*/
     if ((Verdado[0]!=dado[0]) || (Verdado[1]<dado[1]) ||
(Verdado[1]>dado[1]+8))
          return FALSE;

     for (i=2;i<=5;i++)
     {
          if (Verdado[i]!=dado[i])
               return FALSE;
     }

     return TRUE;
}

/*funcao que exibe os horarios recebidos na tela*/
void apresenta_horarios()
{
     struct time HoraRecebida, HoraSistema;
     long diferenca,Recebida,Sistema;

     converte_hora(dado[2],dado[3], &liga[0],&liga[1],&liga[2]);
     printf("\nLiga: %02i:%02i:%02i",liga[0],liga[1],liga[2]);
     converte_hora(dado[4],dado[5],
&desliga[0],&desliga[1],&desliga[2]);
```

```
      printf("\nDesliga:%02i:%02i:%02i",desliga[0],desliga[1],desliga[
2]);
      converte_hora(dado[0],dado[1], &atual[0],&atual[1],&atual[2]);
      printf("\nHorario PIC:
%02i:%02i:%02i",atual[0],atual[1],atual[2]);
      /*horario do computador*/
      gettime(&HoraSistema);
      printf("\nHorario computador:
%02i:%02i:%02i",HoraSistema.ti_hour,HoraSistema.ti_min,HoraSistema.ti_
sec);

      Recebida=atual[0]*3600+atual[1]*60+atual[2];
      Sistema=HoraSistema.ti_hour*3600+HoraSistema.ti_min*60+HoraSiste
ma.ti_sec;
      diferenca=Sistema-Recebida;
      printf("\nDiferenca=%li segundos",diferenca);

}

/*converte horario de 2 bytes em formato hh mm ss*/
void converte_hora(int MsbPIC, int LsbPIC, int *hora, int *minuto, int
*segundo)
{

      unsigned long aux1, aux2;
      double quociente, integer, totalh, totalm, totals, dblHora,
dblMinuto;

      aux1=MsbPIC;
      aux2=LsbPIC;

      totalh=2*(aux1*256+aux2);
      dblHora=floor(totalh/3600);
      *hora=dblHora;

      totalm=totalh-(3600*dblHora);
      dblMinuto=floor(totalm/60);
      *minuto=dblMinuto;

      totals=totalm-60*dblMinuto;
      *segundo=totals;
}

/*converte horario hh mm ss em formato 2 bytes*/
void converte_hexa(int hora, int minuto, int segundo, int *MSByte, int
*LSByte)
{
      unsigned long ulHora, ulMinuto, ulSegundo;
      double msb, lsb, HorarioPIC;

      ulHora=hora;
      ulMinuto=minuto;
      ulSegundo=segundo;
      HorarioPIC=(ulHora*3600+ulMinuto*60+ulSegundo)/2;
      msb=floor(HorarioPIC/256);
      lsb=HorarioPIC-(msb*256);
      *MSByte=msb;
      *LSByte=lsb;

}
```

```
/*obtem horarios do usuario, validando-os*/
void obtem_horarios(char modo)
{
      struct time HrSist;
      int i;
      int OK;

      printf("\nInsira horario para ligar lampada:");
      OK =  FALSE;
      while (OK == FALSE)
      {
            scanf("%i %i %i", &liga[0], &liga[1], &liga[2]);
            if ((liga[0]>23) || (liga[1] > 59) || (liga[2]>59))
                  printf("\nHorario invalido. Favor reinseri-lo: ");
            else
                  OK = TRUE;
      }
      converte_hexa(liga[0], liga[1], liga[2], &dado[2], &dado[3]);
      printf("Insira horario para desligar lampada:");
      OK =  FALSE;
      while (OK == FALSE)
      {
            scanf("%i %i %i", &desliga[0], &desliga[1], &desliga[2]);
            if ((desliga[0]>23) || (desliga[1] > 59) ||
(desliga[2]>59))
                  printf("\nHorario invalido. Favor reinseri-lo: ");
            else
                  OK = TRUE;
      }
      converte_hexa(desliga[0], desliga[1], desliga[2], &dado[4],
&dado[5]);
      if (modo=='m')
      {
            printf("Insira horario atual:");
            OK =  FALSE;
            while (OK == FALSE)
            {
                  scanf("%i %i %i", &atual[0], &atual[1], &atual[2]);
                  if ((atual[0]>23) || (atual[1] > 59) ||
(atual[2]>59))
                        printf("\nHorario invalido. Favor reinseri-lo:
");
                  else
                        OK = TRUE;
            }
            converte_hexa(atual[0], atual[1], atual[2], &dado[0],
&dado[1]);
      }
      else
      {
            gettime(&HrSist);
            converte_hexa(HrSist.ti_hour, HrSist.ti_min, HrSist.ti_sec,
&dado[0], &dado[1]);
      }

      i=0;
      /*5 tentativas de envio*/
      while ((envia_dados()!=TRUE) && (i<5))
      {
            i=i+1;
            printf("\nErro na %i a. tentativa",i);
```

```
	}
	if (i<5)
		printf("\nEnvio realizado com sucesso");
	else
		printf("\nErro grave de comunicacao. Verifique se cabo esta
conectado e circuito ligado.");

	if (modo=='a')
	{
		atual[0]=HrSist.ti_hour;
		atual[1]=HrSist.ti_min;
		atual[2]=HrSist.ti_sec;
	}
}

/*envia dados para o PIC16C74*/
int envia_dados()
{
	int aux, i, x, j;

	printf("\nEnviando...");
	aux = 0x02; /*Palavra de controle que significa envio para o
PIC*/
	put_serial(portnumber, aux); /*envia aux para a serial*/
	delay(ATRASO);    /*espera ATRASO milissegundos*/

	for (i=0; i<=5; i++)
	{
		put_serial(portnumber, dado[i]);
		delay(ATRASO);
	}

	if (verifica()==TRUE)   /*se enviou e recebeui corretamente*/
		return TRUE;
	else
		return FALSE;
}

/*recebe horarios do PIC16C74*/
int recebe_dados(int recebido[8])
{
	int aux;
	int i, limite;

	i=0;
	limite=0;
	aux = 0x04; /*palavra de controle: Recepcao de dados*/
	put_serial(portnumber, aux);  /* envia aux para a serial */

	/*recebe 6 bytes e se previne contra loop infinito (limite)*/
	while ((i<=5) && (limite<2000))
	{
		limite=limite+1;
		if (in_ready(portnumber))
		{
			recebido[i] = get_serial(portnumber);
			i=i+1;
		}
	}

	if (limite>=2000)
```

```
        {
                printf("\nErro na comunicacao!");

                while (in_ready(portnumber))
                {
                        putch(get_serial(portnumber));
                }
                return FALSE;
        }
        else
        {
                return TRUE;
        }
}


/* Abre porta serial utilizando as interrupcoes/rotinas da ROM BIOS */
int open_port(int n)
{
        union REGS regs;
        printf("\nAguarde...");

        regs.h.ah = 0x00; /*rotina de inicializacao da  porta serial*/
        /*1200 bauds, 8bits, sem paridade, 1 stop bit*/
        regs.h.al = 0x83;
        regs.x.dx = n;          /*numero da porta*/
        int86(0x14, &regs, &regs);    /*executa*/

        /*verifica se cabo esta conectado*/
        regs.h.ah = 0x02; /*rotina de recepcao*/
        regs.x.dx = n;          /*numero da porta*/
        int86(0x14, &regs, &regs);    /*executa*/
        regs.h.ah = 0x03; /*rotina de status*/
        regs.x.dx = n;          /*numero da porta*/
        int86(0x14, &regs, &regs);    /*executa*/
        if (regs.h.al & 0x30)   /*verifica flags CTS e DSR*/
                return TRUE;
        else
                return FALSE;
}

/* envia caractere pela porta serial */
void put_serial(int n, int c)
{
        union REGS regs;

        regs.h.ah = 0x01; /*rotina de envio*/
        regs.h.al = c;          /*carrega caractere*/
        regs.x.dx = n;          /*numero da porta*/
        int86(0x14, &regs, &regs);    /*executa*/
}

/* obtem caractere da porta serial */
int get_serial(int n)
{
        union REGS regs;

        regs.h.ah = 0x02; /*rotina de recepcao*/
        regs.x.dx = n;          /*numero da porta*/
        int86(0x14, &regs, &regs);    /*executa*/
```

```
      if (regs.h.ah & 0x80)   /*verifica se estourou o tempo*/
            return EOF;
      return regs.h.al;  /*retorna o caractere recebido*/
}

/* verifica se ha caractere no buffer */
int in_ready(int n)
{
      union REGS regs;

      regs.h.ah = 0x03;       /*rotina de status*/
      regs.x.dx = n;          /*numero da porta*/
      int86(0x14, &regs, &regs);    /*executa*/

      if (regs.h.ah & 1)      /*verifica se ha dado*/
            return TRUE;
      return FALSE;
}

/*Fim do programa PSOLAR.C*/
```

Anexo F. Princípio de funcionamento do rastreador de ponto de potência máxima em painéis solares

As características de rendimento de um painel solar são fortemente influenciadas pela densidade de potência luminosa, pela temperatura e pela carga acoplada ao painel. A figura abaixo mostra a curva característica dos painéis solares para uma temperatura constante e várias incidências luminosas (λ [W/m^2]).

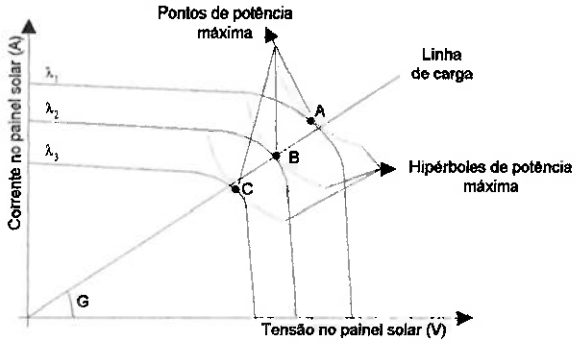


Figura F.1. Curva característica dos painéis solares

Na figura acima, percebe-se que a potência absorvida pelo painel é dada pela intersecção entre a linha de carga (com condutância G) e a curva característica V x I. Além disto, as curvas de potência são hipérboles que tocam as curvas características V x I em apenas um ponto para cada incidência luminosa. A fim de se atingir a máxima transferência, deve-se portanto, variar a condutância G da carga. Esta é a função do "Rastreador do Ponto de Potência Máxima".

Este rastreador é um conversor Boost, elevador de tensão, que apresenta um indutor que se carrega de energia, e um MOSFET, cujo chaveamento controla a carga do indutor ou a sua descarga para a bateria, conforme é apresentado na figura abaixo.

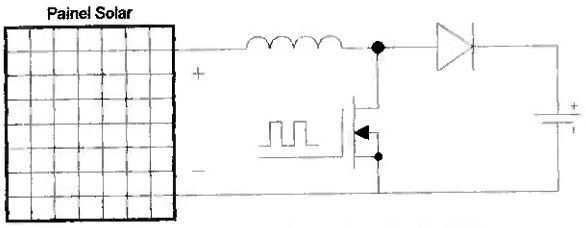


Figura F.2. Circuito do RPPM.

O controle da transferência é feito pelo tempo em que o indutor está se carregando ou descarregando. O microcontrolador PIC, através de seus conversores A/D monitora a tensão e corrente de carga da bateria. O produto das duas medições fornece a potência. Após isto, compara-se se a potência atual é maior ou menor que a média das 32 últimas leituras. Através disto, o PIC varia o sinal de controle do MOSFET, através de seu canal de PWM.

Define-se razão cíclica ou *duty-cycle* à razão $\delta = t_{on}/t_{PWM}$ do PWM. Ou seja, controlar a transferência de potência, nada mais é do que variar a razão cíclica do PWM. O período do PWM é constante e vale 32 μs, ou seja, $t_{PWM} = t_{on} + t_{off} = 32$ μs.

A procura da melhor razão cíclica é "empírica" no sentido de que o microcontrolador começa a variar o PWM diminuindo o t_{on}, por exemplo. Se a potência começar a decrescer ainda mais, então o microcontrolador começa a aumentar o t_{on}, até que a potência começa a diminuir novamente. Então ele começa a diminuir o t_{on} novamente. Desta forma, o microcontrolador está sempre procurando o melhor *duty-cycle*.

Anexo G. "Conservar Energia É Fácil"

Manter as luzes acesas não economiza energia

O mito de que apagar e acender as lâmpadas em espaços curtos de tempo gasta mais energia do que deixá-las acesas é FALSO!! Sempre que não houver ocupação dos ambientes, apague a luz! Em locais onde só é possível acender ou apagar todas as luzes ao mesmo tempo, verifique com o eletricista de sua unidade a possibilidade de instalação de interruptores individualizados para seu ambiente de trabalho.

Aproveite a luz natural

Sempre que possível, desligue a iluminação quando a luz natural estiver disponível. Mantenha as janelas sempre limpas e evite a incidência de luz solar direta através do uso de cortinas ou venezianas, para reduzir o ofuscamento.

Reitor
Jacques Marcovitch

Vice-Reitora
Myriam Krasilchik

Comissão do Programa Permanente para o Uso Eficiente da Energia Elétrica na USP

Presidente:
Hélio Nogueira da Cruz

Comissão:
Antonio Rodrigues Martins
Carlos Américo Morato de Andrade
Célio Taniguchi
Marcelo de Andrade Roméro
Marco Antonio Saidel
Miguel Bussolini

Engenharia:
Paulo Kanayama

Divisão de Artes Gráficas
Diretor:
Valtemir Tacinari Sasso

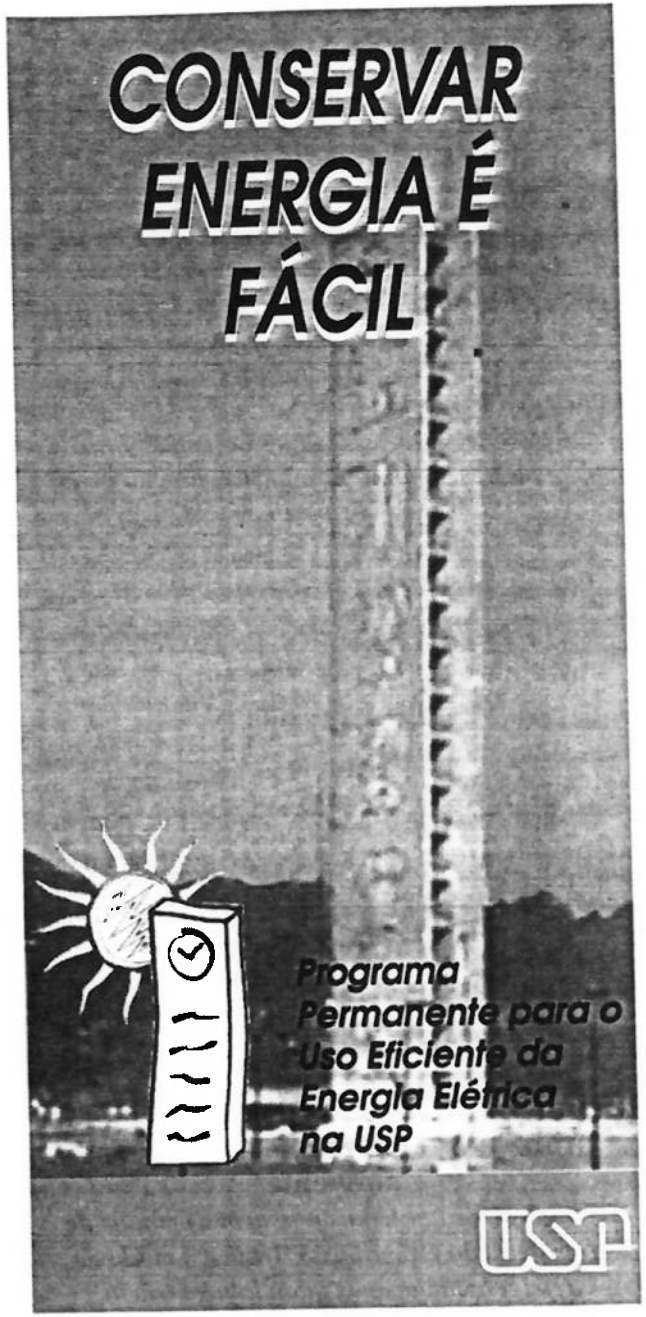
Edição de Arte:
Luis Carlos de Araújo Bezerra

Fotolitos, Impressão e Acabamento:
USP CCS DIVISÃO GRÁFICA

INFORMAÇÕES
Programa Permanente para o Uso Eficiente da Energia Elétrica na USP
conserva pea.usp.br

GEPEA—EPUSP
Tel. (011) 818-5349

CODAGE
Tel. (011) 818-3389



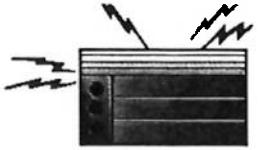
Condicionadores de ar
Aparelhos de janela

Mantenha portas e janelas fechadas

Impeça a entrada de ar externo com temperatura mais elevada. Esta medida faz com que o aparelho gaste menos energia. Em outras palavras, a presença de frestas e janelas ou portas abertas faz com que o aparelho procure refrigerar também o ambiente externo.

A orientação aos usuários para mantê-las fechadas através de cartazes, ou a instalação de braços mecânicos com molas para fechamento automático das portas ajudam a economizar energia.

Evite o frio excessivo



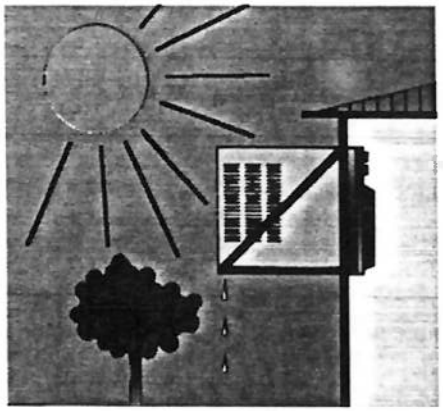
Temperaturas muito baixas, além de aumentarem o consumo de energia, podem causar mal à saúde. Procure trabalhar com temperaturas que lhe proporcionem conforto.

Em dias de temperatura não muito alta, utilize o ar externo para refrescar o ambiente

Em geral, os aparelhos possuem os modos de refrigeração e ventilação. Utilize o modo de refrigeração apenas quando necessário.

Proteja do sol as partes externas

do aparelho e não bloqueie as grades de ventilação.



Iluminação

Lâmpadas piscando? Troque o reator.

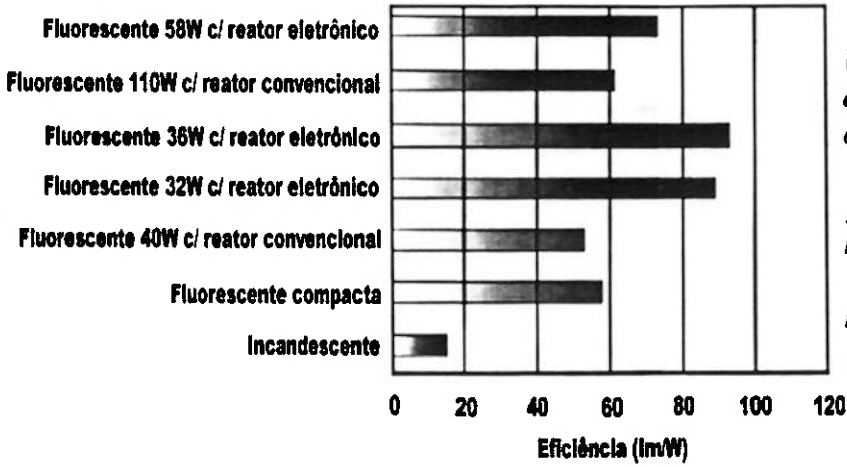
Lâmpadas fluorescentes com cintilação causam desconforto, irritabilidade e comprometem a produtividade. Neste caso, procure substituir os reatores, dê preferência aos ***eletrônicos*** *de marcas confiáveis, que além de não causarem o fenômeno da cintilação, produzem menos ruido e economizam energia.*

Mas cuidado com os reatores muito baratos que não contenham especificações técnicas de distorção harmônica ou fator de potência. Eles podem prejudicar a rede elétrica.

Evite as lâmpadas incandescentes

Sempre que for substituir lâmpadas incandescentes queimadas, dê preferência às lâmpadas fluorescentes compactas, que funcionam com o mesmo soquete e são muito mais econômicas.

Embora sejam mais caras, as lâmpadas fluorescentes compactas duram até 10 vezes mais, diminuindo o trabalho de substituição periódica,e consomem menos energia.



Veja o ***gráfico de eficiência luminosa*** *medida em lumens por watt. As lâmpadas fluorescentes iluminam muito mais que as incandescentes por watt utilizado.*