Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Engenharia Elétrica

Projeto de Engenharia Elétrica

Projeto de uma placa de circuito impresso de um circuito de comando para um inversor monofásico em ponte completa à IGBT

João Helder Gonzaga Muniz da Silva

Orientador: Talvanes Meneses Oliveira

Agosto de 2009

Dedicatória

Aos meus pais João e Auristela pelas preocupações que passaram por minha causa.

À minha noiva Renata com quem compartilho sonhos e objetivos.

As minhas tias Alvânia e Alvanira pelas incontáveis ajudas que me deram e dão.

As minhas irmãs Fabiana e Fernanda.

Ao amigo Weidson do Amaral pelo apoio e estímulo.

Aos meus Avós Antônio e Maria que não está mais conosco.

Aos meus familiares por fazerem parte da minha vida.

Agradecimento

À minha noiva Renata por todo amor e carinho doado.

Aos amigos que contribuíram de forma direta e indireta para realização deste trabalho.

Ao meu orientador Talvanes, por ter confiado no meu trabalho e capacidade.

SUMÁRIO

1. OBJETIVOS

1.1. Objetivo específico

Confecção de uma placa de circuito impresso (PCI) de um circuito de comando para um inversor monofásico.

1.2. Objetivos gerais

- Estudar a teoria básica dos conversores *CC/CA* monofásicos;
- Adquirir conhecimentos sobre técnicas de chaveamento de inversores, ou seja, familiarização com técnicas de modulação em inversores.

2. INTRODUÇÃO TEÓRICA

Os conversores estáticos realizam o controle de fluxo de potência elétrica entre fonte e carga, sejam eles do tipo alternada *CA* ou do tipo contínuo *CC*. Os conversores *CC/CA* também conhecidos como inversores, convertem uma tensão de entrada *CC* em uma tensão de saída *CA* simétrica de amplitude e freqüência desejadas. A utilização do conversor permite-se obter uma tensão variável na saída, variando-se amplitude da tensão *CC* na entrada, no entanto. Se a tensão *CC* na entrada é fixa e não controlável, uma tensão variável de saída pode ser obtida pela variação do ganho do inversor, onde geralmente é realizada pelo controle de modulação, que será discutido nas secções posteriores.

As formas de ondas de saída dos inversores ideais devem ser senoidais. No entanto, na prática, as formas de ondas obtidas são distorcidas e com conteúdo harmônico. As formas de ondas quadradas ou quase – quadradas são aceitáveis em aplicações de médias e baixas potências. Todavia, para aplicações de alta potência são necessárias formas de onda com baixo conteúdo harmônico.

A presença de harmônicos na saída do inversor pode ser reduzida utilizando dispositivos semicondutores de alta velocidade ou ainda pelas técnicas de chaveamento utilizadas no comando do chaveamento do inversor.

Nos dias atuais, os inversores têm sido utilizados amplamente em: acionamentos de máquinas *CA* em velocidade variável; aquecimento indutivo; fontes auxiliares; fonte ininterrupta (Nobreak ou UPS); compensadores estáticos; filtros ativos; transmissão em alta tensão *CC*, entre outros.

Os dispositivos de chaveamento utilizados nos inversores possuem disparo e/ou bloqueio controlado, são eles: O GTO (“gate turn-off”) utilizadas nas aplicações de média a alta potência e de baixas a média freqüência; o transistor a junção bipolar de alta potência (BJT) usadas nas aplicações de baixa a média potência e freqüência; o MOSFET (transistor a efeito de campo MOS de potência) com uso nas aplicações de baixa potência e freqüência muito elevadas; o transistor bipolar a gatilho isolado (IGBT) que suporta maiores tensões e podem operar em mais altas freqüências que os

transistores bipolares de potência e podem suportar maiores tensões e correntes que os MOSFET's de potência, entre outros.

A classificação dos inversores se dá através de vários critérios como: o número de fases; a utilização de dispositivos semicondutores de potência; os princípios de comutação; as formas de ondas de saída e a corrente e tensão de entrada. Neste trabalho, serão analisados apenas os inversores de fonte de tensão monofásicos, meia ponte e em ponte completa, bem como as técnicas de controle de chaveamento empregadas nestes tipos de conversores.

2.1 Inversor em meia – ponte

O inversor em meia ponte, usado em aplicações de baixa potência, é o alicerce básico dos circuitos inversores. O circuito deste inversor consiste de duas chaves (S_1 e S_2) conectadas em série com duas fontes de alimentação *CC*. Durante o período de 0 a $T/2$, a chave S_1 conduz, fazendo com que a tensão na carga seja $V_0 = V_s/2$. No instante $T/2$, S_1 fica aberta e S_2, fecha. No intervalo de $T/2$ a T, a tensão de saída é $V_0 = -V_s/2$. Como se pode ver, essa tensão tem uma forma de onda retangular alternada com freqüência $f = 1/T$. O circuito de comando do chaveamento deve projetado de tal forma que as chaves S_1 e S_2 não conduzam simultaneamente.

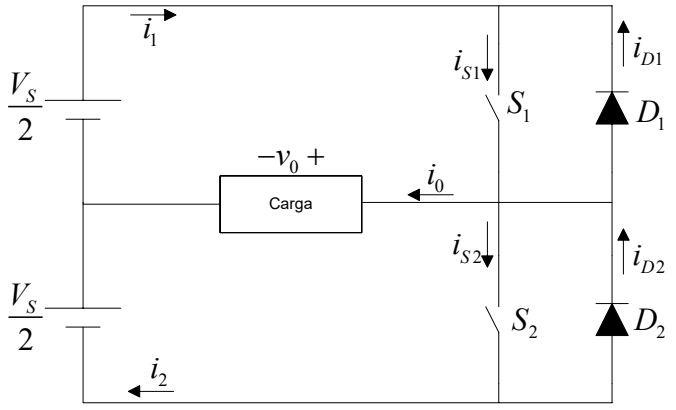

Figura 2.1.1: Inversor monofásico em meia-ponte.

Se a carga presente no inversor for indutiva, a corrente não pode mudar instantaneamente junto com a tensão de saída. Quando S_1 for aberta em $t = T/2$, a corrente de carga continuará a fluir por D_2, pela carga e pela metade inferior da fonte *CC* até que a corrente se anule. Da mesma forma, quando S_2 for aberta em $t = T$, a corrente de carga fluirá através de D_1, da carga e da metade superior da fonte. Os diodos D_1 e D_2 são conhecidos como *diodos de realimentação*, *diodos de recuperação* ou ainda *diodos de roda livre*. Quando estes conduzem, a potência flui da carga para fonte, portanto os inversores devem ser capazes de operar nos quatros quadrantes

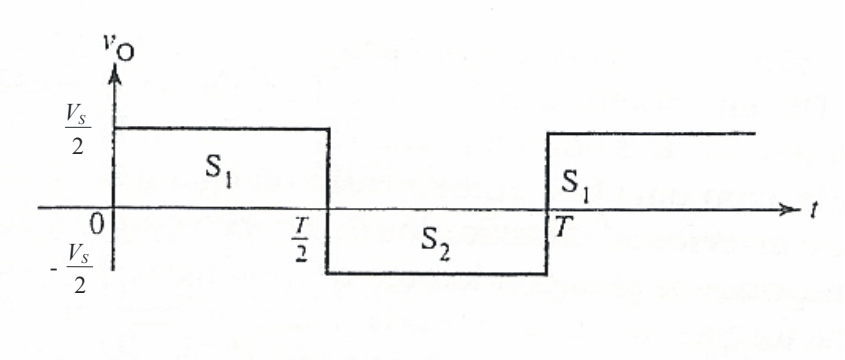

Figura 2.1.2: Forma de onda de saída com carga resistiva [2].

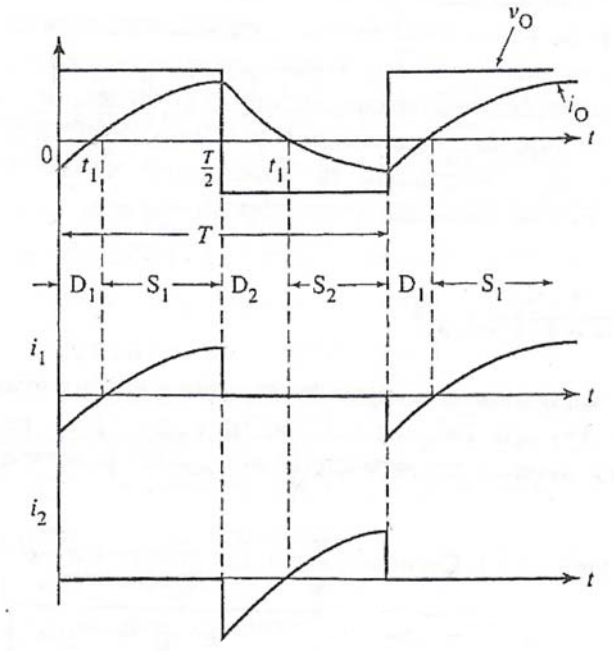

Figura 2.1.3: Formas de onda com carga indutiva [2].

A tensão eficaz de saída pode ser encontrada a partir da seguinte equação:

$$V_0 = \left(\frac{2}{T} \int_0^{T/2} \frac{V_s^2}{4} dt \right)^{1/2} = \frac{V_s}{2} \tag{1}$$

Desde que a forma de onda da tensão de pólo é uma onda quadrada de amplitude $V_s / 2$ sua expansão em série de Fourier, envolvendo harmônicos ímpares, é dado por:

$$v_0 = \sum_{n=1,3,5,\ldots}^{\infty} \frac{2V_s}{n\pi} sen\left(n\omega t\right) \tag{2}$$

$$v_0 = 0 \text{ para } n = 2,4,\ldots$$

Onde $\omega = 2\pi f_0$ é a freqüência da tensão de saída em rad/s. Para $n = 1$, encontra-se o valor eficaz da componente fundamental como:

$$V_1 = \frac{2V_s}{\sqrt{2}\pi} = 0,45V_s \tag{3}$$

Para uma carga indutiva, a corrente carga i_0 pode ser encontrada a partir da seguinte equação:

$$i_0 = \sum_{n=1,3,5,\ldots}^{\infty} \frac{2V_s}{n\pi\sqrt{R^2 + \left(n\omega L\right)^2}} sen\left(N\omega t - \theta_n\right) \tag{4}$$

Em que $N = tg^{-1}\left(n\omega L / R\right)$.

2.2 Inversor monofásico em ponte completa

O inversor monofásico em ponte completa é constituído de quatro chaves e quatro diodos de retorno. Nesse inversor a amplitude da tensão de saída e, portanto a potência de saída é o dobro do modelo em meia – ponte. As chaves são ligadas e desligadas aos pares em diagonal (uma variação dessa operação é obtida, quando

apenas uma chave é bloqueada do par em diagonal). Dessa forma, ou as chaves S_1 e S_4 ou as S_2 e S_3 vão para o estado ligado em um semiciclo $(T/2)$, ficando assim a fonte ligada de maneira alternada à carga. Se os pares de chaves diagonais passarem para o estado ligado em tempos iguais, teremos na saída uma onda quadrada com um pico de amplitude V_s.

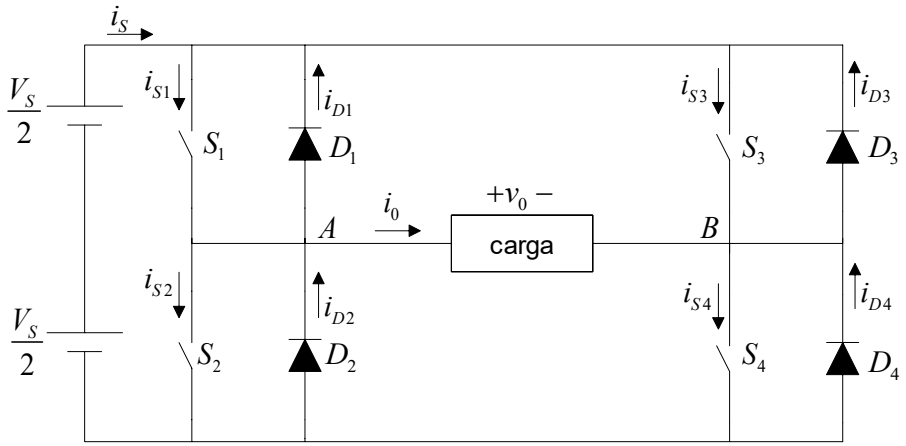

Figura 2.2.1: Inversor de fonte de tensão em ponte completa

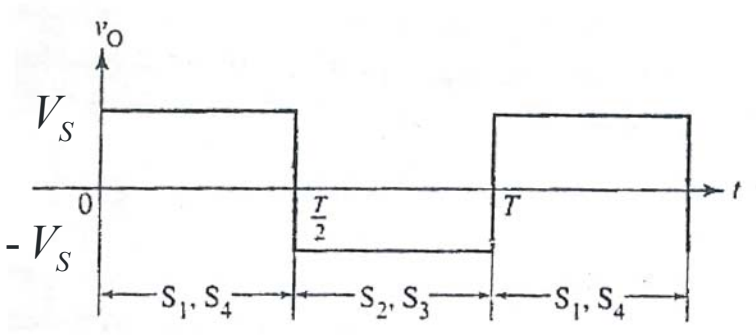

Figura 2.2.2: Forma de onda de saída com carga resistiva [2].

Quando se passa do estado de condução para o bloqueio (e vice-versa), ambas as chaves de um mesmo braço, devem estar desligadas por um período curto de tempo, evitando dessa forma que ocorra um curto na fonte *CC*. Portanto o tempo que as chaves levam para passar do estado ligado para o desligado deve ser o menor

possível, enquanto que quando passam do estado desligado para o ligado deve ser dado um retardo apropriado.

Com uma carga indutiva, a forma de onda da corrente na saída tem a forma exponencial. Quando a tensão na carga é positiva, a corrente cresce exponencialmente. No semiciclo seguinte quando a tensão assume um valor negativo, a corrente decresce também de forma exponencial.

Quando as chaves S_2 e S_3 são abertas em $t = 0$, os diodos D_1 e D_4 começam a conduzir. A corrente da carga começa em um valor negativo e cresce exponencialmente a uma taxa dada pela constante de tempo da carga $(t = L/R)$. Chegando a corrente de saída à zero, D_1 e D_4 param de conduzir e as chaves S_1 e S_4 entram em condução, a corrente continua a crescer atingindo seu valor máximo em $t = T/2$. Neste instante S_1 e S_4 se abrem e a tensão de saída se inverte. A corrente na carga continua no mesmo sentido, fluindo agora por D_2 e D_3. Quando a corrente de carga chega à zero, as chaves S_2 e S_3 começam a conduzir. Sempre que os diodos de recuperação estão conduzindo, a carga fornece energia à fonte.

A tensão eficaz de saída pode ser encontrada a partir de:

$$V_0 = \left(\frac{2}{T} \int_0^{T/2} V_s^2 dt \right)^{1/2} = V_s \tag{5}$$

A tensão de saída expandida em série de Fourier é dada por:

$$v_0 = \sum_{n=1,3,5,\ldots}^{\infty} \frac{4V_s}{n\pi} sen(n\omega t) \tag{6}$$

Pode-se verificar, da expressão (6), que o valor RMS da componente fundamental da tensão na carga é:

$$V_1 = \frac{4V_s}{\sqrt{2}\pi} = 0,90V_s \tag{7}$$

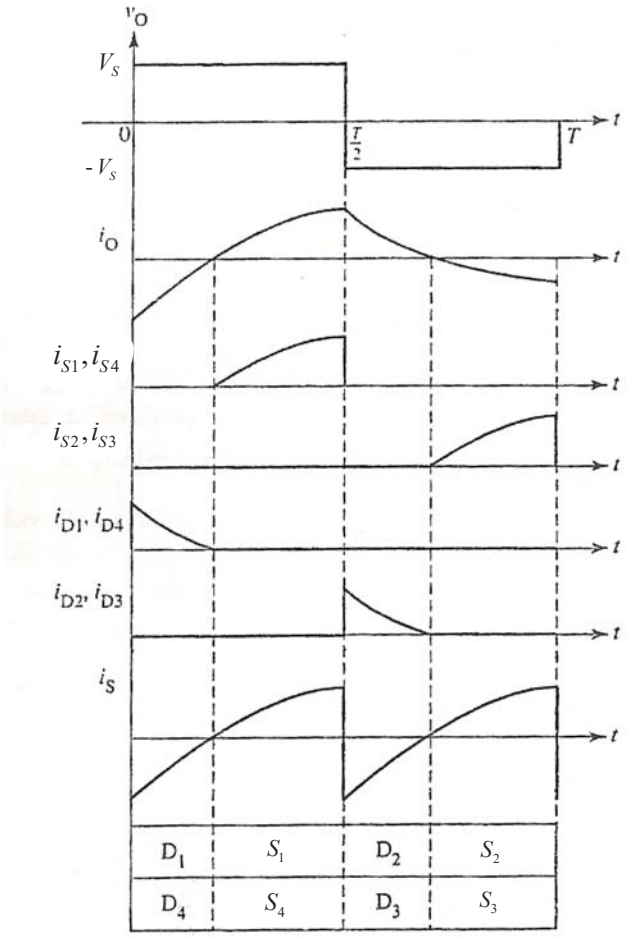

Figura 2.2.3: Formas de onda com carga indutiva [2].

Para uma carga indutiva, a corrente na carga defasada da tensão aplicada, é dada pela seguinte equação:

$$i_0 = \sum_{n=1,3,5,\ldots}^{\infty} \frac{4V_s}{n\pi\sqrt{R^2 + (n\omega L)^2}} sen(n\omega t - \theta_n) \tag{8}$$

Onde: $\theta_n = tg^{-1}\left(n\omega L / R\right)$.

2.3 Técnicas de modulação em inversores monofásicos

Em muitas aplicações industriais, é sempre necessário controlar a tensão de saída em inversores para solucionar alguns problemas, tais como: variações da tensão de entrada *CC*; regulação de tensão em inversores e para manutenção da relação tensão/freqüência. Para o controle da tensão, são usadas várias técnicas de modulação, dentre os quais o mais difundido pela sua simplicidade é a modulação por largura de pulso (PWM).

O PWM tem como característica o fato de que o instante de disparo dos interruptores do inversor é definido pela interseção de uma onda triangular (portadora) com um sinal modulante na freqüência da fundamental desejada no pólo do braço do inversor.

Com relação ao número de quantidade de pulsos por semiciclo de chaveamento, pode se ter a modulação por largura de pulso único ou a modulação de pulsos múltiplos. A técnica de modulação empregada classifica-se quanto ao tipo de forma da onda modulante por: modulação por largura de pulso de onda quadrada; modulação por largura de pulso de onda triangular; modulação por largura de pulso de onda trapezoidal; modulação por largura de pulso de onda senoidal; etc.

A seguir é discutida o PWM senoidal (SPWM), já que a maioria das aplicações dos inversores (motores *CA*, por exemplo) é projetada para operar com tensões senoidais, a tensão de saída do inversor devendo ser o mais senoidal possível.

A idéia do SPWM é obter uma forma de onda pulsada cuja componente fundamental se aproxime o máximo possível da senoide de referência. Em outras palavras, procura-se obter em cada segmento um pulso cuja altura seja definida pela alimentação do inversor e cuja área equivalha à área da senoide no mesmo intervalo. O esquema SPWM é ilustrado na figura 2.3.1 a seguir:

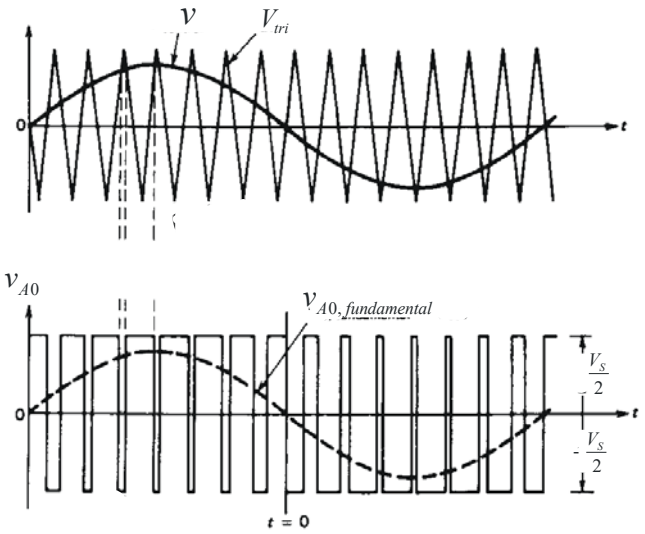

Figura 2.3.1: Esquema de modulação por largura de pulso senoidal [3].

O sinal triangular tem geralmente uma amplitude constante, A_T, e uma freqüência de chaveamento (ou freqüência de da portadora) f_T. O sinal de controle, A_r usado para modular o ciclo de operação, possui uma freqüência f_r (também chamada freqüência da modulante) que é a freqüência fundamental desejada da tensão de saída do inversor. Para um pico de amplitude do sinal de controle, A_r o índice de modulação em amplitude é definido como

$$m_a = \frac{A_r}{A_T} \tag{9}$$

O índice de modulação em amplitude calculado como

$$m_f = \frac{f_T}{f_r} \tag{10}$$

A saída do sinal senoidal é composta por uma componente fundamental mais harmônicas. A amplitude da componente fundamental é calculada como

$$V_{Ao} = m_a \frac{V_s}{2} \tag{11}$$

As componentes harmônicas são calculadas em função do índice de modulação em freqüência e da amplitude da fonte de tensão E_d. As harmônicas contidas no sinal de saída são determinadas nas equações (12) e (13) a seguir.

$$n = lm_f \pm k \tag{12}$$

$$l = 1,3,5,... \text{ e } k = 2,4,6,...$$

$$V_{nm} = 0,9\frac{V_s}{n} \tag{13}$$

A modulação SPWM possui como características, poder alterar a amplitude, a freqüência do sinal de saída e o conteúdo harmônico através do índice de modulação em amplitude e o índice de modulação em freqüência.

Para índice de modulação em freqüência, m_f, menor que 21 deve-se sincronizar o sinal modulante e a portadora atribuindo-se um número inteiro a m_f. Assim elimina-se as componentes sub-harmônicas e diminui-se a distorção do sinal de saída. Entretanto para m_f inteiro e maior que 21 os efeitos das componentes sub-harmônicas são desprezíveis.

A amplitude do sinal de saída pode ser aumentada em 27% utilizando-se a técnica denominada sobremodulação, que consiste na atribuição de índice de modulação em amplitude maior que 1. São ilustrados os gráficos da técnica nas figuras 2.3.2 e 2.3.3.

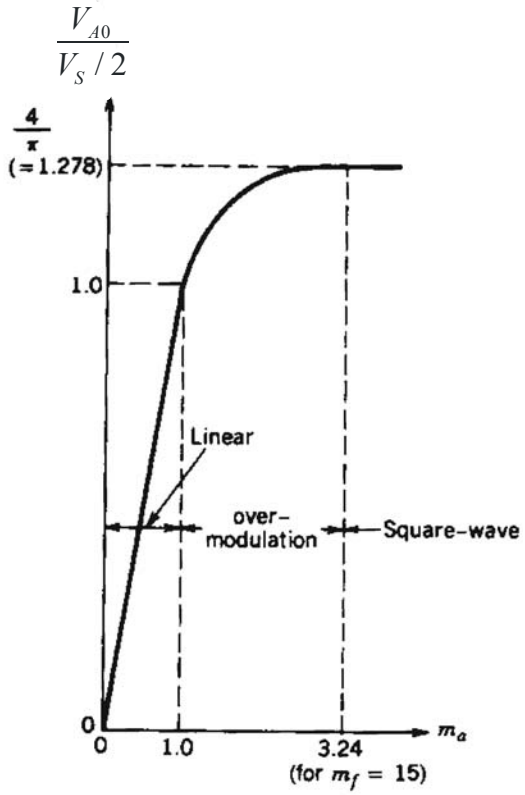

Figura 2.3.2: Amplitude da saída em função de m_a [3].

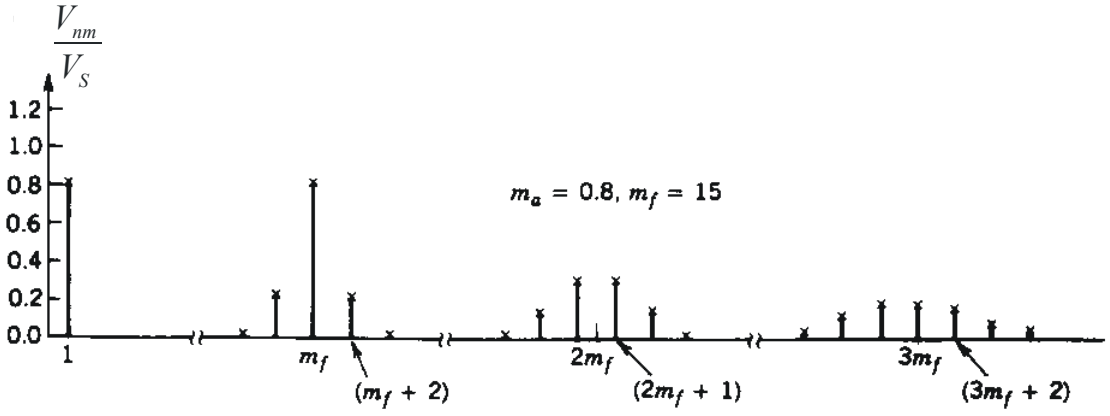

Figura 2.3.3: Espectro de freqüência do sinal de saída [3].

3. DESCRIÇÃO DO PROJETO

O projeto da PCI do circuito de comando do inversor é parte da execução do projeto de um inversor, que como um todo foi dividido em duas partes: a unidade de potência e a unidade lógica

A unidade de potência agrupa um retificador, um filtro e um inversor, e a unidade lógica por sua vez agrupa o microcontrolador e a interface homem-máquina.

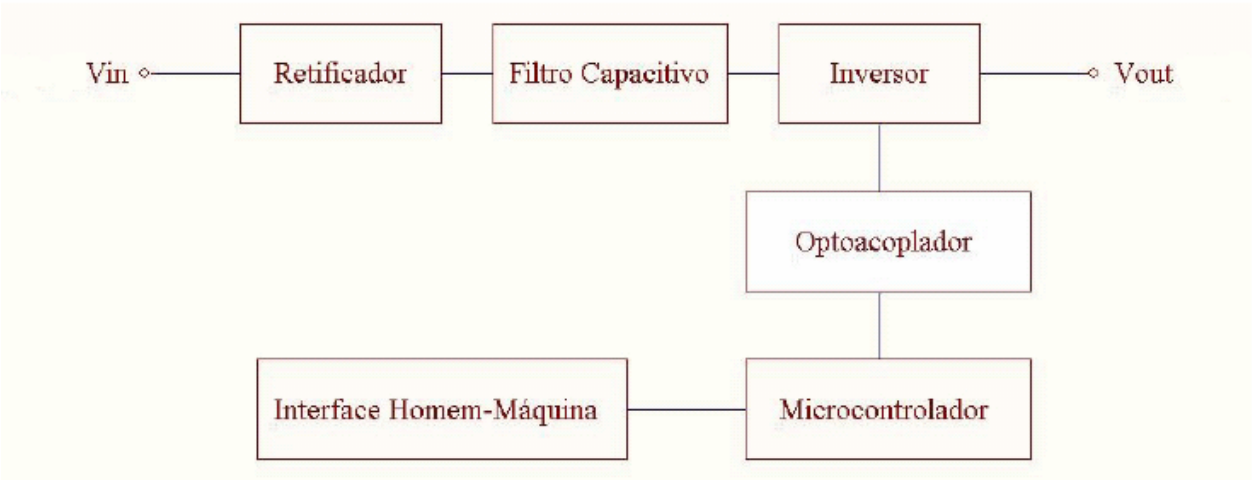

Figura 3.1: Unidades lógica e de potência [4].

O conversor projetado utiliza o esquema de modulação por largura de pulso (PWM) para acionamento das chaves. Entretanto, para fins didáticos também utiliza-se o esquema de modulação por onda quadrada.

As características do sinal de saída como amplitude, freqüência, sobremodulação e “tempo morto” são ajustadas através da interface homem-máquina, que consiste em um teclado (3 botões) e um LCD (2 linhas com 16 caracteres por linha).

3.1. Circuitos

Foi adicionado à placa da unidade lógica, um circuito de gravação para o microcontrolador. Este gravador possui a técnica de gravação *In-Circuit Serial Programing (ICSP)* [7], desenvolvida pela Microchip onde o dispositivo programável é programado depois de colocado no circuito, ou seja, a gravação é feita na própria placa do circuito. O gravador escolhido foi desenvolvido pela empresa OLIMEX, disponibilizado na sua página da internet [11].

O gravador ICSP faz uso de seis pinos do PIC (Peripheral Interface Controller). São eles: o VPP (ou MCLR), por onde é fornecida a tensão de programação; o VDD, que é o pino de alimentação do PIC; o GND onde é conectado o terra do microcontrolador; os pinos PGC e PGD, por onde são enviados os sinais de sincronismo e dados respectivamente; e por último o PGM, que é usado para habilitar os modos de gravação em alta ou baixa tensão.

O circuito de gravação é alimentado através da porta serial. O modo de gravação escolhido é o de alta tensão, quando aplica-se um sinal de 13,5 volts a porta MCLR e 5 volts ao pino de alimentação do microcontrolador respectivamente e aterra-se o pino PGM.

O dispositivo de entrada da interface homem-máquina consiste em um teclado com 3 botões, que utiliza lógica negativa. O estado do teclado é capturado através da técnica de *polling*. O filtro para eliminar o *debounce* está implementado em *software*.

O dispositivo de saída da IHM consiste em um LCD de 2 linhas e 16 caracteres por linha com ajuste de contraste e iluminação de fundo. Em tempo real conforme a necessidade de uso exibe-se as informações relevantes.

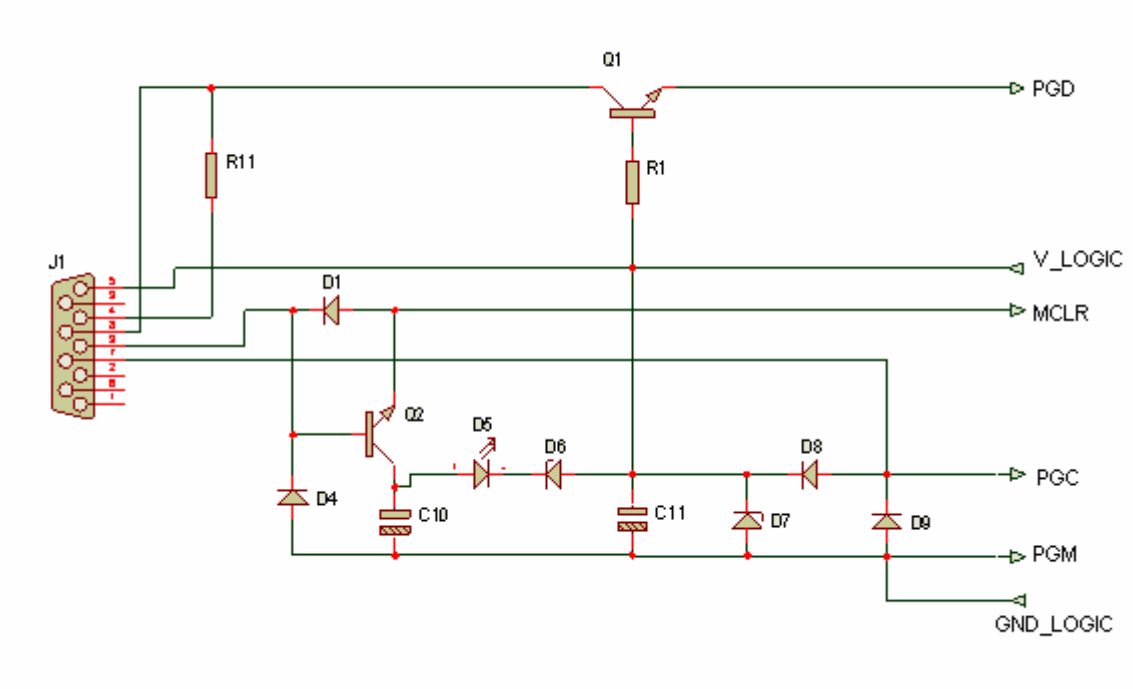

Figura 3.1.1: Circuito ICSP (In-Circuit serial programming)

O LCD utilizado na montagem é o modelo FDCC1602E da FORDATA ELETRONIC CO.,LTDA. Esse dispositivo possui oito pinos para entrada de dados, no entanto, ele possibilita o uso de apenas quatro pinos de dados. Optou-se por utilizar apenas quatro pinos em função da simplificação do circuito.

As principais funcionalidades do LCD são:

- Contraste ajustável;
- Alimetação de 5 V;
- 8 pinos para entrada de dados, podendo ser usado apenas 4;
- *"Duty Cycle"* de 1/16.

O microcontrolador selecionado é o 18F4550 onde utiliza arquitetura *RISC* da Microchip [8], com baixo consumo de energia. A escolha deste modelo se deu devido o mesmo dispor de diversas funções como: multiplicador em *"hardware"*, instruções de manipulação de contexto e interrupções com atribuição de prioridade, entre outros, além de trabalhar com uma freqüência de até 48MHZ. Estas características permitiram

a redução no tempo de execução da tarefa e complexidade do código. Uma vez que este código além de realizar a função IHM, também gera PWM que é uma tarefa que requer uma boa quantidade de memória e capacidade de processamento do microcontrolador.

As principais características do PIC 18F4550, são:

- 32 kbytes de memória flash;
- 1536 bytes de memória de dados volátil (RAM);
- 256 bytes de memória de dados não volátil (E²PROM);
- 15 interrupções;
- 31 portas I/O's;
- 4 timers (1 de 8 bits e 3 de 16 bits);
- 2 Capture/Compare/PWM;
- USART;
- Porta USB;
- Clock de até 48MHZ;
- 13 canais de conversão A/D com 10 bits cada.

Figura 3.1.2: PIC 18F4550 da Microchip

O diagrama elétrico da interface homem-máquina e microcontrolador é ilustrado na figura 3.1.3:

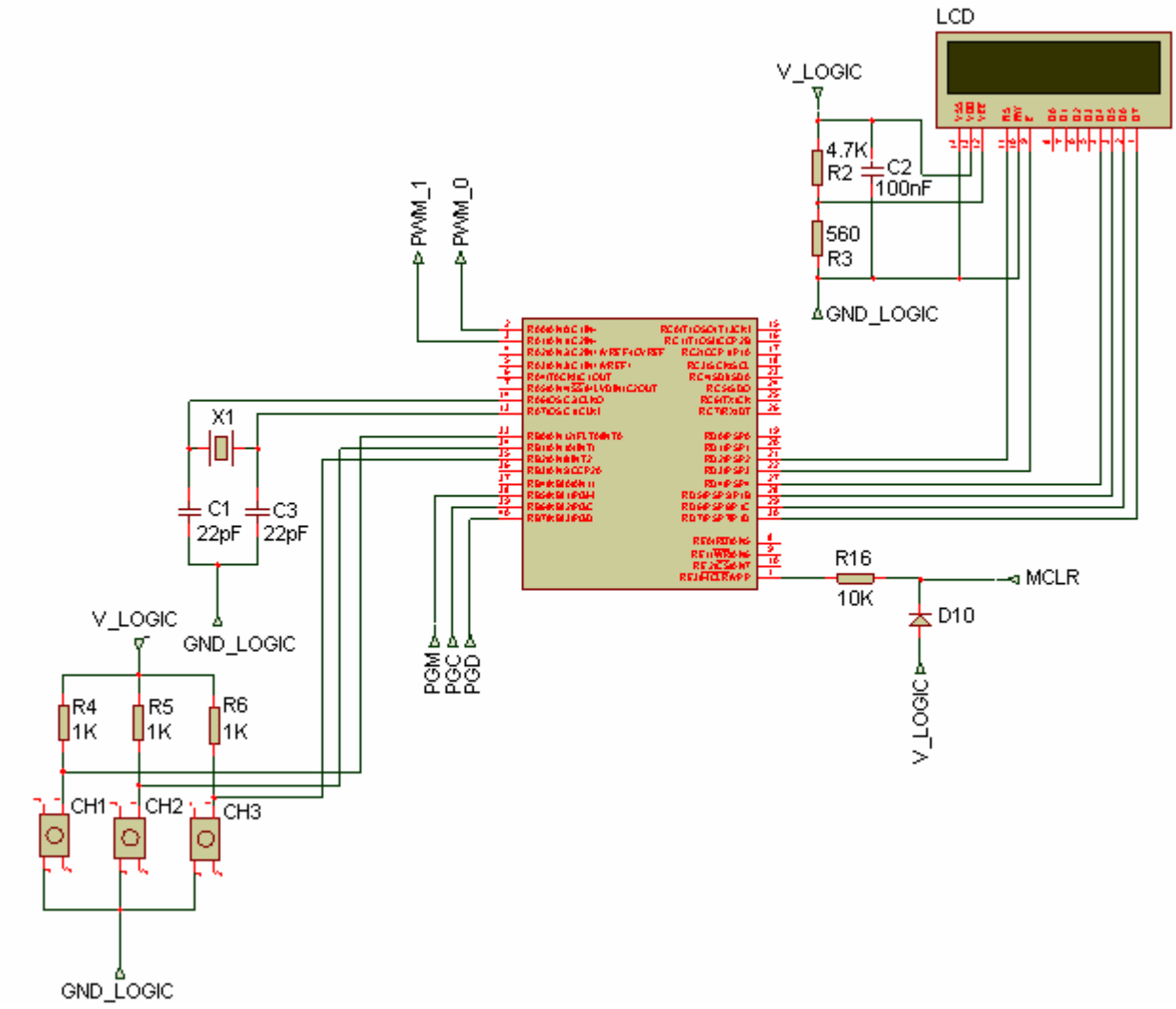

Figura 3.1.3: IHM e microcontrolador.

O circuito para isolamento elétrico entre a unidade lógica e de potência consiste em um *buffer* e um optoacoplador devidamente alimentados. Foi utilizado o *buffer*, na saída do microcontrolador apenas como forma de precaução para garantir que a corrente na entrada do optoacoplador fosse a mínima recomendada. No entanto o circuito foi testado sem a presença do *buffer* e funcionou de modo adequado ao

desejado, pois o microcontrolador usado fornece a corrente mínima requerida para o correto funcionamento do optoacoplador. Portanto, o uso do *buffer* é opcional. Além de o *buffer* desempenhar a função descrita acima, ele também oferece também um certo isolamento entre as unidades lógicas e de potência.O modelo de dispositivo para implementação do *buffer* foi o SN74LS17 da Texas Instruments [10].

O optoacoplador constitui o dispositivo principal no circuito de isolamento. O modelo utilizado foi o 6N137 da Fairchild Semiconductor [9]. Este dispositivo consiste de um LED AlGaAS de 850 nm, opticamente acoplado com uma porta lógica fotodetectora de alta velocidade, sua saída é em coletor aberto fazendo-se necessário o uso de resistores, ligados do pino de alimentação VCC ao coletor do transistor de saída. Como esse optoacoplador possui apenas um canal, foram utilizados quatro dispositivos já que o inversor é monofásico em ponte e necessita de quatro sinais de comando para o seu funcionamento. Para alimentação foram utilizados dois reguladores de tensão 74L05, pois é recomendado na folha de dados do dispositivo que o 6N137 seja alimentado com 5 V e não é possível a utilização da mesma fonte de 5 V que alimenta a unidade lógica porque. O uso da mesma fonte resulta na perda da isolação do optoacoplador, e a solução portanto foi utilizar reguladores de tensão, cujas entradas foram conectadas às fontes de tensão de 15 V utilizadas para alimentar os acionadores do inversor.

As principais características do optoacoplador 6N137, são listadas a seguir:

- Transmissão de sinais de até 10Mbits/s;
- CMR superior a 10KV/ μ s;
- Saída com porta lógica;
- Saída em coletor aberto;

Para fazer o acionamento das chaves do inversor foi utilizado o acionador IR2110 da International Rectifier [5]. É um acionador de alta tensão que trabalha com alta velocidade de chaveamento, usado no chaveamento de MOSFETs de potência e IGBTs. Possui duas entradas e duas saídas independentes, portanto, utiliza-se apenas

um acionador por braço de inversor em ponte. As chaves são acionadas em função do estado das entradas HIN e LIN. O IR2110 possui três pontos de alimentação, o VDD que alimenta o circuito de entrada e as entradas VB e VCC usadas para alimentação dos dois circuitos de saída do dispositivo. O VDD foi alimentado com uma tensão de 5V, advinda dos reguladores usados para alimentar os optoacopladores. E para alimentação dos circuitos de saída dos acionadores foram utilizadas duas fontes de 15 V, uma para cada acionador.

Como principais características do acionador IR2110, pode-se citar:

- Suporta uma tensão de bloqueio em cada chave de 500 V, suficiente para aplicação;
- Fornece uma corrente pulsada de curto – circuito de até ± 2 A para as chaves;
- Tensão V_{gs} fornecida de 3,3 V até 20 V;
- $T_{on/off}(typ) = 120 \,\&\, 94ns$ e delay de $10\mu s$;
- Alimentação de 10 a 20 V;
- Entrada CMOS Schmitt – triggered, que permite instaurar um tempo morto analógico;
- Baixa potência consumida;
- Altos limites térmicos $150^{o}C$.

O circuito de acionamento completo (circuito de isolamento elétrico mais acionador) de um dos braços do inversor é ilustrado na figura 3.1.6.

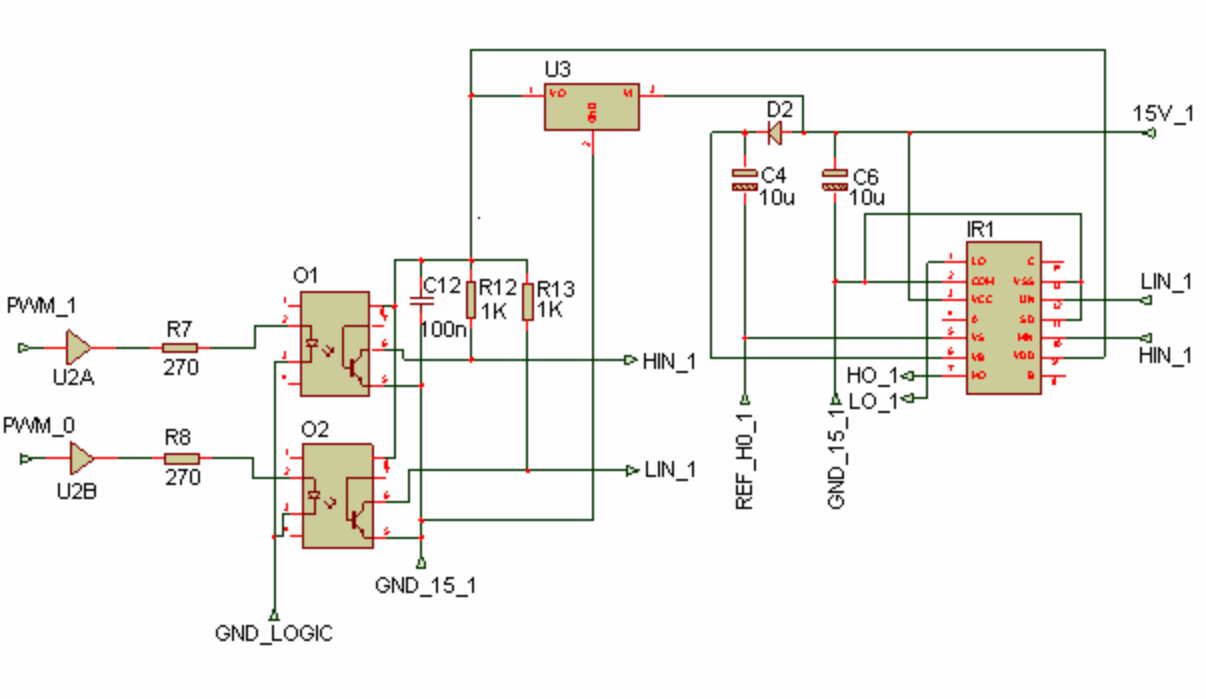

Figura 3.1.6: Circuito de acionamento de um braço do inversor

Todos os sinais elétricos de saída e entrada da placa passam por um conector DIN de 32 pinos. O esquema da ligação do conector com todos os sinais de entrada e saída é apresentado na figura 3.1.7.

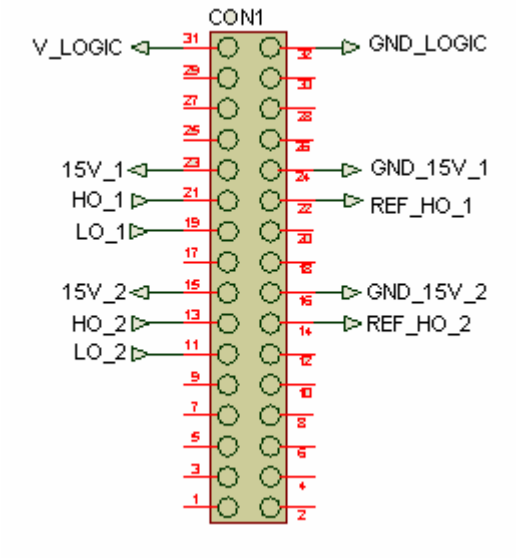

Figura 3.1.7: Conector DIN de 32 pinos

3.2. *Layout* da PCI

Para o desenvolvimento do "*layout"* da PCI foi utilizado o programa Proteus 7, que é uma suíte que agrega o ambiente de simulação de circuitos eletrônicos ISIS e o programa para desenho de circuito impresso Ares professional. O ISIS simula circuitos analógicos, digitais e microcontroladores.

Inicialmente foi feito o esquema elétrico de todo o circuito no ISIS, utilizando alguns modelos já existentes em sua biblioteca, no entanto alguns dispositivos não existiam na biblioteca, então foram criados alguns modelos necessários a montagem, já que o mesmo possui essa facilidade.

Depois da montagem do esquema elétrico, o circuito foi transferido para o Ares, onde foi projetado o layout. A exemplo do ISIS, o Ares possui uma biblioteca de modelos de dispositivos, assim quando se faz um modelo inexistente no ISIS, é necessário fazer o seu correspondente no Ares.

Na montagem do layout, procurou-se localizar os componentes de forma que o roteamento das trilhas ficasse o mais simples possível. O programa possui posicionamento automático dos componentes, no entanto, essa tarefa foi feita manualmente pelo motivo já exposto acima.

Uma vez alocado os componentes, foi feito o roteamento das trilhas. O Ares possui também o roteamento automático, algumas trilhas foram roteadas automaticamente e outras manualmente.

Como a placa possui uma quantidade razoável de componentes para sua dimensão (100mm x 160mm), foi utilizada uma placa de fibra de vidro duas faces facilitando dessa forma o roteamento.

Os Pad´s (região onde é feito o furo para soldar os componentes) foram feitos com 2mm de diâmetro e as trilhas com 1mm de largura.

Na figura 3.2.1, é mostrado o layout da PCI, as trilhas em cor azul estão localizadas na parte inferior da placa, em cor vermelha na parte superior.

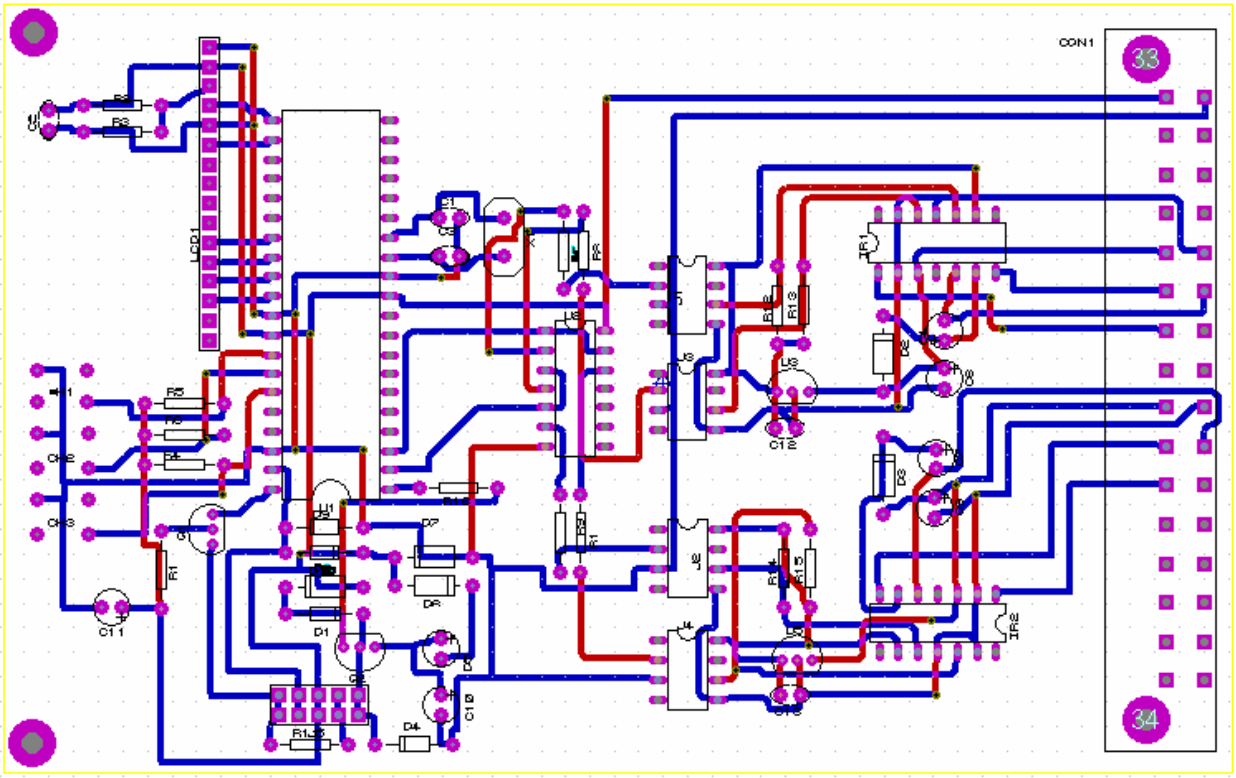

Figura 3.2.1: Layout da PCI

Para confecção da PCI, foi utilizada a Protomat S62 da LPKF Laser e Eletrocnics, esta máquina popularmente chamada de “fresa”, é uma ferramenta específica para fabricação de placas de circuito impresso. Possui dez posições para colocação das fresas, nos quais ela usa para fazer as trilhas, furar e cortar a placa.

A fresa é manuseada através de um computador, e necessita de alguns arquivos chamados de gerber para confecção da placa. Necessita de um arquivo para o corte das trilhas, um para os furos, outro para fazer o corte da placa, etc. Esses arquivos são gerados pelo programa usado para fazer o layout. Todos os programas atuais destinados a esse fim possuem essa funcionalidade.

Figura 3.2.2: Protomat S62 da LPKF Laser e Eletrocnics

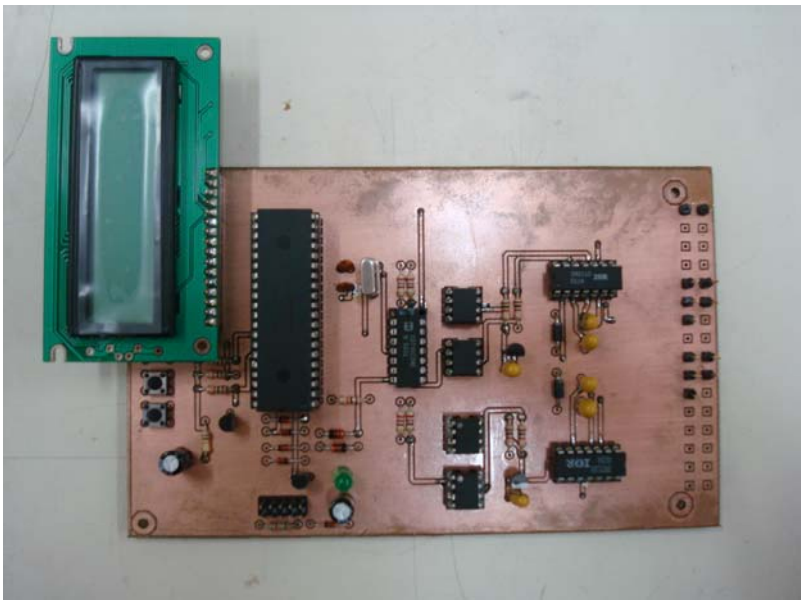

Figura 3.2.3: PCI de um circuito de comando para um inversor monofásico.

3.3. Programa

O programa utilizado para gerar os sinais PWM foi desenvolvido por Bruno Emmanuel de Oliveira Barros em seu estágio supervisionado do curso de Engenharia Elétrica da Universidade Federal de Campina Grande [4].

Foram feitas algumas modificações e adaptações no programa. A primeira modificação foi a mudança das portas de entrada e saída do microcontrolador que fazia a comunicação com o display de LCD, com a finalidade de facilitar o roteamento das trilhas. E a segunda alteração foi no sentido de acrescentar ao programa algumas diretivas de processamento, já que o microcontrolador utilizado é 18F4550 e possui muito mais funções e variáveis de inicialização que o modelo 16F877 a que o programa foi inicialmente destinado.

O algoritmo implementado permite o ajuste da amplitude e frequência do sinal de saída utilizando-se o esquema de modulação por largura de pulso senoidal (SPWM). Para fins didáticos permite-se a utilização do esquema de modulação por onda quadrada. Outras características do sinal de saída como sobremodulação e “tempo morto” também são realizáveis.

4. TESTE DE FUNCIONAMENTO

Primeiramente foi testado o gravador do microcontrolador, utilizando tarefas simples como reconhecimento do PIC pelo gravador, gravação, leitura e limpeza da memória. Para realização desses testes foram utilizados o programa WinPic.

Em seguida foi feito o teste da interface homem – máquina, alterando o índice de modulação em amplitude, o índice de modulação em freqüência e o “tempo morto” através do teclado de três teclas.

Por último foram verificados os sinais PWM nas saídas da placa com a ajuda de osciloscópio.

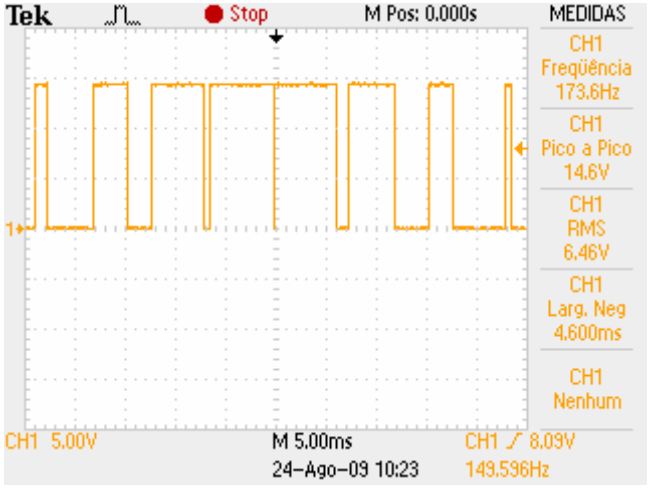

Figura 4.1: Forma de onda da saída LO_1(m_a = 0,5 e m_f = 9).

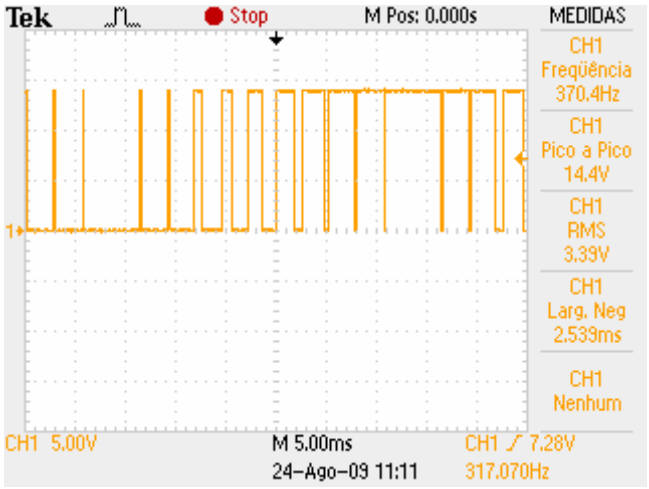

Figura 4.2: Forma de onda da saída HO_1(m_a = 1,0 e m_f = 21).

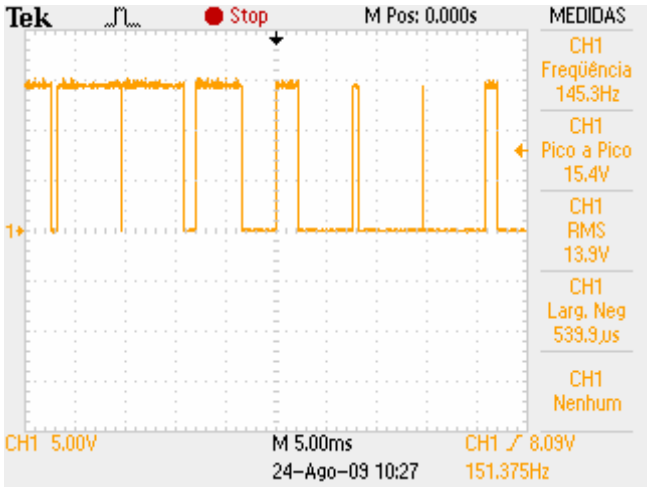

Figura 4.3: Forma de onda da saída LO_2(m_a = 0,5 e m_f = 9).

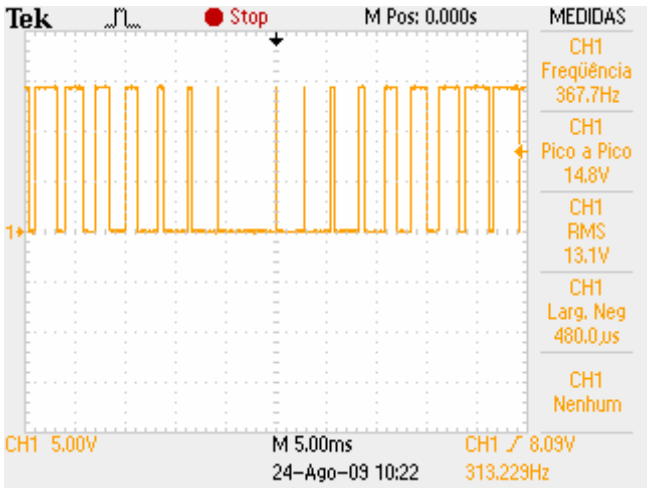

Figura 4.4: Forma de onda da saída HO_2(m_a = 1,0 e m_f = 21).

5. CONSIDERAÇÕES FINAIS

O gravador utilizado na placa suporta microcontroladores PIC de 8,18, 28 e 40 pinos, no entanto utilizou-se apenas um soquete de 40 pinos. Significando que outros PIC´s de 40 pinos podem ser usados, devendo-se apenas ter cuidado com a capacidade de processamento do microcontrolador, o qual depende do programa utilizado para gerar os sinais PWM e com a pinagem utilizada no circuito.

A dificuldade de roteamento das trilhas é diretamente proporcional a sua espessura, entretanto é extremamente recomendado desprender um pouco mais de tempo nesta etapa, com a finalidade de facilitar o processo de soldagem dos componentes.

Antes de fazer a PCI, todo o circuito foi montado em um *“protoboard”* (Matriz de contato) para verificação do seu funcionamento. Inicialmente foi utilizado o optoacoplador 6N136. Entretanto, o sinais PWM apresentaram distorções. Uma solução foi utilizar o optoacoplador 6N137 que opera com uma freqüência de sinais dez vezes maior que o 6N136.

Em termos gerais, pode-se concluir através das formas de ondas coletadas, que a PCI funcionou de maneira bastante satisfatória, uma vez que os sinais de saída da placa não apresentaram distorções.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Rashid, Muhammad H. Eletrônica de potência: circuitos, dispositivos e aplicações; tradução Carlos Alberto Favato; revisão técnica Antonio Pertence Júnior. – São Paulo: Makron Books, 1999.

[2] Ahmed, Ashfaq. Eletrônica de potência; tradução Bazán Tecnologia e Lingüística; revisão técnica João Antonio Martino. —São Paulo: Pearson Prentice Hall, 2000.

[3] Mohan, Ned; Undeland, Tore M.; Robbins, William P. Power Eletronics: converters, applications, and design. 2° edição. Toronto: John Wiley & Sons, Inc.

[4] de Oliveira Barros Luna, Bruno Emmanuel. “Conversor CC-CA - Inversor”. Campina Grande: UFCG, 2008. Relatório de Estágio.

[5] IR2110 Datasheet. <www.irf.com/product-info/datasheets/data/ir2110.pdf>. Acesso em: 29.07.2009.

[6] Application Note da International Rectifier. Disponível em <www.irf.com/technical- info/appnotes/an 978.pdf >. Acesso em: 29.07.2009.

[7] In-Circuit Serial Programming (ICSP) Guide. Disponível em <http://ww1.microchip.com/downloads/en/DeviceDoc/30277d.pdf>. Acesso em: 29.07.2009.

[8] PIC18F4550 Datasheet. Disponível em <http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf>. Acesso em: 29.07.2009

[9] Optocoupler 6N137 Datasheet. Disponível em <http://www.fairchildsemi.com/ds/6N/6N137.pdf>. Acesso em: 29.07.2009.

[10] Hex buffer SN74LS17 Datasheet. Disponível em <http://focus.ti.com/lit/ds/symlink/sn74ls17.pdf>. Acesso em: 29.07.2009.

[11] Gravador de PIC PG2. Disponível em <http://www.olimex.com/dev/index.html>. Acesso em: 29.07.2009

7. ANEXO

```
/*****************************************************************************

    Universidade Federal de Campina Grande
    Centro de Engenharia Elétrica e Informática
    Unidade Acadêmica de Engenharia Elétrica

      Trabalho de conclusão de curso

      Autor: Bruno Emmanuel de O.B. Luna
      Modificado por: João Helder Gonzaga Muniz

      Natural PWM
      ------------------------------

      Inicialização dos Periféricos
      ------------------------------
        A/D - Desligado
        SPI - Desligado
        I2C - Desligado
        CCP - Desligado
        USART - Desligado
        MSSP - Desligado
        TIMER 0 - Interno, 1:256
        TIMER 1 - Desligado
        TIMER 2 - Fosc/4, 48, 1 = 19.6us
      Prioridades das Interrupções
        TIMER 0 - Baixa
        TIMER 2 - Alta
      Configuração das Portas PWM
        Porta A (0, 1)
      Configuração das Portas Botões
        Porta B (0, 1, 2)
      Configuração das Portas LCD
        Porta D (2, 3, 4, 5, 6, 7)
      Configuração do Registradores TRISX
        TRISA - 0x00
        TRISB - 0x03
        TRISC - 0x00

      Interface Homem-Máquina
      ------------------------------
        3 Submenus
          MA_SUBMENU - Ajustar Índice de Modulação em Amplitude
          MF_SUBMENU - Ajustar Índice de Modulação em Amplitude
          DT_SUBMENU - Ajustar 'Dead Time'
        Limites do Menu
          SUBMENU_ITENS - Número de submenus

*****************************************************************************/
```

```
#include <18F4550.h>
#device adc=8

#FUSES NOWDT                 //No Watch Dog Timer
#FUSES WDT128                //Watch Dog Timer uses 1:128 Postscale
#FUSES HS                    //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPROTECT             //Code not protected from reading
#FUSES BROWNOUT              //Reset when brownout detected
#FUSES BORV20                //Brownout reset at 2.0V
#FUSES NOPUT                 //No Power Up Timer
#FUSES NOCPD                 //No EE protection
#FUSES NOSTVREN              //Stack full/underflow will not cause reset
#FUSES NODEBUG               //No Debug mode for ICD
#FUSES LVP                   //Low Voltage Programming on B3(PIC16) or B5(PIC18)
#FUSES NOWRT                 //Program memory not write protected
#FUSES NOWRTD                //Data EEPROM not write protected
#FUSES NOIESO                //Internal External Switch Over mode disabled
#FUSES NOFCMEN               //Fail-safe clock monitor disabled
#FUSES NOPBADEN              //PORTB pins are configured as digital I/O on RESET
#FUSES NOWRTC                //configuration not registers write protected
#FUSES NOWRTB                //Boot block not write protected
#FUSES NOEBTR                //Memory not protected from table reads
#FUSES NOEBTRB               //Boot block not protected from table reads
#FUSES NOCPB                 //No Boot Block code protection
#FUSES MCLR                  //Master Clear pin enabled
#FUSES LPT1OSC               //Timer1 configured for low-power operation
#FUSES NOXINST               //Extended set extension and Indexed Addressing mode disabled
(Legacy mode)
#FUSES PLL5                  //Divide By 5(20MHz oscillator input)
#FUSES CPUDIV1               //No System Clock Postscaler
#FUSES NOUSBDIV              //USB clock source comes from primary oscillator
#FUSES VREGEN                //USB voltage regulator enabled
#FUSES NOICPRT               //ICPRT disabled

#use delay(clock=40000000)
#define SRAM_SCL   PIN_C3
#define SRAM_SDA   PIN_C4

#use fast_io (A)
#use fast_io (B)
#use fast_io (D)

// Registradores ------------------------------------------------------------
#define PORTA     0xF80
#define PORTB     0xF81
#define PORTD     0xF82
#define PIR1      0xF9E
#define TMR2IF    1
#define IPR1      0xF9F
#define TMR2IP    1
#define RCON      0xFD0
#define IPEN      7
#define TMR0L     0xFD6
#define TMR0H     0xFD7
#define STATUS    0xFD8
```

```
#define C        0
#define Z        2
#define BSR      0xFE0
#define FSR1L    0xFE1
#define FSR1H    0xFE2
#define INDF1    0xFE7
#define WREG     0xFE8
#define INTCON2  0xFF1
#define TMR0IP   2
#define INTCON   0xFF2
#define TMR0IF   2
#define PEIE_GIEL 6
#define GIE_GIEH  7
#define PRODL    0xFF3
#define PRODH    0xFF4
#define PCLATH   0xFFA
#define PCLATU   0xFFB

// Contexto ---------------------------
#define FAST     1
#define NOT_FAST  0

// Constantes do PWM ------------------------------------------------------------
// 0 - Carrier Slope
// 1 - Modulating Quadrant
// 2 - Carrier Sign
// 3 - Modulating Sign
// 4 - Dead Time
// 5 - Next PWM Value
#define PWM_STATUS       0x80

// Configurar Saida do PWM ------------
#define PWM_OUTPUT       PORTA
#define CHANNEL_A        0
#define CHANNEL_B        1

// Constantes MENU --------------------------------------------------------------

// TimeOut Message ---------------------
#define TIME_OUT_RESET    0x01
#define TIME_OUT_SHOW     0x02
#define TIME_OUT_INCREMENT 0x03
#define TIME_OUT_SECONDS   5

// Submenu ----------------------------
#define MA_SUBMENU       0x01
#define MF_SUBMENU       0x02
#define DT_SUBMENU       0x03
#define SUBMENU_ITENS     0x03

// Adições ----------------------------------------------------------------------
#include <lcd_16x2.c>
#include <keyboard_3x1.c>

// Protótipos -------------------------------------------------------------------
```

```
// Inicialização ----------------------
void initialization (void);

// Interrupção ------------------------
void pwm_isr (void);

// Menu -------------------------------
void menu_time_out (void);
void reset_time_out (void);
void show_time_out_message (void);
void menu_service_routine (void);
void submenu_enter (void);
void show_submenu (void);
void adjust_ma (void);
void adjust_mf (void);
void adjust_dead_time (void);
void test_submenu_limit (void);

// Variáveis --------------------------------------------------------------

// Apontadores Registradores ----------
#locate fsr1h_pointer          = FSR1H

// MEMÓRIA RAM  - BANCO 00 ------------------------------------------------

// Variável de Estado do PWM ----------
#locate pwm_status_pointer      = 0x80

// Indices de Modulação ---------------
#locate mf                    = 0x81
#locate ma_low                = 0x82
#locate ma_high               = 0x83

// Auxiliar ---------------------------
float ma_float;
#locate ma_float              = 0x84

// Variáveis Modulante e Portadora ----
#locate modulating            = 0x88
#locate modulating_index      = 0x89
#locate carrier               = 0x8A
#locate result                = 0x8B

// Dead Time --------------------------
#locate dead_time             = 0x8C
#locate temp                  = 0x8D

// Contexto ---------------------------
#locate saved_WREG            = 0x8E
#locate saved_STATUS          = 0x8F
#locate saved_BSR             = 0x90

// Menu Time Out ----------------------
#locate time_out              = 0x91
```

```
#locate time_out_semaphore      = 0x92
#locate submenu                 = 0x93


// MEMÓRIA RAM  - BANCO 01 ----------------------------------------------------

// Tabela SENO ------------------------
int8 sine_table [0x100] = {
  0x00, 0x02, 0x03, 0x05, 0x06, 0x08, 0x09, 0x0B, 0x0D, 0x0E,
  0x10, 0x11, 0x13, 0x14, 0x16, 0x17, 0x19, 0x1B, 0x1C, 0x1E,
  0x1F, 0x21, 0x22, 0x24, 0x25, 0x27, 0x29, 0x2A, 0x2C, 0x2D,
  0x2F, 0x30, 0x32, 0x33, 0x35, 0x36, 0x38, 0x39, 0x3B, 0x3C,
  0x3E, 0x3F, 0x41, 0x43, 0x44, 0x46, 0x47, 0x49, 0x4A, 0x4C,
  0x4D, 0x4F, 0x50, 0x51, 0x53, 0x54, 0x56, 0x57, 0x59, 0x5A,
  0x5C, 0x5D, 0x5F, 0x60, 0x62, 0x63, 0x64, 0x66, 0x67, 0x69,
  0x6A, 0x6C, 0x6D, 0x6E, 0x70, 0x71, 0x73, 0x74, 0x75, 0x77,
  0x78, 0x7A, 0x7B, 0x7C, 0x7E, 0x7F, 0x80, 0x82, 0x83, 0x84,
  0x86, 0x87, 0x88, 0x8A, 0x8B, 0x8C, 0x8E, 0x8F, 0x90, 0x92,
  0x93, 0x94, 0x95, 0x97, 0x98, 0x99, 0x9A, 0x9C, 0x9D, 0x9E,
  0x9F, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA7, 0xA8, 0xA9, 0xAA,
  0xAB, 0xAC, 0xAE, 0xAF, 0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5,
  0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF, 0xC0,
  0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA,
  0xCB, 0xCC, 0xCD, 0xCE, 0xCF, 0xD0, 0xD0, 0xD1, 0xD2, 0xD3,
  0xD4, 0xD5, 0xD6, 0xD7, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC,
  0xDC, 0xDD, 0xDE, 0xDF, 0xDF, 0xE0, 0xE1, 0xE2, 0xE2, 0xE3,
  0xE4, 0xE4, 0xE5, 0xE6, 0xE7, 0xE7, 0xE8, 0xE8, 0xE9, 0xEA,
  0xEA, 0xEB, 0xEC, 0xEC, 0xED, 0xED, 0xEE, 0xEE, 0xEF, 0xF0,
  0xF0, 0xF1, 0xF1, 0xF2, 0xF2, 0xF3, 0xF3, 0xF4, 0xF4, 0xF4,
  0xF5, 0xF5, 0xF6, 0xF6, 0xF7, 0xF7, 0xF7, 0xF8, 0xF8, 0xF8,
  0xF9, 0xF9, 0xF9, 0xFA, 0xFA, 0xFA, 0xFB, 0xFB, 0xFB, 0xFC,
  0xFC, 0xFC, 0xFC, 0xFC, 0xFD, 0xFD, 0xFD, 0xFD, 0xFD, 0xFE,
  0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF,
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
#locate sine_table_ram = 0xFF

void main()
{
  initialization ();
  menu_service_routine ();
}

// Vetor de Interrupção -------------------------------------------------------

// Prioridade Alta --------------------
#org 0x0008, 0x000A
void high_priority (void)
{
  pwm_isr ();
}

// Prioridade Baixa -------------------
#org 0x0018, 0x0040
void low_priority (void)
{
```

```
  // Salvar Contexto -----------------
  saved_WREG = *WREG;
  saved_STATUS = *STATUS;
  saved_BSR = *BSR;
  *BSR = 0;

  if (*INTCON,TMR0IF)
    menu_time_out ();

  // Restaurar Contexto --------------
  *BSR = saved_BSR;
  *WREG = saved_WREG;
  *STATUS = saved_STATUS;
  bit_clear (*INTCON, TMR0IF);
  bit_set (*INTCON, PEIE_GIEL);
}

// Interrupção PWM ------------------------------------------------------------
void  pwm_isr (void)
{
  #asm
    save_context:
      clrf  BSR

    // Atualizar Saidas -------------
    test_pwm_state:
      btfsc PWM_STATUS.5
      goto  pwm_high
    pwm_low:
      nop
      bcf   PWM_OUTPUT.CHANNEL_A
      btfss PWM_STATUS.4
      call  dead_time_function, NOT_FAST
      bsf   PWM_OUTPUT.CHANNEL_B
      goto  end_pwm_set_function
    pwm_high:
      bcf   PWM_OUTPUT.CHANNEL_B
      btfsc PWM_STATUS.4
      call  dead_time_function, NOT_FAST
      bsf   PWM_OUTPUT.CHANNEL_A
    end_pwm_set_function:

    // Portadora Triangular ---------
    carrier_function:
      movf  mf, w
      btfsc PWM_STATUS.0
      goto  negative_slope

    positve_slope:
      addwf carrier, f
      btfss STATUS.C
      goto  end_carrier_function
      comf  carrier, f
      decf  carrier
      bsf   PWM_STATUS.0
```

```
   goto  end_carrier_function

negative_slope:
   subwf carrier, f
   btfsc STATUS.C
   goto  end_carrier_function
   comf  carrier, f
   incf  carrier
   bcf   PWM_STATUS.0
   btg   PWM_STATUS.2
end_carrier_function:

// Modulante Senoidal -----------
modulating_index_function:
   movlw 0x01
   btfsc PWM_STATUS.1
   goto  even_quadrant

odd_quadrant:
   addwf modulating_index, f
   btfss STATUS.C
   goto  end_modulating_function
   comf  modulating_index, f
   decf  modulating_index
   bsf   PWM_STATUS.1
   goto  end_modulating_function

even_quadrant:
   subwf modulating_index, f
   btfsc STATUS.C
   goto  end_modulating_function
   comf  modulating_index, f
   incf  modulating_index
   bcf   PWM_STATUS.1
   btg   PWM_STATUS.3
end_modulating_function:

// Comparação --------------------
test_functions_signals:
   btfsc PWM_STATUS.2
   goto  modulating_positive

modulating_negative:
   btfsc PWM_STATUS.3
   goto  set_next_pwm_low
   goto  memory_management_unit

modulating_positive:
   btfss PWM_STATUS.3
   goto  set_next_pwm_high

// MMU --------------------------
memory_management_unit:
   movff modulating_index, FSR1L
   movf  INDF1, w
```

```
ma_multiply:
  mulwf ma_low
  movff PRODH, result

test_ma_sum:
  tstfsz ma_high
  goto  ma_sum
  movf  result, w
  goto  end_sum

ma_sum:
  mulwf ma_high
  tstfsz PRODH
  goto  satured
  movf  PRODL, w
  addwf result, w
  btfsc STATUS.C

satured:
  movlw 0xFF
end_sum:

// Modulante > Portadora --------
positive_test:
  subwf carrier, w
  btfsc PWM_STATUS.3
  goto  negative_test
  btfss STATUS.C
  goto  set_next_pwm_high
  goto  set_next_pwm_low
negative_test:
  btfsc STATUS.C
  goto  set_next_pwm_high

// Ajustar próximo ciclo --------
set_next_pwm_low:
  bcf   PWM_STATUS.5
  goto  end_set_next_pwm
set_next_pwm_high:
  bsf   PWM_STATUS.5
end_set_next_pwm:

// Restaurar Contexto -----------
restore_context:
  bcf   PIR1.TMR2IF
  bsf   INTCON.GIE_GIEH

end_restore_context:
  return FAST

// Dead Time --------------------
dead_time_function:
  movff  dead_time, temp
toggle_dead_time:
```

```
      btg   PWM_STATUS.4

    // Loop - (1us) -----------------
    nop
    nop
    decrement_dead_time:
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        decfsz temp,f
        goto decrement_dead_time
    return NOT_FAST
  #endasm
}

// Inicialização -------------------------------------------------------------
void initialization (void)
{
  // Inicializar Periféricos ---------
  setup_adc_ports(NO_ANALOGS|VSS_VDD);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(SPI_SS_DISABLED);
  setup_wdt(WDT_OFF);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DIV_BY_4,48,1);
  setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);

  // Configurar Porta LCD ------------
  output_a (0x00);
  set_tris_a (0x00);

  // Configurar Porta Botões ---------
  output_b (0x00);
  set_tris_b (0x0F);

  // Configurar Porta PWM ------------
  output_d (0x00);
  set_tris_d (0x00);

  // Variáveis de Estado -------------
  pwm_status_pointer = 0x24;

  // Variáveis do PWM ----------------
  mf      = 0x09;
  ma_low   = 0x00;
  ma_high  = 0x01;
  ma_float = ma_high;
```

```
  // Contadores ----------------------
  carrier  = 0x00;
  modulating_index = 0x00;

  // Dead Time -----------------------
  dead_time = 0x02;

  // Ajustar Endereçamento Indireto --
  fsr1h_pointer = 0x01;

  // Configurar LCD ------------------
  LCD_init ();

  // Atribuir Prioridades ------------
  bit_clear(*INTCON2, TMR0IP);
  bit_set  (*IPR1, TMR2IP);
  bit_set  (*RCON, IPEN);

  // Exibir Messagem Inicial ---------
  show_time_out_message ();

  // Habilitar Interrupções ----------
  enable_interrupts(INT_TIMER2);
  enable_interrupts(INT_TIMER0);
  enable_interrupts(GLOBAL);
}

// Menu Time Out ------------------------------------------------------------
void menu_time_out (void)
{
  if (++time_out == 3)
  {
    show_time_out_message ();
    while (!keyboard ());
    time_out_semaphore = TRUE;
  }
}

// Reset Time Out -----------------------------------------------------------
void reset_time_out (void)
{
  *TMR0H = 0;
  *TMR0L = 0;
  time_out = 0;
  time_out_semaphore = FALSE;
}

// Mensagem do Time Out -----------------------------------------------------
void show_time_out_message (void)
{
  printf (LCD, "\f Time Out \nMessage");
}

// Rotina de Serviço Interface Homem-Máquina --------------------------------
```

```
void menu_service_routine (void)
{
  int8 key;
  submenu = 0x00;

  do
  {
    key = keyboard ();

    if (key == MENU_KEY)
      submenu_enter ();
    else if (key == UP_KEY)
      ++submenu;
    else if (key == DOWN_KEY)
      --submenu;

    if (key || time_out_semaphore)
    {
      reset_time_out ();
      test_submenu_limit ();
      show_submenu ();
    }
  }
  while (TRUE);
}

// Acessar Submenu ------------------------------------------------------------
void submenu_enter (void)
{
  if (submenu == MA_SUBMENU)
    adjust_ma ();
  else if (submenu == MF_SUBMENU)
    adjust_mf ();
  else if (submenu == DT_SUBMENU)
    adjust_dead_time ();
}

// Exibir Mensagem do Submenu -------------------------------------------------
void show_submenu (void)
{
  if (submenu == MA_SUBMENU)
    printf (LCD, "\f MA Submenu");
  else if (submenu == MF_SUBMENU)
    printf (LCD, "\f MF Submenu");
  else if (submenu == DT_SUBMENU)
    printf (LCD, "\f DT Submenu");
}

// Ajustar Índice de Modulação em Amplitude -----------------------------------
void adjust_ma (void)
{
  int8 key;

  printf (LCD, "\f Adjust MA:%3f", ma_float);
```

```
  do
  {
    key = keyboard ();

    if (key == UP_KEY)
    {
      ++ma_low;
      ma_float = ma_float + (float)(0.00390625);
    }
    else if (key == DOWN_KEY)
    {
      --ma_low;
      ma_float = ma_float - (float)(0.00390625);

      if (ma_float < 0x00)
      {
        ma_float = 0;
        ma_low = 0;
      }
    }
    ma_high = ma_float;

    if (key || time_out_semaphore)
    {
      reset_time_out ();
      printf (LCD, "\fAdjust MA:%3f", ma_float);
    }
  }
  while (key != MENU_KEY);
}

// Ajustar Índice de Modulação em Frequência ---------------------------------
void adjust_mf (void)
{
  int8 key;

  printf (LCD, "\fAdjust MF:%3d", mf);

  do
  {
    key = keyboard ();

    if (key == UP_KEY)
      ++mf;
    else if (key == DOWN_KEY)
      if (!(--mf))
        mf = 1;

    if (key || time_out_semaphore)
    {
      reset_time_out ();
      printf (LCD, "\fAdjust MF:%3d", mf);
    }
  }
  while (key != MENU_KEY);
```

```
}

// Ajustar Dead Time ------------------------------------------------------------
void adjust_dead_time (void)
{
  int8 key;

  printf (LCD, "\fAdjust DT:%3d", dead_time);

  do
  {
    key = keyboard ();

    if (key == UP_KEY)
      ++dead_time;
    else if (key == DOWN_KEY)
      if (!(--dead_time))
        dead_time = 1;

    if (key || time_out_semaphore)
    {
      reset_time_out ();
      printf (LCD, "\fAdjust DT:%3d", dead_time);
    }
  }
  while (key != MENU_KEY);
}

// Ajustar Faixa ----------------------------------------------------------------
void test_submenu_limit (void)
{
  if (submenu == SUBMENU_ITENS + 1)
    --submenu;
  else if (submenu == 0x00)
    ++submenu;
}

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////// END /////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```