Universidade Federal de Itajubá
UNIFEI

Estimador do comportamento do conjugado de motores de indução através de DSP.

Fabiano Valias de Carvalho

Dissertação de mestrado apresentada
no programa de Pós-Graduação em
Engenharia Elétrica, como complementação dos
créditos necessários para obtenção do título
de Mestre em Engenharia Elétrica

Itajubá, MG
2002

Dedico aos meus pais, Vicente e Heliana, e à Carla, pelo apoio e compreensão indispensáveis à realização deste trabalho.

AGRADECIMENTOS

A Deus, por mais esta etapa concluída.

Ao professor Luiz Eduardo Borges da Silva, pelo incentivo, amizade e orientação.

Ao amigo e professor Jocélio Souza de Sá, pelo apoio.

Aos amigos Ricardo Braga, Ricardo Pini e Thiago Rolin, pelo apoio.

Ao empresário Alcenor, por ter aberto as portas de sua empresa para que a montagem do protótipo se tornasse possível.

Ao professor José Carlos Grilo e todo o pessoal do LEPCH pelo apoio na comprovação dos resultados.

À Texas Instruments por fornecer o conversor A/D para a placa de coleta de dados, através programa de amostras.

RESUMO

A proposta deste trabalho é a implementação de um método de estimação do conjugado dinâmico de motores de indução trifásicos, a partir de amostras das correntes e tensões trifásicas, utilizando como ferramenta um circuito baseado em DSP (*Digital Signal Processor)*.

Atualmente, é utilizado o controle vetorial em acionamentos nos quais é exigido um alto desempenho dinâmico. Para avaliação do fluxo utiliza-se o sensoriamento direto no entreferro, através de sondas de efeito Hall ou outra técnica de medição. O fluxo medido no entreferro é realimentado para o sistema de controle e usado para gerar as componentes de corrente responsáveis pelo conjugado. O sensoriamento direto do fluxo tem alto custo e o sistema de medição pode não apresentar a precisão necessária ao processo.

Outro processo consiste na medição da tensão e corrente do estator e a partir dos valores medidos, estimar o fluxo do entreferro através do processamento de algumas equações. A estratégia do processo tem como base a utilização de um filtro passa-baixas em cascata programável (PCLPF - Programmable Cascaded Low-Pass Filter) para a estimação do fluxo do estator. Esta técnica é conhecida por permitir a integração ideal da tensão do estator, desde freqüências extremamente baixas, até altas freqüências na escala de enfraquecimento de campo, sem introduzir nenhum *offset* DC na saída.

Além da medição da tensão e da corrente do estator, o único parâmetro do motor necessário para estimação do fluxo do estator, utilizando-se o a técnica do filtro passa-baixas programável em cascata (PCLPF), é a impedância equivalente ao enrolamento do estator, do qual a resistência representa parcela significativa. Este método de estimação do fluxo do estator elimina sensores de fluxo e de velocidade, diminuindo o custo e aumentando a confiabilidade do sistema.

É importante observar que a implementação do filtro passa-baixas em cascata programável (PCLPF), foi substituída por rede neural recorrente (RNN-Recurrent Neural Network), que permite a estimação do fluxo de forma simples e rápida. A resposta da rede neural recorrente (RNN) comparada com a resposta de um PCLPF baseado em alguma técnica de processamento digital de sinais, apresenta comportamento idêntico em regime permanente, mas mostra superioridade na resposta transitória. Para o treinamento da rede neural recorrente (RNN) é utilizado um algoritmo baseado em filtro de Kalman.

ABSTRACT

The purpose of this work is the implementation of a method for estimation of the dynamic torque to induction motor, using the three phases voltages and currents samples, with a DSP (Digital Signal Processor) based circuit tool.

Nowadays, the vector control is used in drives where high-performance dynamic is required. The flux is measured directly in the gap, with Hall effect sensors, or another measurement technique. The gap measured flux feedback is made available to the drive system and the currents, which produce torque, are synthesized. The direct flux measurement is expensive and the process accuracy may not be enough.

Another process consists in the measurement of the stator voltages and currents and then, the stator flux is calculated by means of equations processing. The process strategy is the stator flux estimation by the programmable cascaded low-pass filter (PCLPF). This method is well known because it permits the stator voltage integration, since extremely low frequency to high frequency on the field-weakening range, without introducing any offset at the output.

Besides the stator voltage and current measurement, the only parameter under consideration to the stator flux estimation, using the programmable low-pass filter (PCLPF) method, is the stator equivalent impedance, where the significant component is the resistance. This stator flux estimation method eliminates flux and speed sensors, decreasing cost and making the system reliably better.

It is important to notice that the programmable cascaded low-pass filter (PCLPF) was substituted by recurrent neural network (RNN), which permits the

fast and simple flux estimation. The recurrent neural network (RNN) response, compared with another digital signal processing based PCLPF implementation, shows equal behavior at steady state, but is superior at transient response. A Kalman filter based algorithm is used to recurrent neural network (RNN) training.

SUMÁRIO

1- INTRODUÇÃO 1

2- PROPOSTA 3

2.1 O Motor de Indução Trifásico 3

2.2 O Processamento Digital de Sinais 3

2.3 O Uso de DSP para Estimar o Conjugado 5

3- MODELO DINÂMICO d,q,0 DE MOTORES DE INDUÇÃO 7

3.1 Circuitos Equivalentes para o MIT 7

1.2 Transformação para o Sistema d,q,0 – dois eixos 10

1.3 Equação do Conjugado 16

4- ESTIMAÇÃO DO CONJUGADO DINÂMICO USANDO FILTRO PASSA-BAIXAS PROGRAMÁVEL IMPLEMENTADO COM REDE NEURAL RECORRENTE (RNN-PCLPF) 18

4.1 Estimação do Vetor de Fluxo por Filtro Passa-Baixas em Cascata 18

4.2 Implementação do PCLPF baseado em Rede Neural Recorrente 19

1.3 Algoritmo de Treinamento da Rede Neural Recorrente por Filtro de Kalman Estendido (EKF) 23

1.4 Aplicação do Procedimento de Treinamento da Rede Neural Recorrente ao PCLPF 28

1.5 Estimação do Conjugado a partir dos Vetores de Fluxo do estator 29

2- DSP 56000 DA MOTOROLA 32

2.1 Características Principais do DSP56002 32

2.1.1. Gerador de Clock com PLL 34

2.1.2. Unidade Lógica e Aritmética de Dados 35

2.1.3. Registros de Entrada (X1, X0, Y1, Y0) 35

2.1.4. Unidade Lógica e MAC 35

2.1.5. Acumuladores A e B 36

2.1.6. Portas A, B e C 37

2.1.7. Representação Numérica Fracionária de Ponto Fixo (Formato Q) 38

3- SIMULAÇÃO DO ESTIMADOR DE CONJUGADO 40

3.1 Diagrama Geral da Simulação 40

3.2 Amostragem e Retenção 41

3.3 Conversão de Três Eixos para Dois Eixos (Três para Duas Fases) 42

3.4 O Filtro Passa-Baixas PCLPF (Programmable Cascaded Low-Pass Filter) 44

3.5 Cálculo do Conjugado Dinâmico a Partir dos Vetores de Fluxo de Campo Estacionário 47

3.6 O Modelo do Motor de Indução Trifásico 52
3.7 Resultados da Simulação 53
4- MONTAGEM DO PROTÓTIPO DO ESTIMADOR 59
4.1 Aquisição de Dados 59
4.1.1. O ADS7864 60
4.1.1.1. Entradas de Controle do ADS7864 61
4.1.1.2. Sinal Analógico de Entrada 63
4.1.1.3. As Etapas da Conversão e Leitura de Dados 64
4.1.2. Placa para o ADS7864 66
4.1.2.1. Fonte de Alimentação 66
4.1.2.2. Circuito Oscilador para o Conversor ADS7864 67
4.1.2.3. Configuração da Placa do ADS7864 68
4.1.2.4. Conectores da Placa 70
4.2 Os Sensores de Efeito Hall 74
4.2.1. Os Sensores de Tensão 74
4.2.2. Os Sensores de Corrente 75
4.2.3. A Placa para os Sensores de Tensão e Corrente 76
4.3 O DSP 77
4.4 Sistema Completo 82
5- TESTES E RESULTADOS 86
6- CONCLUSÕES E SUGESTÕES 96
7- LISTA DE ACRÔNIMOS 100
8- BIBLIOGRAFIA 101
9- ANEXOS 103
9.1 Placa para o ADS7864 103
9.1.1. Esquema Elétrico 103
9.1.2. Placa de Circuito Impresso 104
9.2 O DSP 105
9.2.1. Portas A, B e C 105
9.2.2. Emulador de Chip 106
9.2.3. Conjunto de Instruções 106
9.2.4. Vetor de Interrupções 108
9.2.5. Arquivo de Configuração para o Conversor A/D e D/A CS4215 108
9.2.6. Arquivo de Configuração das Interrupções do Canal Serial para Comunicação com o DSP 113
9.3 O EVM 114

9.3.1. Hardware do DSP56002EVM 115
9.3.1.1. Depurador e Interface Gráfica do Usuário (GUI) 116
9.4 Simulação..... 117
9.4.1. Diagrama Completo do Modelo do Motor de Indução Trifásico..... 117
9.4.2. Arquivo de Configuração do Modelo do MIT Utilizado na Simulação.... 119
9.5 Configuração da Placa de Coleta de Dados Paralela do PC..... 119
9.6 Listagem Completa do Programa Utilizado 120
9.7 Rotina em DELPHI para Mostrar o Conjugado no PC 132

ESTIMADOR DO COMPORTAMENTO DO CONJUGADO DE MOTORES DE INDUÇÃO ATRAVÉS DE DSP

1- INTRODUÇÃO

Existem diversas técnicas para o controle vetorial de motores de indução trifásicos. Uma delas, proposta por Hasse [1], lança mão do sensoriamento indireto do fluxo, ou seja, o fluxo é estimado em tempo real através de medição da corrente, tensão e velocidade do motor. O fluxo estimado é então realimentado para um sistema de controle mostrado na Figura 1-1. No método indireto é necessário o conhecimento preciso dos parâmetros do motor, pois caso contrário este método pode provocar a saturação ou a subexcitação da máquina, causando a deterioração do desempenho dinâmico.

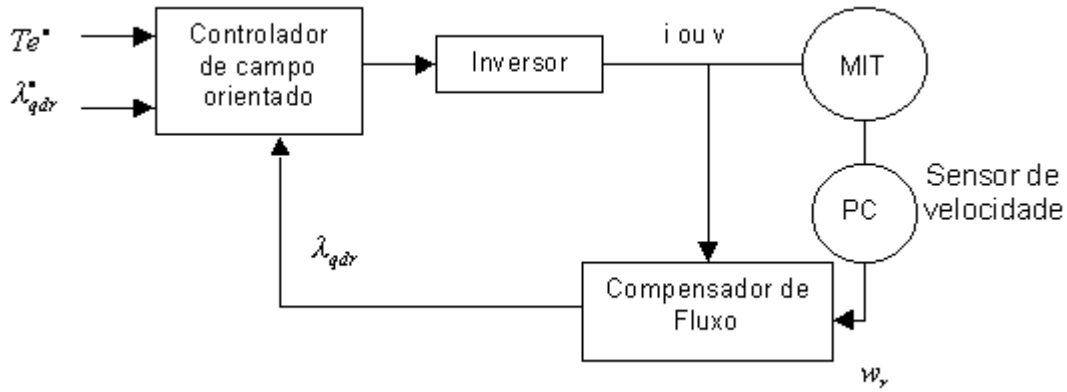

Figura 1-1 - Método Indireto para Estimar o Fluxo

Outra técnica de controle vetorial, desenvolvida por Blaschke [2], utiliza o sensoriamento direto do fluxo do entreferro por uso de sondas de efeito Hall ou outra técnica de medida. O fluxo medido no entreferro é realimentado para o controlador, mostrado na Figura 1-2. Esta segunda técnica é praticamente insensível à variação dos parâmetros do rotor [3], porém a necessidade de

utilização de sensores de fluxo torna o sistema de custo elevado, alem do que podem ocorrer erros na medidas de fluxo.

Com a utilização do DSP (*Digital Signal Processor*), o fluxo e consequentemente o conjugado, podem ser estimados em função das correntes e tensões nos terminais da máquina, eliminando-se assim sensores de velocidade, de fluxo ou de conjugado. Como conseqüência o custo cai e a confiabilidade aumenta.

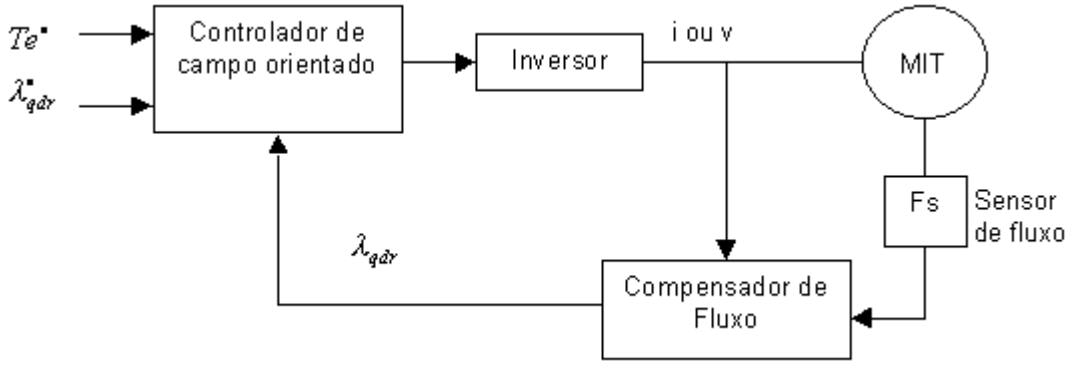

Figura 1-2 - Método Direto para o Sensoriamento do Fluxo

Os métodos usuais para a estimação do fluxo podem ser trabalhosos, pois os algoritmos são complexos e sensíveis às variações de parâmetros. Para estimar o fluxo é feita uma integração numérica que apresenta problema de *offset* DC. Por outro lado a resistência do estator é único parâmetro do motor a ser considerado, podendo ser facilmente compensado. O problema do *offset* DC devido à integração, pode ser resolvido pelo uso da técnica de filtro passa-baixas em cascata programável (*Programmable Cascaded Low-Pass Filter* - PCLPF) [4] e [5].

O uso de vários filtros em cascata permite a implementação de filtros com constante de tempo pequena, resolvendo o problema do *offset* DC para baixas freqüências [4].

2- PROPOSTA

2.1 O Motor de Indução Trifásico

Os motores de indução trifásicos, devido à simplicidade, robustez e custo relativamente baixo, são utilizados largamente no meio industrial. Em certos locais onde não pode ocorrer faiscamento (típico de motores de CC) devido à presença de gases, o motor de indução é o mais indicado. É o caso, por exemplo, de refinarias de petróleo.

A grande desvantagem do motor de indução trifásico consiste na dificuldade de controle de velocidade. Com o desenvolvimento da eletrônica de potência, modernos sistemas de controle eletrônicos vêm sendo implementados de maneira a possibilitar um melhor controle da velocidade para o motor de indução trifásico. O custo de implementação destes sistemas cai a cada dia e já é bem razoável.

2.2 O Processamento Digital de Sinais

As bases teóricas do processamento digital de sinais (DSP) remontam desde a metade dos anos 60, tendo evoluído notavelmente nas décadas seguintes, até os dias de hoje. Até meados dos anos 80, a maioria das aplicações de DSP consistia naquelas que eram passíveis de processamento *off-line*, devido ao pequeno poder computacional dos processadores e ao seu alto custo. Um crescente número de aplicações na área de processamento de sinais têm sido implementadas em tempo real, através de processadores digitais de sinais integrados, devido ao contínuo avanço da tecnologia de

semicondutores. As vantagens deste enfoque são inúmeras quando comparadas à implementação analógica:

- Número menor de componentes,
- Desempenho estável e determinístico,
- Tolerâncias menores e ausência de ajustes,
- Maior imunidade a ruído e interferência,
- Faixa maior de aplicações realizáveis e/ou implementações mais efetivas.

O processamento digital de sinais requer, geralmente, um grande número de operações aritméticas sobre cada amostra do sinal. Assim, apesar de ser possível a realização de algum processamento de sinais em processadores convencionais, isto geralmente não é eficaz, devido ao pequeno poder computacional aritmético destes. Portanto, o que realmente diferencia um processador convencional de um específico para DSP é a sua arquitetura interna. De modo a otimizar o desempenho, a maioria dos processadores DSP utiliza dois recursos diferenciados:

- *Arquitetura Harvard*: - Otimiza a capacidade de processamento, estabelecendo duas estruturas de barramento de memória separadas, de programa e de dados. Isto permite que, internamente, tanto os códigos quanto os operandos das instruções possam ser transferidos simultaneamente, em um único ciclo.

- *Unidade Lógica e Aritmética MAC*: - A operação aritmética básica em DSP é a de multiplicação e acumulação concatenada (MAC). A unidade lógica e aritmética (ALU) do DSP é estruturada de forma a possibilitar estas operações em uma única instrução e realizando-as, geralmente, em um único ciclo de máquina.

2.3 O Uso de DSP para Estimar o Conjugado

A proposta deste trabalho é a implementação de um circuito baseado em DSP (*Digital Signal Processor*) para estimar o conjugado dinâmico de um motor de indução trifásico. Para isto o fluxo do entreferro é estimado em tempo real, pelo método do filtro passa-baixas em cascata programável (PCLPF) [4]. A partir do fluxo estimado calcula-se o conjugado do motor. O parâmetros necessários para a estimação são as correntes e tensões nas três fases e a resistência do estator.

A implementação de um filtro passa-baixas em cascata programável (PCLPF), permite a estimação do fluxo do estator, livre de *offset* DC. Um diagrama simplificado da implementação pode ser visto na Figura 2-1. Para a implementação do filtro é utilizada uma rede neural recorrente (*Recurrent Neural Network* - RNN) treinada por filtro de Kalman. A estimação do fluxo baseada na RNN é simples, permite uma rápida implementação e apresenta uma resposta transitória superior, quando comparada com PCLPF implementado com alguma outra técnica de processamento digital de sinais [5] e [6].

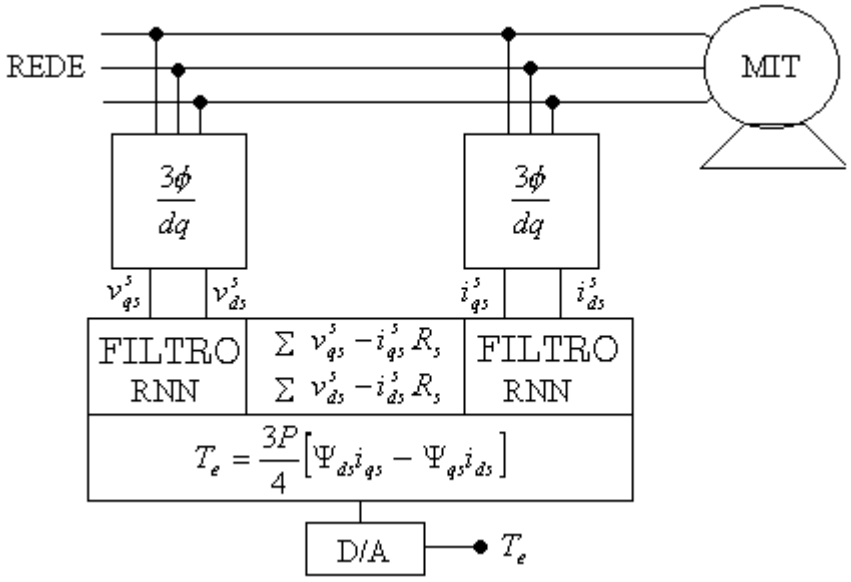

Figura 2-1 - Diagrama Simplificado do Trabalho Proposto

Redes neurais artificiais (ANN - *Artificial Neural Network*) são utilizadas em robótica, em engenharia biomédica, em telecomunicações e também representam um grande avanço quando aplicadas ao controle de máquinas elétricas, baseado em DSP (*Digital Signal Processor*). A rede neural artificial é rápida, apresenta tolerâncias a ruído e erro.

Além da rede neural recorrente (*Recurrent Neural Network* - RNN) que é o tipo de rede utilizada neste trabalho, existem outros tipos de redes neurais artificiais, alguns tipos são, *feedforward network*, *Kohonen network, Hamming network* e *Hopfield network*. Cada tipo de ANN tem sua própria estrutura de rede e algoritmos de treinamento.

As características de computação paralela rápida, capacidade de aprendizagem e tolerância à falhas das redes neurais, não são comuns à computação baseada em técnicas convencionais de processamento digital de sinais. Redes neurais têm sido aplicadas em transformação vetorial, modulação por largura de pulso, estimação do vetor de fluxo, estimação do conjugado e velocidade, controle inteligente, diagnóstico de falha, etc.

Muitas aplicações de redes neurais em sistemas de eletrônica de potência utilizam redes do tipo *feedforward* [6]. Elas são fáceis de treinar mas somente podem executar funções de mapeamento não linear estática de entrada e saída, limitando sua capacidade de operar com sistemas dinâmicos. Portanto estas redes não são indicadas para controle de alto desempenho, com estimação e realimentação, uma vez que na maioria das vezes estes requerem a emulação de sistemas dinâmicos. Por outro lado, o tipo de rede neural utilizada neste trabalho, rede neural recorrente (RNN), é eficiente para resolver tal problema, mas o seu treinamento é mais difícil se comparado com as redes do tipo *feedforward*. Para o treinamento da rede neural recorrente deste trabalho, é utilizado um algoritmo baseado em filtro de Kalman.

3- MODELO DINÂMICO d,q,0 DE MOTORES DE INDUÇÃO

3.1 Circuitos Equivalentes para o MIT

Para os circuitos equivalentes do MIT será considerado que este possui entreferro uniforme, circuito magnético linear, enrolamentos do estator idênticos e as bobinas do rotor são dispostas de tal forma que a FMM do rotor possa ser considerada senoidal no espaço, tendo o mesmo número de pólos que a FMM do estator. Os circuitos equivalentes do estator e do rotor podem ser vistos na Figura 3-1.

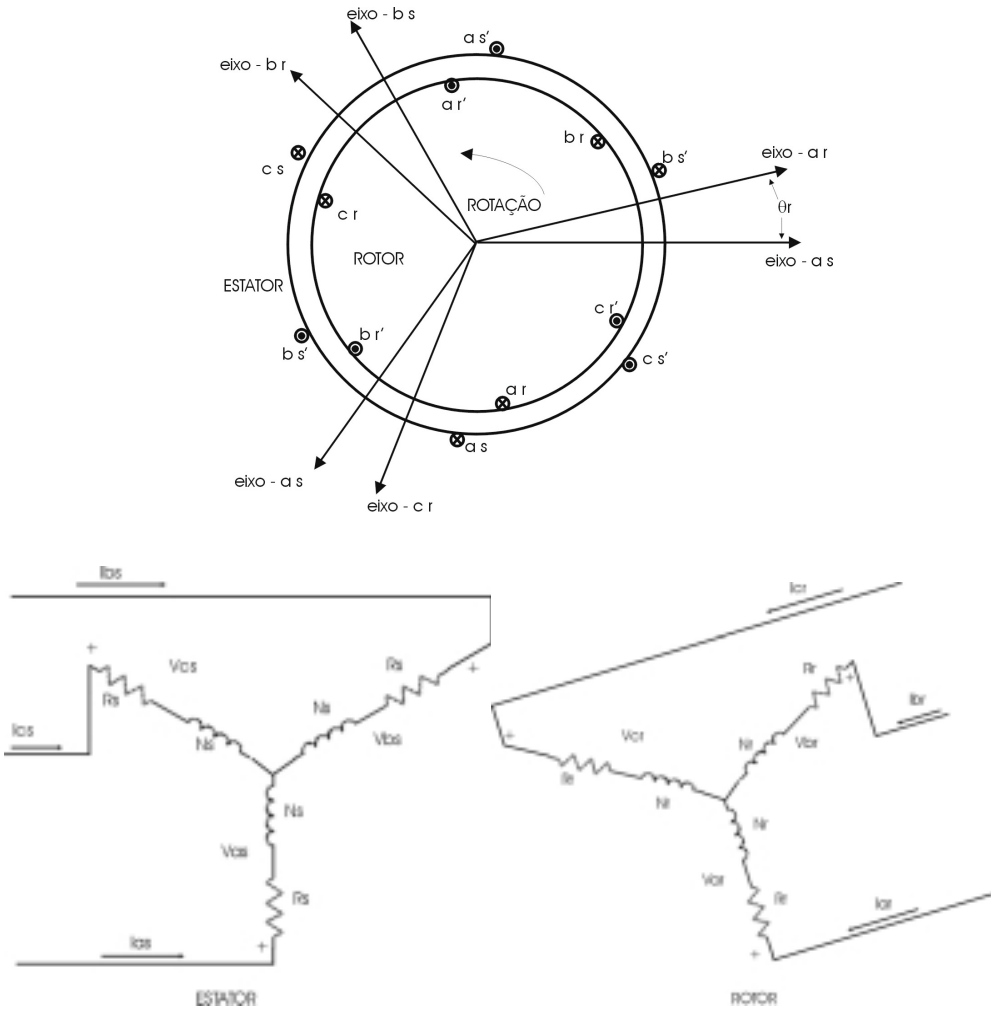

Figura 3-1 - Circuitos equivalentes do estator e do rotor

Sendo os enrolamentos do estator idênticos e também os enrolamentos do rotor, as equações do estator e rotor considerando um motor com 2 pares de pólos podem ser dadas por:

$$v_{as} = R_s i_{as} + \frac{d\psi_{as}}{dt} \qquad v_{bs} = R_s i_{bs} + \frac{d\psi_{bs}}{dt} \qquad v_{cs} = R_s i_{cs} + \frac{d\psi_{cs}}{dt}$$

$$v_{ar} = R_r i_{ar} + \frac{d\psi_{ar}}{dt} \qquad v_{br} = R_r i_{br} + \frac{d\psi_{br}}{dt} \qquad v_{cr} = R_r i_{cr} + \frac{d\psi_{cr}}{dt} \tag{3-1}$$

onde:

v_{is} – tensões fase-neutro do estator

v_{ir} – tensões fase-neutro do rotor

i_{is} – correntes fase-neutro do estator

i_{ir} – correntes fase-neutro do rotor

ψ_{is} – fluxo fase-neutro do estator

ψ_{ir} – fluxo fase-neutro do rotor

i – fases a, b e c

R_s – resistência do estator

R_r – resistência do rotor referida ao estator

As equações dos fluxos podem ser definidas pelas equações

(3-2). A dificuldade da modelagem do MIT em um sistema de três eixos pode ser observada em (3-1) e

(3-2), uma vez que os parâmetros variam dinamicamente com θ_r.

(3-2)

$$\begin{bmatrix} \psi_{as} \\ \psi_{bs} \\ \psi_{cs} \\ \psi_{ar} \\ \psi_{br} \\ \psi_{cr} \end{bmatrix} = \begin{bmatrix} L_s & 0 & 0 & L_{sr}\cos\theta_r & L_{sr}\cos(\theta_r + \frac{2\pi}{3}) & L_{sr}\cos(\theta_r + \frac{2\pi}{3}) \\ 0 & L_s & 0 & L_{sr}\cos(\theta_r - \frac{2\pi}{3}) & L_{sr}\cos\theta_r & L_{sr}\cos(\theta_r + \frac{2\pi}{3}) \\ 0 & 0 & L_s & L_{sr}\cos(\theta_r + \frac{2\pi}{3}) & L_{sr}\cos(\theta_r - \frac{2\pi}{3}) & L_{sr}\cos\theta_r \\ L_{sr}\cos\theta_r & L_{sr}\cos(\theta_r - \frac{2\pi}{3}) & L_{sr}\cos(\theta_r + \frac{2\pi}{3}) & L_r & 0 & 0 \\ L_{sr}\cos(\theta_r + \frac{2\pi}{3}) & L_{sr}\cos\theta_r & L_{sr}\cos(\theta_r - \frac{2\pi}{3}) & 0 & L_r & 0 \\ L_{sr}\cos(\theta_r - \frac{2\pi}{3}) & L_{sr}\cos(\theta_r + \frac{2\pi}{3}) & L_{sr}\cos\theta_r & 0 & 0 & L_r \end{bmatrix} * \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \\ i_{ar} \\ i_{br} \\ i_{cr} \end{bmatrix}$$

Onde:

L_s – auto-indutância do estator

L_{sr} – indutância mútua referida ao estator

L_r – auto-indutância do rotor referida ao estator

θ_r – ângulo entre as fases do estator e rotor

3.2 Transformação para o Sistema d,q,0 – dois eixos

O estudo do comportamento do MIT é complexo devido a efeitos do acoplamento entre as fases do rotor e estator, onde os coeficientes de acoplamento variam com a posição do rotor, variando no tempo.

Se a alimentação trifásica é balanceada pode-se usar a teoria de eixos ortogonais dq, que também é conhecida por d,q,0, para a modelagem dinâmica do MIT. Neste sistema as variáveis e parâmetros em três eixos do sistema de C.A. original, são transformados para dois eixos ortogonais, denominados direto e em quadratura. Desta forma os coeficientes variantes no tempo que dependem da posição do rotor, são eliminados, eliminando o ângulo θ_r do sistema de equações diferenciais representativo da máquina. Se a alimentação é desbalanceada pode-se utilizar a transformação 1,2,0, [7] ao invés da d,q,0. Neste trabalho é considerada uma alimentação trifásica balanceada.

A transformação consiste em decompor o sistema trifásico em componentes d e q localizadas nos eixos ditos direto e de quadratura do rotor e com o referencial no rotor. A Figura 3-2 mostra a posição dos eixos d e q sobre o rotor e o enrolamento trifásico localizado no estator de uma máquina. O eixo direto está alinhado com o eixo de campo e o eixo de quadratura é localizado 90º a frente do eixo direto. O ângulo θ é definido como o ângulo entre o eixo da fase a e o eixo direto.

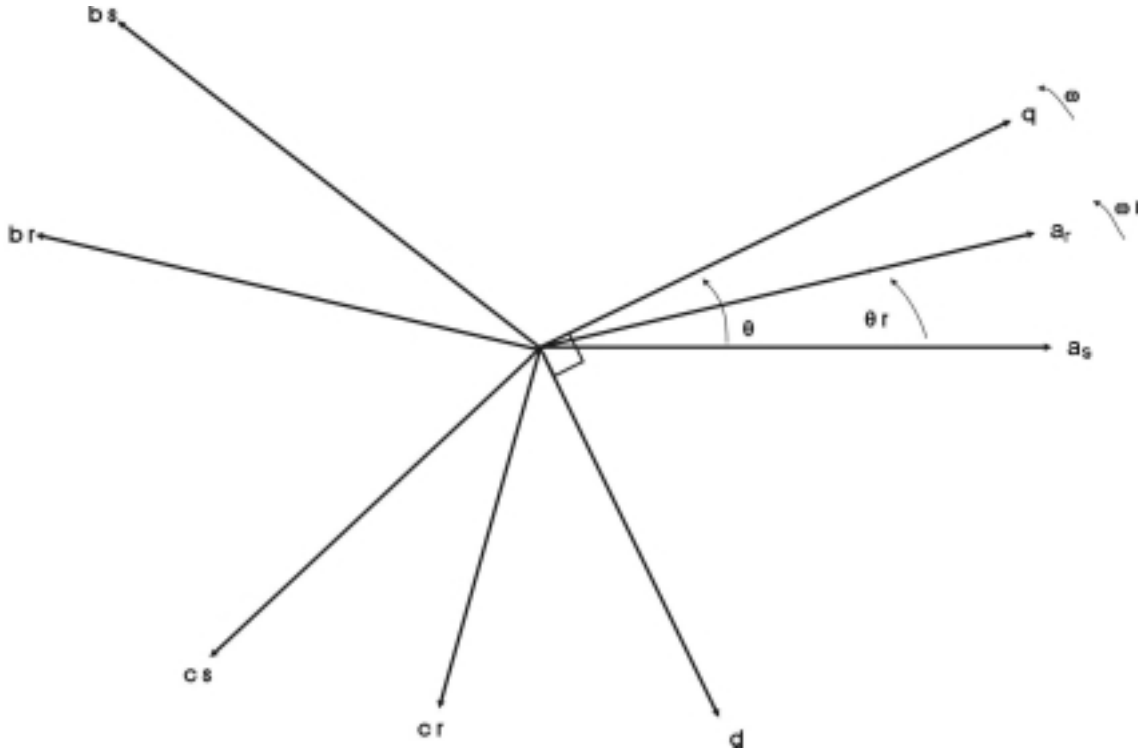

Figura 3-2 - Transformação para o Sistema dq

O modelo dinâmico dq pode ser expresso em uma estrutura de referência estacionária ou em uma estrutura de referência síncrona. Os eixos dq ficam fixos no estator para a estrutura de referência estacionária, e na estrutura de referência síncrona, os eixos estão girando na velocidade síncrona (w). Nesta última estrutura as variáveis aparecem em quantidades C.C. em regime permanente [3]. Neste trabalho é usada a estrutura de referência estacionária, onde w = 0.

As equações de transformação dos eixos para esta estrutura referência podem ser definidas como:

$$f_{di} = \frac{2}{3}[f_{ai}\cos\theta_i + f_{bi}\cos(\theta_i - \frac{2\pi}{3}) + f_{ci}\cos(\theta_i + \frac{2\pi}{3})] \quad (3\text{-}3)$$

$$f_{qi} = \frac{2}{3}[f_{ai}\,\mathrm{sen}\,\theta_i + f_{bi}\,\mathrm{sen}(\theta_i - \frac{2\pi}{3}) + f_{ci}\,\mathrm{sen}(\theta_i + \frac{2\pi}{3})] \quad (3\text{-}4)$$

$$f_{0i} = \frac{1}{3}(f_{ai} + f_{bi} + f_{ci}) \quad (3\text{-}5)$$

onde:

- i *:* indica se a transformação é do estator ou do rotor.
- f_{qi}, f_{di} e f_{0i} *:* pode ser a tensão, a corrente ou o fluxo transformados para o eixo em quadratura, eixo direto e eixo de componente zero.
- $\theta_{is} = \theta$ *:* ângulo inicial entre o eixo q e o eixo da fase A do estator.
- $\theta_{ir} = \theta - \theta_r$ *:* ângulo inicial entre o eixo q e o eixo da fase A do rotor.
- Fatores multiplicativos $\frac{2}{3}$ e $\frac{1}{3}$*:* fator para igualar as amplitudes dos sinais antes e depois da conversão para o sistema dq.

As expressões (3-3), (3-4) e (3-5) podem ser expressas em notação matricial da forma:

$$\begin{bmatrix} f_{di} \\ f_{qi} \\ f_0 \end{bmatrix} = \begin{bmatrix} \frac{2}{3}\cos\theta & \frac{2}{3}\cos(\theta - \frac{2\pi}{3}) & \frac{2}{3}\cos(\theta + \frac{2\pi}{3}) \\ -\frac{2}{3}sen\theta & -\frac{2}{3}sen(\theta - \frac{2\pi}{3}) & -\frac{2}{3}sen(\theta + \frac{2\pi}{3}) \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} \quad (3\text{-}6)$$

Lembrando que f pode ser tensão, corrente ou fluxo.

As equações de fluxo podem ser obtidas trabalhando as equações de transformação (3-3), (3-4) e (3-5) na matriz

(3-2) :

$$\psi_{qs} = L_s i_{qs} + \frac{3}{2} L_{sr} i_{qr} \quad (3\text{-}7)$$

$$\psi_{ds} = L_s i_{ds} + \frac{3}{2} L_{sr} i_{dr} \quad (3\text{-}8)$$

$$\psi_{qr} = L_r i_{qr} + \frac{3}{2} L_{sr} i_{qs} \quad (3\text{-}9)$$

$$\psi_{dr} = L_r i_{dr} + \frac{3}{2} L_{sr} i_{ds} \quad (3\text{-}10)$$

onde considerando um sistema equilibrado as equações para a componente zero do sistema d,q,0 não são necessárias.

Aplicando as equações de transformação (3-3), (3-4) e (3-5) nas equações de tensão (3-1):

$$v_{qs} = R_s i_{qs} + \frac{d\psi_{qs}}{dt} + \psi_{ds} w \quad (3\text{-}11)$$

$$v_{ds} = R_s i_{ds} + \frac{d\psi_{ds}}{dt} - \psi_{qs} w \quad (3\text{-}12)$$

$$v_{0s} = R_s i_{0s} + \frac{d\psi_{ds}}{dt} \quad (3\text{-}13)$$

$$v_{qr} = R_r i_{qr} + \frac{d\psi_{qr}}{dt} + \psi_{dr} (w - w_r) \quad (3\text{-}14)$$

$$v_{dr} = R_r i_{dr} + \frac{d\psi_{dr}}{dt} - \psi_{qr} (w - w_r) \quad (3\text{-}15)$$

$$v_{0r} = R_r i_{0r} + \frac{d\psi_{dr}}{dt} \quad (3\text{-}16)$$

onde w_r é a velocidade angular elétrica do rotor. Para as tensões são consideradas as equações das componentes zero do sistema d,q,0, pois serão úteis na determinação da equação do conjugado mais adiante.

A equação que relaciona a tensão e a corrente independente do ângulo de deslocamento entre as fases do estator e rotor, é chamada transformação de Park [8], podendo ser obtida pela substituição de (3-7) a (3-10) em (3-11) a (3-16) e pode ser escrita na forma matricial:

(3-17)

$$v = \begin{bmatrix} v_{qs} & v_{ds} & v_{qr} & v_{dr} \end{bmatrix}^T = \begin{bmatrix} R_s + \frac{dL_s}{dt} & wL_s & \frac{d\frac{3}{2}L_{sr}}{dt} & w\frac{3}{2}L_{sr} \\ -wL_s & R_s + \frac{dL_s}{dt} & w\frac{3}{2}L_{sr} & \frac{d\frac{3}{2}L_{sr}}{dt} \\ \frac{d\frac{3}{2}L_{sr}}{dt} & (w-w_r)\frac{3}{2}L_{sr} & R_r + \frac{dL_r}{dt} & (w-w_r)L_r \\ -(w-w_r)\frac{3}{2}L_{sr} & \frac{d\frac{3}{2}L_{sr}}{dt} & -(w-w_r)L_r & R_r + \frac{dL_r}{dt} \end{bmatrix} \begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{qr} \\ i_{dr} \end{bmatrix}$$

A equação (3-17) pode ser particularizada para a estrutura de referência estacionária, onde os eixos dq são fixos no estator, então $w = 0$ em (3-17). As tensões, correntes e fluxos serão agora denotados por v_{ds}^s, i_{ds}^s, v_{qs}^s, i_{qs}^s, Ψ_{ds}^s e Ψ_{qs}^s, onde o sobrescrito s indica referência estacionária (w = 0).

Então as equações de tensão podem ser escritas:

$$v_{qs}^s = v_{as} \qquad (3\text{-}18)$$

$$v_{ds}^s = -\frac{1}{\sqrt{3}}v_{bs} + \frac{1}{\sqrt{3}}v_{cs} \qquad (3\text{-}19)$$

onde v_{as}, v_{bs} e v_{cs} são as tensões entre fase e neutro do estator e podem ser escritas como:

$$v_{as} = \sqrt{2} * V \operatorname{sen}(\theta)$$

$$v_{bs} = \sqrt{2} * V \operatorname{sen}(\theta - \frac{2\pi}{3})$$

$$v_{cs} = \sqrt{2} * V \operatorname{sen}(\theta + \frac{2\pi}{3})$$

Sendo o modelo de referência estacionário, os eixos dq estão fixos no estator (w = 0), então o ângulo θ pode ser igualado a zero em (3-6) para a simplificação das expressões.

Logo as equações de tensão podem ser escritas na forma matricial:

$$\begin{bmatrix} v_{ds}^{s} \\ v_{qs}^{s} \\ v_{0s}^{s} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} \qquad \text{(3-20)}$$

Sob condições equilibradas $v_{as} + v_{bs} + v_{cs} = 0$, então a matriz (3-20) fica:

$$\begin{bmatrix} v_{ds}^{s} \\ v_{qs}^{s} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} \qquad \text{(3-21)}$$

Análogamente as matrizes de transformação (3-21) podem ser utilizadas para a corrente e para o fluxo.

3.3 Equação do Conjugado

A potência instantânea que pode ser medida nos terminais da máquina é dada por:

$$P = [\, e_a\, i_a + e_b\, i_b + e_c\, i_c \,] \quad \text{(3-22)}$$

Em componentes d,q,0 a expressão de potência fica [9]:

$$P = \frac{3}{2}[e_d i_d + e_q i_q + 2e_0 i_0] \quad \text{(3-23)}$$

Em um sistema equilibrado e_0 e i_0 são nulos, portanto:

$$P = \frac{3}{2}[e_d i_d + e_q i_q] \quad \text{(3-24)}$$

Para uma análise mais completa da expressão da potência, as expressões de tensão (3-11) a (3-16), podem ser substituídas na expressão da potência (3-23):

$$P = \frac{3}{2}[i_{di}(R_i i_{di} + \frac{d\psi_{di}}{dt} - \psi_{qi} w) + i_{qi}(R_i i_{qi} + \frac{d\psi_{qi}}{dt} + \psi_{di} w) + 2i_{0i}(R_i i_{0i} \frac{d\psi_{0i}}{dt})] \quad \text{(3-25)}$$

Reordenando os termos em (3-25) fica:

$$P = \frac{3}{2}(i_{di} \frac{d\psi_{di}}{dt} + i_{qi} \frac{d\psi_{qi}}{dt} + 2i_{0i} \frac{d\psi_{0i}}{dt}) + \frac{3}{2}(i_{qi}\psi_{di} - i_{di}\psi_{qi})w + \frac{3}{2} R_i (i_{di}^2 + i_{qi}^2 + 2i_{0i}^2) \quad \text{(3-26)}$$

O primeiro termo da equação (3-26) representa a taxa da redução da energia magnética da armadura [9], o segundo termo indica a transferência de potência através do entreferro, e o terceiro e último termo representa a perda na resistência da armadura.

Finalmente a equação do conjugado pode ser obtida a partir do segundo termo de (3-26). Para isto a potência do entreferro será dividida pela velocidade do rotor w, $(w \equiv \frac{d\theta}{dt})$.

Então o conjugado elétrico no sistema dq para um motor com dois pares de pólos pode ser dado por [10]:

$$T_e = \frac{3}{2}(i_q \psi_d - i_d \psi_q) \quad (3\text{-}27)$$

Nota-se então que a expressão do conjugado não depende do ângulo θ_r, dependendo apenas das componentes do fluxo e da corrente, onde estas componentes são representadas no sistema dq. A corrente é facilmente mensurável e o fluxo será estimado por rede neural recorrente (RNN), treinada por filtro de Kalman.

A expressão do conjugado para a estrutura de referência estacionária pode ser escrita:

$$T_e = \frac{3P}{4}(i_q \psi_d - i_d \psi_q) \quad (3\text{-}28)$$

onde P = número de pares de pólos.

4- ESTIMAÇÃO DO CONJUGADO DINÂMICO USANDO FILTRO PASSA-BAIXAS PROGRAMÁVEL IMPLEMENTADO COM REDE NEURAL RECORRENTE (RNN-PCLPF)

4.1 Estimação do Vetor de Fluxo por Filtro Passa-Baixas em Cascata

Os vetores de fluxo de campo estacionário do estator Ψ_{ds}^{s} e Ψ_{qs}^{s}, podem ser estimados pela resolução da equações (4-1) e (4-2). Dois filtros passa-baixas em cascata programáveis (PCLPF) [4] idênticos são utilizados para estimação de Ψ_{ds}^{s} e Ψ_{qs}^{s}, como mostrado na Figura 4-1.

$$\Psi_{ds}^{s} = \int (v_{ds}^{s} - i_{ds}^{s} R_{s}) dt \qquad (4\text{-}1)$$

$$\Psi_{qs}^{s} = \int (v_{qs}^{s} - i_{qs}^{s} R_{s}) dt \qquad (4\text{-}2)$$

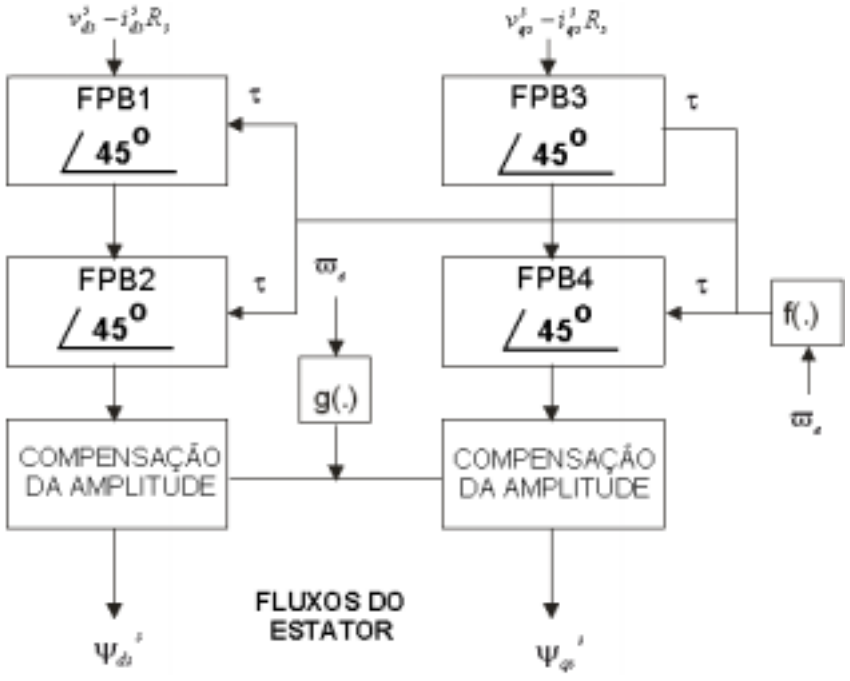

Figura 4-1 – Filtro Passa-Baixas em Cascata Programável (PCLPF)

O PCLPF essencialmente permite a integração ideal da tensão desde freqüências extremamente baixas até altas freqüências na escala de enfraquecimento de campo, sem introduzir nenhum *offset* na saída [4].

É utilizada uma série de *n* (a Figura 4-1 mostra n = 2) filtros passa-baixas programáveis, em cascata, com o objetivo de obter um deslocamento de fase total de 90^0 para as componentes Ψ_{ds}^{s} e Ψ_{qs}^{s} e um ganho adequado, para se conseguir uma integração ideal em qualquer freqüência. O ganho de compensação de amplitude ajusta o ganho total do PCLPF com o objetivo de alcançar a integração ideal da tensão do estator, compensando a queda de tensão no estator.

A constante de tempo (τ) dos componentes do filtro e o ganho de compensação da amplitude (G) do PCLPF, são funções não lineares da freqüência. O τ e o ganho (G) programáveis, mantêm o deslocamento de fase e o ganho de cada filtro idênticos para qualquer freqüência. Estes filtros serão substituídos por rede neural recorrente, e tanto a constante de tempo quanto o ganho dos filtros serão representados pelos pesos W da rede neural, no sistema discretizado da seção 4.2 a seguir.

4.2 Implementação do PCLPF baseado em Rede Neural Recorrente

Para uma integração ideal é necessário um deslocamento de fase de 90° que pode ser conseguido com vários estágios de PCLPF, para a implementação da rede neural serão considerados apenas dois estágios (n=2) por causa da complexidade computacional. Cada estágio de filtro passa-baixas na Figura 4-1 pode ser discretizado e representado pela função de transferência com transformada Z na forma

$$\frac{Y(z)}{U(z)} = \left(\frac{K}{z-a} \right) \qquad (4\text{-}3)$$

Portanto os dois estágios de filtro em cascata podem ser representados na forma

$$\frac{Y_1(z)}{U(z)} = \left(\frac{Kz^{-1}}{1-az^{-1}} \right) \qquad (4\text{-}4)$$

$$\frac{Y_2(z)}{Y_1(z)} = \left(\frac{Kz^{-1}}{1-az^{-1}} \right) \qquad (4\text{-}5)$$

e as equações correspondentes discretas no tempo são

$y_1(k) = a\, y_1(k\text{-}1) + K\, u(k\text{-}1)$ (4-6)

$y_2(k) = a\, y_2(k\text{-}1) + K\, y_1(k\text{-}1)$ (4-7)

onde $y_1(k)$ = saída do primeiro estágio, $y_2(k)$ = saída do segundo estágio e $u(k)$ = entrada do filtro. As Equações (4-6) e (4-7) podem ser expressas na forma matricial

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} a & 0 \\ K & a \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} K \\ 0 \end{bmatrix} = u(k) \qquad (4\text{-}8)$$

A RNN equivalente será representada pela equação (4-9)

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} W_{13} \\ W_{23} \end{bmatrix} = u(k) \quad (4\text{-}9)$$

$W_{12} = W_{23} = 0$, logo

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} W_{11} & 0 \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} W_{13} \\ 0 \end{bmatrix} = u(k) \quad (4\text{-}10)$$

A Figura 4-2 mostra a estrutura de uma rede neural recorrente (RNN) [5] para implementação do PCLPF, onde a equação (4-10) é representada pela RNN e W_{11}, W_{21}, W_{22} e W_{13} são os pesos da rede. O pesos são determinados pelo treinamento da rede, utilizando para isto um algoritmo baseado em filtro de Kalman, seção 4.3.

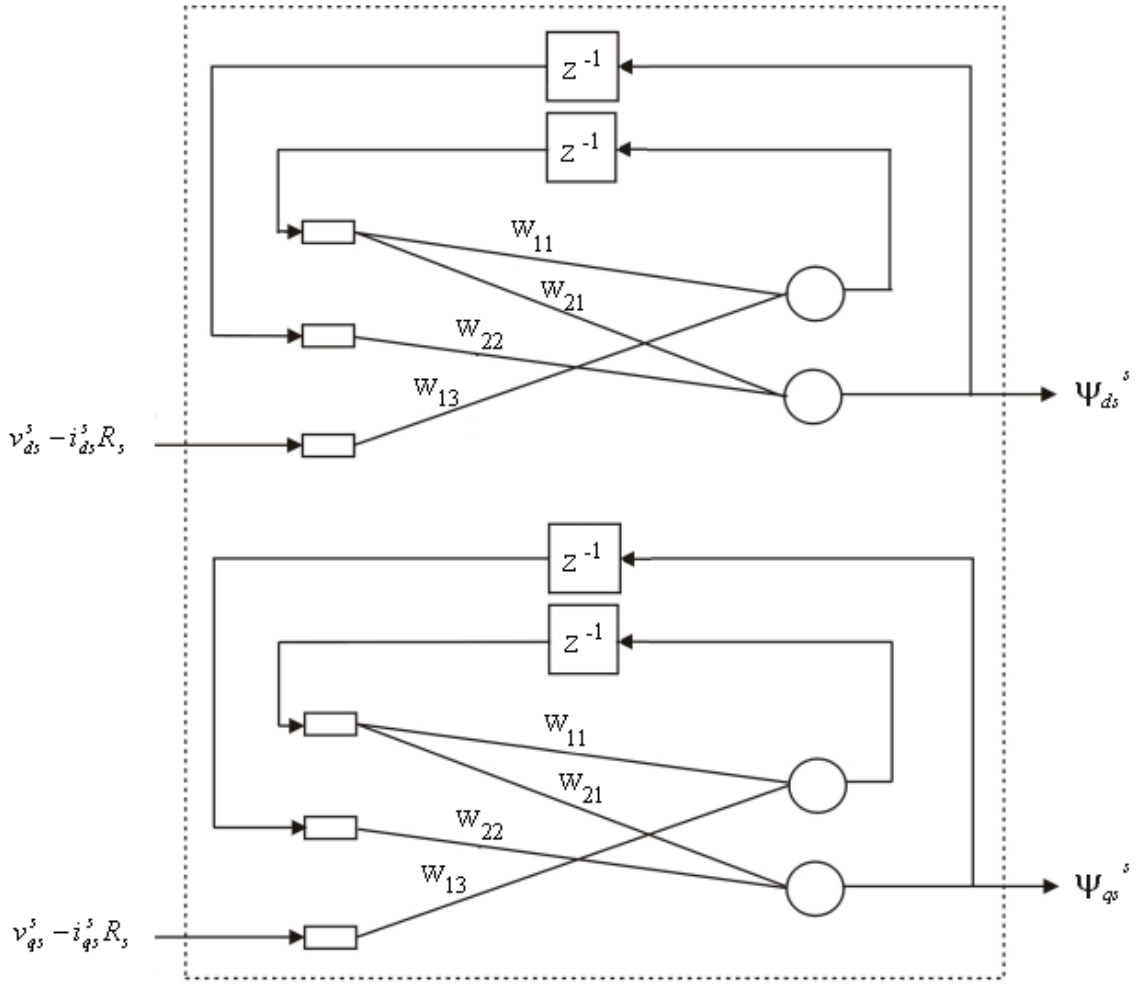

Figura 4-2 - Rede Neural Recorrente – RNN

Basicamente o sistema de estimação consiste em duas redes neurais recorrentes (RNN) de segunda ordem ou redes com realimentação, idênticas,

para estimação das componentes de fluxo Ψ_{ds}^{s} e Ψ_{qs}^{s}. Na rede (Figura 4-2), Z^{-1} indica o atraso de uma amostra. As entradas da rede são definidas como um vetor $u(k)\ Mx1$, na figura M=3, aplicado em um instante discreto k, as saídas são definidas como um vetor $Y(k+1)\ Nx1$, na figura N=2. W_{11} a $W_{N(N+M)}$ são pesos da rede.

A equação da RNN relacionando entrada e saída [5], é descrita por

$$U(k) = [U_1(k) \ldots U_{(N+M)}(k)]^T = [Y_1(k) \ldots Y_N(k)\ u_1(k) \ldots u_M(k)]^T \qquad (4\text{-}11)$$

A equação de um neurônio j no tempo discreto k é dada por [5]

$$net_j(k) = \sum_{i=1}^{N+M} W_{ji}(k) U_i(k) \qquad (4\text{-}12)$$

ou na forma matricial

$$\begin{bmatrix} net_1 \\ \vdots \\ \vdots \\ net_N \end{bmatrix} = \begin{bmatrix} W_{11} & \cdots & W_{1(N+M)} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ W_{N1} & \cdots & W_{N(N+M)} \end{bmatrix} \begin{bmatrix} U_1 \\ \vdots \\ \vdots \\ U_{(N+M)} \end{bmatrix} \qquad (4\text{-}13)$$

A saída desejada da RNN $Y(\ k\ +\ 1\ )$ é dada por $Y_j(\ k\ +\ 1\ ) = \gamma\ (\ net_j(\ k\ ))$, onde $\gamma\ (.)$ é a função de ativação.

A rede neural recorrente (RNN) foi treinada *off-line* por um algoritmo baseado em Filtro de Kalman Estendido (EKF), o qual foi desenvolvido no ambiente do programa MATLAB. Os dados, exemplo para treinamento, foram gerados por simulação. O treinamento foi feito para a freqüência de 60 Hz,

desta forma a constante de tempo (τ) dos componentes do filtro e o ganho de compensação da amplitude (G) são constantes.

Para que o PCLPF opere em uma faixa de freqüências e não somente em 60 Hz, é necessário o treinamento da rede em toda a faixa de freqüências. Sendo a faixa de freqüências conhecida, o treinamento pode ser feito *off-line*, onde os pesos W resultantes do treinamento para cada freqüência são colocados em uma tabela (*lookup table*), para serem lidos pela rotina de programa que implementa o PCLPF. A cada alteração na freqüência do sistema, novos valores W devem ser lidos da tabela e carregados no PCLPF.

4.3 Algoritmo de Treinamento da Rede Neural Recorrente por Filtro de Kalman Estendido (EKF)

Algoritmo baseado em EKF tem mostrado méritos significativos para o treinamento de redes neurais do tipo *feedforward* e também de redes neurais recorrentes (RNN). O treinamento de rede neural por algoritmo gradiente descendente é mais usual do que o treinamento por algoritmo baseado em filtro de Kalman, mas com este último, tanto os dados para o treinamento quanto o tempo total de treinamento podem ser substancialmente reduzidos [11]. Computacionalmente, este algoritmo é mais simples que os algoritmos gradiente descendentes embora ele requeira a mesma derivada. A estimação dos estados é computada recursivamente, isto é, cada estimativa do estado é computada usando-se a última estimativa e o dado disponível atualmente, sendo que o único dado que requer armazenamento é a estimativa anterior. O algoritmo EKF não necessita processamento em lote de dados, e portanto, é um algoritmo apropriado para treinamento *on-line*, embora neste trabalho o treinamento tenha sido feito *off-line*.

O algoritmo EKF é bem conhecido para identificação de parâmetros e para problemas estimação de estados na presença de ruído [6]. O treinamento da RNN para determinar os pesos desconhecidos também pode ser visto como um problema de estimação de parâmetros. Aqui, o problema envolve a computação de derivadas das saídas da rede em relação aos pesos treináveis os quais são para ser ajustados. O algoritmo de treinamento é formulado como um problema de mínimos quadrados ponderados, onde o vetor de erro é a diferença entre as funções dos nós da saída da rede e os valores desejáveis dessas funções.

A idéia do treinamento é mostrada pela Figura 4-4, onde o filtro passa-baixas em cascata programável (PCLPF), fornece em sua saída o sinal desejado $d(k)$, no instante de tempo k. O sinal da saída da rede neural recorrente, $h(k)$, é então subtraído do sinal desejado, $d(k)$, gerando o erro $\xi(k)$. O erro é realimentado para o algoritmo baseado em filtro de Kalman, gerando os novos pesos W da rede neural. A rede gera o novo sinal $h(k)$ e o processo se repete até que o erro seja zerado.

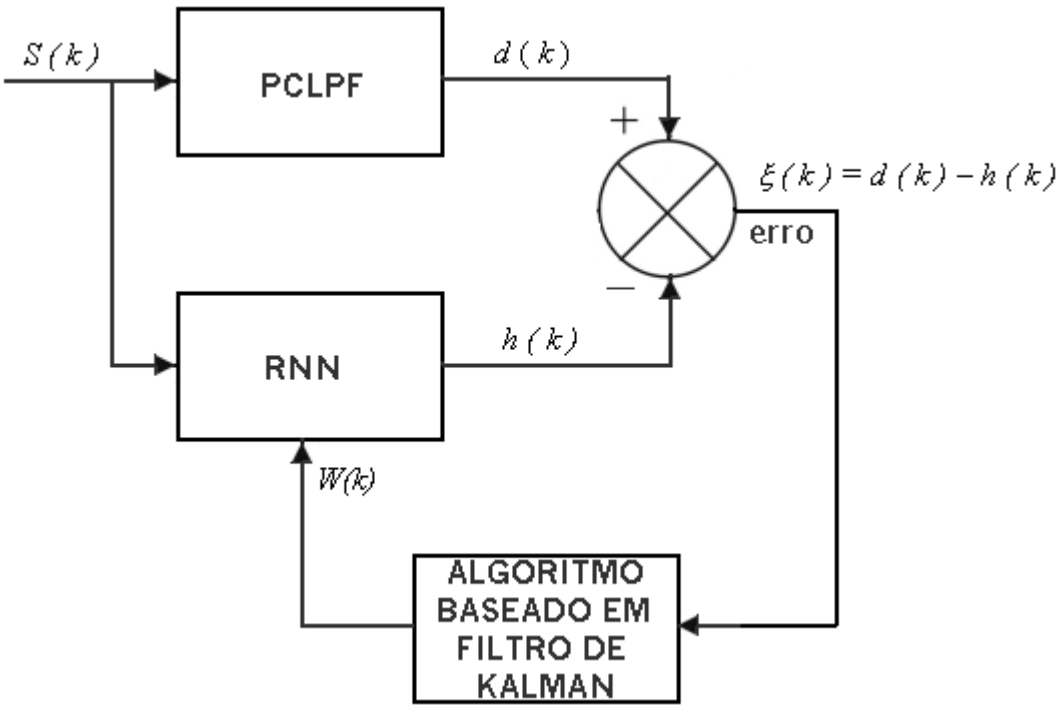

Figura 4-3 – Treinamento da Rede Neural Recorrente (RNN)

O vetor desejável no tempo *k* é dado por

$$d(k) = [\, d_1(k) \ldots d_N(k) \,]^T \qquad (4\text{-}14)$$

onde o vetor é de tamanho *N*. Fazendo a saída $y(k)$ de uma rede ser representada pelo vetor $h(k)$, do mesmo tamanho do vetor de erro, este vetor de erro fica

$$\xi(k) = d(k) - h(k) \qquad (4\text{-}15)$$

A função de custo $E(k)$ para treinamento da rede neural é dada por

$$E(k) = \frac{1}{2}\xi(k)^T S(k)\, \xi(k) \qquad (4\text{-}16)$$

onde $S(k)$ = matriz de pesos definitiva não negativa definida pelo usuário.

Os pesos treináveis da RNN podem ser organizados em um vetor M-dimensional $W(k)$. O algoritmo EKF atualiza os pesos da rede e aproxima a matriz de covariância de erro $P(k)$, a qual modela a correlação entre cada par de pesos na rede.

O algoritmo trabalha de tal maneira que em um tempo discreto k, os sinais de entrada e as saídas dos nós recorrentes, são propagados através da rede e a função $h(k)$ é calculada. O vetor de erro $\xi(k)$ é computado e avaliado para estimar os correntes pesos W. As derivadas dinâmicas de cada componente de $h(k)$ são formadas com relação aos pesos da rede. Estas derivadas são organizadas na forma de uma matiz $MxN\ H(k)$ como se segue:

$$H(k) = \begin{bmatrix} \frac{\partial(net_1)}{\partial W_1} & \cdots & \frac{\partial(net_N)}{\partial W_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial(net_1)}{\partial W_m} & \cdots & \frac{\partial(net_N)}{\partial W_m} \end{bmatrix} \quad (4\text{-}17)$$

onde $net_1 \ldots net_N$ = respectivo neurônio de saída. Então, $W(k)$ e $P(k)$ são atualizados usando as seguintes equações de recursão:

$$A(k) = [(\eta(k)S(k))^{-1} + H(k)^T P(k)H(k)]^{-1} \quad (4\text{-}18)$$

$$K(k) = P(k)H(k)A(k) \quad (4\text{-}19)$$

$$\hat{w}(k+1) = \hat{w}(k) + K(k)\xi(k) \quad (4\text{-}20)$$

$$P(k+1) = P(k) - K(k)H(k)^T P(k) + Q(k) \quad (4\text{-}21)$$

onde $\eta(k)$ = parâmetro de aprendizagem escalar e $Q(k)$ = uma matriz de covariância diagonal que provê um mecanismo para atenuar a interferência do ruído nos sinais envolvidos no processo de treinamento [6]. A presença da matriz $Q(k)$ ajuda a evitar divergências numéricas do algoritmo e também elimina a parada em um mínimo local.

A seqüência de treinamento da rede pelo filtro de Kalman é mostrado na Figura 4-4.

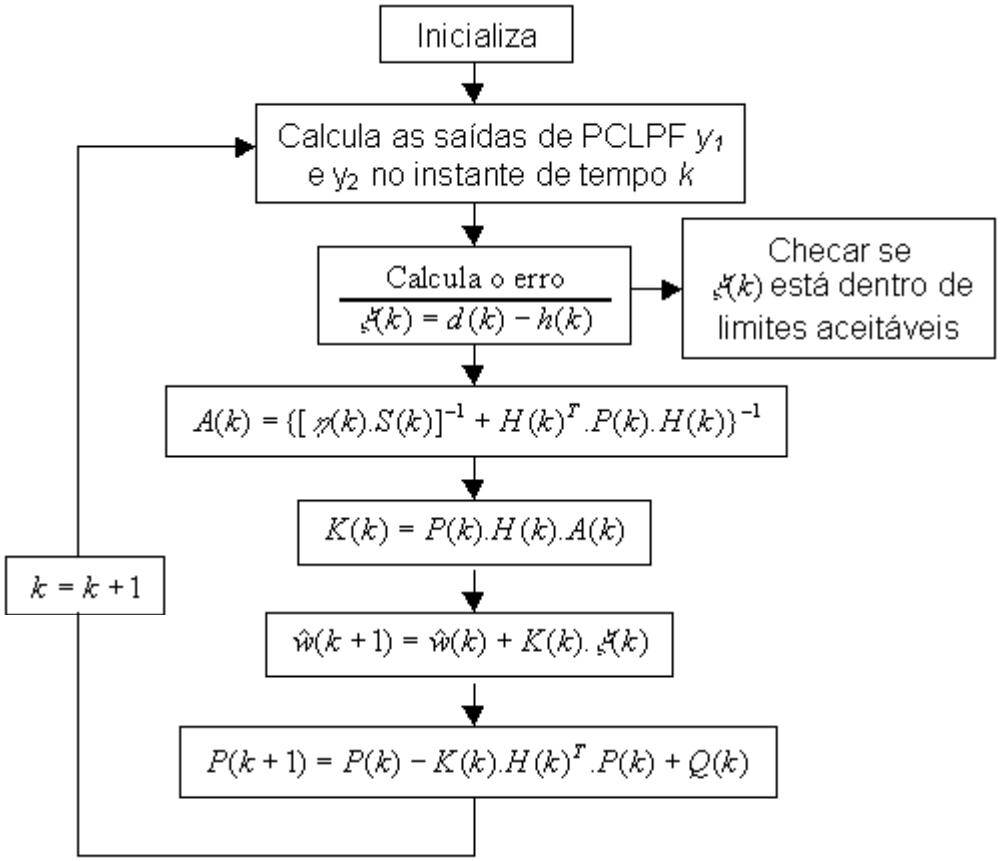

Figura 4-4 - Seqüência de computações no algoritmo EKF para o treinamento do PCLPF.

4.4 Aplicação do Procedimento de Treinamento da Rede Neural Recorrente ao PCLPF

O algoritmo EKF para treinamento de uma rede neural recorrente será agora aplicado ao PCLPF de dois estágios descrito na seção 4.2.

Definindo $net_1 = y_1$ e $net_2 = y_2$ na Figura 4-2, a equação (4-13) pode ser escrita na forma

$$net_1 = W_{11}\, y_1\,(k-1) + W_{13}\, u\,(k-1) \qquad (4\text{-}22)$$

$$net_2 = W_{21}\, y_1\,(k-1) + W_{22}\, y_2\,(k-1) \qquad (4\text{-}23)$$

As derivadas parciais de net_1 e net_2 em relação aos pesos são dados por

$$\frac{\partial(net_1)}{\partial W_{11}} = y_1(k-1) \qquad \frac{\partial(net_1)}{\partial W_{21}} = 0$$

$$\frac{\partial(net_1)}{\partial W_{12}} = 0 \qquad \frac{\partial(net_1)}{\partial W_{13}} = u(k-1)$$

$$\frac{\partial(net_2)}{\partial W_{11}} = 0 \qquad \frac{\partial(net_2)}{\partial W_{21}} = y_1(k-1) \qquad (4\text{-}24)$$

$$\frac{\partial(net_2)}{\partial W_{22}} = y_2(k-1) \qquad \frac{\partial(net_2)}{\partial W_{21}} = 0$$

Contudo, a matriz $H(k)$ em (4-17) pode ser expressa como

$$H(k) = \begin{bmatrix} y_1(k-1) & 0 \\ 0 & y_1(k-1) \\ 0 & y_2(k-1) \\ u(k-1) & 0 \end{bmatrix} \qquad (4\text{-}25)$$

Neste ponto, a única informação desconhecida necessária para rodar o algoritmo EKF é o vetor de erro, o qual pode ser calculado a partir da saída desejada do filtro e a atual saída RNN, por exemplo

$\xi(k) = d(k) - h(k)$ onde

$$d(k) = \begin{bmatrix} y_{1d}(k) \\ y_{2d(k)} \end{bmatrix} \qquad \text{e} \qquad h(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} \qquad (4\text{-}26)$$

Toda a informação necessária para rodar o conjunto de equações recursivas (31) – (34) relacionadas ao algoritmo EKF estão agora disponíveis. A Figura 4-4 mostra a seqüência de computações no algoritmo EKF para o treinamento do PCLPF.

A resposta da ANN baseada em rede neural recorrente (RNN) comparada com a resposta de um PCLPF baseado em alguma técnica de processamento digital de sinais, apresenta comportamento idêntico em regime permanente, mas mostra superioridade na resposta transitória [5] e [6].

4.5 Estimação do Conjugado a partir dos Vetores de Fluxo do estator

Os vetores de fluxo de campo estacionário do estator Ψ_{ds}^{s} e Ψ_{qs}^{s}, equações (4-1) e (4-2), que foram estimados pelo PCLPF baseado em rede neural recorrente (Figura 4-1), podem agora serem utilizados para estimar o conjugado dinâmico.

Com os vetores de fluxo de campo estacionário, algumas componentes podem ser calculadas e são representadas pelas equações a seguir.

Fluxo do estator dado por

$$\hat{\Psi}_s = \sqrt{(\Psi_{ds}^s)^2 + (\Psi_{qs}^s)^2} \qquad (4\text{-}27)$$

Ângulo θ_e é definido por

$$\theta_e = \text{sen}^{-1}\left(\frac{\Psi_{qs}^s}{\hat{\Psi}_s}\right) \qquad (4\text{-}28)$$

O seno e coseno de θ_e podem ser determinados pelas equações

$$\text{sen}\,\theta_e = \frac{\Psi_{qs}^s}{\hat{\Psi}_s} \quad \text{e} \qquad (4\text{-}29)$$

$$\cos\theta_e = \frac{\Psi_{ds}^s}{\hat{\Psi}_s} \qquad (4\text{-}30)$$

Finalmente são obtidas as correntes de campo girante do estator i_{ds} e i_{qs} e também os fluxos de campo girante do estator Ψ_{ds} e Ψ_{qs}, pelas equações

$$i_{ds} = i_{qs}^s \cos\theta_e - i_{ds}^s \,\text{sen}\,\theta_e \qquad (4\text{-}31)$$

$$i_{qs} = i_{qs}^s \,\text{sen}\,\theta_e + i_{ds}^s \cos\theta_e \qquad (4\text{-}32)$$

$$\Psi_{ds} = \Psi_{qs}^s \cos\theta_e - \Psi_{ds}^s \,\text{sen}\,\theta \qquad (4\text{-}33)$$

$$\Psi_{qs} = \Psi_{qs}^s \,\text{sen}\,\theta_e + \Psi_{ds}^s \cos\theta \qquad (4\text{-}34)$$

Multiplicando-se o fluxo do estator pela corrente do estator, o conjugado é determinado pela seguinte equação:

$$T_e = \frac{3P}{4}\left[\Psi_{ds} i_{qs} - \Psi_{qs} i_{ds}\right] \qquad (4\text{-}35)$$

onde P é o número de pares de pólos da máquina.

A Figura 4-5 mostra as equações e a seqüência de cálculo necessários à determinação do conjugado, que foram implementados na rotina em linguagem *assembly*.

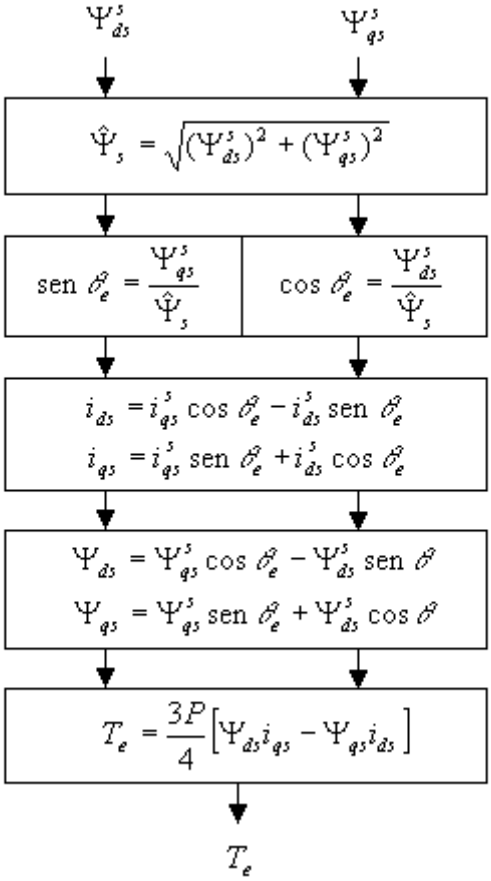

Figura 4-5 - Seqüência de Cálculo para Estimar o Conjugado

5- DSP 56000 DA MOTOROLA

Os DSPs mais modernos utilizam o recurso de paralelismo, estabelecendo diversas unidades de controle separadas e independentes, e até mesmo processadores múltiplos, de modo a multiplicar a capacidade total de processamento em um único *chip*. Assim, a capacidade de processamento de um DSP depende, basicamente, de duas variáveis: a freqüência de clock máxima e dos recursos inerentes à arquitetura empregada.

A família de processadores digitais de sinais de ponto fixo DSP56K tem um núcleo de processamento central em comum. Na área de expansão ao redor do módulo central, cada membro incorpora diversos periféricos embutidos e configurações de memória interna. A Figura 5-1 mostra um diagrama em blocos do DSP56002, incluindo o módulo de processamento central, as memórias internas e os seus periféricos.

5.1 Características Principais do DSP56002

O DSP 56002 trabalha com um barramento de dados de 24 *bits*, provendo uma faixa dinâmica de 144 dB. Como os acumuladores (A e B) são de 56 *bits* é possível uma escala de até 336 dB.

Algumas características do DSP 56002 podem ser destacadas:

- 33 MIPS – ciclo de instrução de 30,3 ns a 66 MHz.
- 198 MOPS máx. a 66 MHz.
- FFT complexa de 1024 pontos em 59.898 *clocks*.
- Dois acumuladores de 56 bits, incluindo byte de extensão.
- Instruções MAC de 24 $\times$ 24 bits em um ciclo de máquina (2 *clocks*).
- Multiplicação de precisão dupla (48 $\times$ 48 bits) com resultado em 96 bits em 6 ciclos.

- Adição / subtração de 56 bits em 1 ciclo.
- Suporte de hardware para FFT em bloco de ponto flutuante.
- Implementação de DO *loops* aninhados em *hardware*.
- Quatro barramentos de dados internos de 24 bits e três barramentos de endereço internos de 16 bits .
- RAM de programa de 512 x 24 bits.
- Duas RAMs de dados de 256 x 24 bits. (Memória X e memória Y).
- Porta de interface serial síncrona (SSI).
- Porta de interface serial assíncrona (SCI).
- 24 pinos de entrada / saída de uso geral. (I/O).
- Contador de eventos / temporizador de 24 bits.
- Interface Hospedeira (HI) de 8 bits com suporte a acesso direto à memória.
- Três pinos de interrupção (um não mascarável).
- Porta para Emulação no próprio Chip (OnCE) para depuração.
- Sintetizador de freqüências com PLL para a lógica interna, programável por *software*.
- Modos de economia de energia *Wait* e *Stop*.
- Lógica HCMOS estática, operando em 66 MHz ou 40 MHz até DC.
- Alimentação de 5 V (DSP56002) ou 3,3 V (DSP56L002).

O diagrama em blocos do DSP pode se visto na Figura 5-1.

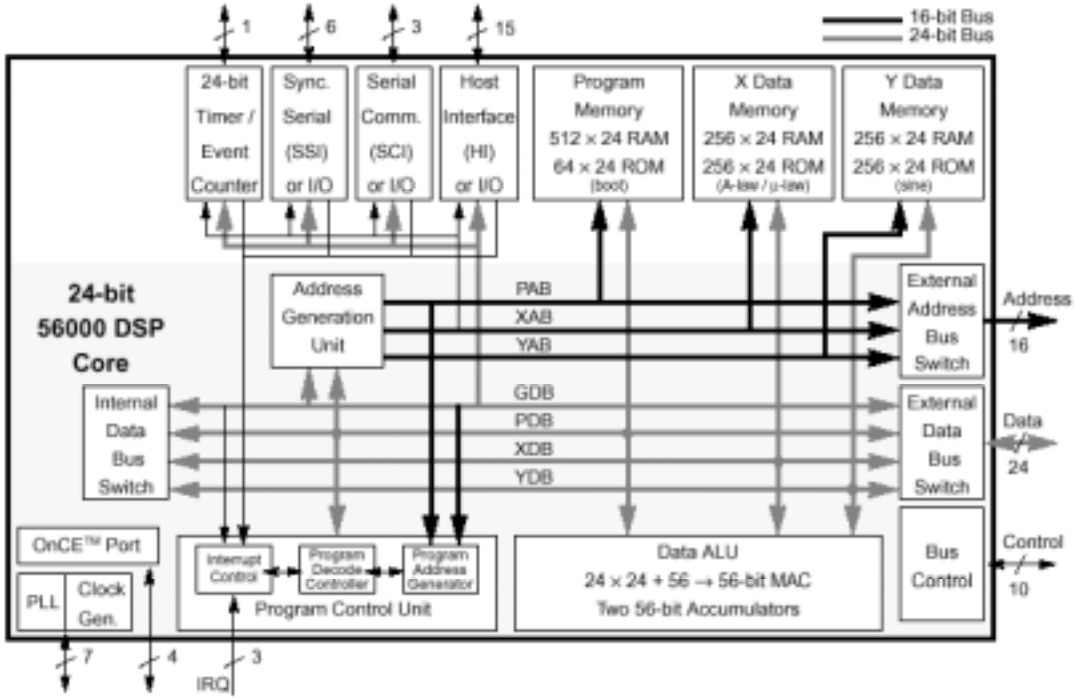

Figura 5-1 – Diagrama em Blocos do DSP56002

5.1.1. Gerador de Clock com PLL

O circuito PLL (*Phase Locked Loop*) permite que o DSP utilize qualquer fonte externa de clock disponível, para a geração do clock interno em freqüência máxima, mantendo o processamento em alta velocidade. Com baixas freqüências de clock de entrada a interferência eletromagnética gerada pelo sistema diminui e a possibilidade de oscilar em diferentes freqüências reduz o custo, pois elimina a necessidade de sistemas de osciladores adicionais.

O PLL realiza as operações de multiplicação de freqüência e eliminação de desalinhamentos nas temporizações (*skew*), e fornece, também, uma saída de clock sincronizada com o clock interno.

5.1.2. Unidade Lógica e Aritmética de Dados

A ALU é composta de quatro registros de entrada de 24 bits, dois acumuladores de 48 bits, dois registros de extensão dos acumuladores de 8 bits, um deslocador do acumulador, dois circuitos deslocadores/limitadores de saída, e uma unidade multiplicadora/acumuladora (MAC) paralela, capaz de operar em ciclo único.

A ALU executa qualquer uma das seguintes operações em um único ciclo de instrução: multiplicação, multiplicação e acumulação positiva ou negativa, arredondamento convergente, multiplicação e acumulação positiva ou negativa com arredondamento convergente, adição, subtração, uma iteração de divisão, uma iteração de normalização, deslocamento, e operações lógicas.

5.1.3. Registros de Entrada (X1, X0, Y1, Y0)

Estes registros operam como buffers de entrada entre os barramentos de dados XDB e YDB, e a unidade MAC, fornecendo os operandos de dados para a ALU. Os novos operandos podem ser carregados para a próxima instrução enquanto a instrução corrente utiliza o conteúdo atual dos registros. Os registros de entrada podem ser tratados como quatro registros de 24 bits independentes, ou como dois registros de 48 bits, concatenando-se X1:X0 e Y1:Y0.

5.1.4. Unidade Lógica e MAC

Para as instruções aritméticas, a unidade aceita até três operandos e fornece na saída um resultado de 56 bits na forma: (extensão : produto mais significativo : produto menos significativo, EXT:MSP:LSP). A operação da

unidade MAC ocorre independentemente e em paralelo com as atividades dos barramentos XDB e YDB. Os resultados de todas as instruções aritméticas são operandos válidos em 56 bits, podendo ser carregados nos acumuladores A ou B na forma EXT:MSP:LSP (A2:A1:A0 ou B2:B1:B0). Isto significa que todos os resultados parciais de uma operação MAC são mantidos com uma precisão de 56 bits. Quando uma instrução determina que o resultado seja armazenado como um operando de 24 bits, o LSP pode ser simplesmente truncado ou, opcionalmente, arredondado para o MSP.

A unidade lógica realiza operações lógicas AND, OR, EOR e NOT, de 24 bits, nos registros da ALU, operando nos dados na porção MSP do acumulador. As porções LSP e EXT do acumulador não são afetadas.

5.1.5. Acumuladores A e B

Cada um dos acumuladores A e B consistem em três registros concatenados (A2:A1:A0 e B2:B1:B0). A extensão de sinal de 8 bits (EXT) é armazenada em A2 ou B2, e é usada quando uma precisão maior que 48 bits é necessária; os 24 bits do MSP são armazenados em A1 ou B1, e os 24 bits do LSP em A0 ou B0. Na representação numérica da família DSP56K, os valores extremos que uma palavra de 24 bits pode assumir são -1 e $1 - 10^{-23}$. Se o resultado de uma operação ultrapassa estes limites, ocorre underflow ou overflow. Neste caso, os registros de extensão de 8 bits podem registrar o resultado de até 255 underflows e overflows, assinalando a ocorrência destes eventos por meio do bit V no registro de status.

A extensão de sinal é feita automaticamente quando o conteúdo do acumulador é carregado a partir de um operando menor, de 48 ou 24 bits. Um operando de 24 bits é escrito no MSP (A1 ou B1), o LSP (A0 ou B0) é completado com zeros, e o EXT (A2 ou B2) é estendido no sinal a partir do MSP.

Uma proteção por saturação ocorre quando o conteúdo total de 56 bits de A ou B é especificado para ser transferido através de XDB e YDB, fazendo com que, caso tenha ocorrido underflow ou overflow, o valor seja limitado a – 1 ou $1 – 10^{-23}$, respectivamente. O conteúdo de A ou B não são afetados.

5.1.6. Portas A, B e C

As Portas A, B e C consistem de interfaces versáteis, integrando diversas facilidades de hardware para as funções de entrada / saída de dados entre o DSP e o mundo externo. A Figura 12-3 do anexo 12.2.1 fornece uma visão geral destas portas.

A Porta A fornece uma interface para acesso à memória externa, permitindo a conexão com memórias e dispositivos externos. É composta de um barramento de endereço externo de 16 bits, um barramento de dados externo de 24 bits e um conjunto de sinais de controle.

A Porta B é uma porta de I/O de dupla função. Ela pode funcionar como uma interface de I/O de propósito geral de 15 bits (GPIO), com cada um dos pinos podendo ser programados individualmente como de entrada ou saída, sendo útil neste caso, para o controle geral de dispositivos externos. Alternativamente, a Porta B pode ser configurada como uma interface hospedeira bidirecional de 8 bits, incluindo sinais de controle e fornecendo, assim, um modo conveniente de conexão com um outro processador, pois, a interface emula o comportamento de uma memória estática.

A Porta C é uma porta de I/O de nove pinos, capaz de desempenhar três funções distintas (Figura 12-3). Três pinos podem ser configurados como I/O de propósito geral ou como uma interface de comunicação serial (SCI). Os demais seis pinos também podem ser configurados como pinos de entrada e saída de uso geral, ou como uma interface serial síncrona (SSI).

5.1.7. Representação Numérica Fracionária de Ponto Fixo (Formato Q)

O formato Q é uma representação numérica binária, em complemento dois, que estipula uma posição de *bit* fixa nas palavras, para demarcar a fronteira entre a parte inteira do número e a parte fracionária. A convenção utilizada para demarcar esta fronteira é a letra “Q” seguida do número de bits que compõem a sua parte fracionária. Assim, por exemplo, uma palavra binária de 16 bits em formato Q8 representa um número com parte fracionária de 8 bits, parte inteira de 7 bits e 1 bit de sinal (bit mais à esquerda). Deste modo, uma palavra de 16 bits em formato Q8 pode assumir o valor extremo positivo igual a $(2^7 - 1) + (1 - 2^{-8}) = 2^7 - 2^{-8} \cong 127{,}996$; e o valor extremo negativo é igual a $-(2^7 + 1) = -128$. Neste formato, o número hexadecimal \$0180 = 0000 0001 1100 0000 representa o número fracionário 1,75 e o número \$FE40 = 1111 1110 0100 0000 equivale a $-1{,}75$.

Observa-se que os números inteiros binários, sem parte fracionária, podem ser interpretados como se estivessem representados no formato Q0, enquanto que os números puramente fracionários, entre –1 e 1, serão representados no formato Q_{N-1}, onde N é o número total de bits das palavras. Em DSP, este último formato é a representação mais usual. O DSP56000 é um processador de 24 bits e, por isso, normalmente, os operandos são representados no formato Q23 e as palavras longas de 48 bits em Q47. A exceção são os conteúdos dos acumuladores de 56 bits que são compostos por três palavras distintas: a extensão de sinal de 8 bits (EXT), o produto mais significativo de 24 bits (MSP) e o produto menos significativo (LSP). Neste caso a palavra concatenada de 56 bits, EXT:MSP:LSP, é representada no formato Q47, de forma que ponto decimal fique sempre alinhado com o MSP dos demais registros.

As instruções aritméticas na família DSP56K são totalmente voltadas para a representação numérica fracionária e, por essa razão, os dados devem ser escalonados para esta representação. Ao assumir que os números são

fracionários, o DSP56002 justifica os operandos numéricos à esquerda. Por exemplo, ao se mover o valor $3F para um registro de 24 bits da ALU, resulta que o mesmo será carregado com o conteúdo de $3F0000, e não $00003F como em outros processadores convencionais. A maneira mais efetiva de se evitar problemas é escalonar todas as variáveis para a representação fracionária de ponto fixo, com os números limitados entre ± 1.

Os números com representação inteira ou fracionária são equivalentes sob o ponto de vista da adição e subtração, porém, se comportam de maneira diversa sob as operações de multiplicação e divisão. Maiores detalhes sobre a multiplicação e divisão podem ser encontrados em [12] e [13].

6- SIMULAÇÃO DO ESTIMADOR DE CONJUGADO

Para a simulação foi utilizado o programa SIMULINK que é parte integrante do programa MATLAB, onde foi simulado um motor de indução trifásico usando o modelo dq. Ao motor de indução foi aplicada uma seqüência de valores em função do tempo, para simular uma carga, variando o conjugado de carga. Amostras dos sinais trifásicos de tensão e corrente, foram retirados por amostragem e retenção com uma freqüência de amostragem de 8 kHz.

Antes de iniciar a simulação é necessário executar um arquivo de configuração para inicializar os parâmetros do modelo do MIT. Este arquivo de configuração, bem como mais detalhes sobre o modelo dq do MIT podem ser encontrados nas seções 12.4.1 e 1.1.1 respectivamente. Lembrando que o único parâmetro do MIT, necessário para a estimação do conjugado é a resistência do estator (Rs), os demais parâmetros do arquivo de configuração são parâmetros necessários apenas à simulação do MIT.

6.1 Diagrama Geral da Simulação

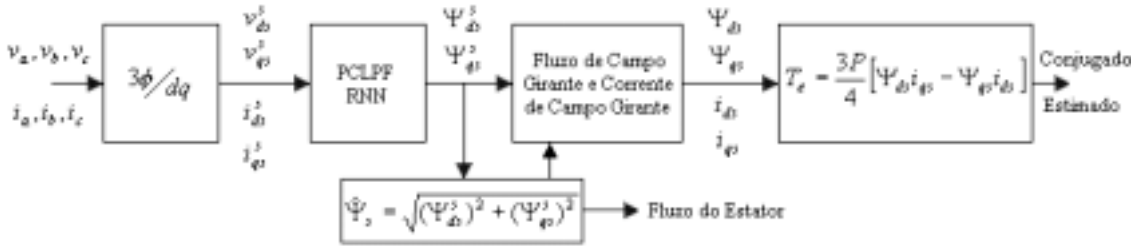

Figura 6-1 - Diagrama Geral da Simulação

A Figura 6-2 mostra a geração das tensões trifásicas V_{abc}. Estas tensões alimentam o modelo do motor de indução trifásico.

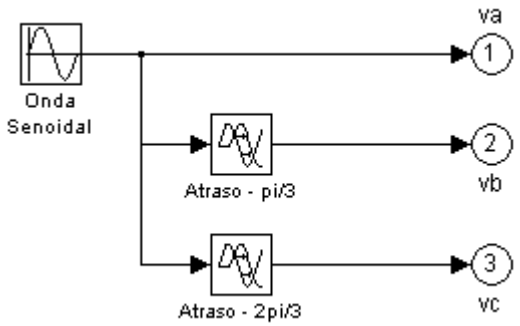

Figura 6-2 - Gerador do Sistema Trifásico

6.2 Amostragem e Retenção

As tensões e correntes de linha são amostradas ao mesmo tempo, na saída do modelo do motor de indução trifásico. A freqüência de amostragem utilizada é de 8 kHz (125 μs).

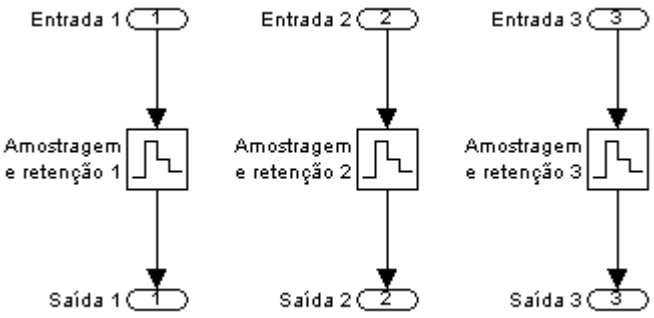

Figura 6-3 - Amostragem e Retenção

6.3 Conversão de Três Eixos para Dois Eixos (Três para Duas Fases)

Após a amostragem das tensões e correntes trifásicas, a uma taxa de 8kHz, os sinais são convertidos de um sistema de três eixos para um sistema de dois eixos, através da matriz:

$$\begin{bmatrix} v_{ds}^{s} \\ v_{qs}^{s} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} \quad (6\text{-}1)$$

Maiores detalhes sobre a conversão de três eixos para dois eixos foram tratados na seção 3.2.

A Figura 6-4 mostra a conversão de três para duas fases na simulação, que é a implementação da matriz (6-1). Os sinais V_{abc} e I_{abc} chegam à matriz de conversão 3 eixos para 2 eixos através de um multiplex 3 x 1. Após a conversão para duas fases o sinal é colocado em um demultiplexador 1 x 2 que disponibiliza os sinais em dois eixos equivalentes aos sinais em três eixos. Para a conversão I_{abc}-I_{dq} é utilizado um bloco idêntico, onde as entradas são as componentes da corrente em três fases (I_{abc}) e as saídas as componentes da corrente em duas fases (I_{dq}).

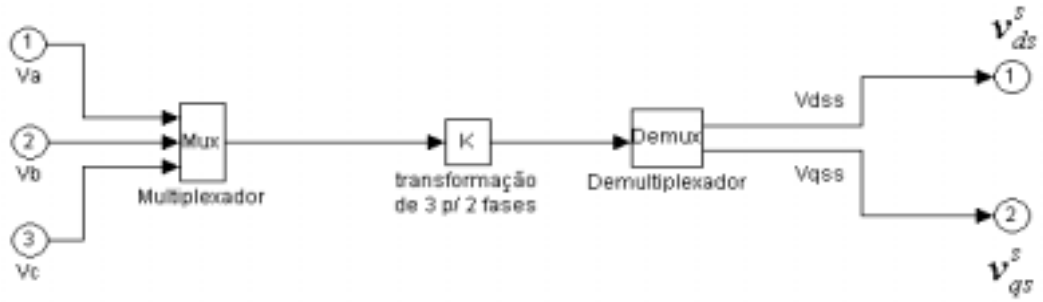

Figura 6-4 - Transformação 3 Fases para 2 Fases

Os sinais convertidos para um sistema de 2 eixos, sistema dq, são sinais defasados entre si de 90^0, mostrado na Figura 6-6, a componente q está adiantada 90^0 em relação à componente d. A amplitude da componente q é igual à amplitude da componente d, sendo estas amplitudes proporcionais às amplitudes dos sinais abc em três fases. As componentes i_{ds}^{s} e i_{qs}^{s} são mostrados na Figura 6-5 em todo o intervalo da simulação.

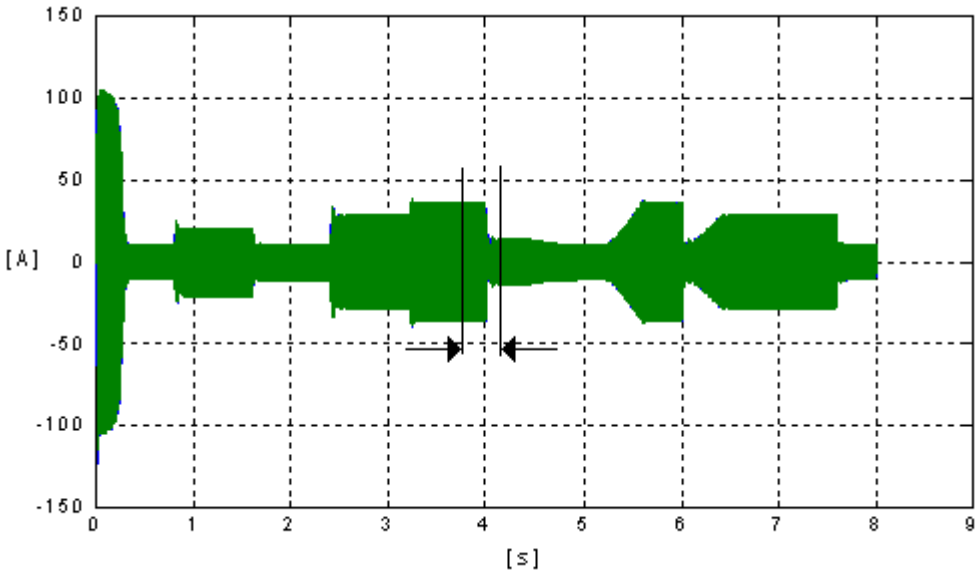

Figura 6-5 - Componentes i_{ds}^{s} e i_{qs}^{s} em todo o Intervalo da Simulação

Na Figura 6-6 é mostrado com *zoom* o intervalo em torno de 4[s], destacado na Figura 6-5, onde se pode observar a variação das correntes do estator, sistema dq, e a defasagem entre elas. Neste instante o conjugado de carga aplicado ao modelo do MIT passa de 80 N.m para 20 N.m.

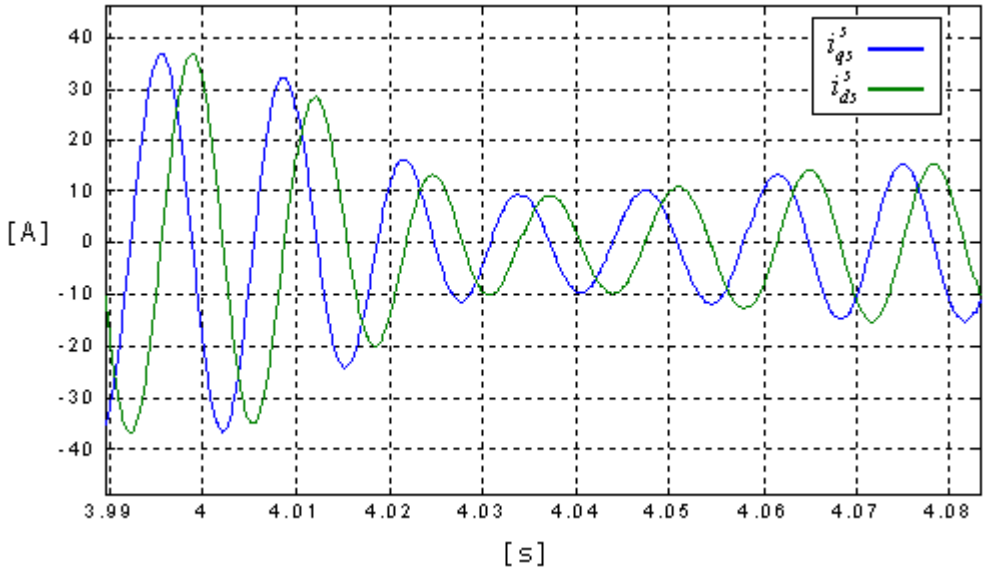

Figura 6-6 – Correntes do estator i^s_{ds} e i^s_{qs}

6.4 O Filtro Passa-Baixas PCLPF (Programmable Cascaded Low-Pass Filter)

Após a transformação dos sinais de corrente e tensão para um sistema de dois eixos, os sinais $v^s_{qs} - i^s_{qs} R_s$ e $v^s_{ds} - i^s_{ds} R_s$ são calculados. Estes sinais são então convertidos para vetores de fluxo de campo estacionário, por filtros passa-baixas (PCLPF) baseado em rede neural, equações (4.1) e (4.2).

A implementação de PCLPF foi feita pelo uso de ANN (*artificial neural network*) do tipo RNN *(recurrent neural network*). A rede neural foi treinada *off-line* por um algoritmo baseado em filtro de *Kalman* estendido, seção 4.3 e referências [4] e [14].

A Figura 6-7 mostra o diagrama da Rede Neural Recorrente (RNN), que exerce o papel de filtro passa-baixas na simulação.

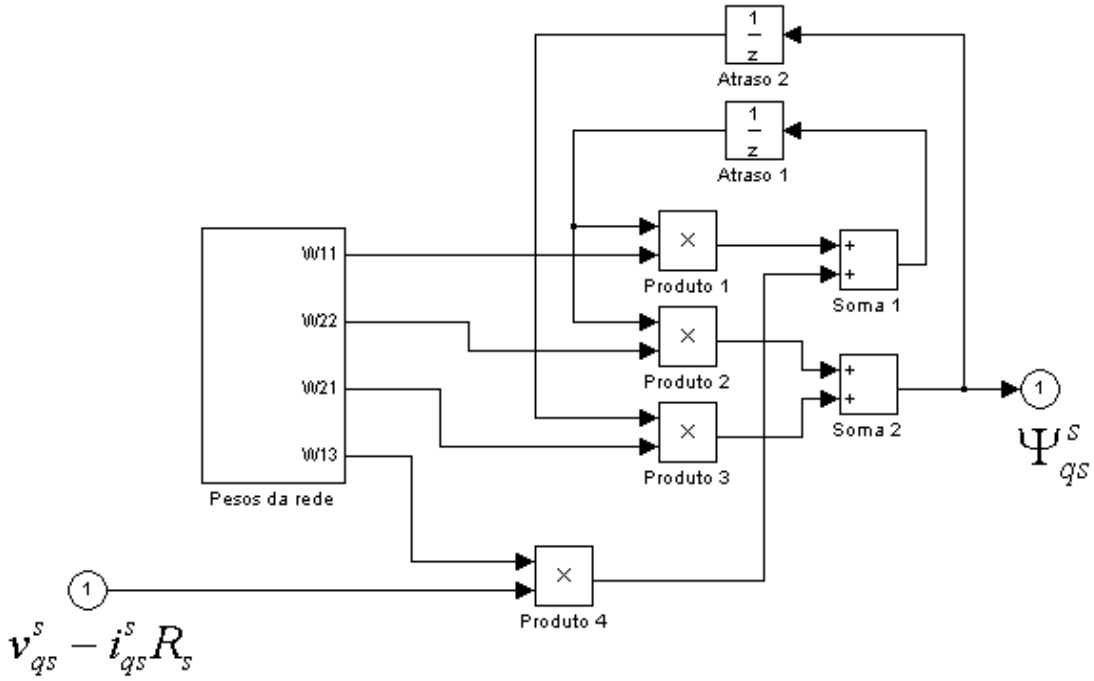

Figura 6-7 - ANN-RNN-PCLPF

Dois filtros passa-baixas idênticos são implementados para obtenção dos vetores de fluxo de campo estacionário, Ψ^s_{ds} e Ψ^s_{qs}. Um bloco para obtenção do vetor para a componente d, que é a componente produtora de corrente, o outro bloco para a componente q, responsável pela produção do conjugado. As equações do vetor de fluxo de campo estacionário são obtidas nas saídas dos filtros:

$$\Psi^s_{ds} = \int (v^s_{ds} - i^s_{ds} R_s) dt \quad \text{(6-2)}$$

$$\Psi^s_{qs} = \int (v^s_{qs} - i^s_{qs} R_s) dt \quad \text{(6-3)}$$

A Figura 6-8 mostra o sinal na saída do filtro passa-baixas. Este sinal é o vetor Ψ^s_{qs} representado pela equação (6-3) em um intervalo de 1,2 segundos.

O vetor de fluxo Ψ_{ds}^{s} é um sinal idêntico ao sinal da Figura 6-8, porém atrasado de 90^0.

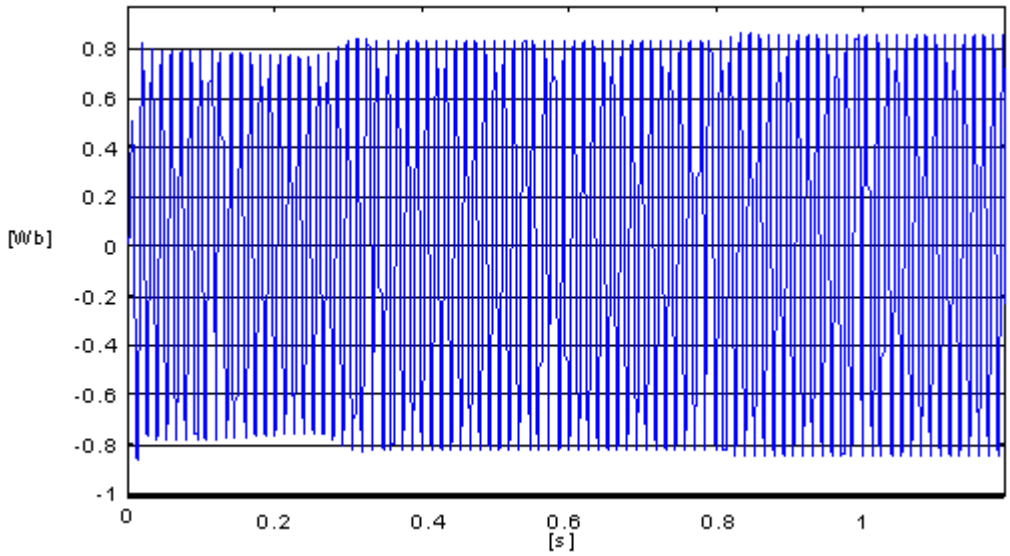

Figura 6-8 – Componente q do Fluxo do estator Ψ_{qs}^{s}, referência estacionária

Pela Figura 6-9 observa-se os sinais na entrada e saída do integrador (filtro passa-baixas PCLPF).

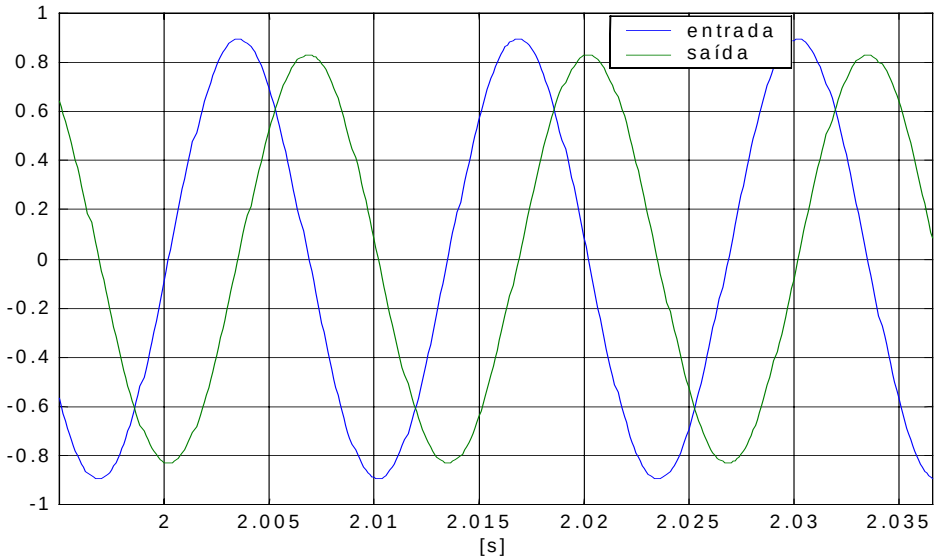

Figura 6-9 – Sinais de Entrada e Saída do Filtro Passa-Baixas

6.5 Cálculo do Conjugado Dinâmico a Partir dos Vetores de Fluxo de Campo Estacionário

Os sinais Ψ_{ds}^{s} e Ψ_{qs}^{s}, gerados pelos filtros passa-baixas, são entregues ao bloco que irá calcular o conjugado, bloco 2 (Figura 6-10). Este bloco implementa as equações de fluxo do estator ($\hat{\Psi}_s$), seno e coseno do ângulo θ_e, correntes de campo girante do estator (i_{ds} e i_{qs}), fluxo de campo girante do estator (Ψ_{ds} e Ψ_{qs}) e finalmente a equação do conjugado estimado. Estas equações estão descritas na seção 4.5.

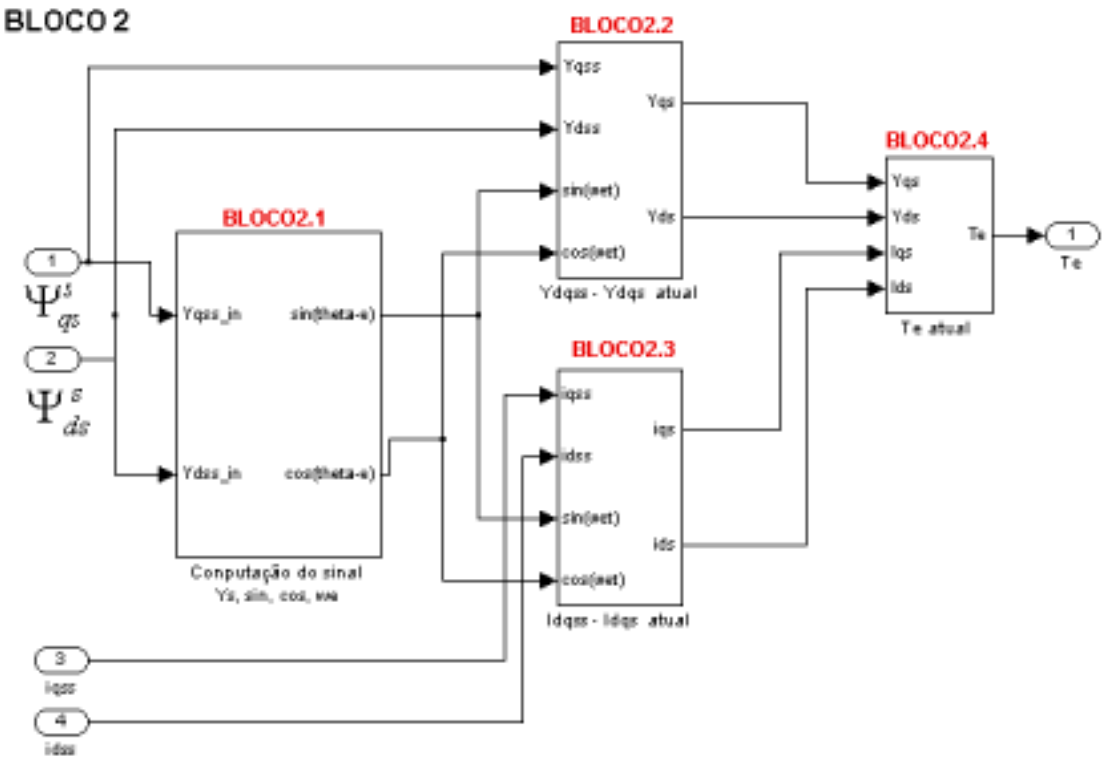

Figura 6-10 - Computação para a Estimação do Conjugado

A partir do fluxo do estator mostrado na Figura 6-11, são obtidos o seno e o coseno de θ_e, pelas equações abaixo:

$$\hat{\Psi}_s = \sqrt{(\Psi_{ds}^{s})^2 + (\Psi_{qs}^{s})^2} \qquad (6\text{-}4)$$

$$\operatorname{sen}\theta_e = \frac{\Psi_{qs}^s}{\hat{\Psi}_s} \qquad (6\text{-}5)$$

$$\cos\theta_e = \frac{\Psi_{ds}^s}{\hat{\Psi}_s} \qquad (6\text{-}6)$$

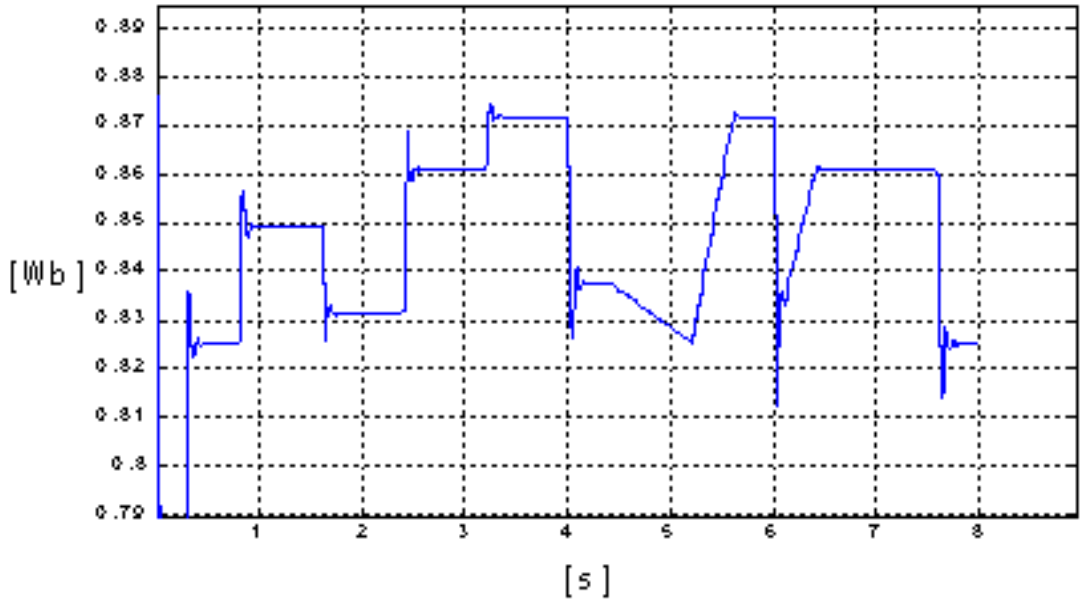

Figura 6-11 – Fluxo do estator, $\hat{\Psi}_s$, em um Intervalo de 8 Segundos

O seno e o coseno de θ_e serão agora utilizados para cálculo das correntes de campo girante do estator e para o cálculo dos fluxos de campo girante do estator.

As correntes de campo girante do estator são mostradas na Figura 6-13 e são calculadas pelo bloco 2.3 (Figura 6-12) através da computação das equações (6-7) e (6-8) a seguir:

$$i_{ds} = i_{qs}^s \cos\theta_e - i_{ds}^s \operatorname{sen}\theta_e \qquad (6\text{-}7)$$

$$i_{qs} = i_{qs}^s \operatorname{sen}\theta_e + i_{ds}^s \cos\theta_e \qquad (6\text{-}8)$$

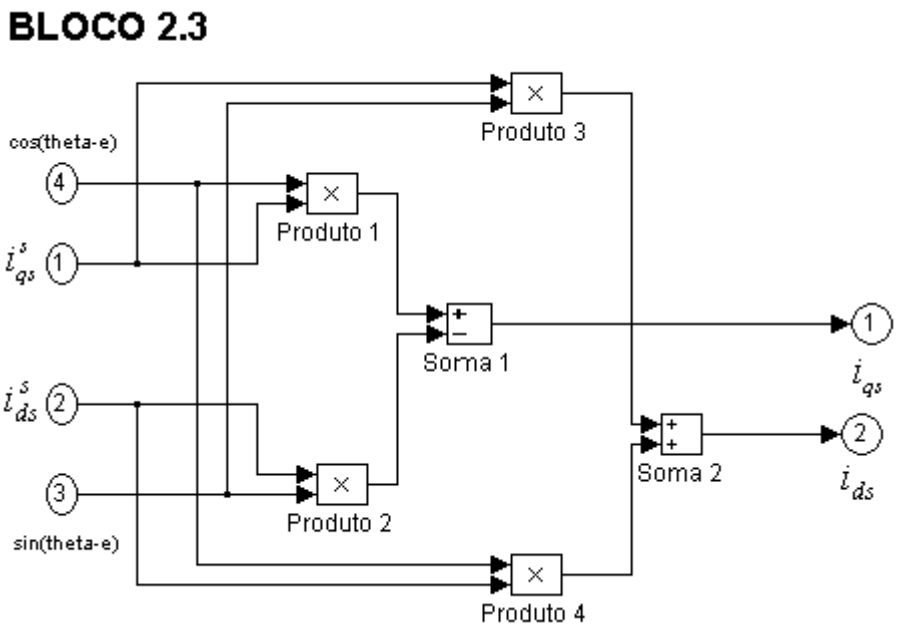

Figura 6-12 – Cálculo das Correntes de Campo Girante do Estator

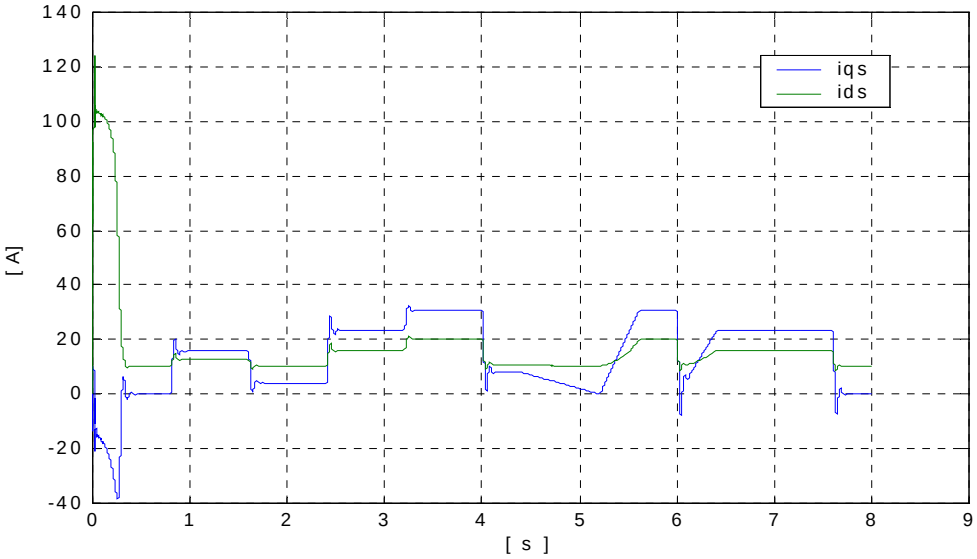

Figura 6-13 – Correntes de Campo Girante do Estator

A Figura 6-13 mostra que a componente d (i_{ds}), que é a componente produtora de fluxo, nunca atinge o valor zero, enquanto a componente produtora de conjugado i_{qs} tem valor zero em determinado instante. A corrente i_{ds} é a corrente de magnetização da máquina, portanto nunca atinge o valor zero, enquanto a máquina estiver ligada.

Os fluxos de campo girante do estator são calculados na simulação pelo bloco 2.2 (Figura 6-14).

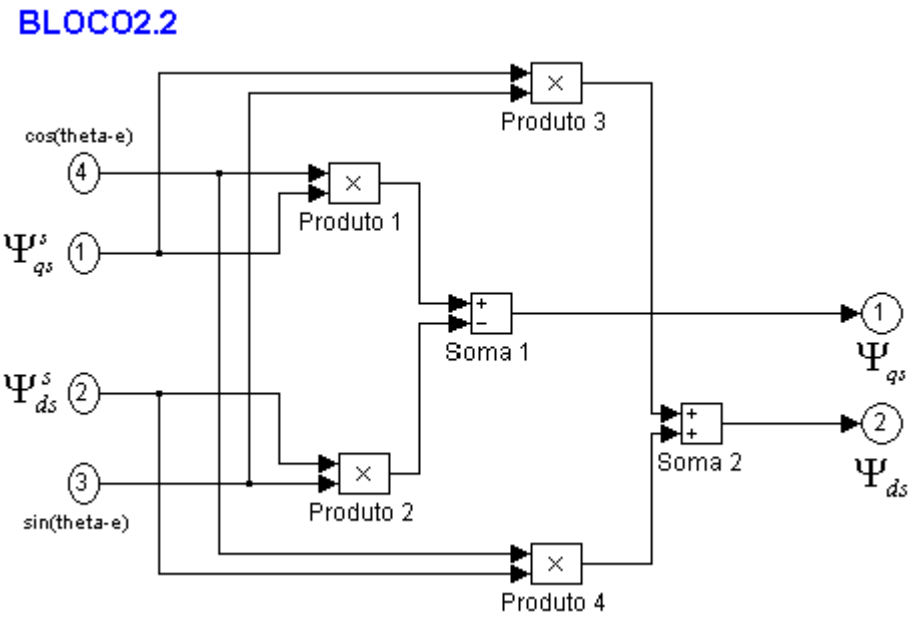

Figura 6-14 – Cálculo das Componentes do Fluxo de Campo Girante do Estator

Este bloco implementa as equações de fluxo de campo girante do estator através da computação das equações (6-9) e (6-10) a seguir:

$$\Psi_{ds} = \Psi_{qs}^{s} \cos\theta_e - \Psi_{ds}^{s} \operatorname{sen}\theta \quad (6\text{-}9)$$

$$\Psi_{qs} = \Psi_{qs}^{s} \operatorname{sen}\theta_e + \Psi_{ds}^{s} \cos\theta \quad (6\text{-}10)$$

O fluxo de campo girante do estator, componente d, pode ser observado na Figura 6-15, a componente q tem valor zero.

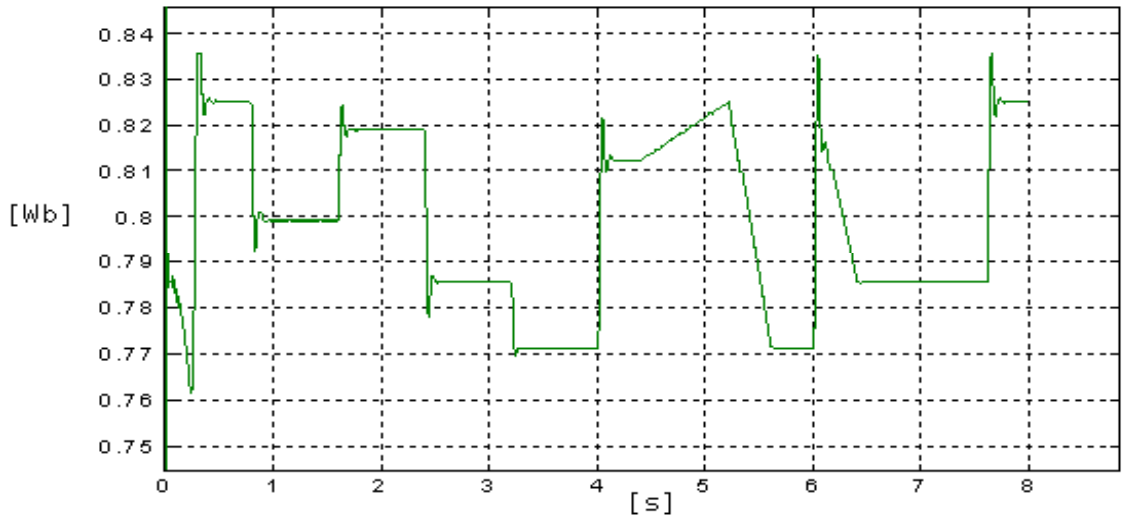

Figura 6-15 – Fluxo de Campo Girante do Estator, Componente d

Finalmente o conjugado dinâmico do motor de indução é calculado pelo bloco 2.4 (Figura 6-16).

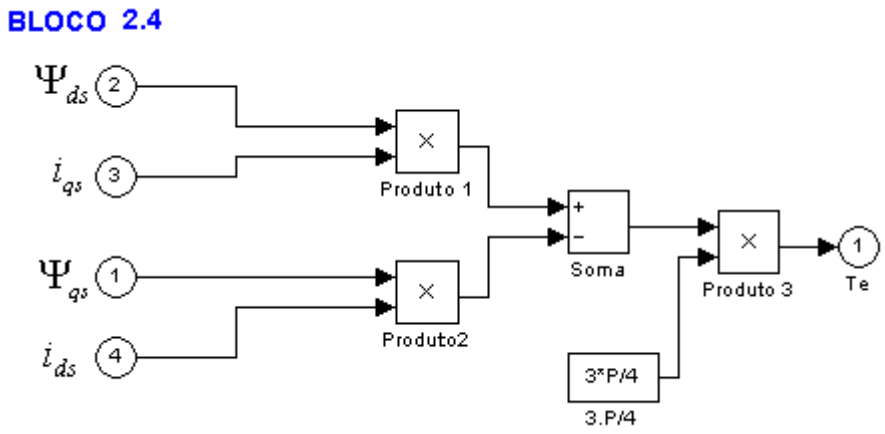

Figura 6-16 – Cálculo do Conjugado Dinâmico

A Figura 6-18 mostra o conjugado dinâmico estimado, em um intervalo de 8 segundos. Nesta figura também é mostrada a curva do conjugado de carga aplicado à máquina e o conjugado elétrico do modelo do motor de indução trifásico. A equação (6-11) representa o conjugado dinâmico do motor.

$$T_e = \frac{3P}{4}\left[\Psi_{ds} i_{qs} - \Psi_{qs} i_{ds}\right] \quad (6\text{-}11)$$

6.6 O Modelo do Motor de Indução Trifásico

O modelo do motor de indução utilizado é um modelo em dois eixos (dq). Sendo as componentes de corrente i_{qs} e i_{ds}, existe a necessidade da conversão para um sistema em três fases, que pode ser observada na Figura 6-17. Estas três componentes de corrente e as três componentes de tensão serão amostradas para a estimação do fluxo e cálculo do conjugado dinâmico. O diagrama completo do modelo do motor de indução trifásico pode ser visto no anexo 12.4.1.

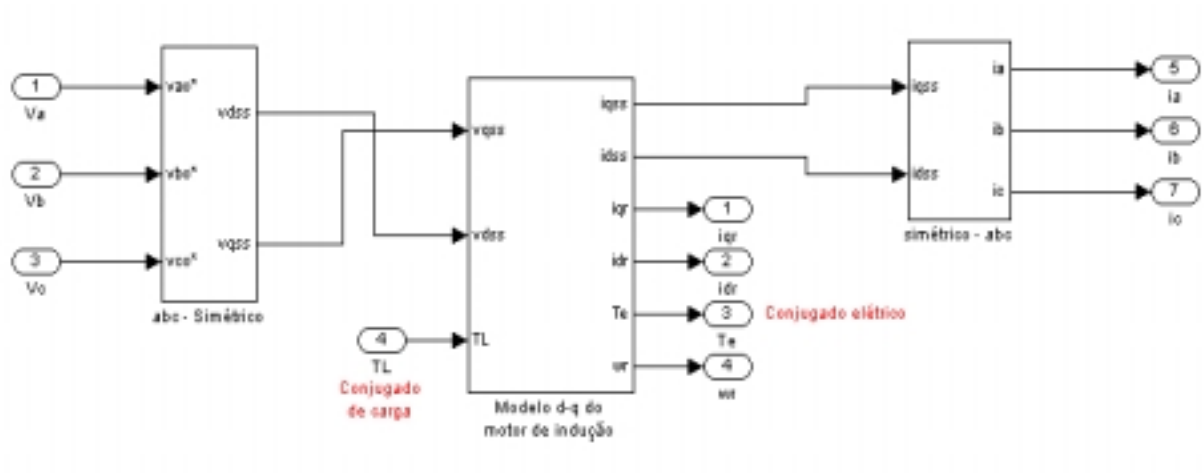

Figura 6-17 - Transformação das Componentes de Corrente do Modelo dq para o Modelo em Três Eixos

6.7 Resultados da Simulação

Uma variação de carga é aplicada ao motor de indução trifásico durante 8 segundos. A Figura 6-18 mostra a variação de carga aplicada à máquina, o conjugado gerado pelo modelo do motor e o conjugado estimado pelo método do filtro passa-baixas em cascata programável (PCLPF), implementado por rede neural recorrente (RNN), treinada por filtro de Kalman.

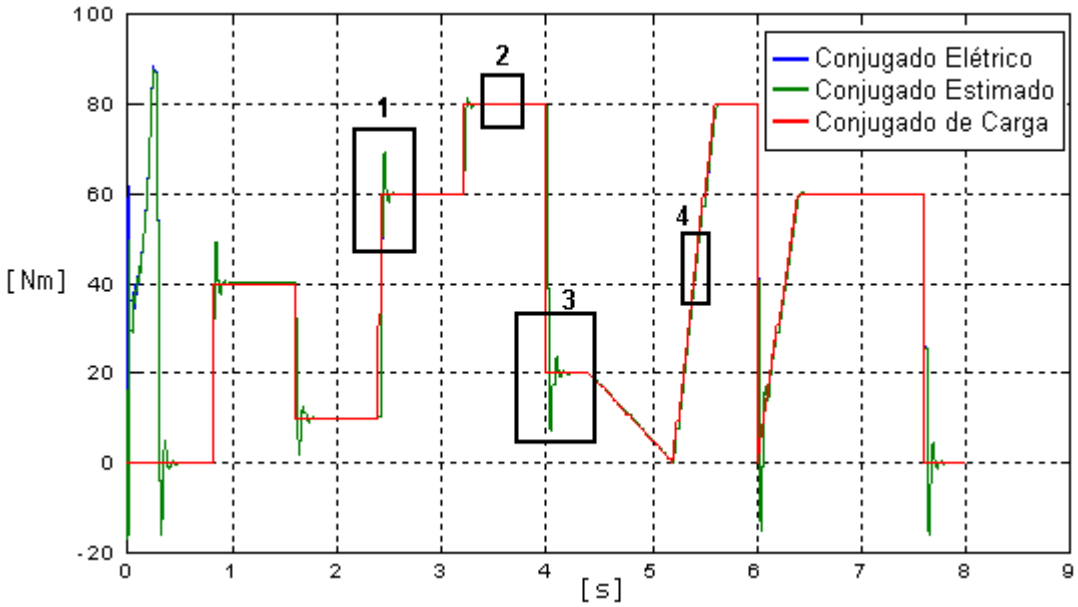

Figura 6-18 – Simulação do Conjugado Dinâmico do MIT

A variação de carga aplicada ao modelo do MIT é provocada por uma seqüência de repetição em função do tempo, gerada pelo programa SIMULINK.

Os três sinais da Figura 6-18 serão explorados nos quatro intervalos assinalados na figura.

A Figura 6-19 mostra o intervalo (1) ampliado, explorando o instante onde o conjugado de carga passa bruscamente de 10 [Nm] para 60 [Nm]. A curva do conjugado elétrico e a do conjugado estimado são coincidentes.

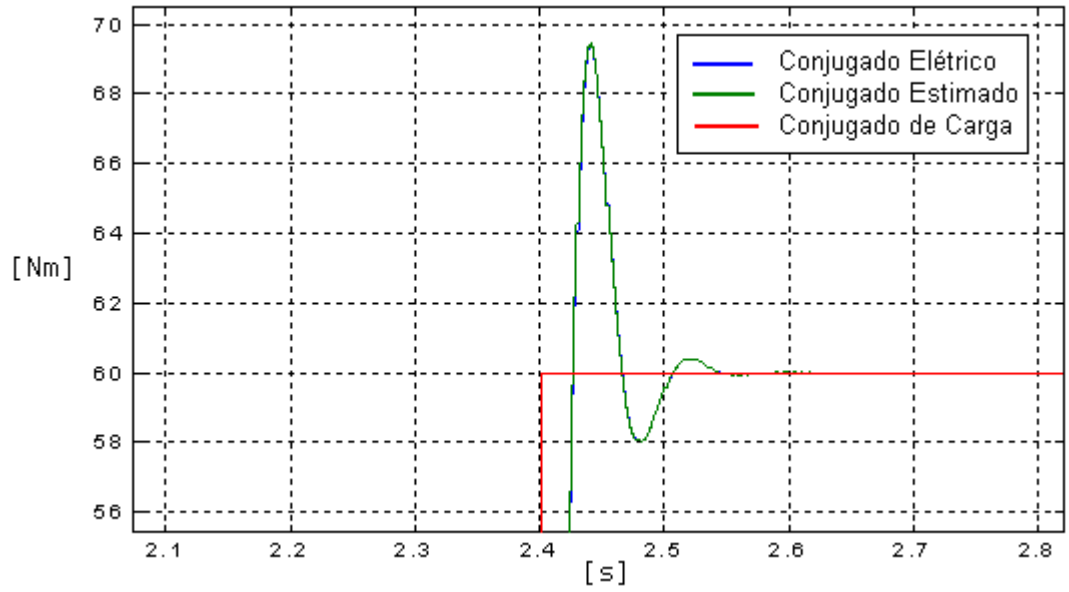

Figura 6-19 – Aumento do Conjugado de Carga

A Figura 6-20 mostra o intervalo (2) da curva do conjugado, é um intervalo onde não há variação do conjugado de carga. O erro entre o conjugado elétrico e o conjugado estimado está destacado na figura e não ultrapassa 10^{-3}.

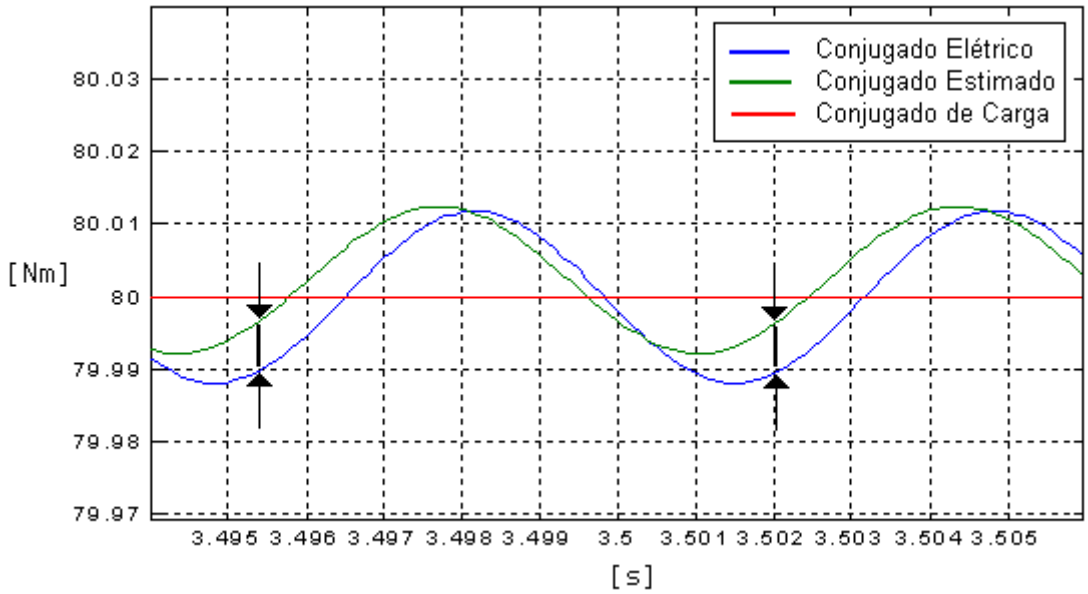

Figura 6-20 – Erro Entre o Conjugado Elétrico e o Conjugado Estimado

Uma variação brusca no conjugado de carga, agora uma diminuição, pode ser observada no intervalo (3) da curva do conjugado, este intervalo está destacado na Figura 6-21, onde o conjugado de carga passa de 80 [Nm] para 20 [Nm].

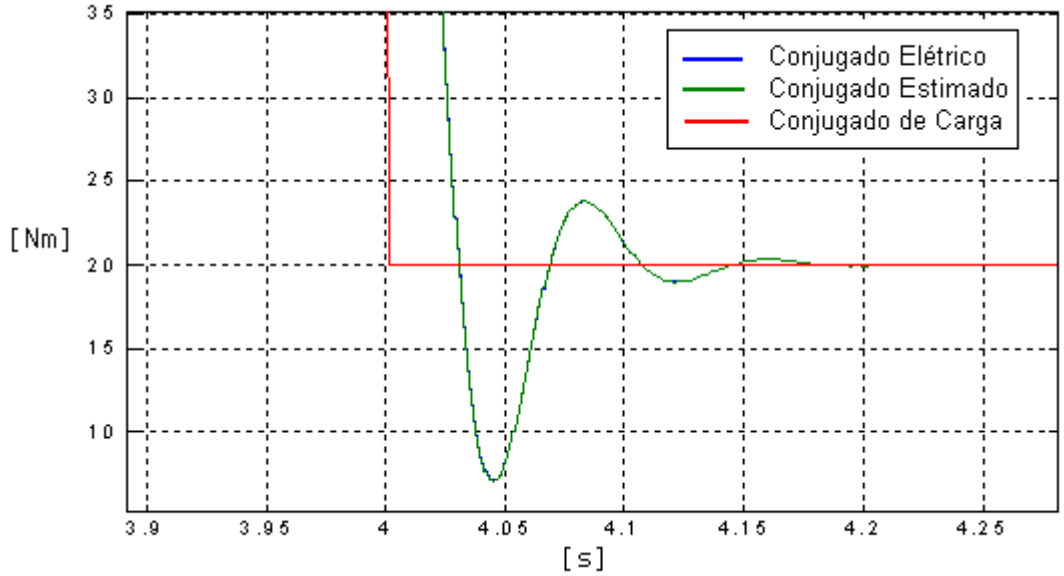

Figura 6-21 – Diminuição Brusca do Conjugado de Carga

No intervalo (4) da curva do conjugado mostrado na Figura 6-22, a variação do conjugado de carga é gradual.

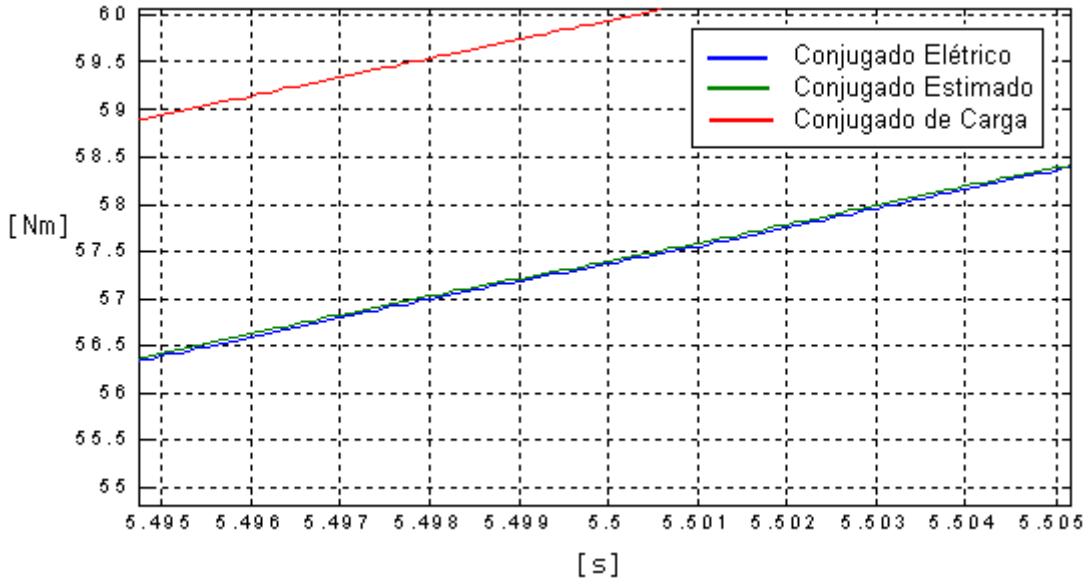

Figura 6-22 – Variação Gradual do Conjugado de Carga

Ampliando-se ainda mais o intervalo (4) da curva do conjugado, observa-se que, o erro em relação ao conjugado de carga e o conjugado estimado, está em torno de 10^{-2} (Figura 6-23).

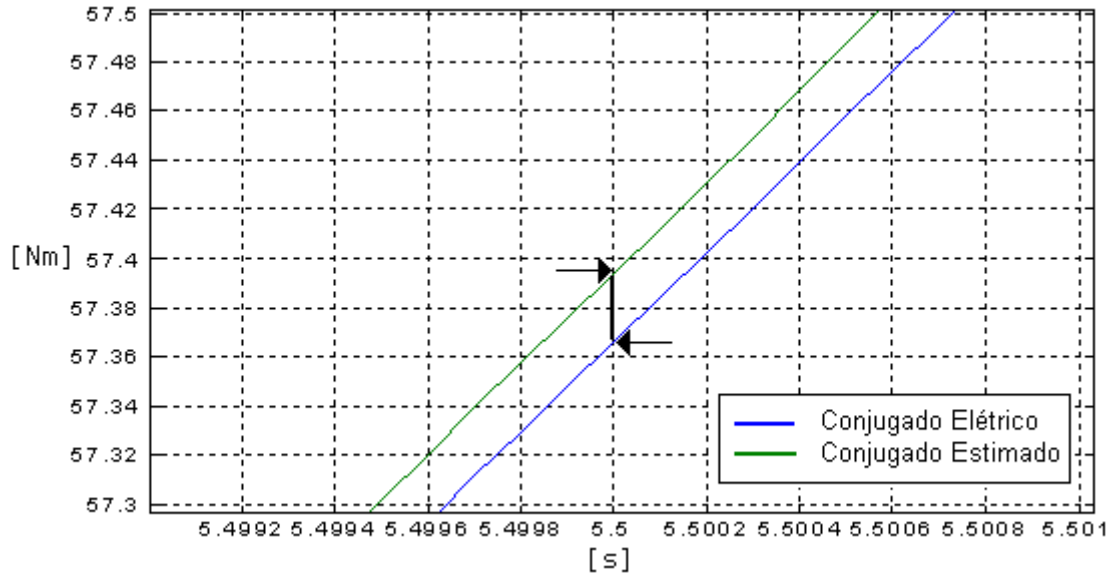

Figura 6-23 – Erro em Uma Transição Gradual do Conjugado de Carga

Para o sistema desequilibrado, existe uma oscilação senoidal do conjugado elétrico e do conjugado estimado, em torno do conjugado de carga, mostrado na Figura 6-24. Para simular o sistema desequilibrado a tensão em uma das três fases foi reduzida em 3% e a tensão em outra das três fases, reduzida em 5%.

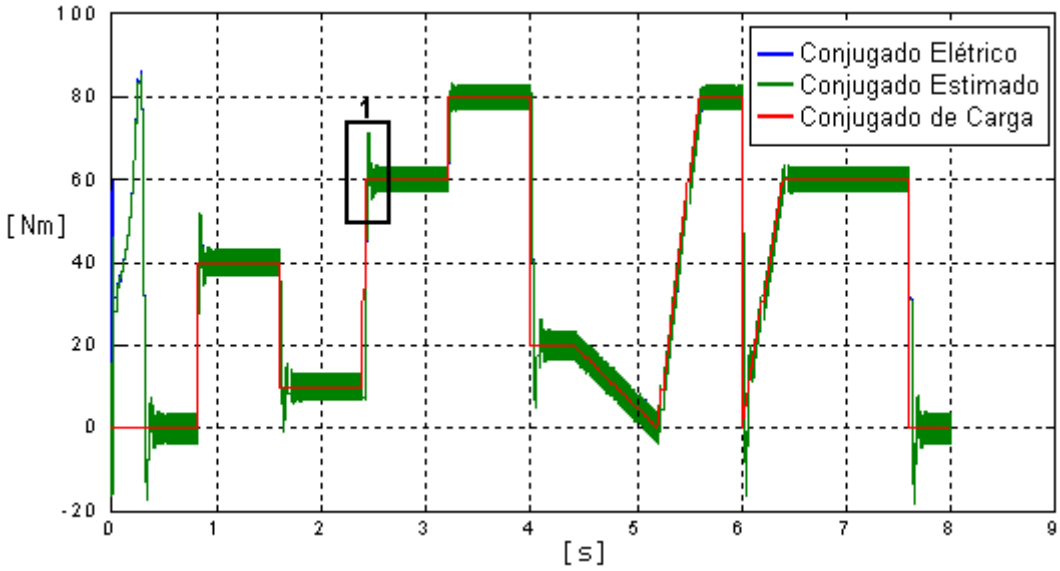

Figura 6-24 – Curvas de Conjugado para o Sistema Desequilibrado

Para melhor visualização da oscilação do conjugado, provocada pelo desequilíbrio do sistema, foi ampliado o intervalo (1) da Figura 6-24. Esta oscilação mostrada na Figura 6-25, atinge 3 [Nm]. Portanto o sistema de eixos ortogonais d,q,0 não é apropriado para o sistema em desequilíbrio, pode-se utilizar o sistema 1,2,0 no caso do sistema desequilibrado [7].

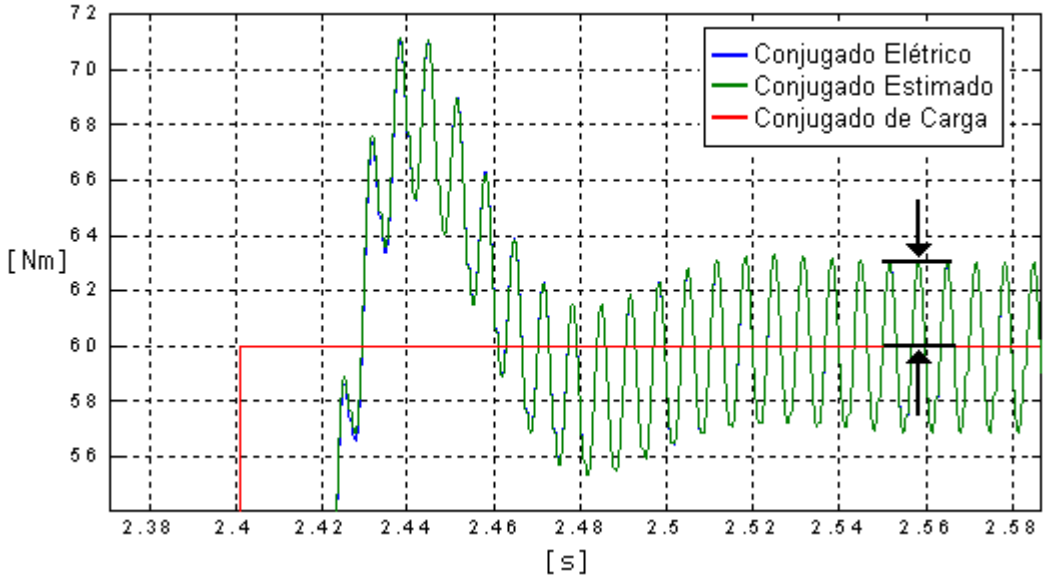

Figura 6-25 – Oscilação do Conjugado para o Sistema Desequilibrado

A curva do conjugado elétrico é coincidente com a curva do conjugado estimado, pois o modelo do MIT utilizado é um modelo no sistema d,q,0, apresentando também a oscilação no conjugado gerado pelo MIT (conjugado elétrico).

No sistema desequilibrado a amplitude do fluxo não fica constante, variando no tempo (Figura 6-26). Esta variação na amplitude do fluxo provoca a variação no conjugado estimado.

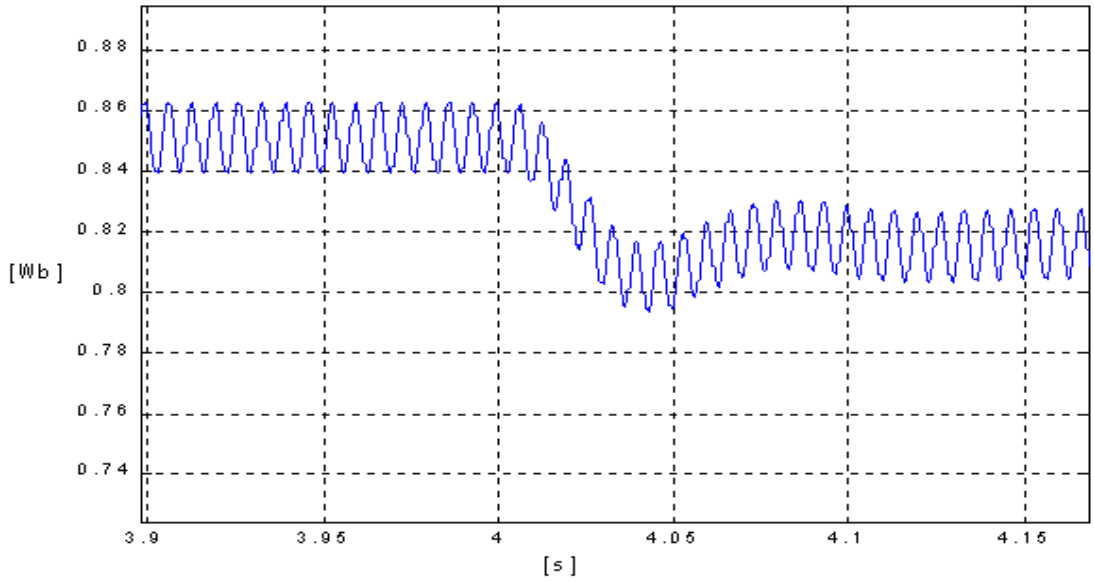

Figura 6-26 – Variação do Fluxo no Tempo Devido ao Desequilíbrio do Sistema

A Figura 6-26 mostra o instante próximo a 4 segundos, onde há variação do conjugado de carga e consequentemente, variação do fluxo.

7- MONTAGEM DO PROTÓTIPO DO ESTIMADOR

Para a implementação do estimador do conjugado é utilizado o DSP56002 da Motorola, operando em 40 MHz. A amostragem e digitalização dos sinais trifásicos da corrente e da tensão, são executadas por uma placa com o conversor A/D (Analógico para Digital) ADS7864 da Texas Instruments, este conversor possui seis canais analógicos de entrada que são amostrados simultaneamente.

7.1 Aquisição de Dados

O EVM 56002 (*kit* da Motorola) já possui na placa o CODEC (conversor A/D e D/A) CX4215XL1, que é um CODEC de áudio e portanto dispõe de apenas dois canais, esquerdo e direito, que utiliza a técnica de conversão tipo sigma-delta.

Para a implementação do estimador é necessário a amostragem de seis sinais, as três tensões e as três correntes de linha, portanto uma solução seria multiplexar os sinais antes de serem convertidos, três sinais no canal esquerdo e três sinais no canal direito. Essa multiplexagem poderia ser feita por *hardware*. Contudo com a técnica de conversão sigma-delta uma amostra do sinal analógico vai interferir na amostra seguinte, desta forma não é possível multiplexar sinais na entrada do A/D, porque um sinal vai interferir no outro. Por este motivo o conversor A/D já presente na placa do EVM 56002 não foi utilizado, sendo necessário um conversor A/D externo à placa do EVM 56002. Somente o conversor D/A do CODEC foi utilizado, para mostrar o resultado do conjugado estimado.

Para uma correta estimação do conjugado é necessário que os sinais de corrente e tensão sejam amostrados exatamente no mesmo instante, sem

defasagem entre as amostras dos sinais. Havendo defasagem na amostragem dos sinais, esta poderia ser compensada via programa, sendo necessário a confecção de uma rotina no programa, para compensar esta defasagem. Esta alternativa foi descartada e foi escolhido um ADC (*Analog to Digital Converter*) que faz a amostragem e retenção dos sinais ao mesmo tempo. O conversor A/D utilizado foi o ADS7864, da Texas Instruments.

Quando o HOLD do conversor é habilitado, as seis amostras ficam retidas até que o HOLD seja habilitado novamente (Figura 7-1).

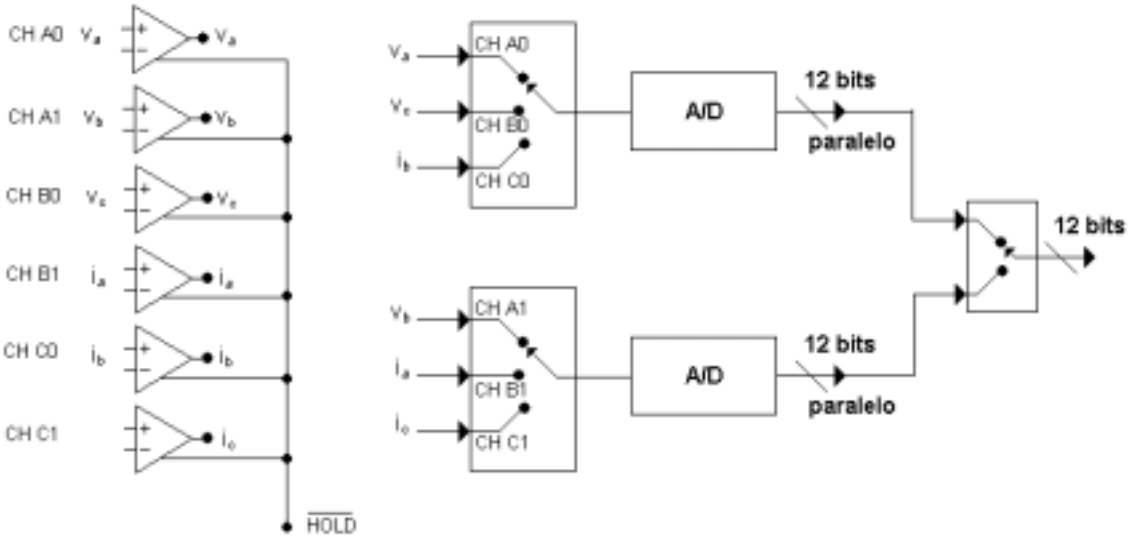

Figura 7-1 - Amostragem e Retenção dos Seis Sinais Simultaneamente.

7.1.1. O ADS7864

O ADS7864 é um conversor analógico para digital (A/D) dual de 12 *bits*, 500kHz, com seis canais de entrada diferenciais agrupados em três pares para aquisição simultânea de sinais, em alta velocidade. Como são dois conversores A/D, cada um recebe três entradas que são multiplexadas (A0/A1, B0/B1 e C0/C1) portanto duas entradas podem ser amostradas e logo após, convertidas simultaneamente. É importante ressaltar que as seis amostras são retidas no

mesmo instante e só após a retenção, as amostras são digitalizadas duas a duas (Figura 7-1), preservando assim a informação de fase relativa aos sinais das entradas analógicas.

7.1.1.1. Entradas de Controle do ADS7864

Este componente dispõe de uma interface paralela e algumas entradas de controle, para minimizar sobrecarga de programa. Estas entradas de controle são descritas a seguir:

RD e CS (*read e chip-select)* – estas entradas baixo-ativas de controle, habilitam a saída paralela do A/D, saída que quando desabilitada fica em alta-impedância (*three-state).* Existem algumas formas diferentes para se utilizar a entrada **RD** *(read)*, formas que serão tratadas mais à frente.

HOLDA, HOLDB e HOLDC – habilitadas por nível lógico baixo, quando acionadas retêm amostras dos canais A, B e C respectivamente. Lembrando que cada canal possui 2 entradas que são amostradas simultaneamente.

Estes pinos devem ser interligados para permitir amostragem simultânea dos 6 canais. Apenas um pino de saída do DSP foi reservado para acionar as três entradas ao mesmo tempo.

Quando as três entradas *hold* são acionadas simultaneamente o canal A é convertido primeiro, logo após o canal B e depois o canal C. Se a entrada *hold* de um canal em particular for acionada antes do término da conversão do referido canal, este impulso para a habilitação é ignorado.

BYTE – Apesar do conversor A/D ser de 12 *bits*, este fornece em sua saída paralela uma informação com 16 *bits*, sendo 12 *bits* de dados e 4 *bits* de endereço que indicam a qual canal pertence os 12 *bits* de dados. Quando o pino **BYTE** é acionado o A/D disponibiliza em sua saída a palavra de 16 *bits* em duas metades de 8 *bits*, primeiro a metade menos significativa e depois a

mais significativa. O pino **BYTE** é habilitado neste trabalho, para economia de pinos de saída do DSP.

A0, A1 e A2 – O ADS7864 possui 3 modos de leitura diferentes que podem ser selecionados pelos pinos A2, A1 e A0. No primeiro modo de leitura pode ser escolhido um dos 6 canais a ser lido dos registros internos ao A/D. Nos outros 2 modos, modo cíclico e modo FIFO, todos os canais convertidos são lidos. No modo FIFO em cada leitura efetuada o dado lido é apagado do registro do A/D, enquanto que no modo cíclico os dados dos 6 canais convertidos ficam armazenados nos registros, até que nova conversão seja efetuada, podendo ser lidos quantas vezes for necessário. Foi utilizado o modo cíclico, onde o dado do canal A0 é lido primeiro, depois o A1, seguidos por Bo, B1, Co e finalmente C1. A Tabela 7-1 mostra os 3 modos de leitura de dados na saída do conversor A/D.

A2	A1	A0	**Canal Selecionado/ Modo de Operação**
0	0	0	A0
0	0	1	A1
0	1	0	B0
0	1	1	B1
1	0	0	C0
1	0	1	C1
1	1	0	Modo Cíclico
1	1	1	Modo FIFO

Tabela 7-1 Modos de Leitura de Dados

São necessários 12 pulsos no pino $\overline{RD}$ para a leitura de todos os canais digitalizados (Figura 7-2), porque cada amostra é lida em duas etapas, parte baixa e parte alta da informação de 16 bits.

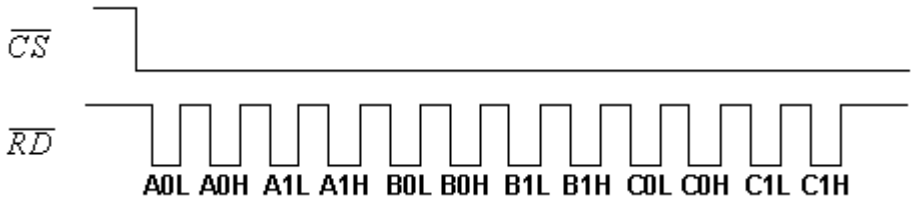

Figura 7-2 - Leitura no Modo Cíclico – 12 etapas de leitura

7.1.1.2. Sinal Analógico de Entrada

O ADS7864 aceita um sinal de entrada analógico variando entre –VREF à +VREF, com um nível de referência interna de +2.5V. Os sinais das tensões e correntes de linha são sinais bipolares, sendo necessário o uso de um circuito de deslocamento de nível conforme Figura 7-3. O valor de R_1 é sugerido pela folha de dados do ADS7864 de acordo com excursão do sinal bipolar de entrada e foi escolhido R_1 = 3,9 KΩ e P_1 = 100 kΩ. O valor R_1+P_1 ajusta o ganho do circuito.

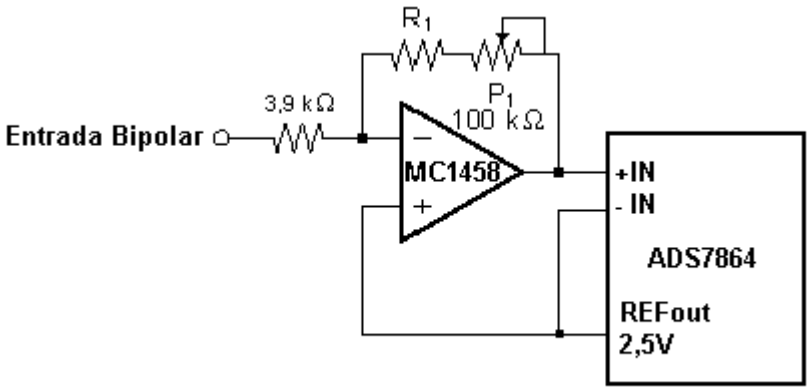

Figura 7-3 Utilização do Circuito de Deslocamento de Nível

7.1.1.3. As Etapas da Conversão e Leitura de Dados

Cada conversão é iniciada quando os três pinos de $\overline{HOLD}$ ficam em nível lógico baixo por no mínimo 20ns (Figura 7-4). A figura mostra apenas um sinal $\overline{HOLD}$ porque os pinos dos três pares de canais estão interligados. A partir daí, o processo de conversão é iniciado nos três pares de canais consecutivamente. A saída $\overline{BUSY}$ assume nível lógico zero para indicar que o conversor está ocupado e retorna após cada conversão. Como são três conversões consecutivas (A0 e A1, B0 e B1 e C0 e C1), há três pulsos de $\overline{BUSY}$. Os dados podem ser acessados pela via de dados após o acionamento dos pinos $\overline{RD}$ e $\overline{CS}$. A Figura 7-4 mostra um ciclo completo de conversão. Os tempos mínimos de cada sinal de controle do A/D podem ser encontrados na folha de dados do ADS7864.

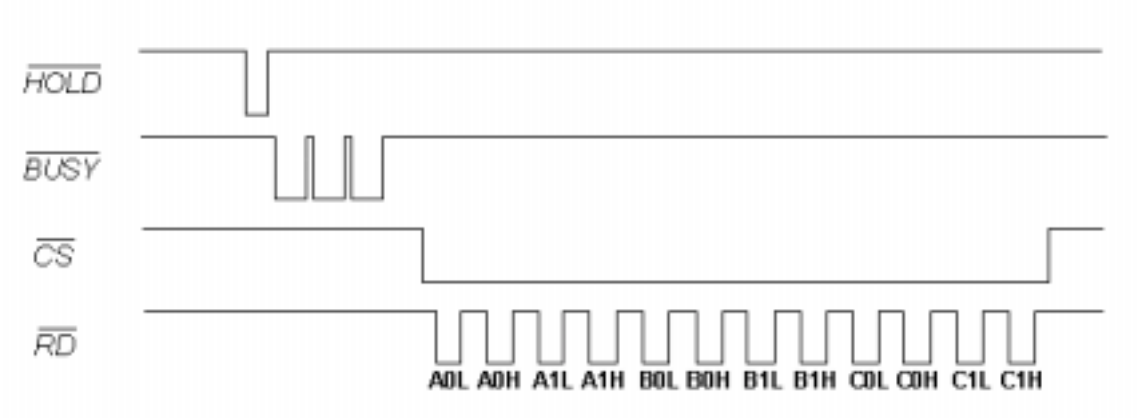

Figura 7-4 - Diagrama de Tempo de um Ciclo de Conversão e Leitura dos Dados

A Figura 7-4 também ilustra o procedimento de leitura dos dados. Após a conversão dos 3 pares de canais, os 6 sinais digitalizados ficam disponíveis em registros internos ao conversor A/D, até que nova conversão seja finalizada. Quando o A/D está trabalhando no modo cíclico de leitura e conversão (Tabela 7-1), os dados armazenados nos registros internos do A/D podem ser lidos quantas vezes for necessário, ficando lá armazenados indefinidamente se outra conversão não for iniciada.

Os pinos da saída paralela de 16 bits estão normalmente em *three-state*. Os pinos $\overline{RD}$ e $\overline{CS}$ têm que ficar em nível lógico baixo por no mínimo 30ns para que esta saída possa ser habilitada e seus dados possam ser válidos. Depois de 12,5 ciclos de *clock* após o início de uma conversão, os novos dados são armazenados (*latched*) dentro seu registro de saída, mas como os 3 pinos $\overline{HOLD}$ são habilitados ao mesmo tempo, é necessário aguardar três vezes este tempo para que os 6 canais sejam convertidos.

O ADS7864 possui 16 bits em seu portal de saída, DB0 a DB15. O pino DB15 confirma a validade dos dados da saída quando seu estado lógico é 1. Os pinos DB14, DB13 e DB12 indicam a qual canal pertence a informação presente na saída digital do A/D, conforme indicado na Tabela 7-2. Os 12 bits de dados da saída são representados pelos *bits* DB11 (valor mais significativo) até DB0 (valor menos significativo). Lembrando que esta informação de 16 *bits* pode ser lida em duas metades de 8 *bits*, ativando-se o pino *BYTE* do A/D, e assim foi feito para economia de pinos na conexão com o DSP.

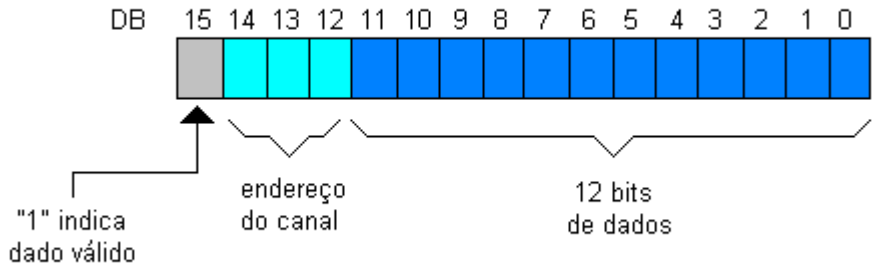

Figura 7-5 - Palavra de 16 *bits* da Saída Digital do Conversor A/D

CANAL DE DADOS	DB14	DB13	DB12
A0	0	0	0
A1	0	0	1
B0	0	1	0
B1	0	1	1
C0	1	0	0
C1	1	0	1

Tabela 7-2 Tabela de Endereços dos Canais de Dados

Quando o ADS7864 trabalha no modo cíclico e com o pino *BYTE* habilitado, a primeira informação a ser disponibilizada na saída digital do conversor A/D é a parte baixa em 8 *bit*s do canal A0, depois a parte alta do canal A0, em seguida a parte baixa de A1 e assim sucessivamente (Figura 7-4). Apesar desta informação estar contida na folha de dados do A/D, esta seqüência de leitura nem sempre é conseguida, gerando inconsistência no resultado da estimação do conjugado. Para contornar este problema foi necessário implementar uma rotina de programa que faz o teste do endereço do canal e também verifica se os 8 *bits* são da parte alta ou da parte baixa da informação. Com esta rotina o programa ficou robusto.

7.1.2. Placa para o ADS7864

O ADS7864 é um *chip* SMD e para sua utilização foi confeccionada uma placa de circuito impresso para sua fixação e operação. A placa foi projetada para conexão com a placa do DSP, disponibilizando conectores para tal fim.

7.1.2.1. Fonte de Alimentação

O ADS7864 necessita de fontes de alimentação separadas para a parte analógica e a parte digital do *chip*, devido ao ruído gerado pela parte digital. Foi construída uma fonte para a parte analógica. Como os sinais de entrada a serem digitalizados são sinais bipolares, uma fonte simétrica foi construída

para alimentar aos amplificadores operacionais e a parte analógica do *chip*. A alimentação da parte digital do *chip* foi retirada da própria placa do DSP.

A placa do ADS7864 foi projetada com a opção de se usar a mesma fonte de alimentação para as alimentações digital e analógica, através de um *jumper*, mas não foi alcançado sucesso porque o funcionamento do A/D fica inconsistente, sendo realmente necessário fontes separadas.

A fotografia da placa montada pode ser vista na Figura 7-6.

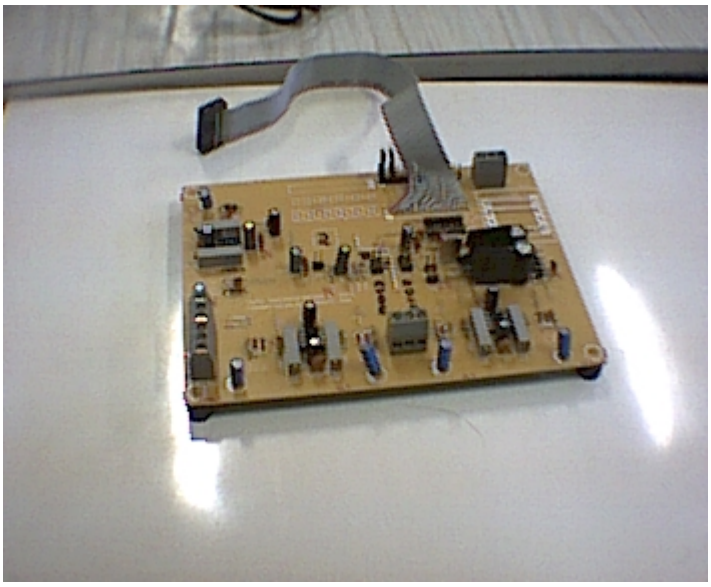

Figura 7-6 - Fotografia da Placa Montada para o Conversor A/D ADS7864

7.1.2.2. Circuito Oscilador para o Conversor ADS7864

O *chip* suporta uma freqüência de *clock* de até 8 MHz. Foi colocado um cristal de 3,579545 MHz que é uma freqüência de cristal facilmente encontrada no mercado. O circuito escolhido foi um circuito baseado em portas lógicas e malha RC, a Figura 7-7 mostra o circuito oscilador utilizado.

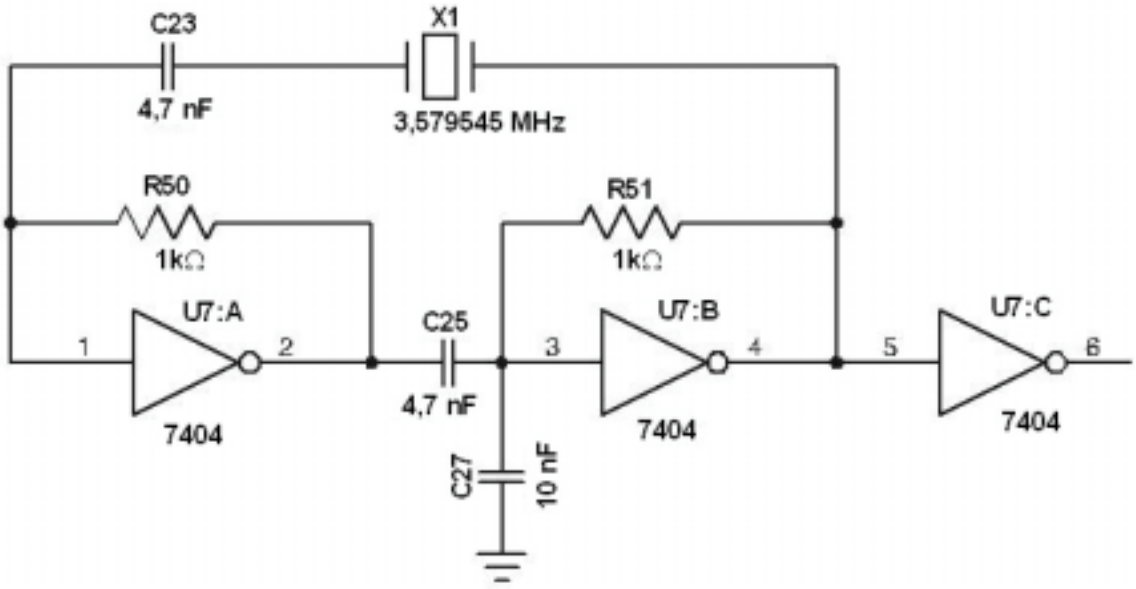

Figura 7-7 - Circuito Oscilador para o ADS7864

7.1.2.3. Configuração da Placa do ADS7864

A configuração é feita por *jumpers*, conforme Tabela 7-3.

JUMPER	FUNÇÃO
JP1	Conecta o RFIN com RFOUT do ADS7864, a tensão de referência do A/D pode ser alterada através deste *jumper.*
JP2	São três *jumpers* que determinam o modo de leitura do A/D, conforme Tabela 7-4.
JP3	Três *jumpers* para selecionar qual(is) pino(os) $\overline{HOLD}$ serão conectados ao DSP.
JP4	Conecta o Vcc digital com o Vcc analógico.
JP5	Conecta o terra digital (DGND) com o terra analógico (AGND).

Tabela 7-3 *Jumpers* de Configuração da Placa

Algumas observações sobre os *jumpers*:

JP1 – O ADS7864 fornece uma tensão de referência interna de 2,5V, conforme citado no item 7.1.1.2, mas se caso necessário pode ser colocada uma tensão de referência externa através deste *jumper*.

JP2 – Configurado conforme Tabela 7-4. Neste trabalho está sendo utilizado o modo de leitura cíclico.

A2	A1	A0	**Canal Selecionado / Modo de Operação**	$JP2_{A2}$	$JP2_{A1}$	$JP2_{A0}$
0	0	0	A0	Fechado	Fechado	Fechado
0	0	1	A1	Fechado	Fechado	Aberto
0	1	0	B0	Fechado	Aberto	Fechado
0	1	1	B1	Fechado	Aberto	Aberto
1	0	0	C0	Aberto	Fechado	Fechado
1	0	1	C1	Aberto	Fechado	Aberto
1	1	0	Modo Cíclico	Aberto	Aberto	Fechado
1	1	1	Modo FIFO	Aberto	Aberto	Aberto

Tabela 7-4 *Jumpers* para seleção do Modo de Leitura

JP3 – Apenas um pino do DSP foi reservado para o acionamento do $\overline{HOLD}$, portanto este *jumper* deve ter seus três pinos fechados, para interconectar os pinos $\overline{HOLDA}$, $\overline{HOLDB}$ e $\overline{HOLDC}$. A opção de deixar a possibilidade de acionar os pinos $\overline{HOLD}$ separadamente, foi para facilitar o teste do ADS7864 em caso de algum problema com a digitalização dos 3 pares de canais simultaneamente.

JP4 – *Jumper* para conectar o Vcc analógico ao Vcc digital. Este *jumper* deve ficar aberto pois se fechado o funcionamento do A/D fica

inconsistente. A alimentação (Vcc) digital é retirada da placa do DSP e ligada neste *jumper*.

7.1.2.4. ***Conectores da Placa***

Para acesso á placa do ADS7864 foram disponibilizados 3 conectores descritos pelas tabelas a seguir.

Pino	***ADS7864***	***DSP***	***Função***
1	$\overline{CS}$	PB8	Habilitar o ADS7864 para leitura de dados
2	NC	PB14	Livre
3	$\overline{RD}$	PB9	Ler informação na saída digital
4	$\overline{BUSY}$	PB12	Indicar que o ADS7864 está ocupado
5	$\overline{HOLD}$	PB10	Amostrar o sinal para início da conversão
6	NC	PB11	Livre
7	NC	PB13	Livre
8	DB6	PB6	*Bit* 6 da palavra de dados
9	DB7	PB7	*Bit* 7 da palavra de dados
10	DB4	PB4	*Bit* 4 da palavra de dados
11	DB5	PB5	*Bit* 5 da palavra de dados
12	DB2	PB2	*Bit* 2 da palavra de dados
13	DB3	PB3	*Bit* 3 da palavra de dados
14	DB0	PB0	*Bit* 0 da palavra de dados
15	DB1	PB1	*Bit* 1 da palavra de dados
16	GND	GND	Terra

Tabela 7-5 Pinagem do Conector 1 – Conexão com o DSP

Através do **Conector 1** a placa do conversor A/D é conectada à placa do DSP e todas as informações de controle e dados são trocadas por este conector. Notar que apesar do conector possuir 16 pinos, apenas 15 são

usados para troca de informação e um pino é a referência de terra entre a placa do A/D e a placa do DSP. Isto porque a foi utilizada a Porta B do DSP que possui apenas 15 pinos de entrada/saída.

O acesso à Porta B do DSP é feito através de J7 na placa do DSP, sendo que J7 é um conector de 16 pinos onde um dos pinos é o terra, que foi aproveitado para a ligação com o conversor A/D.

O **Conector 2** permite o acesso aos 16 pinos de saída de dados. Este conector foi colocado por segurança, caso o acesso da informação de 16 bits em duas metades de 8 bits não funcionasse, mas não foi necessário sua utilização.

O **Conector 3** serve para conectar um terceiro dispositivo ao sistema, como um *display* de cristal líquido ou uma placa com *latch* para comunicação paralela com um PC.

A Tabela 7-6 a seguir mostra a relação de pinagem entre os 3 conectores da placa.

CN1	CN2	CN3	ADS7864	DSP
1	NC	NC	$\overline{CS}$	PB8
2	NC	NC	NC	PB14
3	NC	11	$\overline{RD}$	PB9
4	NC	NC	$\overline{BUSY}$	PB12
5	NC	NC	$\overline{HOLD}$	PB10
6	NC	9	NC	PB11
7	NC	NC	NC	PB13
8	7	7	DB6	PB6
9	8	8	DB7	PB7
10	5	5	DB4	PB4
11	6	6	DB5	PB5
12	3	3	DB2	PB2
13	4	4	DB3	PB3
14	1	1	DB0	PB0
15	2	2	DB1	PB1
16	NC	NC	GND	GND
NC	NC	10	GND	NC
NC	NC	12	*	NC
NC	NC	13	VCC	NC
NC	NC	14	GND	NC
NC	NC	15	NC	NC
NC	NC	16	NC	NC

Tabela 7-6 - Relação de Pinagem entre os três Conectores da placa do A/D

- Conector 3 pino 12 pode ser ligado à GND ou VCC na placa do A/D, ou ainda pode ser utilizado como pino de ajuste de contraste se neste conector for ligado um *display* de cristal líquido.
- Os pinos 9 a 16 do Conector 2 correspondem aos pinos de saída de dados do ADS7864 de DB8 a DB15 e não estão representados na Tabela 7-6.

Estes 8 pinos não estão sendo utilizados neste trabalho porque a leitura de dados de 16 *bits* é feita em 2 metades de 8 *bits* (DB0 a DB7).

Conector das Entradas Analógicas

Os sinais analógicos são conectados à placa do ADS7864 pelo conector TP e o programa foi confeccionado para receber os sinais analógicos conforme Tabela 7-7.

Sinal Analógico	Canal do ADS7864	Conector
Va	A0	TP1
Vb	A1	TP6
Vc	B0	TP2
Ia	B1	TP5
Ib	C0	TP3
Ic	C1	TP4

Tabela 7-7 Conexão dos Sinais Analógicos à Placa do ADS7864

Como foi exposto no item 7.1.1.2 são necessários circuitos deslocadores de nível para deslocar o sinal analógico de entrada que é um sinal bipolar, transformando-o em um sinal unipolar compatível com conversor A/D. Foram utilizados 3 CIs MC1458 que possuem dois amplificadores operacionais na mesma pastilha, atendendo desta forma aos 6 sinais analógicos de entrada. O acesso às entradas dos amplificadores operacionais se dá pelo conector TP, descrito na Tabela 7-7.

7.2 Os Sensores de Efeito Hall

Para obter amostras de três sinais de corrente e três sinais de tensão foram montados três conjuntos com seis sensores no total. Os sensores de corrente não interrompem o circuito da máquina porque os sensores de corrente são capazes de perceber a corrente induzida (*efeito Hall*), não sendo necessário desta forma nenhum tipo de resistência *shunt*, sendo eliminados os problemas de perda por *efeito Joule*. Os sensores apresentam excelente precisão, boa linearidade, baixo tempo de resposta e alta imunidade à interferências externas.

7.2.1. Os Sensores de Tensão

Para medida de tensão, uma corrente proporcional à tensão medida, deverá passar por uma resistência externa (R_1), a qual deve ser instalada em série com o circuito primário do transdutor. A tensão nominal deve ser medida com uma corrente no primário do transdutor mais próxima possível de 10 mA, com isto R_1 deve ser calculado para garantir uma corrente no primário do transdutor, próxima a 10 mA. Neste caso R_1 = 47 kΩ e tensão nominal = 220 V_{ac}, então I_P = 4,68 mA. R_M foi escolhido igual a 100 Ω.

Parâmetros do transdutor de tensão:

- tensão nominal no primário ⇒ **10 a 500 V**
- corrente nominal no secundário(I_{SN}) ⇒ **25 mA r.m.s.**
- escala de conversão ⇒ **2,5:1**
- precisão ⇒ **0,8 %**
- linearidade ⇒ **< 0,2 %**
- tempo de resposta ⇒ **40 μs**

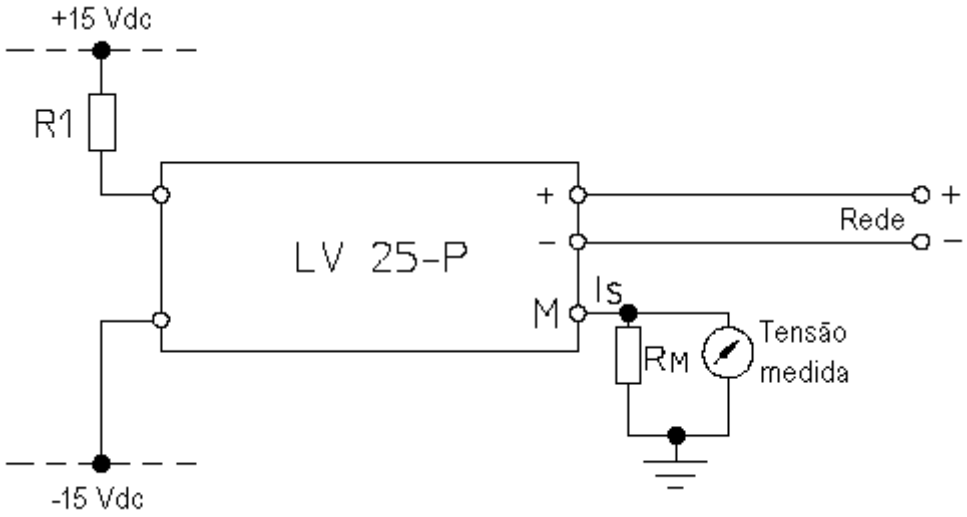

Figura 7-8 - Circuito do Sensor de Tensão

7.2.2. Os Sensores de Corrente

Para que o transdutor de corrente funcione é necessário uma alimentação de +V e –V e uma resistência R_M onde seu valor é sugerido pelo fabricante. Neste caso foi escolhido R_M igual a 180 Ω.

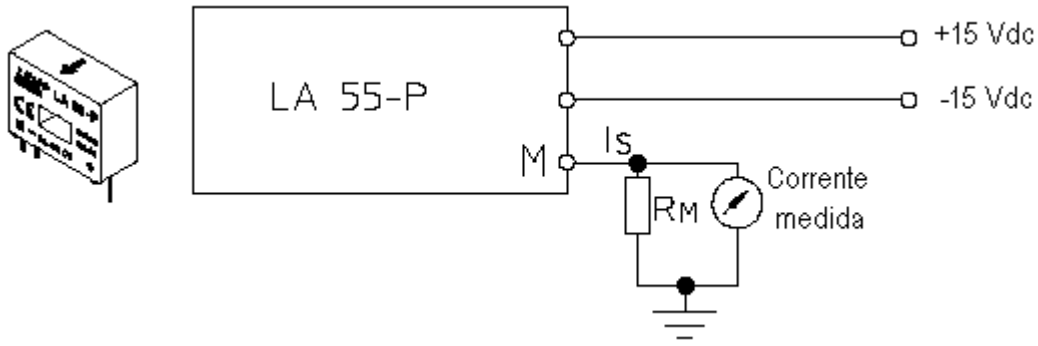

Figura 7-9 - Circuito do Sensor de Corrente

Parâmetros do transdutor de corrente:

- corrente nominal no primário ⇒ **50 A**
- faixa de medida da corrente no primário ⇒ **0 a 70 A**
- corrente nominal no secundário ⇒ **50 mA r.m.s.**
- escala de conversão ⇒ **1:1000**
- tensão de isolação ⇒ **2,5 kV_{ac} r.m.s.**
- precisão ⇒ **0,8 %**
- linearidade ⇒ **< 0,15 %**
- tempo de resposta ⇒ **< 1 μs**

7.2.3. A Placa para os Sensores de Tensão e Corrente

Foi montada uma placa para os sensores Hall, incluindo a fonte de alimentação para os sensores. Esta placa foi acondicionada em uma caixa onde existem dois pares de bornes para a entrada dos sinais de tensão e corrente, caixas pretas na Figura 7-10. Os sinais de saída dos sensores são entregues em 3 fios conforme Tabela 7-8.

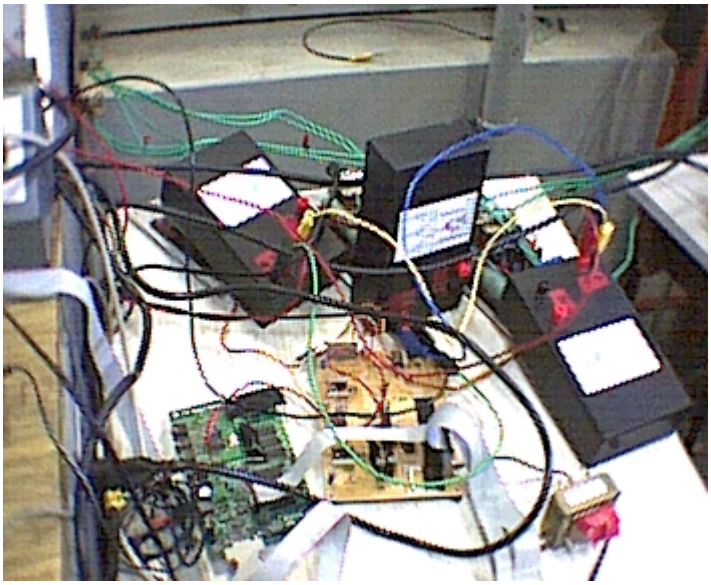

Figura 7-10 - Módulo onde estão os Sensores Hall (caixas pretas)

Comum	Laranja
Tensão	Vermelho
Corrente	Amarelo

Tabela 7-8 Sinais nas Saídas dos Sensores Hall

Em cada placa foi instalado um sensor de corrente e um sensor de tensão, onde a ligação dos sensores à placa do conversor A/D ADS7864 é feita através do conector TP na placa do ADS7864, conforme Tabela 7-7.

7.3 O DSP

Os dados digitalizados pelo ADS7864 são transferidos para o DSP através da porta B. Esta porta é uma porta bidirecional de 15 bits e é controlada por 3 registros, PBC (Port B Control Register), PBDDR (Port B Data Direction Register) e PBD (Port B Data Register).

O registro PBC determina a função da porta B para entrada/saída de uso geral ou *host interface*, quando é necessária a comunicação entre dois ou mais DSPs.

O registro PBDDR determina se os *bits* da porta B são *bits* de entrada ou saída, sendo que os *bits* podem ser programados individualmente e a qualquer momento durante a execução do programa, podendo o mesmo *bit* funcionar como entrada e saída, em instantes diferentes.

O registro PBD é o registro que reflete o valor binário dos pinos da porta B do DSP. Os pinos da porta B são acessados simplesmente escrevendo no registro PBD quando a porta funciona como saída, ou lendo do registro quando este funciona como entrada. Os *bits* da porta B podem ser lidos/modificados individualmente. Em [12] podem se vistos detalhes dos 3 registros.

O acesso à porta B é feito através de J7 na placa do EVM (evaluation module), mostrado na Figura 12-5.

A comunicação do DSP com o conversor D/A (DSP e conversor D/A estão na mesma placa) é feita pela porta A que é o canal serial síncrono do DSP (Figura 7-11). Toda a comunicação serial do DSP com o conversor D/A é executada pela rotina contida no arquivo txrx_isr.asm, anexo 12.2.6. Mais detalhes sobre a porta A podem ser encontrados em [12].

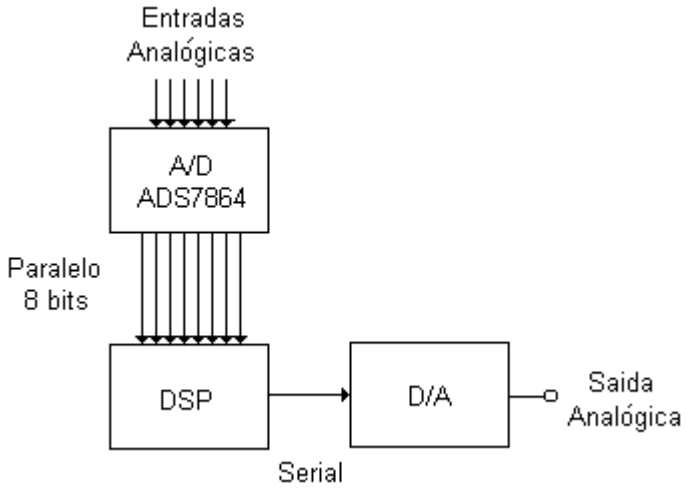

Figura 7-11 Comunicação A/D DSP e DSP D/A

As instruções aritméticas na família DSP56K são totalmente voltadas para a representação numérica fracionária e, por essa razão, os dados devem ser escalonados para esta representação. Os dados são representados neste trabalho no formato Q_{23} ou Q_{16}. O formato Q está descrito na seção 5.1.7. A informação digitalizada entregue ao DSP está na notação complemento 2 formato Q_{23}, para se conseguir esta representação os potenciômetros da placa do ADS7864 foram ajustados para o resultado da conversão ficar entre –1 e +1, em notação complemento 2. Embora os dados de informação de entrada para o DSP estarem no formato Q_{23}, em alguns pontos do programa é necessário que esses dados sejam operados com números inteiros sendo preciso fazer a conversão do formato Q_{23} para Q_{16} e vice-versa. Neste sistema

que opera com 24 *bits* o formato Q_{16} nos permite representar números inteiros com 8 *bits* (7 de magnitude e um de sinal) mais os 16 *bits* fracionários, sendo possível a representação de inteiros na faixa –128 a +127.

Na Figura 7-12 é apresentado o fluxograma do programa do sistema, que foi desenvolvido em linguagem *assembly*. O funcionamento do programa (fluxograma) pode ser resumido a seguir:

- A inicialização do DSP e do CODEC se faz necessária para "carregar" alguns registros com o objetivo de definir níveis de prioridade de interrupção, definir o ponteiro de pilha, carregar o valor de contagem para o *timer* entre outras necessidades. O CODEC também trabalha com registros para definir freqüência de amostragem, habilitar ou desabilitar pré-amplificadores. Apesar do CODEC estar na placa do DSP, neste trabalho é utilizada apenas a função D/A do CODEC, não é utilizada a função A/D porque a técnica de conversão utilizada é a sigma-delta, sendo imprópria para a multiplexagem de sinais. A multiplexagem seria necessária porque o CODEC possui apenas dois canais, e seis sinais precisam ser convertidos.
- Após a inicialização do sistema, o DSP fica aguardando a interrupção gerada pelo *timer*. Esta interrupção é gerada a cada 125 μs, que equivale a uma freqüência de amostragem de 8 kHz.
- A cada 125 μs o DSP aciona o ADS7864 para coletar as 6 amostras dos sinais e converte-las para digital e após a conversão executa o restante do fluxograma para estimação do fluxo e cálculo do conjugado, retornando para a espera de uma nova interrupção .
- Após ser disparada a conversão A/D dos 6 sinais de entrada, o DSP aguarda um tempo que seja suficiente para que a conversão seja finalizada com segurança. Este tempo é de 20 μs.
- A leitura dos dados digitalizados é concluída após 12 repetições, pois cada uma das 6 informações digitalizadas de 12 bits é lida do ADS7864 em duas metades, pois foi disponibilizado pelo DSP através da porta B apenas 8 bits de dados, não sendo possível a leitura dos

12 *bits* paralelos simultâneos. Isto para economia de pinos da porta B que também é utilizada para outros fins.

- Após o DSP ter disponível as 6 informações de entrada, Va, Vb, Vc, Ia, Ib e Ic, os sinais são convertidos de 3 fases para o sistema d,q,0 pela matriz descrita na seção 3.2. Esta conversão facilita a manipulação dos sinais, eliminando por exemplo o ângulo entre as fases, facilitando as operações matemáticas.
- Os sinais já no sistema d,q,0 são aplicados à um filtro passa-baixas baseado em redes neurais artificiais (ANN-PCLPF). Este filtro nos permite a integração dos sinais, fornecendo após a sua rotina de programa, os vetores de fluxo de campo estacionário Ψ_{ds}^{s} e Ψ_{qs}^{s}.
- Com os vetores de fluxo de campo estacionário é possível obter o fluxo do estator $\hat{\Psi}_s$, o ângulo θ_e, as correntes de campo girante i_{ds} e i_{qs} e o fluxo de campo girante do estator Ψ_{ds} e Ψ_{qs}.
- Com os dados obtidos no passo anterior, o conjugado é calculado pela equação $T_e = \frac{3P}{4}\left[\Psi_{ds} i_{qs} - \Psi_{qs} i_{ds}\right]$.
- Após o cálculo do conjugado o resultado é enviado para um *latch* onde fica disponível para a leitura e plotagem no PC, por uma rotina em *DELPHI*. O resultado é também enviado para o conversor D/A da placa do EVM (evaluation module), podendo ser mostrado na tela de um osciloscópio.

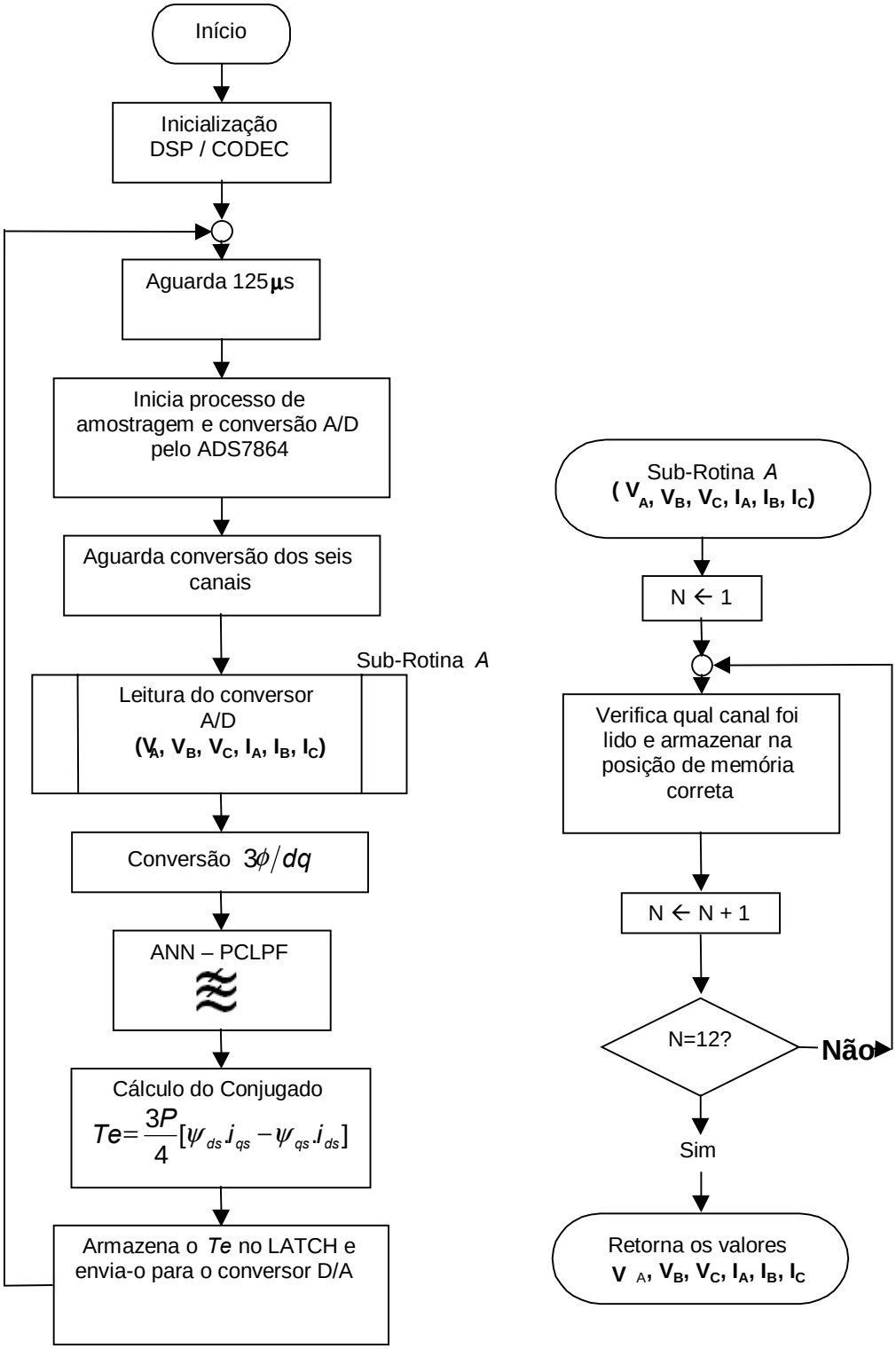

Figura 7-12 Fluxograma do Programa Desenvolvido em Linguagem *Assembly*

7.4 Sistema Completo

Nesta seção é descrito o sistema, incluindo a ligação entre os componentes do sistema. A Figura 7-13 ilustra como os diversos componentes do sistema se relacionam.

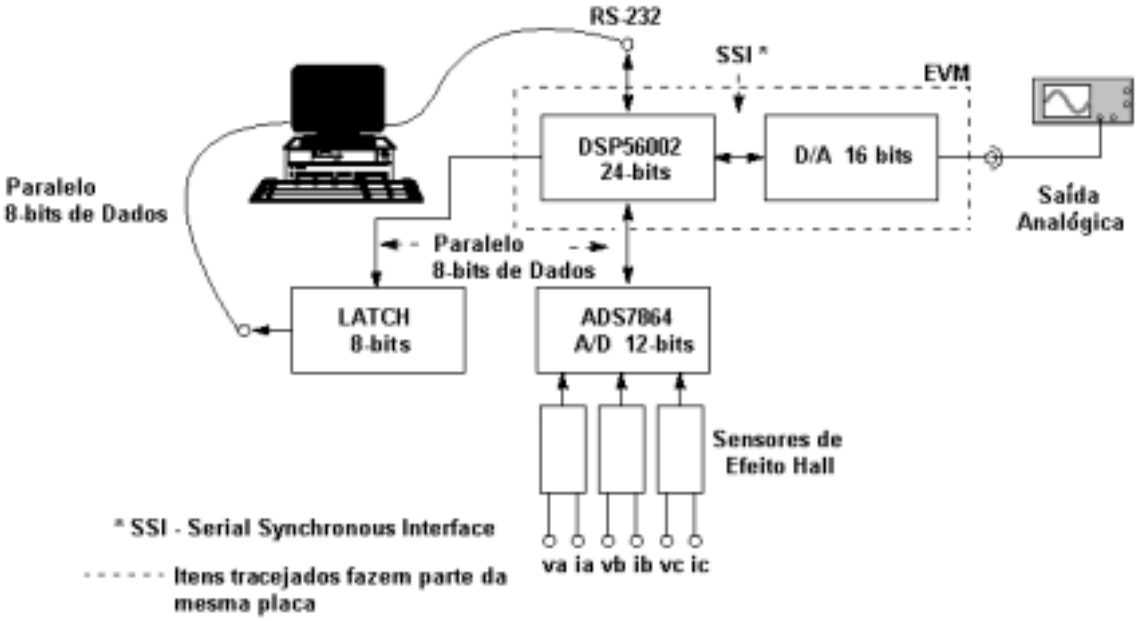

Figura 7-13 Diagrama Completo do Sistema

A placa do DSP (EVM) é conectada ao PC por uma porta serial a uma velocidade de comunicação de 19200 bps. O programa desenvolvido para o estimador necessita ser carregado para o DSP via porta serial, toda vez que o sistema é ligado, pois o programa é carregado em memória RAM. A comunicação entre o PC e o DSP pela porta serial ocorre somente no momento em que o programa está sendo carregado, após o carregamento o DSP executa o programa independentemente do PC, exceto quando a opção de *debugger* está habilitada, onde neste caso o DSP atualiza os registros e posições de memória na tela (GUI). A placa do EVM permite a colocação de uma memória E^2PROM para funcionamento *stand-alone*. O carregamento do programa para o DSP é feito pela GUI (*graphic user interface*) que é parte integrante do *kit* do EVM. O programa da GUI é chamado “EVM56K.EXE”.

Todo o controle e gerenciamento do sistema é feito pelo DSP que comunica com 3 outros dispositivos além do PC, o conversor A/D ADS7864 apresentado na seção 7.1.1, o conversor D/A (na própria placa do DSP) e a placa do *latch*, que disponibiliza ao PC o resultado do conjugado estimado. O DSP pode comunicar também com um *display* de cristal líquido, pois a placa do ADS7864 foi projetada disponibilizando um conector para a conexão de um *display*, caso necessário.

A conexão do DSP à placa do ADS7864 é feita pelo J7 no lado do DSP e pelo CN1 no lado da placa do ADS7864, conforme Tabela 7-5.

Apesar do ADS7864 ser um conversor A/D de 12 *bits* a comunicação entre o DSP e o A/D é feita por 8 *bits* (Figura 7-13), sendo que a parte alta da informação é multiplexada com a parte baixa. Além dos 12 *bits* de dados existem ainda 4 *bits* de controle. O pino de controle $\overline{HOLD}$, é usado pelo DSP para sinalizar ao A/D para dar início à um novo ciclo de conversão, o $\overline{HOLD}$ é acionado a cada 125μs e o processo de conversão dura cerca de 8 μs. O pino $\overline{BUSY}$ é um pino usado pelo A/D para indicar que o processo de conversão está em andamento, este pino não está sendo testado pelo sistema pois para que o programa ficasse mais robusto uma temporização foi utilizada para aguardar o processo de conversão. O pino $\overline{CS}$ é utilizado para habilitar o A/D. Por último, o pino $\overline{RD}$ é utilizado pelo DSP para fazer a leitura dos dados digitalizados da saída do A/D, lembrando que os pinos da saída de dados do A/D ficam em alta-impedância até que os pinos $\overline{RD}$ e $\overline{CS}$ sejam habilitados simultaneamente.

O conversor A/D é ligado aos sensores de efeito Hall através do conector TP da placa do A/D (Tabela 7-7). É importante observar a ligação dos sensores na placa do A/D, esta ligação deve respeitar a correta relação das tensões e correntes das fases com os respectivos canais do A/D, pois se trocada alguma ligação o resultado do conjugado estimado não estará correto.

Da mesma forma deve-se tomar o cuidado de ligar cada tensão e corrente das fases no sensor de efeito Hall correspondente.

Após o DSP receber os dados digitalizados do ADS7864 e processar a informação obtendo o conjugado estimado, o DSP disponibiliza o resultado em uma saída analógica através do conversor D/A da própria placa do DSP. Este conversor D/A é o circuito integrado CX4215XL1 da *Crystal Semiconductor Corporation*, este circuito integrado é um CODEC, ou seja, tem a função de D/A e de A/D. A função A/D não foi utilizada pelo motivo exposto na seção 7.1. A comunicação entre o DSP e este conversor D/A é feita pelo canal serial síncrono do DSP. O fabricante do DSP já fornece o arquivo em linguagem *assembly*, anexo 12.2.6, para a configuração do canal serial do DSP e do D/A, pois o D/A se encontra na própria placa do DSP. Para enviar uma informação ao D/A é necessário apenas escrevê-la em uma posição de memória, sendo transparente ao usuário toda a configuração do canal serial. Mais informações sobre o canal serial podem ser encontradas em [12].

O conjugado estimado pode ser observado em um osciloscópio colocado na saída analógica do sistema, mostrado na Figura 7-13. O resultado também pode ser verificado na tela de um PC em uma rotina feita em linguagem DELPHI, porque o resultado também é enviado para a placa do *latch* de 8 *bits*, ficando disponível para ser lido pelo PC. O circuito integrado utilizado para a placa do *latch* é o 74573 e é conectado ao DSP via CN3 da placa do ADS7864 (Figura 7-14).

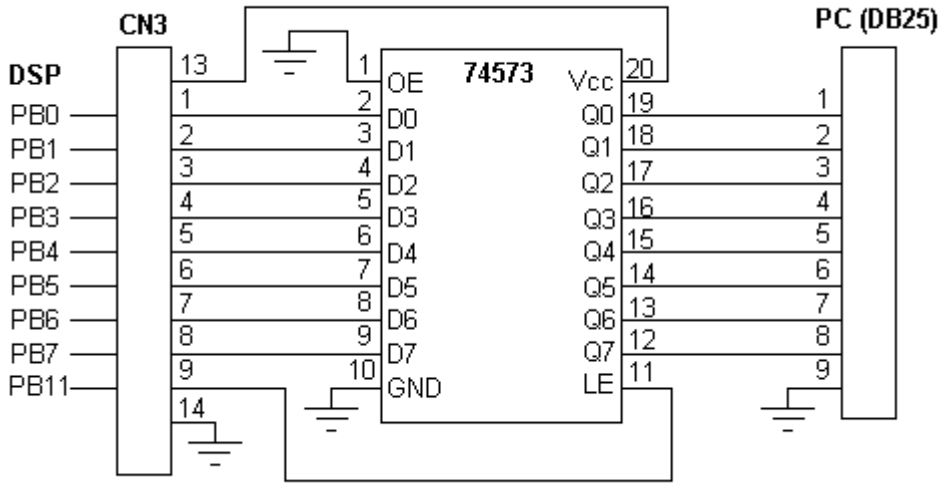

Figura 7-14 Ligação do *latch* ao PC e ao DSP via placa do ADS7864

O DSP utiliza o mesmo barramento de dados para ler os dados digitalizados provenientes do ADS7864 e para escrever o resultado do conjugado estimado no *latch*, portando o ADS7864 deve ser desabilitado (saída digital em alta impedância) antes do envio da informação ao *latch*.

O a informação do conjugado estimado é disponibilizada ao *latch* a cada 125 μs, ficando disponível para a leitura pelo PC. Esta leitura feita pelo PC independe da temporização do DSP, ou seja, não existe nenhum sincronismo entre o PC e o DSP, facilitando a implementação da rotina em linguagem DELPHI.

A aquisição dos dados pelo PC é feita por uma placa paralela de entrada/saída de 8 *bits*, que necessita ser configurada para trabalhar como entrada ou como saída. Esta configuração é realizada via *software* e consiste em carregar alguns registros com valores específicos. A configuração desta placa de aquisição de dados pode ser realizada pela própria rotina em linguagem DELPHI. A placa de entrada/saída também fornece a alimentação de 5V para o CI 74573. Maiores detalhes sobre a configuração da placa de aquisição de dados podem ser encontrados no anexo 12.5, e a rotina em DELPHI citada é apresentada no anexo 12.7.

8- TESTES E RESULTADOS

Para se obter as curvas do conjugado, o MIT foi acoplado à um gerador de corrente contínua mostrado na Figura 8-1. Aplicando-se uma variação de carga ao gerador é obtida a variação do conjugado de carga do MIT.

Figura 8-1 – Máquina de Corrente Contínua Acoplada ao MIT

A variação de carga do gerador é provocada por pás imersas em água (Figura 8-2), onde quanto maior a superfície da pá imersa na água, maior a carga.

Figura 8-2 – Pás Funcionando como Carga para a Máquina de Corrente Contínua.

Dados do motor trifásico utilizado para a comprovação dos resultados:

Marca	EBERLE
Tensão Nominal	220 V
Ligação	Triângulo
Rotação	1765 R.P.M.
Corrente Nominal	26 A
Modelo	B132 S 4 / ESP
Número de Série	9958 D5
FS	1.0
Ip/In	8,6
Potência Nominal	10 CV
Isolação	B
Categoria	H
Res.do Estator	0,477 Ω

Tabela 8-1 – Dados do Motor de Indução Utilizado para a Comprovação dos Resultados

A resistência do estator foi medida com uma ponte de Winstone da marca Siemens e o valor encontrado foi de 0,318 Ω. Como a ligação do motor é triângulo, o valor medido corresponde a uma fase em paralelo com a soma das outras duas. Portanto o valor da resistência do estator para cada fase é $\frac{3}{2}$ do valor medido (0,477 Ω).

O DSP disponibiliza a informação do conjugado estimado através da Porta B de 8 *bits*. Esta informação é armazenada em um *latch* de oito *bits* (CI 74573) conforme Figura 7-14, ficando disponível para a rotina em DELPHI. Não existe nenhum sincronismo entre o DSP e a rotina em DELPHI, ou seja, o DSP escreve a informação na Porta B a cada 125μs independente se a rotina em DELPHI já leu a informação anterior. Isto causa uma certa perda de resolução no gráfico plotado no PC pela rotina em DELPHI, uma vez que a rotina em DELPHI é mais lenta e portanto não aproveita todas as informações

disponibilizadas pelo DSP. Mas levando em consideração que a rotina no PC será usada somente para mostrar o resultado de uma forma mais amigável, isto não é um problema. Circuitos que porventura forem utilizar o conjugado estimado para atuar em algum tipo de controle, necessitarão deste sincronismo.

Foi implementado um circuito adicional (Figura 7-14), para disponibilizar o conjugado estimado digitalizado. Os resultados são apresentados nas figuras desta seção.

A Figura 8-3 a seguir mostra uma situação onde o conjugado de carga cresce abruptamente e depois vai diminuindo.

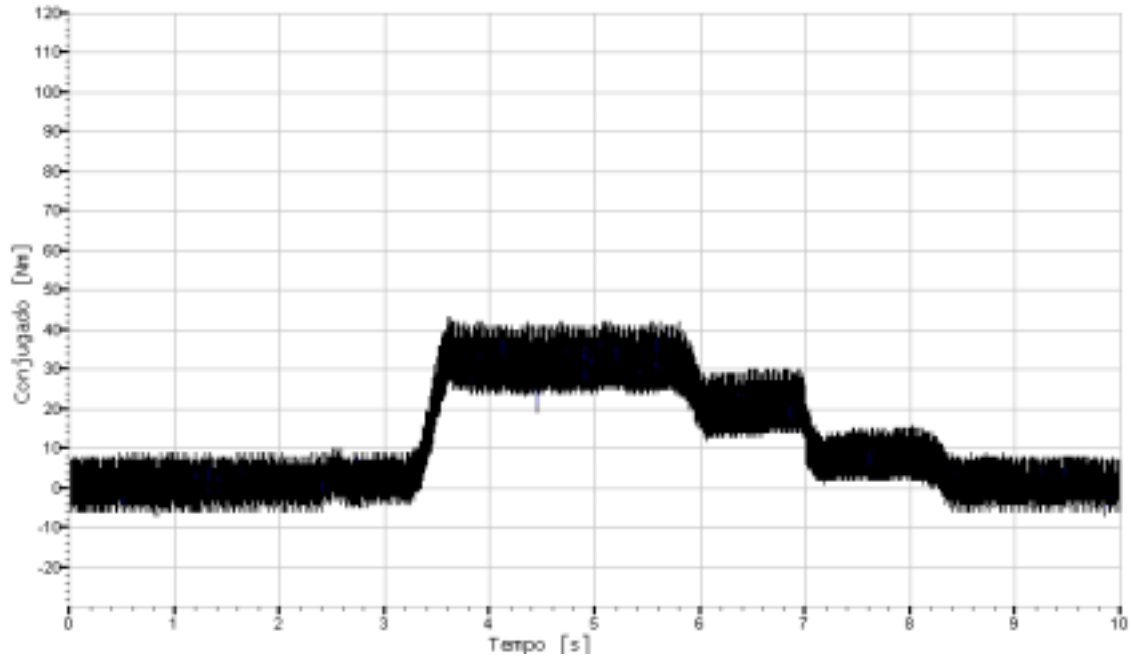

Figura 8-3 – Variação do Conjugado Exigido do MIT em um Intervalo de 10 segundos

Também se pode observar a variação do conjugado pela Figura 8-4 e Figura 8-5. A Figura 8-5 apresenta uma escala menor.

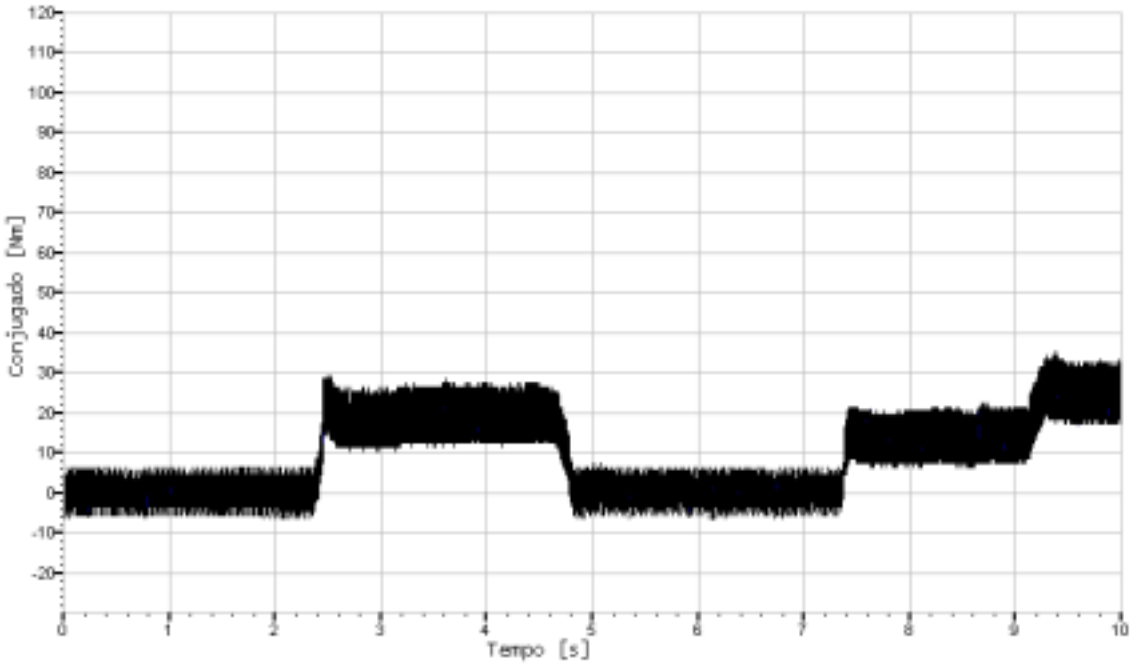

Figura 8-4 – Variação do Conjugado

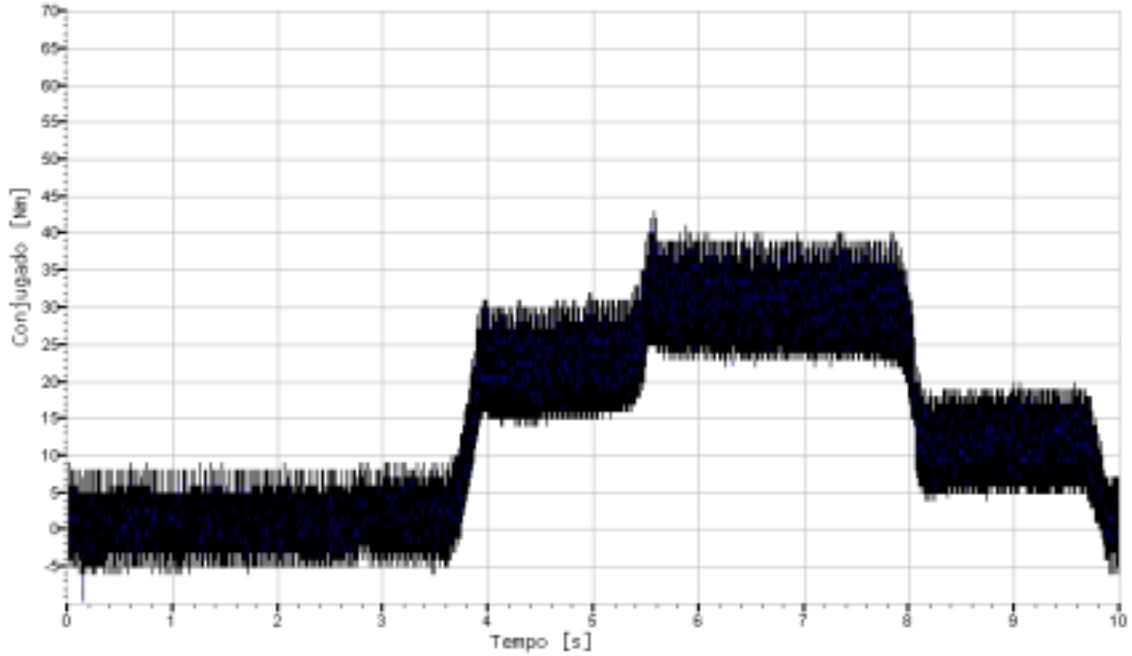

Figura 8-5 – Variação do Conjugado com Fundo de Escala de 70 Nm

A Figura 8-6 mostra a resposta transitória para um aumento repentino na exigência do conjugado. Na Figura 8-7 observa-se a resposta transitória com *zoom*.

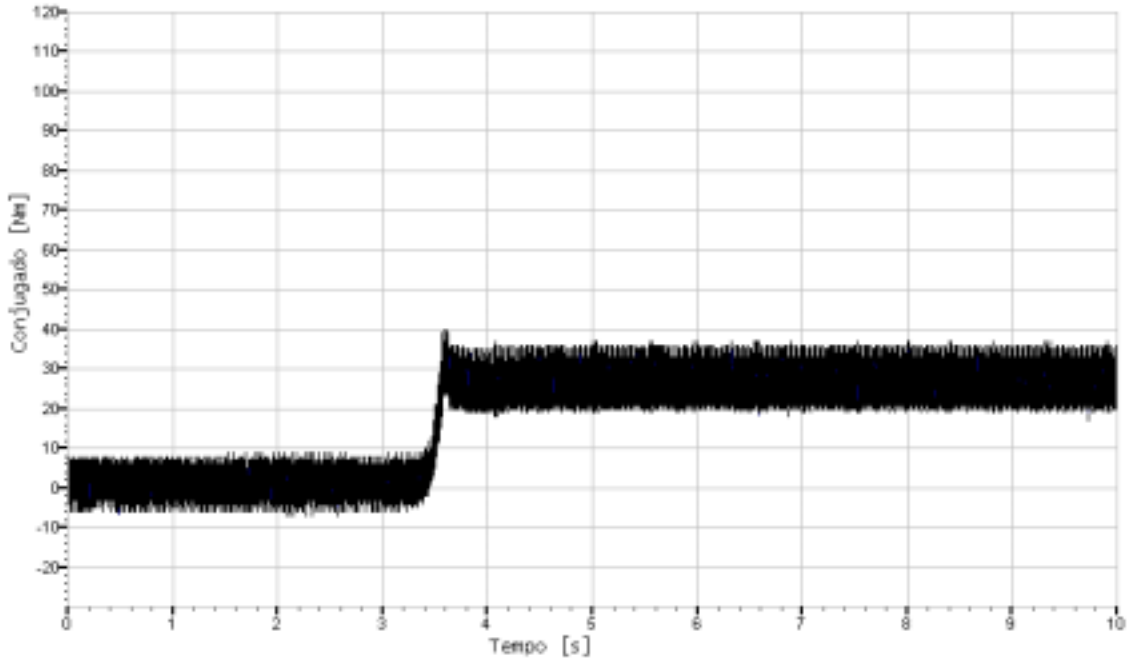

Figura 8-6 – Resposta Transitória para um Aumento do Conjugado de Carga

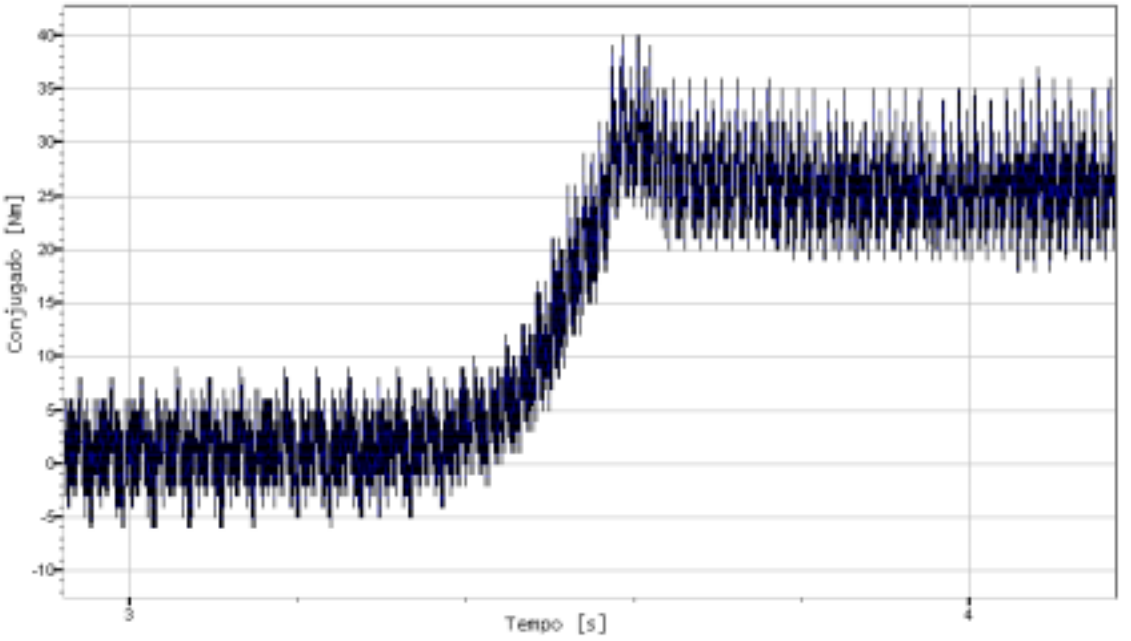

Figura 8-7 – Resposta Transitória com a Escala de Tempo mais Aberta

Devido ao processo de variação da carga da máquina de corrente contínua, não se consegue uma variação muito abrupta do conjugado de carga do MIT. Esta variação é feita manualmente pela inserção ou retirada das pás na água, um processo mecânico lento.

A variação do conjugado para uma diminuição da carga pode ser observada pela Figura 8-8 e Figura 8-9.

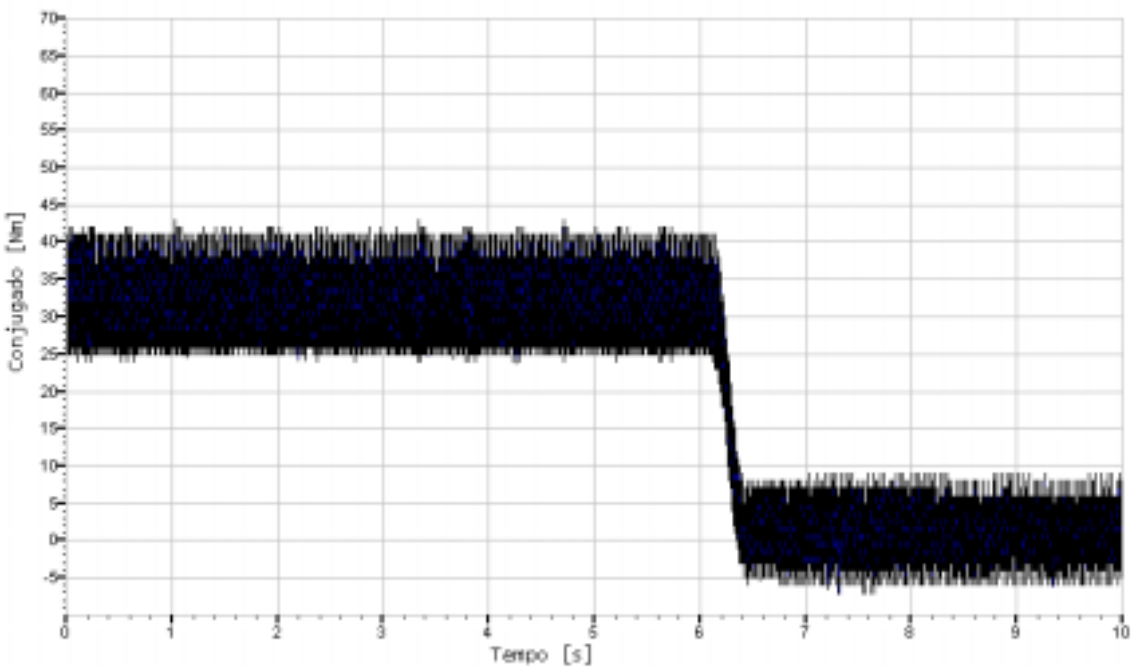

Figura 8-8 – Conjugado Caindo de Aproximadamente 32 Nm para Zero

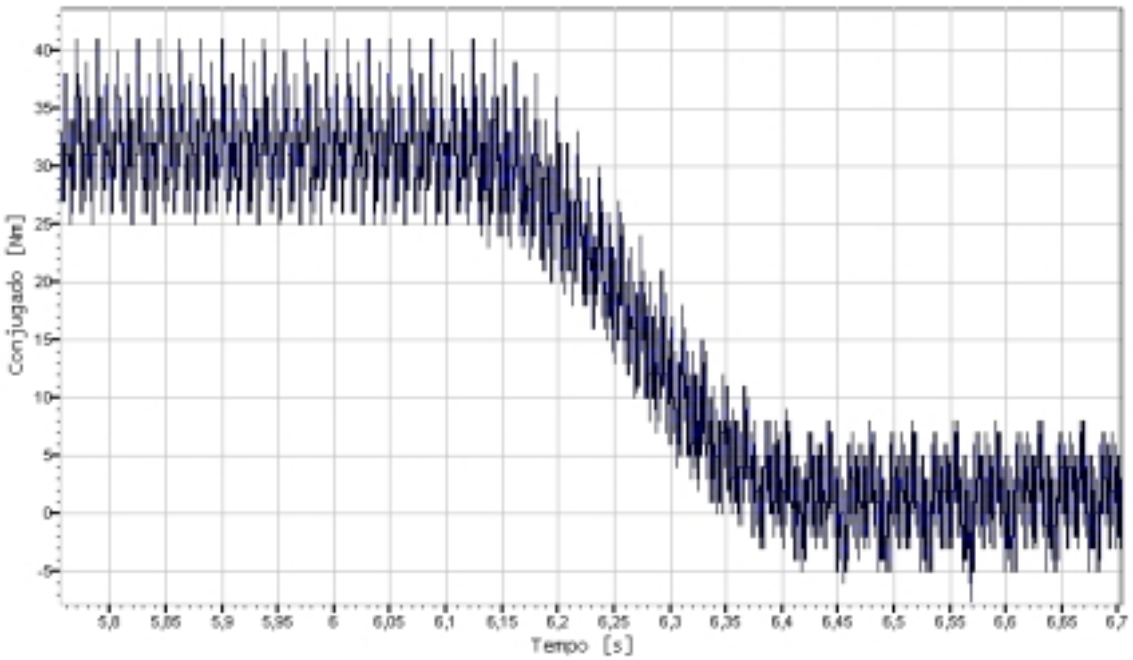

Figura 8-9 – Resposta Transitória com *Zoom*

O conjugado estimado pode também ser observado através de um osciloscópio, mas para melhor visualização o conjugado é mostrado na tela do PC pela rotina na linguagem de programação DELPHI, anexo 12.7.

Quando o resultado é mostrado na tela de um osciloscópio, este resultado precisa antes ser convertido para um sinal analógico. Esta conversão pode ser feita pelo conversor digital para analógico que acompanha o sistema mínimo do DSP. O arquivo de configuração do conversor D/A se encontra na seção de anexos, juntamente com o arquivo que configura a comunicação serial entre o DSP e o D/A.

Para a comprovação dos resultados foi utilizado o equipamento medidor de conjugado da marca Monitek (*Three Phase Power Analiser*), disponível no LEPCH (laboratório de ensaios de pequenas centrais hidrelétricas) da UNIFEI. Este equipamento é baseado em *strain gauge*.

O sensor do equipamento é colocado no eixo da máquina conforme mostrado na Figura 8-10 e além do conjugado fornece velocidade, potência mecânica, potência elétrica, rendimento e etc.

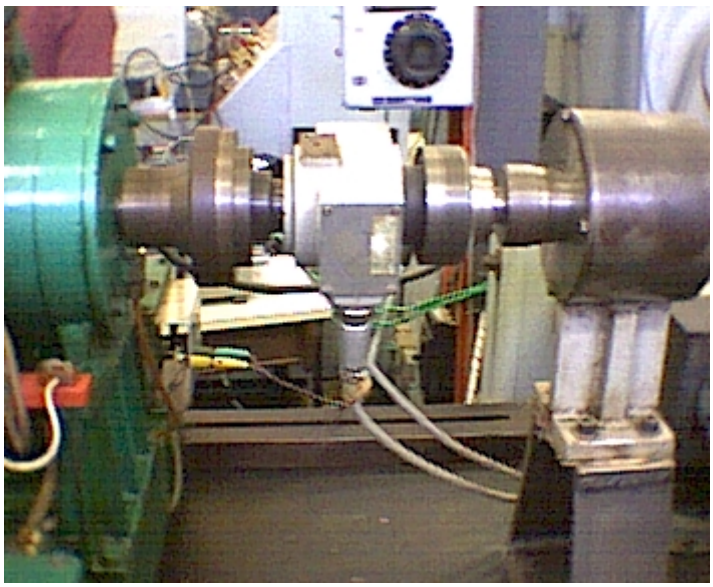

Figura 8-10 – Sensor baseado em *strain gauge* que Envia Sinais para o Monitek, Acoplado ao Eixo das Máquinas

A Tabela 8-2 mostra os valores obtidos pelo estimador de conjugado implementado neste trabalho, e os valores obtidos pelo equipamento Monitek do laboratório.

Monitek [Nm]	Estimador [Nm]	Corrente MIT [A]
11,55	11,75	15,04
19,12	18,75	17,25
23,45	23,8	19,02
28,52	28,6	21,24
32,45	32,5	23,1

Tabela 8-2 Tabela Comparativa Entre os Valores Obtidos pelo Estimador e o Equipamento do Laboratório – Monitek

A Figura 8-11 a seguir mostra as curvas do conjugado estimado, do conjugado medido pelo Monitek e o valor da corrente do MIT. O conjugado a vazio foi descartado porque o conjugado fornecido pelo MONITEK nesta condição não é constante, apresentando uma grande variação na medida.

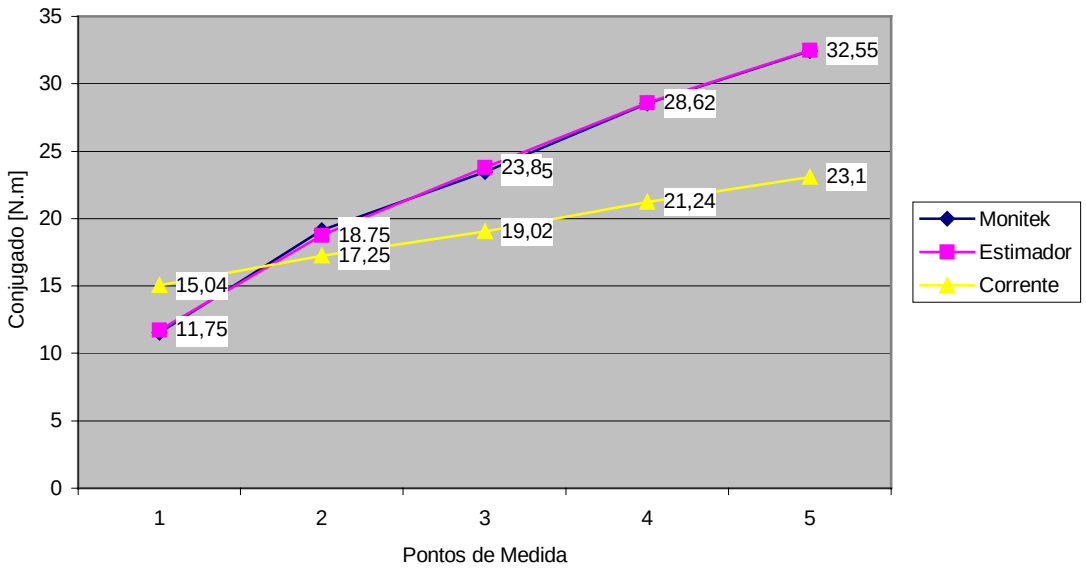

Figura 8-11 Conjugado Estimado, Conjugado Medido pelo Monitek e Corrente do MIT

Observa-se na seção anterior que, as curvas do conjugado estimado obtidas pela rotina em DELPHI, apresentam uma oscilação de conjugado, por exemplo a Figura 8-3. Esta oscilação existe porque as tensões das três fases não são exatamente iguais, portanto o sistema está em desequilíbrio. Este efeito foi observado também na simulação, mostrado na Figura 6-25, e é

devido à variação do fluxo no tempo, mostrado na Figura 6-26. Para se determinar os valores do conjugado estimado (Tabela 8-2) a rotina em linguagem DELPHI armazena o maior e o menor valor do conjugado estimado , estes valores são somados e o resultado da soma é dividido por dois, resultando no valor central do conjugado estimado.

A Figura 8-12 a seguir mostra o conjugado em função da velocidade, onde observa-se o aumento do conjugado em função da diminuição da velocidade.

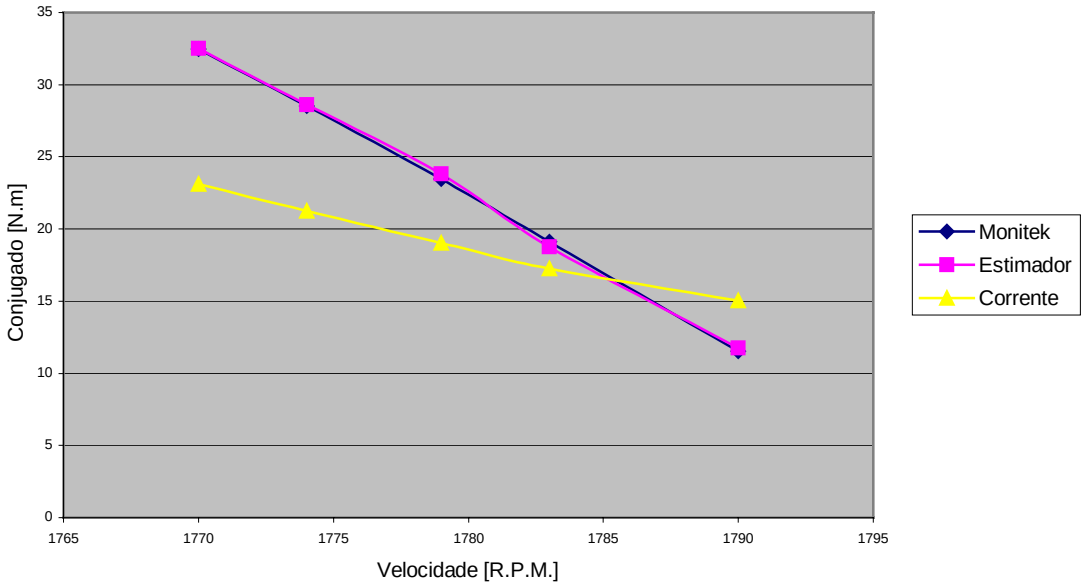

Figura 8-12 – Conjugado em Função da Velocidade

O erro entre o conjugado medido e o conjugado estimado pode ser observado na Figura 8-13, o erro máximo não atinge 2%.

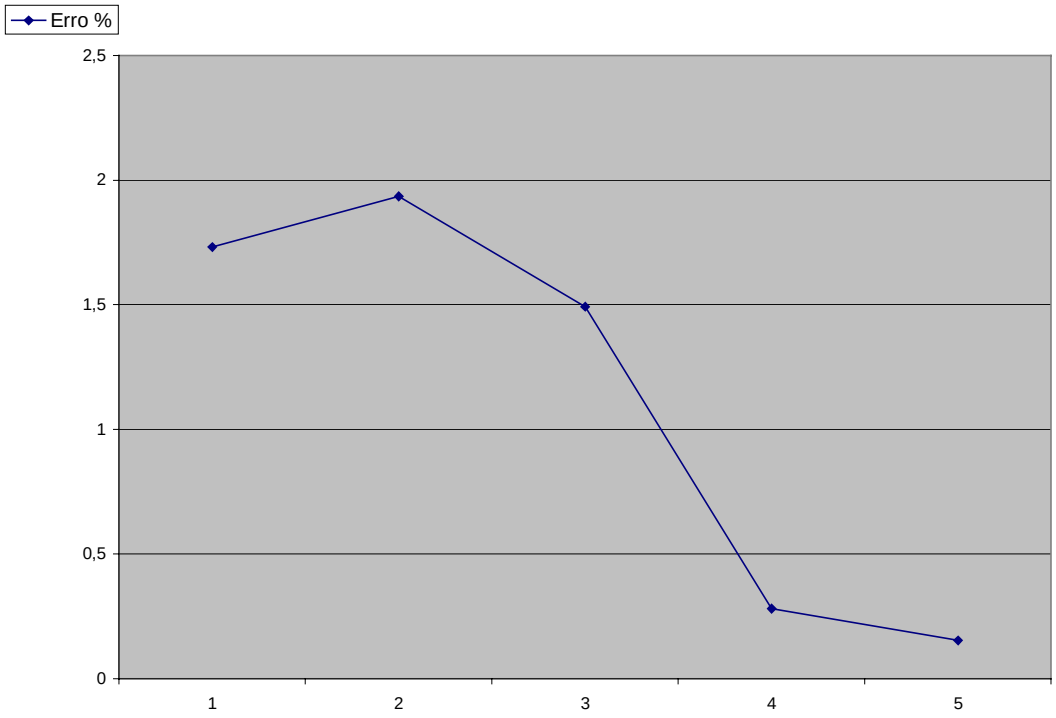

Figura 8-13 – Erro, Conjugado Medido menos Conjugado Estimado

Para o gráfico da Figura 8-13 foi utilizada a equação:

$$Erro\% = \left| \frac{ConjugadoMedido - ConjugadoEstimado}{ConjugadoMedido} \right| * 100\%$$

Uma tabela mais completa com os dados fornecidos pelo Monitek é apresentada a seguir.

Rotação [RPM]	Conjugado [N.m]	Potência Mecânica [kW]	Corrente [A]	Tensão [V]	Potência Elétrica [kW]	cos ϕ	Rendimento %	s %	Perdas [kW]
1798	1,53	0,2881	13,11	224,2	0,9522	0,188	32,2	0,1	0,6745
1790	11,55	2,1652	15,04	224,4	2,9063	0,500	74,5	0,6	0,7503
1783	19,75	3,5698	17,25	223,3	4,3906	0,661	81,3	0,9	0,8227
1779	23,45	4,3693	19,02	223,6	5,2532	0,716	83,2	1,1	0,8775
1774	28,52	5,2980	21,24	223,4	6,2836	0,767	84,3	1,4	0,9780
1770	32,45	6,0141	23,10	223,3	7,1007	0,797	84,7	1,7	1,0861
1766	35,22	6,5132	25,33	223,7	8,0394	0,820	81,0	1,9	1,5454

Tabela 8-3 – Tabela Resumida dos Dados Obtidos Através do Monitek

9- CONCLUSÕES E SUGESTÕES

O desenvolvimento, a implementação e a disponibilização de um estimador de conjugado para motores de indução trifásicos utilizando DSP, com a técnica de filtros passa-baixas em cascata programáveis, construídos com auxílio de redes neurais treinadas com algoritmo de filtro de Kalman torna-se mais uma opção para as ferramentas usadas no controle vetorial destes motores.

Do trabalho desenvolvido pode-se tirar as seguintes conclusões e sugestões:

- a utilização do DSP torna o processo estimação mais rápido e portanto mais preciso. Considere que um ciclo de máquina (tempo necessário para executar uma instrução) do DSP 56002 dura dois períodos de *clock* (2T). Operando em 40 MHz (25 ns) o tempo total para coletar as amostras dos três sinais de corrente, dos três sinais de tensão, digitalizar as amostras e calcular o conjugado a partir das amostras digitalizadas, é de aproximadamente 68 μs. Para comparação foi considerado um microcontrolador da família do 8051, um microcontrolador bastante popular, executando o mesmo algoritmo da rotina utilizada no DSP 56002. O ciclo de máquina deste microcontrolador é de vinte e quatro períodos de *clock* (24T), operando também em 40 MHz o microcontrolador levaria quase 600 μs para fazer a mesma operação com as amostras dos sinais digitalizados. O gráfico da Figura 9-1 compara o tempo gasto pelo DSP 56002 e o tempo gasto por um microcontrolador da família do 8051.

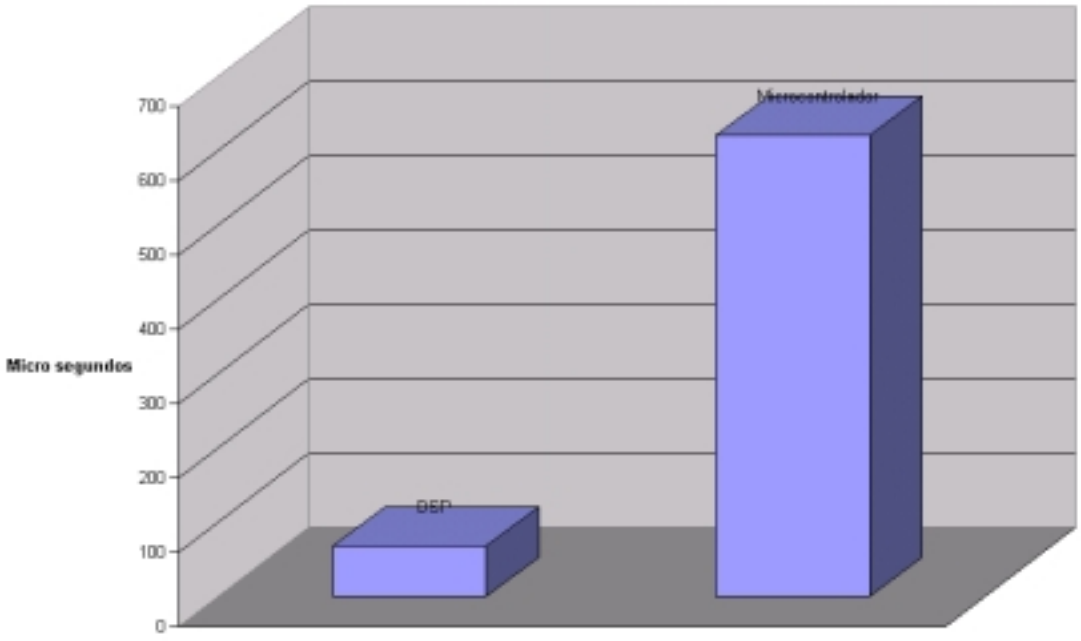

Figura 9-1 – Comparação entre o Tempo Gasto pelo DSP e pelo Microcontrolador 8051

O tempo gasto pelo microcontrolador 8051 é apenas uma estimativa, nenhum teste foi feito neste sentido.

A freqüência de amostragem é de 8 kHz (125 μs), portanto se o estimador de conjugado for implementado com o microcontrolador 8051, não será possível processar as amostras em 125 μs, será necessário diminuir a taxa de amostragem.

Além do problema do tempo, o microcontrolador da família do 8051 não trabalha com números fracionários, o que causa uma dificuldade ainda maior, pois a implementação do estimador do conjugado dinâmico, é baseada em operações aritméticas com números fracionários.

- a literatura disponível sobre o assunto, considera apenas a resistência do enrolamento do estator como parâmetro para correção entre a tensão aplicada no motor e a f.e.m. induzida no ramo de magnetização do circuito

equivalente. Vale comentar que o correto seria considerar a impedância do enrolamento, pois a reatância de dispersão do mesmo é, principalmente para motores pequenos (P_N < 30 kW), considerável. Acrescente-se a isto o método clássico de medição da resistência, que introduz erro, uma vez que não considera o efeito pelicular.

- para motores grandes, o erro introduzido com a medição apenas da resistência do enrolamento do estator é pequeno, pois a impedância deste enrolamento é muito baixa. O erro pode tornar-se sensível para motores de pequeno porte.

- a comparação entre os valores medidos e estimados do conjugado mostrou que para o motor ensaiado o erro foi muito baixo, mesmo o motor sendo de pequeno porte. Sugere-se então que o equipamento desenvolvido seja testado em outros motores de diversos tipos e fabricantes. A título de exemplo, pode-se supor que o resultado para motores de alto rendimento seja diferente do resultado para motores normais (devido a grande diferença entre os valores da resistência do estator de motores de mesma potência).

- além da utilização no controle vetorial, este estimador de conjugado pode ter outras aplicações importantes, que recomenda-se sejam verificadas em outros trabalhos:

  - medição em tempo real do rendimento do motor.
  - medição da potência entregue pelo motor, com a finalidade de verificar o sobre ou subdimensionamento do mesmo.

- recomenda-se substituir a medição da resistência pelo cálculo da impedância do enrolamento do estator, através da utilização de uma rotina de software que determine o circuito equivalente do mesmo, a partir dos dados da placa.

- neste trabalho o treinamento da rede neural foi feito apenas para a freqüência de 60 Hz, portanto a constante de tempo (τ) dos componentes do filtro e o ganho de compensação da amplitude (G) que são mostrados na Figura 4-1, são valores fixos, ou seja, os pesos W da rede neural obtidos pelo treinamento são constantes, válidos apenas para a freqüência de 60Hz. Como a constante de tempo (τ) e o ganho de compensação da amplitude (G) do PCLPF são funções não lineares da freqüência, se for desejado uma estimação de conjugado em qualquer freqüência, devemos treinar a rede neural em toda a faixa de freqüências desejada, para manter o deslocamento de fase e o ganho de cada filtro idênticos para qualquer freqüência. Com os pesos W da rede encontrados para cada freqüência deve ser montada uma tabela que será utilizada pela rotina em linguagem *assembly*, para cada freqüência a rotina busca os pesos W correspondentes na tabela.

10- LISTA DE ACRÔNIMOS

AGND – Analogic Ground

ALU – Aritmetic and Logic Unit

ANN – Artificial Neural Network

ASIC – Application Specific Integrated Circuit

CA – Corrente Alternada

CC – Corrente Contínua

CODEC – Codificador-Decodificador

DGND – Digital Ground

DSP – Digital Signal Processing

EKF – Extended Kalmam Filter

E^2PROM – Electrically Erasable Programmable Read-Only Memory

EVM – Evaluation Module

FFANN – Feed Forward Artificial Neural Network

FMM – Força Magneto Motriz

GMDH – Group Method of Data Handling

GND – Ground

GUI – Graphic User Interface

MIT – Motor de Indução Trifásico

NC – Not Connected

PBC – Port B Control Register

PBD – Port B Data Register

PBDDR – Port B Data Direction Register

PC – Personal Computer

PCLPF – Programmable Cascaded Low-Pass Filter

RAM – Random Access Memory

RNN – Recurrent Neural Network

SMD – Surface Mounted Devices

SSI – Serial Synchronous Interface

ZOH – Zero Order Hold

11- BIBLIOGRAFIA

1] K. Hasse, “On the Dynamics of Speed Control Static AC Drive with Squirrel-Cage Induction Machine”, Ph. D. Dissertation, Techinsche Hochschule Darmstadt, 1969.

2] F. Blaschke, “The Method of Field Orientation for Control of Three Phase Machine”, Ph. D. Dissertation, TU Braunschweig, 1974.

3] Müller E. Luiz Jr., Identificação da Resistência Rotórica e Estimação de Estados do Motor de Indução Alimentado por Inversor PWM, com Utilização de Filtros de Kalman”, Dissertação de Mestrado, Escola Federal de Engenharia de Itajubá, 1991.

4] B. K. Bose and N. R. Patel, “A Programmable Cascaded Low-Pass Filter-Based Flux Synthesis for a Stator Flux-Oriented Vector-Controlled Induction Motor Drive”, IEEE Trans. Ind. Electron., vol. 44, pp. 140-143, Feb. 1997.DSP56000 Family Manual – DSP56KFAMUM/AD, Motorola

5] Luiz E. B. da Silva, Bimal K. Bose, and João O. P. Pinto, “Reccurrent-Neural-Network-Based Implementation of a Programmable Cascaded Low-Pass Filter Used in Stator Flux Synthesis of Vector-Controlled Induction Motor Drive”, IEEE Trans. Ind. Electron., vol. 46, no.3, pp. 662-665, Jun. 1999.

6] João O. P. Pinto, Bimal K. Bose and Luiz E. B. da Silva, “A Stator Flux Oriented Vector-Controlled Induction Motor Drive With Space Vector PWM and Flux Vector Synthesis by Neural Networks”, IEEE Trans. Ind. Electron., vol. 37, no.5, pp. 1308-1318, Set/Out. 2001.

7] Luciana Altino, “Máquinas Síncronas: Teoria e Aplicações”, Editora Universitária – Universidade Federal de Pernambuco, Cap. 3 pp. 57 a 61, 1984.

8] Ivo Barbi, “Teoria Fundamental do Motor de Indução” Universidade Federal de Santa Catarina, 1985.

9] F. P. de Melo, “Dinâmica das Máquinas Elétricas”, Universidade Federal de Santa Maria, 1979

10] B. K. Bose, “Power Eletronics and AC Drives”, Englewood Cliffs, N. J., Prentice Hall, 1987.

11] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of nonlinear dynamic systems with Kalman filter trained recurrent networks“, *IEEE Trans. Neural Networks*, vol. 5, pp. 279-297, Mar. 1994.

12] DSP56002 User’s Manual – DSP56002UM/AD, Motorola

13] DSP56002EVM Quick Start Guide, Motorola

14] Simon Haykin, “Neural Networks a comprehensive foundation”, Second Edition *Prentice Hall*, 1999.

15] K. Gulez, B. Karlik, S. Vakkas Ustun, “Designing Artificial Neural Networks for Fault Detection in Induction Motors with the TMS320C30 DSP” – ESIEE, Paris September 1996 SPRA333

16] A. P. A. Silva, P. C. Nascimento, G. L. Torres, and L. E. Borges da Silva, “Alternative Approach for adaptive real-time control using a nonparametric neural network”, in *Conf. Rec. IEEE-IAS Annu. Meeting,* 1995, pp. 1788-1794.

12- ANEXOS

12.1 Placa para o ADS7864

12.1.1. Esquema Elétrico

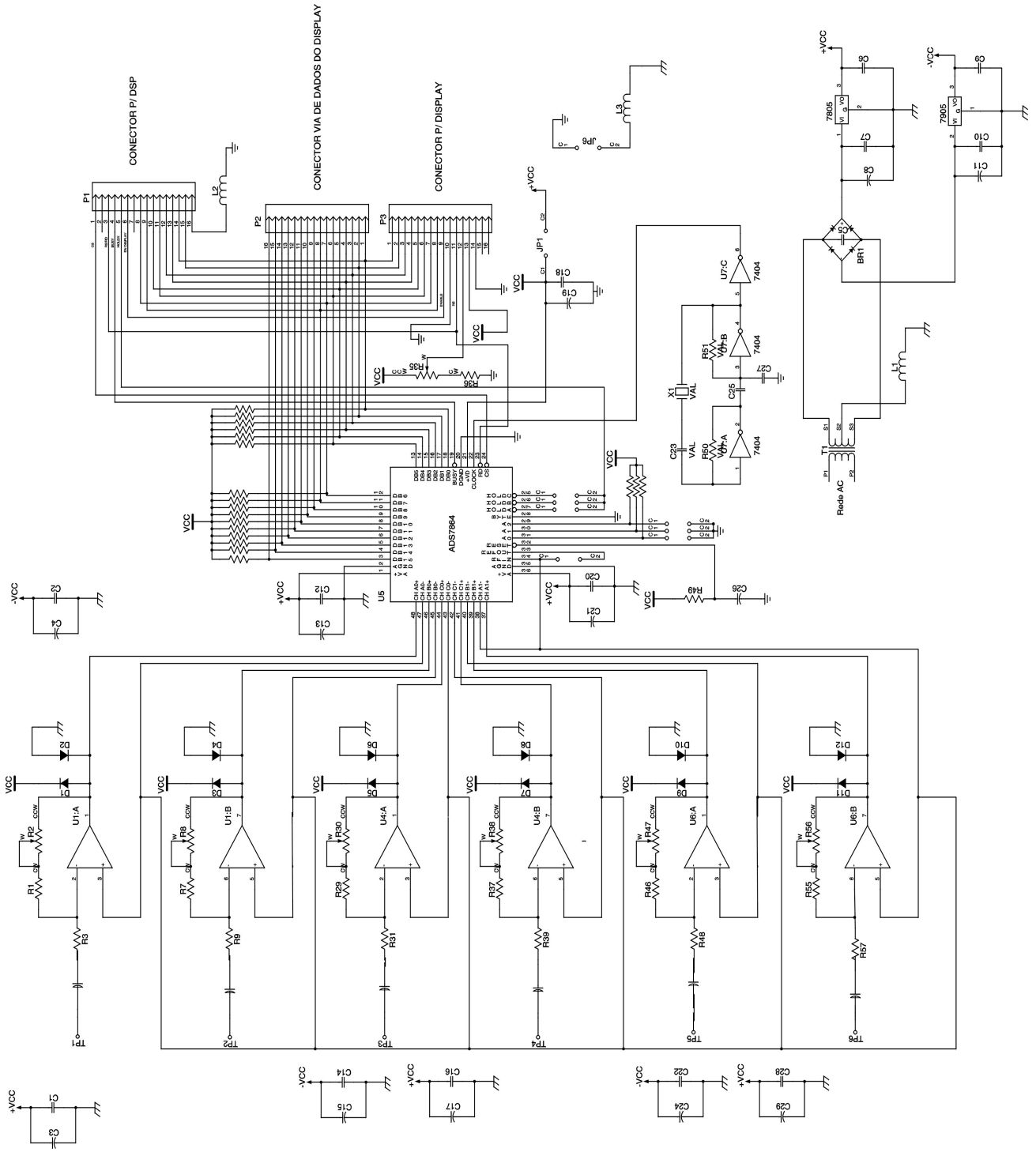

Figura 12-1 – Esquema Elétrico da Placa do Conversor A/D – ADS7864

12.1.2. Placa de Circuito Impresso

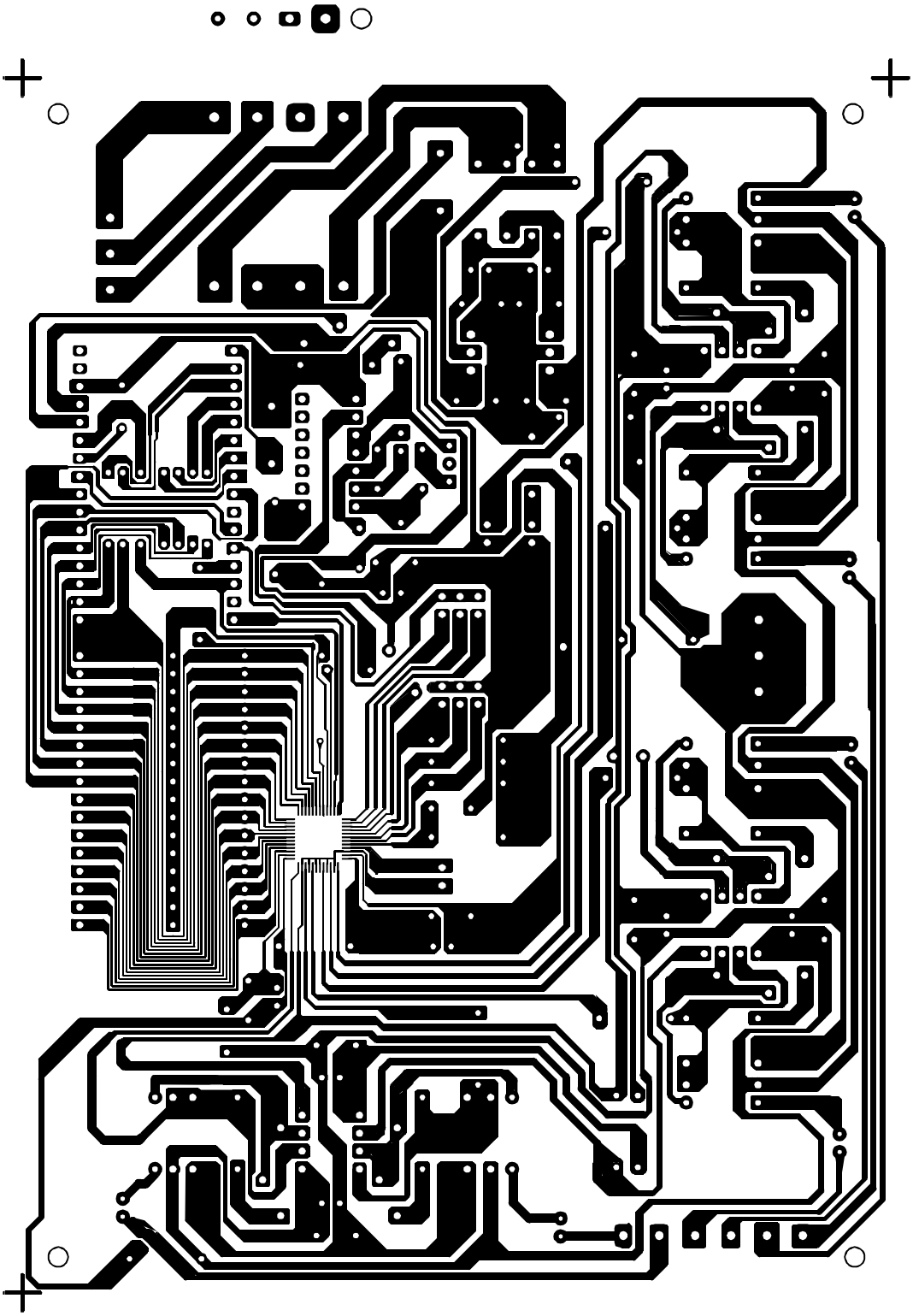

Figura 12-2 – Placa de Circuito Impresso para o Conversor A/D, ADS7864
Lado da Solda

12.2 O DSP

12.2.1. Portas A, B e C

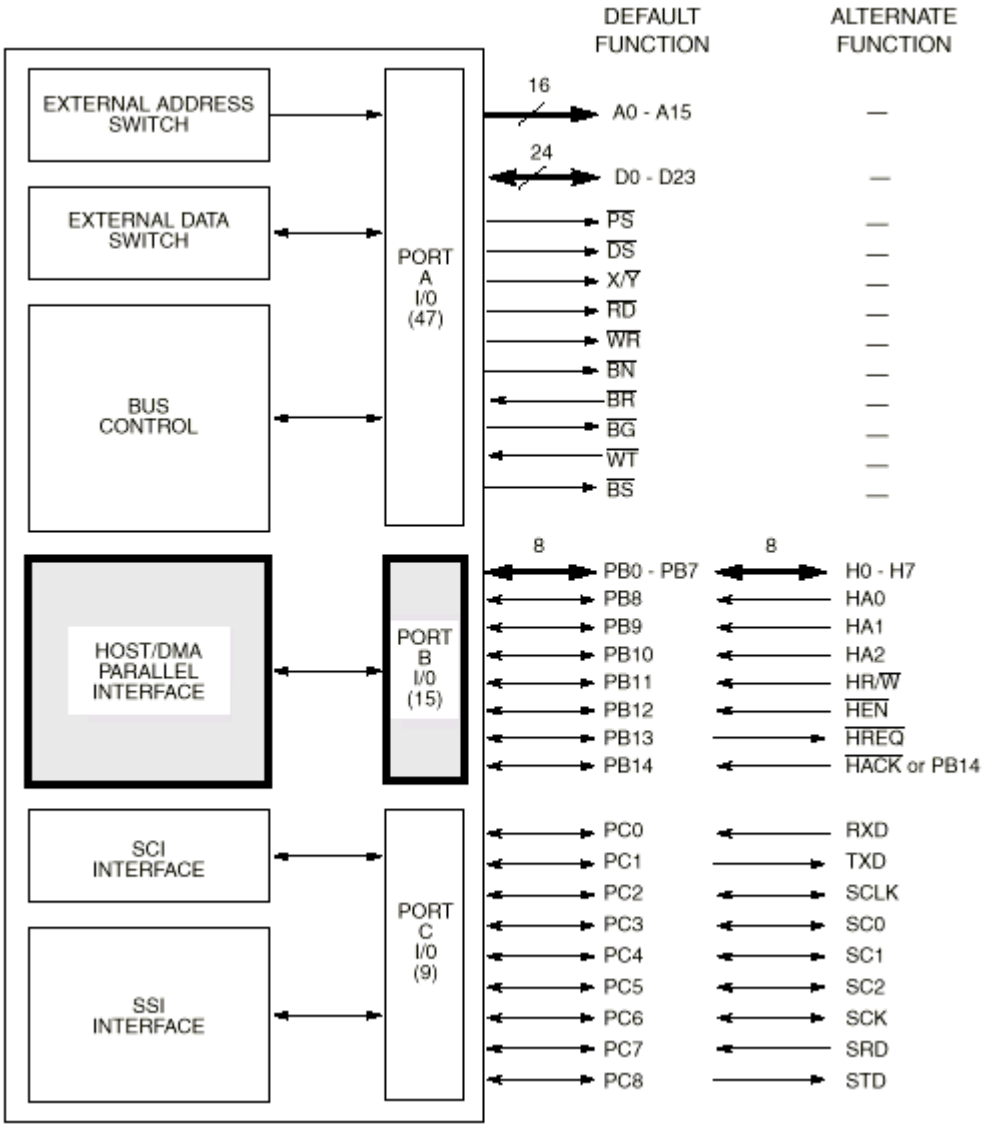

Figura 12-3 – Portas de Comunicação do DSP 56002

12.2.2. Emulador de Chip

A família DSP56K dispõe de circuitos internos específicos para possibilitar a emulação do *chip* (*on-chip emulator* – OnCE). O OnCE é uma ferramenta de depuração sofisticada que permite o acesso aos registros e periféricos internos do processador de maneira simples, inexpensiva e independente da velocidade de processamento. O emulador disponibiliza uma série de informações precisas sobre o estado dos registros, posições de memória, barramentos e, ainda, as últimas cinco instruções executadas.

12.2.3. Conjunto de Instruções

Como a maioria dos processadores, o DSP56000 possui os seguintes grupos de instruções:

- *Grupo Aritmético*: - Composto por instruções que realizam operações aritméticas sobre os operandos em memória, tais como: adição, subtração, multiplicação, divisão, comparação, etc.. Um subgrupo de instruções aritméticas muito importante em DSP é o das instruções de multiplicação e acumulação concatenada (MAC) que, na realidade, executa duas operações aritméticas, multiplicação e adição, em uma única instrução. Os algoritmos de filtragem digital são baseados, geralmente, em operações MAC.

- *Grupo Lógico*: - Reúne as instruções que realizam operações de lógica binária (bit a bit) sobre os operandos. As operações mais comuns são: AND, OR, EOR, SHIFT e ROTATE.

- *Grupo de Manipulação de Bit*: - Composto por instruções capazes de testar e modificar bits individuais dos operandos (palavras de memória): BCLR, BSET, etc..

- *Grupo de Movimentação de Dados*: - São as instruções capazes de transferir dados entre posições de memória e/ou registros. A instrução típica é a de MOVE.

- *Grupo de Controle de Programa*: - Reúne aquelas instruções que controlam a seqüência de operações do programa, tais como: desvio de programa (condicional e incondicional), chamada e retorno de subrotina, etc. Em DSP, são particularmente importantes as instruções de controle de *loop* de programa (DO - ENDDO) e de repetição de instruções (REP).

A maioria das instruções da família DSP56K são similares a de outros microprocessadores convencionais. As maiores diferenças estão associadas às instruções aritméticas, conduzidas em representação numérica fracionária de ponto fixo. Além disso, a instrução mais importante (MAC), normalmente, não existe nos processadores convencionais.

• **Bit Field** Manipulation BFTSTL BFTSTH BFCLR BFSET BFCHG • **Loop** DOLoop DO FOREVER ENDDO	BRKcc • **Move** LEA MOVE MOVE(C) MOVE(I) MOVE(M) MOVE(P) MOVE(S) • **Arithmetic** ABS	ADC ADD ASL ASL4 ASR ASR4 ASR16 CLR CLR24 CMP CMPM	DEC DEC24 DIV DMAC IMPY INC INC24 MAC MACR MPY MPYR	MPY(su,uu) MAC(su,uu) NEG NEGC NORM RND SBC SUB SUBL SWAP Tcc	TFR TFR2 TST TST2 ZERO • **Logical** AND ANDI EOR LSL LSR	NOT OR ORI ROL ROR • **Program Control** Bcc BSR BRA BScc	DEBUG DEBUGcc Jcc JMP JSR JScc NOP REP REPcc RESET RTI	RTS STOP SWI WAIT

Tabela 12-1 – Conjunto de Instruções do ADS864

12.2.4. Vetor de Interrupções

A Figura 12-4 mostra o endereço inicial de cada interrupção e as fontes de interrupção do DSP56002.

Interrupt Starting Address	IPL	Interrupt Source
$0000	3	Hardware $\overline{\text{RESET}}$
$0002	3	Stack Error
$0004	3	Trace
$0006	3	SWI
$0008	0-2	$\overline{\text{IRQA}}$
$000A	0-2	$\overline{\text{IRQB}}$
$000C	0-2	SSI Receive Data
$000E	0-2	SSI Receive Data with Exception Status
$0010	0-2	SSI Transmit Data
$0012	0-2	SSI Transmit Data with Exception Status
$0014	0-2	SCI Receive Data
$0016	0-2	SCI Receive Data with Exception Status
$0018	0-2	SCI Transmit Data
$001A	0-2	SCI Idle Line
$001C	0-2	SCI Timer
$001E	3	NMI
$0020	0-2	Host Receive Data
$0022	0-2	Host Transmit Data
$0024	0-2	Host Command (default)
$0026	0-2	Available for Host Command
•		•
•		•
•		•
$003A	0-2	Available for Host Command
$003C	0-2	Timer
$003E	3	Illegal Instruction
$0040	0-2	Available for Host Command
•		•
•		•
•		•
$007E	0-2	Available for Host Command

Figura 12-4 – Endereços das Interrupções

12.2.5. Arquivo de Configuração para o Conversor A/D e D/A CS4215

```
;*********************************************************
;    ADA_INIT.ASM    ver.1.1
;    Initialization for the Crystal Semi. CS4215 A/D &
;          D/A Converters
;    NOTES:
;1-The DSP5002EVM analog inputs are connected to "MIC"
;input pins only.  The "LINE" input pins are connected
;to analog GROUND.  Line level inputs are supported
;by disabling the 20 dB Microphone Pre-Amplifier.
;2-  As shipped, the DSP56002EVM has a 24.576 MHz crystal
;connected to the XTAL2 inputs of the CD4215.  Do
;not select any other clock source unless the board
```

```
;has been modified to support the alternate clock
;source(s).
;;   Copyright (c) MOTOROLA 1995
;               Semiconductor Products Sector
;               Digital Signal Processing Division
;
;*********************************************************
;      portc usage:
;         bit8: SSI TX (from DSP to Codec)
;    bit7: SSI RX (from Codec to DSP)
;    bit6: SSI Clock
;    bit5: SSI Frame Sync
;    bit4: codec reset (from DSP to Codec)
;    bit3:
;         bit2: data/control bar
;             0=control
;             1=data
;*********************************************************
;*********************************************************
;  Defined Constants used to construct data for the CONTROL
TIME SLOTS
;*********************************************************
NO_PREAMP       equ     $100000 ;0 == enable 20 dB pre-amp
LO_OUT_DRV      equ     $080000 ;0 == 2.8 Vp-p line (1Vrms)
                                ;0 == 4.0 Vp-p headphones
                           ;1 == Line and Headphone 2.0Vp-p
HI_PASS_FILT    equ     $008000 ;0 == HPF disabled
SAMP_RATE_9     equ     $003800 ; 9.6 kHz sample rate
SAMP_RATE_48    equ     $003000 ;48   kHz sample rate
SAMP_RATE_32    equ     $001800 ;32   kHz sample rate
SAMP_RATE_27    equ     $001000 ;27.4 kHz sample rate
SAMP_RATE_16    equ     $000800 ;16   kHz sample rate
SAMP_RATE_8     equ     $000000 ; 8   kHz sample rate
STEREO          equ     $000400 ;1 == stereo, 0 == mono
DATA_8LIN       equ     $200300 ; 8-bit unsigned linear
DATA_8A         equ     $200200 ; 8-bit A-law
DATA_8U         equ     $200100 ; 8-bit u-law
DATA_16         equ     $200000;16-bit 2s complement linear
IMMED_3STATE    equ    $800000;1=SCLK&FS3-state immediately
XTAL2_SELECT    equ     $200000 ;REQUIRED is the only clock
                                ;  source on the board
BITS_64         equ     $000000 ; 64 bits per frame
BITS_128        equ     $040000 ;128 bits per frame
BITS_256        equ     $080000 ;256 bits per frame
CODEC_MASTER    equ     $020000 ;1==codec generates SCLK&FS
                                ;0==codec receives SCLK& FS
CODEC_TX_OFF    equ     $010000 ;0 == enable codec TX toDSP
                              ;1==disable(Hi-Z)codec output
```

```
;*********************************************************
;Defined Constants construct data for DATA TIME SLOTS(5-8)
;*********************************************************
HEADPHONE_EN equ $800000 ;1==headphone output enabled, 0 ==
                         ; muted
LINEOUT_EN   equ $400000 ;1==line output enabled, 0 ==
                         ; muted
LEFT_ATTN    equ $010000 ;63 steps * 1.5 dB = -94.5 dB
SPEAKER_EN   equ $004000 ;1 == speaker output enabled, 0 ==
                         ; muted
RIGHT_ATTN   equ $000100 ;63 steps * 1.5 dB = -94.5 dB
MIC_IN_SELECT equ $100000 ;1==A/D inputs MIC pins NOTE:
;DSP56002EVM uses these pins.line input pins are not used.
LEFT_GAIN    equ $010000 ;15 steps * 1.5 dB =  22.5 dB
MONITOR_ATTN equ $001000 ;15 steps*6.0 dB =  90.0 dB (mute)
RIGHT_GAIN   equ $000100 ;15 steps * 1.5 dB =  22.5 dB
;---------------------------------------------------------
;  constructed constants for codec set up/initialize
;---------------------------------------------------------
CTRL_WD_12      equ
NO_PREAMP+HI_PASS_FILT+SAMP_RATE_8+STEREO+DATA_16   ;CLB=0
CTRL_WD_34      equ
IMMED_3STATE+XTAL2_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56      equ     $000000
CTRL_WD_78      equ     $000000
;---------------------------------------------------------
;  constructed constants for codec data mode
;---------------------------------------------------------
OUTPUT_SET      equ
HEADPHONE_EN+LINEOUT_EN+(LEFT_ATTN*4)
INPUT_SET       equ
MIC_IN_SELECT+(15*MONITOR_ATTN)+(RIGHT_ATTN*4)
;---DSP56002 on-chip peripheral addresses
IPR             equ     $FFFF   ;Interrupt Priority
Register
BCR             equ     $FFFE   ;Bus Control Register
PLL             equ     $FFFD   ;PLL Control Register
SSIDR           equ     $FFEF   ;SSI Data Register
SSISR           equ     $FFEE   ;SSI Status Register
CRB             equ     $FFED   ;SSI Control Register B
CRA             equ     $FFEC   ;SSI Control Register A
PCD             equ     $FFE5   ;Port C Data Register
PBD             equ     $FFE4   ;Port B Data Register
PCDDR           equ     $FFE3   ;Port C Data Direction
Register
PBDDR           equ     $FFE2   ;Port B Data Direction
Register
PCC             equ     $FFE1   ;Port C Control Register
PBC             equ     $FFE0   ;Port B Control Register
```

```
;---------------------------------------------------------
; Two buffers which are defined below are the source and
; destination storage for the codec Input/Output ISRs
;---------------------------------------------------------
       org    x:0
RX_BUFF_BASE    equ     *
RX_data_1_2     ds      1       ;data time slot 1/2 for RX
ISR
RX_data_3_4     ds      1       ;data time slot 3/4 for RX
ISR
RX_data_5_6     ds      1       ;data time slot 5/6 for RX
ISR
RX_data_7_8     ds      1       ;data time slot 7/8 for RX
ISR
TX_BUFF_BASE    equ     *
TX_data_1_2     ds      1       ;data time slot 1/2 for TX
ISR
TX_data_3_4     ds      1       ;data time slot 3/4 for TX
ISR
TX_data_5_6     ds      1       ;data time slot 5/6 for TX
ISR
TX_data_7_8     ds      1       ;data time slot 7/8 for TX
ISR
RX_PTR          ds      1       ; Pointer for rx buffer
TX_PTR          ds      1       ; Pointer for tx buffer
;*********************************************************
;****     initialize the CS4215 codec                  ****
;*********************************************************
;1.Configure SSI port and the GPIO lines (D/C~ & Reset)
;2.Select Control Mode and reset the codec (50ms @ 40MHz)
;3.Initialize Trans. Data Buffer with Control Words (CLB=0)
;4.  Enable SSI
;5.  Wait until CLB in receive buffer = 0
;6.  Set CLB = 1 in TX buffer
;7.  Send 4 frames of data
;8.  Disable SSI port
;9.  Program SSI to receive Frame Sync and Clock
;10.  Select Data Mode (D/C~ = 1)
;11.  Enable SSI port
;*********************************************************
        org     p:
codec_init
     move    #RX_BUFF_BASE,x0
     move          x0,x:RX_PTR   ; Initialize the rx pointer
     move    #TX_BUFF_BASE,x0
     move          x0,x:TX_PTR   ; Initialize the tx pointer
;----- initialize SSI -----
; Assuming the DSP56002 at 40 MHz,  this section selects a
;2.5 MHz SSI clock to be sent from the 56002's SSI clock
```

```
;to the codec.It also sets up 64-bit frame length(4wordsper
;framex16-bits per word),enables SSI TX&RX, enables TX & RX
;interrupts and sync. operation with bit-length frame sync.
;-----------------------------------------------------------
movep  #$0000,x:PCC;turn off ssi port
movep  #$4303,x:CRA;40MHz/16=2.5MHz SCLK, WL=16 bits,
                              ; 4W/F
movep  #$FB30,x:CRB;RIE,TIE,RE,TE,NTWK,SYN,FSR/RSR->bit
movep   #$14,x:PCDDR    ; setup pc2 and pc4 as outputs
movep   #$0,x:PCD       ; D/C~ and RESET~ = 0 ==> control
mode
                    ;----reset delay for codec ----
do      #500,_delay_loop
rep     #2000           ; 100 us delay
nop
delay_loop
bset    #4,x:PCD        ; RESET~ = 1
movep   #$3000,x:IPR    ; set interrupt priority level
movep   #$01E8,x:PCC    ; Turn on ssi port
;--- set up the TX buffer with control mode data
     move    #CTRL_WD_12,x0
     move                        x0,x:TX_BUFF_BASE
     move    #CTRL_WD_34,x0
     move                        x0,x:TX_BUFF_BASE+1
     move    #CTRL_WD_56,x0
     move                        x0,x:TX_BUFF_BASE+2
     move    #CTRL_WD_78,x0
     move                        x0,x:TX_BUFF_BASE+3

     andi    #$FC,mr           ; enable interrupts

; CLB == 0 in TX Buffer,  wait for CLB == 1 in RX Buffer
     jclr    #3,x:SSISR,*      ; wait until rx frame bit==1
     jset    #3,x:SSISR,*      ; wait until rx frame bit==0
     jclr    #3,x:SSISR,*      ; wait until rx frame bit==1
     jset    #18,x:RX_BUFF_BASE,*    ; loop until CLB set

; CLB == 1 in RX Buffer, send 4 frames and then disable SSI
     bset    #18,x:TX_BUFF_BASE      ;set CLB
     do      #4,_init_loopB ;Delay as 4 full frames to pass
     jclr    #2,x:SSISR,*      ; wait until tx frame bit==1
     jset    #2,x:SSISR,*      ; wait until tx frame bit==0
_init_loopB
     movep   #0,x:PCC          ;reset SSI (disable SSI...)
;CLB should be 1--re-prog.fsync and sclk direction to input
     movep   #$5B00,x:CRB    ; rcv,xmt & int
                           ; ena,netwk,syn,sclk==inp,msb 1st
     movep   #$14,x:PCD  ; D/C~ pin = 1  ==> data mode
     movep   #$01E8,x:PCC; turn on ssi (enable SSI now...)
```

```
;-----------------------------------------------------------
;this is the end of the ada_init.asm routine...
;-----------------------------------------------------------
```

12.2.6. Arquivo de Configuração das Interrupções do Canal Serial para Comunicação com o DSP

```
;*************************************************************
;   TXRX_ISR.ASM
;  File contains the basic Interrupt Service Routines (ISR)
; used to service SSI Transmit and SSI Receive Interrupts:
;               ssi_rx_isr
;               ssi_tx_isr
; Before using these routines, the appropriate interrupt
; vectors must be initialized to jump to the associated
; handler(e.g., P:$000C, the SSI RX interrupt vector should
; have an instruction which forces the DSP to jump to the
; address of ssi_rx_isr).  In addition, the DSP's Interrupt
; Priority Register (IPR) must choose interrupt level for
; the SSI interrupts and the DSP56002's Mode Register must
; enable that interrupt level.
; ISRs which follow require that r6 has been established as
; a pointer to a stack which grows towards increasing addr.
;*************************************************************
;    Copyright (c) MOTOROLA 1995
;               Semiconductor Products Sector
;               Digital Signal Processing Division
;*************************************************************
;-----------------------------------------------------------
;SSI Receive ISR
;Interrupt Service Routine is the destination of the SSI RX
;vector located at p:$000C.In many cases SSI RX w/Exception
;vector will also jump here.
;R6 should point the first free location on the stack (in
;X:memory).
;The data is placed in a 1 frame (4 word) buffer and sync.
;is verified/restored every frame.
;-----------------------------------------------------------
ssi_rx_isr
move    r0,x:(r6)+         ; Save r0 to the stack.
move    m0,x:(r6)+         ; Save m0 to the stack.
move    #3,m0              ; Modulo 4 buffer.
move    x:RX_PTR,r0    ; Load the pointer to the rx buffer.
jclr   #3,x:SSISR,next_rx;If not fr.syc,jump to receive dat
move   #RX_BUFF_BASE,r0; If frame sync, reset base pointer.
nop
```

```
next_rx
movep   x:SSIDR,x:(r0)+ ; Read out received data to buffer.
move    r0,x:RX_PTR        ; Update rx buffer pointer.
move    x:-(r6),m0         ; Restore m0.
move    x:-(r6),r0         ; Restore r0.
rti
;-----------------------------------------------------------
;SSI Transmit ISR
;Interrupt Service Routine is the destination of the SSI TX
;vector located p:$0010.many cases the SSI TX w/Exception
;vector will also jump here.
;R6 should point first free location on the stack (in
;  X:memory).
;The data is taken from a 1 frame (4 word) buffer and sync.
;is verified/restored every frame.
;-----------------------------------------------------------
ssi_tx_isr
move    r0,x:(r6)+          ; Save r0 to the stack.
move    m0,x:(r6)+          ; Save m0 to the stack.
move    #3,m0               ; Modulus 4 buffer.
move    x:TX_PTR,r0    ; Load the pointer to the tx buffer.
jclr    #2,x:SSISR,next_tx  ; If not frame sync, jump to
                              ; transmit data.
move    #TX_BUFF_BASE+1,r0  ; If frame sync, reset pointer.
nop
next_tx
movep   x:(r0)+,x:SSIDR     ; SSI transfer data register.
move    r0,x:TX_PTR         ; Update tx buffer pointer.
move    x:-(r6),m0          ; Restore m0.
move    x:-(r6),r0          ; Restore r0.
rti
```

12.3 O EVM

O DSP56002EVM é uma plataforma para familiarizar a utilização do DSP56002 da Motorola. A precisão de 24 bits deste DSP, combinada com as facilidades oferecidas no módulo de 32k words de SRAM e um codec estéreo de áudio, fazem do EVM uma plataforma capaz de implementar e demonstrar diversos algoritmos de processamento digital. O *kit* do EVM inclui um montador assembler (MS-DOS) e um programa de depuração com interface gráfica (Windows). A comunicação entre o PC e o EVM se faz através de uma porta serial RS-232.

12.3.1. Hardware do DSP56002EVM

O módulo de avaliação agrega em uma única placa de circuito impresso todos os componentes eletrônicos necessários para se efetuar experiências com processamento digital de sinais, conforme ilustrado na Figura 12-5.

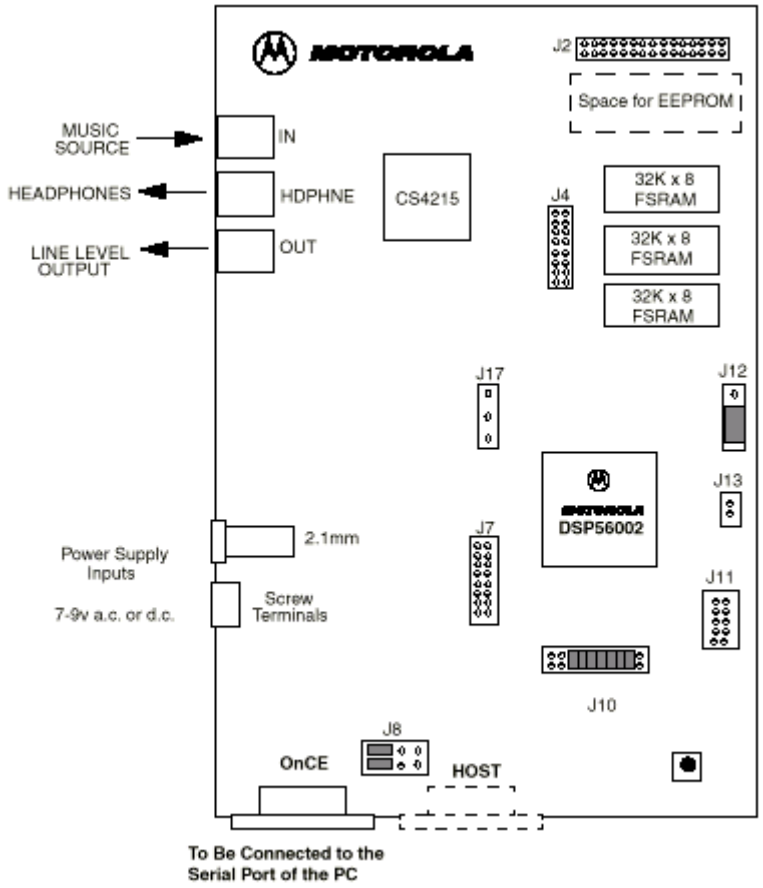

Figura 12-5 – Plataforma de Hardware do DSP56002EVM

A plataforma de hardware do EVM é composto dos seguintes componentes principais:

- DSP56002 operando em 40 MHz.
- CODEC estéreo CS4215 (conversor A/D e D/A *sigma-delta* de 16 bits).

- Cristal de 24,576 MHz para taxas de amostragem de 48, 32, 16, 9.6 e 8 kHz.
- 32k words de RAM ($3 \times 32K \times 8$ bits).
- Microcontrolador de supervisão e gerência.
- Porta comunicação serial RS-232.
- Conectores de entrada e saída de áudio, e saída para fone de ouvido.
- Conector de entrada para fonte de alimentação (7 – 9 Volts AC ou DC).

A Figura 12-6 mostra o diagrama em blocos do DSP56002EVM.

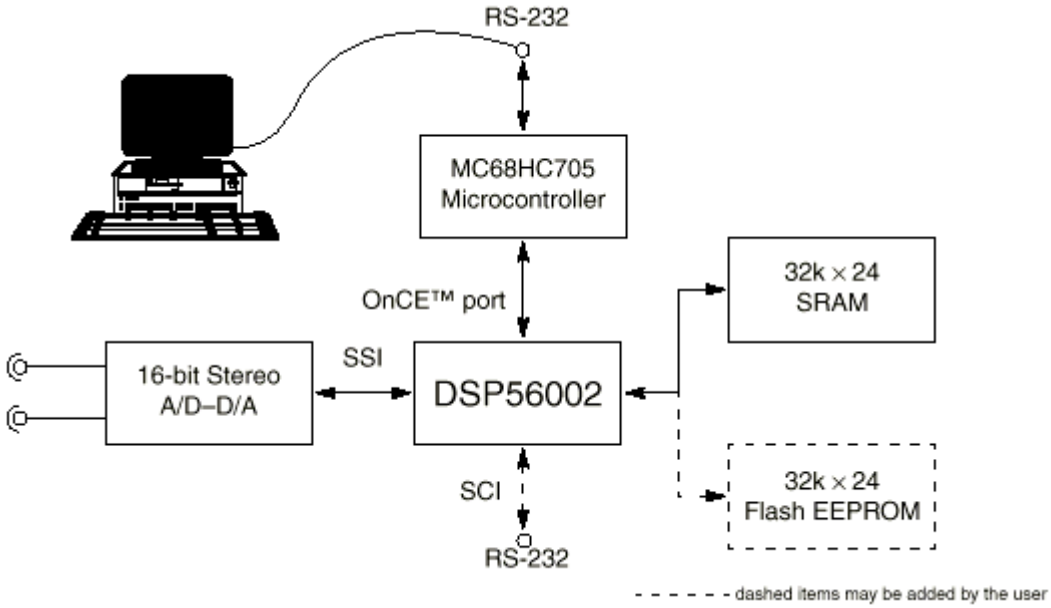

Figura 12-6 – Diagrama em Blocos do DSP56002EVM

12.3.1.1. Depurador e Interface Gráfica do Usuário (GUI)

Para iniciar a GUI deve-se executar o programa “evm56K”. Como resultado, obtém-se uma tela com quatro janelas: DATA, UNASSEMBLE, REGISTERS, e COMMAND.

12.4 Simulação

12.4.1. Diagrama Completo do Modelo do Motor de Indução Trifásico

O modelo do motor de indução trifásico é um modelo d,q,0, onde a alimentação do motor é feita com as componentes dq (duas fases).

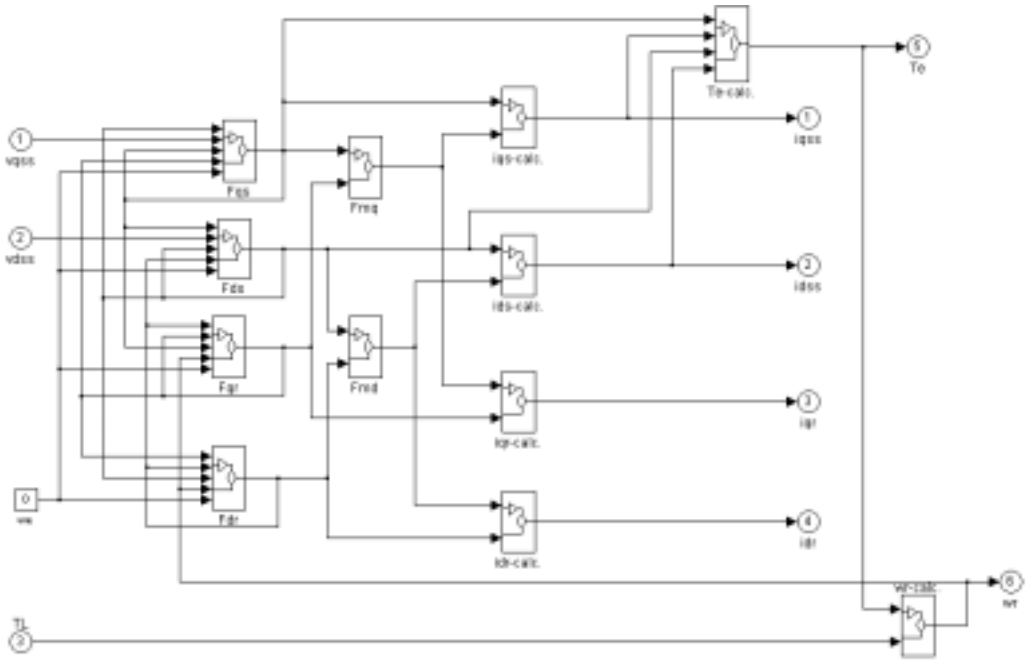

Figura 12-7 – Diagrama do Motor de Indução

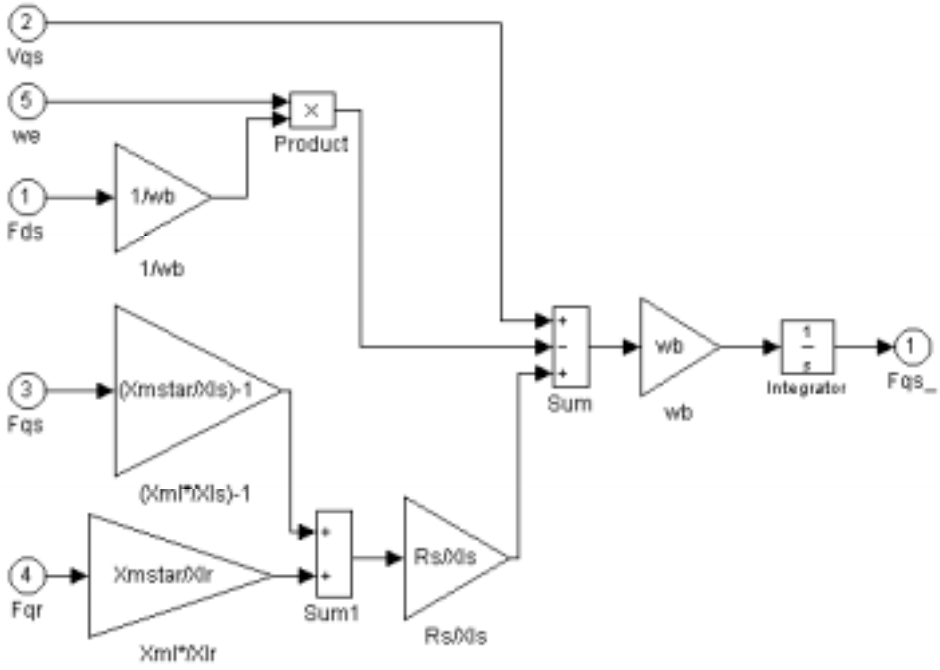

Figura 12-8 – Bloco Fqs

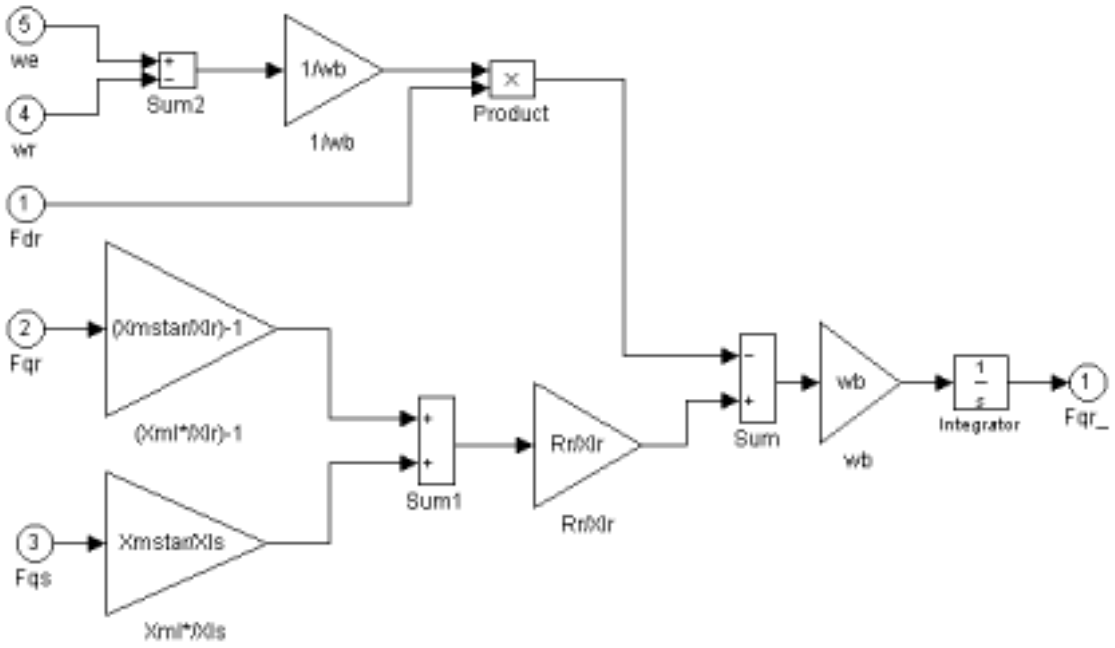

Figura 12-9 – Bloco Fqr

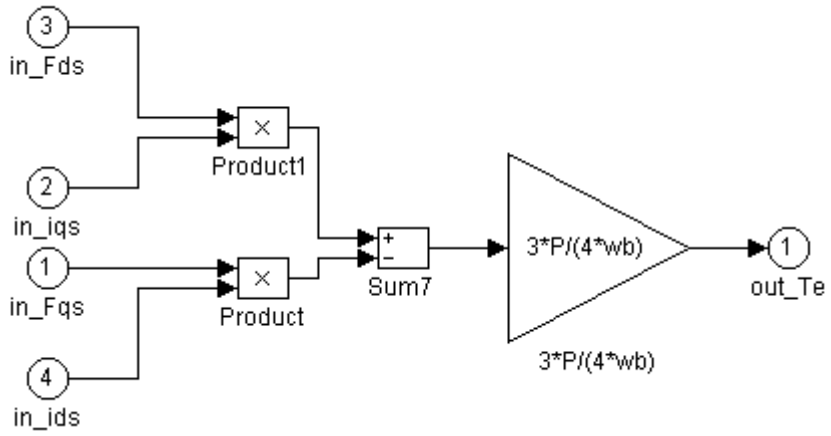

Figura 12-10 – Bloco que fornece o conjugado estimado

12.4.2. Arquivo de Configuração do Modelo do MIT Utilizado na Simulação

```
% 5hp m/c parameters for the d-q induction motor model
% ######################################
% ############# imparam.m ##############
% #####   Programa que gera os   #######
% ### parâmetros do motor de Indução ###
% ######################################
% initialization
Rr=.4165; %rotor resistance
Rs=.5814; %stator resistance
Lls=3.479e-3;  %stator inductance
Llr=4.15e-3; %rotor inductance
Lm=78.25e-3; %magnetizing inductance
fb=60;    %base frequency
P=4;   %number of poles
J=.1;     %moment of inertia
Lr=Llr+Lm;
Tr=Lr/Rr;
Ts=1/(20000);
%impedance and angular speed calculations
%we=2*pi*fe;
wb=2*pi*fb;  %base speed
Xls=wb*Lls;  %stator impedance
Xlr=wb*Llr;  %rotor impedance
Xm=wb*Lm; %magnetizing impedance
Xmstar=1/(1/Xls+1/Xm+1/Xlr);
```

12.5 Configuração da Placa de Coleta de Dados Paralela do PC

Para o correto funcionamento da placa paralela de 8 bits é necessário carregar no registro 303H o valor 82H. O endereço para leitura de dados (entrada da placa) é o endereço 301H.

12.6 Listagem Completa do Programa Utilizado

```
;*************************************************************************
;Torque.ASM :Programa para a estimação do fluxo e calculo do conjugado,
;          a partir das correntes e tensões de linha
;*************************************************************************
START               equ    $40
EN_LATCH            equ    $B
BUSY                equ    $C
EN                  equ    $B
HOLD                equ    $A
RD_RS               equ    $9
CS                  equ    $8
PCTL                equ    $FFFD
TCR                 equ    $FFDF   ;Timer Count Register
TCSR                equ    $FFDE   ;Timer Control and Status Register
TE                  equ    $0
;*************************************************************************
;      Área de memória x usada para parâmetro de entrada
;*************************************************************************
        org    x:$0    ;reservado posição 0 de x
        org    x:$10

Rs      dc      -0.05
                        ;resistência estator, negativa por causa da subtração no
                        ;bloco 5, esta dividida por 9,6 para não ficar maior que 1.
;Coeficientes dos Filtros RNN
w11     dc      0.95799988334559        ;w12
w22     dc      0.04412711596785        ;w23
w21     dc      0.94949390803254        ;w24
w13     dc      0.00025598835534        ;w14
a11     dc      0.66666666666667        ;2/3 (não da certo colocar dc 2/3, tem que
ser 0.666)
a12     dc      -0.33333333333333       ;-1/3
a13     dc      -0.33333333333333       ;-1/3
a21     dc      0
a22     dc      -0.5773502691
a23     dc      0.5773502691
masc    dc      $0000FF
masc2   dc      $000FFF         ;mascara para retirar os bits de end do
                                ;canal, deixando apenas os 12 bits de dado
;******ADS7864******
va      ds      1               ;CHA0
vb      ds      1               ;CHA1
vc      ds      1               ;CHB0
ia      ds      1               ;CHB1
ib      ds      1               ;CHC0
```

```
ic      ds      1               ;CHC1
;*************************************************************************
;       Área de memória y usada para parâmetros de saída(calculados)
;*************************************************************************
        org     y:$10
vqss            ds      1
vdss            ds      1
iqss    ds      1       ;r1  usados no bloco 2.3
idss    ds      1
vqssl           ds      1       ;usados na rotina de calculo de vqss' e vdss'
vdssl           ds      1
somaqss dc      0       ;r2 (somaqss 1/z) (0.57 p/ teste)
yqss            dc      0       ;usados na rot de calc de yqss, bloco 3 (yqss 1/z)
(0.29)
somadss dc              0       ;(somadss 1/z) (0.77 p/ teste)
ydss            dc      0       ;usados na rot de calc de ydss, bloco 4 (ydss 1/z)
(0.16)
produt1 ds              1       ;usados nos blocos 3, 4, 2.2 e 2.3
produt2 ds              1       ;r4
produt3 ds              1
produt4 ds              1
soma            ds      1       ;r5
soma1  ds               1
sint    ds      1       ;r0
cost    ds      1
yqs     ds      1       ;r7
yds     ds      1
iqs     ds      1
ids     ds      1
torq_a  ds              1
torq_b  ds              1
seno            ds      1
TORQUE  ds              1
DECIMAL ds1
contador dc             0
contcanais  dc          0       ;contador para fase b
contc   dc              0       ;contador para fase c
aux     ds      1
inf     ds      1
;*************************************************************************
; Polinomio p/ calculo da raiz quadrada, valido para:  .5<= x < 1.0
;sqrt(x)=a2*x^2+a1*x+a0
;sqrt(x)=-.1985987*x*x+.8803385*x+.3175231
pcoef   dc      .8803385,-.1985987,.3185231     ;a1,a2,a0
;*************************************************************************
;###### Macro converte num. o formato MN e coloca resultado em a
CONVMN  macro xmn                       ;macro definition
        move #(xmn-@cvi(xmn)),x0        ;fractional part to X0
```

```
        move #$010000,y1                ;shift constant in y1
        mpyr y1,x0,a #@cvi(xmn),x1      ;shift X0 ;integer part in X1
        add x1,a                        ;concatenate int. and fract.
        endm
;###### Macro tira raiz quadrada de x0 e devolve raiz em x0
sqrt    macro
        mpyr    x0,x0,a y:(r2)+,y1      ;x**2, pega a1
        mpy     x0,y1,a a,x1 y:(r2)+,y1         ;a1*x, move x**2, pega a2
        macr    x1,y1,a y:(r2)+,y1      ;a2* x**2, pega a0
        add     y1,a                    ;soma com a0
        move    a,x0                    ;resultado em x0
        endm
        org     p:$0
inicio  jmp     START
        org     p:$000c
        jsr     ssi_rx_isr      ;SSI receive data
        jsr     ssi_rx_isr      ;SSI receive data with exception
        jsr     ssi_tx_isr      ;SSI transmit data
        jsr     ssi_tx_isr      ;SSI transmit with exception
        org     p:$003c         ;interrupção do timer
        jsr     loop_1
        org     p:START
        ori     #3,mr           ;não precisa pois cond reset = 03 IPL0,1,2
mascarados
        movec   #0,sp           ;clear hardware stack pointer
        move    #0,omr          ;operating mode 0
        move    #$40,r6         ; initialise stack pointer(esta na memoria x, cuidado
para
                                ; nao atingir este endereco 40h)
        move    #-1,m6          ; linear addressing
        movep   #0000,x:BCR     ; zera os wait states da porta A (reset=FFFF)
        movep   #$260009,x:PCTL ;configura o clock para 40MHz no PLL
        include 'ada_init.asm'
TONE_OUTPUT     EQU
HEADPHONE_EN+LINEOUT_EN+(4*LEFT_ATTN)+(4*RIGHT_ATTN)
TONE_INPUT      EQU     MIC_IN_SELECT+(15*MONITOR_ATTN)
;*************************************************************************
;Leitura do ADS7864
;*************************************************************************
        ;configura porta B para I/O
movep   #$0F00,x:PBDDR  ;seta PB para pinos I/O,PB0 a PB7 inicialmente
saida(display)
movep   #$0700,x:PBD    ;desab. disp, A/D e latch
;*****************************************************************
; o contador interno decrementa a uma taxa de fclock/2, fclock=40MHz
; portanto decrementa a cada 1/20MHz (50 ns). 125e-6/50e-9 = 2500
; gerando uma interrupção a cada 125e-6 s
; TCSR  ...  5   4   3   2   1  0
```

```
;          TC2 TC1 TC0 INV TIE TE
;           0    0     0     0    1   1
;*****************************************************************
        movep   #$000202,x:TCSR ;habilita interrupçao no timer
        movep   #$023000,x:IPR    ;TIL1=TIL1=1 prioridade nivel 2 timer, mesmo
nivel SSI p/
                                  ;evitar que SSI interrompa o timer
                                    ;SSL0=SSL1=1 prioridade nivel 2 SSI
        movep   #2500,x:TCR
        MOVEP   #10,X:TCR
        andi    #$CC,MR         ;MR faz parte do Status Register(SR) os
                                ;dois bits menos significativos vao para 1
                                ;no reset, entao e necessario zerar pela
                                ;operacao E para desmascarar as interrup.
        bset    #TE,x:TCSR      ;liga o timer
        jmp     *               ;fica aguardando as interrupcoes
;*************************************************************************
;Leitura do ADS7864
;*************************************************************************
loop_1
        bclr    #HOLD,x:<<PBD   ;amostra e retem sinal no AD
        nop                     ;500 ns (30 ns min)
        bset    #HOLD,x:<<PBD
        rep     #400
        nop     ;20 us
        jclr    #BUSY,x:<<PBD,pula
        jset    #BUSY,x:<<PBD,loop_1
        bclr    #CS,x:<<PBD
        ;*************************************************************************
*
        ;Leitura do A/D, ao final a informacao de 16 bits do A/D fica disponivel na
memoria,
        ;ja com as partes L e H apendadas de cada canal .
        ;*************************************************************************
*
        ;  va  |  vb  |  vc  |  ia  |  ib  |  ic
        ; CHA0   CHA1   CHB0   CHB1   CHC0   CHC1
        ;A/D configurado para modo de leitura ciclico, sequencia de
        ;leitura: A0L->A0H->A1L->A1H->B0L->B0H->B1L->B1H->C0L->C0H-
>C1L->C1H
        move    #va,r1                  ;carrega endereco da fase a
le_dado                         ;le A0 e A1
        bclr    #RD_RS,x:<<PBD
        nop                             ;500 ns (CS e RD necessitam estar em low
juntos
                                ;por pelo menos 30 ns para a saida de 12 bits sair
                                ;de alta impedancia e apresentar os dados)
```

```
        movep   x:PBD,a             ;le a parte LOW do A/D, MV e para a e nao
a1, logo a0=a2=0
        move    x:masc,x0
        and     x0,a1           ;x0=0000FF, zera a partir do nono bit de a1
        bset    #RD_RS,x:<<PBD
        nop                 ;500 ns (RD necessita ficar em high por pelo menos 30
ns)
        bclr    #RD_RS,x:<<PBD
        nop
        movep   x:PBD,b         ;le a parte HIGH do A/D
        nop
        bset    #RD_RS,x:<<PBD
;####rotina para verificar qual canal foi lido
verifica
        move    b1,y0           ;salva b1
        move    #$F0,b1         ;mascara p/ deixar apenas endereco
        and     y0,b1           ;x0=0000F0, zera a partir do nono bit de b1
        move    b1,y1           ;salva b1 apos mascara
        move    #$80,b1         ;end de A0 + BIT VALIDO
        eor     y1,b1           ;verifica se end do canal A0
        jeq     segueA0         ;se for end do canal A0 vai carregar n1
        move    #$90,b1
        eor     y1,b1
        jeq     segueA1
        move    #$A0,b1
        eor     y1,b1
        jeq     segueB0
        move    #$B0,b1
        eor     y1,b1
        jeq     segueB1
        move    #$C0,b1
        eor     y1,b1
        jeq     segueC0
        move    #$D0,b1
        eor     y1,b1
        jeq     segueC1
        bclr    #RD_RS,x:<<PBD  ;le novamente se nenhum end corresponde
        nop                 ;isto significa que foi lida a parte baixa
        movep   x:PBD,b         ;(ou dado invalido), entao guardar a parte
        nop                 ;baixa para ser apendada com a proxima
        bset    #RD_RS,x:<<PBD  ;parte(alta)
        move    y0,a1           ;coloca parte baixa em a
        move    x:masc,x0
        and     x0,a1           ;x0=0000FF, zera a partir do nono bit de a1
        jmp     verifica
segueA0 move        #0,n1
        jmp             segue
segueA1 move  #1,n1
```

```
        jmp             segue
segueB0 move        #2,n1
        jmp             segue
segueB1 move        #3,n1
        jmp             segue
segueC0 move        #4,n1
        jmp             segue
segueC1 move        #5,n1
segue   move        y0,b1           ;retorna valor de b1 que foi salvo
;#####################################################
        rep     #8
        lsl     b1              ;lsl ja coloca zero no LSB de b1, quando b0=0
        move    b1,x1
        or      x1,a1           ;apenda b1 em a1
        ;*************************************************************
        ;valores aproximados na saida do A/D de 12 bits:
        ; -2,3 V -> 800h
        ; -1   V -> B40h
        ;  1,4 V -> 160h
        ;  3   V -> 570h
        ;o bit MSB (sinal) e sempre 0 (positivo) para
        ;tensoes positivas. Sendo assim é necessário deslocar 12 bits para
        ;que o bit de sinal fique na posicao correta (24).
        ;*************************************************************
        rep     #12             ;ajusta de 12 bits(A/D) p/ 24 bits(DSP)
        asl     a
        move    a1,y0
        CONVMN  2.80127389      ;multiplica antes da matriz para garantir
0,9142857
                                ;como valor maximo, pois para a tensao de 227Vrms
                                ;(320 Vp) o sensor entrega 1,667 Vp que dividido por
5.344
                                ;e igual a 0.311862(com AMP OP ajustado. Este
valor colocado
                                ;na memoria do DSP,que multiplicado por 2,9315657
= ,914285
                                ;estava saturando entao foi colocado 2.8649392
        move    a1,x1
        mpy             x1,y0,a         ;MN*FRAC=MN
        rep     #7              ;converte de MN para frac
        asl     a
        move            a1,x:(r1+n1)            ;guarda a informacao high+low (16 bits)
        move            y:contcanais,a0 ;contador de canais ja armazenados em a
        inc     a               ;inc contcanais
        move            #6,b0
        move    b0,y0
        move            a0,a1
        eor     y0,a1           ;verifica se canais armazenados=6
```

```
        jeq     fim_cont                ;se dif de 6 continua leitura se =6 interrompe
leitura
        move    a0,y:contcanais  ;salva novo valor do contador
        jmp     le_dado
fim_cont
        move    #$0,a1
        move    a1,y:contcanais
        bset    #CS,x:<<PBD     ;desabilita A/D
;###### Blocos 7 e 8, Calcula a matriz ########
;ao final da rotina vqss, vdss, iqss e idss estao diponiveis na memoria
;*********************************************************************
bloco7
;       |qss| = |2/3   -1/3       -1/3   |
;       |dss|   |0   -1/srt(3)  1/sqrt(3)|
;       vqss = a11*va + a12*vb + a13*vc
;       vdss = a21*va + a22*vb + a23*vc
;###### Tensao #########
        move            #a11,r1
        move            #vqss,r2
        do      #2,tensao
        move            #va,r0          ;uso local de r0, r1 e r2 va(frac)
        nop
        move    x:(r0)+,x0
        move    x:(r1)+,x1
        mpy             x0,x1,a         ;frac*frac=frac
        do      #2,v
        move            x:(r0)+,x0
        move            x:(r1)+,x1
        mac     x0,x1,a         ;frac*frac=frac
v
        move    a1,y:(r2)+      ;1 vez guarda vqss, 2 vez vdss r2 aponta p/ iqss
tensao
;###### Corrente #######
        move            #a11,r1
        do      #2,corr
        move            #ia,r0          ;uso local de r0 e r1
        nop
        move    x:(r0)+,x0
        move    x:(r1)+,x1
        mpy             x0,x1,a         ;frac*frac=frac
        do      #2,i
        move    x:(r0)+,x0
        move            x:(r1)+,x1
        mac             x0,x1,a         ;frac*frac=frac
i
        move    a1,y:(r2)+      ;1 vez guarda iqss, 2 vez guarda idss
corr
;###### Blocos 5 e 6 #############
```

```
        move    #vqssl,r0       ;seta ponteiro para guardar resultado de vqssl
        move    #iqss,r1
        move    #vqss,r2
bloco5  do              #$2,calcula_vss ;na primeira vez guarda resultado em
vqssl, na segunda
                                ;guarda em vdssl,porque r0 e incrementado dentro
da rotina
q
        move y:(r1),y1          ;Divide iqss e idss por 3.22564 (multiplica por
0.310016) apos usar
        move #0.310016,x1  ;na rotina dos Blocos 5 e 6, pois Bloco 2 precisa da
divisao
                        ;o processo de conversão divide por 108,506,
(108.506*3.22564=350)
        mpy x1,y1,a
        move a1,y:(r1)+         ;guarda valor já dividido na memoria (frac)
        move    y:(r2)+,b       ;carrega vss em b
        jsr     vss             ; calcula bloco 5 e 6 ( usa R0 ) (tudo frac)
calcula_vss
;###################################
;###### Blocos 3 e 4 #############
        move    #somaqss,r2             ;ponteiro aponta p/ somaqss do ultimo
calculo(1/z)
        move    #vqssl,r0               ;vqssl e vdssl serao usados para calc yqss r
ydss
bloco3  do              #$2,calcula_yss
        move    #w11,r3                 ;r3, r4 e r5 sao reinicializados dentro do
'do'
        move    #produt1,r4             ;porque sao usados para calcular os
blocos 3 e 4
        move    #soma,r5
        jsr     yss                     ;o resultado e um MN
calcula_yss
;###################################
        move    #yqss,r1
        move    #sint,r0        ;pont p/ sin e cos(theta-e)
;###### Bloco 2.1 ################
bloco21 move    y:(r1),y0       ;yqss em y0 (MN)
        move    y:(r1),x0       ;yqss em x0 (MN)
        move    #ydss,r1        ;TENTAR MOVER DIRETAMENTE P/ a P/
ECONOMIZAR UMA INSTRUCAO
        mpy     y0,x0,a         ;a=yqss*yqss (MN*MN)
        rep     #14             ;restaura o result para MN (7*asl) e transforma em
frac (+7)
        asl     a
        move    a1,x1                   ;x1=square1
        move    y:(r1),y1               ;ydss em y1
        move    y:(r1),x0               ;ydss em x0
```

```
        mpy         y1,x0,a     ;a=ydss*ydss(square2) (MN*MN)
        rep     #14             ;restaura o result para MN (7*asl) e
transforma em frac (+7)
        asl     a
        add         x1,a            ;obtem a=sum5 (frac)
        move        #pcoef,r2       ;ponteiro p/ coef do polinomio
        move        a1,x0       ;guarda a em x0
        sqrt                ;obtem raiz quadrada de sum5 (frac)
fracao  move    #$010000,y1     ;shift constante em y1, esta constante
                            ;multiplicada por um frac gera um Mixed N.
        mpyr    x0,y1,a #100,x1 ;transforma sum5 em MN e guarda em a
        move    a1,x0           ;coloca sum5 MN em x0
        mpy     x0,x1,a         ;mutipl sum5 MN por 100d (MN*MN -> precisa
de ajuste)
        rep     #7          ;ajusta acc para MN apos multiplicacao de MN*MN
        asl     a
        move    a1,x0           ;raiz quadrada * 100 (MN)
        move    #1,a        ;1d em a (dividendo)
divisao abs a a,b
        eor x0,b b,x:$0         ;ou-exclusivo(bits 47~24) save rem. sign in
x:$0, quo, sign in N
        and #$fe,ccr
        rep #$18
        div x0,a
        tfr a,b
        jpl savequo
        neg b
savequo tfr x0,b b0,x0
        abs b
        add a,b
        jclr #23,x:$0,done
        move #$0,b0
        neg b
done                        ;no final é obtido um fracionario(frac) em x0
        mpy     x0,x1,a         ;mutipl frac. por 100d MN (frac*MN=MN)
        move    #yqss,r1
        move    a1,x0           ;coloca resultado da mult. (MN) em x0 (x0
esta mult por 100)
calcsin move    y:(r1),y0       ;yqss em y0 (MN)
        mpy     y0,x0,a         ;obtem product (sin) (MN*MN-> precisa de
ajuste)
        rep     #14         ;ajusta acc. para MN apos mult MN*MN(7)
        asl     a
        move    #ydss,r1
        move    a1,y:(r0)+      ;a=product [sin(theta-e)] e armazena sin (frac)
calccos move    y:(r1),y0       ;ydss em y0 (MN)
        mpy         y0,x0,a     ;obtem product3 (cos) (MN*MN-> precisa de
ajuste)
```

```
        rep     #14             ;ajusta acc. para MN apos mult MN*MN(7)
        asl     a
        move            a1,y:(r0)-              ;a=product3 retorna ponteiro p/ sin, pois
sin
                                ; sera usado primeiro na rotina seguinte
;###################################
;###### Blocos 2.2 e 2.3 ########
;x0=sin(wet), x1=ydss, y0=yqss, y1=cos(wet),
;           x1=idss  y0=iqss
bloco22 move    #ydss,r1
        move    y:(r0)+,x0              ;carrega sin(wet) em x0 e inc r0 p/ cos (frac)
        move    y:(r1),x1               ;carrega ydss em x1 (MN)
        move    #yqss,r1
        mpy             x0,x1,a         ;obtem produt1 (frac*MN=MN)
        move    y:(r0)-,y1              ;carrega cos(wet) em y1 e dec r0 p/ sin (frac)
        move    y:(r1),y0               ;y0=yqss (MN)
        move    #yqs,r7                 ;r7 aponta p/ yqs
        mpy             y1,y0,b         ;obtem product (frac*MN=MN)
        sub             a,b             ;(MN)
        move    b1,y:(r7)+              ;obtem yqs e guarda
        mpy     y0,x0,a         ;obtem product2 (yqss*sin) (MN)
        mpy             x1,y1,b         ;obtem product3 (ydss*cos) (MN)
        move    #idss,r1
        add             b,a             ;(MN)
        move    a1,y:(r7)+              ;obtem yds e guarda
bloco23 move            y:(r1)-,x1              ;x1=idss (frac)
        mpy             x0,x1,a         ;obtem product1 (sin*idss) (frac)
        move            y:(r1),y0       ;y0=iqss
        mpy             y0,y1,b         ;obtem product (iqss*cos)
        sub             a,b             ;b-a e armazena em b (obtem iqs)
        move            b1,y:(r7)+              ;obtem iqs e guarda (frac)
        mpy             y0,x0,a         ;obtem product2 (iqss*sin)
        mpy             x1,y1,b         ;obtem product3 (idss*cos)
        add             b,a
        move            a1,y:(r7)       ;obtem ids e guarda (frac)
;###################################
;###### Bloco 2.4 ################
bloco24 move    #yqs,r7
        nop
        move    y:(r7)+,x0              ;x0=yqs (MN)
        move    y:(r7)+,x1              ;x1=yds (MN)
        move    y:(r7)+,y0              ;y0=iqs (frac)
        move    y:(r7),y1               ;y1=ids (frac)
        mpy             x1,y0,a         ;product (yds*iqs) (MN*frac=MN)
        mpy             x0,y1,b         ;product1 (yqs*ids) (MN*frac=MN)
        sub             b,a     a1,y0           ;obtem Sum e coloca em y0 (MN)
        CONVMN  3                       ;macro p/ converter p/ MN (3P/4 P num de
polos)
```

```
        move    a1,x0                   ;macro devolve resultado em a
        move            #yds,r7
        nop
        mpy             x0,y0,a         ;(Sum * 3P/4) obtem Te (MN*MN-> precisa de
ajuste)
        rep     #7
        asl     a               ;ajusta p/ MN novamente
        rep     #7              ;multiplica produt4 por 256
        asl     a
        move            a1,y0
        CONVMN  1.3671875
        move            a1,y1
        mpy             y0,y1,a         ;resultado = TE*350 (MN)
        rep     #7
        asl     a
        move    a1,y:TORQUE
        ;******* rotina para enviar informacao para o latch******
        move    #$15,a1
        move    a1,y:inf
        jsr     vaidisplay
        bset    #EN_LATCH,x:<<PBD
        nop
        bclr    #EN_LATCH,x:<<PBD
        movep   #$0F00,x:PBDDR  ;00 -> entrada
        MOVE    x:(vb),b
        MOVEX:(va),a
        jset    #2,x:SSISR,*    ;wait for frame sync to pass
        jclr    #2,x:SSISR,*    ;wait for frame sync
        move    a,x:TX_BUFF_BASE        ;transmit left
        move    b,x:TX_BUFF_BASE+1      ;transmit right
        move    #TONE_OUTPUT,y0 ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3
        rti
;###################################
;+++++++++++++++++++++++++++++++++
vss     move            a1,x0           ; x0 componente de corrente de entrada,
x0=iss (frac)
        move    b,a             ; componente de tensao carregado em a, a=vss
(frac)
        move    x:(Rs),y0               ; y0 usado p/ parametros do motor, y0=-Rs
        mpy             x0,y0,b         ; iss*(-Rs)
        rep     #2              ; multiplica iss*Rs por 4 pois Rs estava dividido por 4
        asl     b
        add             b,a             ; Vss-iss*Rs*4
        move            a1,y:(r0)+              ; guarda rsultado de vss e inc r0
```

```
                        ; na primeira vez do 'do'r0->vqss e na segunda r0-
>vdss
        rts                         ; retorna de vss
;+++++++++++++++++++++++++++++++
;+++++++++++++++++++++++++++++++
yss     move        x:(r3)+,x0          ;carrega w11 em x0 e incrementa r3
(frac)
        move    y:(r2)+,y0          ;carrega somass(1/z) em yo (MN)
                        ;sera usado no proximo produto
        mpy         x0,y0,a             ;obtem product1 (frac*MN=MN)
        move    a1,y:(r4)+          ;guarda a em produt1 e inc r4 para produt2
        move    x:(r3)+,x0          ;carrega w22 em x0 e inc r3 p/ w21 (frac)
        mpy         x0,y0,a             ;obtem product2 (frac*MN=MN)
        move    a1,y:(r4)+          ;guarda a em produt2 e inc r4 para produt3
        move    x:(r3)+,x0          ;carrega w21 em x0 e inc r3 p/ w13 (frac)
        move    y:(r2)-,y0          ;carrega yss(1/z) em y0 e dec r2 para guardar
proximo (MN)
                        ;valor de somass(1/z)
        mpy         x0,y0,a             ;obtem product3 (frac*MN=MN)
        move    a1,y:(r4)+          ;guarda a em produt3 e inc r4 p/ produt4
        move    x:(r3),x0           ;carrega w13 em x0 e nao inc mais(ultimoW)
(frac)
        move    y:(r0)+,y0          ;carrega vssl em y0 e inc r0 para pegar
proximo vssl (frac)
        mpy     x0,y0,a             ;obtem produt4 (frac*frac=frac)
        rep     #8              ;multiplica produt4 por 256
        asl     a               ;(frac)
        move    a1,y0
        CONVMN  1.3671875       ;1.3671875*256=350
        move    a1,y1               ;(MN)
        mpy         y0,y1,a         ;resultado = produt4*350 (frac*MN=MN)
        move    a1,y:(r4)           ;guarda a em product4 e nao inc mais (ultimo)
        move    a1,x0
        move    #produt1,r4
        nop
        move    y:(r4)+,a           ;product1 em a
        add         x0,a            ;MN+MN=MN
        move    a1,y:(r2)+          ;guarda soma em somass(1/z) e inc r2 p/
yss(1/z)
        move        y:(r4)+,x0          ;produt2 em x0
        move        y:(r4),a        ;guarda produt3 em a
        add         x0,a            ;(MN+MN=MN)
        move    a1,y:(r2)+          ;guarda soma1 em yss(1/z) e inc r2
                                ;p/ calc do bloco 4
        rts         ; retorna de yss
;+++++++++++++++++++++++++++++++
mili2
        rep     #4000           ; 2 ms delay (4+mv oscillator clock cycles)
```

```
        nop                 ; 2 oscillator clock cycles
        rep    #4000
        nop
        rts
;++++++++++++++++++++++++++++++++++
delay1ms
        rep    #2000
        nop
        rts
delay5ms
        do     #5,_fim
        rep    #2000
        nop
_fim
        rts
delay1s
        do     #1000,_fim1
        rep    #4000
        nop
_fim1
        rts
        end
```

12.7 Rotina em DELPHI para Mostrar o Conjugado no PC

```
unit Mestrado;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  TeEngine, Series, TeeProcs, Chart, StdCtrls, Buttons, ExtCtrls, IOport ;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    Chart1: TChart;
    Series1: TFastLineSeries;
    IOport1: TIOport;
    Edit1: TEdit;
    BitBtn3: TBitBtn;
    Timer1: TTimer;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;
var
  Form1: TForm1;
  y,x : Real;
  yaux, xaux : Integer;
  hab : Boolean;
implementation
{$R *.DFM}
        {x:=0;
        while x<= 480000 do
          begin
          inc(x);
          IOport1.PortAddress:=$378;
          IOport1.Read;
          mat[x]:=IOport1.PortData;
          end;
        z:=0;
        while z<= 480000 do
          begin
          inc(z);
          q:=q+0.00001
          y:=mat[z];
          Chart1.Series[0].AddXY(q,y,'',ClTeeColor); }
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
x:=0;
IOport1.PortAddress:=$303;
IOport1.PortData := $82;
IOport1.Write;
Series1.Clear;
while x <=10 do
    begin
        x := x + 0.00001;
        IOport1.PortAddress:=$301;   {define endereço de leitura}
        IOport1.Read;   {lê o dado}
        yaux:=IOport1.PortData; {armazena dado em uma variável}
        if (yaux>=128) then {faz o Complemento 2 do valor e multiplica por -1}
        Begin
        yaux := yaux xor 255;
        yaux := yaux + 1;
        yaux := yaux * (-1);
        End;
        y := yaux/5;
        Chart1.Series[0].AddXY(x,y,'',ClTeeColor);
        Edit1.Text:=FloatToStr(y);
    end;
end;
procedure TForm1.BitBtn3Click(Sender: TObject);
begin
Close;
end;
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
Series1.Clear;
end;
end.
```