Unifei - Universidade Federal de Itajubá

Grupo de Estudos em Manutenção Eletro-Eletrônica e Instalações

UNIFEI / ISEE / GEMEI

IMPLEMENTAÇÃO DE UM SISTEMA DE PARTIDA E CONTROLE DE VELOCIDADE PARA MÁQUINA SÍNCRONA.

Dissertação de Mestrado

Autor: Eben-Ezer Prates da Silveira

Orientador: Ângelo José Junqueira Rezek, Dr.

Co-orientador: Valberto Ferreira da Silva, Dr.

Itajubá 2004

"Ensina-nos a contar os nossos dias,
de tal maneira que alcancemos
corações sábios"
(Salmo 90:12).

“O temor do Senhor é o princípio de toda sabedoria,
e o conhecimento do Santo é entendimento.”
(Provérbios 9:10)

Agradecimentos

A Deus pelo dom da vida e pelas bênçãos concedidas, muito mais do que sou merecedor.

Aos meus pais, João Batista da Silveira e Dennilza Prates da Silveira, por acreditar em mim e me incentivar sempre. Assim como também, minha irmã Helencler e meu irmão Júnior.

A minha namorada Fabíola Rosa Faria, pela compreensão, apoio e carinho que foram para mim uma fonte de motivação.

Ao professor Ângelo José Junqueira Rezek por sua amizade, seu empenho, brilhante na orientação deste trabalho.

Ao professor Valberto Ferreira da Silva, co-orientador deste trabalho, pelo auxílio e apoio técnico, fundamentais no desenvolvimento desta dissertação de mestrado.

Ao aluno Fabio Camillo Mascarim pela importante contribuição na montagem da fase inicial deste projeto.

A servidora do Laboratório Instituto de Sistemas Elétricos de Energia, Ana Maria Tavares, pela atenciosa e prestativa ajuda, bem como aos servidores Edmundo F. da Silva, J. C. Anselmo e Raimundo R. da Silva.

A FAPEMIG pelo apoio financeiro que possibilitou a realização deste trabalho.

Um Agradecimento especial ao professor Antônio Tadeu Lyrio de Almeida, por sua amizade e incentivo, que foram de grande peso, na decisão de iniciar esta jornada.

Resumo

O propósito deste trabalho é apresentar um sistema de partida e controle de velocidade para uma máquina síncrona. O mesmo utiliza duas pontes conversoras de seis tiristores, uma operando como retificadora e outra como inversora, com comutação natural, em regime permanente.

Um controle de malha fechada foi utilizado para permitir a regulação da velocidade do motor e também para limitar o valor da corrente, para a proteção do motor.

Em baixas velocidades o motor não possui tensão suficiente em seus terminais para permitir a comutação natural da ponte inversora. Para contornar esta situação foram desenvolvidos dois circuitos que realizam uma comutação forçada, o que possibilita o funcionamento da máquina em baixas rotações, até 150 [RPM]. O primeiro foi implementado utilizando componentes analógicos e digitais, e o segundo baseado em um microcomputador do tipo PC.

O circuito implementado utilizando microcomputador, para o controle da ponte inversora, permite também que a partida do motor seja realizada de forma automática sem a necessidade de uma máquina auxiliar.

O programa criado para o controle de disparo da ponte inversora foi desenvolvido em linguagem C, para ser executado no sistema operacional MS-DOS. O mesmo possuí uma interface gráfica, permitindo uma melhor interação do usuário com o sistema. O programa permite ao usuário partir e parar a máquina, visualizar o valor da velocidade do motor (estimada pela tensão nos terminais do mesmo), monitorar graficamente a variação da velocidade e a condição de operação da ponte, que pode ser, modo de comutação forçada ou natural.

Os resultados obtidos durante os ensaios em laboratórios demonstram que o desempenho geral do sistema foi satisfatório, correspondendo à expectativa.

Palavras chaves: Máquinas síncronas, reguladores PID, inversor de corrente, métodos eletrônicos de partida para máquinas síncronas.

Abstract

The aim of this work is to show a start-up and control system to synchronous machines. Using two six thyristors converter bridges, one is operating as rectifier and the other one as inverter with natural commutation, in steady-state.

One closed-loop control was used to motor speed regulation and also to limit the current value, for the motor protection.

When the motor is working in low speed, there is no voltage enough in motor terminals to allow the natural commutation of the inverter bridge. To work around this situation was developed two circuits to execute a forced commutation, what makes the motor be able to work in low speed, until 150 [RPM]. The first one was implemented using analog e digital components and the second one using a pc-based microcomputer.

The microcomputer based circuit to the inverter bridge allows also an automatic motor start-up. So, there is no necessity of use an auxiliary machine.

The software created to control the firing pulses of the inverter bridge was developed in C language to be executed in MS-DOS operational system. This one has a graphical user interface, which allows a better interaction between the user and the system. The software allows to the user to start and stop de machine, to visualize the motor speed value (estimated by the motor terminals voltage), to monitor graphically the speed variation and operation mode of the bridge. That can be, forced commutated or natural commutated.

The obtained results from the essays in laboratory showed that the overall performance of the system was satisfactory, according to the expectation.

Keywords: Synchronous Machines, PID regulators, Current Source Inverter, Synchronous Machine Electronic Starting Method.

Índice das Figuras

Figura 2.1 - Acionamento completo 11
Figura 2.2 - Ponte conversora CA-CC de seis pulsos. 12
Figura 2.3 - Formas de onda das correntes nos tiristores e na saída da ponte retificadora para α=30° 13
Figura 2.4 – Formas de onda das correntes nas fases da ponte retificadora para α=30° 14
Figura 2.5 - Formas de onda das tensões fase-neutro da ponte retificadora para α=30° 14
Figura 2.6 – Formas de onda das tensões fase-fase e tensão entre anodo e catodo do tiristor 1 da ponte retificadora para α=30° 15
Figura 2.7 – Circuito da Ponte Retificadora e da Ponte Inversora. 15
Figura 2.8 – Formas de onda das correntes nos tiristores e na saída da ponte retificadora para α=150° 16
Figura 2.9 - Formas de onda das correntes nas fases da ponte retificadora para α=150° 16
Figura 2.10 – Formas de onda das tensões fase-neutro da ponte retificadora para α=150°. 17
Figura 2.11 – Formas de onda das tensões fase-fase e tensão entre anodo e catodo do tiristor 1 da ponte retificadora para α=150°. 17
Figura 3.1 – Geração de rampa do TCA-780 20
Figura 3.2 – Geração de rampa do TCA-780 20
Figura 3.3 – Diagrama das conexões para 180° 21
Figura 3.4 – Circuito de disparo da ponte inversora utilizando o TCA-780. 22
Figura 3.5 – Circuito amplificador de pulsos. 23
Figura 3.6 – Transformadores de Alimentação dos Pulsos de Disparo 24
Figura 3.7 – Circuito somador 25
Figura 3.8 – Circuito gerador de pulsos para comutação forçada 26
Figura 3.9 – Diagrama de blocos completo 26
Figura 4.1: Representação da parte mecânica e do circuito elétrico da armadura do motor síncrono [2] 28
Figura 4.2 - Diagrama de blocos da parte mecânica da máquina síncrona [2] 30
Figura 4.3 - Diagrama de blocos equivalente do circuito da armadura da máquina síncrona [2] 31
Figura 4.4 - Diagrama de blocos completo da máquina [2] 33
Figura 4.5 - Malha de regulação de corrente [2] 35
Figura 4.6 - Diagrama de blocos da malha de regulação de velocidade [2] 36
Figura 4.7 - Reguladores e filtros 37
Figura 5.1 – Circuito divisor de tensão. 39
Figura 5.2 – Formas de onda na entrada e na saída do circuito divisor de tensão 39
Figura 5.3 – Condição de Disparo do Tiristor 1 40
Figura 5.4 – Condição de Disparo do Tiristor 2 40
Figura 5.5 – Condição de Disparo do Tiristor 3 41
Figura 5.6 – Condição de Disparo do Tiristor 4 41
Figura 5.7 – Condição de Disparo do Tiristor 5 41
Figura 5.8 – Condição de Disparo do Tiristor 6 41
Figura 5.9 – Tensão na fase A e Pulso de disparo no Tiristor 1. 41
Figura 5.10 – Alinhamento do rotor para Campo gerado por Ia e Ib. 42

Figura 5.11 – Alinhamento do rotor para Campo gerado por Ib e Ic. 43
Figura 5.12 – Alinhamento do rotor para Campo gerado por Ia e Ic. 43
Figura 5.13 – Diagrama de blocos dos sistema utilizando microprocessador. 44
Figura 5.14 – Software Controle da Ponte Inversora (Rotina Principal)..... 45
Figura 5.15 – Rotina de Interrupção. 45
Figura 5.16 – Tela do software 47
Figura 6.1 – Partida do Motor Síncrono. 49
Figura 6.2 – Corrente no Link DC. 49
Figura 6.3 – Corrente na fase A do MS. 50
Figura 6.4 – Tensão na fase A do MS. 50
Figura 6.5 – Aplicação à carga. 51
Figura 6.6 – Retirada da carga. 51
Figura 6.7 - Tensão na ponte retificadora e inversora - 1000[RPM]. 52
Figura 6.8 - Tensão na ponte retificadora e inversora - 576[RPM]. 52
Figura 6.9 - Tensão na ponte retificadora e inversora - 340[RPM]. 53
Figura A1.1 - Pinagem do TCA-780 (Vista Superior). 55
Figura A3.1 – 1) Motor síncrono; 2) Motor C.C. 92
Figura A3.2 – 3) Reostato de campo do motor síncrono; 4) Reostato de campo do MCC. . 92
Figura A3.3 – 5) Voltímetro utilizado para medir a tensão na armadura do MCC. 92
Figura A3.4 – 6) Amperímetro utilizado para medir a corrente de excitação do MS. 92
Figura A3.5 – 7) Fonte de alimentação auxiliar para o circuito de disparo do inversor. 92
Figura A3.6 - 8) LED para monitoração dos pulsos de disparo. 92
Figura A3.7 – 9) 1º Circuito de disparo do Inversor. 93
Figura A3.8 – 10) Driver amplificador de pulsos. 93
Figura A3.9 – 11) Gerador de pulsos para comutação forçada, parte B. 93
Figura A3.10 - 3) Circuito detector de zero. 93
Figura A3.11 – 14) Transformador utilizado para o circuito gerador de pulsos para comutação forçada. 93
Figura A3.12 – 15) Tiristor volante. 93
Figura A3.13 – 16) Módulo do regulador de velocidade 94
Figura A3.14 – 17) Varivolt utilizado para ajustar tensão de entrada do retificador. 94
Figura A3.15 – 18) Ponte inversora. 94
Figura A3.16 – 19) Fonte de alimentação do circuito do inversor. 94
Figura A3.17 – 20) Osciloscópio utilizado para monitorar os pulsos para a ponte inversora. 94
Figura A3.18 – 21) Microcomputador utilizado; 22) circuito de disparo do inversor; 23) Elementos para monitoração e ligação do link DC. 94
Figura A3.19 – 24) Osciloscópio utilizado para monitorar a ponte inversora. 95
Figura A3.20 – 25) Bornes da placa de aquisição de dados. 95
Figura A3.21 – 26) Placa de interface entre a porta paralela e o buffer do circuito de disparo do inversor. 95
Figura A3.22 – 27) Fonte de alimentação do amplificador de pulsos. 95
Figura A3.23 – 28) Circuito somador utilizado para comutação forçada; 29) Gerador de pulsos para comutação forçada, parte A. 95
Figura A3.24 – 31) Chave do Link DC. 95

Índice das Tabelas

Tabela 3.1 – Defasamento angular para os TAPs do transformador. 21
Tabela 4.1 - Dados nominais do motor síncrono utilizado [2] 34
Tabela A1.1 - Larguras dos pulsos dos pinos 14 e 15 para valores determinados de C12... 58

Sumário

Capítulo 1 - Introdução 10
Capítulo 2 – Acionamento da Máquina Síncrona 11
2.1 - Introdução 11
2.2 - Ponte Conversora CA-CC e Ponte Inversora 12
2.3 - Circuito de disparo da Ponte Retificadora 18
Capítulo 3 – Controle de Disparo da Ponte Inversora 19
3.1 – Introdução 19
3.2 – Circuito de disparo para a ponte inversora 19
3.2 – Operação em Baixas Velocidades 25
Capítulo 4 - Malha de Controle do Sistema 28
4.1 - Introdução 28
4.2 - Equacionamento e Diagrama de Blocos do Motor 28
4.3 – Estratégia de Controle[2] 33
4.3.1 – Escolha e Ajuste dos Reguladores [2] 34
4.3.2 – Diagrama do Circuito do Regulador 37
Capítulo 5 – Controle de Disparo da Ponte Inversora – Por Microcomputador 38
5.1 - Introdução 38
5.2 - Hardware 38
5.3 - Software 40
5.3.1 - O Algoritmo Para Geração dos Pulsos de Disparo 40
5.3.2 - Partida 42
5.3.3 – Interface do Usuário 47
Capítulo 6 – Resultados Experimentais 49
Capítulo 7 – Conclusão 54
Anexo 1 – Circuitos Eletrônicos 55
A1.1 - TCA-780 55
A1.2 - C.I. 555 59
A1.3 - TIL 113 60
A1.4 - Drive 61
Anexo 2 - Código Fonte do Programa 63
A2.1 – Rotina Principal – main.c 63
A2.2 – Cabeçalho – public.h 80
A2.3 –Driver para a Placa de Aquisição de Dados – drive.hx 83
A2.4 – Rotinas para Controle da Interrrupção – timer.hx 85
A2.5 – Rotinas para Detecção da Seqüência de Pulsos – zeroc.hx 89
Anexo 3 – Fotos dos Elementos do Sistema 92
Bibliografia 96

Capítulo 1 - Introdução

A máquina de corrente contínua possui a vantagem da facilidade de controle da velocidade, a qual pode ser controlada pela variação da tensão da armadura ou da corrente de campo, ou ainda ambos. No entanto, são máquinas caras e de custo de manutenção relativamente elevado. Além disto, a presença do comutador impossibilita a utilização de motores de corrente contínua em ambientes com risco de explosão, ou com grande quantidade de poeira ou outras partículas em suspensão.

A evolução da eletrônica de potência e das técnicas de controle tem permitido uma maior utilização de motores síncronos em substituição a acionamentos que utilizam máquinas de corrente contínua. Além disto o motor síncrono possuí um custo menor e uma manutenção mais simples e de custo inferior ao da máquina de corrente contínua.

O controle da máquina síncrona por injeção de corrente apresenta um bom desempenho como sistema de velocidade controlável. Além disto, as características deste tipo de acionamento permitem que este possa ser considerado como um motor de corrente contínua sem comutador.

O acionamento da máquina síncrona, proposto neste trabalho, opera em mono-quadrante e para uma velocidade variável entre 150 a 1200 [RPM].

Foram utilizados circuitos analógicos para o regulador de velocidade e controle do disparo da ponte retificadora. Foi, também, implementado um circuito que utiliza componentes digitais e analógicos para o circuito de disparo da ponte inversora, utilizando como base o circuito integrado TCA 780, desenvolvido pela ICOTRON / SIEMENS. Bem como circuito para realizar a comutação forçada para baixas velocidades.

A redução no custo e a evolução dos microprocessadores fazem com que estes se apresentem como uma interessante opção para implementação de sistemas de controle. Neste trabalho foram implementados um sistema de partida e uma rotina para possibilitar o funcionamento do motor em baixas rotações utilizando um microcomputador.

O sistema implementado, além de apresentar uma maior flexibilidade, possibilitou a o cálculo da velocidade sem a necessidade do uso do taco gerador utilizado no sistema para fornecer o sinal de referência do circuito regulador de velocidade.

Capítulo 2 – Acionamento da Máquina Síncrona

2.1 - Introdução

A máquina síncrona apresenta algumas características que fazem com seja vantajoso sua utilização como motor, tais como, possibilidade de correção do fator de potência e custo reduzido em relação ao motor de corrente contínua.

A velocidade da máquina síncrona é determinada pela freqüência da sua tensão de alimentação. Portanto, sua utilização em acionamentos que exigem velocidade variável torna-se uma tarefa complexa. O modelo proposto é uma solução para se utilizar à máquina síncrona em um acionamento de velocidade variável. Este modelo permite, não só executar o acionamento da máquina, mas também simplificar o seu controle, o que o torna similar ao de uma máquina de corrente contínua.

O acionamento é composto, principalmente, por uma ponte retificadora e uma ponte inversora de seis pulsos utilizando tiristores, as quais são conectadas pelo link DC. Existindo, ainda, um indutor (L) entre as pontes no lado de corrente contínua. O circuito completo é mostrado na figura 2.1. Na mesma figura estão representados os outros componentes do sistema, que são: o regulador de velocidade (o diagrama de blocos), o motor síncrono, o tacogerador, o gerador de corrente contínua e o banco de resistores. Estes dois últimos serão utilizados para simular uma carga fixa ao eixo do motor síncrono.

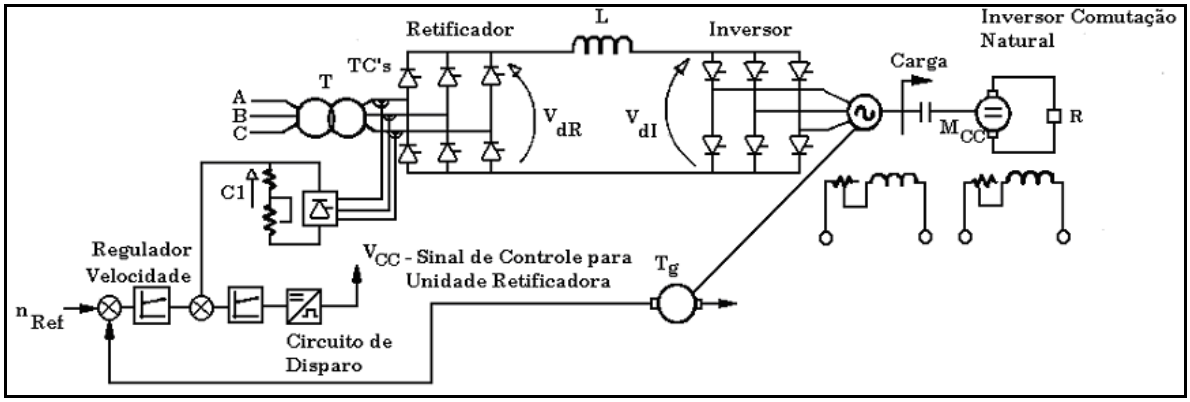

Figura 2.1 - Acionamento completo

A velocidade da máquina síncrona e a corrente no link DC são ajustados por um controle de malha fechada. Este define o ângulo de disparo dos tiristores da ponte

retificadora, determinando o valor da tensão no link DC. A comutação ocorre naturalmente pela rede. A ponte inversora, em regime permanente, opera com ângulo fixo, sendo que a comutação é possível devido à tensão do motor, por isso dito que é comutada pela carga.

O circuito do regulador de velocidade e o circuito de disparo da ponte retificadora, implementados neste trabalho, são inteiramente analógicos. O circuito de disparo da ponte inversora utiliza sensor de tensão para gerar os pulsos de gatilho.

Neste capítulo, são mostradas as formas de ondas para as pontes retificadora e inversora e também a descrição do circuito de disparo da ponte retificadora. Os circuitos implementados para a ponte inversora serão mostrados nos capítulos seguintes.

2.2 - Ponte Conversora CA-CC e Ponte Inversora

O objetivo da ponte trifásica tiristorizada é converter uma potência C.A. em uma potência C.C, ou vice-versa, pois cada ponte pode operar nos dois modos, dependendo do ângulo de disparo dos tiristores.

Para um ângulo de disparo α entre 0º e 90º a ponte opera como circuito retificador, convertendo a tensão C.A. em tensão C.C. O sinal na saída da ponte (terminais CC) depende do instante em que os pulsos de disparado são aplicados nos tiristores. Controlando o ângulo de disparo pode-se determinar qual segmento do sinal de entrada será entregue à saída, controlando assim o valor médio da tensão C.C.

A figura 2.2, mostra o circuito de ligação da ponte retificadora comutada pela rede.

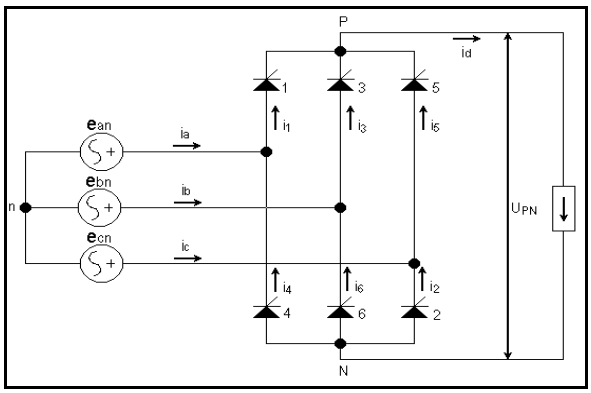

Figura 2.2 - Ponte conversora CA-CC de seis pulsos.

Na figura 2.2, os tiristores estão numerados conforme a seqüência em que entram em condução.

Grandezas da ponte retificadora:

e_{an}, e_{bn}, e_{cn}: Tensão de fase neutro de alimentação da ponte;

i_a, i_b, i_c: Corrente nas fases a, b e c de alimentação da ponte;

i_1, i_2, i_3, i_4, i_5, i_6: Correntes nos tiristores;

i_d: Corrente contínua na saída da ponte;

U_{PN}: Tensão entre o pólo positivo e o pólo negativo, na saída ponte;

U_{Pn}: Tensão entre o pólo positivo e o neutro;

U_{Nn}: Tensão entre o pólo negativo e o neutro;

U_{AC1}: Tensão anodo-catodo do tiristor 1.

As formas de onda típicas para a ponte operando com um ângulo de disparo de 30º, podem ser visualizadas nas figuras 2.3, 2.4, 2.5 e 2.6.

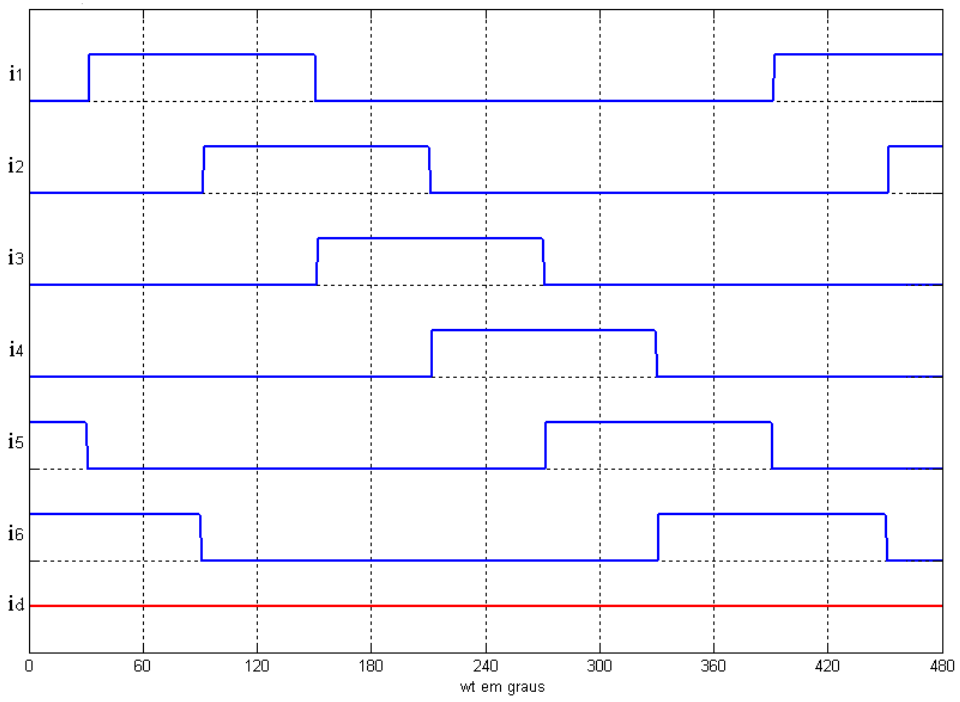

Figura 2.3 - Formas de onda das correntes nos tiristores e na saída da ponte retificadora para α=30°

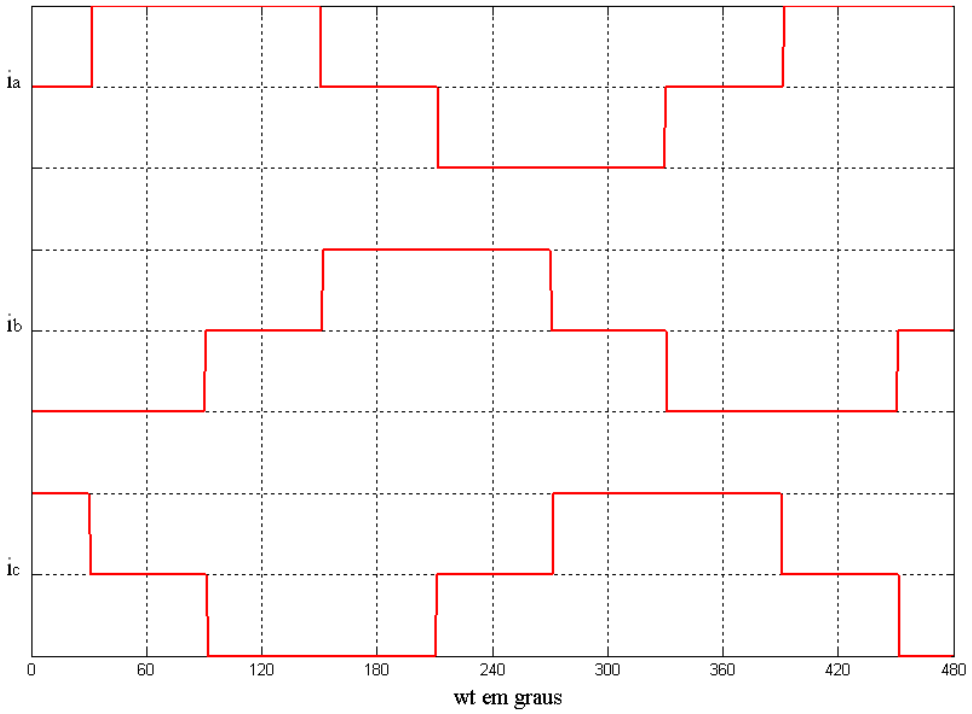

Figura 2.4 – Formas de onda das correntes nas fases da ponte retificadora para α=30°

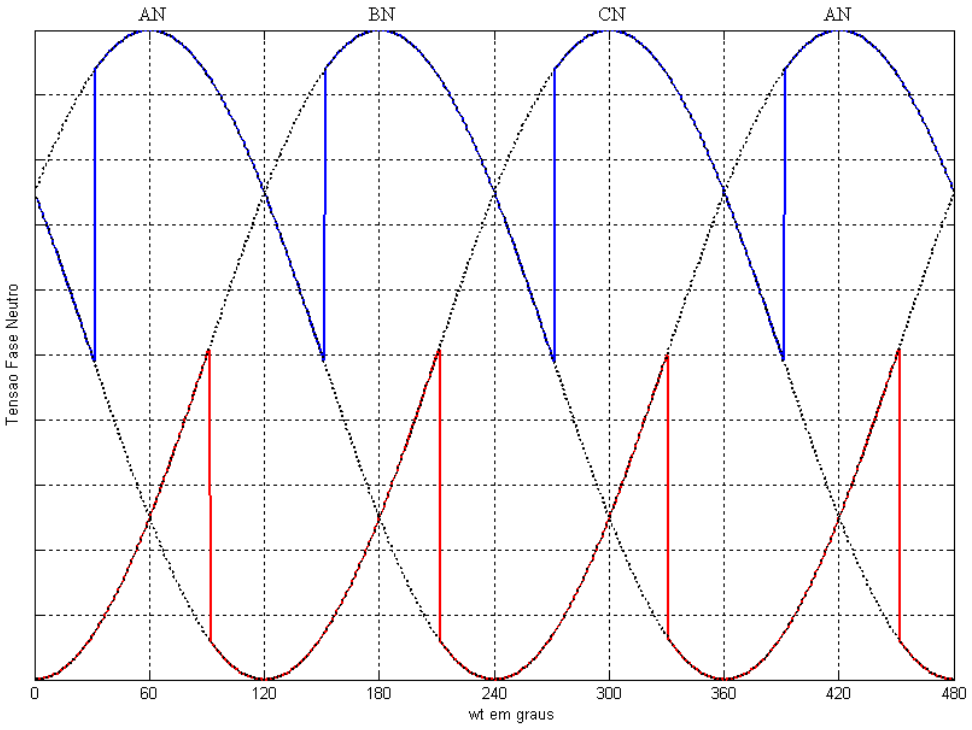

Figura 2.5 - Formas de onda das tensões fase-neutro da ponte retificadora para α=30°

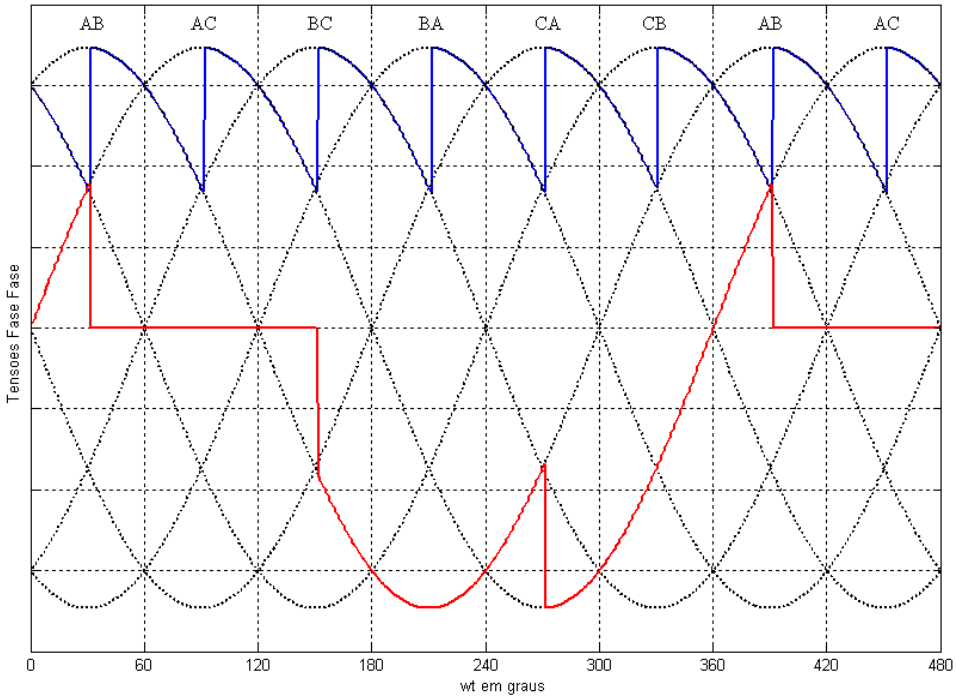

Figura 2.6 – Formas de onda das tensões fase-fase e tensão entre anodo e catodo do tiristor 1 da ponte retificadora para α=30°

Para um ângulo de disparo α entre 90º e 180º, não considerando o ângulo de comutação, a ponte opera como inversora, convertendo a tensão C.A. em tensão em C.C. A figura 2.7, mostra o circuito de ligação da ponte retificadora e inversora comutada pela carga.

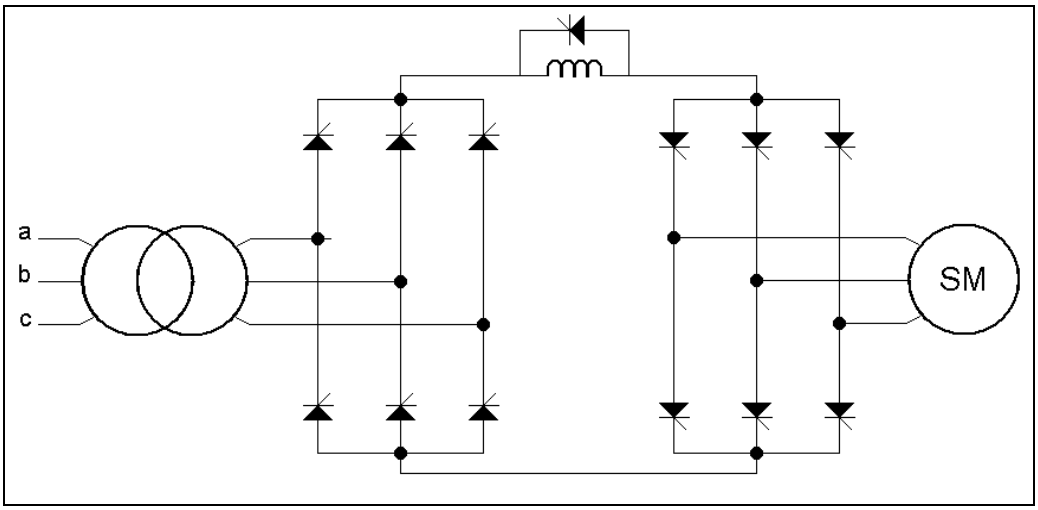

Figura 2.7 – Circuito da Ponte Retificadora e da Ponte Inversora.

As formas de onda típicas para a ponte operando com um ângulo de disparo de 150º, podem ser visualizadas nas figuras 2.8, 2.9, 2.10 e 2.11.

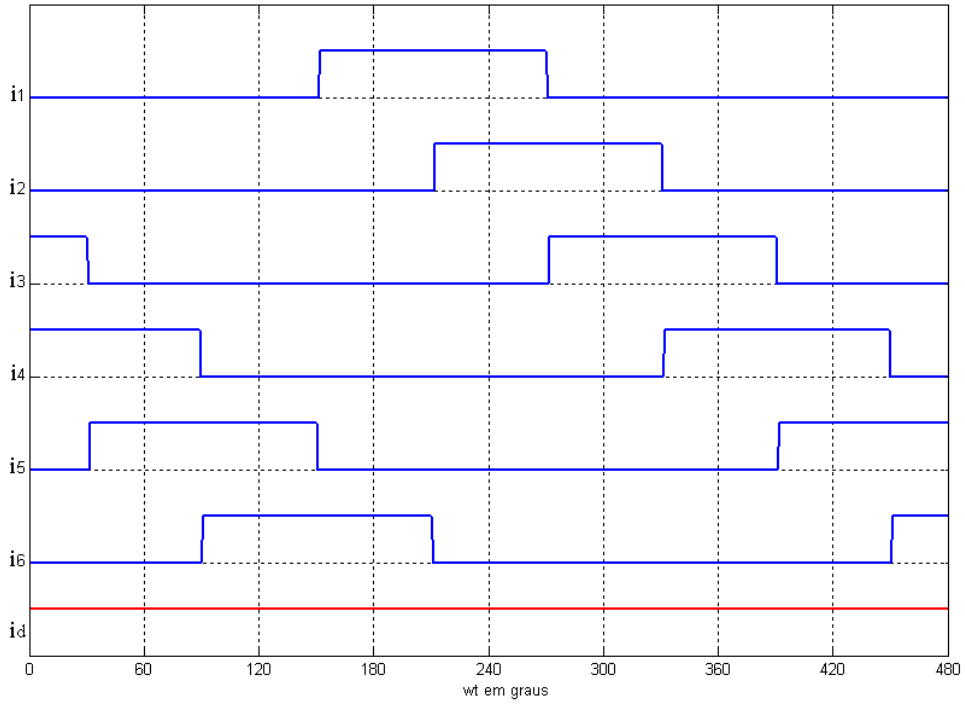

Figura 2.8 – Formas de onda das correntes nos tiristores e na saída da ponte retificadora para α=150°

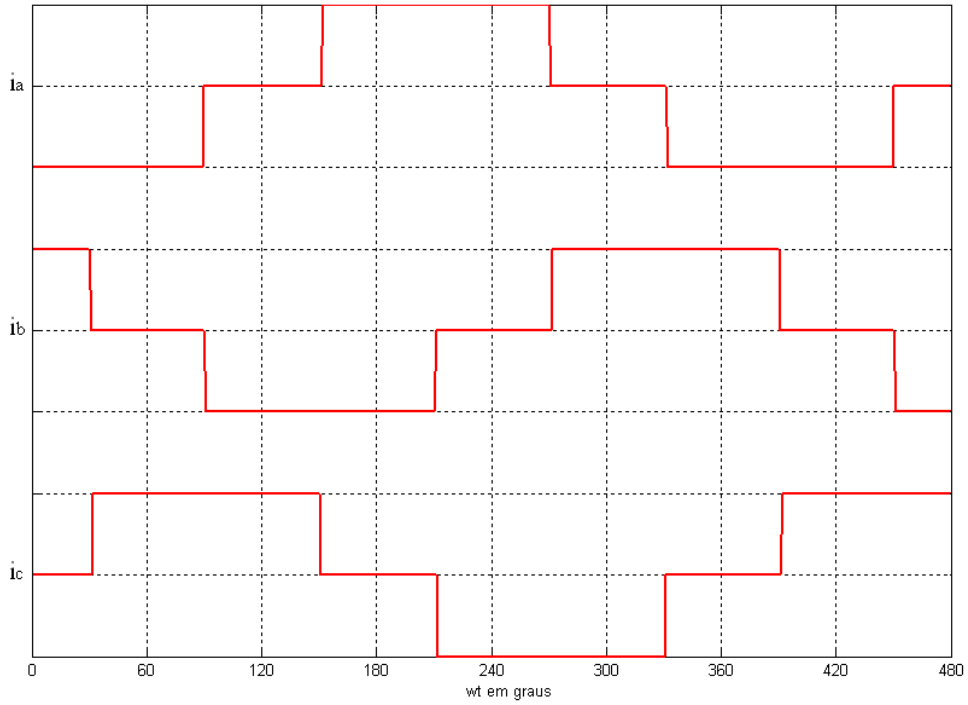

Figura 2.9 - Formas de onda das correntes nas fases da ponte retificadora para α=150°

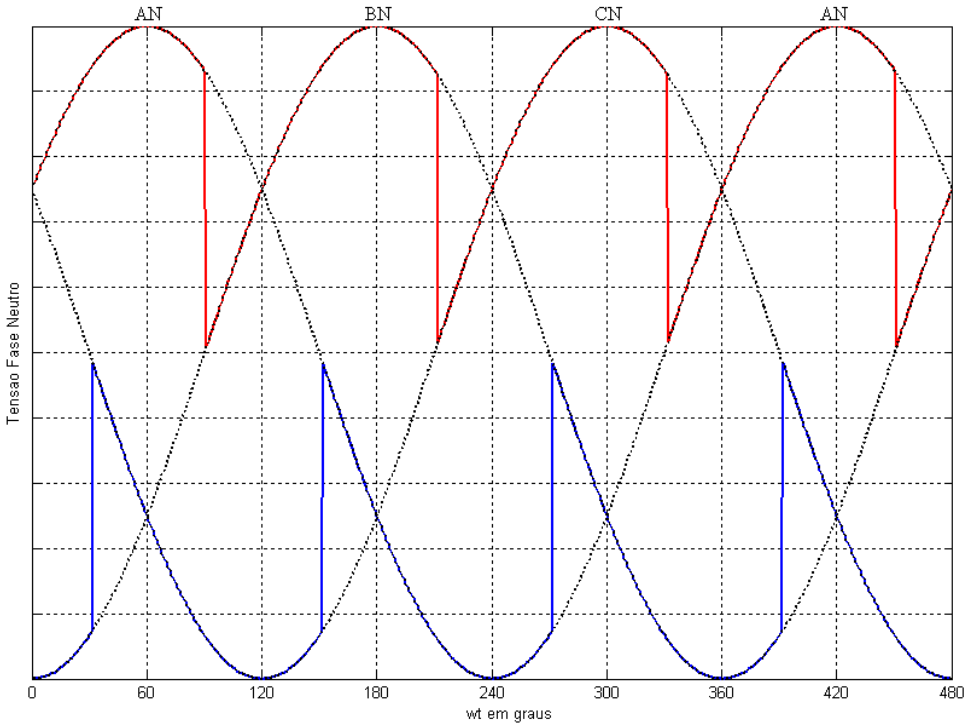

Figura 2.10 – Formas de onda das tensões fase-neutro da ponte retificadora para α=150°

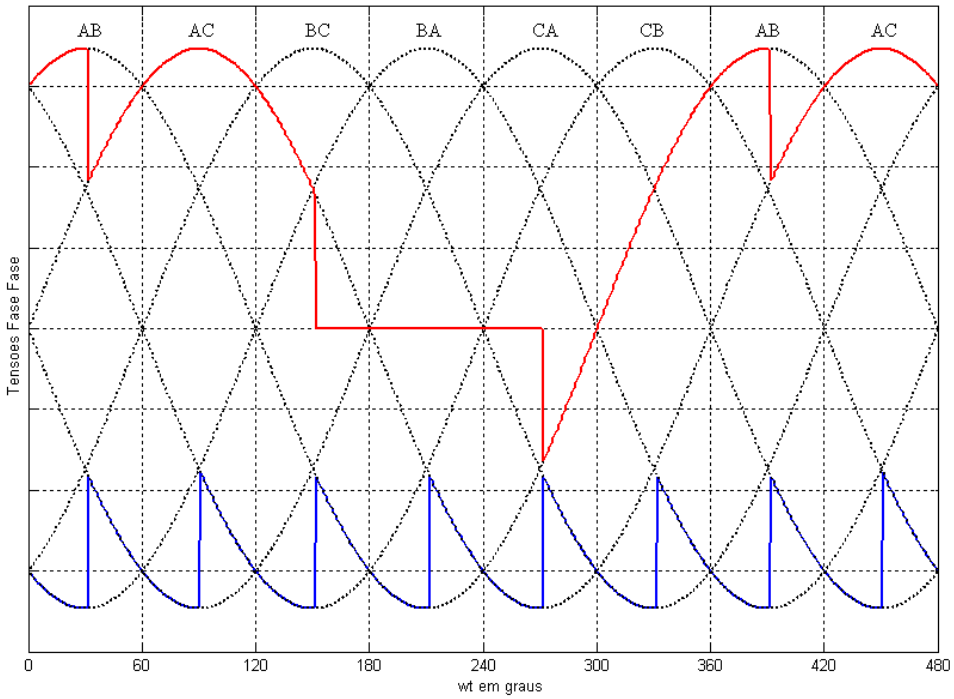

Figura 2.11 – Formas de onda das tensões fase-fase e tensão entre anodo e catodo do tiristor 1 da ponte retificadora para α=150°.

2.3 - Circuito de disparo da Ponte Retificadora

Este circuito de disparo é composto de quatro etapas:

1. Geração do pulso de disparo;
2. Ajuste da largura do pulso de disparo;
3. Isolamento elétrico das partes de baixa e alta potência;
4. Driver de corrente.

A primeira etapa é formada por três circuitos integrados TCA-780 (ou seu substituto TCA-785). Estes recebem os sinais de tensão das fases a, b e c e a tensão de referência, a qual é gerada pelo regulador de velocidade, e geram os pulsos para os tiristores 1, 2, 3, 4, 5 e 6. O funcionamento do integrado bem como suas formas de onda características, estão detalhados no Anexo 1.1.

A segunda etapa destina-se a ajustar a largura dos pulsos, os quais foram ajustados para 120°. Isto é realizado utilizando-se o circuito integrado 555, para o qual o funcionamento é descrito no Anexo 1.2.

A terceira etapa é importante, pois é necessário realizar o isolamento elétrico entre as etapas de baixa e alta potência, o que impede uma possível corrente de curto entre estes estágios. Para este isolamento foi utilizado o opto-acoplador TIL 113, cuja descrição pode ser vista no Anexo 1.3.

A última parte é responsável por amplificar a corrente do pulso de saída do TIL113, até o valor necessário para disparar o gate do tiristor. O detalhamento do circuito pode ser visto no Anexo 1.4.

Capítulo 3 – Controle de Disparo da Ponte Inversora

3.1 – Introdução

Para se otimizar o valor do torque para uma dada corrente injetada, o motor síncrono funciona sobreexcitado, o que faz com que o mesmo possa ser dito como alimentado por corrente. A relação entre o torque e a corrente pode ser aproximada pela equação 3.1.

$$M = K \cdot \phi \cdot I_{DC} . \cos\alpha \quad (3.1)$$

Onde:

M = Torque desenvolvido pelo motor

K = Constante de proporcionalidade

ϕ = Fluxo do motor síncrono

I_{DC} = Corrente no link DC

α = Ângulo de disparo do inversor

Desta forma, quanto mais próximo o valor do cos α for da unidade, melhor será o aproveitamento da corrente para produção do torque. Para que isto ocorra, a ponte inversora opera com um ângulo fixo de 150°. Este valor foi definido para que se possa obter uma margem de comutação de 30°.

Para o disparo do inversor foi utilizada a técnica do sensor de tensão. Como este circuito difere do que foi utilizado para o disparo do retificador, foram implementados dois novos circuitos de disparo.

3.2 – Circuito de disparo para a ponte inversora

O circuito de disparo, utilizando o C.I. TCA-780, como no caso do retificador, gera os pulsos de disparo através da interseção entre uma tensão de referencia Vcc e uma rampa gerada pelo integrado, conforme está ilustrado na figura 3.1.

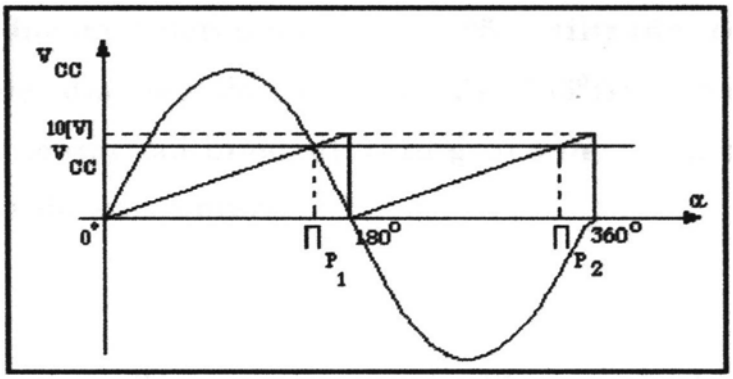

Figura 3.1 – Geração de rampa do TCA-780

Para o funcionamento deste circuito o valor de Vcc deve ser ajustado para corresponder ao ângulo de disparo desejado. No caso da figura 3.1, este foi ajustado para velocidade nominal da máquina. Mas como a velocidade do motor varia, este valor deve ser re-ajustado para cada novo valor de velocidade ou o ângulo de disparo não será constante, como ilustra a figura 3.2.

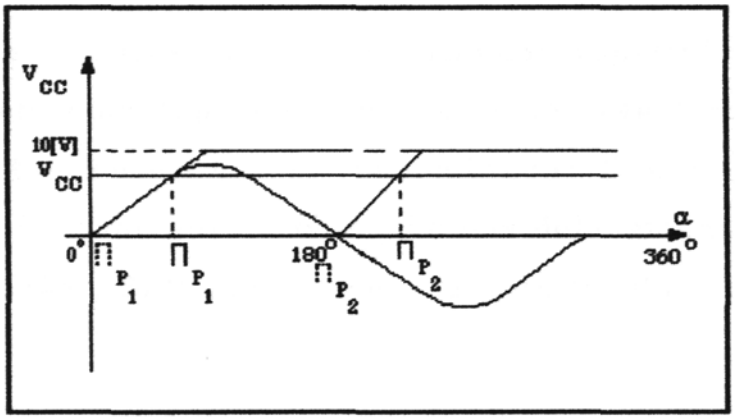

Figura 3.2 – Geração de rampa do TCA-780

Na figura 3.2, a velocidade do motor é a metade da nominal. Pode-se notar o deslocamento do ângulo de disparo em relação à figura 3.1.

Para manter o ângulo de disparo constante, independente da velocidade da máquina, o circuito foi modificado. Nesta nova configuração a tensão Vcc foi deslocada para zero, assim o TCA-780 passou a ser utilizado como detector de zero. Para obter o ângulo de disparo desejado (150°), a solução empregada foi utilizar um transformador de sincronismo

especial delta / zigue-zague, com um defasamento angular de 180°. Desta forma descontando os 30° de referência, obtém-se os 150° desejados.

O transformador utilizado permite através de conexões de seus TAPs os seguintes defasamentos: 165°, 180°, 195° e 210°. A tabela 3.1 mostra a relação entre os ângulos e as conexões. O novo circuito implementado pode ser visto na figura 3.3.

Defasamento Angular	TAP 1º Enrolamento	TAP 2º Enrolamento
165°	141,42% N	51,76% N
180°	100% N	100% N
195°	51,76% N	141,42% N
210°	0% N	173,21% N

Tabela 3.1 – Defasamento angular para os TAPs do transformador.

Sendo N o número de espiras. Este número também define a relação de transformação. O diagrama das conexões para 180°, pode ser visto na figura 3.3.

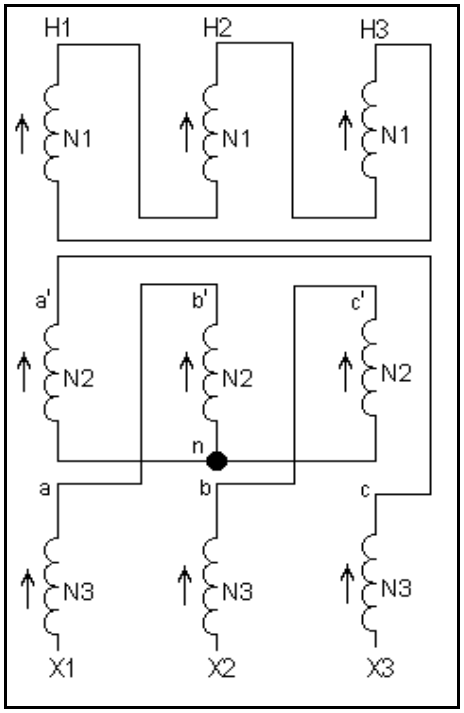

Figura 3.3 – Diagrama das conexões para 180°

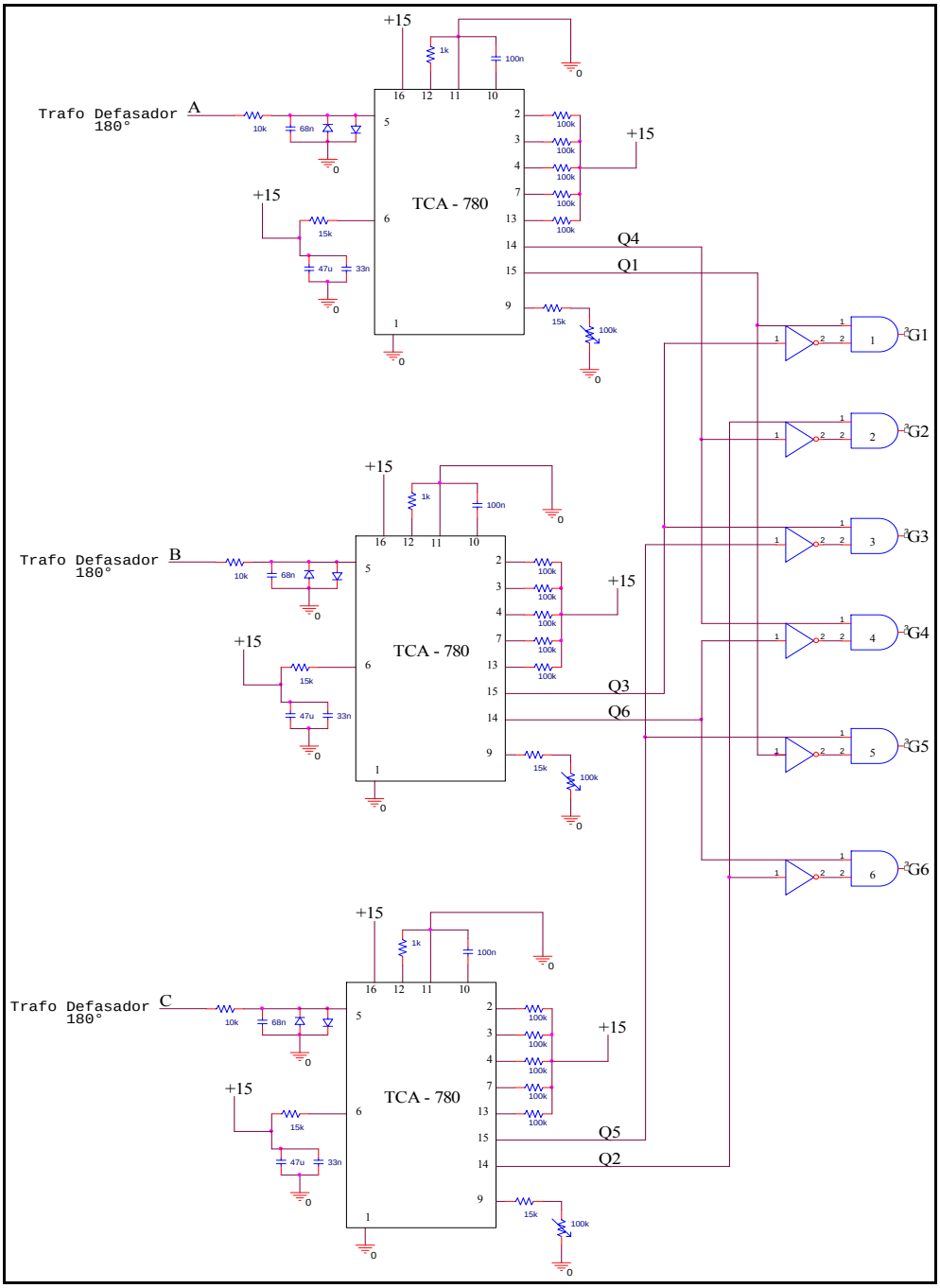

Figura 3.4 – Circuito de disparo da ponte inversora utilizando o TCA-780.

A saída deste circuito é conectada diretamente ao novo circuito de amplificação de pulsos (Driver), ilustrado na figura 3.4.

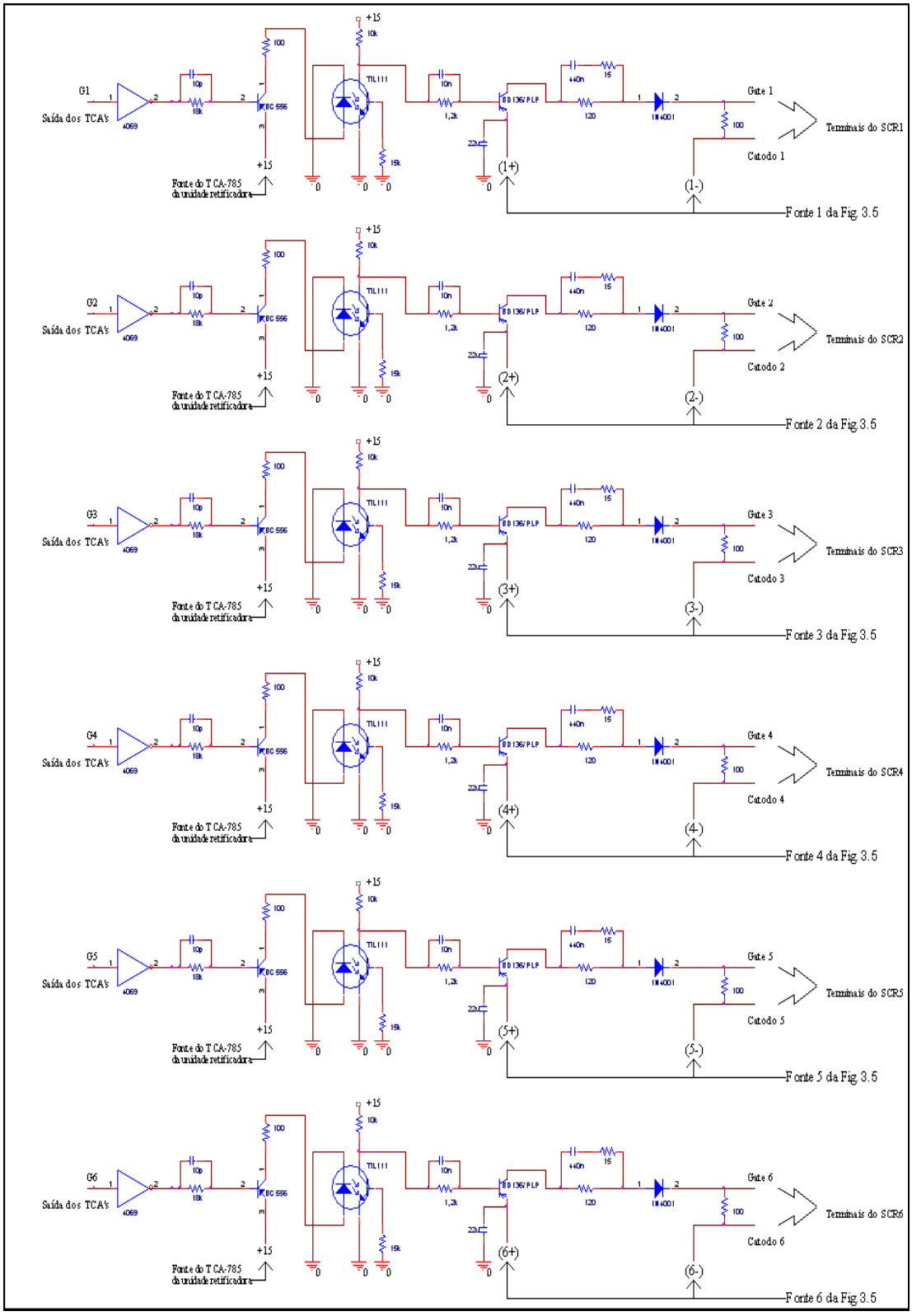

Figura 3.5 – Circuito amplificador de pulsos.

O circuito da fonte utilizada para o circuito da figura 3.4, pode ser visto na figura 3.6.

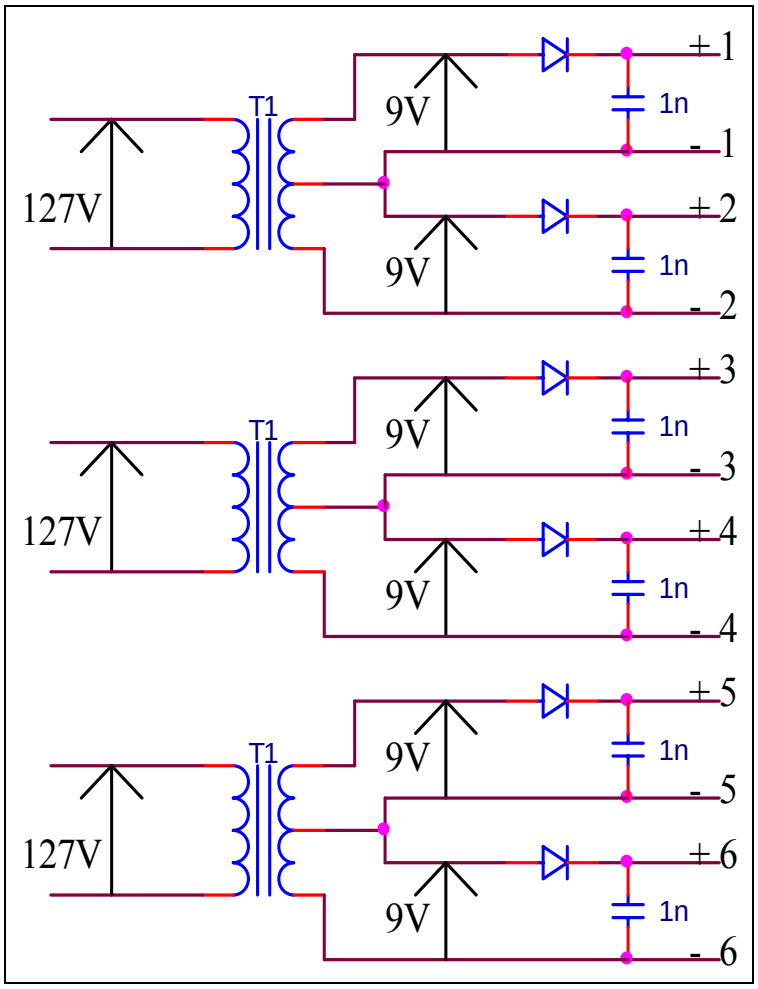

Figura 3.6 – Transformadores de Alimentação dos Pulsos de Disparo

3.2 – Operação em Baixas Velocidades

Em baixas velocidades o valor da tensão nos terminais do motor não é suficiente para que ocorra a comutação natural na ponte inversora. Nestas circunstâncias torna-se necessário utilizar um circuito auxiliar para realizar a comutação forçada. Em geral, esta é feita alterando-se o ângulo de disparo da ponte retificadora de modo que a faixa de variação esteja entre 60° e 120°. Nesta implementação o ângulo de disparo da ponte retificadora será alterado para o valor fixo de 150°. O circuito utilizado para executar esta alteração no ângulo consiste em um somador composto por amplificadores operacionais. Este por sua vez soma um sinal, de um gerador de pulsos, ao valor de tensão na saída do regulador de velocidade, ilustrado na figura 3.7.

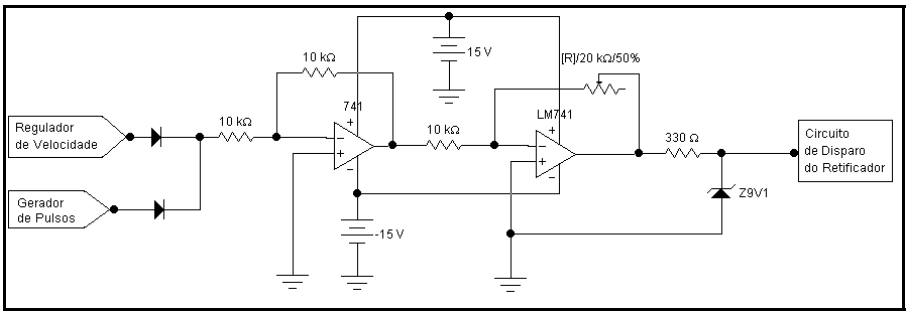

Figura 3.7 – Circuito somador

Os pulsos são gerados em sincronismo com o sinal de tensão do motor, através do transformador com um ângulo de defasagem de 240°, para que haja um pulso sempre que estiver para ocorrer o disparo de um dos seis tiristores. Desta forma a ponte retificadora irá operar na região de descontinuidade [9], extinguindo a corrente no link DC e permitindo a comutação dos tiristores. Este procedimento é ativado para velocidades inferiores a 400 [RPM]. Este valor de velocidade foi definido experimentalmente em ensaios no laboratório.

O circuito gerador de pulsos utiliza circuitos integrados 555 monoestáveis. A largura dos pulsos foi ajustada em laboratório. A figura 3.8 ilustra o circuito gerador de pulsos.

Para reduzir o tempo de extinção da corrente foi inserido um tiristor em antiparalelo ao indutor de alisamento. Este tiristor é disparado pelo circuito gerador de pulsos, através de um driver idêntico ao da figura 3.5.

Para melhorar a qualidade do sinal, utilizou-se um conjunto formado por um transformador Δ/Δ e um circuito detector de zero, inserido entre o transformador com defasagem de 240° e o circuito gerador de pulsos. A figura 3.8, mostra o diagrama de blocos completo do acionamento.

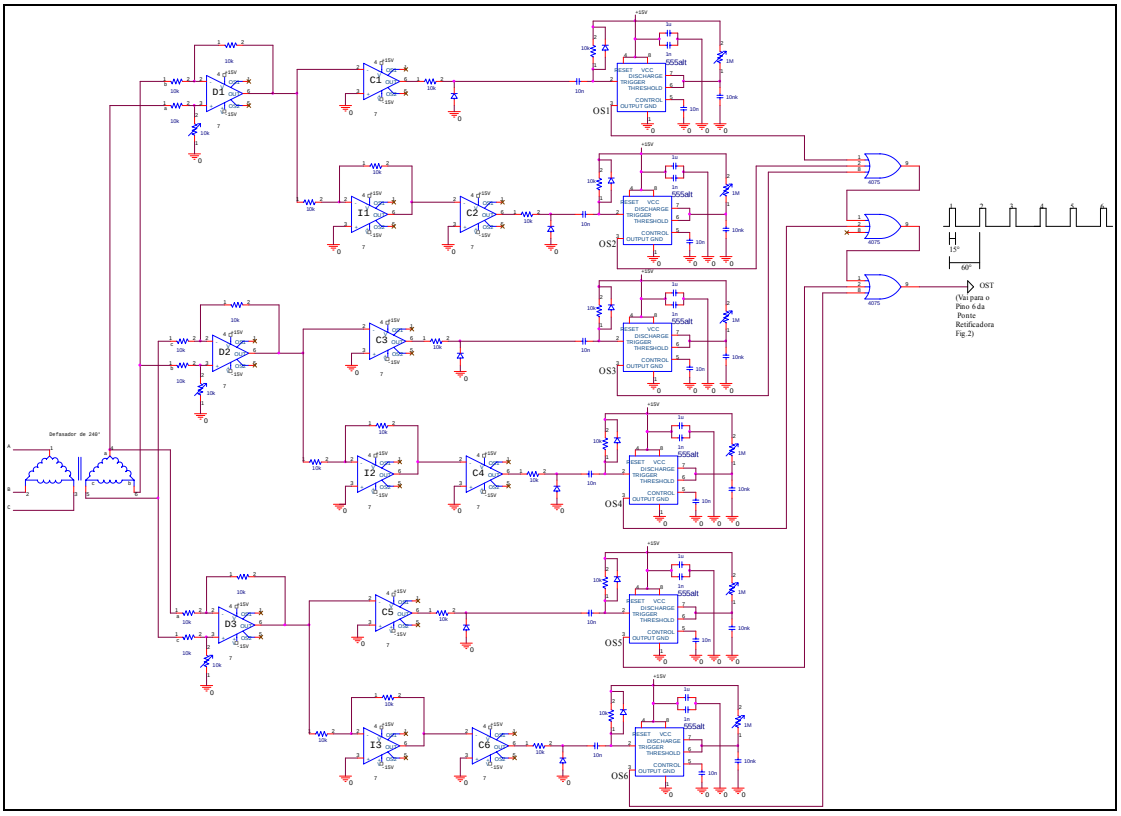

Figura 3.8 – Circuito gerador de pulsos para comutação forçada

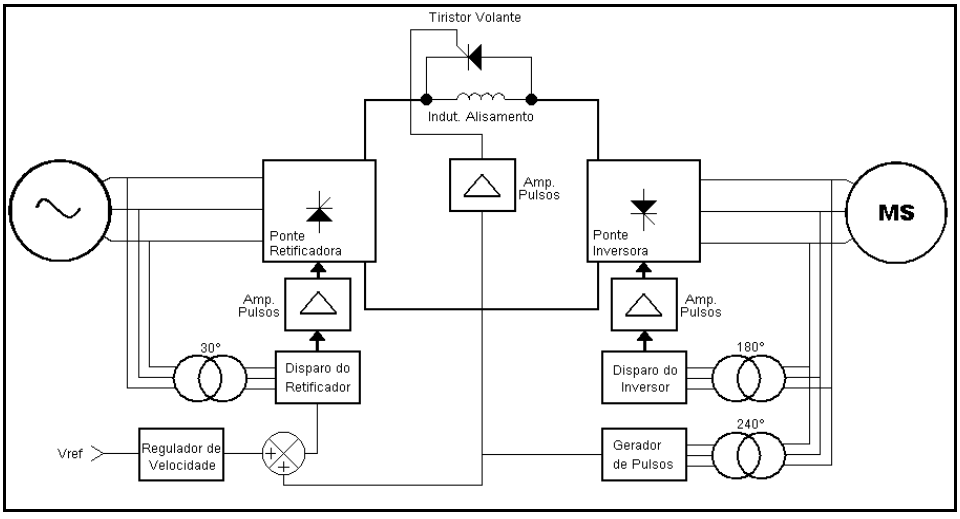

Figura 3.9 – Diagrama de blocos completo

O circuito mostrado não permite a partida do motor e para que este possa funcionar é necessário utilizar uma máquina auxiliar de partida. O motor opera em mono quadrante.

Capítulo 4 - Malha de Controle do Sistema

4.1 - Introdução

O circuito regulador é responsável pelo ajuste do nível de tensão no link DC, o que é feito através do ajuste da tensão de referência para o circuito de disparo da ponte retificadora a qual é utilizada pelo circuito para determinar o ângulo de disparo (α). O circuito regulador, além do controle da velocidade, tem a função de limitar a corrente no motor durante a partida, em condições de variação de carga ou outras situações de distúrbios, o que permite a proteção do motor e do próprio acionamento.

Para que haja um funcionamento adequado do acionamento, deve ser realizado um dimensionamento adequado dos reguladores, tanto da malha de controle de velocidade quanto da malha de controle da corrente. A seguir será mostrado o equacionamento e os diagramas de blocos para esta implementação [2].

4.2 - Equacionamento e Diagrama de Blocos do Motor

A representação do circuito da armadura e a parte mecânica do motor síncrono sobreexcitado estão ilustrados na figura 4.1.

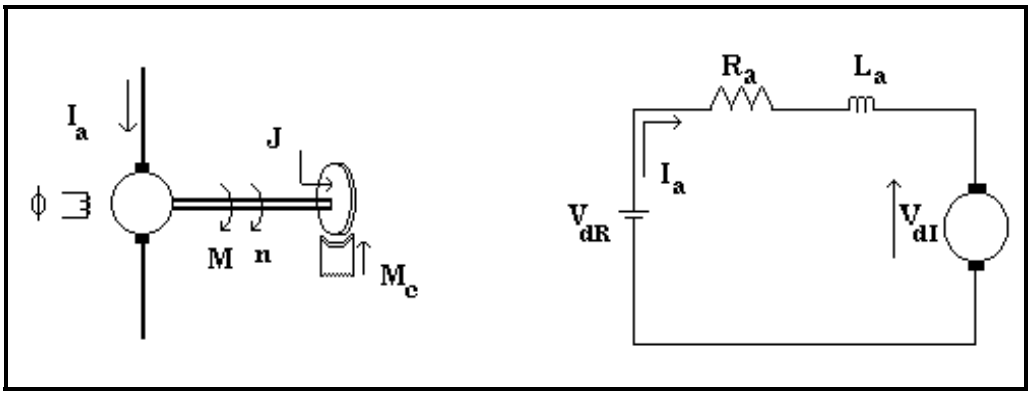

Figura 4.1: Representação da parte mecânica e do circuito elétrico da armadura do motor síncrono [2]

Onde,

$V_{dR} \rightarrow$ Tensão CC de saída do retificador

$V_{dI} \rightarrow$ Tensão CC de saída do inversor

$L_a \rightarrow$ Indutância total (incluindo máquina e indutor do link DC)

$R_a \rightarrow$ Resistência total (incluindo máquina e indutor do link DC)

$\phi \rightarrow$ Fluxo do motor

$I_a \rightarrow$ Corrente de armadura

$M \rightarrow$ Conjugado motor

$M_C \rightarrow$ Torque de carga ou conjugado resistente

$B \rightarrow$ Conjugado acelerante $(B = M - M_C)$

$J \rightarrow$ Momento de inércia (motor + carga)

$n \rightarrow$ Velocidade (RPM)

$\omega \rightarrow$ Rotação (Rad/s)

O conjugado da máquina pode ser obtido como sendo:

$$M = K_1 \cdot \phi \cdot I_a \qquad (4.1)$$

Se K_1 e ϕ forem constantes, o fator de potência do motor também será constante. A constante de tempo de aceleração T_H é obtida por:

$$T_H = \frac{2 \cdot \pi}{60} \cdot \frac{J \cdot n_N}{M_n} \qquad (4.2)$$

Esta constante representa o tempo necessário para o motor atingir a velocidade nominal partindo do repouso, quando acelerado por um conjugado resultante igual ao conjugado nominal do motor.

A figura 4.2 ilustra o diagrama de blocos referente à parte mecânica do motor.

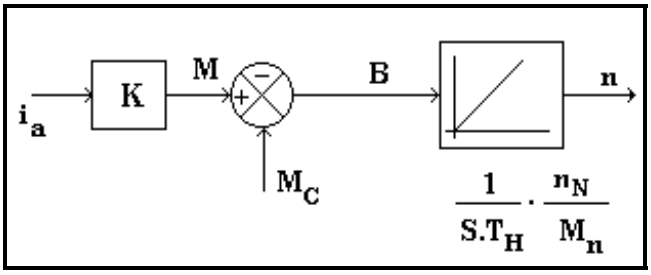

Figura 4.2 - Diagrama de blocos da parte mecânica da máquina síncrona [2]

As grandezas corrente, conjugado de carga, conjugado acelerante, rotação e conjugado motor em "pu" são:

$$i_a = \frac{I_a}{I_N} \qquad \text{(pu)}$$

$$m_C = \frac{M_C}{M_n} \qquad \text{(pu)}$$

$$n_u = \frac{n}{n_N} \qquad \text{(pu)}$$

$$m = \frac{M}{M_n} \qquad \text{(pu)}$$

Para o circuito da armadura, temos:

$$V_{dR} = R_a I_a + L_a \frac{dI_a}{dt} + V_{dI} \qquad (4.3)$$

As grandezas corrente da armadura e as tensões, V_{dR} e V_{dI} (em p.u.), são:

$$i_a = \frac{I_a}{I_N} \qquad \text{(pu)}$$

$$v_{dR} = \frac{V_{dR}}{V_N} = e \text{ (pu)}$$

$$v_{dI} = \frac{V_{dI}}{V_N} = u \text{ (pu)}$$

Fazendo $v_i = \frac{V_N}{R_a I_N}$ podemos apresentar o diagrama de blocos das grandezas elétricas da armadura conforme a figura 4.3.

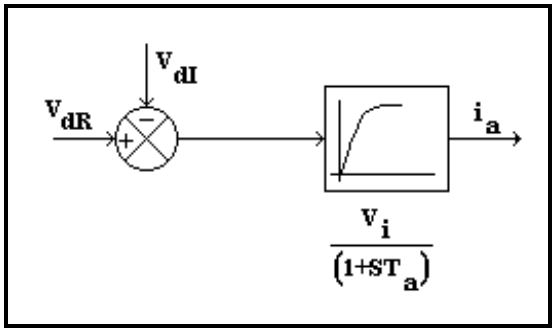

Figura 4.3 - Diagrama de blocos equivalente do circuito da armadura da máquina síncrona [2]

Sendo que,

$$i_a = (v_{dR} - v_{dI}) \cdot \frac{v_i}{1 + ST_a} \qquad [2] \ (4.4)$$

onde:

$V_N \rightarrow V_{dR}$ nominal

$I_N \rightarrow I_a$ nominal

$I_a \rightarrow$ corrente no link DC

Substituindo,

$$V_{dR} = E \qquad (4.5)$$

$$V_N = E_N \qquad (4.6)$$

$$E = \frac{3.\sqrt{2}}{\pi} \cdot U_2 \cdot \cos\alpha \quad (4.7)$$

$$\frac{d_E}{d\alpha} = -\left(\frac{3.\sqrt{2}}{\pi}\right) \cdot U_2 \cdot sen\alpha \qquad (4.8)$$

Onde:

$U_2 \rightarrow$ Tensão de alimentação AC do retificador (f - f). (V)

$\alpha \rightarrow$ Ângulo de disparo do retificador.

$\alpha_u \rightarrow$ Ângulo de disparo em pu.

$$\frac{d\left(E/E_N\right)}{d\left(\alpha/\Pi\right)} = -1{,}35 \times \frac{U_2}{E_N} \times \Pi \times \text{sen } \alpha \qquad (4.9)$$

Em pu:

$$\frac{d(e)}{d(\alpha_u)} = -1{,}35 \times \frac{200}{187} \times \Pi \times \text{sen } \alpha \qquad (4.10)$$

Definindo $V_S \rightarrow$ ganho estático do conversor, temos para $\alpha = 90^o$ o ganho máximo:

$$V_{S1} = \left|\frac{de}{d\alpha_u}\right|_{\alpha=90^o} = 4{,}53 \qquad (4.11)$$

Para $\alpha = 46^o$ o ganho mínimo igual a,

$$V_{S2} = \left|\frac{de}{d\alpha_u}\right|_{\alpha=46^o} = 3{,}26 \text{ (nominal)} \qquad (4.12)$$

E para o ganho estático médio do conversor, temos:

$$V_S = 3{,}90 \qquad (4.13)$$

A constante v_i pode ser interpretada como a relação entre a corrente com rotor bloqueado e a corrente nominal, com tensão nominal aplicada à armadura, sendo calculada por:

$$v_i = \frac{E_N}{R_a I_N} = 12{,}57 \qquad (4.14)$$

E para o ganho total:

$$V_{Sia} = v_s \cdot v_i \qquad (4.15)$$

$$V_{Sia} = 49,02 \quad (4.16)$$

O fator $\cdot \frac{v_i}{1 + ST_a}$ pode ser considerado como elemento retardador de 1ª ordem.

A constante T_a, pode ser obtida medindo-se a indutância L_a e a resistência R_a do circuito da armadura, sendo que a indutância de alisamento deve ser incluída em série com o circuito.

$$T_a = \frac{L_a}{R_a} \qquad (4.17)$$

O diagrama de blocos completo da máquina, incluindo o circuito da armadura e a parte mecânica está ilustrado na figura 4.4.

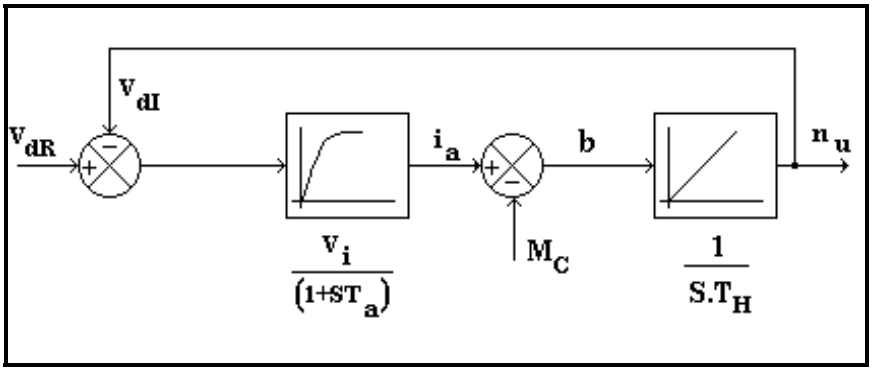

Figura 4.4 - Diagrama de blocos completo da máquina [2]

4.3 – Estratégia de Controle[2]

A estratégia utilizada para a escolha e ajuste dos parâmetros dos reguladores é baseada no método de otimização simétrica [2].

O método de otimização em função da simetria (OS) é indicado para sistemas que apresentam elementos retardadores, elementos de ação proporcional, elementos de tempo morto de pouca importância e elementos de ação integral. Sua principal vantagem é a rápida correção dos efeitos devido a perturbações. Porém podem apresentar na resposta a degrau um overshoot elevado de até 43%. A introdução de um componente de alisamento

do sinal de referência reduz o overshoot, mas aumenta o tempo de acomodação da resposta a degrau do sistema.[2].

4.3.1 – Escolha e Ajuste dos Reguladores [2]

Os dados da máquina síncrona podem ser vistos na tabela 4.1.

Fabricante	Equacional
Tensão de alimentação	$V_{dR} = 230\,[V]$
Potência nominal	$P_N = 2[KVA]$
Velocidade nominal	$n_N = 1800\,[RPM]$
Corrente armadura nominal	$I_N = 5[A]$

Tabela 4.1 - Dados nominais do motor síncrono utilizado [2]

As constantes T_H e T_a foram obtidas em laboratório e equivalem a:

$$T_H = 1{,}2[s]$$

$$T_a = 70\,[ms]$$

Foi adotado um valor para a constante de tempo do circuito de disparo igual a:

$$T_{SS} = 2{,}5[ms] \quad (4.18)$$

O sinal de corrente é obtido utilizando um transdutor de corrente constituído de uma ponte tiristorizada alimentada pelo lado trifásico através de TC's, ocasionando a inserção de um filtro, cuja constante pode ser obtida pela equação (4.19).

$$T_{gi} = 1{,}5[ms] \text{ (adotado)} \qquad (4.19)$$

A malha de regulação de corrente completa é apresentada na figura 4.5.

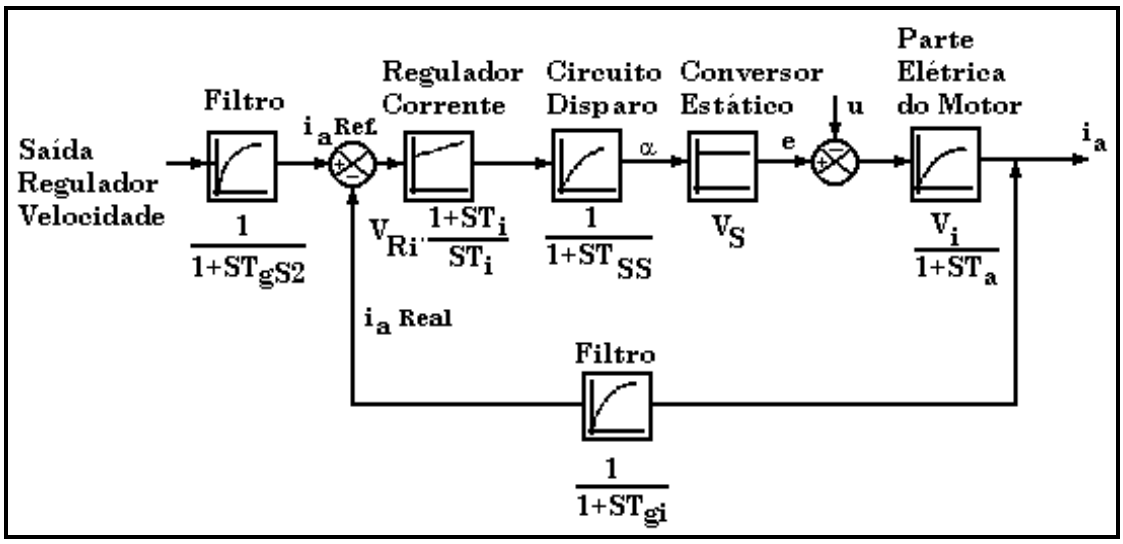

Figura 4.5 - Malha de regulação de corrente [2]

Aplicando a metodologia de otimização e controle proposta por FROHR; ORTTENBURGER (1986) [8], obteve-se as constantes de tempo e o tipo de regulador requerido. No caso, o tipo PI (Proporcional Integral).

Constante de tempo σ:

$$\sigma = T_{SS} + T_{gi}$$

$$\sigma = 4[ms]$$

Ganho direto V_{Sia}:

$$V_{Sia} = v_s \cdot v_i$$

$$V_{Sia} = 49,02$$

O valor da constante de tempo de alisamento de sinal, para o filtro do valor de referência para a malha de corrente foi:

$$T_{gS2} = 14,55[ms]$$

O valor do ganho e da constante de tempo do regulador equivalem a:

$$V_{Ri} = \frac{T_a}{2 \cdot V_{Sia} \cdot \sigma} \qquad (4.20)$$

$$V_{Ri} = 0{,}14$$

$$T_i = 4 \cdot \sigma \cdot \frac{T_a}{T_a + 3 \cdot \sigma} \quad (4.21)$$

$$T_i = 13{,}11 [\text{ms}]$$

A constante de tempo equivalente de otimização da malha de corrente, incluindo o regulador e o filtro de referência é dada por:

$$T_e = 2 \cdot \sigma + \frac{1}{2} \cdot T_{gS2} \quad (4.22)$$

$$T_e = 15 [\text{ms}]$$

A partir destes dados pode-se obter a malha de regulação de velocidade, como pode ser visualizada na figura 4.6.

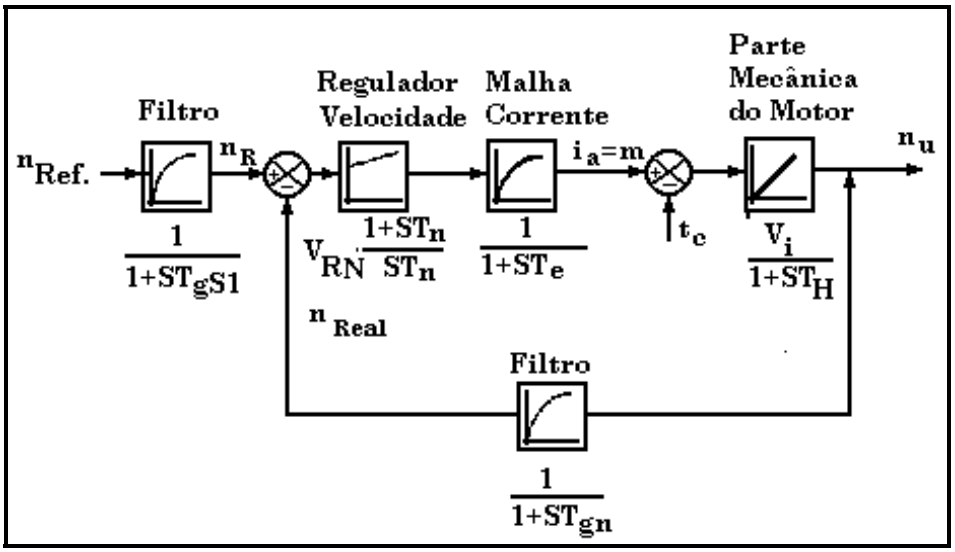

Figura 4.6 - Diagrama de blocos da malha de regulação de velocidade [2]

A constante para o filtro foi adicionado, devido ao transdutor de velocidade é:

$$T_{gN} = 100\,[\text{ms}]$$

O mesmo procedimento utilizado para a malha de regulação de corrente, foi utilizado para malha de regulação de velocidade [2].

Sendo:

$$\sigma' = 115\,[\text{ms}]$$

De [2], obteve-se que o regulador requerido é do tipo PI.

$$T_{gSl} = T_{e'} = 4\sigma' = 460[\text{ms}]$$

O valor do ganho e da constante de tempo equivalem a:

$$V_{RN} = \frac{T_H}{2 \cdot \sigma'}$$

$$V_{RN} = 5{,}22 \cong 5$$

$$T_n = 4 \cdot \sigma'$$

$$T_n = 460\,[\text{ms}]$$

4.3.2 – Diagrama do Circuito do Regulador

A figura 4.7 ilustra o diagrama do circuito utilizado. Os ajustes dos ganhos são realizados através de potenciômetros instalados no painel do módulo. Este possui também um mostrador onde se pode visualizar a velocidade, medida através da tensão do tacogerador.

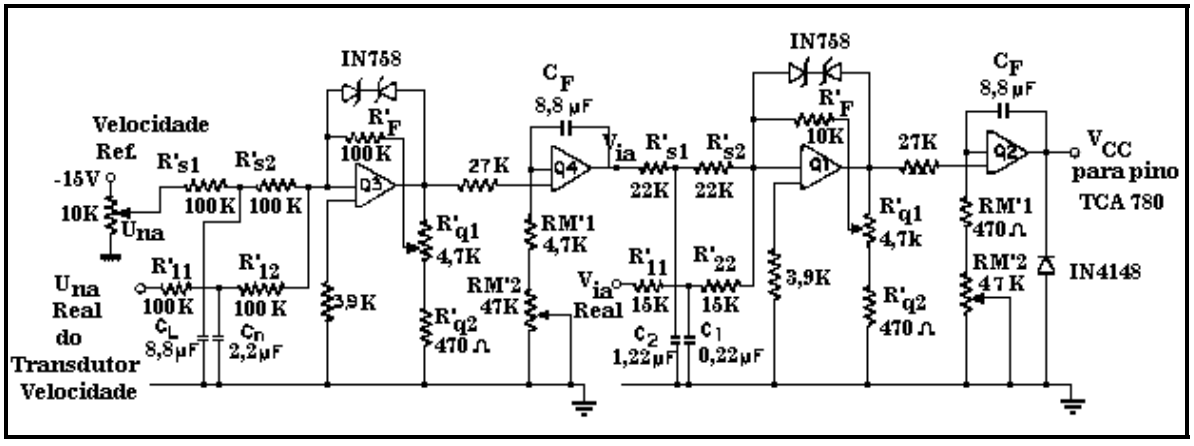

Figura 4.7 - Reguladores e filtros

Capítulo 5 – Controle de Disparo da Ponte Inversora – Por Microcomputador

5.1 - Introdução

Para executar a partida do motor síncrono sem a necessidade de uma máquina auxiliar de partida, foi implementado um novo circuito de controle de disparo da ponte inversora.

Para esta etapa foi utilizado um microcomputador. O programa foi implementado em linguagem C para rodar em ambiente operacional MS-DOS. A placa de aquisição de dados utilizada foi a PCL-711B da Advantech. No entanto, a aquisição foi feita utilizando o modo de acesso direto às portas de entradas e saídas, sem a utilização do "driver" fornecido pelo fabricante para permitir ao software uma maior independência do hardware.

O avanço na tecnologia dos microprocessadores e microcontroladores, permitiu que os mesmos se tornassem cada vez mais compactos e velozes e, ainda, que houvesse uma significativa redução no preço dos mesmos. Estes fatores aliados a facilidade de manutenção, a redução do número de componentes do circuito e a flexibilidade devido ao software, fizeram com que os circuitos microcontrolados ou microprocessados se tornassem uma solução bastante eficiente.

5.2 - Hardware

Foram utilizados três canais A/D da placa PCL-711B (de um total de 8) para a aquisição das tensões das fases a, b e c nos terminais de baixa tensão do transformador defasador Δ/Z (180°). O tempo de conversão da placa é de 25[μs], com uma resolução de 12 bits e uma faixa de tensão de entrada de ±5[V]. Para acomodar o sinal do transformador para a faixa de entrada da placa utilizou-se um divisor de tensão com uma relação ½, utilizando diodos do tipo zener para limitar a tensão na saída. O circuito está ilustrado na figura 5.1. A figura 5.2 mostra as formas de onda para o sinal de entrada e o sinal de saída do mesmo.

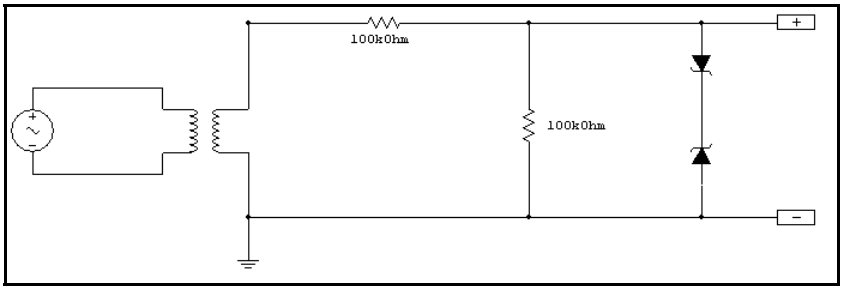

Figura 5.1 – Circuito divisor de tensão.

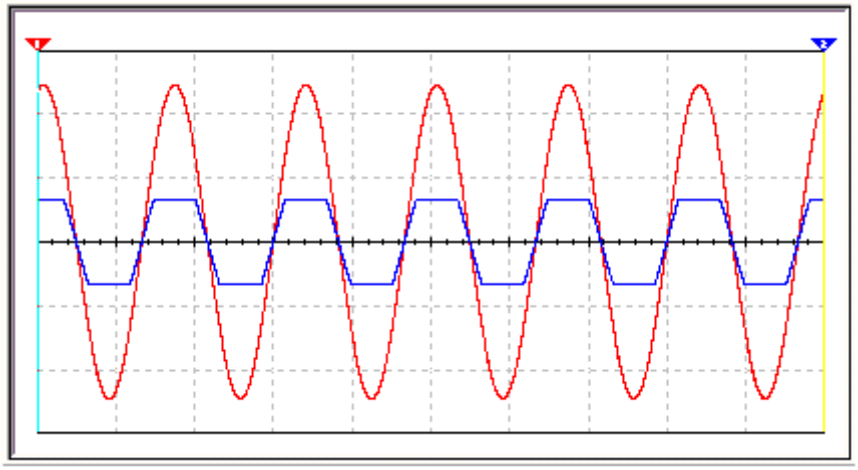

Figura 5.2 – Formas de onda na entrada e na saída do circuito divisor de tensão.

O circuito divisor perde sua linearidade quando a tensão em sua entrada se aproxima do dobro da tensão de zener (4,3[V]). No entanto, o valor da tensão apenas será utilizado enquanto o motor estiver funcionando em baixa velocidade, onde o valor da tensão é baixo o suficiente para estar dentro da faixa de linearidade do circuito. Fora desta faixa, apenas o sinal da tensão será considerado.

Embora a placa possua saída digital, foi utilizada a porta paralela do PC para gerar os pulsos de disparo do inversor e para ativar ou desativar a comutação forçada.

Os bits de 0-5 da porta (endereço 0x378) geram, respectivamente, os pulsos para os tiristores de 1-6. A saída da porta paralela está conectada a um buffer e este conecta as saídas correspondentes aos bits 0-5 ao amplificador de pulsos. Um conjunto de seis LED's foram colocados no circuito para que se possa monitorar visualmente o estado dos pulsos. A saída do buffer para o bit 6 é utilizada para ativar o modo de comutação forçada. Os bits de controle da porta paralela (endereço 0x37A) são utilizados para que se possa alternar entre o circuito de disparo microprocessado e o não microprocessado.

5.3 - Software

A tensão nos terminais do motor é utilizada como referência para os pulsos de disparo dos tiristores. Conforme mencionado no Capítulo 3, a ponte opera com ângulo fixo de disparo de 150°, obtido através de um transformador de referência especial delta / zigue-zague. O pulso para o disparo dos tiristores é gerado através de um algoritmo que identifica a seqüência correta a ser disparada:

5.3.1 - O Algoritmo Para Geração dos Pulsos de Disparo

Sinal da Tensão nas Fases

SA ← (va>0);
SB ← (vb>0);
SC ← (vc>0);

Onde: va, vb e vc são os valores de tensão nas fases do transformador de referência, obtidos através do conversor A/D. E as variáveis SA, SB e SC, indicam o sinal da tensão.

Condições Para o Disparo dos Tiristores

P(1) ← SA and not SB

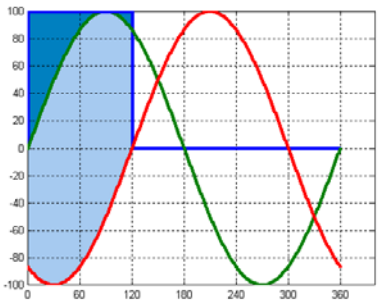

Figura 5.3 – Condição de Disparo do Tiristor 1

P(2) ← SA and not SC

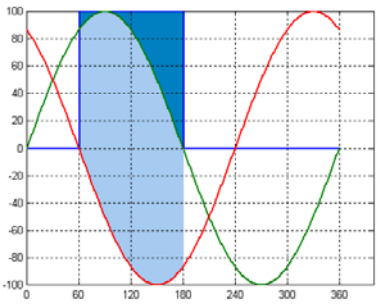

Figura 5.4 – Condição de Disparo do Tiristor 2

P(3) ← SB and not SC

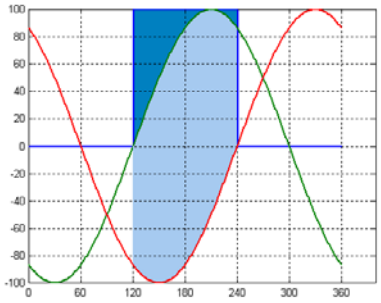

Figura 5.5 – Condição de Disparo do Tiristor 3

P(4) ← SB and not SA

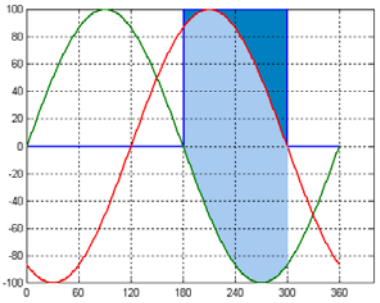

Figura 5.6 – Condição de Disparo do Tiristor 4

P(5) ← SC and not SA

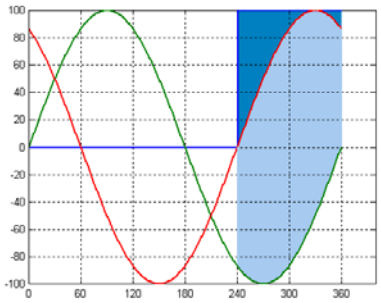

Figura 5.7 – Condição de Disparo do Tiristor 5

P(6) ← SC and not SB

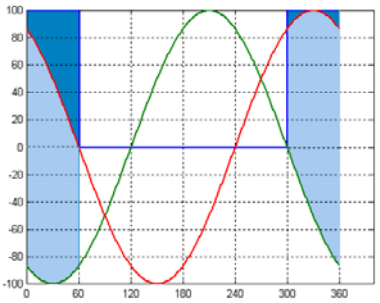

Figura 5.8 – Condição de Disparo do Tiristor 6

O vetor P(n) representa o estado dos pulsos nos tiristores.

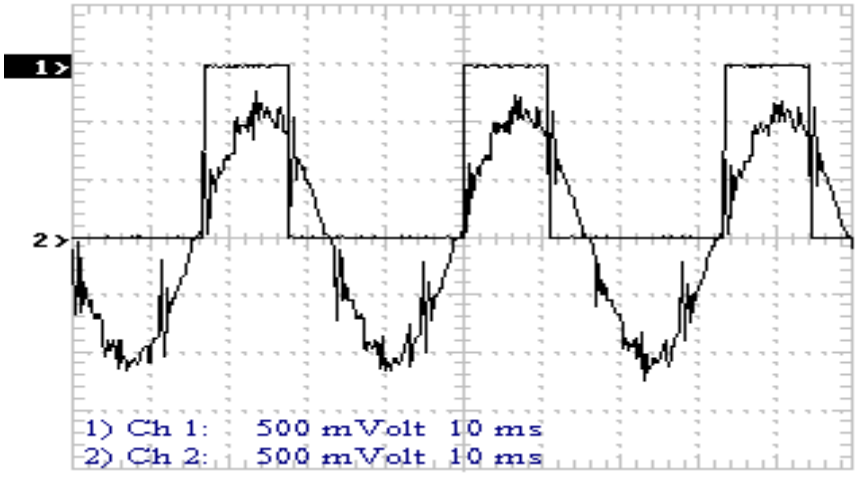

Figura 5.9 – Tensão na fase A e Pulso de disparo no Tiristor 1.

A figura 5.9 mostra os sinais reais da tensão na fase A do transformador de referência e o sinal gerado para o pulso do tiristor 1, obtidos em laboratório.

5.3.2 - Partida

Conforme já foi mencionado anteriormente, em velocidades muito baixas, da ordem de 10% da velocidade nominal, a tensão nos terminais da máquina possui um valor muito baixo, o que pode não permitir a comutação natural dos tiristores. Entretanto, quando o motor está em repouso, não há tensão na máquina. Portanto, como a geração de pulsos é realizada por sensor de tensão, está não funcionará. Então é necessário obter um valor mínimo de tensão nos terminais do motor para que a máquina possa partir utilizando o modo de comutação forçada.

Se dois tiristores forem disparados de forma que haja a circulação de corrente em duas das fases do motor, ocorrerá (dependendo da posição do rotor) uma tentativa do campo no rotor de se alinhar com campo gerado no estator. Conforme ilustrado nas figuras 5.10, 5.11 e 5.12.

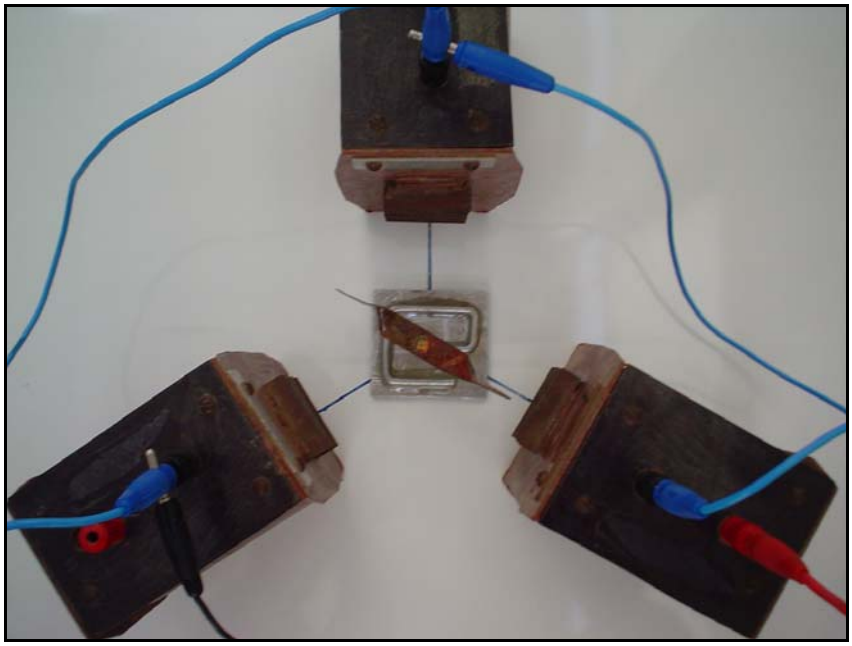

Figura 5.10 – Alinhamento do rotor para Campo gerado por Ia e Ib.

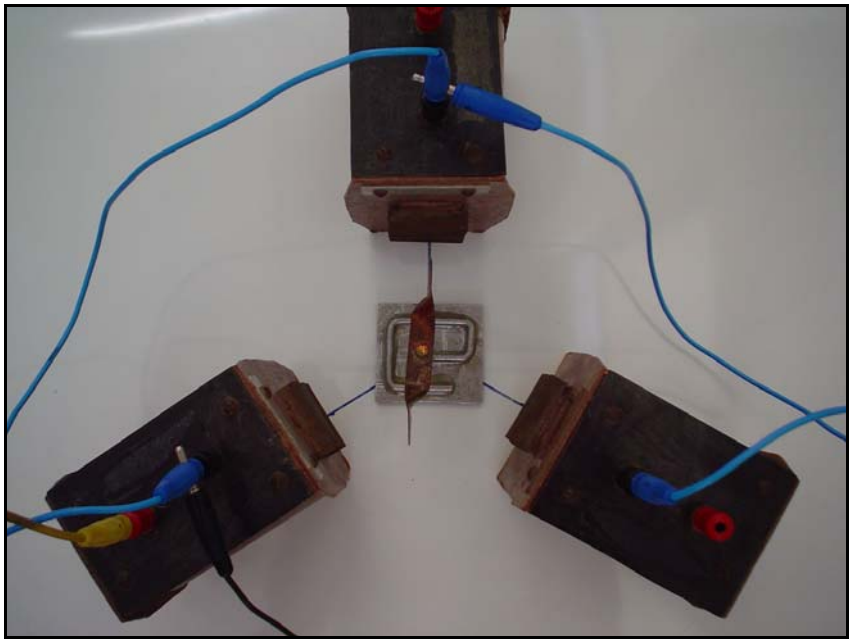

Figura 5.11 – Alinhamento do rotor para Campo gerado por Ib e Ic.

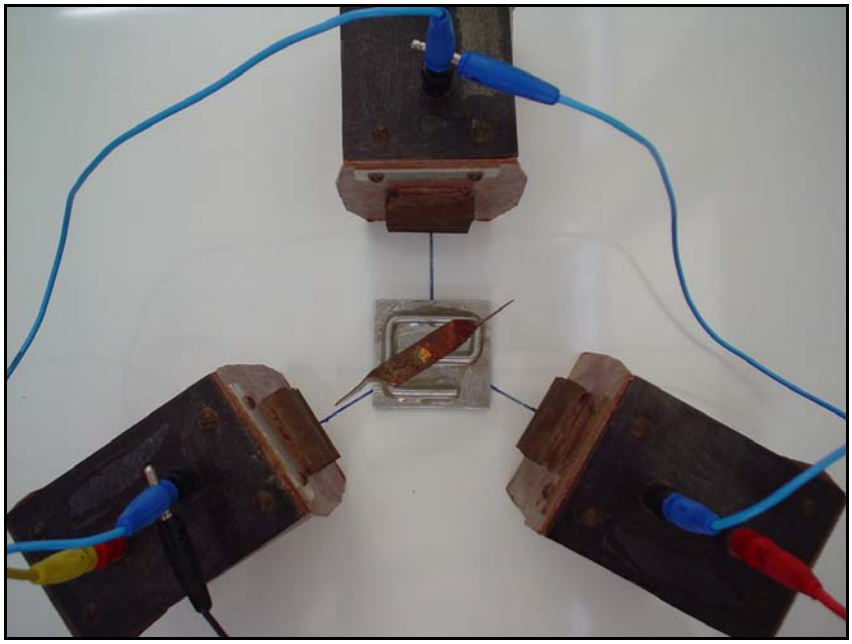

Figura 5.12 – Alinhamento do rotor para Campo gerado por Ia e Ic.

Nas figuras 5.10, 5.11 e 5.12, temos três posições de alinhamento do rotor para o campo formado respectivamente pelas correntes circulando nas fases **a, b; b, c; a, c**. Portanto, para este motor teremos um total de seis possíveis posições de alinhamento do rotor.

Mas como a máquina utilizada neste trabalho possui dois pares de pólos, o número de posições dobra, ou seja, doze. Levando-se em conta esta característica foi implementado o seguinte procedimento.

A rotina de partida dispara arbitrariamente um par de tiristores, injetando um pulso de corrente nos enrolamentos do motor com um período ajustado, experimentalmente, para permitir o alinhamento do rotor com o campo gerado. Como o conjugado depende da posição do rotor, é necessário gerar uma seqüência de disparos para garantir o alinhamento do rotor na posição desejada. Para que haja a troca do disparo de um par de tiristores para o par seguinte é necessário forçar a comutação, utilizando o método descrito anteriormente. Em seguida, é gerada uma nova seqüência de disparos com um período menor o que faz com que o motor atinja um valor mínimo de velocidade, para que o algoritmo que identifica a seqüência de disparo dos tiristores possa funcionar. No entanto, a comutação continua sendo feita por software, ocorrendo sempre quando houver a troca de disparo de um tiristor. Este procedimento é mantido até que o motor atinja uma velocidade onde possa ocorrer a comutação natural da ponte. As figuras 5.14 e 5.15 mostram o fluxograma do Software. A figura 5.13 ilustra o diagrama de bloco do sistema.

O software trabalha com uma freqüência de amostragem fixa de 2[kHz], sendo que a leitura dos canais de entrada, o algoritmo de detecção da seqüência de disparo e o de geração de pulsos para partida são executados através de uma rotina de interrupção. Esta é executada de forma síncrona na freqüência mencionada.

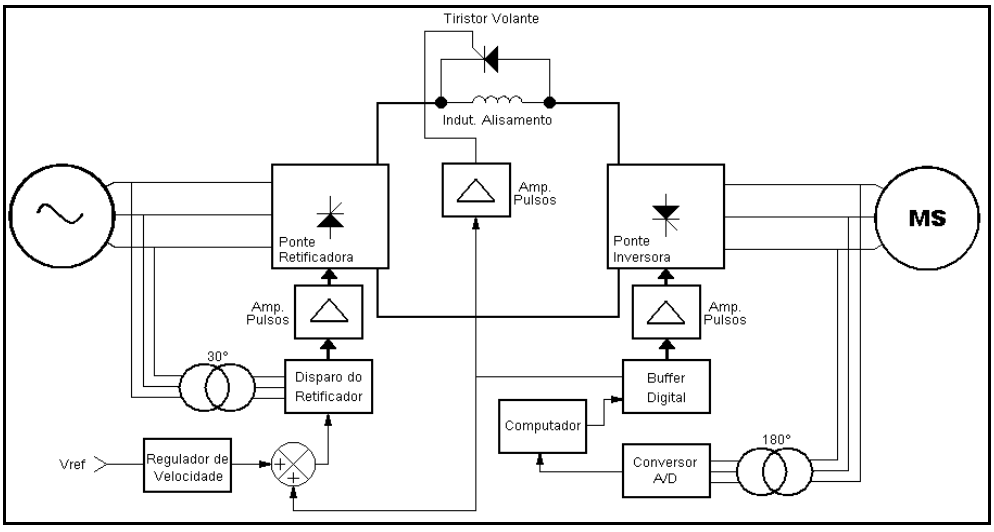

Figura 5.13 – Diagrama de blocos dos sistema utilizando microprocessador.

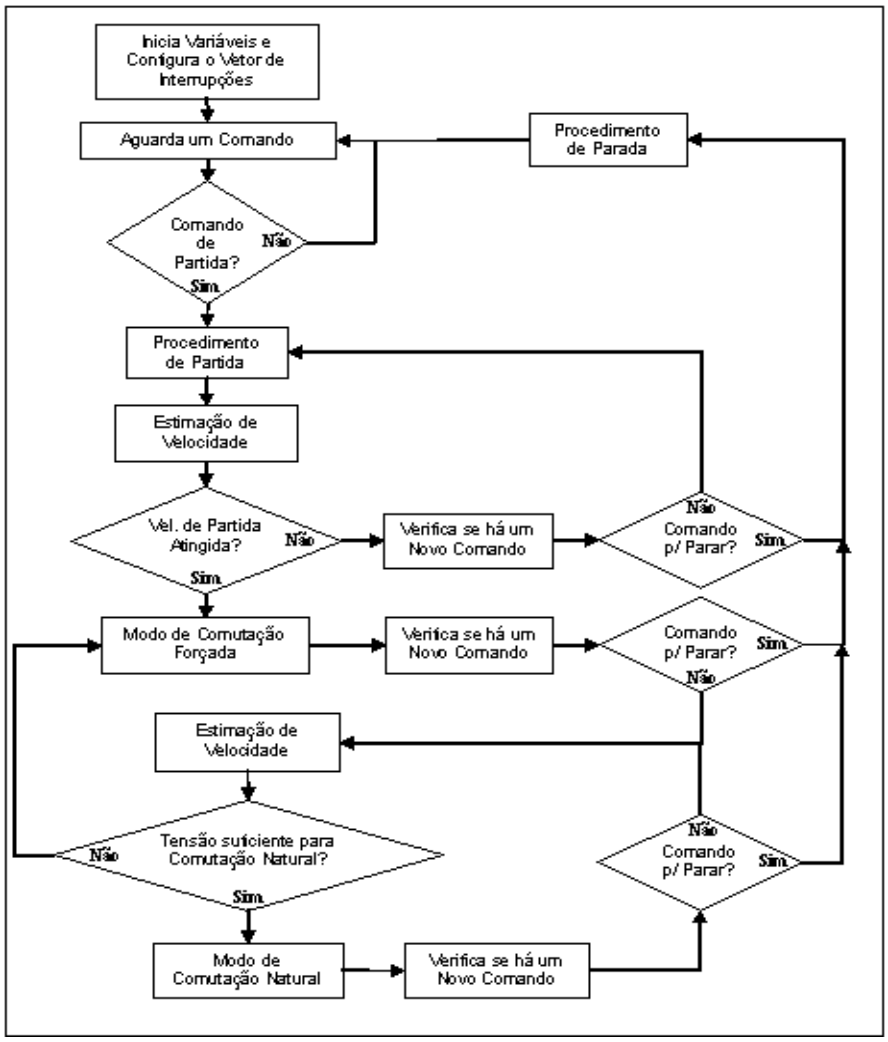

Figura 5.14 – Software Controle da Ponte Inversora (Rotina Principal).

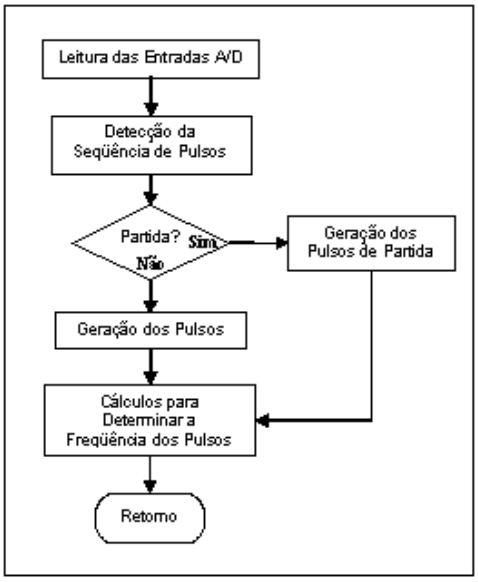

Figura 5.15 – Rotina de Interrupção.

Embora o regulador de velocidade utilize um tacogerador para obter o sinal de velocidade do motor, o software de disparo do inversor utiliza o próprio sinal de tensão da máquina para estimar a velocidade e permitir a execução do procedimento de partida. O que é feito de duas maneiras, a saber:

- Freqüência da tensão nas fases da máquina – sendo a freqüência da tensão diretamente proporcional à velocidade da máquina. No entanto, a freqüência não é medida diretamente pela tensão de fase. O valor medido é o período dos pulsos de disparo dos tiristores, pois este sinal é mais limpo em relação aos ruídos. O algoritmo calcula a freqüência dos seis tiristores e determina a partir destes valores a velocidade da máquina.
- Valor rms médio da tensão nas fases – dentro da faixa de linearidade do circuito divisor de tensão é possível medir o valor da tensão nas fases do motor. Este valor rms é multiplicado por uma constante de proporcionalidade (levantada de forma experimental) para resultar na velocidade do motor, pois a relação entre a tensão e a freqüência é constante. O valor utilizado para o cálculo é a média dos três valores rms (valor médio quadrado) das tensões de fase.

O dois métodos foram implementados, pois o cálculo da velocidade pela medição de freqüência apenas demonstra boa precisão em velocidades superiores a 400[RPM]. O segundo método, entretanto, apresenta bons resultados, mas devido às características do circuito divisor de tensão, já citadas, este circuito apenas pode ser utilizado em velocidades inferiores a 500[RPM]. Então o valor da velocidade final é o resultado das duas técnicas combinadas.

5.3.3 – Interface do Usuário

A figura 5.16 ilustra a tela do programa de controle do inversor. Uma interface gráfica foi implementada para facilitar a interação do usuário com o sistema, uma vez que o aplicativo foi desenvolvido para o sistema operacional MS-DOS.

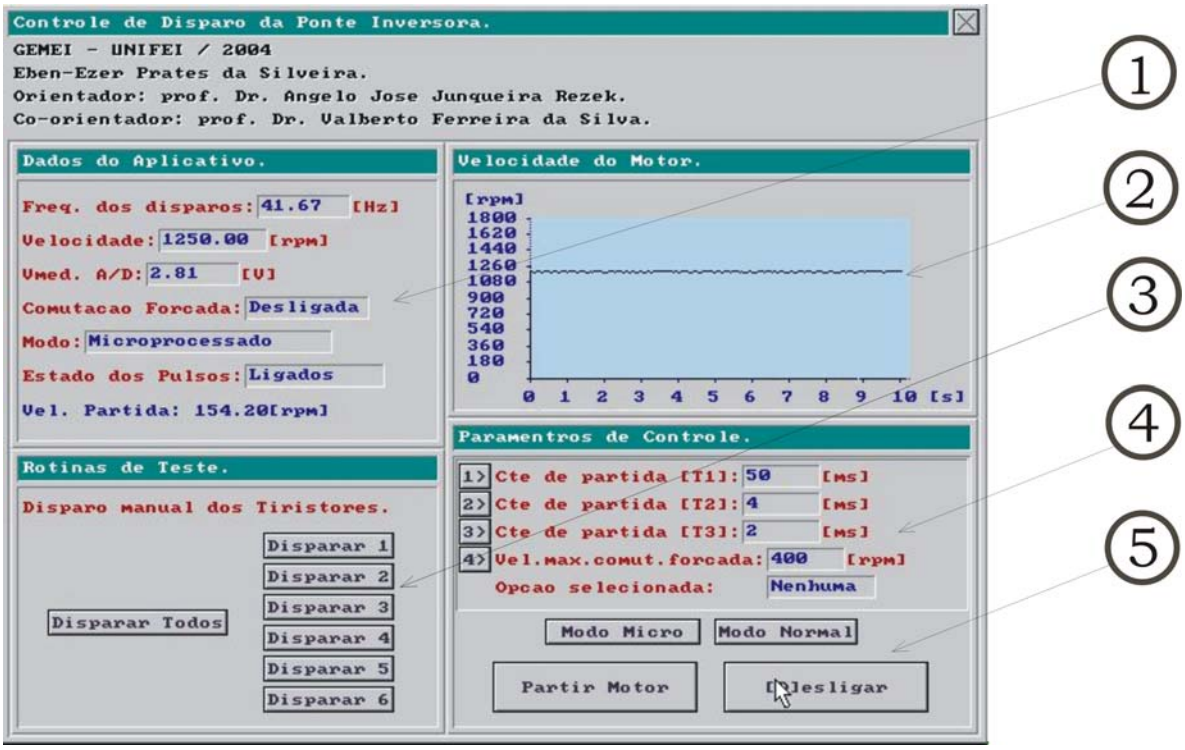

Figura 5.16 – Tela do software

O programa foi desenvolvido de forma que sua utilização seja simples. Para facilitar a explicação, a tela foi dividida em cinco partes:

1. Janela de dados – Exibe informações referentes ao funcionamento do sistema, como: freqüência dos pulsos, velocidade do motor, tensão rms média lida pelos conversores A/D, estado do procedimento de comutação forçada, modo de funcionamento (microprocessado ou não), estado do gerador de pulsos e outras informações como velocidade obtida na partida.
2. Janela do gráfico de velocidade – exibe a variação da velocidade ao longo do tempo, em uma janela de 10[s], com um intervalo de amostragem de 100[ms].

3. Janela de disparo dos tiristores – permite testar os tiristores. Os botões numerados de 1 a 6, executam o disparo dos tiristores. Porém, a numeração não se refere aos tiristores mas a seqüência da dupla a ser disparada, exemplo: “Disparar 1”, irá disparar os tiristores 1 e 2. “Disparar 2”, por sua vez, disparará os tiristores 2 e 3. Já o botão “Dispara Todos”, dispara em seqüência os seis pares.

4. Parâmetros de controle – possibilita ajustar as constantes de tempo utilizadas para gerar os pulsos de partida e também ajustar o limite máximo de velocidade para a comutação forçada.

5. Botões de controle – estes permitem alternar entre o controle de disparo microprocessado e o circuito mostrado no Capítulo 3. Os outros dois botões executam a partida ou o desligamento do motor.

Capítulo 6 – Resultados Experimentais

As figuras a seguir mostram os resultados obtidos com o acionamento implementado no laboratório.

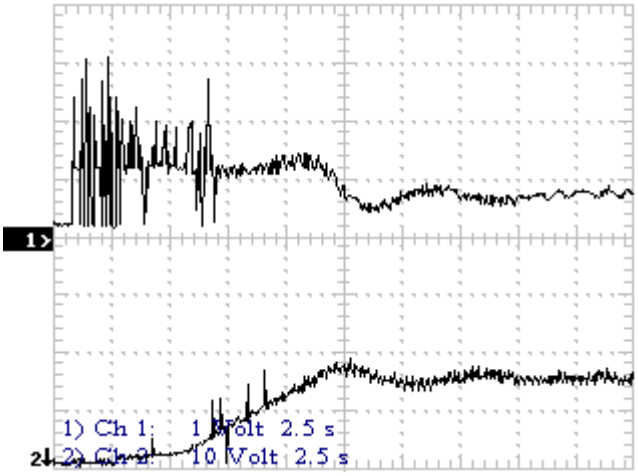

Figura 6.1 – Partida do Motor Síncrono.

A figura 6.1 mostra a variação da corrente (parte superior, escala 1[A/div]) e da velocidade (parte inferior, escala 800 [RPM/div]) durante o processo de partida do motor síncrono. A partida foi realizada com o motor em vazio. Na figura 6.2, pode ser vista a corrente no link DC durante a comutação forçada.

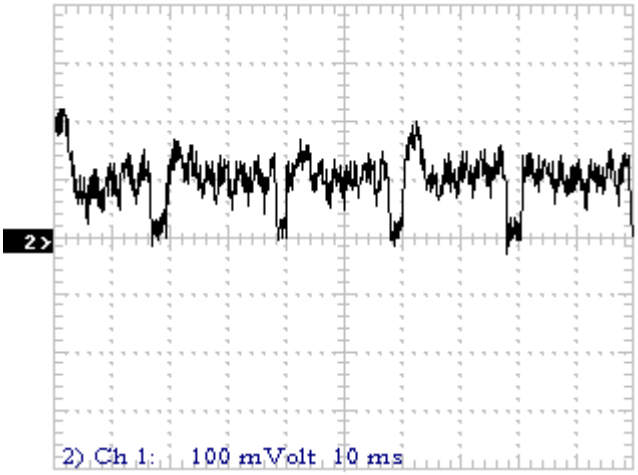

Figura 6.2 – Corrente no Link DC.

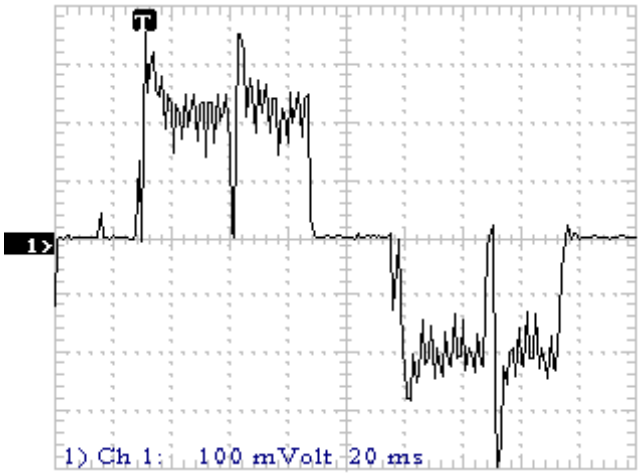

Figura 6.3 – Corrente na fase A do MS.

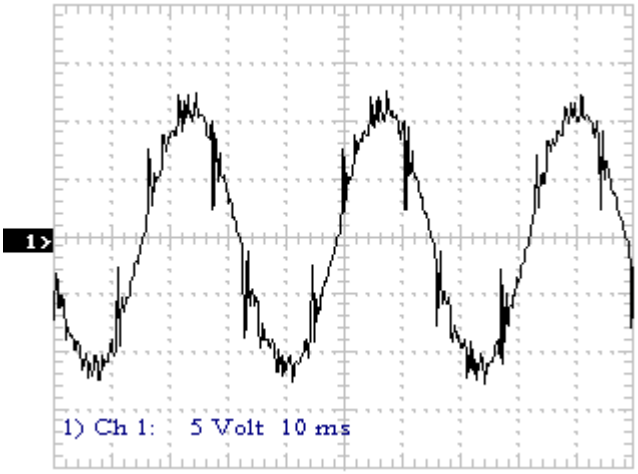

Figura 6.4 – Tensão na fase A do MS.

Pode-se observar na figura 6.3, a corrente na fase "a" do motor síncrono (escala 1[A/div]), também durante a comutação forçada. Na figura 6.4, a tensão na fase "a" do motor (escala 50[V/div]).

Para verificar a atuação do regulador aplicou-se um degrau de carga na máquina, utilizando-se um gerador de corrente contínua acoplado ao eixo do motor. Este gerador estava alimentando um banco de resistências, as quais representavam a carga variável.

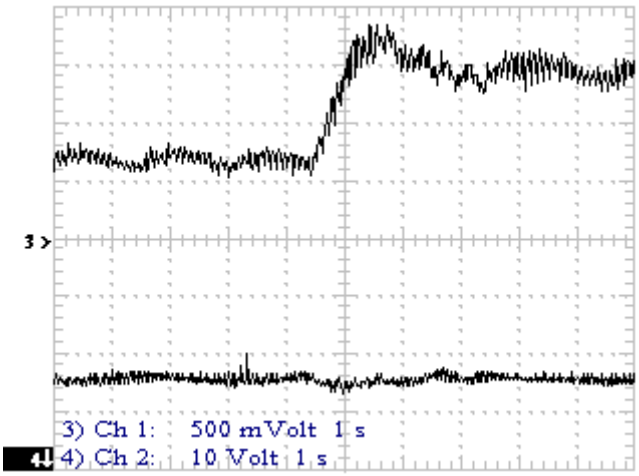

Figura 6.5 – Aplicação à carga.

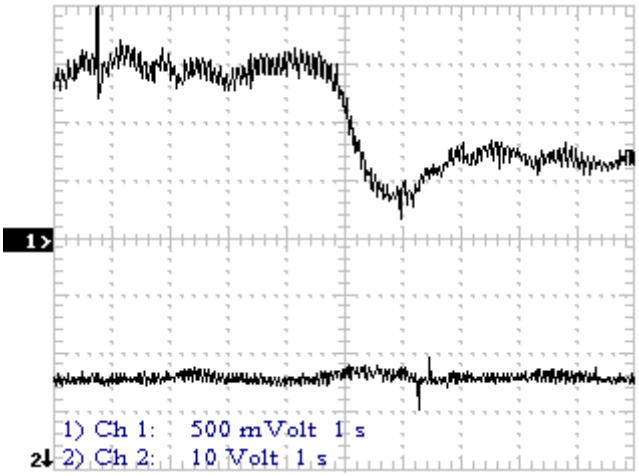

Figura 6.6 – Retirada da carga.

As figuras 6.5 e 6.6 mostram, respectivamente, a aplicação e retirada da carga. Onde parte superior das figuras, pode ser vista a variação da corrente no link DC (escala 1[A/div]), enquanto na parte inferior a variação da velocidade do motor (escala 800 [RPM/div]).

As figuras 6.7, 6.8, 6.9, 6.10 e 6.11 mostram as tensões no link DC e na ponte retificadora para três diferentes valores de velocidade do motor.

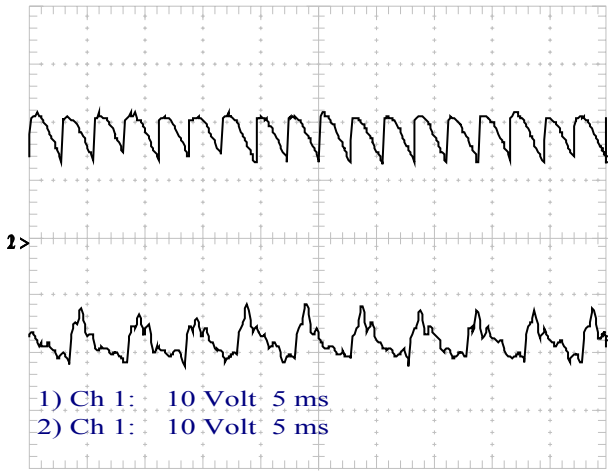

Figura 6.7 - Tensão na ponte retificadora e inversora - 1000[RPM].

A figura 6.7 mostra as formas de onda nas tensões do link DC (parte superior da figura) e no inversor (parte inferior) para uma velocidade de 1000[RPM] (escala 50[V/div]).

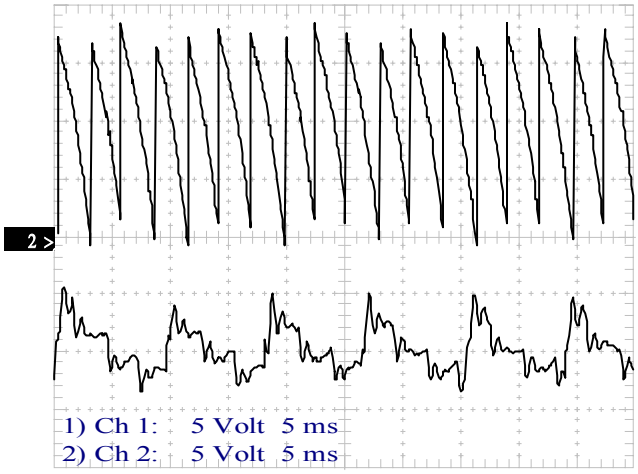

Figura 6.8 - Tensão na ponte retificadora e inversora - 576[RPM].

A figura 6.8 mostra as formas de onda nas tensões do link DC (parte superior da figura) e no inversor (parte inferior) para uma velocidade de 576[RPM] (escala 25[V/div]).

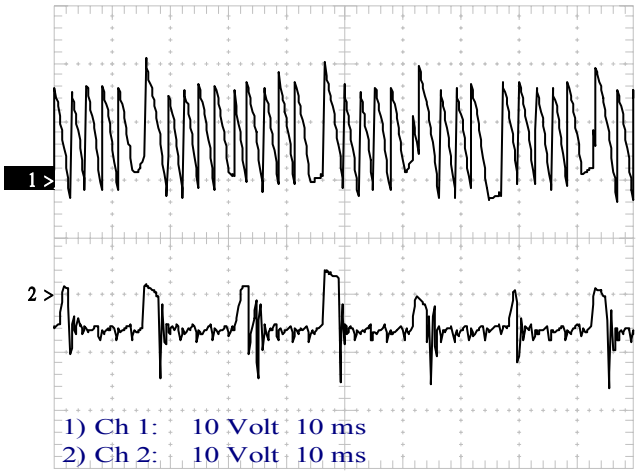

Figura 6.9 - Tensão na ponte retificadora e inversora - 340[RPM].

A figura 6.9 mostra as formas de onda nas tensões do link DC (parte superior da figura) e no inversor (parte inferior) para uma velocidade de 340[RPM] (escala 50[V/div]). Para esta velocidade utiliza-se a comutação forçada. Os sinais foram deslocados para facilitar a visualização, as setas no canto esquerdo indicam as referências (terra) dos sinais.

Com base nos resultados pode-se considerar que a atuação do regulador de velocidade foi satisfatória.

Os dados obtidos foram gerados com o circuito microprocessado, pois este substitui completamente o circuito mostrado no Capítulo 3, além do fato de que, apenas, este circuito é que possui a opção da partida automática.

Capítulo 7 – Conclusão

Os resultados obtidos neste acionamento, mono quadrante, para o motor síncrono foram considerados satisfatórios. O circuito de disparo, apesar de não utilizar filtros digitais, mostrou-se suficientemente imune aos ruídos na tensão do motor, mesmo em baixas rotações (até 150[RPM]). O conjunto formado pelo circuito regulador analógico e o circuito microprocessado de disparo mostrou ser funcional. Possibilitando o controle da velocidade do motor através de um simples potenciômetro dentro de uma faixa de 150 a 1200[RPM]. O sistema permite com um simples clicar de um botão partir e parar o motor.

A utilização do microcomputador permitiu uma maior flexibilidade no desenvolvimento do sistema. O software implementado permite uma melhor interação do operador através de sua interface gráfica.

Embora neste trabalho o controle de disparo da ponte inversora tenha sido implementado em linguagem C para rodar em um microcomputador utilizando o ambiente operacional MS-DOS, este foi concebido para permitir, com as devidas modificações, a possibilidade de ser implementado em microcontroladores. Pois o avanço na tecnologia dos mesmos permitiu que tornassem cada vez mais compactos e velozes. Acompanhando este desenvolvimento houve uma significativa redução de preço. Estes fatores aliados à facilidade de manutenção e a redução do número de componentes do circuito, fazem com que os circuitos microcontrolados se tornem uma solução bastante atraente.

O trabalho desenvolvido pode ser considerado um primeiro protótipo, portanto, sugere-se as seguintes melhorias para futuras dissertações:

1. Implementação digital do regulador de velocidade, sendo possível como alternativa aos reguladores PID a utilização de lógica Fuzzy;
2. Implementação da operação do sistema em quatro quadrantes;
3. Implementação de um método para estimar a posição inicial do rotor, o que evitará possíveis oscilações durante a partida;
4. Eliminação completa do tacogerador, estimando a velocidade e usando somente o método “sensorless”;
5. Implementação do sistema utilizando microcontroladores, para redução de custo e compactação do circuito final.

Anexo 1 – Circuitos Eletrônicos

A1.1 - TCA-780

O circuito de disparo da ponte retificadora possui como componente principal o integrado TCA-780 (Icotron/Siemens). Este componente possibilita a redução no número de componentes total do circuito.

Este CI é indicado para controle de tiristores, triacs e transistores. O ângulo dos pulsos de gatilho pode variar de 0° a 180°. Algumas de suas características são:

- Consumo interno de apenas 5 mA;
- Lógica digital altamente imune a ruídos;
- Necessidade de apenas três TCA 780 em sistema trifásico;
- Duração dos pulsos de saída determinada pela colocação de um capacitor externo ;
- Saída de tensão regulada em 3,1 V ;
- Possibilidades de inibição simultânea de todas as saídas;

As figuras A1.1 e A1.2 mostram, respectivamente, a pinagem e o diagrama interno do TCA-780.

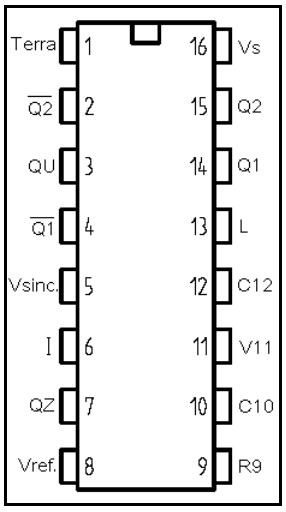

Figura A1.1 - Pinagem do TCA-780 (Vista Superior).

As funções dos pinos são:

01 – Terra.

02 – Saída complementar do pino 15, em coletor aberto.

03 – Saída de pulso positivo, em coletor aberto.

04 – Saída complementar do pino 14, em coletor aberto.

05 – Entrada de sincronismo (diodos em antiparalelo).

06 – Inibe todas as saídas (quando aterrada).

07 – Saída em coletor aberto para acionar TRIACS.

08 – Fornece 3,1 V estabilizado.

09 – Potenciômetro de ajuste da rampa ($20 < R9 < 500k\Omega$).

10 – Capacitor de formação da rampa ($C10 \leq 0{,}5\ \mu F$).

11 – Entrada da tensão de controle (nível CC).

12 – Controla a largura dos pulsos das saídas 14 e 15.

13 – Controla a largura dos pulsos das saídas 02 e 04.

14 – Saída de pulso positivo no semiciclo positivo.

15 – Saída de pulso positivo no semiciclo negativo.

16 – Alimentação CC.

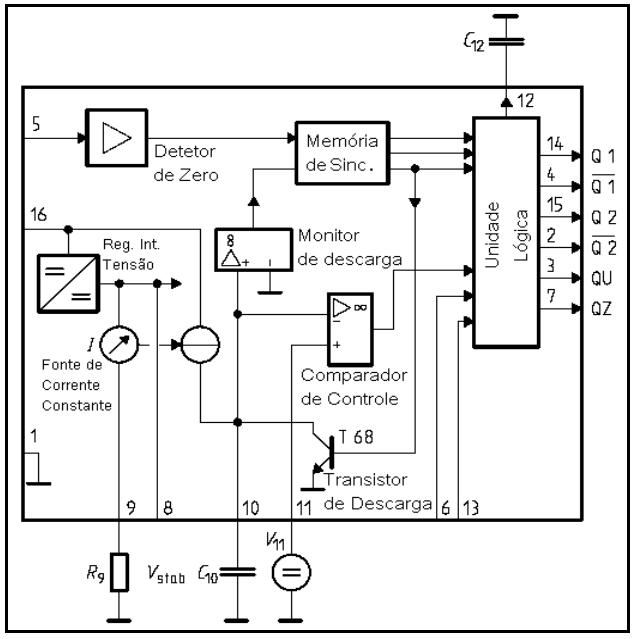

Figura A1.2 - Diagrama interno do TCA-780.

O sinal de sincronismo é obtido através do pino 16. Um detector de zero identifica a passagem pelo zero e transfere para o registrador de sincronismo. Este controla o gerador

de rampa, o capacitor C10, o mesmo é carregado por uma corrente constante, determinada por R9. Esta tensão vai a zero sempre que a tensão de sincronismo passa por zero. A combinação R9 e C10 determina a inclinação da rampa.

Se a tensão da rampa (V10) superar a tensão de controle (V11) a unidade lógica envia pulsos para a saída. Obtendo-se, V15, no semiciclo positivo da tensão de sincronismo, e V14, no semiciclo negativo da tensão de sincronismo.

A largura destes pulsos é de 30[μs]. Podendo ser prolongada pela adição de um capacitor externo, C12, entre o pino 12 e o pino terra. Na tabela A1.1 pode ser vista a relação entre a capacitância e a largura dos pulsos. Os pinos 2 e 4 são as saídas complementares dos pinos 14 e 15, respectivamente, em coletor aberto. Todas as saídas do TCA-780 podem ser inibidas aterrando o pino 6. A figura A1.3 mostra as formas de onda para o TCA-780.

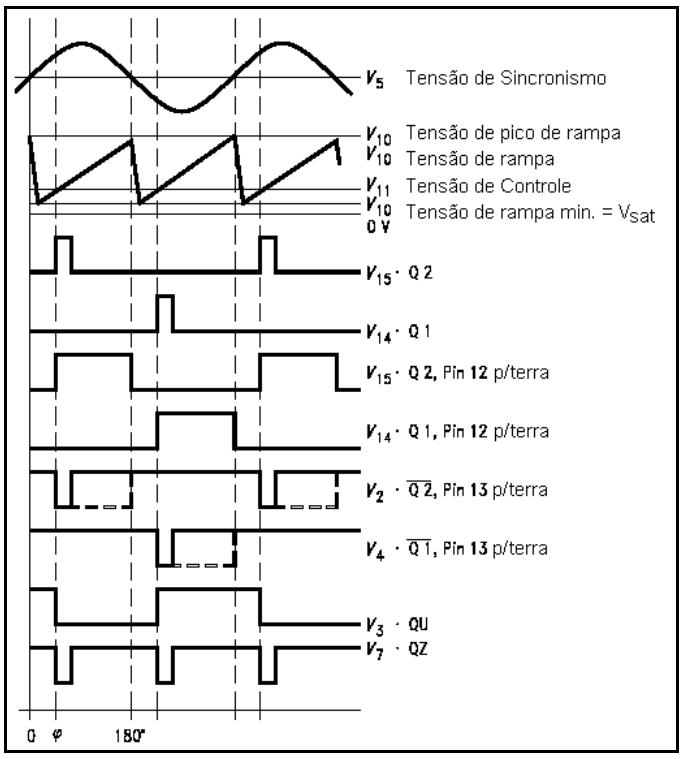

Figura A1.3 - Formas de onda do TCA 780

C12 [pF]	100	220	330	680	1000
Duração dos pulsos [ms]	0,080	0,130	0,200	0,370	0,550

Tabela A1.1 - Larguras dos pulsos dos pinos 14 e 15 para valores determinados de C12

Formulário básico [8]

Corrente de carga do capacitor C10:

$$I10 = \frac{Vref \times K}{R9} \qquad (1)$$

Tensão da rampa:

$$V10 = \frac{Vref \times K \times \Delta t}{R9 \times C10} \qquad (2)$$

Ponto de disparo:

$$tz = \frac{R9 \times C10}{Vref \times K} \times V11 \qquad (3)$$

$$\alpha = \frac{V11}{V10} \times 180° \qquad (4)$$

Largura de pulso:

$$tp = 30\mu s \text{ sem C12}$$

$$tp = \frac{430\mu s}{nF} \text{ com C12} \qquad (5)$$

Onde :

Vref = V8 = 3,1V ;

K = 1,25 ;

C10 ≤ 0,5 μF ;

25 kΩ < R9 < 500 KΩ .

A1.2 - C.I. 555

Com C.I. 555 é possível produzir sinais pulsados, operando como monoastável a duração do pulso pode ser controlada por um resistor e um capacitor externos. Este está sendo utilizado neste circuito para ampliar a largura do pulso que saí do TCA-780 para um pulso de 120°.

A figura A1.4 mostra o diagrama interno do 555. A figura A1.5 mostra a configuração do 555 como monoastável. Na figura A1.6 podem ser vistas as formas de onda para esta configuração.

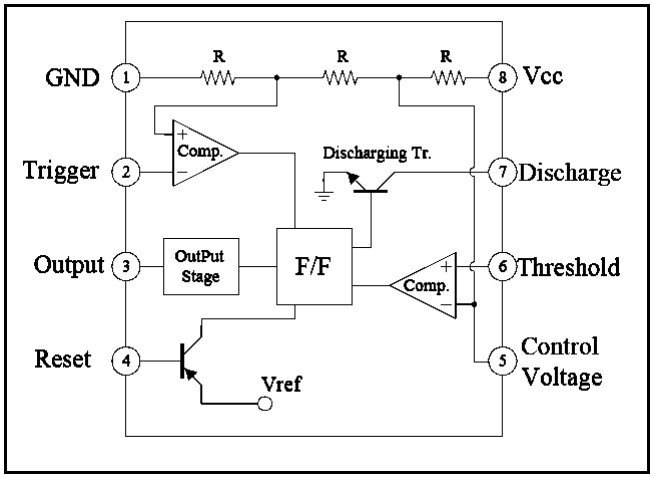

Figura A1.4 - Diagrama interno do 555.

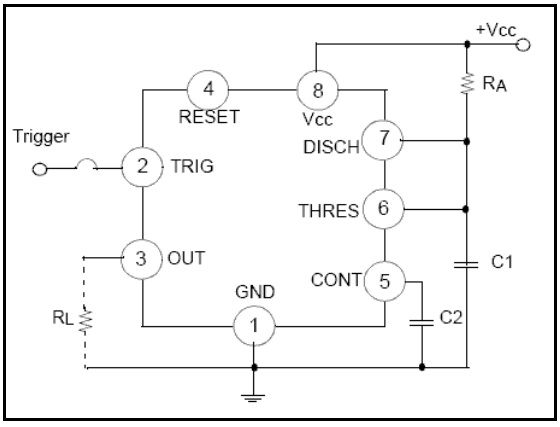

Figura A1.5 - 555 como monoastável.

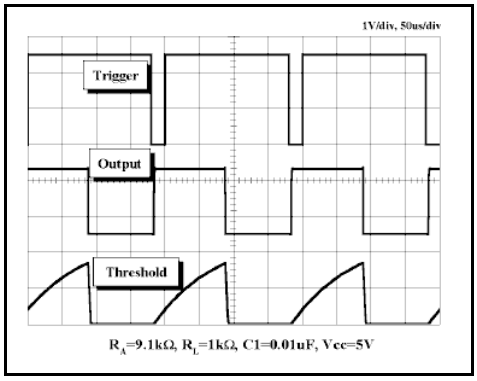

Figura A1.6 - formas de onda para o 555 como monoastável.

A1.3 - TIL 113

O TIL113 é um optoacoplador. Este é composto por um LED e um fototransistor NPN Darlington. O diagrama interno do TIL113 pode ser visto na figura A1.7.

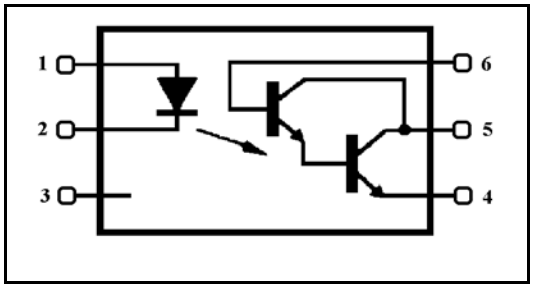

Figura A1.7- Diagrama interno do TIL113.

O LED e o fototransistor do TIL113 devem ser alimentados por fontes de tensão independentes.

A1.4 - Drive

O objetivo deste circuito é amplificar a corrente do pulso de saída do TIL113, até o valor necessário para disparar o gate do tiristor e também impedir que uma tensão negativa seja aplicada na junção gatilho – catodo. O circuito é composto por um transistor BC 548, funcionando como chave, e em sua base há um capacitor "speed up" com a função de acelerar o chaveamento do transistor; um BD 135, configurado como seguidor de emissor. O potenciômetro na sua base permite o ajuste da tensão no gate. O circuito é acoplado ao gate do tiristor através de um diodo, o qual impede que tensões negativas cheguem ao gate. A figura A1.8 ilustra o circuito do driver.

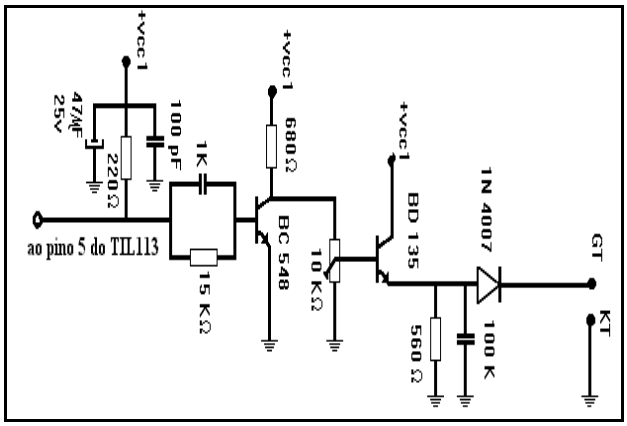

Figura A1.8 - Circuito do driver.

A figura A1.9 mostra o diagrama completo do circuito de disparo da ponte retificadora.

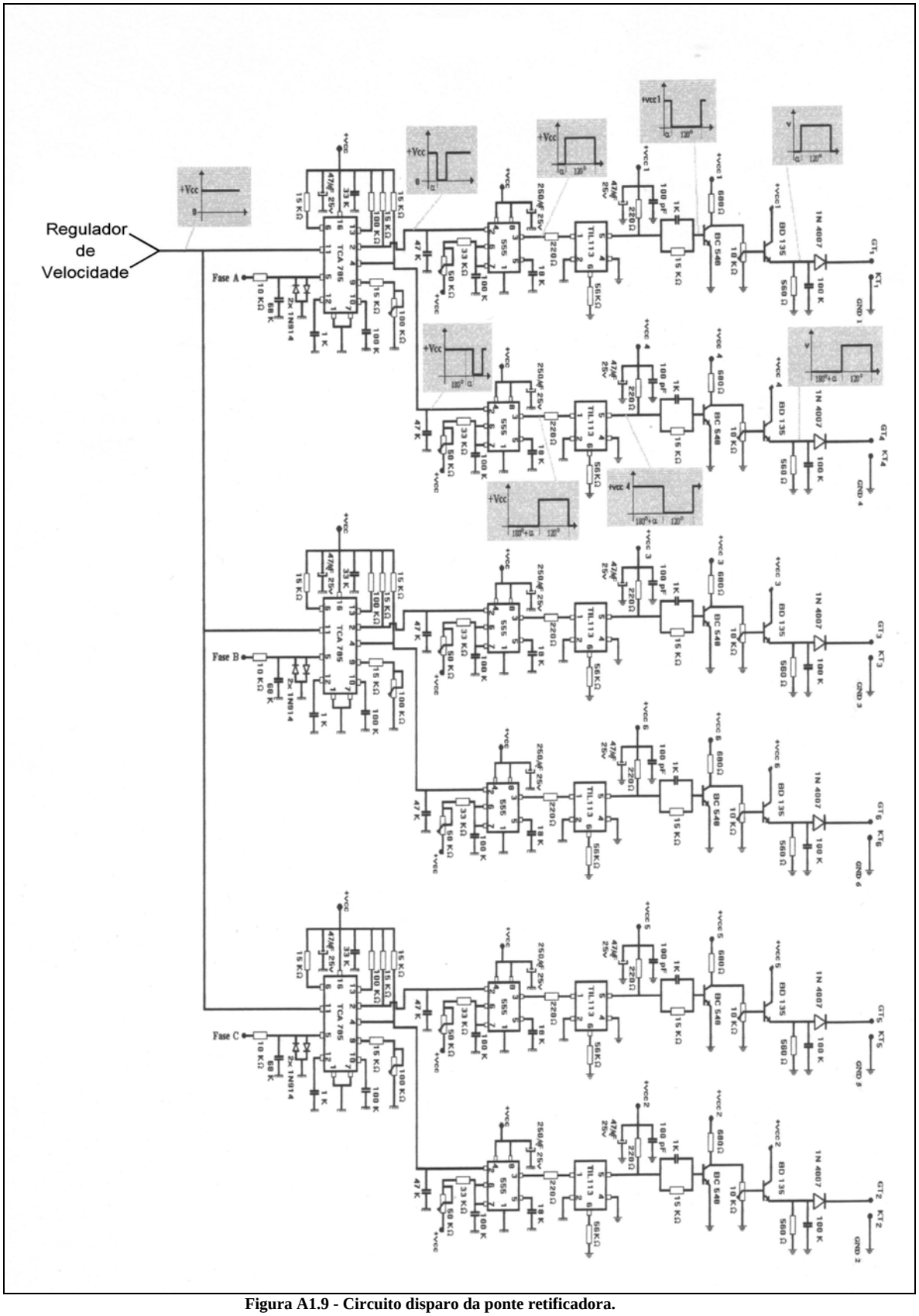

Figura A1.9 - Circuito disparo da ponte retificadora.

Anexo 2 - Código Fonte do Programa

A2.1 – Rotina Principal – main.c

```
/*---------------------------------------------------------------------------
                    UNIFEI - UNIVERSIDADE FEDERAL DE ITAJUBA
                    ----------------------------------------
GEMEI - GRUPO DE ESTUDOS EM MANUTENCAO E INSTALACOES ELETRICAS
                    ----------------------------------------------------------

                    Controlador Digital Para Ponte Inversora Controlada
                    Por Tensao De Um Acionamento De Maquina Sincrona Controlado
                    Por Injecao De Corrente.

                    Desenvolvido por Eben-Ezer P. Silveira
                    Orientador: prof. Dr. Angelo Jose Junqueira Rezek.
                    Co-orientador: prof. Dr. Valberto Ferreira da Silva.

                    Agosto de 2004
                    ---------------------------------------------------------------------------
                    */

#include <math.h>
#include "public.h"
#include "driver.hx"
#include "timer.hx"
#include "zeroc.hx"
#include "graficos.hx"

//---------------------------------------------------------------------------
//Ajuste do tamanho da pilha.
extern unsigned _stklen  =  16000;
extern unsigned _heaplen =  64000;

// ---------------------------------------------------------------------------
// Variaveis Globais..
  int      resolucao = 2; //Resolucao da tela 2=640x480x256.
  X_jan    JM;           //Janela principal.
  X_jan    J[4];         //Janelas filhas.
  X_but    bts[17];      //botoes utilizados.
  X_but    mbts[3];      //botoes exclusivos da janela de aviso.
  X_rect   dados_1[12];  //Caixa das variaveis na tela.
  X_rect   Graf;         //Caixa do grafico.
  float    Valores1[3]={0.0, 0.0, 0.0}; //Valores informados.
  byte     Valores2[3]={0, 0, 0};       //Valores informados.
  int      btnum=16; //Numero de botoes utilizados.
  int      eidx=-1; //Indice do botao de parametro selecionado.

  DWORD    tbloqpart=0;
  byte     bloqpartf=0;
```

```
// ---------------------------------------------------------------------------
// Prototipo das funcoes.

void DesJan(void);
void teste_btn(void);
void dados_aquisicao(void);
void parametros(void);
void draw_graf(X_rect W, float *rx, float *ry, byte corf);
void des_graf(float ox,  float oy,
              float vx,  float vy,
              float rx,  float ry,
              byte corl, byte corf,
              byte limpa);
void teclado(int codt);
void des_TempDisp(int eidx);
void des_Valores1(int eidx);
void des_Valores2(int eidx);
void des_OpSel(int op);
void des_aviso(char *info);
void BloqPulsos(void);

//---------------------------------------------------------------------------

//Rotina principal.
int main(void)
{
 //Definicao das variaveis locais.
 float   RelVF  = 0.113; //Relacao entre Va e a freq. (experimental)
 float   VfPart = 0.50;  //Valor de tensao minimo para partida.
 WORD    VComut = 400;   //Velocidade max. para comutacao forcada.
 int     tec=0; //Codigo da tecla pressionada.
 //Variaveis auxiliares.
 DWORD   ulaux, trbloq    = 0;
 WORD    i, k, j, r, sair = 0;
 float   fm      = 0.0, faux;
 float   VfAm[3] = {0.0,0.0,0.0}, tref = 0.0;
 WORD    kvm=0, mxkvm=VMedSampPart;
 float   fVpmed[VMedSampPart+1], fVpm  = 0.0;
 X_rect  mb;
 char    str1[50], str2[50];
 byte    mba=0, clique=0;
 X_rect  graf;
 //X_rect  figura;
 float   grafrx, grafry;
 float   ox=0, oy=0, vx, vy, rx;
 DWORD   gcount=0, gcmax;
 DWORD   tdisplay =   0;
 int     intdisp  = 100;
 int     ACFlag   =   0;

 // ---------------------------------------------------------------------------
 //Muda a tela para o modo grafico.
 if (!modo_grafico(resolucao)) {
   printf("\nErro ao acessar o modo grafico...\n");
   exit(0);   }
```

```
// ---------------------------------------------------------------------------
board_default(); //Define a placa padrao.
init_AD();       //Inicia conversor analogico/digital.
setport2(0x00);  //Muda para o modo digital.
BloqPulsos();    //Bloqueia os pulsos.

// ---------------------------------------------------------------------------
// Desenha as janelas...
DesJan();
teste_btn();
dados_aquisicao();
parametros();
draw_graf(J[1].carea, &grafrx, &grafry, 102);

// ---------------------------------------------------------------------------
//Altera a interrupcao do relogio do DOS para a interrupcao de aquisicao.
Def_Interrupt = (fptr) _dos_getvect( 0x08 );
DefIrqFreq(2000);
_dos_setvect( 0x08, AD_Interrupt );

// ---------------------------------------------------------------------------
//Ativa o mouse.
mouse();
minit(100, 100, getmaxx(), getmaxy());
mshow();
smouse_btn.bt=0;
memset(mbts, 0, sizeof(mbts));
mCursor();

// ---------------------------------------------------------------------------
//Exibe os valores das variaveis na tela.
for (k=0;k<4;k++) des_TempDisp(k);
for (k=0;k<3;k++) des_Valores1(k);
for (k=0;k<3;k++) des_Valores2(k);

des_OpSel(eidx+1);

tdisplay = lContTmp;
rx       = (float)intdisp/(float)iSampFreq;
gcmax    = (DWORD)(10.0/rx);

// ---------------------------------------------------------------------------
//"Loop" principal...
do {

  mCursor();

  //Calcula as frequencias e utiliza o menor valor.
  fm   = 1.0e6;
  faux = 0.0;
  for (i=0;i<6;i++)
  {
   tref = (float)PulsePT[i]/(float)iSampFreq;
   if (tref>1.0e-5) faux=(1.0/tref);
   if (faux<fm) fm=faux;
  }
```

```
//Calcula o valor rms das tensoes de fase.
for (i=0;i<3;i++)
  if (!fVfase[i]&&kVfase[i]>0)
  {
    //Salva as tensoes aquisitadas em um arquivo...
    if (SalVabc) for (k=0;k<kVfase[i];k++) SalvaAD(Vfase[i][k], i);
    faux = VFase_rms((float *)Vfase[i], kVfase[i]);
    if (faux>=0.0) VfAm[i] = faux;
    fVfase[i]=1;
    kVfase[i]=0;
  }

//Valor medio das tensoes.
Valores1[2] = (VfAm[0]+VfAm[1]+VfAm[2])/3;

//Calcula a vel. do motor.
if (Valores1[2]>1.5) Valores1[0] = fm;
else if (fm>0) Valores1[0] = Valores1[2]/RelVF;

//Calcula a velocidade 1800rpm = 60Hz.
Valores1[1] = Valores1[0]*30.0;

//Define parametros para Comutacao forcada e para os filtros.
if (Valores1[1]<VComut)
{
  if (DigCtrl)
  {
    CFFlag = 1;
    MinChngTime = 30;
  }
  else if (!ACFlag)
  {
    ACFlag = 1;
    setport2(ANALOGKEY|ANALOGPULSE);
  }
}
else if (CFFlag||ACFlag)
{
  CFFlag      = 0;
  ACFlag      = 0;
  MinChngTime = 0;
  if (!DigCtrl) setport2(ANALOGKEY);
}

//Rotina de partida.
if (iModoTeste&&!bloqpartf)
{
  if (!Enabled)
  {
    //Calcula a velocidade media.
    fVpmed[kvm++] = Valores1[2];
    if (kvm>mxkvm)
    {
          fVpm = 0.0;
          for (i=0;i<kvm;i++)
            fVpm += fVpmed[i];
```

```
            fVpm /= kvm;
            kvm=0;
    }

    //Se a tensao media for maior do que a tensao de partida,
    //ativa a geracao de pulsos e desliga o modo de partida.
    if (fVpm>VfPart&&DigCtrl)
    {
          iModoTeste = 0; //Desabilita o modo de teste.
          Enabled = 1;
          //Informa alguns dados da partida.
          sprintf(str1, "Vel. Partida: %2.2f[rpm]", fVpm/RelVF*30.0);
          des_aviso(str1);
    }

  }
  else
  {
    iModoTeste = 0; //Desliga a partida.
    fVpm  = 0.0;    //Zera a velocidade media de partida.
  }
}
else
{
  //Verifica se o bloqueio de partida esta ativo.
  if (bloqpartf)
  {
    ulaux = labs(lContTmp-tbloqpart);
    if (ulaux>TmpPartBloq)
    {
          trbloq = 0;
          bloqpartf=0;
          tbloqpart=0;
          des_aviso(" ");
    }
    else if (!(ulaux%2)) trbloq = (int)((TmpPartBloq-ulaux)/iSampFreq);
  }
}

if (kbhit())
{
   tec = getch();
   if (!tec) tec = getch();
   teclado(tec);
}

if (smouse_btn.bt&&!clique)
{

   smouse_btn.bt = 0; clique = 1;
```

```
for (k=0; k<btnum; k++)
{
      r = fn_button_click(bts[k], smouse_btn.x, smouse_btn.y);
      if (r) switch (k)
      {
        case 0: if (!mba)
                  {
                    strcpy(str1, "Confirma a saida?");
                    strcpy(str2, "Sim/Nao");
                    msgboxdim  (&mb, str1, str2);
                    salva_rect(mb);
                    messagebox (str1, str2, 1, mbts);
                    mba=1;
                  }
                  break;

        case 1: case 2: case 3:
        case 4: case 5: case 6:
                  if (!Enabled&&!iModoTeste) DefPosicao(k-1);
                  break;

        case 7: if (!Enabled&&!iModoTeste)
                  for (j=0;j<2;j++)
                    for (i=0;i<6;i++)
                    {
                      setport(PsSeq[i], 1);
                      wait(100);
                      setport(0, 0);
                      wait(2);
                     }
                  break;

        case 8: DigCtrl=1; //Ativa o modo uProcessado.
                  setport2(0x00);
                  break;

        case 9: if (Valores1[0]>10) //Ativa o modo nao uProcessado, somente, apos uma determinada vel.
                  {
                    Enabled = 0;
                    DigCtrl = 0;
                    setport2(ANALOGKEY|ANALOGPULSE);
                  }
                  break;

        case 10: if (!bloqpartf) if (!Enabled&&!iModoTeste)
                   {
                     DefPosicao(4); wait(10);
                     DefPosicao(5); wait(20);
                     PartdIdx=1; PartIdx=0; SeqDir=1;
                     des_aviso("Partida ativada...");
                     fVpm=0; kvm=0;
                     PartdT1=0; PartdT2=0;
                     PartdT=lContTmp;
                     iModoTeste = 1; //Habilita o modo de teste.
                   }
                   break;
```

```
            case 11: if (Enabled||iModoTeste)
                     {
                       Enabled=0;
                       iModoTeste=0;
                       BloqPulsos();
                       if (!bloqpartf)
                       {
                            tbloqpart=lContTmp;
                            bloqpartf=1;
                       }
                       des_aviso("Desligado...");
                     }
                     break;

            case 12: case 13:
            case 14: case 15: eidx = k-12;
                              des_OpSel(eidx+1);
                     break;
          }
    }

    if (mba)
    {
          if (fn_button_click(mbts[0], smouse_btn.x, smouse_btn.y)) sair=1;
          if (fn_button_click(mbts[1], smouse_btn.x, smouse_btn.y))
          {
            memset(mbts, 0, sizeof(mbts));
            carrega_rect(mb);
            mba=0;
          }
    }

  }
  else
  {
    if (clique&&!smouse_btn.bt) clique = 0;
  }

 //Atualiza a tela
 if (tempod(tdisplay, intdisp)&&!mba&&!iModoTeste)
 {
   //Outras informacoes...
   Valores2[0] = CFFlag;
   Valores2[1] = DigCtrl;
   Valores2[2] = Enabled;

   for (k=0;k<3;k++) des_Valores1(k);
   for (k=0;k<3;k++) des_Valores2(k);
   tdisplay = lContTmp;

   if (trbloq) {
     sprintf(str1, "Temp. Bloq. da partida: %d[s]", trbloq);
     des_aviso(str1);
   }
```

```
    if (gcount<gcmax)
    {
      vx = (float)gcount*rx;
      vy = Valores1[1];
      des_graf (ox, oy, vx, vy, grafrx, grafry, 0, 102, 0);
      ox = vx;
      oy = vy;
      gcount++;
    }
    else
    {
      gcount=0;
      vx = 0; vy = 0;
      ox = 0; oy = 0;
      des_graf (ox, oy, vx, vy, grafrx, grafry, 102, 102, 1);
    }

   }

 } while(tec!=27&&!sair);

 //Encerrando...
 DefIrqFreq( 18 ); //Restaura a frequencia original.
 _dos_setvect( 0x08, Def_Interrupt ); //Restaura a interrupcao original.
 DigCtrl=1; //Ativa o controle digital.
 BloqPulsos(); //Desliga os pulsos.
 FechaAD();

 return 0;

 //FIM...

}

// ---------------------------------------------------------------------------
// Funcoes...

// ---------------------------------------------------------------------------
//Desenha a janela principal
void DesJan(void)
{
  int jx1, jy1, jx2, jy2, dyle=20;

  strcpy(JM.lines[0], "GEMEI - UNIFEI / 2004");
  strcpy(JM.lines[1], "Eben-Ezer Prates da Silveira.");
  strcpy(JM.lines[2], "Orientador: prof. Dr. Angelo Jose Junqueira Rezek.");
  strcpy(JM.lines[3], "Co-orientador: prof. Dr. Valberto Ferreira da Silva.");

  JM.linesnum=4;

  win_janelac(&JM, 0, 0, getmaxx(), getmaxy(), 6,
              "Controle de Disparo da Ponte Inversora.");
  win_janelas(JM, 6, JM.name);

  //Botao para encerrar o programa.
  bts[0] = fn_button_setc(JM.win.x2-21, JM.win.y1+3, 32|BT_ATIV);
```

```
  jx1 = JM.carea.x1+2;
  jy1 = JM.carea.y1+2;

  jx2 = (int)(((float)JM.carea.x2/2.0)-30.0);
  jy2 = (int)(((float)JM.carea.y2)/2.0)+30.0;

  win_janelac(&J[0], jx1, jy1, jx2, jy2+dyle, 8,
              "Dados do Aplicativo.");
  win_janelas(J[0], 8, J[0].name);

  win_janelac(&J[1], jx2+2, jy1, JM.carea.x2-2, jy2, 8,
              "Velocidade do Motor.");
  win_janelas(J[1], 8, J[1].name);

  win_janelac(&J[2], jx1, jy2+dyle+2, jx2, JM.carea.y2-2, 8,
              "Rotinas de Teste.");
  win_janelas(J[2], 8, J[2].name);

  win_janelac(&J[3], jx2+2, jy2+2, JM.carea.x2-2, JM.carea.y2-2, 8,
              "Paramentros de Controle.");
  win_janelas(J[3], 8, J[3].name);

}

// ----------------------------------------------------------------------
//Desenha o grafico da velocidade.
void draw_graf(X_rect W, float *rx, float *ry, byte corf)
{
    int  x, y, wx, wy;
    int  c, i, dc, ngrad=10;
    int  dx=50, dy=25, dx1=-40, dy1=-20;
    float gx, gy, j;

    x  = W.x1+dx;
    y  = W.y1+dy;
    wx = W.x2+dx1;
    wy = W.y2+dy1;

    setfillstyle(SOLID_FILL, corf);
    bar(x, y, wx, wy);

    setcolor(110);
    setlinestyle(4,0x5555, 1);
    line (x, y, x, wy);
    line (x, wy, wx, wy);

    gx = (float)(wx-x)/(float)ngrad;
    gy = (float)(wy-y)/(float)ngrad;

    Graf.x1 = x;   Graf.y1 = y;
    Graf.x2 = wx;  Graf.y2 = wy;

    // Eixo Y.
    setlinestyle(0,0xFFFF, 1);
    setcolor(1);
```

```
    says(x-42, y-16, "[rpm]", 1);

    dc  = (int)(1800.0/(float)ngrad);
    *ry = (float)(wy-y)/(float)(dc*gy);

    for (c=0, j=wy, i=wy; i>=y;)
    {
      sayl(x-40, i-4, c, 1);
      line (x-2, i, x, i);
      c+=dc; j -= gy; i = (int)j;
    }

    dc = 1;
    *rx = gx;

    for (c=0,i=x;i<=wx;i+=gx)
    {
      sayl(i-4, wy+8, c, 1);
      line (i, wy+2, i, wy);
      c+=dc;
    }

    says(i-4, wy+8, "[s]", 1);

}

// ---------------------------------------------------------------------------
//Preenche a janela de testes de disparo.
void teste_btn(void)
{
  int  k, dy=30, dx=160;
  char str1[25];
  char str2[25];

  says (J[2].carea.x1+4, J[2].carea.y1+10,
          "Disparo manual dos Tiristores.", 4);

  //Botoes para testar o disparo dos tiristores.
  for (k=1; k<7; k++)
  {
    strcpy(str2,"Disparar ");
    itoa(k, &str2[9], 10);
    bts[k] = fn_button_sets(J[2].carea.x1+dx,
                            J[2].carea.y1+(k-1)*20+dy,
                            32|BT_ATIV,
                            str2);
  }

  //Botoes para testar o disparo dos tiristores.
  strcpy(str2,"Disparar Todos");
  bts[7] = fn_button_sets(J[2].carea.x1+20,
                          J[2].carea.y1+dy+50,
                          32|BT_ATIV,
                          str2);
```

```
}

// ---------------------------------------------------------------------------
// Preenche a janela de dados do sistema.
void dados_aquisicao(void)
{
  int  dy=14, dx=4, sdy=22;
  char str1[40];
  //char str2[25];

  //.....................................................
  strcpy(str1, "Freq. dos disparos:");
  says (J[0].carea.x1+dx, J[0].carea.y1+dy, str1, 4);

  dados_1[0].x1 = J[0].carea.x1+dx+strlen(str1)*8;
  dados_1[0].x2 = J[0].carea.x1+dx+strlen(str1)*8+60;
  dados_1[0].y1 = J[0].carea.y1+dy-4;
  dados_1[0].y2 = dados_1[0].y1+14;
  shrectr (dados_1[0], 1, 1);

  says (dados_1[0].x2+2, J[0].carea.y1+dy, "[Hz]", 4);

  //.....................................................
  strcpy(str1, "Velocidade:");
  says (J[0].carea.x1+dx, J[0].carea.y1+dy+sdy,
            str1, 4);
  dados_1[1].x1 = J[0].carea.x1+dx+strlen(str1)*8;
  dados_1[1].x2 = J[0].carea.x1+dx+strlen(str1)*8+70;
  dados_1[1].y1 = J[0].carea.y1+dy+sdy-4;
  dados_1[1].y2 = dados_1[1].y1+14;
  shrectr (dados_1[1], 1, 1);

  says (dados_1[1].x2+2, J[0].carea.y1+dy+sdy, "[rpm]", 4);

  //.....................................................
  strcpy(str1, "Vmed. A/D:");
  says (J[0].carea.x1+dx, J[0].carea.y1+dy+(sdy*2),
            str1, 4);
  dados_1[2].x1 = J[0].carea.x1+dx+strlen(str1)*8;
  dados_1[2].x2 = J[0].carea.x1+dx+strlen(str1)*8+60;
  dados_1[2].y1 = J[0].carea.y1+dy+(sdy*2)-4;
  dados_1[2].y2 = dados_1[2].y1+14;
  shrectr (dados_1[2], 1, 1);

  says (dados_1[2].x2+2, J[0].carea.y1+dy+(sdy*2), "[V]", 4);

  //.....................................................
  strcpy(str1, "Comutacao Forcada:");
  says (J[0].carea.x1+dx, J[0].carea.y1+dy+(sdy*3),
            str1, 4);
  dados_1[3].x1 = J[0].carea.x1+dx+strlen(str1)*8;
  dados_1[3].x2 = J[0].carea.x1+dx+strlen(str1)*8+80;
  dados_1[3].y1 = J[0].carea.y1+dy+(sdy*3)-4;
  dados_1[3].y2 = dados_1[3].y1+14;
  shrectr (dados_1[3], 1, 1);
```

```
  //.....................................................
  strcpy(str1, "Modo:");
  says (J[0].carea.x1+dx, J[0].carea.y1+dy+(sdy*4),
            str1, 4);
  dados_1[4].x1 = J[0].carea.x1+dx+strlen(str1)*8;
  dados_1[4].x2 = J[0].carea.x1+dx+strlen(str1)*8+160;
  dados_1[4].y1 = J[0].carea.y1+dy+(sdy*4)-4;
  dados_1[4].y2 = dados_1[4].y1+14;
  shrectr (dados_1[4], 1, 1);

  //.....................................................
  strcpy(str1, "Estado dos Pulsos:");
  says (J[0].carea.x1+dx, J[0].carea.y1+dy+(sdy*5),
            str1, 4);
  dados_1[5].x1 = J[0].carea.x1+dx+strlen(str1)*8;
  dados_1[5].x2 = J[0].carea.x1+dx+strlen(str1)*8+90;
  dados_1[5].y1 = J[0].carea.y1+dy+(sdy*5)-4;
  dados_1[5].y2 = dados_1[5].y1+14;
  shrectr (dados_1[5], 1, 1);

  //Caixa de informacoes...
  dados_1[11].x1 = J[0].carea.x1+dx;
  dados_1[11].x2 = J[0].carea.x2-dx;
  dados_1[11].y1 = J[0].carea.y1+dy+(sdy*6);
  dados_1[11].y2 = dados_1[11].y1+8;

}

// ---------------------------------------------------------------------------
// Preenche a janela de paramentros do sistema.
void parametros(void)
{
  int  dy=10, dx=28, sdy=18;
  char str1[40];
  //char str2[25];

  shrect(J[3].carea.x1+2, J[3].carea.y1+2,
             J[3].carea.x2-2, J[3].carea.y1+dy+(sdy*5),1 , 1);

  //.....................................................
  strcpy(str1, "Cte de partida [T1]:");
  says (J[3].carea.x1+dx, J[3].carea.y1+dy,
            str1, 4);
  dados_1[6].x1 = J[3].carea.x1+dx+strlen(str1)*8;
  dados_1[6].x2 = J[3].carea.x1+dx+strlen(str1)*8+50;
  dados_1[6].y1 = J[3].carea.y1+dy-4;
  dados_1[6].y2 = dados_1[6].y1+14;
  shrectr (dados_1[6], 1, 1);
  says (dados_1[6].x2+2, J[3].carea.y1+dy, "[ms]", 4);

  bts[12] = fn_button_sets(J[3].carea.x1+4, J[3].carea.y1+dy-5,
                                    32|BT_ATIV, "1>" );

  //.....................................................
  strcpy(str1, "Cte de partida [T2]:");
  says (J[3].carea.x1+dx, J[3].carea.y1+dy+sdy, str1, 4);
```

```
dados_1[7].x1 = J[3].carea.x1+dx+strlen(str1)*8;
dados_1[7].x2 = J[3].carea.x1+dx+strlen(str1)*8+50;
dados_1[7].y1 = J[3].carea.y1+dy-4+sdy;
dados_1[7].y2 = dados_1[7].y1+14;
shrectr (dados_1[7], 1, 1);
says (dados_1[7].x2+2, J[3].carea.y1+dy+sdy, "[ms]", 4);

bts[13] = fn_button_sets(J[3].carea.x1+4, J[3].carea.y1+dy-5+sdy,
                                32|BT_ATIV, "2>" );

//.....................................................
strcpy(str1, "Cte de partida [T3]:");
says (J[3].carea.x1+dx, J[3].carea.y1+dy+sdy*2,
        str1, 4);
dados_1[8].x1 = J[3].carea.x1+dx+strlen(str1)*8;
dados_1[8].x2 = J[3].carea.x1+dx+strlen(str1)*8+50;
dados_1[8].y1 = J[3].carea.y1+dy-4+sdy*2;
dados_1[8].y2 = dados_1[8].y1+14;
shrectr (dados_1[8], 1, 1);
says (dados_1[8].x2+2, J[3].carea.y1+dy+sdy*2, "[ms]", 4);

bts[14] = fn_button_sets(J[3].carea.x1+4, J[3].carea.y1+dy-5+sdy*2,
                                32|BT_ATIV, "3>" );

//.....................................................
strcpy(str1, "Vel.max.comut.forcada:");
says (J[3].carea.x1+dx, J[3].carea.y1+dy+(sdy*3),
        str1, 4);

dados_1[9].x1 = J[3].carea.x1+dx+strlen(str1)*8;
dados_1[9].x2 = J[3].carea.x1+dx+strlen(str1)*8+50;
dados_1[9].y1 = J[3].carea.y1+dy-4+(sdy*3);
dados_1[9].y2 = dados_1[9].y1+14;
shrectr (dados_1[9], 1, 1);

says (dados_1[9].x2+2, J[3].carea.y1+dy+(sdy*3), "[rpm]", 4);

bts[15] = fn_button_sets(J[3].carea.x1+4, J[3].carea.y1+dy-5+(sdy*3),
                                32|BT_ATIV, "4>" );

//.....................................................
strcpy(str1, "Opcao selecionada:    ");
says (J[3].carea.x1+dx, J[3].carea.y1+dy+(sdy*4),
        str1, 4);
dados_1[10].x1 = J[3].carea.x1+dx+strlen(str1)*8;
dados_1[10].x2 = J[3].carea.x1+dx+strlen(str1)*8+70;
dados_1[10].y1 = J[3].carea.y1+dy-4+(sdy*4);
dados_1[10].y2 = dados_1[10].y1+14;
shrectr (dados_1[10], 1, 1);

//Botoes para ativar ou desativar o controle.
dx=60; dy=106;

strcpy(str1, " Modo Micro ");
bts[8] = fn_button_sets(J[3].carea.x1+dx,
                                J[3].carea.y1+dy,
```

```
                                32|BT_ATIV,
                                str1);

  strcpy(str1, "Modo Normal");
  bts[9] = fn_button_sets(J[3].carea.x1+dx+110,
                                J[3].carea.y1+dy,
                                32|BT_ATIV,
                                str1);

  dx=26; dy=134;

  strcpy(str1, "  Partir Motor  ");
  bts[10] = fn_button_sets(J[3].carea.x1+dx,
                                J[3].carea.y1+dy,
                                64|32|BT_ATIV,
                                str1);

  strcpy(str1, "   [D]esligar   ");
  bts[11] = fn_button_sets(J[3].carea.x1+dx+150,
                                J[3].carea.y1+dy,
                                64|32|BT_ATIV,
                                str1);

}

// ----------------------------------------------------------------------
//Utilizada para a janela de grafico da velocidade.
void des_graf(float ox,  float oy,
              float vx,  float vy,
              float rx,  float ry,
              byte corl, byte corf,
              byte limpa)
{
    int    x1, y1, x2, y2;
    X_rect Z = Graf;

    if (limpa)
    {
      setfillstyle(SOLID_FILL, corf);
      bar(Z.x1+1, Z.y1, Z.x2, Z.y2-1);
    }

    // Plotando os Dados...
    setlinestyle(4, 0xFFFF, 1);
    setcolor(corl);

    x1 = rx*ox+Z.x1;     y1 = Z.y2-(ry*oy);
    x2 = rx*vx+Z.x1;     y2 = Z.y2-(ry*vy);

    line(x1, y1, x2, y2);

}
```

```
// ---------------------------------------------------------------------------
//Leitura do teclado.
void teclado(int codt)
{
  int k;

  switch(codt)
  {
    case 72:
    case 80: if (eidx>=0&&!iModoTeste)
             {
                TempDisp[eidx] += (codt==72) ? 2:-2;
                des_TempDisp(eidx);
             }
             break;

    case 73:
    case 81: if (eidx>=0&&!iModoTeste)
             {
                TempDisp[eidx] += (codt==73) ? 20:-20;
                des_TempDisp(eidx);
             }
             break;

    case 32: case 'd':
             Enabled=0;
             iModoTeste=0;
             BloqPulsos();
             if (!bloqpartf)
             {
                tbloqpart=lContTmp;
                bloqpartf=1;
             }
             break;

    case 'b': setport(0, 0);
                des_aviso("Modo Bloqueio.");
                break;

    case 'n': setport(0, 1);
                des_aviso("Modo Normal.");
                break;

    case 'l': if (Enabled)
                {
                  Enabled = 0;
                  BloqPulsos();
                  if (!bloqpartf) { tbloqpart=lContTmp; bloqpartf=1; }
                }
                else if (DigCtrl)
                {
                  iModoTeste = 0;
                  Enabled = 1;
                }
                break;
```

```
    case 'p': savepcx("tela.pcx", 0, 0, max_x-1, max_y);
                    break;

    case 'w': SalVabc=0;
                    des_aviso("Arquivo fechado...");
                    break;

    case 'q': FechaAD();
                    for (k=0;k<3;k++) adfout[k]=NULL;
                    SalVabc=AbreAD("vdata");
                    des_aviso("Salvando...");
                    break;

    default: if (((codt-48)>=1)&&((codt-48)<=6)) DefPosicao(codt-49);
                 break;
  }

}

// ---------------------------------------------------------------------
// Utilizada para atualizar os dados na tela.
void des_TempDisp(int eidx)
{
 char str1[30];

 if (eidx<3) itoa(TempDisp[eidx]/2, str1, 10);
 else itoa(TempDisp[eidx], str1, 10);
 eidx+=6;
 setfillstyle(SOLID_FILL, _winbakhi);
 bar (dados_1[eidx].x1+1, dados_1[eidx].y1+1,
     dados_1[eidx].x2-1, dados_1[eidx].y2-1);
 says(dados_1[eidx].x1+3, dados_1[eidx].y1+3, str1, 1);

}

// ---------------------------------------------------------------------
//Utilizada para atualizar os dados na tela.
void des_Valores1(int eidx)
{
 char str1[30];

 sprintf(str1, "%3.2f", Valores1[eidx]);
 setfillstyle(SOLID_FILL, _winbakhi);
 bar (dados_1[eidx].x1+1, dados_1[eidx].y1+1,
     dados_1[eidx].x2-1, dados_1[eidx].y2-1);
 says(dados_1[eidx].x1+3, dados_1[eidx].y1+3, str1, 1);

}
```

```
// ----------------------------------------------------------------------------
// Utilizada para atualizar os dados na tela.
void des_Valores2(int eidx)
{
  char Lig[3][30]  = {"Ativa","Microprocessado","Ligados"};
  char Desl[3][30] = {"Desligada","Nao microprocessado","Desligados"};
  byte idx=eidx;
  byte estado=Valores2[eidx];

  eidx+=3;

  setfillstyle(SOLID_FILL, _winbakhi);
  bar (dados_1[eidx].x1+1, dados_1[eidx].y1+1,
      dados_1[eidx].x2-1, dados_1[eidx].y2-1);

  if (estado)
    says(dados_1[eidx].x1+3, dados_1[eidx].y1+3, Lig[idx], 1);
  else
    says(dados_1[eidx].x1+3, dados_1[eidx].y1+3, Desl[idx], 1);

}
// ----------------------------------------------------------------------------
// Utilizada para atualizar a selecao de opcoes de ajuste de parametros.
void des_OpSel(int op)
{
  char str1[30];
  byte eidx=10;

  if (op>0) itoa(op, str1, 10);
  else strcpy(str1, "Nenhuma");

  setfillstyle(SOLID_FILL, _winbakhi);
  bar (dados_1[eidx].x1+1, dados_1[eidx].y1+1,
      dados_1[eidx].x2-1, dados_1[eidx].y2-1);
  says(dados_1[eidx].x1+3, dados_1[eidx].y1+3, str1, 1);

}
// ----------------------------------------------------------------------------
// Mostra mensagens na tela.
void des_aviso(char *info)
{
   setfillstyle(SOLID_FILL, _winbakhi);
   bar (dados_1[11].x1, dados_1[11].y1,
            dados_1[11].x2, dados_1[11].y2);
   says(dados_1[11].x1, dados_1[11].y1,
            info, 1);
}

//----------------------------------------------------------------------------
//Funcao de bloqueio dos pulsos.
void BloqPulsos(void)
{
   setport(0, 0); //Modo bloqueado.
   wait(100);
   setport(0, 1); //Modo normal.
   wait(10); }
```

A2.2 – Cabeçalho – public.h

```
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <stdio.h>
#include <math.h>

#ifdef __cplusplus
   #define __CPPARGS ...
#else
   #define __CPPARGS
#endif

//Parametros
#define OK              55
#define FAIL            0xAA
#define ON              '1'
#define OFF             '0'
#define KNULL           -1
#define TRUE            0x50
#define FALSE           0x05
#define TIMEOVER        1000 // A/D trigger time out
#define PCL711          0 // Codigo da placa
#define MAXCHNUM        0x08

//---------------------------------------------------------------------------
//Definicao dos "types" que serao utilizados.
typedef void interrupt (*fptr)(__CPPARGS);
typedef unsigned short BOOL;
typedef unsigned short BYTE;
typedef unsigned int   WORD;
typedef unsigned long  DWORD;

//---------------------------------------------------------------------------
//Variaveis relacionadas a placa de aquisicao.
int             baseI=0x300;         //Endereco base de E/S.
volatile float   ad_value[MAXCHNUM];   //Valores aquisitados A/D   (0-4096).
volatile float   ad_nvalue[MAXCHNUM];  //Valores aquisitados (A/D) (0-5[V]).
int             ad_no=8;             //Num. canais A/D.
int             da_no=1;             //Num. canais D/A.
int             di_no=16;            //Num. canais digitais (entrada).
int             do_no=16;            //Num. canais digitais (saida).

/* ------------------------------------------------------------------------ */
#define MAX_VFASEAPNTS 1000  /*Numero de valores de tensao que sera armazenados*/
#define MimValFreq
#define VMedSampPart     10  /*Numero de valores de tensao para calcular a tensao media de partida*/
#define TmpPartBloq    16e3  /*Tempo que a partida ficara bloqueada*/
#define ANALOGKEY       0x2  /*Bit da chave que desativa o circuito uProcessado e ativa o nao
uProcessado*/
#define ANALOGPULSE     0x4  /*Bit da chave que liga ou desliga o circuito nao uProcessado de comutacao
forcada*/
```

```
/* ---------------------------------------------------------------------- */
int          PriCh = 1; //Primeiro canal a ser aquisitado.
int          UltCh = 3; //Ultimo canal a ser aquisitado.
volatile DWORD PulseCT[6]={0,0,0,0,0,0}; // Utilizados para verificar a transicao dos pulsos
volatile DWORD PulseLT[6]={0,0,0,0,0,0}; // nos tiristores.
volatile DWORD PulsePT[6]={0,0,0,0,0,0}; // Duracao dos pulsos. O valor em seg. depende da
                                         // taxa de amostragem.
/* ---------------------------------------------------------------------- */
volatile float far uVfase[3]={0.0, 0.0, 0.0}; //Guarda os ultimos valores de tensao nas fases.
volatile float far Vfase[3][MAX_VFASEAPNTS];  //Guarda os valores de tensao nas fases.
volatile int   far kVfase[3]={0,0,0}; //Contador utilizado como indice para o a matriz acima.
volatile int   far fVfase[3]={0,0,0}; //Indica quando kVfase for maior ou igual a MAX_VFASEAPNTS.
/* ---------------------------------------------------------------------- */
volatile long  far lContTmp    =  0; //Contador para base de tempo.
volatile long  far LstChange   = -1; //Ultima mudanca dos pulsos.
volatile int   far MinChngTime =  4; //Minimo tempo para mudanca dos pulsos.
volatile int   far BloqTemp    =  5; //Tempo de bloqueio x 2.
volatile int   far Enabled     =  0; //Indica se os pulsos estao habilitados.
volatile int   far iModoTeste  =  0; //Indica se o modo teste esta ativo ou nao.
volatile int   far SeqDir      =  1; //Indica sequencia de rotacao.
volatile int   far DigCtrl     =  1; //Controle digital ou analogico.
/* ---------------------------------------------------------------------- */
// Variaveis globais do modulo Timer.
fptr         Def_Interrupt;   //Guarda o end. da interrupcao padrao.
volatile int   PulsoAnt  =  0;  //Ultimo pulso enviado para a porta de saida.
volatile int   CFFlag    =  1;  //Indica se a comutacao forcada esta ou nao ativa.
volatile int   CFCount   = -1;  //Contador para tempo de bloqueio do inversor.
volatile DWORD iSampFreq;       //Frequecia da interrupcao.
volatile DWORD iOrigIrqNum;     //Num. de contagens para chamar a Irq original.
volatile DWORD iOIrqCont;       //Contador da Irq original.
volatile DWORD PartdT    =  0;  //Tempo entre os pulsos de partida.
volatile int   PartIdx;         //Indice do pulso de partida.
volatile int   PartdIdx;        //Incremento do indice do pulso de partida.
volatile int   TempDisp[4]={100, 8, 4, 400}; //Parametros ajustaveis.
//TempDisp[0] //Intervalo entre os pulsos de partida.
//TempDisp[1] //Duracao do pulso de partida.
//TempDisp[2] //Tempo de comutacao forcada.
//TempDisp[2] //Vel limite para comutacao forcada.

/* ---------------------------------------------------------------------- */
// Variaveis globais do modulo ZeroC.
volatile BYTE Pulse[6]={0,0,0,0,0,0};     //Valor dos pulsos.
volatile BYTE PsSeq[6]={3,6,12,24,48,33}; //Valor dos pulsos em binario.

/* ---------------------------------------------------------------------- */
// Prototipos das funcoes do modulo ZeroC.
//Envia um BYTE contendo informacoes sobre os pulsos para porta paralela.
void setport(unsigned short vout, short op);
//Envia um BYTE contendo informacoes sobre as chaves dos cicruitos.
void setport2(unsigned short pout);

/* ---------------------------------------------------------------------- */
// Sequencia de ligacoes:
// ----------------------
// 1 - Ligar o microcomputador e executar o programa de controle;
// 2 - Ligar a fonte principal (-15V, 0V, 15V);
```

```
// 3 - Ligar a fonte auxiliar (0V, 5V);
// 4 - Ligar instrumentos de medicao;
// 5 - Ligar o circuito do regulador de velocidade;
// 6 - Ligar o circuito de alimentacao CC;
// 7 - Ligar o circuito de alimentacao da ponte retificadora;
// 8 - Ligar o circuito de alimentacao da ponte inversora;

// Instrucoes antes de acionar o motor:
// ------------------------------------
// 1 - Abrir a chave do link DC;
// 2 - Verificar o ganho do cicruito somador, nao pode ser menor do 1;
// 3 - Verificar se o tiristor volante esta funcionando;
// 4 - Verificar se todos os tiristores da ponte inversora estao dispando;

// Instrucoes para partida:
// ------------------------
// 1 - Ajustar a tensao do varivolt para 200V.
// 2 - Pressionar o botao de partir.
```

A2.3 –Driver para a Placa de Aquisição de Dados – drive.hx

```
/* Adaptado do arquivo: Driver.h do programa PCLSTEST (Advantech).*/

// Ajuste dos parametros da Placa
void board_default(void)
{
  baseI=0x300;
  ad_no=8;
  da_no=1;
  di_no=16;
  do_no=16;
}

// Zera os vetores de tensao.
void init_AD(void) {
  int indx;

  for (indx=0;indx<8;indx++)
  {
    ad_value[indx]=0;
    ad_nvalue[indx]=0;
  }
}

// Zera a tensao na saida analogica da placa.
void init_DA(void){
  outportb(baseI+4,0);
  outportb(baseI+5,0);
  outportb(baseI+6,0);
  outportb(baseI+7,0);
}

// Le as portas A/D com disparo via software.
int read_AD(int StCh, int EndCh)
{
 unsigned int adhigh;
 unsigned int adlow;
 int indx;
 int timcnt;

 for (indx=StCh;indx<=EndCh;indx++)
 {
   timcnt=0;

   outportb(baseI+10, indx);
   outportb(baseI+12, 0);   // Dispara duas vezes para estabilizar.

   do {
      adhigh=inportb(baseI+5);
      if (++timcnt>=TIMEOVER) return(FAIL);
   } while ((adhigh&0x10)!=0);

   adlow=inportb(baseI+4);
   outportb(baseI+12,0);
```

```
   do {
     adhigh=inportb(baseI+5);
     if (++timcnt>=TIMEOVER) return(FAIL);
   } while ((adhigh&0x10)!=0);

   adlow=inportb(baseI+4);
   ad_value[indx]  = (adhigh<<8)+adlow;
   ad_nvalue[indx] = 10.0*ad_value[indx]/4096.0-5.0;

 }

 return(OK);

}

// Saida D/A
void write_DA(int chno, double da_out)
{
  unsigned int dahigh;
  unsigned int dalow;
  double cal;

  cal =  4096.0/10.0;
  cal *= (da_out+10.0);

  dahigh=cal/256;
  dalow=cal;
  dalow%=256;
  outportb(baseI+4+(chno<<2), dalow);
  outportb(baseI+5+(chno<<2), dahigh);

}

// Le as entradas digitais D/I
void read_DI(void)
{
  int indx;
  unsigned int di_byte[5];
  int bitcnt;
  int byteno;
  int bitno;

  di_byte[0]=inportb(baseI+6);
  di_byte[1]=inportb(baseI+7);
}

//Escreve nas saidas digitais.
void write_DO(unsigned int do_byte[])
{
  outportb(baseI+13,do_byte[0]);
  outportb(baseI+14,do_byte[1]);
}
```

A2.4 – Rotinas para Controle da Interrrupção – timer.hx

```
//Retorna positivo se o tempo dTemp ja passou.
BYTE tempod(DWORD Temp, DWORD dTemp)
{
  return labs(lContTmp-Temp)>dTemp;
}

//Rotina de geracao dos pulsos de partida.
volatile DWORD PartdT1=0;
volatile DWORD PartdT2=0;
void PulsosPartida(void)
{
    //Intervalo entre os pulsos de partida.
    //Tempo/2[ms]
    if (!PartdT1&&!PartdT2)
    {
          if (tempod(PartdT, TempDisp[0]))
          {
            DefPartPos(PartIdx);
            PartdT1=lContTmp;
          }
    }
    else if (PartdT1&&!PartdT2)
    {
          if (tempod(PartdT1, TempDisp[1]))
          {
            DefPartPos(-1);
            PartdT1=0;
            PartdT2=lContTmp;
          }
    }
    else if (!PartdT1&&PartdT2)
    {
          if (tempod(PartdT2, TempDisp[2]))
          {
            DefPartPos(-2);
            PartdT2=0;
            PartIdx += PartdIdx;
            if (PartIdx>5) PartIdx=0;
            else if (PartIdx<0) PartIdx=5;
            PartdT=lContTmp;
          }
    }
}

//=============================================================================
//Salva as tensoes em um arquivo.
int  SalVabc = 0;
FILE *adfout[3];

void SalvaAD(float Vn, int op)
{
  if (adfout[op]!=NULL&&SalVabc)
    fprintf(adfout[op], "%f\n", Vn);
```

```
}

//=============================================================================
int AbreAD(char *fname)
{
  char strn[100];
  char strk[10] = "_x.txt";
  int k;

  //for (k=0;k<100;k++) strn[k]=0;

  for (k=0;k<=2;k++)
  {
   strk[1]=k+49;
   strk[6]=0;
   sprintf(strn, "%s%s", fname, strk);
   if ((adfout[k] = fopen(strn, "wt")) == NULL)
   {
    adfout[k]=NULL;
    return 0;
   }
   fprintf(adfout, "[\n");
  }

  return 1;

}

//=============================================================================
int FechaAD(void)
{
  int k;

  for (k=0;k<=2;k++)
  {
   if (adfout[k]!=NULL)
   {
    fprintf(adfout[k], "]");
    fclose(adfout[k]);
   }
  }
  return 0;
}

//=============================================================================
//Rotina de aquisicao de dados.
void interrupt far AD_Interrupt(__CPPARGS)
{
   int    idx, i;
   float  tref, vf;

   //Rotina de leitura.
   read_AD(PriCh, UltCh);

   //Detecta a sequencia de pulsos.
   //if (SeqDir) idx=ZeroCrossing();
```

```
//else       idx=ZeroCrossingI();

//Detecta a sequencia de pulsos.
idx=ZeroCrossing();

//Rotina de pulsos
if (Enabled)
{

 if (PulsoAnt!=idx&&tempod(LstChange,MinChngTime))
 {
       if (CFFlag) //Verifica se o bloqueio est  ativo.
       {
        setport(idx, 0);  //Modo bloqueado.
        CFCount=BloqTemp; //Define o tempo de bloqueio.
       }
       else
       {
        setport(idx, 1); //Modo normal.
        CFCount=-1;
       }
       PulsoAnt  = idx;
       LstChange = lContTmp;
 }

 // Conta o tempo em bloqueio.
 if (CFCount>0) CFCount--;
 else if (CFCount==0)
 {
       setport(PulsoAnt, 1); //Modo normal.
       CFCount = -1;
 }

}
else if (iModoTeste) PulsosPartida(); //Executa a partida.

//Detecta a mudanca no estado dos pulsos e salva o intervalo de tempo.
for (i=0;i<6;i++)
{
 if (PulseLT[i]!=PulseCT[i])
 {
       PulsePT[i] = labs(PulseCT[i]-PulseLT[i]);
       PulseLT[i] = PulseCT[i];
 }
}

//Salva o valor de tensao das fases.
for (i=0;i<3;i++)
{
 vf = ad_nvalue[i+1];
 if (fVfase[i]||iModoTeste)
 {
       if (kVfase[i]<MAX_VFASEAPNTS-1) Vfase[i][kVfase[i]++] = vf;
       else fVfase[i]=0;
 }
 if (uVfase[i]<0&&vf>0) fVfase[i]=!fVfase[i];
```

```
      uVfase[i] = vf;
    }

    // Contador, utilizado como base de tempo.
    if (++lContTmp>4e9) lContTmp=0;

    // Chama a interrupcao original.
    if ( --iOIrqCont<=0 )
    {
            iOIrqCont = iOrigIrqNum;
            _chain_intr((fptr)Def_Interrupt); //Chama a interrupcao original.
    }

    outportb( 0x20, 0x20 );

}

//=========================================================================
// Executa a reprogramacao do chip de interrupcoes.
void DefIrqFreq( int iFreq )
{
    // Desabilita as interrupcoes.
    asm { cli; }

    if ( iFreq<18 ) iFreq=18; //O valor minimo e 18.2...
    iOIrqCont    = 0;         //Zera o contador...
    iOrigIrqNum  = iFreq/18;  //Numero de vezes ate chamar a irq original.
    iSampFreq    = iFreq;

    //Calcula a frequencia.
    if (iFreq==18) iFreq=0;
    else iFreq=((unsigned int)(1193180L/((unsigned long)iFreq)));

    asm {
      mov dx, 0x43;
      mov al, 0x34;
      out dx, al;
      mov dx, 0x40;
      mov ax, iFreq
      out dx, al;
      mov al, ah;
      out dx, al;
    }

    lContTmp = (long)iOrigIrqNum;

    // Reabilita as interrupcoes.
    asm { sti; }

}
```

A2.5 – Rotinas para Detecção da Seqüência de Pulsos – zeroc.hx

```
//Gera um atraso de dtim milisegundos.
void wait (long dtim)
{
  long k;
  for (k=0;k<dtim*1000;k++)
    outportb(0x80, 0);
}

//Gera um atraso de dtim microsegundos.
void uwait (long dtim)
{
  long k;
  for (k=0;k<dtim;k++)
    outportb(0x80, 0);
}

//Envia um BYTE contendo informacoes sobre os pulsos para porta paralela.
void setport(unsigned short vout, short op)
{
  if (!op) outportb(0x378, ~vout);
  else outportb(0x378, ~vout&191);
}

//Envia um BYTE contendo informacoes sobre as chaves dos cicruitos.
void setport2(unsigned short pout)
{
  outportb(0x37A, pout);
}

//Calcula o valor rms da tensao em uma fase X.
float VFase_rms(float x[], int vnum)
{
   float fret=0.0, Vfm=0.0;
   int   i;

   if (vnum>0)
   {
     for (i=0;i<vnum;i++)
            Vfm += (x[i] * x[i]);
     if (Vfm) fret = sqrt(Vfm/(float)vnum);
   }
   return (fret);

}

//Define a posicao do rotor para partida.
int DefPartPos(short dPos)
{
   if (Enabled||!DigCtrl) return (0);
   if (dPos>=0) setport(PsSeq[dPos], 1);
   else if (dPos==-1) setport(0, 0);
   else setport(0, 1);
   return 1;
}
```

```
//Define a posicao do rotor.
int DefPosicao(short dPos)
{
  int  k, t1=100, t2=20;

  if (dPos>5||dPos<0||Enabled||!DigCtrl) return (0);

  for (k=0;k<2;k++)
  {
   setport(PsSeq[dPos], 1); wait(t1);
   setport(0, 0); wait(t2);
  }

  setport(0, 0); wait(10);
  setport(0, 1); wait(t2);

  return (1);

}

//Detecta a sequencia de pulsos para o motor no sentido direto.
int ZeroCrossing(void)
{
   float        VA=ad_nvalue[1];
   float        VB=ad_nvalue[2];
   float        VC=ad_nvalue[3];
   float        L =0.0;
   unsigned int   D1=0, i;
   unsigned short ZA=0, ZB=0, ZC=0, P[6]={0,0,0,0,0,0};

   if (VA>L) ZA=1;    else if (VA<-L) ZA=0;
   if (VB>L) ZB=1;    else if (VB<-L) ZB=0;
   if (VC>L) ZC=1;    else if (VC<-L) ZC=0;

   if ((ZA && !ZB))  { D1  =  1; P[0]=1; }
   if ((ZA && !ZC))  { D1 |=  2; P[1]=1; }
   if ((ZB && !ZC))  { D1 |=  4; P[2]=1; }
   if ((ZB && !ZA))  { D1 |=  8; P[3]=1; }
   if ((ZC && !ZA))  { D1 |= 16; P[4]=1; }
   if ((ZC && !ZB))  { D1 |= 32; P[5]=1; }

   //Detecta a mudanca no estado dos pulsos.
   for (i=0;i<6;i++)
   {
    if (P[i]&&P[i]!=Pulse[i])
          if (tempod(PulseCT[i], MinChngTime))
            PulseCT[i] = lContTmp;
    Pulse[i] = P[i];
   }

   return (D1);

}

//Detecta a sequencia de pulsos para o motor no sentido inverso.
```

```
int ZeroCrossingI(void)
{
   float        VA=ad_nvalue[1];
   float        VB=ad_nvalue[3];
   float        VC=ad_nvalue[2];
   float        L =0.0;
   unsigned int   D1=0, i;
   unsigned short ZA=0, ZB=0, ZC=0, P[6]={0,0,0,0,0,0};

   if (VA>L) ZA=1;    else if (VA<-L) ZA=0;
   if (VB>L) ZB=1;    else if (VB<-L) ZB=0;
   if (VC>L) ZC=1;    else if (VC<-L) ZC=0;

   if ((ZA && !ZB))  { D1  =  1; P[0]=1; }
   if ((ZC && !ZB))  { D1 |=  2; P[1]=1; }
   if ((ZC && !ZA))  { D1 |=  4; P[2]=1; }
   if ((ZB && !ZA))  { D1 |=  8; P[3]=1; }
   if ((ZB && !ZC))  { D1 |= 16; P[4]=1; }
   if ((ZA && !ZC))  { D1 |= 32; P[5]=1; }

   //Detecta a mudanca no estado dos pulsos.
   for (i=0;i<6;i++)
   {
    if (P[i]&&P[i]!=Pulse[i])
          if (tempod(PulseCT[i], MinChngTime))
            PulseCT[i] = lContTmp;
    Pulse[i] = P[i];
   }

   return (D1);

}
```

Anexo 3 – Fotos dos Elementos do Sistema

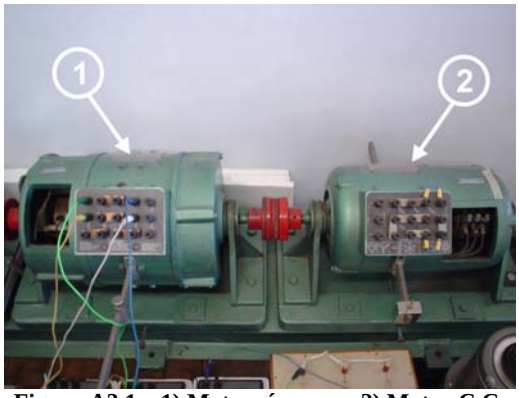

Figura A3.1 – 1) Motor síncrono; 2) Motor C.C.

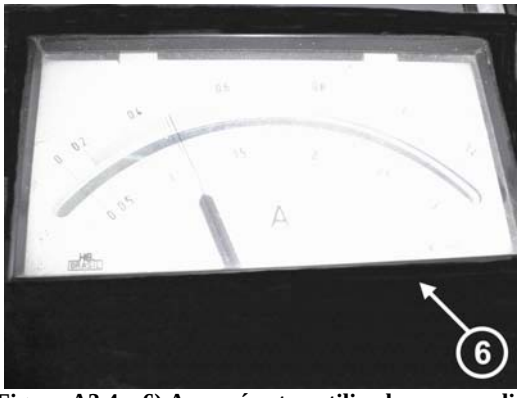

Figura A3.4 – 6) Amperímetro utilizado para medir a corrente de excitação do MS.

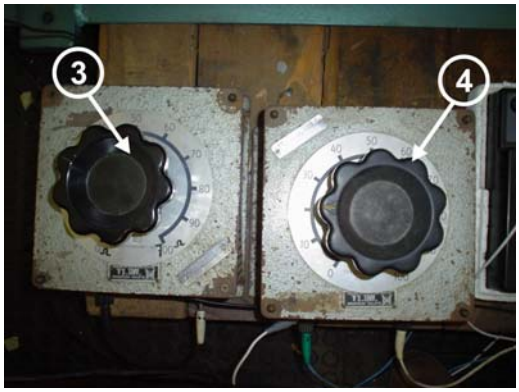

Figura A3.2 – 3) Reostato de campo do motor síncrono; 4) Reostato de campo do MCC.

Figura A3.5 – 7) Fonte de alimentação auxiliar para o circuito de disparo do inversor.

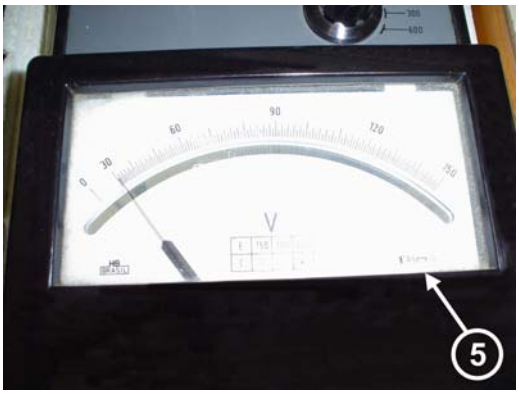

Figura A3.3 – 5) Voltímetro utilizado para medir a tensão na armadura do MCC.

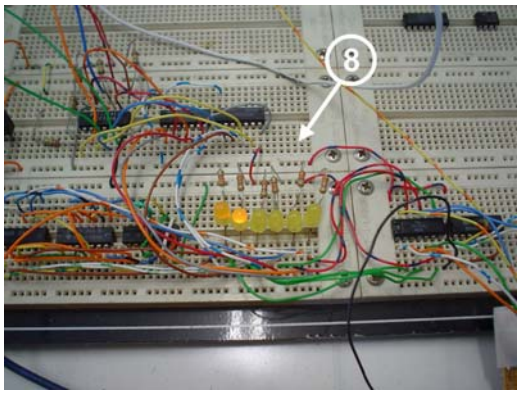

Figura A3.6 - 8) LED para monitoração dos pulsos de disparo.

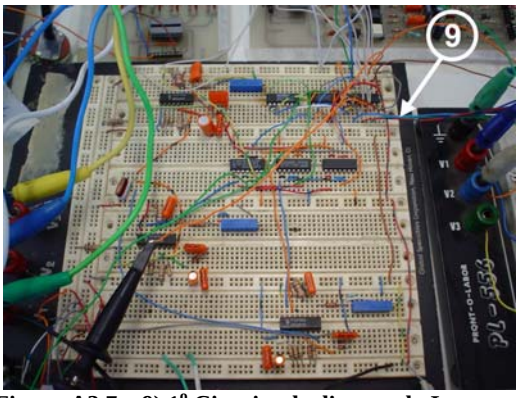

Figura A3.7 – 9) 1º Circuito de disparo do Inversor.

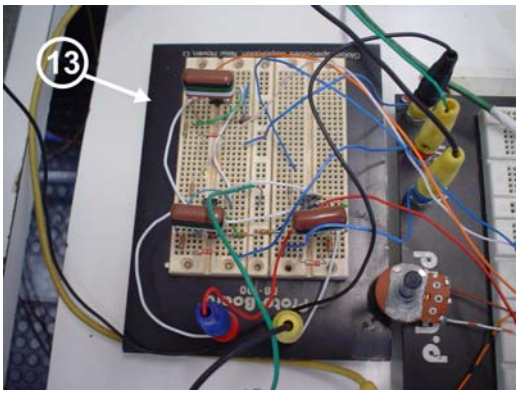

Figura A3.10 - 3) Circuito detector de zero.

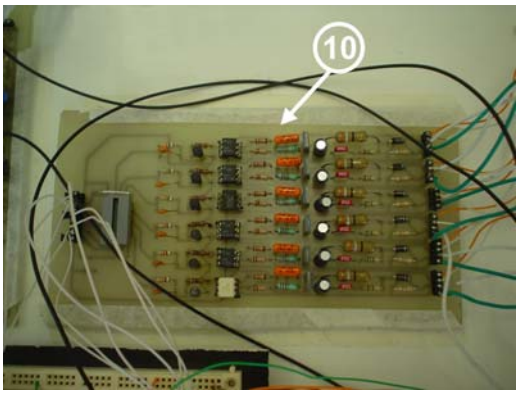

Figura A3.8 – 10) Driver amplificador de pulsos.

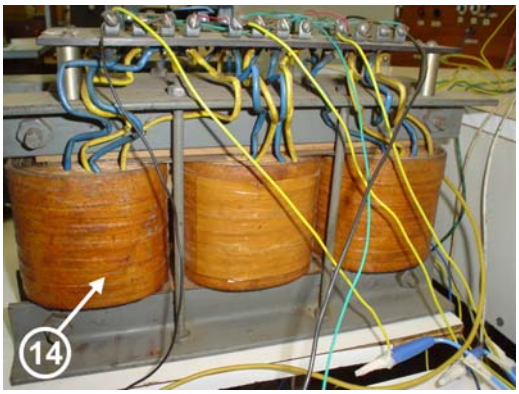

Figura A3.11 – 14) Transformador utilizado para o circuito gerador de pulsos para comutação forçada.

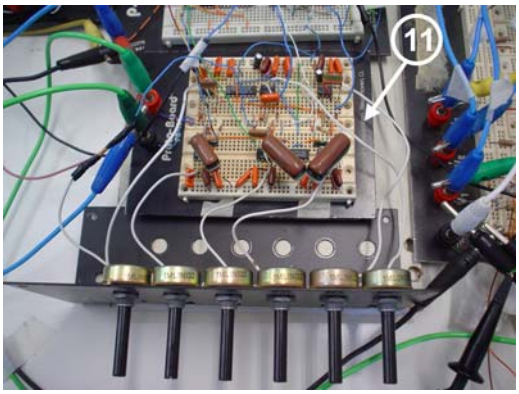

Figura A3.9 – 11) Gerador de pulsos para comutação forçada, parte B.

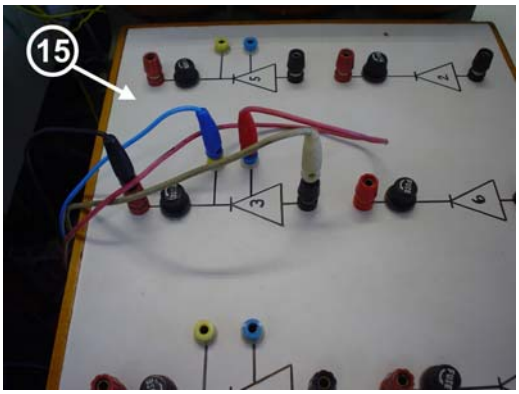

Figura A3.12 – 15) Tiristor volante.

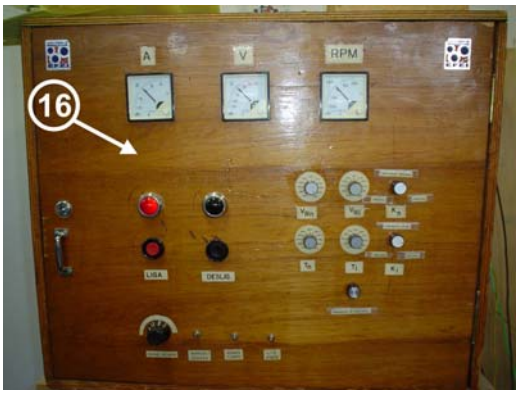

Figura A3.13 – 16) Módulo do regulador de velocidade

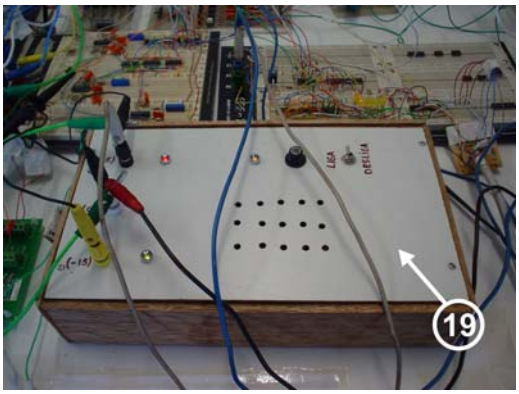

Figura A3.16 – 19) Fonte de alimentação do circuito do inversor.

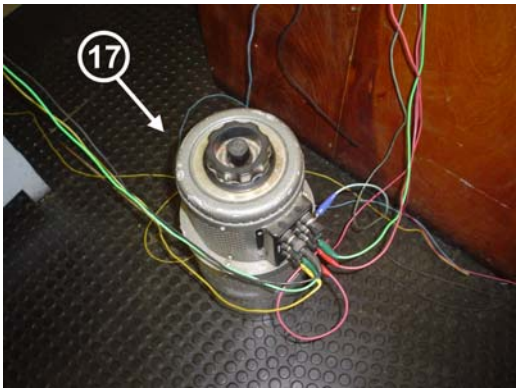

Figura A3.14 – 17) Varivolt utilizado para ajustar tensão de entrada do retificador.

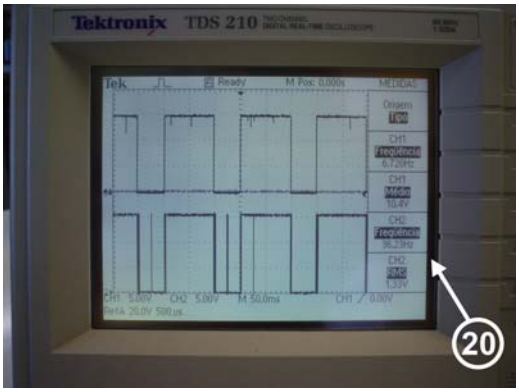

Figura A3.17 – 20) Osciloscópio utilizado para monitorar os pulsos para a ponte inversora.

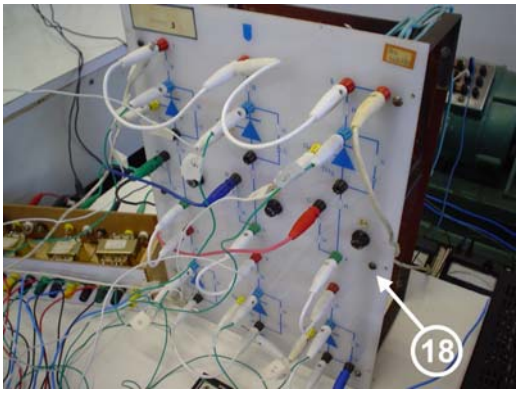

Figura A3.15 – 18) Ponte inversora.

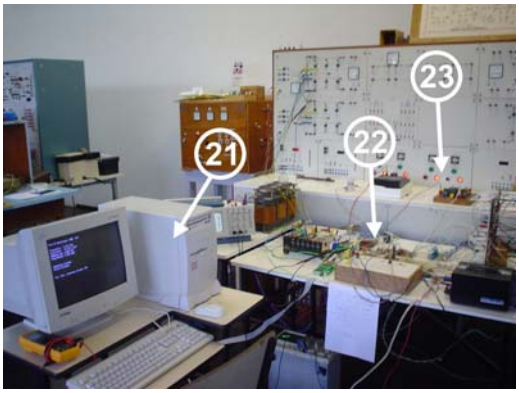

Figura A3.18 – 21) Microcomputador utilizado; 22) circuito de disparo do inversor; 23) Elementos para monitoração e ligação do link DC.

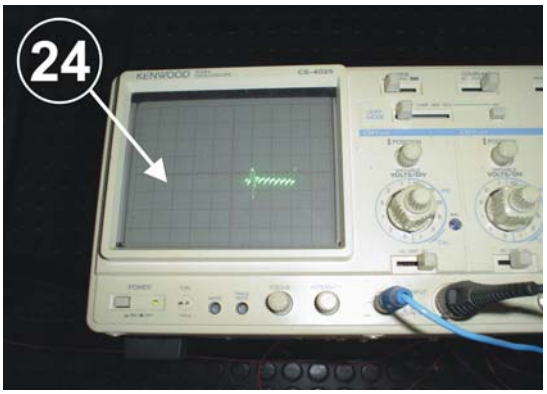

Figura A3.19 – 24) Osciloscópio utilizado para monitorar a ponte inversora.

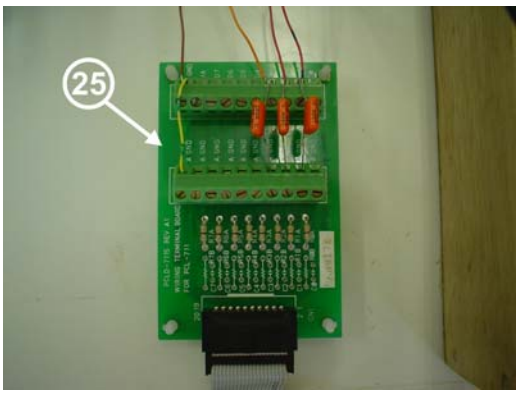

Figura A3.20 – 25) Bornes da placa de aquisição de dados.

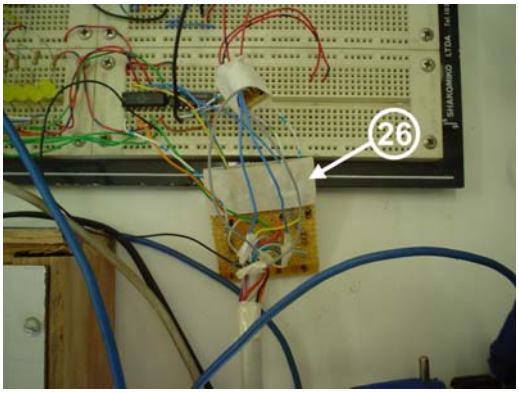

Figura A3.21 – 26) Placa de interface entre a porta paralela e o buffer do circuito de disparo do inversor.

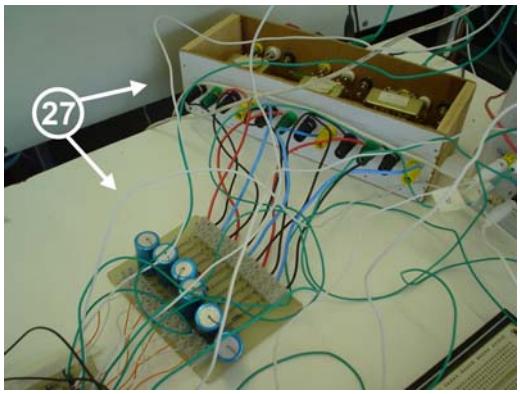

Figura A3.22 – 27) Fonte de alimentação do amplificador de pulsos.

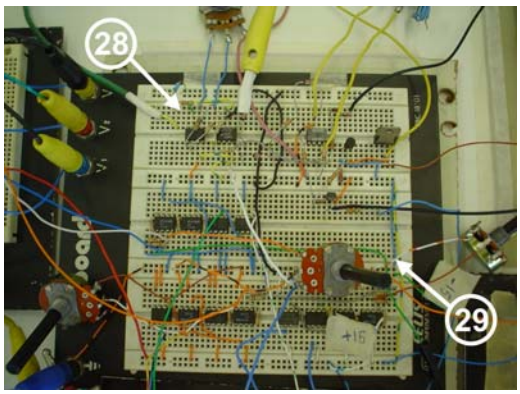

Figura A3.23 – 28) Circuito somador utilizado para comutação forçada; 29) Gerador de pulsos para comutação forçada, parte A.

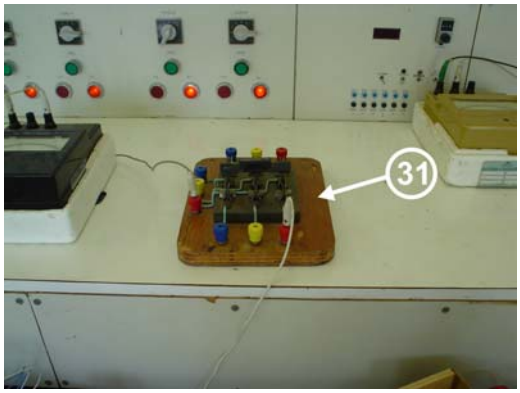

Figura A3.24 – 31) Chave do Link DC.

Bibliografia

[1] ADVANTECH Co., Ltda. “PCL-711B - PC-MultiLab User’s Manual”; Taiwan : Advantech Co., Ltd., aug., 1993.

[2] QUINDERÉ, K. E. B. DE – “Implementação de um Sistema de Acionamento Controlado Para a Máquina Síncrona Utilizando Reguladores Analógicos e Digitais” – Tese de Mestrado, Escola Federal de Engenharia de Itajubá, 1999.

[3] A. Bellini, G. Franceschini, M. Paladini, N. Petrolini, C. Tassoni, “Low Cost Implementation of A Sensorless Control for Synchronous Reluctance Motors”, IEEE SDEMPED 2001, Gorizia, September 2001.

[4] J. Dixon and I. Leal, “Current Control Strategy for Brushless DC Motors, Based on a Common DC Signal”, IEEE Transactions on Industrial Electronics, March 2002.

[5] A. Jakubowicz, M. Nougaret, and R. Perret, “Simplified Model and Closed-Loop Control of a Commutatorless DC Motor”, IEE Trans. Ind. Appl., vol. IA-16,pp.165-172, Mar./Apr. 1980.

[6] B. Mueller, T. Spinanger, D. Wllstein, “Static Variable Frequency Starting and Drive System”, IEEE/IAS 1979 Annual Meeting, pp. 429-438, 1979.

[7] E. R. da Silva, L. P.B. de Oliveira, C. B. Jacobina, “Control Techniques Used in Current Supply Inverters with Pulsed Current in DC Busbar”, XII Congresso Brasileiro de Automática, pp. 1683-1688, 1998.

[8] F. Fröhr & F. Orttenburger “Introduccion al Control Eletronico”, Siemens, Marcombo S/A, Barcelona, 1986.

[9] B. R. Pelly “Thyristor Phase-Controlled Converters and Cycloconverters”, Wiley-Interscience, 1971.