A FEDERAL DE ENGENHARIA DE ITAJUBÁ

IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE MULTIPROCESSADO PARA CONVERSORES DE FREQUÊNCIA UTILIZANDO O ALGORITMO ESCALAR

ANGELO JOSÉ RONCALI DA SILVA

SETEMBRO - 1996

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE MULTIPROCESSADO PARA CONVERSORES DE FREQÜÊNCIA UTILIZANDO O *ALGORITMO ESCALAR*

por

ANGELO JOSÉ RONCALI DA SILVA
INSTITUTO DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ELETRÔNICA

DISSERTAÇÃO APRESENTADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA

Setembro 1996

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE MULTIPROCESSADO PARA CONVERSORES DE FREQÜÊNCIA UTILIZANDO O *ALGORITMO ESCALAR*

por

ANGELO JOSÉ RONCALI DA SILVA

Orientador: FELÍCIO BARBOSA MONTEIRO

Co-orientador: ÊNIO ROBERTO RIBEIRO

INSTITUTO DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ELETRÔNICA

DISSERTAÇÃO APRESENTADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA

Setembro 1996

A meus pais, pela compreensão, carinho
e dedicação durante toda a minha vida.

RESUMO

O presente trabalho descreve um sistema de controle, cujo objetivo é a operação, em tempo real, de conversores diretos de freqüência e utiliza como método de controle o *Algoritmo Escalar*.

Este sistema constitui-se de um controlador multiprocessado, conectado a um computador pessoal, e dos circuitos de interface e condicionamento de sinais.

O controlador multiprocessado, formado por um microcontrolador de 16 bits e quatro microcontroladores de 8 bits, divide a tarefa de controle dos interruptores do conversor entre mais de um processador, possibilitando a sua operação em tempo real.

Diferente de outros métodos de controle para conversores diretos, que tratam vetorialmente as variáveis envolvidas, o *Algoritmo Escalar* baseia-se na comparação escalar dos valores instantâneos das tensões de entrada e da tensão de saída desejada. Ele gera equações simples e concisas para o cálculo dos tempos de acionamento dos interruptores do conversor.

Estas características resultam numa simplificação, a nível de programação, deste sistema de controle em comparação aos sistemas usuais.

Este trabalho apresenta, primeiramente, a teoria do *Algoritmo Escalar*, sendo a seguir detalhado o *hardware* utilizado pelo sistema de controle, concentrando-se nas características do controlador multiprocessado.

Na seqüência descreve-se o programa desenvolvido para a realização do controle dos interruptores do conversor. São mostradas as tabelas utilizadas para a normalização

dos valores de tensão e tempo manipulados pelo programa, as etapas em que o programa se divide e como é realizado o controle dos parâmetros do sistema (a freqüência e a amplitude das tensões de saída do conversor).

Finalizando, afim de comprovar a validade do sistema de controle proposto, resultados experimentais são apresentados.

ABSTRACT

This work describes a real time direct frequency converter control system which uses the *Scalar Algorithm* control method. The system consists of a multiprocessed controller connected to a personal computer, and signal conditioning and interface circuits.

The multiprocessed controller is formed by a 16 bits microcontroller and four microcontroller of 8 bits. The microcontroller divides the task of controlling the converter switches among the processors, which allows a real time operation.

Unlike the vector treatment on the systems variables of other direct frequency converter methods, the *Scalar Algorithm* is based on the instantaneous values of the input and output voltages. It provides simple equations to compute the converter switches activation times. Those characteristics simplifies the programming of this method compared to the others.

This work introduces, initially, the *Scalar Algorithm*. Then, it is presented the hardware required by the control system, emphasizing the characteristics of the multiprocessed controller.

It is presented the program developed to implement the control of the converter switches. Is also shown the tables used to normalize the voltage values and the timing handled by the program, the sections of the program, and the form that the parameters of the system are controlled (frequency and amplitude of the converter output voltages).

Finally, experimental results are shown to demonstrate the operation of the proposed control system.

AGRADECIMENTOS

A Deus dou graças neste momento de vitória, por mais esta etapa concluída.

Ao orientador, Prof. Felício Barbosa Monteiro, e ao co-orientador, Prof. Ênio Roberto Ribeiro, pela proposição do tema desenvolvido e pela disposição e boa vontade, sempre quando foram solicitados.

A George-Emile April, pelo envio de componentes para o controlador multiprocessado.

A Eric Leonard, pela ajuda na compreensão do funcionamento do controlador multiprocessado.

Ao Eng. Luiz Juberto Rossi, pela inestimável ajuda na gravação de componentes programáveis.

Aos professores e funcionários do Departamento de Engenharia Eletrônica da EFEI, pela colaboração e solidariedade a mim dedicada.

A CAPES, pelo suporte financeiro.

SUMÁRIO

PÁGINA

LISTA DE APÊNDICES

PÁGINA

LISTA DE FIGURAS

PÁGINA

LISTA DE TABELAS

PÁGINA

LISTA DE NOTAÇÕES E SÍMBOLOS

v_A, v_B, v_C	Valores instantâneos das tensões de fase na entrada do conversor.
V_i	Valor de pico das tensões de entrada.
ω_i	Freqüência angular das tensões de entrada.
v_o	Valor instantâneo das tensões na saída do conversor.
T_S	Período de comutação dos interruptores e ciclo de operação do programa.
f_S	Freqüência de comutação dos interruptores.
v_K, v_L, v_M	Valores instantâneos das tensões de fase na entrada do conversor.
t_K, t_L, t_M	Tempos de acionamento dos interruptores compondo um ramo de saída do conversor.
ρ_{KL}	Taxa instantânea entre duas tensões de entrada.
V_o	Tensão de pico das tensões na saída do conversor.
q	Ganho de tensão do conversor.
ω_o	Freqüência angular das tensões de saída.
Kbytes	2^{10} bytes.
Mbytes	2^{20} bytes.
h	Utilizado como sufixo, indica um número hexadecimal.
RL	Registro de *latch* dos contadores.
RC	Registro do contador.
INT	Pino de entrada dos coprocessadores.
EV	Pino de saída dos coprocessadores.

CAPÍTULO 1

INTRODUÇÃO

1.1 Considerações gerais

Os conversores de freqüência têm sido, nos últimos anos, em razão de sua vasta gama de aplicações, objeto de estudos no campo da eletrônica de potência. Entre estas destacam-se o controle da velocidade de motores e a geração ou compensação estática de potência reativa. Diversas estruturas, para os mais variados tipos de conversores, têm sido propostas [1]. Além disso, os recentes desenvolvimentos dos semicondutores de potência promoveram um grande impacto sobre a tecnologia e equipamentos de conversão de energia, em termos de tamanho, custo e desempenho.

Os conversores podem ser classificados como retificadores (conversor ca-cc), inversores (conversor cc-ca), recortadores (converdor cc-cc), controladores de potência ca (à mesma freqüência) e cicloconversores (conversores diretos de freqüência) [1]. Os conversores diretos de freqüência (DFC, *Direct Frequency Changer*) devem ter, idealmente, as seguintes características:

- forma de onda senoidal de tensão na saída;
- forma de onda senoidal de corrente na entrada;
- bidirecionalidade (o fluxo de energia pode fluir em ambas as direções através do conversor);
- freqüências de saída independentes da freqüência de entrada;

Os DFCs, compostos por chaves bidirecionais, são de estágio único e requerem o mínimo de componentes passivos. Estes conversores também são conhecidos por conversores em matriz (*matrix converter*). Isto porque os nove interruptores deste conversor são arranjados em uma "matriz"; qualquer tensão da entrada trifásica pode ser conectada a um dos ramos de saída da matriz, como mostra a figura 1.1.

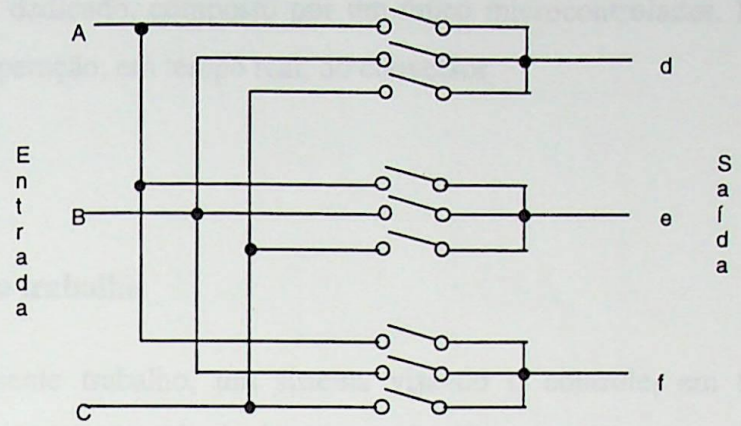


Fig. 1. 1 Conversor em matriz.

A topologia em matriz foi primeiramente investigada em 1976 [2]. Trabalhos posteriores objetivaram a implementação e o controle desta topologia. Venturini e Alesina, [3] - [5], descreveram um novo conversor e uma nova técnica de conversão baseada na topologia em matriz. Uma característica deste conversor é a ausência de um *link dc*, e assim, nenhuma energia é armazenada em capacitores ou indutores. Entretanto, tal técnica de conversão não encontrou larga utilização devido a problemas na implementação [9]. Estes problemas relacionavam-se à confecção, à sincronização e à proteção das chaves bidirecionais. Observou-se, também, a seguinte restrição teórica:

- limitação da máxima amplitude da tensão de saída em relação a amplitude da tensão de entrada, i. e. ganho de tensão, em 50%.

Para contornar os problemas encontrados, diferentes métodos e estratégias de controle foram propostos e estudados em [6] - [12]. Verificou-se, inclusive, a possibilidade da utilização do conversor em matriz como retificador [13]. Todos estes métodos podem ser classificados como métodos de controle vetoriais pois tratam as variáveis envolvidas no processo de forma vetorial [14].

Diferentemente destes métodos, Roy, Duguay, Manias e April [14], propuseram uma técnica singular, o *Algoritmo Escalar* (AE). Este baseia-se somente na comparação escalar dos valores instantâneos das tensões de entrada e da tensão desejada de saída. Com o AE atinge-se um ganho de tensão de tensão de 50%. Porém utilizando-se o AE em conjunto com a técnica de modulação proposta em [6] ou [9] o ganho de tensão é elevado para 87%. As características do AE foram, mais tarde, detalhadas por Roy e April [15]. Suas qualidades operacionais foram verificadas através de sua implementação

em um sistema dedicado, composto por um único microcontrolador. Este sistema não permitiu uma operação, em tempo real, do conversor.

1.2 Objetivo do trabalho

No presente trabalho, um sistema visando o controle, em tempo real, dos conversores diretos de freqüência é apresentado. Este sistema tem como características principais:

- utilização, como método de controle, do *Algoritmo Escalar*;
- implementação do método de controle em um controlador multiprocessado.

Através do AE são geradas equações simples e concisas para o cálculo dos tempos de acionamento dos interruptores que compõem um conversor. Estas equações trabalham apenas com os valores escalares das tensões. O controlador multiprocessado, constituído por microcontroladores, divide a tarefa de controle do conversor entre mais de um processador. O controle dos três interruptores, interligando as três tensões de entrada a um mesmo ramo de saída, é realizado por um único processador, como apresentado na figura 1.2. Nesta figura, *C1* representa os circuitos de aquisição dos valores das tensões de entrada. Estas características possibilitaram o controle, praticamente instantâneo, da freqüência e da amplitude das tensões de saída.

Sendo o primeiro trabalho desenvolvido na área de conversores diretos pela Escola Federal de Engenharia de Itajubá, este se propõe a ser um ponto de partida para futuras pesquisas. Assim o objetivo concentra-se na elaboração do sistema de controle em si, possibilitando o seu uso por outros trabalhos, pois este sistema independe da potência do conversor direto utilizado.

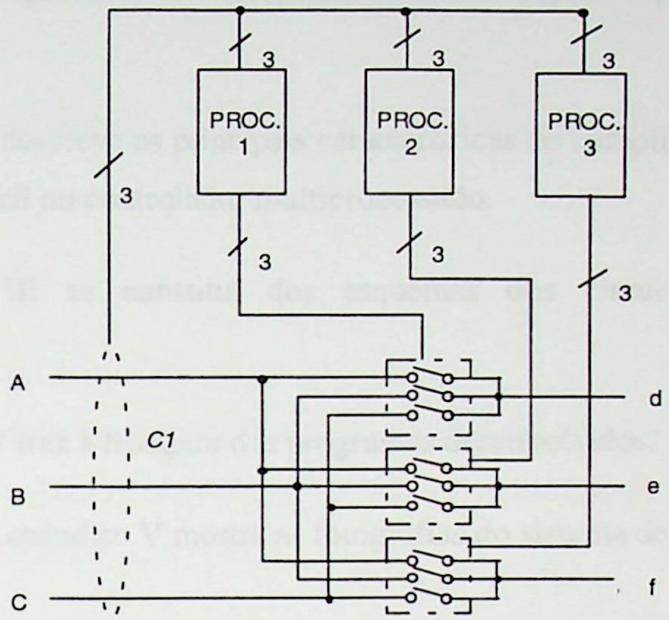


Fig. 1. 2 Sistema de controle multiprocessado para conversores diretos de freqüência.

1.3 Estrutura da dissertação

A dissertação está estruturada em seis capítulos.

O capítulo 2 descreve o método utilizado para a obtenção do controle, em tempo real, de conversores diretos de freqüência, o *Algoritmo Escalar (AE).*

O capítulo 3 expõe sobre o *hardware* utilizado na implementação do AE. O *hardware* se compõe do controlador multiprocessado e dos circuitos de interface e condicionamento dos sinais necessários à esta implementação.

O capítulo 4 detalha os programas desenvolvidos para a obtenção do controle, em tempo real, dos conversores diretos de freqüência.

O capítulo 5 mostra os resultados conseguidos pela implementação do AE em um controlador multiprocessado.

O capítulo 6 apresenta as conclusões e sugestões para trabalhos futuros.

Encerrando a dissertação, temos cinco apêndices.

O apêndice I apresenta um complemento sobre alguns aspectos abordados na dissertação.

O apêndice II descreve as principais características do compilador *Forth* residente em memória não volátil no controlador multiprocessado.

O apêndice III se constitui dos esquemas dos circuitos de interface e condicionamento.

O apêndice IV traz a listagem dos programas desenvolvidos.

Finalmente, o apêndice V mostra as fotografias do sistema de controle.

CAPÍTULO 2

O MÉTODO DE CONTROLE: *ALGORITMO ESCALAR*

2.1 Amostragem Rápida

Considere o conjunto dos sinais de tensão apresentados na figura 2.1, onde v_A, v_B e v_C designam os sinais na entrada e v_0 o sinal de referência para um ramo de saída do conversor direto de freqüência (*DFC*).

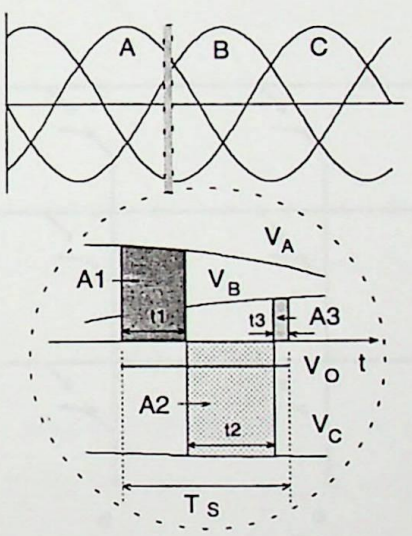


Fig. 2. 1 Amostragem Rápida de um sistema trifásico.

Toma-se um intervalo de tempo T_S, suficientemente pequeno, durante o qual os sinais de entrada serão considerados constantes. Dentro do intervalo T_S amostra-se o sinal v_A durante o período de tempo t_1, o sinal v_C durante t_2 e o sinal v_B durante t_3, com $t_1 + t_2 + t_3 = T_S$. Dos produtos amplitude-tempo resultam as áreas A1, A2 e A3, respectivamente. Da integração destas últimas resulta o sinal v_0, com um valor que poderá ser positivo, negativo ou mesmo nulo, dependendo da duração dos tempos t_1, t_2 e t_3. Logo, através da regulagem apropriada destes tempos, é possível a obtenção de um valor qualquer para o sinal de saída v_0. Evidentemente, os valores instantâneos de v_0 devem respeitar os limites superiores e inferiores dos sinais de entrada disponíveis

quando da amostragem. Esta operação de amostragem das tensões denomina-se *Amostragem Rápida* (AR).

2.2 O *Algoritmo Escalar*

Pelo processo de AR, quaisquer tensões de entrada podem servir para gerar uma tensão qualquer de saída. Isto permite estabelecer uma relação com a idéia apresentada pelo *Algoritmo Escalar* (AE). Este baseia-se na comparação escalar dos valores instantâneos das tensões de fase da entrada e da tensão de saída desejada, para a determinação dos tempos de acionamento dos interruptores. Estes interruptores conectam as tensões de entrada a um ramo de saída do conversor, gerando assim, a tensão de saída desejada (v_0).

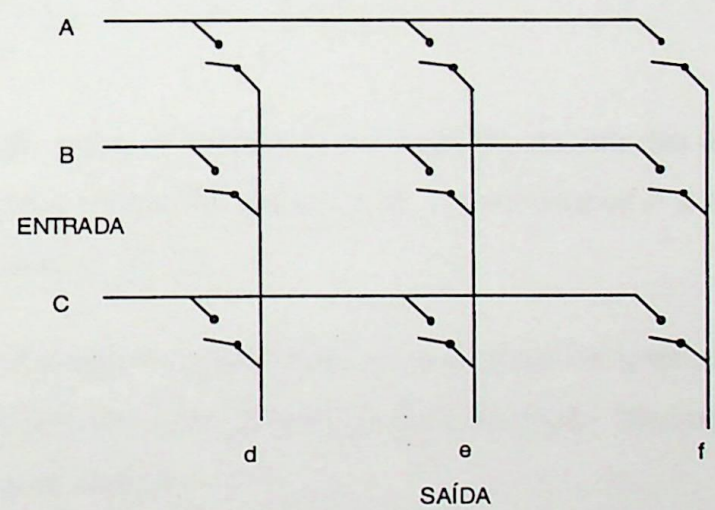


Fig. 2. 2 Matriz de interruptores do conversor direto de freqüência.

Admitindo-se, na entrada do conversor da figura 2.2, as seguintes tensões de fase:

$$v_A = V_i \cos(\omega_i t) \tag{1}$$

$$v_B = V_i \cos(\omega_i t - 2\pi/3) \tag{2}$$

$$v_C = V_i \cos(\omega_i t + 2\pi/3) \tag{3}$$

podemos obter, em cada ramo de saída, uma tensão v_0 cujo valor instantâneo pode ser expresso por:

$$v_0 = \frac{1}{T_S}[t_K v_k + t_L v_L + t_M v_M] \tag{4}$$

onde

$$t_K + t_L + t_M = T_S \tag{5}$$

sendo K, L e M índices variáveis, podendo serem associados às tensões de fase A, B ou C, de acordo com as seguintes regras:

1) A todo instante considerado, a tensão de fase de entrada que possui a polaridade oposta às outras, é designada como v_M.

2) As duas tensões de fase compartilhando a mesma polaridade são designadas como v_K e v_L. A menor, em valor absoluto, sendo designada como v_K. Então v_K e v_L são escolhidos tais que:

$$\frac{t_K}{t_L} = \frac{v_K}{v_L} \tag{6}$$

para o intervalo onde

$$0 \le \frac{v_K}{v_L} \le 1 \tag{7}$$

A equação (6) define os tempos de acionamento de dois dos três interruptores de um ramo de saída; esta relação de tempo (t_K/t_L) é proporcional a relação dos valores instantâneos das tensões (v_K/v_L).

A seguir é mostrado o procedimento para se obter os respectivos valores de t_K, t_L e t_M durante um período T_S da freqüência de comutação. Durante o intervalo onde $0 \le v_K/v_L \le 1$, define-se o termo:

$$\rho_{KL} = \frac{v_K}{v_L} \tag{8}$$

Deste modo os tempos t_K, t_L e t_M podem ser encontrados:

$$t_K = \rho_{KL} t_L \tag{9}$$

$$t_M = T_S - (\rho_{KL} + 1) t_L \tag{10}$$

$$t_L = \frac{T_S(v_0 - v_M)}{\rho_{KL} v_K + v_L - (1 + \rho_{KL}) v_M} \tag{11}$$

Usando novamente o valor de ρ_{KL} na expressão (11) resulta:

$$t_L = \frac{T_S(v_0 - v_M) v_L}{[v_K^2 + v_L^2 + v_M^2 - (v_K + v_L + v_M) v_M]} \tag{12}$$

Em um sistema trifásico equilibrado, a soma dos valores instantâneos das tensões de fase é nula. Logo, os tempos de acionamento dos interruptores podem ser reescritos da seguinte forma:

$$\frac{t_L}{T_S} = \frac{(v_0 - v_M)v_L}{v_K^2 + v_L^2 + v_M^2} = \frac{(v_0 - v_M)v_L}{1,5V_i^2} \tag{13}$$

$$\frac{t_K}{T_S} = \frac{(v_0 - v_M)v_K}{1,5V_i^2} \tag{14}$$

$$\frac{t_M}{T_S} = 1 - \frac{t_K + t_L}{T_S} \tag{15}$$

Os tempos t_K e t_L são proporcionais aos valores instantâneos de suas respectivas tensões de entrada v_K e v_L multiplicados pela diferença entre a tensão de saída desejada v_0 e a tensão de fase da entrada v_M. Deve ser observado que a tensão de saída v_0 pode ser qualquer tipo de forma de onda, inclusive contínua. O apêndice I mostra as deduções dos termos e equações aqui apresentados.

Resolvendo as equações (13) - (15) para um ganho de tensão, $V_0/V_i = q$, menor ou igual a 0,5, serão sempre produzidos valores positivos para os tempos t_K, t_L e t_M. Para ganhos de tensão maiores que 0,5 alguns valores de tempo negativos começarão a aparecer por causa da limitação das tensões instantâneas na entrada do conversor. Entretanto, trabalhos realizados por Maytum e Colman [6] e por Venturini e Alesina [9] permitiram elevar q para até 0,87.

CAPÍTULO 3

HARDWARE

O hardware divide-se em duas partes: o controlador multiprocessado e os circuitos de interface e condicionamento de sinais entre o controlador e o conversor.

3.1 Controlador multiprocessado

3.1.1 Generalidades

O *Algoritmo Escalar* foi implementado utilizando-se um controlador multiprocessado (figura 3.1), desenvolvido por Leonard [16].

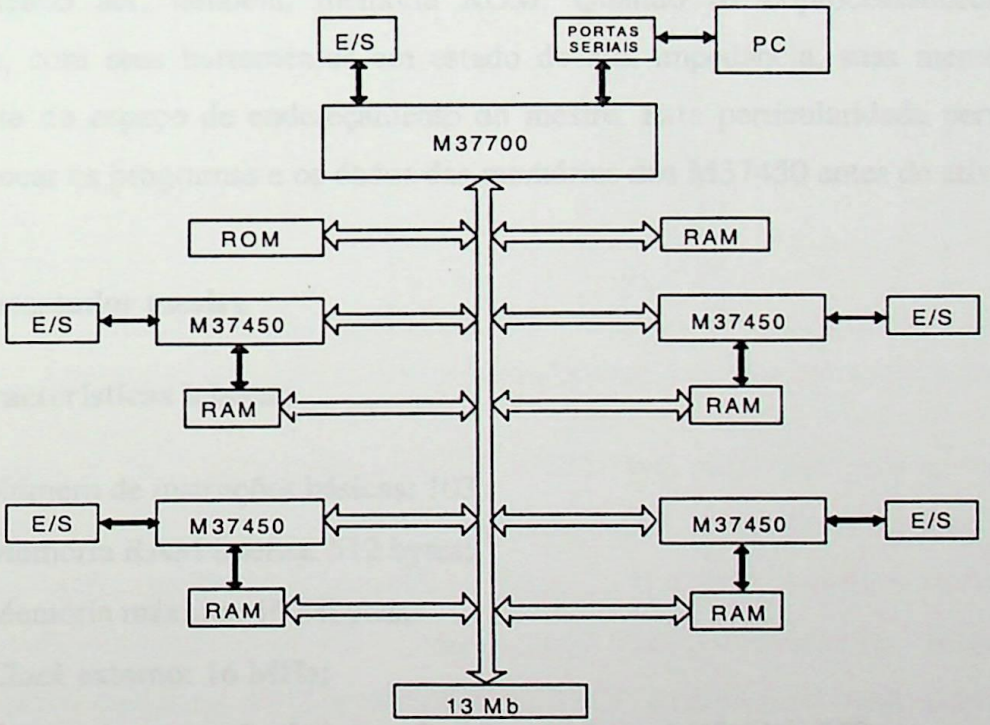


Fig. 3. 1 Controlador multiprocessado.

O controlador multiprocessado é composto por um microcontrolador de 16 bits (M37700) denominado mestre, e quatro microcontroladores de 8 bits (M37450) denominados coprocessadores, todos fabricados pela Mitsubishi Electric Corporation [17] - [18]. O controlador possui um compilador da linguagem *Forth* residente em

memória não volátil [19]. Logo os programas para os referidos processadores podem ser escritos tanto em linguagem de alto nível (*Forth*) como em linguagem de baixo nível (*Assembly*) e testados nas memória de acesso aleatório (RAM) do próprio controlador.

Uma configuração do tipo estrela conecta os microcontroladores. A comunicação entre o mestre e os coprocessadores se faz paralelamente. Para o mestre os coprocessadores são vistos como endereços de memória. Entretanto, estes estão conectados ao mestre por uma interface de barramento própria para uso em sistemas multiprocessados. Isto permite a transferência de um byte em 500 ns ou menos. Apesar do M37700 receber informações de apenas um M37450 em um determinado instante, ele pode transmitir a mais de um simultaneamente.

Como visto na figura 3.1, o mestre possui sua própria memória (RAM/ROM). Cada coprocessador também está munido de uma memória, no presente caso memória RAM, podendo ser, também, memória ROM. Quando os coprocessadores estão desativados, com seus barramentos em estado de alta impedância, suas memórias se tornam parte do espaço de endereçamento do mestre. Esta particularidade permite ao M37700 trocar os programas e os dados das memórias dos M37450 antes de ativá-los.

3.1.2 O processador mestre

3.1.2.1 Características básicas

- Número de instruções básicas: 103;
- Memória RAM interna: 512 bytes;
- Memória máxima: 16 Mbytes;
- *Clock* externo: 16 MHz;
- Tempo de execução de instruções (instrução mais rápida): 250 ns;
- Interrupções: 19 tipos, 7 níveis;
- Contadores de 16 bits: 8;
- UART's (também pode ser síncrona): 2;
- Conversor Analógico/Digital (resolução de 8 bits): 8 canais de entrada;
- *Watchdog timer* de 12 bits;

- Entradas/saídas digitais programáveis: 68.

3.1.2.2 Espaço de endereçamento

O espaço de endereçamento do M37700 é de 16 Mbytes, do endereço 0*h* até o FFFFFF*h*. Este espaço é dividido em unidades de 64 Kbytes denominadas bancos. Assim ele pode acessar até 256 bancos. Os três primeiros Mbytes do espaço de endereçamento (os bancos 00 até 2F*h*) estão localizados internamente ao controlador e estão divididos de acordo com a figura 3.2.

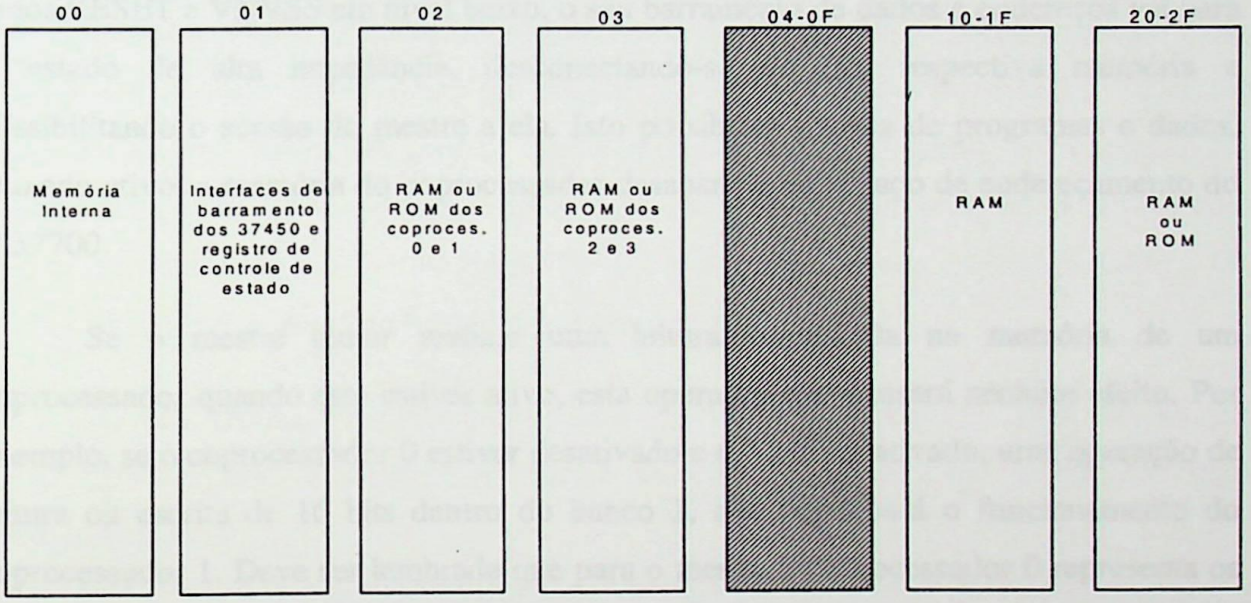


Fig. 3. 2 Mapa da memória interna ao controlador (três primeiros Mbytes).

Um espaço de 2 Mbytes (bancos 10*h* a 2F*h*) está reservado para as memórias dedicadas do M37700 (RAM ou ROM). As memórias dos coprocessadores 0 e 1, quando estes estão desativados, formam o banco 2. O mestre acessa a memória do primeiro através dos endereços pares e a do segundo através dos endereços ímpares. Da mesma forma, as memórias dos coprocessadores 2 e 3, quando igualmente desativados, formam o banco 3. O banco 1 permite a comunicação diretamente com os coprocessadores, via interface de barramento. Ele comporta igualmente um registro de controle de estado. Finalmente o banco 0 contém as memórias internas do M37700. Os 13 Mbytes não localizados no controlador podem ser acessados através de um conector especial.

3.1.2.3 Memórias dedicadas

O controlador possui quatro soquetes de 32 pinos onde podem ser inseridos os circuitos integrados de memória de 8 bits. Um primeiro grupo, 2 soquetes, contém as memórias RAM. O segundo grupo, também 2 soquetes, pode conter tanto RAM como ROM. As memórias RAM permitidas são de dimensões 32Kx8 à 512Kx8 e as ROM de 16Kx8 à 64Kx8.

3.1.2.4 Acesso as memórias dos coprocessadores pelo mestre

O controlador tira proveito do fato de que quando um M37450 está desativado, pinos RESET e VNVSS em nível baixo, o seu barramento de dados e endereços vai para o estado de alta impedância, desconectando-se de sua respectiva memória e possibilitando o acesso do mestre a ela. Isto possibilita a troca de programas e dados. Quando ativo, a memória do coprocessador desaparece do espaço de endereçamento do M37700.

Se o mestre tentar realizar uma leitura ou escrita na memória de um coprocessador quando este estiver ativo, esta operação não causará nenhum efeito. Por exemplo, se o coprocessador 0 estiver desativado e o 1 estiver ativado, uma operação de leitura ou escrita de 16 bits dentro do banco 2, não perturbará o funcionamento do coprocessador 1. Deve ser lembrado que para o mestre o coprocessador 0 representa os endereços pares e o 1 os endereços ímpares no banco.

3.1.2.5 Interface de barramento dos coprocessadores

Como visto anteriormente, o mestre comunica-se com os coprocessadores através do banco 1. As interfaces de barramento de cada M37450 aparecem como dois endereços dentro deste banco.

O endereço de base é o registro de dados principal. Uma escrita neste endereço transmite um byte ao registro de entrada da interface de barramento do M37450 com o pino de controle da interface A0 mantido em 0. Uma leitura permite obter um byte do registro de saída da interface. Deve-se notar que os chamados registro de entrada e registro de saída são na realidade um único registro nos coprocessadores. A diferença

nos nomes serve apenas para indicar se o dado foi armazenado no registro pelo mestre ou pelo coprocessador, respectivamente.

O segundo endereço, situado dois bytes acima do endereço de base, permite enviar um byte ao registro de entrada juntamente com o pino de controle da interface A0 mantido em 1. A leitura deste endereço permite ao M37700 obter o registro de estado da interface do coprocessador.

Nota-se que o estado dos bits 0, 1 e 3 do registro de estado da interface são bits somente para leitura, indicando o estado do *buffer* do barramento de dados, sendo que o bit 3 representa o valor do pino A0. Assim, lendo o valor do bit 3, o coprocessador pode diferenciar o valor armazenado no registro de entrada, ou seja, este bit pode ser considerado um nono bit para o valor armazenado no registro de entrada. Na implementação aqui realizada, se o valor do bit A0 for 1 o valor armazenado no registro é o da amplitude; se o valor do bit A0 for 0 o valor armazenado é o da freqüência. Os outros bits têm suas funções definidas pelo programador.

O bit 2 indica, por convenção, que a interface não está pronta para receber um byte. Por isso, para utilização da interface é necessário que o bit 2 seja mantido em zero todo o tempo.

A figura 3.3 ilustra o circuito para seleção das interfaces de barramento dos coprocessadores. A linha *IBS** provém dos decodificadores de endereços e seleciona o banco 1. Verifica-se que as linhas *n:IBS** só ativarão as interfaces de barramento dos respectivos coprocessadores, se as entradas de seleção tiverem, ambas, nível lógico zero (a seleção das interfaces se faz por nível lógico zero). Com isto os endereços de base estão mostrados na tabela 3.1.

Ressalta-se que apesar de ler apenas um coprocessador por vez, o mestre pode escrever em mais de um simultaneamente. A operação de escrita simultânea nos coprocessadores é facilitada pela escolha dos endereços da tabela 3.1, através da combinação destes por operações lógicas E.

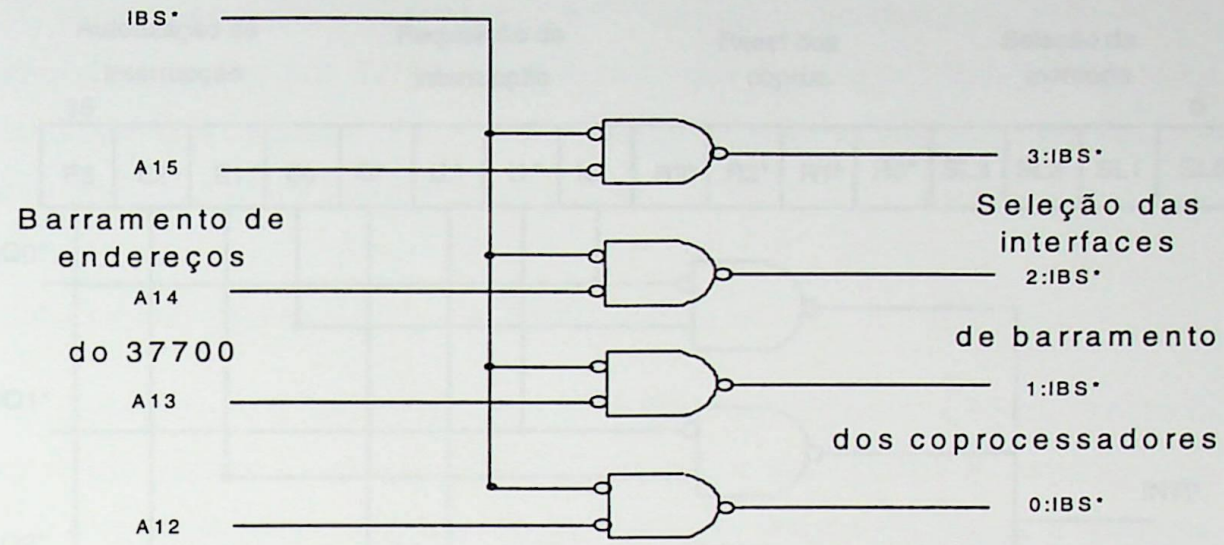


Fig. 3. 3 Circuito de seleção das interfaces de barramento dos coprocessadores.

Tabela 3. 1 Endereços de base, para seleção das interfaces de barramento dos coprocessadores, dentro do banco 1 do processador mestre.

coproc.	endereço	A15	A14	A13	A12
0	E000*h*	1	1	1	0
1	D000*h*	1	1	0	1
2	B000*h*	1	0	1	1
3	7000*h*	0	1	1	1

Logo:

a) para escrevermos nos coprocessadores 0 e 2, executamos uma operação de armazenamento (*store*) nos endereços E000*h* E B000*h*, resultando em A000*h*, ou seja, os bits A14 e A12 estão em zero, o que seleciona as interfaces dos coprocessadores 0 e 2;

b) para escrevermos em todos os coprocessadores, executamos uma operação de armazenamento nos endereços E000*h* E D000*h* E B000*h* E 7000*h*, resultando em 0000*h*, selecionando em conseqüência todas as interfaces.

3.1.2.6 Registro de controle de estado

O banco 1 comporta igualmente um registro externo de 16 bits, localizado no endereço F000*h*, possuindo diversas funções (figura 3.4).

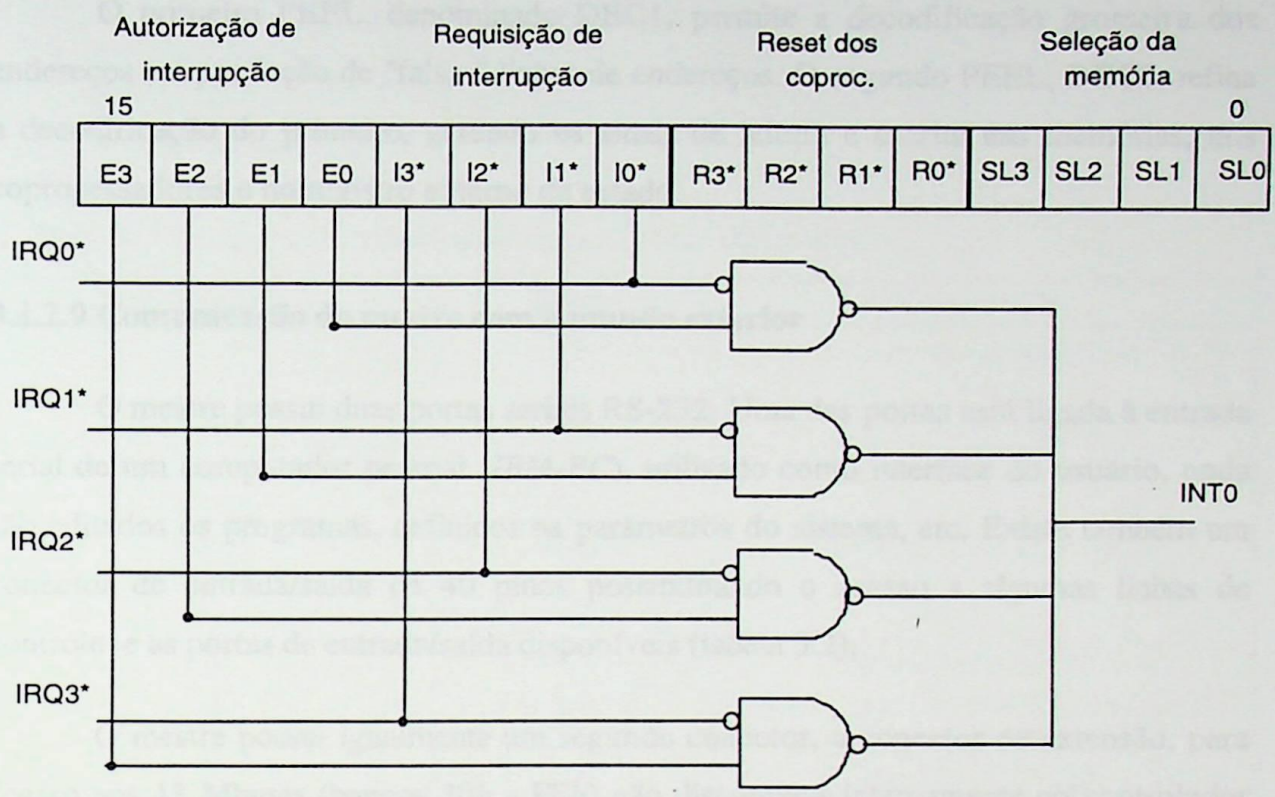


Fig. 3. 4 Registro externo de controle de estado do mestre.

Os bits SL0 a SL2 são utilizados nos circuitos de decodificação de endereços. O bit SL3 está reservado para acesso aos 13 Mbytes de memória externos ao controlador. Os bits 4 a 7 indicam se os coprocessadores estão ou não no estado de *reset*. Os bits 8 a 11 são as requisições de interrupções dos coprocessadores feitas ao mestre. Finalmente os bits 12 a 15 são as máscaras das interrupções anteriores. As linhas IRQn^* são as linhas de requisição de interrupção vindas dos coprocessadores. A linha INT0 é a linha da interrupção 0 (zero) do mestre.

3.1.2.7 Banco 0

Os registros de controle e os registros das portas de entrada e saída, os vetores de interrupção e a memória interna do M37700 residem dentro do banco 0.

3.1.2.8 PEELs

Um PEEL é um componente do tipo PAL que é programável e apagável eletricamente. O controlador comporta dois PEELs.

O primeiro PEEL, denominado DEC1, permite a decodificação grosseira dos endereços e a produção de "falsas" linhas de endereços. O segundo PEEL, DEC2, refina a decodificação do primeiro, gerando os sinais de leitura e escrita nas memórias, nos coprocessadores e no registro externo de estado.

3.1.2.9 Comunicação do mestre com o mundo exterior

O mestre possui duas portas seriais RS-232. Uma das portas está ligada à entrada serial de um computador pessoal (*IBM-PC*), utilizado como interface do usuário, onde são editados os programas, definidos os parâmetros do sistema, etc. Existe também um conector de entrada/saída de 40 pinos possibilitando o acesso a algumas linhas de controle e as portas de entrada/saída disponíveis (tabela 3.2).

O mestre possui igualmente um segundo conector, o conector de extensão, para acesso aos 13 Mbytes (bancos 30*h* a FF*h*) não disponíveis internamente ao controlador (tabela 3.3).

3.1.3 Coprocessadores

3.1.3.1 Características básicas

- Número de instruções básicas: 71;
- Memória RAM interna: 512 bytes;
- Memória externa: 64 Kbytes;
- *Clock* externo: 10 MHz;
- Tempo de execução de instruções (instrução mais rápida): 0,8 μs;
- Interrupções: 15 eventos;
- Contadores de 16 bits: 3;
- Contador de 8 bits (utilizado pela entrada/saída serial): 1;
- Entrada/saída serial (UART ou clock síncrono): 1;
- Conversor Analógico/Digital (resolução de 8 bits): 8 canais;
- Conversor Digital/Analógico (resolução de 8 bits): 2 canais;
- Saída PWM (8 ou 16 bits): 1;

- Entradas/saídas digitais programáveis: 48;
- Entrada Digital: 8;
- Saída Digital: 2.

Tabela 3. 2 Pinagem do conector de entrada/saída do mestre.

Pino do conector	Nome	Observação
1	Vin	Alimentação sem regulação(7 a 20 Vdc)
2	+5	Alimentação regulada
24 26 28 30 32 34 36 38 40	Gnd	Terra digital e analógico
4 6 5 7 39	RST* CLR* PHI E* Vref	Sinal de *reset* do 37700 Sinal de *reset* do controlador (saída) *Clock* do 37700 *Enable* do 37700 Referência para conversão A/D
37 35 33 31 29 27 25 23	P77/AN7 P76/AN6 P75/AN5 P74/AN4 P73/AN3 P72/AN2 P71/AN1 P70/AN0	•Porta de entrada/saída digital •Entrada analógica dos conversores analógicos/digitais
22 21 20 19 18 3 17 16	P67/TB2in P66/TB1in P65/TB0in P64/Int2* P63/Int1* Int0* P61/TA4in P60/TA3out	•Porta de entrada/saída digital •Entrada das interrupções externas Int2*, Int1* e Int0* (esta usada pelos coprocessadores) •Entrada dos contadores tipo B •Entrada/saída dos contadores tipo A
15 14 13 12 11 10 9 8	P57/TA3in P56/TA3out P55/TA2in P54/TA2out P53/TA1in P52/TA1out P51/TAOin P50/TAOout	•Porta de entrada/saída digital •Entrada/saída dos contadores tipo A

Tabela 3. 3 Conector de extensão.

Pino do conector	Nome	Observação
2 40	+5	Alimentação regulada
1 19 39	Gnd	Terra digital e analógico
28	R/W*	*Read/Write*
29	BHE*	*Byte High Enable*
30	ALE*	*Address latch Enable*
31	HLDA*	*Hold Acknowlwdge*
32	E*	*Enable*
33	PHI	*Clock*
34	CLR*	Reset
35	RDY*	*Ready*
36	HOLD*	*Hold*
37	EXT	Extensão: nível 1 quando mestre acessa o barramento de extensão
38	SL3	Bit do registro externo
3	A0	Barramento de endereços e dados
4	A1	
5	A2	
6	A3	
7	A4	
8	A5	
9	A6	
10	A7	
11	A8/D8	
12	A9/D9	
13	A10/D10	
14	A11/D11	
15	A12/D12	
16	A13/D13	
17	A14/D14	
18	A15/D15	
20	A16/D0	
21	A17/D1	
22	A18/D2	
23	A19/D3	
24	A20/D4	
25	A21/D5	
26	A22/D6	
27	A23/D7	

3.1.3.2 Memórias

Sendo capazes de acessarem até 64 Kbytes de memória, os coprocessadores dentro do controlador estão limitados ao máximo de 32 Kbytes. Isto se deve ao fato de

que, como cada banco de memória do mestre possui 64 Kbytes, e como um banco (2 ou 3) contém a memória de dois coprocessadores, uma sendo acessada pelos endereços pares e outra pelos ímpares, cada coprocessador pode, então, acessar no máximo 32 Kbytes.

3.1.3.3 Comunicação dos coprocessadores com o mundo exterior

Cada M37450 também possui seu próprio conector de entrada/saída (tabela 3.4).

Tabela 3. 4 Conector externo dos coprocessadores.

Pino do conector	Nome	Obseravação
1	Vin	Alimentação sem regulação (7 a 20 Vdc)
2	+5	Alimentação regulada
3 18 21 26 31 34	Gnd	Terra analógico e digital
17	RAZ*	*Reset* do 37450
4	PHI	*Clock* do 37450
32	ADRef	Referência para conversão A/D
33	DARef	Referência para conversão D/A
20	DA1	Saídas conversores D/A
19	DA2	
37	P47/AN7	•Portas de entrada/saída digital •Entradas analógicas dos conversores A/D
35	P46/AN6	
33	P45/AN5	
31	P44/AN4	
29	P43/AN3	
27	P42/AN2	
25	P41/AN1	
23	P40/AN0	
8	P63/PR*	•Portas de entrada/saída digital •Requisição de interrupção (PR*) •Entrada para interrupções externas
7	P62/Int3	
6	P61/Int2	
5	P60/Int0	
16	P37/SRDY	•Portas de entrada/saída digital •Sinais para comunicação serial •Saída gerador PWM •Saída/entrada dos contadores (EVn)
15	P36/SCLK	
14	P35/TXD	
13	P34/RXD	
12	P33/PWM	
11	P32/EV3	
10	P31/EV2	
9	P30/EV1	

3.1.4 Utilização do controlador multiprocessado

Neste trabalho, o controle, em tempo real, de conversores diretos de freqüência, foi dividido entre os vários processadores do controlador. Três dos coprocessadores ficaram responsáveis, cada um, pelo controle dos três interruptores que conectam as três tensões de entrada a um mesmo ramo de saída. Eles implementam as equações para o cálculo dos tempos de acionamento dos interruptores, equações (13), (14) e (15).

O quarto coprocessador obtém o valor de pico das tensões de entradas, e o envia aos outros coprocessadores. Este valor de pico é utilizado pelas equações (13) e (14). A obtenção do valor de pico das tensões de entrada é necessária, evitando-se assim a geração de valores falsos de tempos de acionamento, como ilustrado nos exemplos a seguir.

Considerando:

$v_M = 5$ V, $v_K = -2{,}5$ V, $v_L = -2{,}5$ V, $v_0 = -2$ V, $V_i = 5$ V, T_S igual a uma unidade qualquer de tempo e substituindo os valores nas equações (13) - (15), temos:

$$t_K = \frac{(-2-5).(-2,5)}{37,5} \cong 0,47$$

$$t_L = \frac{(-2-5).(-2,5)}{37,5} \cong 0,47$$

$$t_M = 1 - 0,47 - 0,47 = 0,06$$

Logo a tensão de saída será de:

$$v_0 = 5.0,06 - 2,5.0,47 - 2,5.0,47 = -2,05 \cong -2 \text{ V}$$

que é a tensão desejada.

Se houver uma diminuição no valor da tensão de entrada, por exemplo, em 10%, os valores das tensões se tornarão:

$v_M = 4{,}5$ V, $v_K = -2{,}25$ V, $v_L = -2{,}25$ V

Para um $v_0 = -2$ V e utilizando um valor de pico de 4,5 V temos que $1{,}5\ V_i^2 = 30{,}375$ e:

$$t_K = \frac{(-2-4,5).(-2,25)}{30,375} \cong 0,48$$

$$t_L = \frac{(-2-4,5).(-2,25)}{30,375} \cong 0,48$$

$$t_M = 1 - 0,48 - 0,48 = 0,04$$

chegamos a uma tensão de saída de:

$$v_0 = 5.0,04 - 2,25.0,48 - 2,25.0,48 = -1,96 \cong -2 \text{ V}$$

Porém, realizando o cálculo dos tempos de acionamento com as tensões de entrada reduzidas em 10%, e com o valor de pico mantido em 5 V, temos:

$$t_K = \frac{(-2-4,5).(-2,25)}{37,5} = 0,39$$

$$t_L = \frac{(-2-4,5).(-2,25)}{37,5} = 0,39$$

$$t_M = 1 - 0,39 - 0,39 = 0,22$$

conseguiremos uma tensão de saída de:

$$v_0 = 4,5.0,22 - 2,25.0,39 - 2,25.0,39 = -0,765 \text{ V}$$

totalmente diferente da tensão desejada que é de – 2 V.

Finalmente, fazendo os cálculos para valores de tensões de entrada sem redução, mas com o valor de pico reduzido em 10%:

$$t_K = \frac{(-2-5).(-2,5)}{30,375} \cong 0,58$$

$$t_L = \frac{(-2-5).(-2,5)}{30,375} \cong 0,58$$

$$t_M = 1 - 0,39 - 0,39 = -0,16$$

Ou seja, obtermos um valor negativo para o tempo de acionamento, gerando uma situação que não deve ocorrer.

Observa-se, dos exemplos, que o valor da tensão de saída permanece invariável, mesmo com variações nos valores das amplitudes das tensões de entrada. Porém, estas variações devem ser fornecidas ao programa de controle através de uma aquisição contínua do valor de pico das tensões.

Dos coprocessadores usados para o controle dos interruptores são utilizados: quatro conversores A/D (ADCs, *Analog to Digital Converters*), três portas digitais e três contadores. Os ADCs (trabalhando com tensões analógicas entre 0 e 5 volts) são responsáveis pela aquisição dos valores dos módulos das tensões de entrada e do valor de pico dessas tensões. As portas digitais obtém os valores das polaridades das tensões de entrada. Os contadores são os dispositivos responsáveis pela geração dos sinais de acionamento dos interruptores. Cada contador aciona um interruptor. Do coprocessador usado para a obtenção do valor de pico são utilizados três ADCs, um conversor D/A (DAC, *Digital to Analog Converter*) e uma entrada digital. Os ADCs realizam a aquisição dos valores de pico das três tensões de entrada e o DAC envia este valor aos outros coprocessadores. A entrada digital lê um sinal para se manter o sincronismo do programa para obtenção do valor de pico com a rede elétrica.

Por sua vez, o mestre gerencia a comunicação dos coprocessadores com o *PC*. Através do mestre os programas desenvolvidos no *PC* são carregados nas memórias dos coprocessadores e os valores das referências dos parâmetros (freqüência e amplitude das tensões de saída) são enviados aos M37450. Estas últimas ações são realizadas através de rotinas escritas em *Forth*.

3.2 Circuitos de interface e condicionamento

As tensões de entrada (v_A, v_B e v_C) são obtidas de um sistema trifásico. Estas tensões são reduzidas, através de transformadores, a níveis compatíveis com a operação dos interruptores utilizados, como mostrado na figura 3.5. Para assegurar uma isolação elétrica entre o sistema de controle e o do conversor são usados amplificadores isoladores (ISO122) e optoisoladores (HP2300) como interfaces de entrada e de saída para o controlador, respectivamente. Após os amplificadores isoladores existe um bloco de defasamento que adianta em aproximadamente 10° as tensões de entrada. Isto ocorre para compensar o atraso de 500 μs gerado pelo programa de controle, quando da aplicação dos sinais de acionamento aos interruptores,.pois definimos um ciclo de operação de 500 μs para o programa. Antes dos coprocessadores fazerem a aquisição

dos valores das tensões, é necessário realizar o condicionamento destes valores. Os circuitos usados para o condicionamento dos valores de tensão são:

- retificadores de precisão, gerando o valor do módulo das tensões;
- comparadores, produzindo o valor da polaridade das tensões.

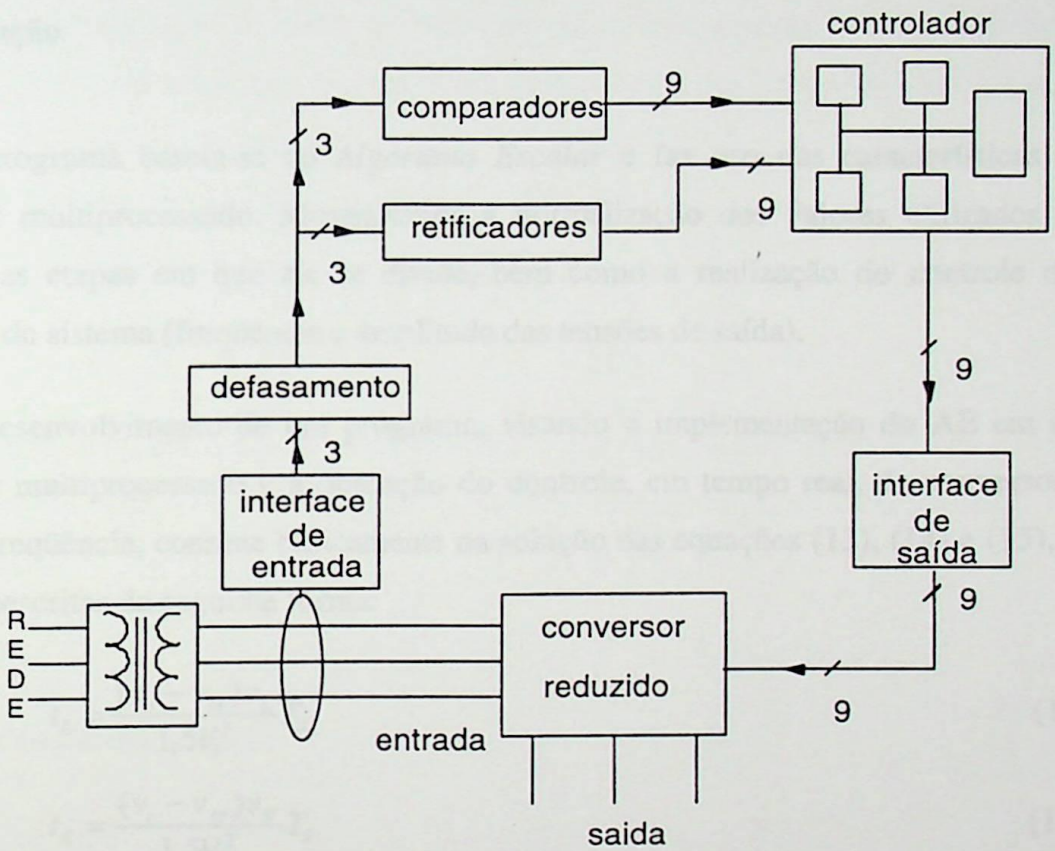


Fig. 3. 5 Circuitos de interface e condicionamento dos sinais.

CAPÍTULO 4

PROGRAMA DE CONTROLE

4.1 Introdução

O programa baseia-se no *Algoritmo Escalar* e faz uso das características do controlador multiprocessado. Mostraremos a normalização dos valores utilizados no programa, as etapas em que ele se divide, bem como a realização do controle dos parâmetros do sistema (freqüência e amplitude das tensões de saída).

O desenvolvimento de um programa, visando a implementação do AE em um controlador multiprocessado e a obtenção do controle, em tempo real, de conversores diretos de freqüência, consiste basicamente na solução das equações (13), (14) e (15), as quais são reescritas da seguinte forma:

$$t_L = \frac{(v_o - v_M)v_L}{1{,}5V_i^2} T_S \tag{18}$$

$$t_K = \frac{(v_o - v_M)v_K}{1{,}5V_i^2} T_S \tag{19}$$

$$t_M = T_S - t_K - t_L \tag{20}$$

Estas equações definem os tempos de acionamento dos três interruptores que compõem um ramo de saída. Três coprocessadores respondem, cada um, pelo controle desses três interruptores. O quarto coprocessador obtém o valor de pico das tensões de entrada, utilizado nas equações (18) e (19), e o envia aos outros coprocessadores. O programa de controle e o de obtenção do valor de pico foram escritos em linguagem *Assembly*, em um editor de texto genérico (*Edit* do MS-DOS), e convertido em linguagem de máquina através de um compilador universal (*Cross-32 Meta-Assembler*). A transferência dos programas para as memórias dos coprocessadores, bem como o

controle dos parâmetros do sistema, são realizados através de comandos escritos em *Forth*.

Um ciclo de operação do programa divide-se nas seguintes etapas:

- obtenção dos valores da tensão de saída desejada v_o e das tensões de entrada v_A, v_B e v_C;
- obtenção do valor de pico das tensões de entrada e sua manipulação para utilização nas equações para cálculo dos tempos de acionamento dos interruptores;
- determinação das tensões v_K, v_L e v_M;
- cálculo dos tempos de ativação dos interruptores t_K, t_L e t_M;
- geração dos sinais de acionamento.

A priori, faremos algumas considerações à respeito da normalização dos valores de tensão e de tempo utilizados para o cálculo dos tempos de acionamento dos interruptores.

4.2 Normalização dos valores

Devido ao fato de que nas equações para cálculo dos tempos de acionamento dos interruptores existem tanto valores de tensão como valores de tempo, devemos realizar uma normalização destes para valores decimais ou hexadecimais, podendo assim, serem trabalhados em *Assembly*.

Como nos coprocessadores os ADCs de 8 bits trabalham com sinais analógicos variando entre zero e +5 volts, podemos obter uma tabela de correspondência entre valores de tensão e valores decimais e hexadecimais (tabela 4.1). Nesta tabela os valores entre zero e +5 volts são definidos pela faixa de conversão dos ADCs, sendo os demais decorrentes desta faixa.

Tabela 4. 1 Correspondência entre valores de tensão e valores decimais e hexadecimais.

tensão (V)	valor decimal	valor hexadecimal
0	0	00
5	255	FF
10	510	1FE
15	765	2FD
20	1020	3FC
25	1275	4FB
30	1530	5FA
35	1785	6F9
40	2040	7F8

Tomemos como exemplo um valor de pico (V_i) igual a +5 volts. Assim temos que o termo $1,5V_i^2$ das equações será igual à 37,5 volts. Então pela tabela 4.1 temos:

$$1,5V_i^2 = 37,5V = 1912 = 778h$$

Será visto adiante que os contadores de 16 bits dos M37450 são usados para a geração dos sinais de acionamento dos interruptores. Devido a isto, o tempo de duração dos sinais de acionamento é definido pelo número de decrementos, cada um durando um ciclo de instrução, do valor armazenado nos registros dos contadores. Assim podemos obter uma segunda tabela, de correspondência entre valores de tempo e valores decimais e hexadecimais (tabela 4.2). Nesta os valores decimais e hexadecimais correspondem ao número de decrementos necessários para os contadores gerarem um pulso com a respectiva duração de tempo. Esta tabela decorre do fato de que o sinal de acionamento começa com o início do decremento do valor carregado nos registros e termina com a ocorrência do decremento de $0000h$ para $FFFFh$. Por isso, se no registro de um contador for carregado o valor zero, este valor sofrerá um decremento. Conseqüentemente gerará um pulso de 0,4 μs, que é a duração de um ciclo de instrução com os coprocessadores operando a uma freqüência de 10 MHz..

Tabela 4. 2 Correspondência entre valores de tempo e valores decimais e hexadecimais.

tempo (μs)	número de decrementos	correspondente decimal	correspondente hexadecimal
0,4	1	0	00
100	250	249	F9
200	500	499	1F3
300	750	749	2ED
400	1000	999	3F7
500	1250	1249	4E1
600	1500	1499	5DE

Escolheu-se uma freqüência de comutação (f_S) de 2000 Hz. Como aplicações usuais na indústria trabalham com conversores gerando freqüências de até 200 Hz, a f_S escolhida está em concordância com o Teorema de Nyquist. Além disso, esta freqüência resulta num período de comutação (T_S) de 500 μs, que também é a duração de um ciclo do programa, tornando viável o controle, em tempo real, dos conversores.

Logo, temos o período T_S correspondendo a:

$$T_S = 500\ \mu s = 1250 \text{ decrementos} = 1249 = 4E1h$$

Nas equações (18) e (19) observamos que, nos termos $(v_o - v_M)v_L$ e $(v_o - v_M)v_K$, ao se fazer a soma algébrica o resultado ficará corretamente dentro da tabela 4.1. Mas na multiplicação, para manter o resultado coerente com a tabela, devemos dividir o produto por 51 ou $33h$, que é o valor correspondente a 1 volt. Veja exemplo a seguir.

Considerando v_o = 2 V, v_M = -5 V, v_L = 3 V e utilizando os valores obtidos pela tabela 4.1 temos:

$$(v_o - v_M)v_L = (2+5).3 = (102+255).153$$
$$= 7 \ .3 = 357 \ .153 \rightarrow 357 \text{ corresponde a 7 V}$$
$$= 21 = 54621 \rightarrow 54621 \text{ não corresponde}$$

a 21V, que pela tabela 4.1 deve ser 1071.

No entanto, dividindo-se o produto por 51, obtemos:

$$\frac{54621}{51} = 1071 \rightarrow \text{valor coerente com a tabela.}$$

Logo, substituindo o valor de T_S por $4E1h$ e fazendo a divisão por $33h$, reescrevemos as equações (18) e (19) como:

$$t_L = \frac{(v_o - v_M)v_L}{1,5V_i^2} \cdot \frac{4E1h}{33h} = \frac{(v_o - v_M)v_L}{1,5V_i^2} .18h$$

$$t_K = \frac{(v_o - v_M)v_K}{1,5V_i^2} \cdot \frac{4E1h}{33h} = \frac{(v_o - v_M)v_K}{1,5V_i^2} .18h$$

Ou de outro modo:

$$t_L = \frac{(v_o - v_M)v_L}{\frac{1{,}5V_i^2}{18h}}$$

$$t_K = \frac{(v_o - v_M)v_K}{\frac{1{,}5V_i^2}{18h}}$$

Substituindo o valor de $1{,}5V_i^2$ por $778h$, chegamos à:

$$t_L = \frac{(v_o - v_M)v_L}{4Eh}$$

$$t_K = \frac{(v_o - v_M)v_K}{4Eh}$$

Os M37450 operam apenas com valores inteiros. Porém a divisão quase sempre resulta em valores fracionários. Para contornar este problema, utilizamos a seguinte sistemática, evitando a manipulação de números fracionários:

- na divisão de um número x por um número y, (x/y), obtém-se o mesmo resultado quando se multiplica o número x pelo inverso do número y, $(x.1/y)$;
- portanto, no cálculo dos tempos, ao invés de se dividir por 4Eh, é realizada a multiplicação por 1/4Eh, e como:

$$\frac{1}{4Eh} \cong 0{,}0348h \cong 348.16^{-4}h$$

as equações (18) e (19) tornam-se:

$$t_L = 348h(v_o - v_M)v_L \qquad (21)$$

$$t_K = 348h(v_o - v_M)v_K \qquad (22)$$

Ressalta-se que o valor do multiplicador é $348h$ e não $348.16^{-4}h$ pois do resultado final de 32 bits da multiplicação são desprezados os dois bytes (os quatro dígitos hexadecimais) menos significativos, ou seja, é considerada apenas a parte inteira do resultado. Vejamos um exemplo a seguir, utilizando-se números decimais.

Multiplicando-se 542 por 0,0348 obtemos:

542.0,0348 = 18,8616

Por sua vez, considerando os números como inteiros:

$$542.348 = 188616$$

e desprezando os quatro dígitos menos significativos temos como resultado 18; somente a parte inteira do resultado é levada em consideração.

4.3 Etapas de operação do programa

4.3.1 Obtenção dos valores da tensão de saída desejada e das tensões de entrada

No início de cada ciclo de operação do programa, obtém-se os valores da tensão de saída desejada (v_o) e das tensões de entrada v_A, v_B e v_C.

4.3.1.1 Obtenção dos valores da tensão de saída desejada

Os valores do módulo de v_o são obtidos através de uma tabela armazenada nas memórias RAM dos coprocessadores. Esta é composta por 256 valores de 8 bits, variando de 00*h* a FF*h*. Estes valores representam um quarto do período de uma onda senoidal, ou seja, 90 graus. Assim, entre dois valores consecutivos da tabela há uma diferença de:

$$\frac{90^0}{256} = 0,352^0$$

A freqüência da senóide de saída é determinada pelo passo de amostragem da tabela. Portanto considerando o ciclo de operação do programa, a cada 500 μs um valor é lido da tabela. A menor freqüência ocorrerá quando a tabela for amostrada com um passo unitário. Assim temos:

500 μs ---------- $0,352^0$

x μs ---------- 360^0

Ou seja, se a cada 500 μs é amostrado um valor defasado $0,352^0$ do anterior, quanto tempo será gasto para se completar 360^0, ou um período completo da senóide? No presente caso, este tempo será de 0,51 s. Em conseqüência, a menor freqüência a ser gerada é de 1,96 Hz. Todas as demais freqüências serão múltiplas desta. Por exemplo,

para geramos uma freqüência dez vezes superior a esta, devemos utilizar um passo de 10. Portanto, a cada amostragem obteremos um valor com uma defasagem de $3,52^0$ do anterior:

500 μs ---------- $3,52^0$

x μs ---------- 360^0

obtendo x = 0,051 s ou 19,60 Hz.

No apêndice I são feitas considerações sobre como diminuir o passo de 1,96 Hz para geração das tensões de saída.

Através do sentido de deslocamento na tabela, se da primeira posição (00*h*) para a última (FF*h*), ou vice-versa, usado em conjunto com o valor da polaridade de v_o, torna-se possível a obtenção da tensão de saída desejada em toda a sua faixa de 0 à 360 graus. Assim, após ser obtido um valor da tabela, a posição deste é incrementada ou diminuída. Se estiver sendo gerado o primeiro ou o terceiro quarto da senóide, a posição é incrementada. No primeiro caso, a polaridade é positiva, no segundo ela é negativa. Se estiver sendo gerado o segundo ou o quarto quarto da senóide, a posição é diminuída, com a polaridade positiva para o primeiro caso e negativa para o segundo. O valor do incremento ou do decremento corresponde ao passo de amostragem da tabela. Após a obtenção de um valor, é chamada uma rotina que realiza a implementação das outras etapas.

Para se fazer o retorno na tabela, na passagem de um quarto para outro da senóide, temos duas situações:

a) Retorno ao se atingir o final da tabela (passagem do primeiro para o segundo e do terceiro para o quarto quartos).

Considera-se a figura 4.1, representando o módulo da tensão de saída.

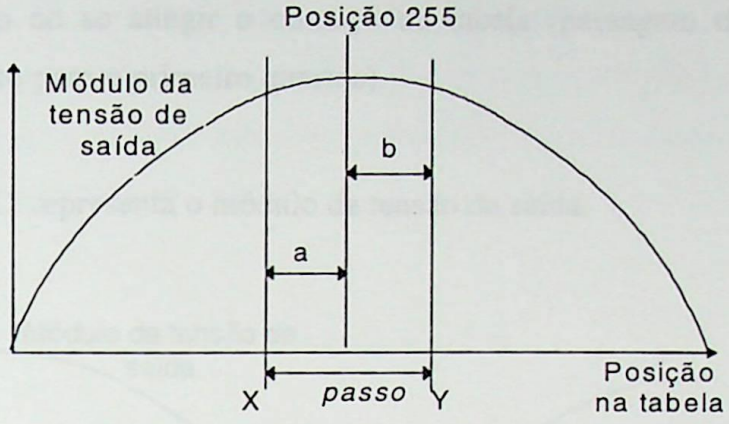


Fig. 4. 1 Retorno quando se atinge o final da tabela.

Na figura a distância *passo* equivale ao valor do passo de amostragem da tabela. Ao atingir-se uma posição maior do que a diferença (255 - *passo*), é o momento de se retornar na tabela.

Designando-se como X a última posição antes do retorno, e como Y a primeira após, para se achar Y procede-se da seguinte maneira:

- calcula-se a, $a = 255 - X$, indicando quantas posições a partir de X faltam para se alcançar a posição 255;
- calcula-se agora b, $b = passo - a$, indicando quantas posições após 255 faltam para se alcançar Y;
- finalmente acha-se a posição Y, $Y = 255 - b$.

Exemplo: considerando-se um *passo* = 10 (freqüência de 19,60 Hz), temos que a diferença 255 - *passo* é igual a 255 - 10 = 245. Logo, supondo que durante a amostragem da tabela se atinja a posição 251, vemos que é o instante de se retornar na tabela ($X = 251 > 245$). Então:

$$a = 255 - 251 = 4$$

$$b = 10 - 4 = 6$$

$$Y = 255 - 6 = 249$$

Assim o próximo valor a ser obtido é o valor da posição 249. E a partir desta posição a amostragem continua, porém com o decremento da posição.

b) Retorno ao se atingir o começo da tabela (passagem do segundo para o terceiro e do quarto para o primeiro quartos).

A figura 4.2 representa o módulo da tensão de saída.

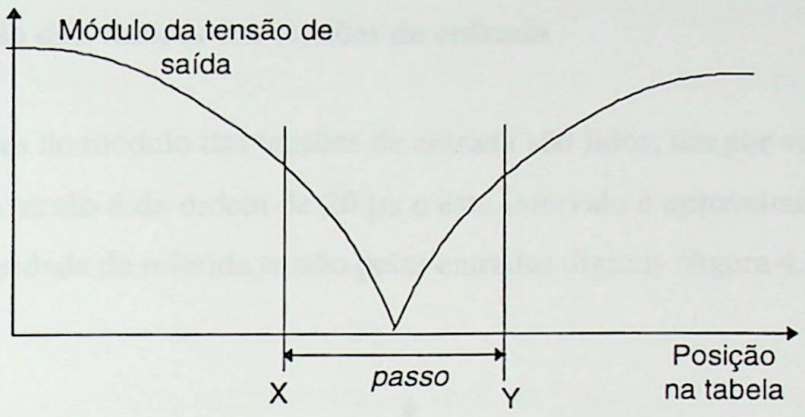


Fig. 4. 2 Retorno quando se atinge o começo da tabela.

Neste caso ocorrerá o retorno na tabela quando for atingida uma posição menor que o valor do *passo*, procedendo-se da seguinte maneira:
- calcula-se simplesmente a posição *Y*, $Y = passo - X$.

Mas logicamente, quando da ocorrência deste retorno, é necessário fazer uma inversão no valor da polaridade da tensão de saída.

Exemplo: valendo-se novamente de um *passo* = 10 e supondo que durante a amostragem da tabela atinja-se a posição 9, vemos que é o instante de se retornar na tabela ($X = 9 < 10$). Então:

$$Y = 10 - 9 = 1$$

Assim, o próximo valor a ser obtido é o valor da posição 1. E a partir desta posição a amostragem continua, no entanto com o incremento da posição.

Para a geração de um sistema trifásico senoidal na saída do conversor, cada coprocessador inicia a varredura da tabela com os valores do módulo de v_0 de um ponto diferente. O primeiro coprocessador inicia na posição zero da tabela e com o valor da

polaridade positiva. O segundo começa na posição 170 e polaridade também positiva (adiantado 120 graus em relação ao primeiro). Finalmente o terceiro inicia a varredura da tabela também na posição 170, mas com polaridade negativa (atrasado 120 graus em relação ao primeiro coprocessador).

4.3.1.2 Obtenção dos valores das tensões de entrada

Os valores do módulo das tensões de entrada são lidos, um por vez, pelos ADCs. O tempo de conversão é da ordem de 20 μs e este intervalo é aproveitado para a leitura do valor da polaridade da referida tensão pelas entradas digitais (figura 4.3).

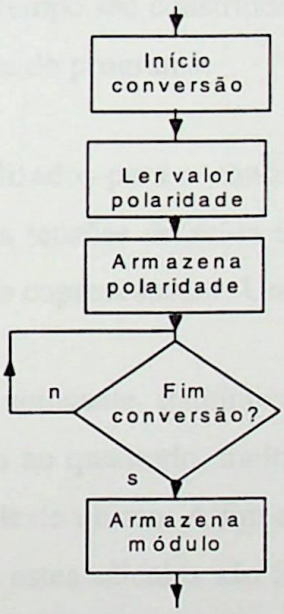


Fig. 4. 3 Fluxograma da leitura de uma tensão de entrada.

4.3.2 Obtenção do valor de pico das tensões de entrada

O valor de pico das tensões de entrada é obtido pelo quarto coprocessador do controlador. A técnica utilizada consiste na leitura direta do valor de pico. Três ADCs são utilizados para a leitura do valor de pico das três tensões de entrada retificadas. Um sinal de sincronismo, a mudança da polaridade da tensão v_c de positiva para negativa

(*edge* negativo), inicia a operação do programa. A partir da ocorrência do sinal de sincronismo, o programa espera um tempo correspondente a 30 graus elétricos (1,39 ms). Completando-se o tempo, é lido o valor de pico da tensão v_A. Envia-se este valor aos outros coprocessadores através da saída de um DAC. A seguir o programa espera um novo tempo, correspondente a 60 graus elétricos (2,78 ms), ao fim do qual lê-se o valor de pico da tensão v_C, também enviado aos outros coprocessadores. Através das etapas anteriores - espera de um tempo de 2,78 ms, leitura do valor de pico e envio deste aos coprocessadores - o programa lê consecutiva e repetidamente as tensões v_B, v_A, v_C e v_B. Logo, o valor de pico é atualizado a cada 2,78 ms. Após a segunda leitura de v_B, o programa aguarda a ocorrência de uma nova mudança da polaridade de v_C, para reiniciar o ciclo de leitura das tensões. O valor da polaridade de v_C é lido por uma entrada digital. Os intervalos de tempo são construídos com artifícios de programação. A figura 4.4 apresenta o fluxograma do programa.

Nos coprocessadores utilizados para o controle dos interruptores do conversor, após a obtenção dos valores das tensões de saída e de entrada, realiza-se a leitura do valor de pico enviado pelo quarto coprocessador. Um ADC lê o valor.

Para obter o valor da constante multiplicativa das equações (21) e (22) é necessário elevar o valor de pico ao quadrado, multiplicar o resultado por 1,5, dividi-lo por $18h$ e determinar o inverso deste último. A fim de contornar esta situação elaborou-se uma tabela, tabela 4.3. Todos estes cálculos são feitos antecipadamente para diversos valores de V_i, valores estes baseados na tabela 4.1. A tabela com os valores da constante multiplicativa é armazenada na memória dos coprocessadores.

Com o valor de pico recebido do quarto coprocessador, através da busca na tabela, obtém-se o valor da constante multiplicativa para utilização no cálculo dos tempos de acionamento dos interruptores. A figura 4.5 mostra o fluxograma desta etapa.

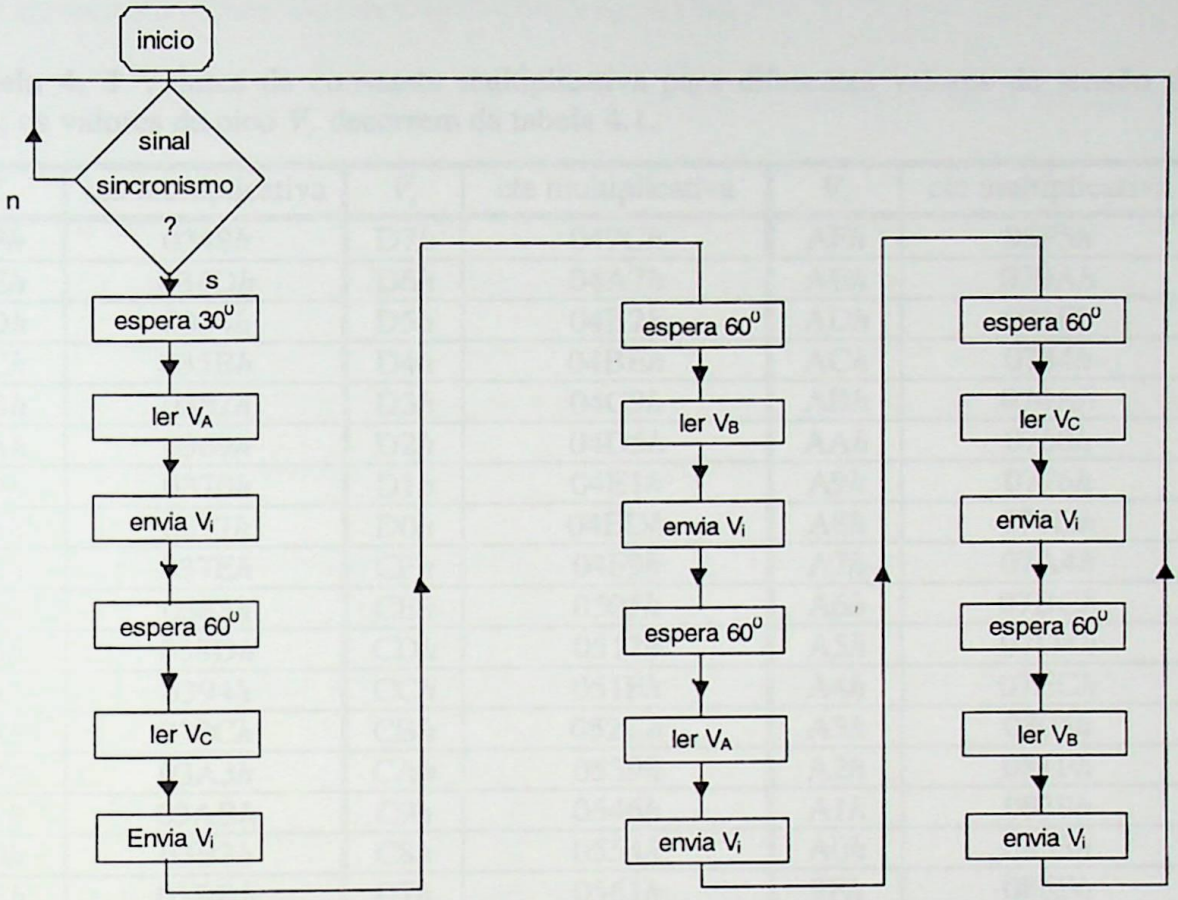


Fig. 4. 4 Fluxograma do programa para aquisição do valor de pico das tensões de entrada.

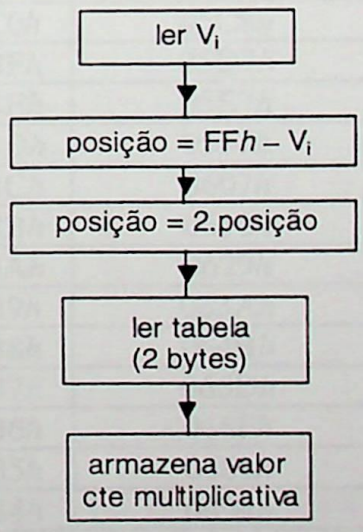


Fig. 4. 5 Obtenção dos valores das constantes multiplicativas.

Tabela 4. 3 Valores da constante multiplicativa para diferentes valores de tensão de pico; os valores de pico V_i decorrem da tabela 4.1.

V_i	cte multiplicativa	V_i	cte multiplicativa	V_i	cte multiplicativa
FF*h*	0348*h*	D7*h*	049C*h*	AF*h*	06F5*h*
FE*h*	034D*h*	D6*h*	04A7*h*	AE*h*	070A*h*
FD*h*	0354*h*	D5*h*	04B2*h*	AD*h*	071F*h*
FC*h*	035B*h*	D4*h*	04BE*h*	AC*h*	0734*h*
FB*h*	0362*h*	D3*h*	04C9*h*	AB*h*	074A*h*
FA*h*	0369*h*	D2*h*	04D5*h*	AA*h*	0760*h*
F9*h*	0370*h*	D1*h*	04E1*h*	A9*h*	0776*h*
F8*h*	0377*h*	D0*h*	04ED*h*	A8*h*	078D*h*
F7*h*	037E*h*	CF*h*	04F9*h*	A7*h*	07A4*h*
F6*h*	0385*h*	CE*h*	0505*h*	A6*h*	07BC*h*
F5*h*	038D*h*	CD*h*	0512*h*	A5*h*	07D4*h*
F4*h*	0394*h*	CC*h*	051F*h*	A4*h*	07EC*h*
F3*h*	039C*h*	CB*h*	052C*h*	A3*h*	0805*h*
F2*h*	03A3*h*	CA*h*	0539*h*	A2*h*	081F*h*
F1*h*	03AB*h*	C9*h*	0546*h*	A1*h*	0839*h*
F0*h*	03B3*h*	C8*h*	0554*h*	A0*h*	0853*h*
EF*h*	03BB*h*	C7*h*	0561*h*	9F*h*	086F*h*
EE*h*	03C3*h*	C6*h*	056F*h*	9E*h*	0889*h*
ED*h*	03CB*h*	C5*h*	057E*h*	9D*h*	08A5*h*
EC*h*	03D3*h*	C4*h*	058C*h*	9C*h*	08C2*h*
EB*h*	03DC*h*	C3*h*	059B*h*	9B*h*	08DF*h*
EA*h*	03E4*h*	C2*h*	05A9*h*	9A*h*	08FC*h*
E9*h*	03ED*h*	C1*h*	05B8*h*	99*h*	091B*h*
E8*h*	03F5*h*	C0*h*	05C8*h*	98*h*	0939*h*
E7*h*	03FE*h*	BF*h*	05D7*h*	97*h*	0959*h*
E6*h*	0407*h*	BE*h*	05E7*h*	96*h*	0979*h*
E5*h*	0410*h*	BD*h*	05F7*h*	95*h*	0999*h*
E4*h*	0419*h*	BC*h*	0607*h*	94*h*	09BB*h*
E3*h*	0423*h*	BB*h*	0618*h*	93*h*	09DD*h*
E2*h*	042C*h*	BA*h*	0629*h*	92*h*	0A00*h*
E1*h*	0435*h*	B9*h*	063A*h*	91*h*	0A23*h*
E0*h*	043F*h*	B8*h*	064B*h*	90*h*	0A41*h*
DF*h*	0449*h*	B7*h*	065D*h*	8F*h*	0A6C*h*
DE*h*	0453*h*	B6*h*	066F*h*	8E*h*	0A92*h*
DD*h*	045D*h*	B5*h*	0681*h*	8D*h*	0AB8*h*
DC*h*	0467*h*	B4*h*	0694*h*	8C*h*	0AE0*h*
DB*h*	0471*h*	B3*h*	06A7*h*	8B*h*	0B08*h*
DA*h*	047C*h*	B2*h*	06BA*h*	8A*h*	0B31*h*
D9*h*	0486*h*	B1*h*	06CD*h*	89*h*	0B5B*h*
D8*h*	0491*h*	B0*h*	06E1*h*	88*h*	0B86*h*

4.3.3 Determinação dos valores das tensões v_K, v_L e v_M

Após a obtenção dos valores de v_A, v_B , v_C e v_0, determinam-se as tensões v_K, v_L e v_M. Primeiro acha-se a tensão de entrada com polaridade oposta às outras, v_M, através da comparação dos valores das polaridades das tensões. Em seguida, pela comparação entre os valores do módulo das tensões restantes determinam-se v_K e v_L. Sendo a de menor valor absoluto designada como v_K e a outra como v_L. Finalmente, o valor do sinal de v_M é comparado com o valor do sinal de v_0. Nas figura 4.6 e 4.7 estão mostrados o fluxograma completo para a determinação das tensões.

Dependendo dos valores das polaridades de v_M e v_0, o programa irá para uma rotina de diferença (*calcmenos*) ou para outra de soma (*calcmais*) dos valores absolutos das duas tensões - figuras 4.8 e 4.9. Estas rotinas calculam os tempos de acionamento dos interruptores.

Nota-se, pela tabela 4.4, que o resultado de $(v_o - v_M)v_X$, onde v_X pode ser v_K ou v_L é sempre positivo, quaisquer que sejam os valores das polaridades das tensões.

Tabela 4. 4 Cálculo de $(v_o - v_M)v_K$ e $(v_o - v_M)v_L$ para valores aleatórios (em volts) de v_K, v_L, v_M e v_0.

	$v_K = -3$	$v_K = 3$	$v_K = -3$	...	$v_K = 2,7$	
	$v_L = -2$	$v_L = 2$	$v_L = -1$	...	$v_L = 1,8$	
	$v_M = 5$	$v_M = -5$	$v_M = 4$	...	$v_M = -4,5$	
$(v_o - v_M)v_K =$	+9	+21	+9	...	+17,55	$v_o = 2$
$(v_o - v_M)v_K =$	+21	+9	+21	...	+6,75	$v_o = -2$
$(v_o - v_M)v_L =$	+6	+14	+2	...	+11,7	$v_o = 2$
$(v_o - v_M)v_L =$	+14	+6	+6	...	+4,5	$v_o = -2$

Com isso, podemos dividir a soma algébrica $(v_o - v_M)$ em duas situações, evitando trabalharmos com números negativos:

a) se v_o e v_M possuem a mesma polaridade, então $(|v_M| - |v_o|)|v_X|$ é positivo. Pela definição do AE, v_M corresponde no mínimo à 86,7% da tensão de pico de entrada. Utilizando-se a proposição de Maytum e Colman (ou Venturini e Alesina), a

maior tensão de saída possível é de 86,7% da tensão de pico de entrada. Portanto v_M é sempre maior ou igual a v_o tornando a expressão anterior sempre positiva;

b) se v_o e v_M possuem polaridades diferentes, então $(|v_M|+|v_0|).|v_X|$ é positivo.

Estas duas condições dão origem as rotinas *calcmemos* e *calcmais*. Elas permitem ao programa manipular apenas números não sinalizados. As rotinas *contn* das figura 4.6 e 4.7 são as responsáveis pela geração dos sinais de acionamento dos interruptores.

4.3.4 Cálculo dos tempos de acionamento dos interruptores (t_K, t_L e t_M)

Depois da determinação das tensões v_K, v_L e v_M, efetua-se o cálculo dos tempos de acionamento dos interruptores (t_K, t_L e t_M). Estes tempos são calculados através da implementação das equações (20), (21) e (22).

Para esta implementação utilizam-se as rotinas *calcmais* e *calcmenos*, cujos fluxogramas estão nas figuras 4.8 e 4.9. Nestes fluxogramas o valor da constante multiplicativa é considerado como 348*h*, ou seja, o valor de pico das tensões de entrada é de 5 V. Na rotina *calcmenos* o fluxo do programa segue direto, pois não há ocorrência de empréstimo (*carry*) nas operações de subtração, já que v_M é sempre maior ou igual a v_o.

Sobre a rotina *calcmais* devem ser feitas considerações, devido a ocorrência ou não de vai um (*carry*) durante as operações de soma, seja para o cálculo de t_L como para o cálculo de t_K.

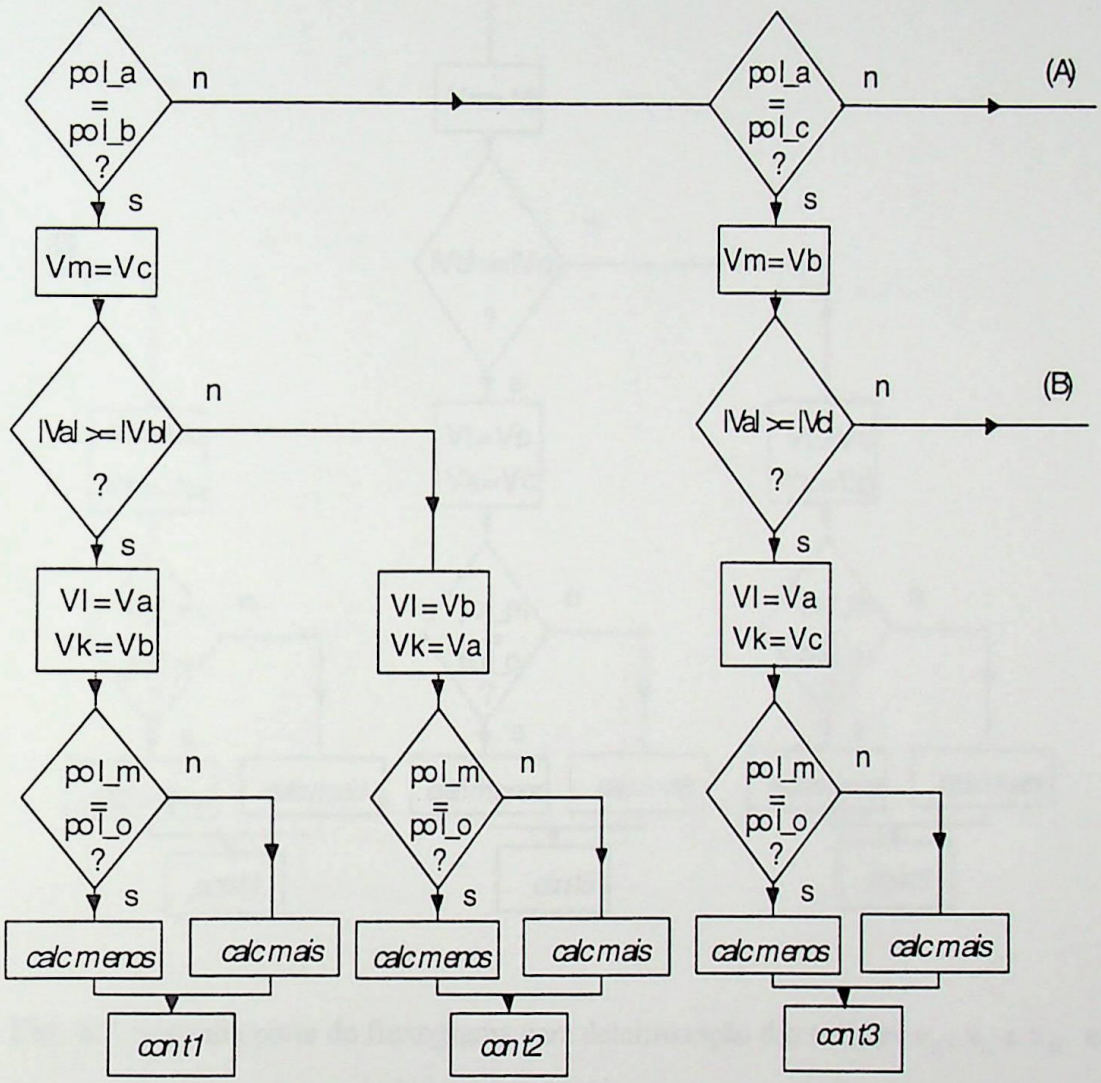


Fig. 4. 6 Primeira parte do fluxograma para determinação das tensões v_K, v_L e v_M e seleção das rotinas *contn*.

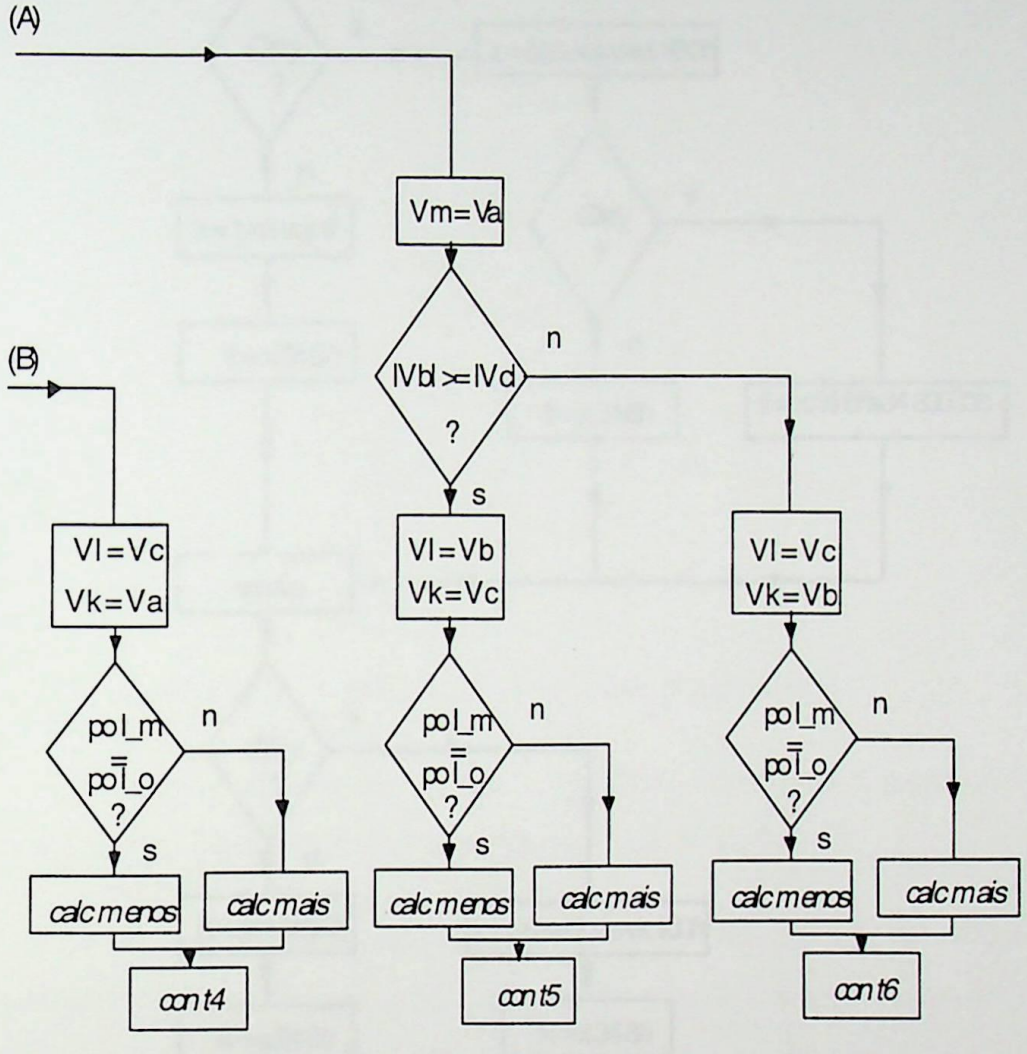


Fig. 4.7 Segunda parte do fluxograma para determinação das tensões v_K, v_L e v_M e seleção das rotinas *contn*.

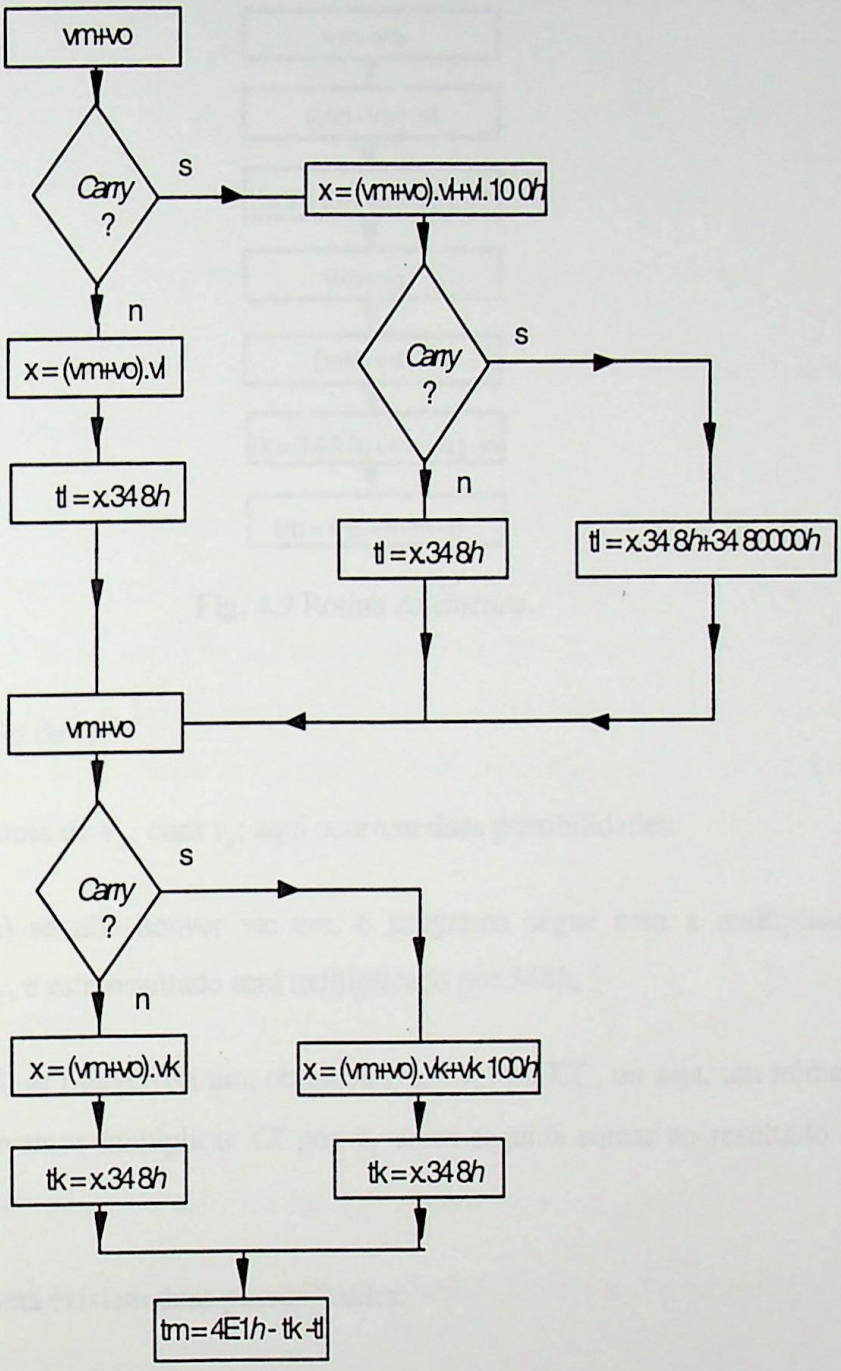


Fig. 4.8 Rotina *calcmais*.

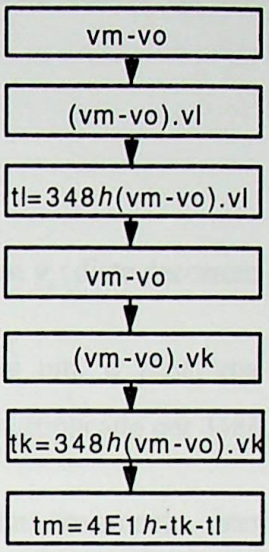


Fig. 4.9 Rotina *calcmenos*.

1) Cálculo de t_L.

Faz-se soma de v_M com v_o; aqui ocorrem duas possibilidades:

a) se não houver vai um, o programa segue com a multiplicação de $(v_M + v_o)$ por v_L, e este resultado será multiplicado por $348h$;

b) se houver vai um, obtemos $(v_M + v_o) = 1XX$, ou seja, um número com 9 bits; então devemos multiplicar XX por v_L e em seguida somar ao resultado o valor $v_L.100h$.

Nesta soma existem duas possibilidades:

b.1) se não houver vai um, o programa prossegue com a multiplicação do resultado por $348h$;

b.2) havendo vai um, ou seja, resultando num número de 17 bits, devemos multiplicar o resultado da soma por $348h$, e a seguir adicionar a este produto o valor $3480000h$. Ressalta-se o fato de que os coprocessadores manipulam instruções matemáticas de 8 bits. Logo a multiplicação por $348h$, sendo em realidade uma operação de dois valores, cada um composto por dois bytes, é realizada através de quatro multiplicações byte a byte. E como explicado anteriormente, os dois bytes menos

significativos deste produto são desprezados (somente a parte inteira do resultado é considerada).

2) Cálculo de t_K.

Efetua-se a soma de v_M com v_o; disto decorrem duas possibilidades:

a) não havendo vai um, o programa segue com a multiplicação de $(v_M + v_o)$ por v_K e este resultado multiplicado por $348h$;

b) ocorrendo vai um, novamente obtemos $(v_M + v_0) = 1XX$, e também devemos multiplicar XX por v_K e em seguida somar ao resultado o valor $v_K.100h$. Mas, em conseqüência da definição de v_K mostrada no capítulo 2, o seu valor máximo é de 50% do valor de pico da tensão de entrada. Logo no resultado da soma nunca ocorrerá vai um. Assim basta multiplicar o resultado obtido por $348h$ e desprezar os dois bytes menos significativos.

3) Cálculo de t_M.

É realizado através da expressão:

$$t_M = 4E1h - t_L - t_K$$

4.3.5 Geração dos sinais de acionamento dos interruptores

Após o cálculo dos tempos t_K, t_L e t_M, eles serão utilizados para a geração dos sinais de acionamento dos interruptores.

Neste trabalho, cada coprocessador utilizado é responsável pelo acionamento dos três interruptores compondo um ramo de saída. Nestes, esta operação é realizada através de três contadores de 16 bits. Cada contador, onde são carregados os tempos t_K ou t_L ou t_M, possui uma saída conectada a um interruptor. Eles operam no modo chamado "geração de pulso monoestável". Dos coprocessadores são utilizados, como saída, os pinos EVn e, como entrada, os pinos INTn (onde n pode assumir os valores 1,2 ou 3, representado cada um dos contadores).

Cada contador possui dois registros, o registro de *latch* (RL*n*) e o do contador (RC*n*) (onde *n* também representa cada um dos contadores). No modo de operação usado, a partir do momento em que é habilitada a contagem, o valor carregado no RL*n* é transferido para o RC*n*. Este, então, sofre decremento até a mudança de 0000*h* para FFFF*h*, gerando um sinal interno de estouro de faixa (*overflow*). Neste ponto o valor no RL*n* é novamente transferido para o RC*n*, continuando o decremento, e assim sucessivamente. O conteúdo do RL*n* permanece inalterado até o carregamento de um novo valor. Mas, se na entrada INT*n* do respectivo contador houver uma transição de um para zero (*edge* negativo), o valor do RL*n* é imediatamente transferido para o RC*n*, e, até a ocorrência do sinal de estouro de faixa, será gerado um pulso na respectiva saída EV*n* (figura 4.10). Este pulso tem a duração definida pelo tempos t_K ou t_L ou t_M, enviados aos RL*n*.

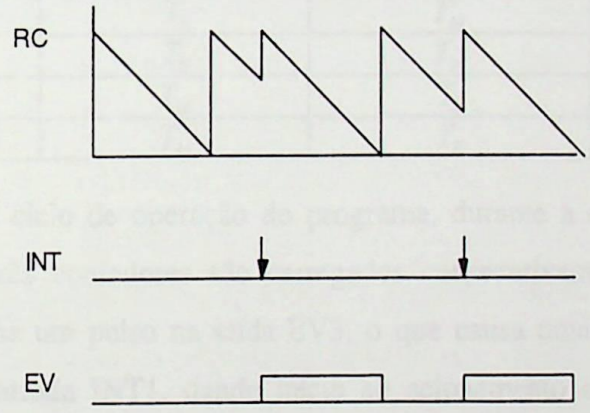


Fig. 4.10 Modo de operação dos contadores "geração de pulso monoestável".

Para garantir o acionamento de apenas um interruptor por vez, a saída Ev*n* de um contador é também ligada a entrada INT*n* do próximo contador (figura 4.11). Isto faz com que o fim do pulso (*edge* negativo) de um interruptor acione o seguinte; os contadores são ligados em anel.

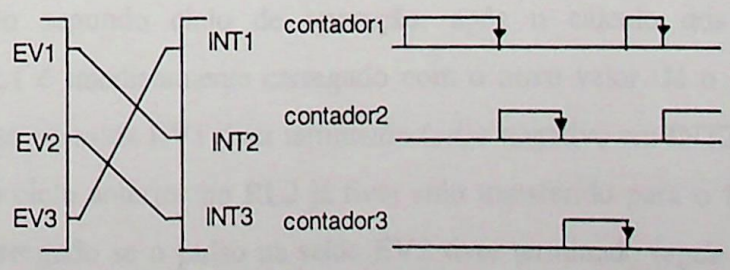


Fig. 4.11 Ligação em anel dos contadores.

A ordem de disparo dos contadores dentro do período T_S é sempre a mesma, contador_1 → contador_2 → contador_3. O que varia é a ordem dos tempos t_K, t_L e t_M dentro do período, pois em determinado momento o contador_1 é carregado com t_K, em outro com t_L e em um outro com t_M. O mesmo ocorre com os contadores 2 e 3. Por isso são usadas seis rotinas (*contn*) de acordo com a tabela 4.5, para fazer o carregamento dos contadores. Cada rotina é selecionada de acordo com o tempo (t_K ou t_L ou t_M) que deverá ser carregado nos contadores.

Tabela 4.5 Seleção das rotinas para carregamento dos contadores com os tempos de acionamento dos interruptores.

.	contador_1	contador_2	contador_3
cont1	T_L	T_K	T_M
cont2	T_K	T_L	T_M
cont3	T_L	T_M	T_K
cont4	T_K	T_M	T_L
cont5	T_M	T_L	T_K
cont6	T_M	T_K	T_L

No primeiro ciclo de operação do programa, durante a execução de uma das rotinas *contn*, os três contadores são carregados consecutivamente e em seguida é gerado por programa um pulso na saída EV3, o que causa uma transição de positivo para negativo na entrada INT1, dando início ao acionamento dos interruptores pelo disparo do interruptor ligado ao contador_1. A seguir o ciclo de operação é reiniciado com a execução da etapa de obtenção dos valores da tensão de saída desejada v_o e das tensões de entrada v_A, v_B e v_C. Como o fluxo do programa pode seguir por caminhos diferentes, sendo uns mais curtos do que outros, neste primeiro ciclo um dos contadores é utilizado como cronômetro, com a finalidade de garantir o início simultâneo do disparo dos interruptores dos três ramos de saída

A partir do segundo ciclo de operação, após o cálculo dos tempos de acionamento, o RL1 é imediatamente carregado com o novo valor. Já o RL2 só será carregado se o pulso na saída EV1 tiver terminado (*edge* negativo em INT2), ou seja, o valor carregado no ciclo anterior no RL2 já tiver sido transferido para o RC2. O RL3 também só será carregado se o pulso na saída EV2 tiver terminado (*egde* negativo em

INT3). O término do pulso em EV3 sinaliza o fim do ciclo de operação (completou-se os 500 μs).

No início de cada ciclo de operação é informado ao programa se desejarmos continuar ou parar a conversão. Uma entrada digital lê esta informação.

Observamos que a geração dos sinais de acionamento pelos coprocessadores ocorre simultaneamente a execução das seguintes tarefas: aquisição das tensões v_o, v_A, v_B e v_C, determinação das tensões v_K, v_L e v_M, cálculo dos tempos t_K, t_L e t_M e carregamento dos contadores; portanto ocorre paralelismo nestes procedimentos..

Na execução desta estratégia é necessário que os contadores 1, 2 e 3 estejam conectados em anel, de modo que o sinal proveniente do primeiro dispare o segundo; este por sua vez dispare o terceiro e este último recomece o ciclo, fazendo redisparar o primeiro contador e o programa de controle. A estratégia adotada permitiu implantar o método de controle, isto é o AE, sem recorrer à nenhum meio de sincronismo entre a evolução no tempo deste último e dos sinais presentes na entrada. Além disso, ela possibilitou também uma operação em tempo real, na medida em que o procedimento para o cálculo dos tempos de acionamento para o presente intervalo (n)T_S leva em conta as mudanças efetuadas sobre as referências de entrada e de saída durante o intervalo precedente (n-1)T_S.

4.4 Controle dos parâmetros do sistema (freqüência e amplitude das tensões de saída)

Para a realização do controle dos parâmetros do sistema foram utilizadas rotinas escritas em linguagem *Forth*.

A) Freqüência

Como visto anteriormente, entre um valor e outro da tabela usada para obtenção do valor do módulo de v_o, existe uma diferença de $0{,}352^0$. E como a amostragem ocorre a cada 500 μs, obtemos:

360^0 ---------- T

x ---------- 0,5 ms

indicando a cada quantos graus uma senóide de período T será amostrada. Por exemplo, considerando-se uma senóide de 60 Hz, o seu período T será de 16,67 ms. Assim:

360^0 ---------- 16,67 ms $\Rightarrow x \cong 10,80^0$

x ---------- 0,5 ms

Por isso, a cada amostragem será lido um valor defasado do anterior $10,80^0$. Logo para sabermos qual o valor da variável *passo* (indicando o valor do passo de amostragem da tabela), devemos dividir o resultado por $0,352^0$.

$$passo = \frac{x}{0,352^0} \tag{23}$$

Ou isolando o valor de x na regra de três anterior, com o tempo sendo dado em segundos e f a freqüência da senóide:

$$x = \frac{360^0.5.10^{-4}}{T} = 360^0.5.10^{-4}.f = 0,18.f \tag{24}$$

temos:

$$passo = \frac{0,18.f}{0,352} = 0,512.f \tag{25}$$

e aproximando-se o resultado:

$$passo \cong 0,5.f = \frac{f}{2} \tag{26}$$

No nosso exemplo temos que:

$$passo = 0,5.60 = 30$$

ou seja, a tabela deverá ser amostrada a cada 30 posições.

Logo, para obtermos o valor do *passo*, basta fornecermos o valor da freqüência da tensão de saída desejada, e então a dividirmos por 2.

B) Amplitude

Para obtermos na saída uma tensão com uma amplitude (ganho de tensão) definida pelo usuário, devemos multiplicar o valor lido da tabela para geração de v_o por

um fator redutor. Sendo 5 voltas, correspondendo a FF*h*, a maior tensão representada pela tabela, temos que o fator redutor *y* é igual a:

$$y = \frac{a}{5} \tag{27}$$

onde *a* representa a amplitude máxima desejada na saída. Por exemplo, se desejarmos uma amplitude máxima de 2 volts, o fator redutor será de:

$$y = \frac{2}{5} = 0,4$$

e todos os valores lidos da tabela deverão ser multiplicados por 0,4.

Colocando os valores em porcentagem, ficamos com:

$$y = \frac{a}{100} \tag{28}$$

e a neste caso sempre sendo dada em porcentagem em relação à 5 V.

C) Implementação do controle da freqüência e da amplitude

Os valores de referência dos parâmetros são obtidos através do teclado do *PC*. A freqüência é fornecida em Hertz e a amplitude em porcentagem em relação à amplitude máxima da tensão de entrada. Todas as operações são executadas por instruções *Forth*.

O valor da freqüência é transmitido ao controlador, onde ocorre a divisão por dois e então o resultado é enviado, através da instrução *>cp*, aos coprocessadores.

Com relação a amplitude, devido ao fato dos coprocesadores trabalharem apenas com valores inteiros, para contornarmos os valores fracionários resultantes no cálculo de *y*, a equação (28) é rescrita da seguinte forma:

$$y = (\frac{a}{100}.256).\frac{1}{256} \tag{29}$$

A parte entre parênteses da equação é realizada pelo compilador *Forth* e a divisão por 256 pelo programa em *Assembly*, executado pelos coprocessadores.

O valor *a* é transmitido ao controlador, onde ocorre a divisão por 100, sendo o resultado multiplicado por 256. Os cálculos até aqui são feitos considerando os números

em notação ponto flutuante. Antes de ser enviado aos coprocessadores através da instrução *>cp'*, o valor de *y* é convertido para notação inteira de 8 bits.

Nos coprocessadores, o valor enviado de *y* é armazenado numa variável. Toda vez que um novo valor é obtido da tabela, este é multiplicado por *y*, gerando um número de 16 bits, e a seguir dividido por 256, ou seja, é desprezado o byte menos significativo do produto. Isto gera o valor desejado da amplitude.

Deve ser ressaltado que foi escolhido o número 256 na equação (27) apenas por facilidade de cálculo. Ao se dividir *a* por 100 obtemos um número fracionário menor do que 1. Multiplicando o resultado por 256 tem-se como resultado um número fracionário maior do que 1. Deste número é desprezada a parte fracionária e enviada aos coprocessadores a parte inteira. Mas para compensar a multiplicação por 256, nos coprocessadores deve ser feita a divisão por 256, que simplesmente é realizada desprezando-se os byte menos significativos da multiplicação de *y* pelo valor lido da tabela.

Para o usuário todas estas operações são reunidas pelos seguintes comandos mostrados na tabela 4.6.

Tabela 4.6 Comandos para mudança das referências de freqüência e/ou amplitude das tensões de entrada.

Comando	Função
f freq	muda a freqüência das tensões de saída para a freqüência *f* (em Hz)
a ampl	muda a amplitude máxima das tensões de saída para a amplitude *a* (em porcentagem em relação a 5 V)
a f seno	muda tanto a freqüência quanto a amplitude

CAPÍTULO 5

RESULTADOS EXPERIMENTAIS

Neste capítulo são mostrados os resultados experimentais gerados com a utilização do sistema de controle proposto, visando a obtenção do controle, em tempo real, dos conversores diretos de freqüência.

Os resultados apresentados foram conseguidos com a utilização de um conversor direto de freqüência reduzido, composto por nove chaves analógicas-digitais (HS-201). Os interruptores trabalham com uma freqüência de comutação de 2000 Hz, correspondendo a um T_s de 500 µs. A freqüência das tensões de entrada é aquela da rede, ou seja, 60 Hz. O conversor alimentava uma carga resistiva trifásica em torno de 10 kΩ. Em cada ramo de saída do conversor, em paralelo à carga, foram usados filtros passa-baixas para se conseguir a visualização do aspecto das tensões (figura 5.1).

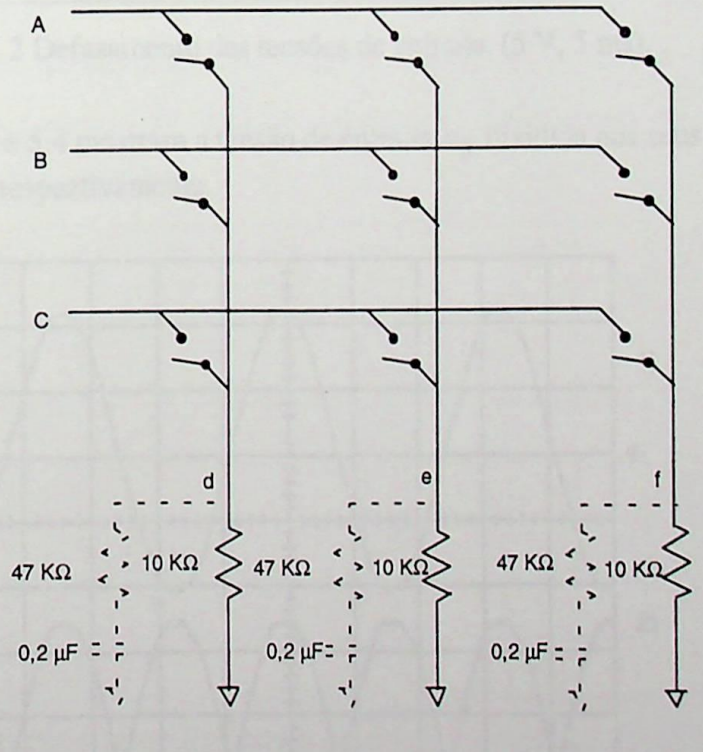


Fig. 5. 1 Carga utilizada para o conversor.

A figura 5.2 apresenta as formas de onda na entrada e na saída de um dos circuitos de defasamento das tensões de entrada, cujo valor é utilizado pelos coprocessadores. Verifica-se que a tensão na saída está adiantada em torno de 10 graus elétricos em relação a tensão na entrada. Como o ciclo de operação do programa é de 500 μs (10,8 graus elétricos), entre a leitura das tensões de entrada e o acionamento dos interruptores ocorre um atraso da ordem deste tempo. O defasamento visa corrigir este atraso.

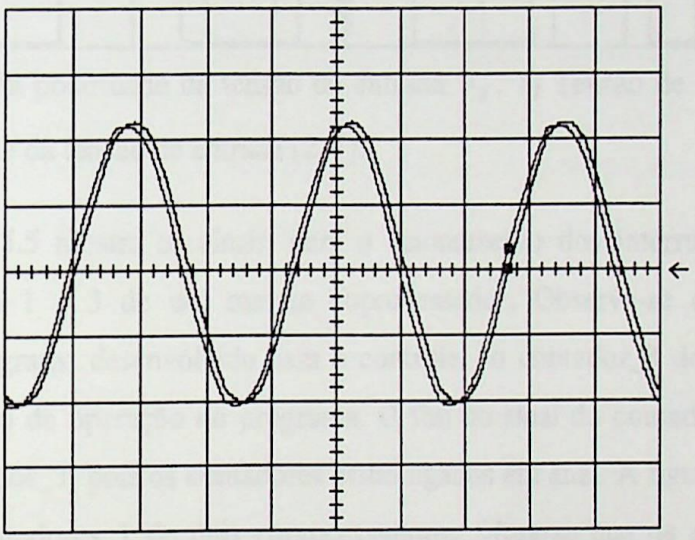


Fig. 5. 2 Defasamento das tensões de entrada. (5 V, 5 ms).

As figuras 5.3 e 5.4 mostram a tensão de entrada v_B dividida nos seus valores de módulo e polaridade, respectivamente.

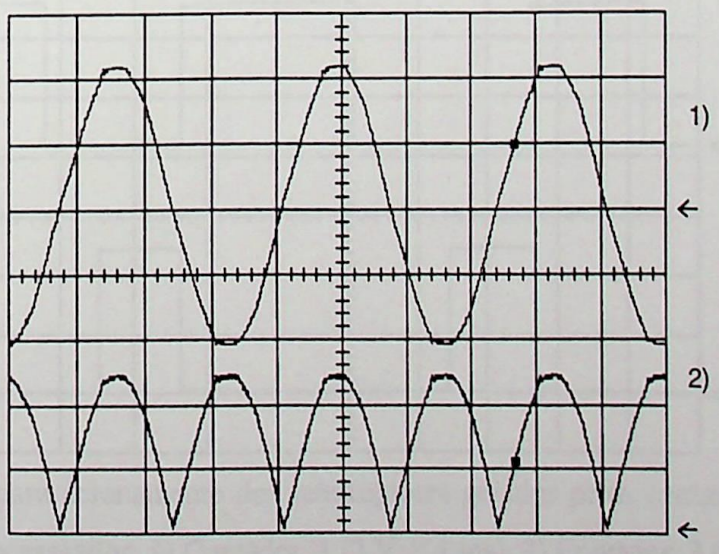


Fig. 5. 3 Valor do módulo da tensão de entrada v_B. 1) Tensão de entrada (5 V, 5 ms). 2) Tensão de entrada retificada (2 V).

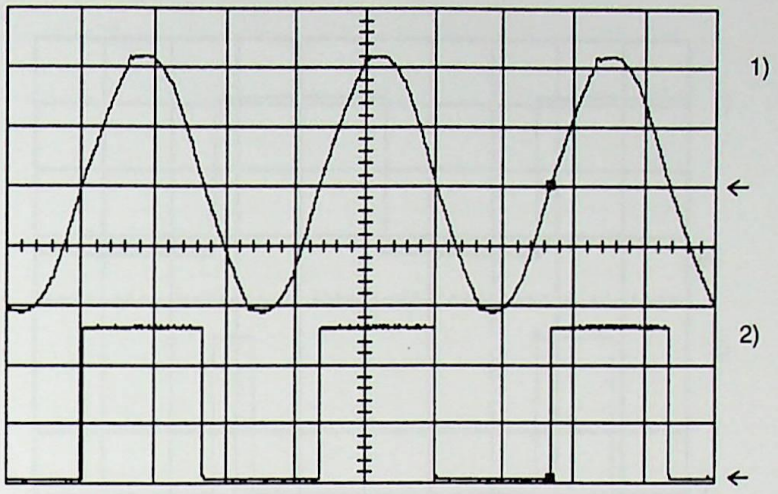


Fig. 5. 4 Valor da polaridade da tensão de entrada v_B. 1) Tensão de entrada (5 V, 5 ms). 2) Polaridade da tensão de entrada (2 V).

A figura 5.5 mostra os sinais, para o acionamento dos interruptores, gerados pelos contadores 1 e 3 de um mesmo coprocessador. Observa-se que, segundo a descrição do programa desenvolvido para o controle, o contador_3 do coprocessador reinicializa o ciclo de operação do programa. O fim do sinal do contador_3 provoca o disparo do contador_1, pois os contadores estão ligados em anel. A figura 5.6 apresenta os sinais dos contadores_1 de dois coprocessadores. Nota-se que os coprocessadores começam simultaneamente um novo ciclo, indicado pelo inicio do sinal do contador_1. Em ambas figuras, pelo início do sinal do contador_1, comprova-se que o ciclo de operação do programa é de 500 μs.

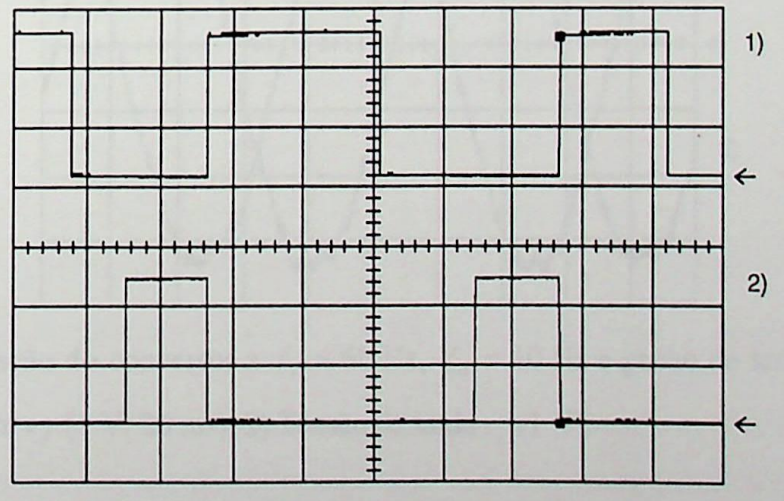


Fig. 5. 5 Sinais para acionamento dos interruptores gerados pelos contadores 1 e 3 de um mesmo coprocesssador. 1) Contador_1 (2 V, 0,1 ms). 2) Contador_3 (2 V).

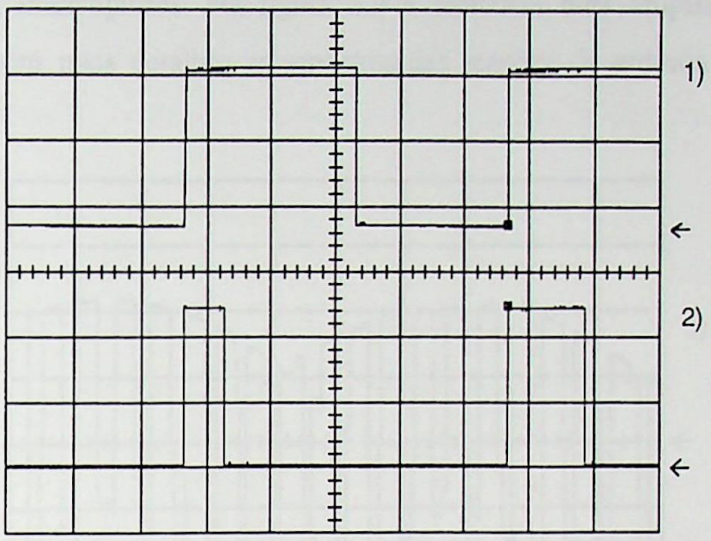


Fig. 5. 6 Sinais para acionamento dos interruptores gerados pelos contadores 1 de dois coprocesssadores. 1) Contador_1 do coprocessador_1 (2 V, 0,1 ms). 2) Contador_1 do coprocessador_2 (2 V).

Na figura 5.7 são visualizadas as tensões filtradas em dois dos três ramos de saída, onde é gerado um sistema trifásico balanceado.

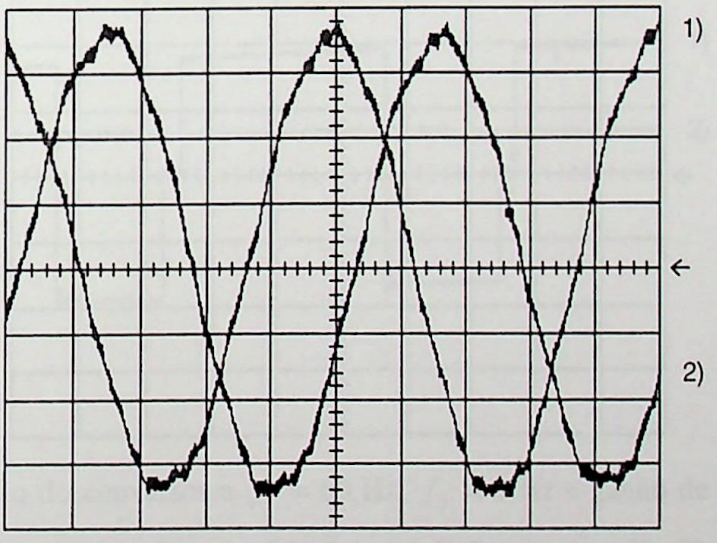


Fig. 5. 7 Operação do conversor a f_i = 60 Hz, f_0 = 10 Hz e ganho de tensão igual a 40%. 1) Tensão v_d (1 V, 20 ms). 2) Tensão de saída v_e (1 V).

Na figura 5.8 temos duas formas de onda, a tensão não filtrada e a filtrada de um ramo de saída, com f_0 = 0 Hz e ganho de tensão igual a 30%. Observa-se no sinal não filtrado a forma correspondente as três tensões de entrada recortadas, quando do

acionamento dos interruptores. Na figura 5.9 é realizada uma ampliação da figura anterior. Vê-se com mais detalhes as amostras das tensões de entrada transferidas à carga.

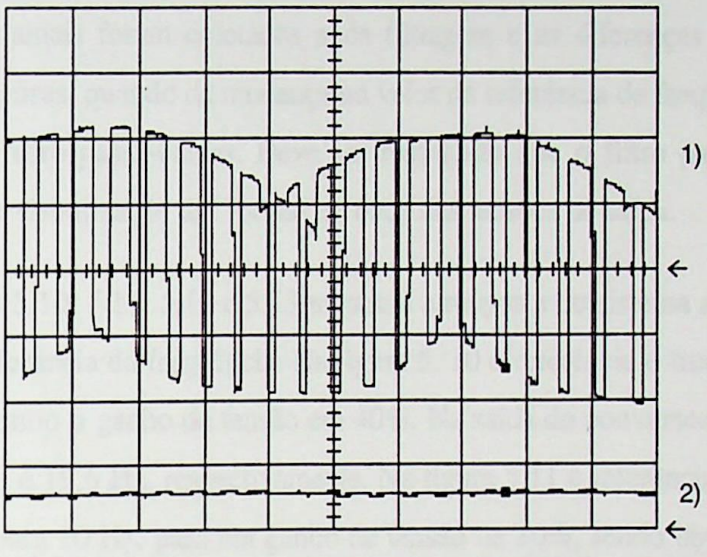


Fig. 5. 8 Operação do conversor a f_i = 60 Hz, f_0 = 0 Hz e ganho de tensão igual a 30%. 1) Tensão de saída não filtrada (5 V, 1 ms). 2) Tensão de saída filtrada (5 V).

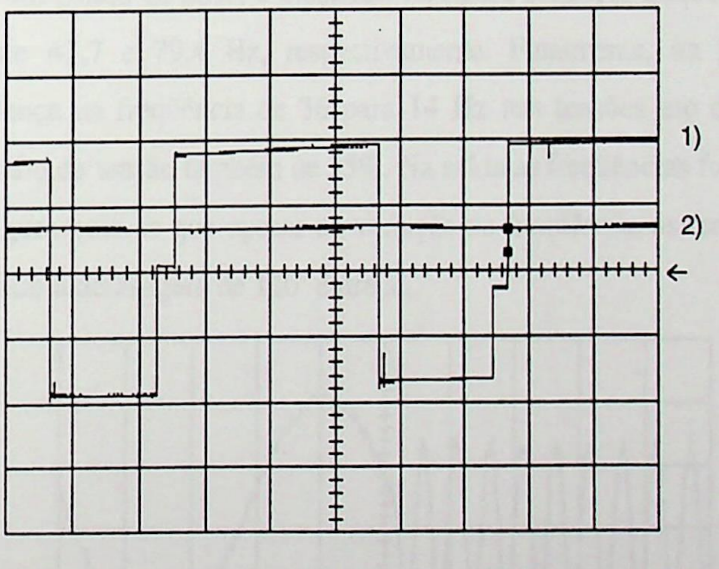


Fig. 5. 9 Operação do conversor a f_i = 60 Hz, f_0 = 0 Hz e ganho de tensão igual a 30%. 1) Tensão de saída não filtrada (5 V, 0,1 ms). 2) Tensão de saída filtrada (5 V).

As figuras seguintes mostram a resposta do sistema em virtude de uma modificação no valor da referência da freqüência e/ou da amplitude das tensões de saída. Nestas figuras também é visualizado o sinal de referência para mudança de parâmetros, a linha de leitura/escrita (R/W) do mestre. Como mostrado no capítulo 3, os coprocessadores são vistos como endereços de memória para o mestre. Logo um valor

baixo (zero volts) na linha R/W indica escrita do mestre na interface de barramento dos coprocessadores, em outras palavras, a transferência de novos valores de referência. Isto determina o momento exato da solicitação de modificação das tensões de saída.

Todos os sinais foram coletados após filtragem e as diferenças de amplitudes observadas nas figuras, quando da mudança no valor da referência de freqüência, devem-se a atuação do filtro passa-baixas. Deve ser ressaltado que o filtro passa-baixas tem como finalidade a visualização das formas de onda das tensões de saída.

As figuras 5.10, 5.11, 5.12 e 5.13 mostram a resposta do sistema a uma mudança nos valores de referência da freqüência. Na figura 5. 10 a referência é trocada de 2 para 20 Hz, permanecendo o ganho de tensão em 40%. Na saída do conversor foram obtidas freqüências de 1,9 e 19,6 Hz, respectivamente. Na figura 5.11 a referência de freqüência é mudada de 40 para 10 Hz, para um ganho de tensão de 30%, sendo obtidas, na saída, freqüências de 39,7 e 9,9 Hz, conforme as referências. Um aumento na freqüência de 44 para 80 Hz, para um ganho de 35%, é mostrado na figura 5.12. As freqüências medidas na saída foram de 43,7 e 79,4 Hz, respectivamente. Finalmente, na figura 5.13 é observada a mudança na freqüência de 36 para 14 Hz nas tensões em dois ramos de saída, para um ganho de tensão também de 35%. Na saída as freqüências foram de 35,6 e 13,7 Hz. Nesta figura nota-se que apesar da variação na freqüência, as tensões na saída continuam mantendo a defasagem de 120° entre si.

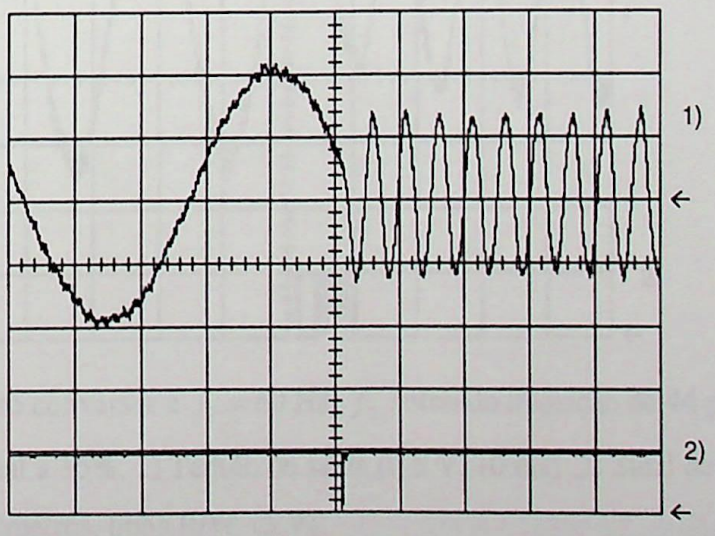


Fig. 5. 10 Operação do conversor a f_i = 60 Hz, f_0 sofrendo transição de 2 para 20 Hz e ganho de tensão igual a 40%. 1) Tensão de saída (2 V, 0,1 s). 2) Sinal de referência para mudança de parâmetros, linha R/W (5 V).

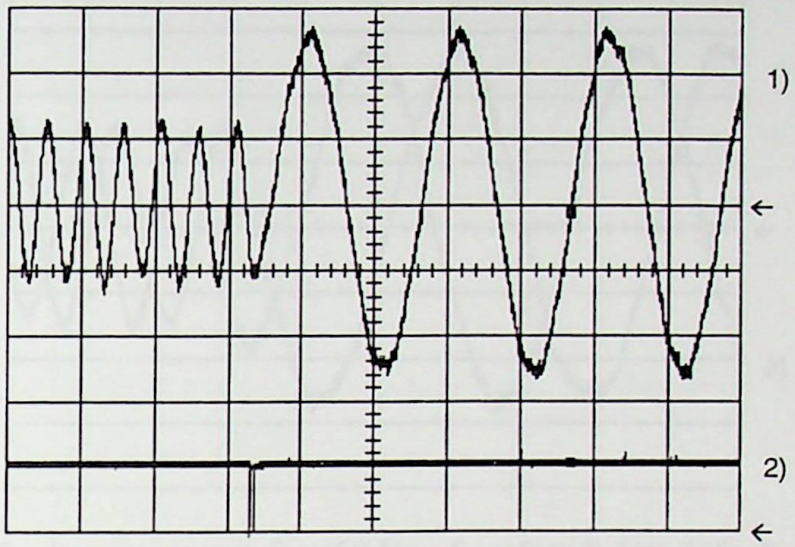


Fig. 5. 11 Operação do conversor a f_i = 60 Hz, f_0 sofrendo transição de 40 para 10 Hz e ganho de tensão igual a 30%. 1) Tensão de saída (1 V, 50 ms). 2) Sinal de referência para mudança de parâmetros, linha R/W (5 V).

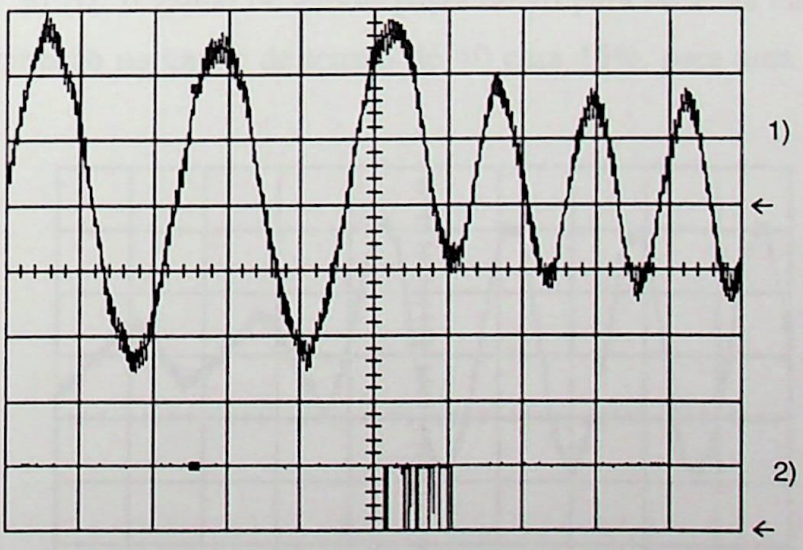


Fig. 5. 12 Operação do conversor a f_i = 60 Hz, f_0 sofrendo transição de 44 para 80 Hz e ganho de tensão igual a 35%. 1) Tensão de saída (0,5 V, 10 ms). 2) Sinal de referência para mudança de parâmetros, linha R/W (5 V).

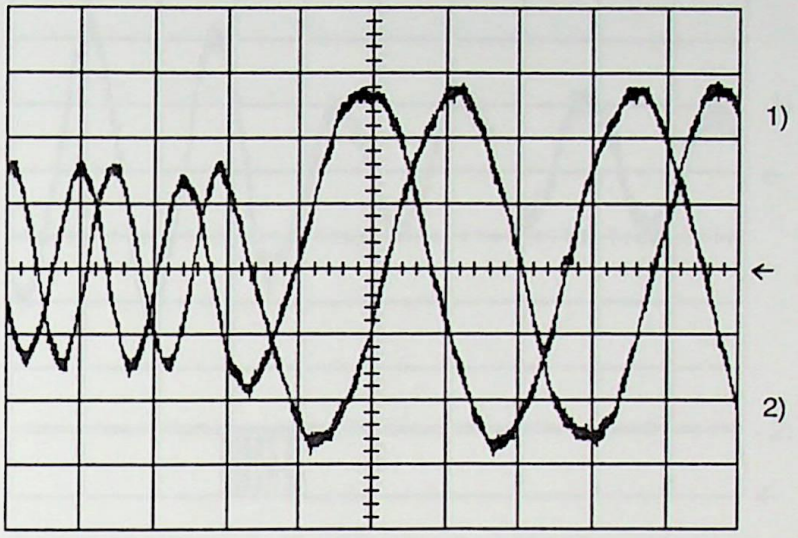


Fig. 5. 13 Operação do conversor a f_i = 60 Hz, f_0 sofrendo transição de 36 para 14 Hz e ganho de tensão igual a 35%. 1) Tensão de saída v_d (1 V, 20 ms). 2) Tensão de saída v_e (1 V).

As figuras 5.14 e 5.15 apresentam a resposta do sistema a uma mudança nos valores de referência da amplitude (ou ganho de tensão). Na figura 5.14, para uma freqüência fixa de 30 Hz, o ganho de tensão passa de 10 para 40%. Já na figura 5.15 é visualizada uma variação no ganho de tensão de 40 para 15%, para uma freqüência de 60 Hz.

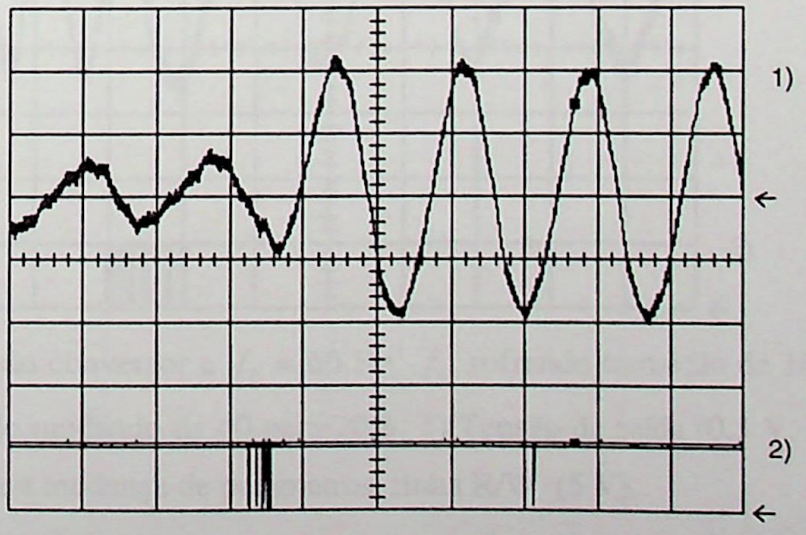


Fig. 5. 14 Operação do conversor a f_i = 60 Hz, f_0 = 30 Hz e ganho de tensão sofrendo transição de 10 para 40%. 1) Tensão de saída (1 V, 20 ms). 2) Sinal de referência para mudança de parâmetros, linha R/W (5 V).

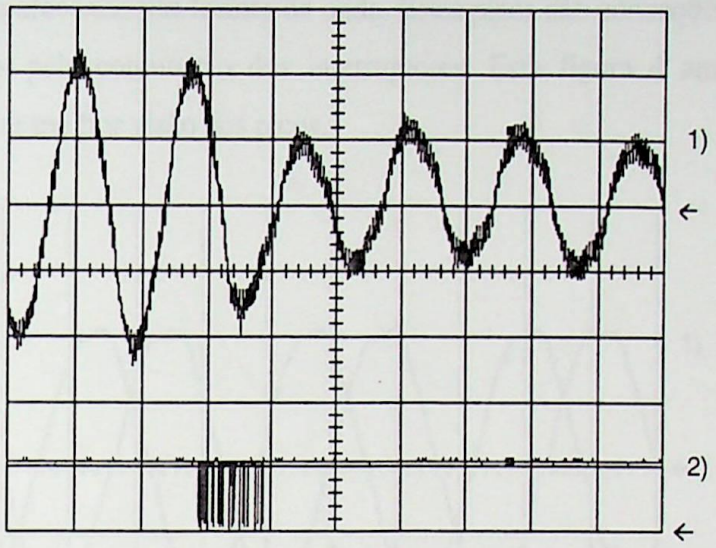


Fig. 5. 15 Operação do conversor a f_i = 60 Hz, f_0 = 60 Hz e ganho de tensão sofrendo transição de 40 para 15%. 1) Tensão de saída (0,5 V, 10 ms). 2) Sinal de referência para mudança de parâmetros, linha R/W (5 V).

Por sua vez, a figura 5.16 mostra uma modificação simultânea dos parâmetros, tanto da freqüência, de 100 para 50 Hz, como do ganho de tensão, de 40 para 20%.

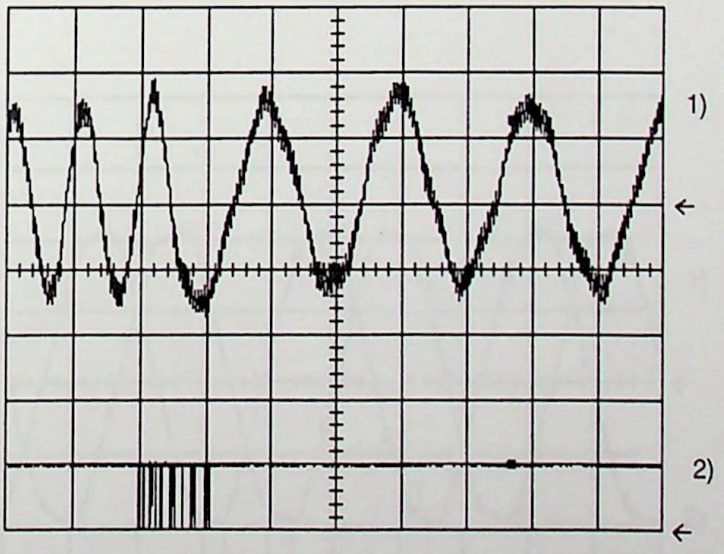


Fig. 5. 16 Operação do conversor a f_i = 60 Hz, f_0 sofrendo transição de 100 para 50 Hz e ganho de tensão mudando de 40 para 20%. 1) Tensão de saída (0,5 V, 10 ms). 2) Sinal de referência para mudança de parâmetros, linha R/W (5 V).

Na figura 5.17 são observadas duas das tensões de fase na entrada do conversor, com todos os interruptores abertos. Na figura 5.18 são apresentadas as mesmas duas tensões de entrada, mas com o conversor em funcionamento. Nesta figura notam-se

picos de tensão aparecendo nas formas de onda. Estes picos são conseqüência de curto-circuitos causados pela comutação dos interruptores. Esta figura é ampliada (figura 5.19), tendo-se uma melhor visão dos picos.

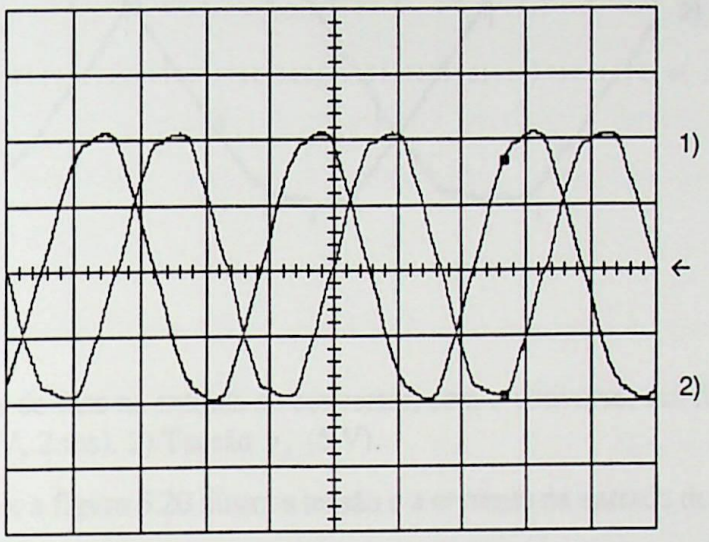


Fig. 5. 17 Tensões de fase na entrada do conversor, com todos os interruptores abertos. 1) Tensão v_A (5 V, 5 ms). 2) Tensão v_C (5 V).

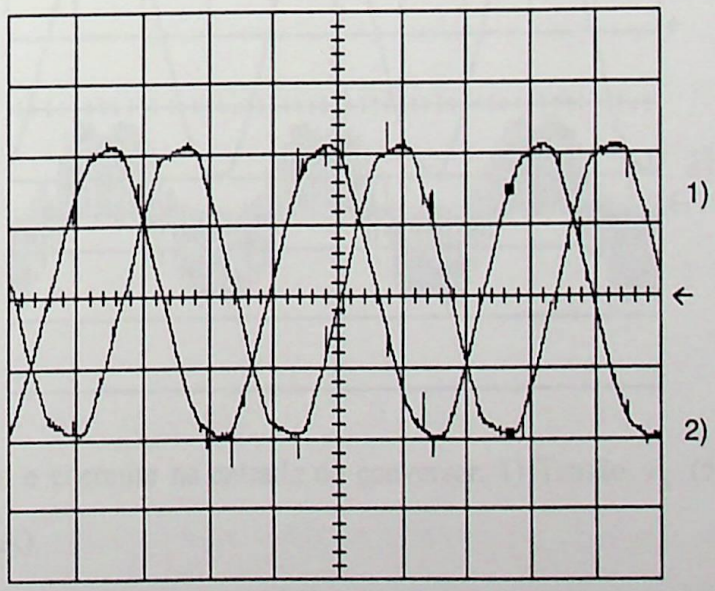


Fig. 5. 18 Tensões de fase na entrada do conversor, com o conversor em funcionamento. 1) Tensão v_A (5 V, 5 ms). 2) Tensão v_C (5 V).

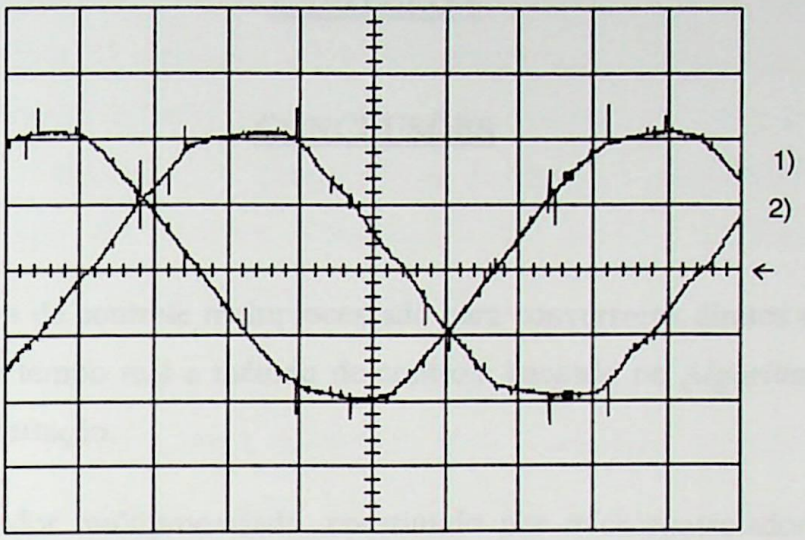


Fig. 5. 19 Tensões de fase na entrada do conversor, com o conversor em funcionamento. 1) Tensão v_C (5 V, 2 ms). 2) Tensão v_A (5 V).

Finalizando, a figura 5.20 ilustra a tensão e a corrente na entrada do conversor. Na forma de onda da corrente observa-se claramente a presença do quinto harmônico.

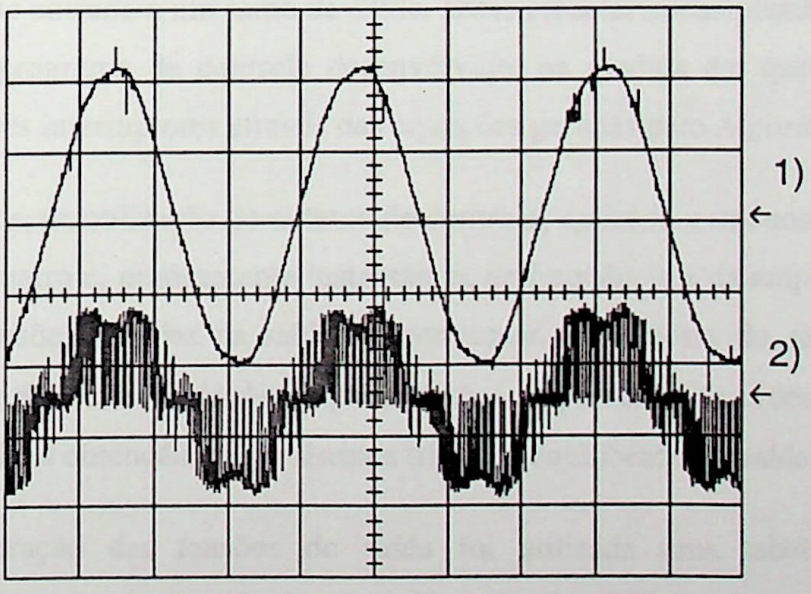


Fig. 5. 10 Tensão e corrente na entrada do conversor. 1) Tensão v_A (5 V, 5 ms). 2) Corrente I_A (5 mA).

Pela observação das figuras anteriores nota-se a ampla faixa de variação de freqüência obtida, tanto para valores de referência abaixo como acima da freqüência de entrada. Mas, principalmente, comprova-se a obtenção do controle, praticamente instantâneo, da freqüência e da amplitude das tensões de saída.

CAPÍTULO 6

CONCLUSÕES

Um sistema de controle multiprocessado para conversores diretos de freqüência, com operação em tempo real e método de controle baseado no *Algoritmo Escalar* foi descrito nesta dissertação.

O controlador multiprocessado, constituído por microcontroladores de 8 bits, respondeu pelo controle do conversor. Os microcontroladores, através de subsistemas internos para funções periféricas genéricas, tais como ADCs, DACs e contadores, provocaram uma redução no *hardware* necessário a implantação do sistema proposto. A cada microcontrolador coube o controle do conjunto de três interruptores, conectando uma das tensões de entrada a um ramo de saída. Estas características possibilitaram uma simplificação no programa de controle desenvolvido, na medida em que o programa gerencia apenas três interruptores através das equações geradas pelo *Algoritmo Escalar*.

Verificamos, na utilização do sistema de controle, aplicado a um conversor direto de freqüência, o controle, praticamente instantâneo, da freqüência e da amplitude (ganho de tensão) das tensões geradas na saída do conversor. A resposta do sistema a uma mudança no valor das referências de freqüência e/ou amplitude é da ordem de 500 μs. Constatou-se, ainda, a obtenção de um sistema trifásico equilibrado na saída.

Para a geração das tensões de saída foi utilizada uma tabela de busca, possibilitando a obtenção de freqüências de saída entre 2 e 250 Hz. Esta tabela, sendo composta por valores de oito bits, resulta numa passo de freqüência de 2 Hz. Para se diminuir este passo, utiliza-se uma tabela com valores representados por um maior número de bits, resultando numa menor resolução. Por exemplo, para 16 bits, o passo será de 0,008 Hz. Porém, para a manipulação de valores de 16 bits, o ideal é o emprego de microcontroladores também de 16 bits, principalmente devido a realização de várias multiplicações.

O sistema de controle não requer, necessariamente, um sistema trifásico equilibrado na entrada do conversor. As tensões na entrada podem estar, por exemplo, defasadas de um ângulo diferente de 120 graus uma da outra, ou a freqüência pode ser diferente de 60 Hz. Manipulando apenas valores escalares, o *Algoritmo Escalar* toma conhecimento apenas dos valores instantâneos das tensões de entrada, não se preocupando com as posições relativas das mesmas.

O sistema desenvolvido considera um sistema trifásico equilibrado na entrada. Por isso o quarto coprocessador é empregado para a obtenção do valor de pico das tensões de entrada, utilizado pelo termo $1{,}5V_i^2$ das equações para cálculo dos tempos de acionamento dos interruptores. Termo este válido apenas para sistemas equilibrados. Em sistemas desequilibrados, a solução é usar o termo geral $v_A^2 + v_B^2 + v_C^2$. Neste caso o quarto coprocessador realizaria a aquisição não do valor de pico, mas sim dos valores instantâneos das tensões de entrada e as subseqüentes operações para cálculo do termo geral.

Atualmente os conversores indiretos de freqüência (compostos por um retificador mais um inversor) prevalecem sobre os conversores diretos. Os conversores diretos, possuem vantagens sobre os indiretos, tais como menor tamanho, não necessitarem de qualquer *link* ca ou cc e permitirem a regeneração da potência. Mas eles têm pouca utilização, principalmente devido a problemas relacionados à confecção, à sincronização e à proteção das chaves bidirecionais. Basicamente os conversores diretos atuais são todos protótipos. Porém o desenvolvimento de novos semicondutores, especialmente dos MCT's (MOS Controlled Thyiristors), que são capazes de operarem como chaves bidirecionais, viabilizará a utilização dos conversores diretos em uma larga faixa de aplicações e a fabricação de modelos comerciais.

Sendo o primeiro trabalho desenvolvido na área de conversores diretos de freqüência na EFEI, como o objetivo da dissertação centrou-se no desenvolvimento do sistema de controle e devido a inexistência de um conversor de potência, empregaram-se chaves analógicas como interruptores.

Apesar do sistema de controle funcionar dentro do esperado, houve um problema na operação do conversor: a presença nas correntes de entrada do conversor de uma componente harmônica de 5ª ordem (300 Hz) elevada. A causa desta distorção na corrente merece um estudo mais aprofundado.

Como sugestão para trabalhos futuros citamos uma análise mais completa do conteúdo harmônico gerado pelo conversor, a possibilidade de operá-lo com ganhos de tensão entre 50 e 87%, o seu funcionamento atuando sobre cargas indutivas e a construção de um conversor direto de freqüência de algumas centenas de watts.

BIBLIOGRAFIA

[1] Bose, B. K., Introduction to power electronics. In: Bose, B. K., Modern Power Electronics : Evolution, Technology, and Applications. New York: IEEE Press, 1992, pp. 3 - 39.

[2] Gyugyi, L. e Pelly, B., Static Power Frequency Changers. New York: Wiley, 1976.

[3] Venturini, M., A new sine wave in, sine wave out, conversion technique eliminates reactive elements. In: Proceedings Powercon 7, San Diego, 1980, pp. E3.1 - E3.15.

[4] Venturini, M. e Alesina, A., The generalized transformer: a new bidirectional sinusoidal waveform sinusoidal frequency converter with continuously adjustable input power factor. In: PESC Conf. Rec., 1980, pp. 242 - 252.

[5] Venturini, M. e Alesina, A., Solid state power conversion: a Fourier analysis approach to generalized transformer synthesis. IEEE Trans. on Circuits Syst., vol. CAS-28, no 4, pp.319 - 330, 1981.

[6] Maytum, M. J. e Colman, D., The implementation and future potential of the Venturini Converter. In: Proceedings of Drives, Motors and Controls, 1983, pp. 108 - 117.

[7] Ma, X., High-performance PWM frequency changers. IEEE Trans. on Ind. Appl., vol. 22, pp. 267 - 280, Mar./Apr. 1986.

[8] Ziogas, P. D., Khan, S. I. e Rashid, M. H., Analysis and design of forced commutated cycloconverter structures with improved transfer characteristics. IEEE Trans. on Ind. Electron., vol. 33, pp. 271 - 280, Aug. 1986.

[9] Venturini, M. e Alesina, A., Analysis and design of optimum-amplitude nine-switch direct ac-ac converters. IEEE Trans. on Power Electron., vol. 4, no 1, Jan. 1989.

[10] Tenti, P., Malesani, L. e Rosseto, L., Optimum control of n-input k-output matrix converters. IEEE Trans. on Power Electron., vol. 7, no 7, pp. 707 - 713, Oct 1992.

[11] Ooi, B. T., Dai, S. Z. E Wang, X., Solid-state series characteristics of pratical step-up nine-switches matrix converter. IEE Proc.-B, vol. 140, no 2, pp. 139 - 146, Mar. 1993 .

[12] Kazerani, M. e Ooi, B.-T., Feasibility of both vector control and displacement factor correction by voltage source type ac-ac matrix converter. IEEE Trans. on Ind. Electron., vol. 42, no 5, Oct. 1995.

[13] Holmes, D. G. e Lipo, A., Implementation of a controlled rectifier using ac-ac matrix converter theory. IEEE Trans. on Power Electron., vol. 7, no 1, pp. 240 - 250, Jan. 1992.

[14] Roy, G., Duguay, L., Manias e S., April, G.-E., Asynchronous operation of cycloconverter with improved gain by employing a scalar control algorithm. In: Proceedings IEEE/IAS 22nd Annual Meeting, Atlanta, 1987, pp. 891 - 898.

[15] Roy, G e April, G.-E., Direct frequency changer operation under a new scalar control algorithm. IEEE Trans. on Power Electron., vol. 6, no.1, pp. 100 - 107, Jan. 1991.

[16] Leonard, Eric. Controleur Forth multiprocesseur pour les aplications en temps réel. Montreal - Canadá: École Polytechnique de Montréal, 1993. (Dissertação, Mestrado em Ciências Aplicadas).

[17] Mitsubish Semiconductors, Single-Chip 16-bit Microcomputers, 1989.

[18] Mitsubish Semiconductors, Single-Chip 8-bit Microcomputers, 1989.

[19] Leonard, E., April, G.-E. e Roy, G. Forth700, compilateur Forth à chaînage direct pour le microcontrôleur M37700, relatório técnico, École Polytechnique de Montréal, 1992.

APÊNDICE I

DEDUÇÕES E CONSIDERAÇÕES

Este apêndice apresenta as deduções das equações do capítulo 2 e tece considerações sobre aspectos abordados no capítulo 4.

I.1 Deduções dos termos e equações apresentados no capítulo 2

Temos as equações (5), (6) e (8):

$$t_K + t_L + t_M = T_S \tag{5}$$

$$\frac{t_K}{t_L} = \frac{v_K}{v_L} \tag{6}$$

$$\rho_{KL} = \frac{v_K}{v_L} \tag{8}$$

Logo as equações (9) e (10) podem ser assim obtidas:

- de (6) e (8) obtemos (9):

$$t_K = \rho_{KL} t_L \tag{9}$$

- de (5) e (9) obtemos (10):

$$t_M = T_S - t_K - t_L = T_S - \rho_{KL} t_L - t_L \Rightarrow t_M = T_S - (\rho_{KL} + 1) t_L \tag{10}$$

A partir da equação (4):

$$v_0 = \frac{1}{T_S}[t_K v_k + t_L v_L + t_M v_M] \tag{4}$$

podemos determinar t_L:

$$T_S v_0 = t_K v_k + t_L v_L + t_M v_M$$

Substituindo na equação acima os valores de t_K e t_M:

$$\begin{aligned} T_S v_0 &= \rho_{KL} t_L v_K + t_L v_L + T_S v_M - (1+\rho_{KL}) t_L v_M = \\ &= T_S v_M + [\rho_{KL} v_K + v_L - (1+\rho_{KL}) v_M] t_L \Rightarrow \end{aligned}$$

$$t_L = \frac{T_S (v_0 - v_M)}{\rho_{KL} v_K + v_L - (1+\rho_{KL}) v_M} \tag{11}$$

Usando novamente o valor de ρ_{KL} na equação (1), temos:

$$t_L = \frac{T_S(v_0 - v_M)}{\frac{v_K}{v_L}v_K + v_L - v_M - \frac{v_K v_M}{v_L}}$$

Multiplicando o denominador e o numerador por v_L e agrupando os termos em v_M:

$$t_L = \frac{T_S(v_0 - v_M)v_L}{v_K^2 + v_L^2 - (v_L + v_K)v_M}$$

Adicionando e subtraindo v_M^2 ao denominador chegamos a equação (12):

$$t_L = \frac{T_S.(v_0 - v_M).v_L}{[v_K^2 + v_L^2 + v_M^2 - (v_K + v_L + v_M).v_M]} \tag{12}$$

Como num sistema trifásico balanceado $v_K + v_L + v_M = 0$, logo:

$$\frac{t_L}{T_S} = \frac{(v_0 - v_M)v_L}{v_K^2 + v_L^2 + v_M^2} = \frac{(v_0 - v_M)v_L}{1{,}5V_i^2} \tag{13}$$

$$\frac{t_K}{T_S} = \frac{(v_0 - v_M)v_K}{1{,}5V_i^2} \tag{14}$$

$$\frac{t_M}{T_S} = 1 - \frac{t_K + t_L}{T_S} \tag{15}$$

O termo $1{,}5V_i^2$ é assim deduzido, a partir de um sistema trifásico equilibrado:

$$v_A = V_i \cos(\omega_i t)$$

$$v_B = V_i \cos(\omega_i t - 120^o)$$

$$v_C = V_i \cos(\omega_i t + 120^o)$$

Fazendo a soma dos quadrados das tensões e considerando x = ωt, temos:

$$v_A^2 + v_B^2 + v_C^2 = V_i^2 \cos^2 x + V_i^2 \cos^2(x + 120^0) + V_i^2 \cos^2(x - 120^0) =$$

$$= V_i^2[\cos^2 x + \cos^2(x + 120^0) + \cos^2(x - 120^0)]$$

Usando as relações trigonométricas da soma e diferença de arcos:

$$v_A^2 + v_B^2 + v_C^2 = V_i^2[\cos^2 x + (\cos x.\cos 120^0 - sen x.sen 120^0)^2 +$$

$$+(\cos x.\cos 120^0 + sen x.sen 120^0)^2] =$$

$$= V_i^2[\cos^2 x + (-\frac{1}{2}\cos x - \frac{\sqrt{3}}{2} senx)^2 + (-\frac{1}{2}\cos x + \frac{\sqrt{3}}{2} senx)^2 =$$

$$= V_i^2[\cos^2 x + \frac{1}{4}\cos^2 x + \frac{\sqrt{3}}{2}\cos x.senx + \frac{3}{4} sen^2 x +$$

$$+ \frac{1}{4}\cos^2 x - \frac{\sqrt{3}}{2}\cos x.senx + \frac{3}{4} sen^2 x =$$

$$= V_i^2[\cos^2 x + \frac{1}{2}\cos^2 x + \frac{3}{2} sen^2 x] = V_i^2.\frac{3}{2}.(\cos^2 x + sen^2 x) \Rightarrow$$

$$v_A^2 + v_B^2 + v_C^2 = 1{,}5V_i^2$$

I.2 Passo das freqüências das tensões de saída

No capítulo 4 mostrou-se que o passo das freqüências das tensões de saída é de aproximadamente 2 Hz. A seguir mostraremos como diminuir o valor deste passo.

Observou-se que a tabela contendo os valores de um quarto de onda (90 graus) do módulo de v_0 possui 256 valores de 8 bits. Assim:

$$\frac{90^0}{256} = 0{,}352^0$$

Como a tabela é amostrada a cada 500 µs temos:

500 µs ---------- $0{,}352^0$

x µs ---------- 360^0

resultando numa freqüência mínima, a ser gerada, de aproximadamente 2 Hz, e todas as demais freqüências múltiplas desta.

Uma possível solução para diminuir este passo de freqüência é aumentar o número de bits dos valores da tabela utilizada para geração de v_0. Por exemplo, utilizando-se uma tabela composta por valores de 16 bits, temos:

$$\frac{90^0}{65536} \cong 0{,}0014^0$$

entre dois valores consecutivos da tabela.

Logo, com a tabela também sendo amostrada a cada 500 µs:

500 μs ---------- $0{,}0014^0$

x μs ---------- 360^0

resultando numa freqüência mínima de aproximadamente 0,008 Hz e com todas as demais freqüências múltiplas desta.

Para a utilização de uma tabela de 16 bits, o ideal é utilizarmos processadores de 16 bits. Mas, processadores de 8 bits também trabalham com dados de dois bytes. Os valores do módulo das tensões de entrada, lidos por ADCs de 8 bits, podem ser convertidos para notação de 16 bits. Para isto, após a conversão do valor, ele é multiplicado por $101h$, resultando num número de 16 bits.

A dificuldade para a manipulação de valores de 16 bits por processadores de 8 bits está na implementação das equações para cálculo dos tempos de acionamento dos interruptores. Para o cálculo de t_K é realizada uma multiplicação de 8 bits e outra de 16 bits. Esta última, na realidade, quatro multiplicações de 8 bits. O mesmo ocorrendo para o cálculo de t_L. Ao trabalharmos com valores de 16 bits, no cálculo de t_K devemos realizar uma multiplicação de 16 bits e outra de 32 bits, resultando em 20 multiplicações de 8 bits. Adicionando-se as operações de cálculo de t_L, totalizamos 40 multiplicações de 8 bits. Isto inviabiliza a manipulação de valores de 16 bits por um microcontrolador de 8 bits, devido ao tempo gasto nas operações e ao aumento na complexidade do programa.

Outra opção para diminuir o valor do passo da freqüência: aumentar o tempo do ciclo de operação do programa, conseqüentemente reduzindo a freqüência de comutação dos interruptores.

Com um ciclo de 1000 μs, o passo passará a 1 Hz e a freqüência de comutação dos interruptores será de 1000 Hz. Mesmo assim, é possível obter freqüências de saída em torno de 200 Hz. A mudança no ciclo de operação aumentará a resposta do sistema a eventuais modificações das referências.

APÊNDICE II

O COMPILADOR *FORTH*

Este apêndice faz uma introdução ao compilador *Forth*, denominado *Forth700,* utilizado pelo controlador e desenvolvido por Leonard [16], [19]. O nome *Forth700* é em conseqüência do fato deste ser um compilador da linguagem Forth desenvolvido para o processador M37700.

Para se acessar os recursos de programação do controlador, é utilizado um programa de comunicação, também desenvolvido por Leonard. Esta comunicação realiza-se através de uma porta serial do *IBM-PC*, ligada a uma das portas seriais do mestre. Quando da utilização do programa devem ser definidos alguns parâmetros, tais como qual porta serial do *PC* está sendo usada, qual a interrupção e qual a taxa de transmissão utilizadas.

A partir do momento em que é estabelecido o contato *PC*-controlador, o programa de comunicação passa a atuar como um editor para o compilador *Forth700,* residente em EPROM no controlador. Assim o *PC* torna-se simplesmente um terminal para o compilador, pois o programa só aceita instruções escritas em *Forth700*, instruções estas que serão executadas não pelo processador do *PC,* mas sim pelos processadores do controlador (através da tecla de função F8 é possível utilizar comandos do sistema operacional do *PC*).

Como características gerais da linguagem *Forth*, temos:

- instruções, denominadas palavras, executadas sobre dados armazenados na pilha (*stack*);
- uso da notação polonesa invertida (parâmetros primeiro, palavras depois);
- linguagem que não é totalmente interpretada nem compilada, mas sim que possui características de ambas as ações;
- conjunto de palavras específicas para operações sobre a pilha;

- possibilidade de criação de novas palavras, cujo conjunto é denominado dicionário, através do uso de outras palavras já existentes. Uma palavra nova definida pelo usuário sempre começa por dois pontos (:) e termina em ponto-e-vírgula (;):

```
: nome
        { palavras já definidas }
;
```

Além destas, o *Forth700* possui algumas características próprias:

- utilização de operandos em notação ponto flutuante e complexa;
- uma pilha para valores inteiros e outra para valores complexos e em ponto flutuante;
- palavras para execução de operações trigonométricas, logarítmicas e exponenciais;
- palavras para conversão entre dados, como de notação inteira para ponto flutuante e vice-versa;
- palavras já definidas para acesso às memórias e à interface de barramento dos coprocessadores;
- palavras criadas pelo usuário podem intercalar instruções em *Forth* e em *Assembly*. As palavras *forth* e *code* definem o início de instruções em *Forth* e *Assembly*, respectivamente:

```
: nome
        { instruções Forth }
        code
        { instruções Assembly }
        forth
        { instruções Forth }
;
```

Logicamente as instruções em *Assembly* devem ser escritas em notação polonesa inversa. Por exemplo, para carregar um valor imediato no acumulador não se escreve:

lda # n

onde *n* é a valor imediato, mas sim:

n # lda,

sendo a vírgula no final obrigatória, indicando o fim de uma instrução em *Assembly.*

A tabela a seguir mostra as palavras *Forth700* para comunicação do mestre com os coprocessadores.

Notação:

b: byte

ad: endereço absoluto de 16 bits

n,: inteiro sinalizado (16 bits)

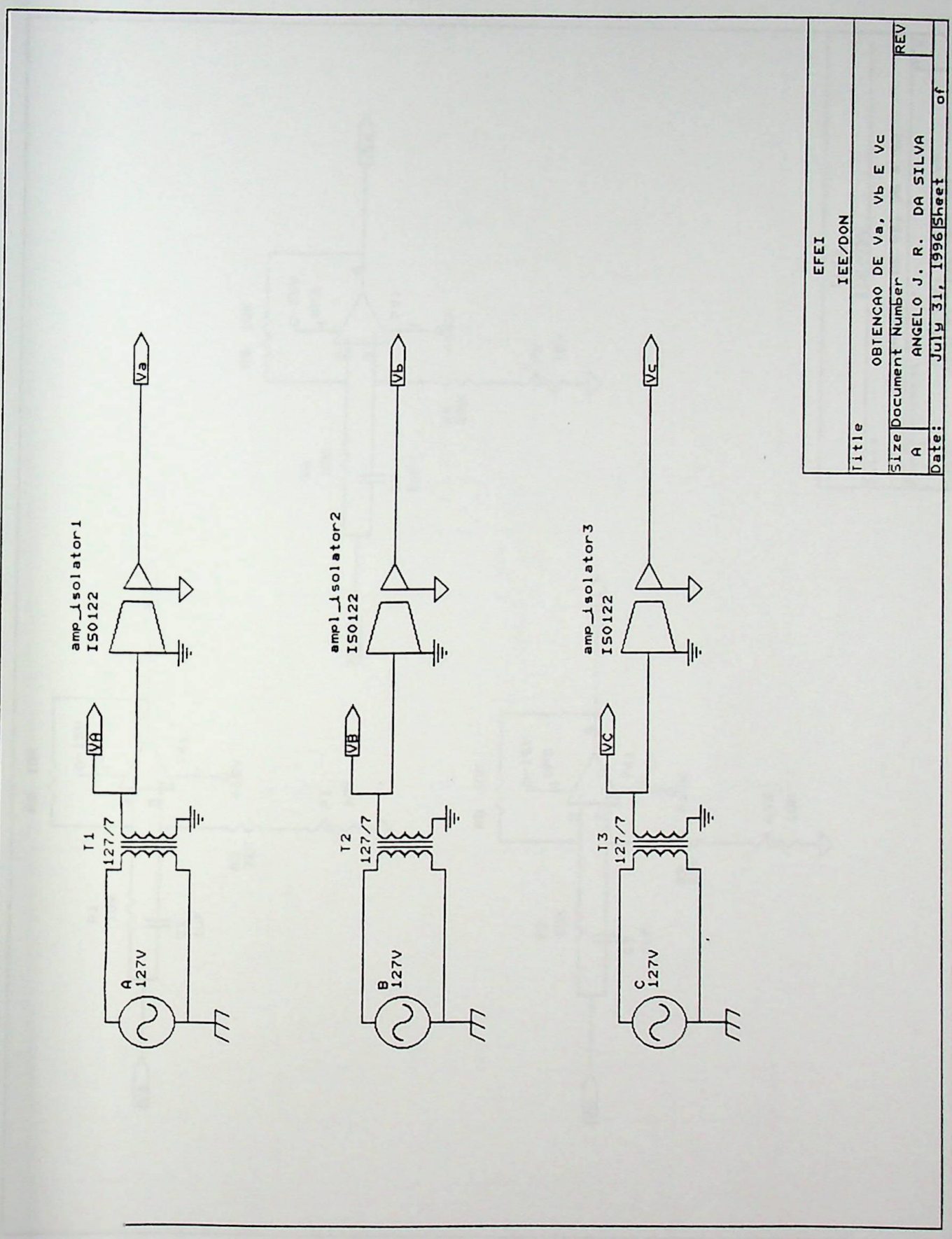
Obs.: valores à esquerda do sinal -- indicam valores na pilha antes da execução da palavra, valores à direita do sinal -- indicam valores na pilha após a execução da palavra.

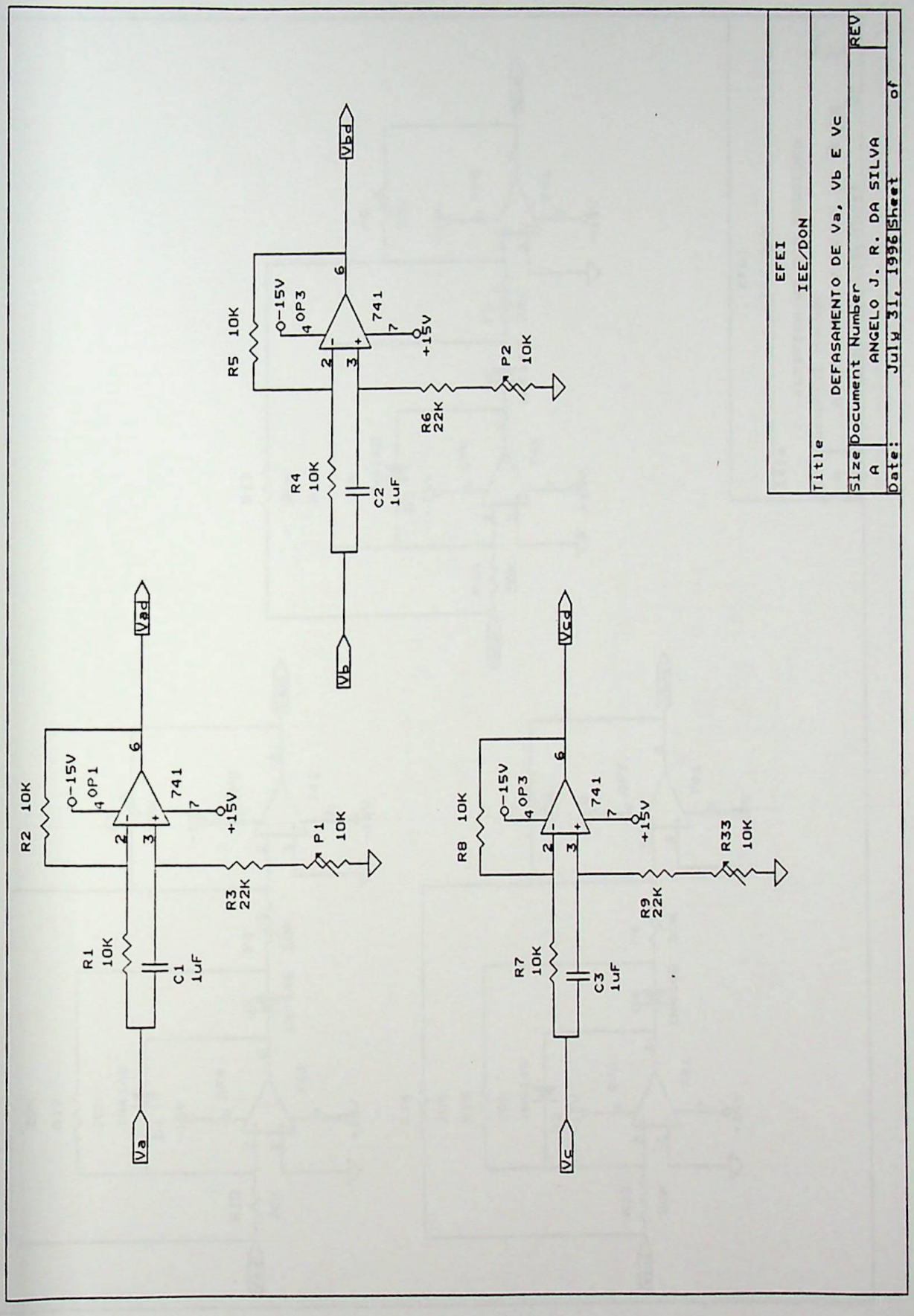
Tabela II. 1 Comunicação com os coprocessadores.

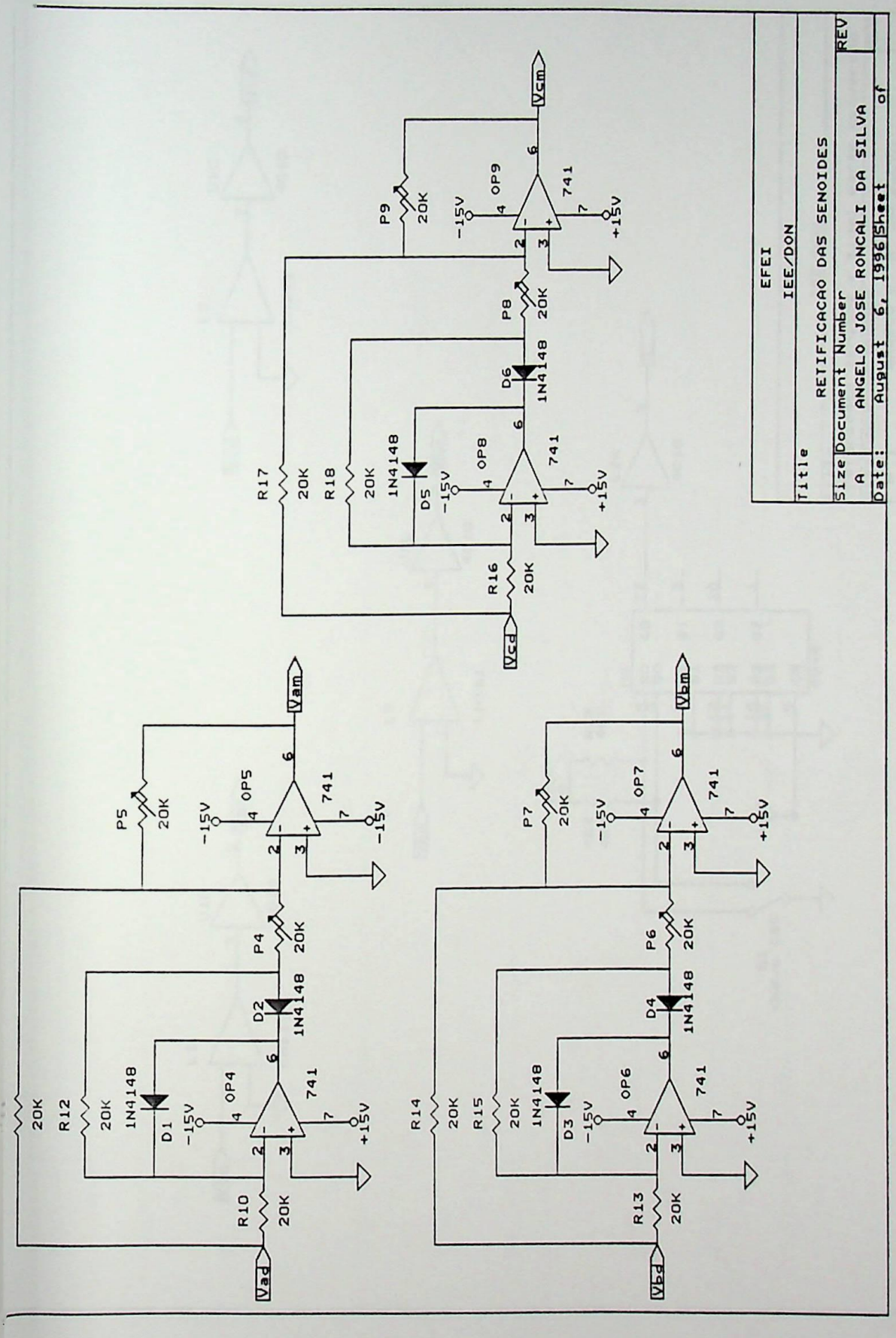
Palavra	Definição
:cpload	(ad n --) Desativar o coproc. especificado (n= 0 a 3) e transmitir um arquivo de formato Intel 8 bits.
?cp	(n -- b) Ler o registro de estado da interface de barramento do coproc. n.
>cp	(b n --) Transmitir o byte b ao registro de entrada do coproc. n com o pino de controle da interface barramento A0 em 0.
>cp'	(b n --) Transmitir o byte b ao registro de entrada do coproc. n com o pino de controle da interface de barramento A0 em 1.
0>1	Copiar toda a memória externa do coproc. 0 para o coproc. 1.
0>123	Copiar a memória externa do coproc. 0 para os coproc. 1,2 e 3.
01>23	Copiar as memórias externas dos coproc. 0 e 1 para os coproc. 2 e 3, respectivamente.
all-off	Desativar todos os coproc.
all-on	Ativar todos os coproc.
cp>	(n -- b) Ler o registro de saída do coproc. n.
cpb!	(b ad n --) Escrever o byte n dentro da memória do coproc. n. O endereço n é o visto pelo coproc.. Funciona apenas com o coproc. desativado.
cpb@	(ad n -- b) Ler um byte da memória do coproc. n. O endereço n é o visto pelo coproc. Funciona apenas com o coproc. desativado
cpl&g	(n --) Desativar o coproc. n, enviar um arquivo de formato Intel 8 bits à memória do coproc. e em seguida o ativar .
off	(n --) Desativar o coproc. n.
on	(n --) Ativar o coproc. n.

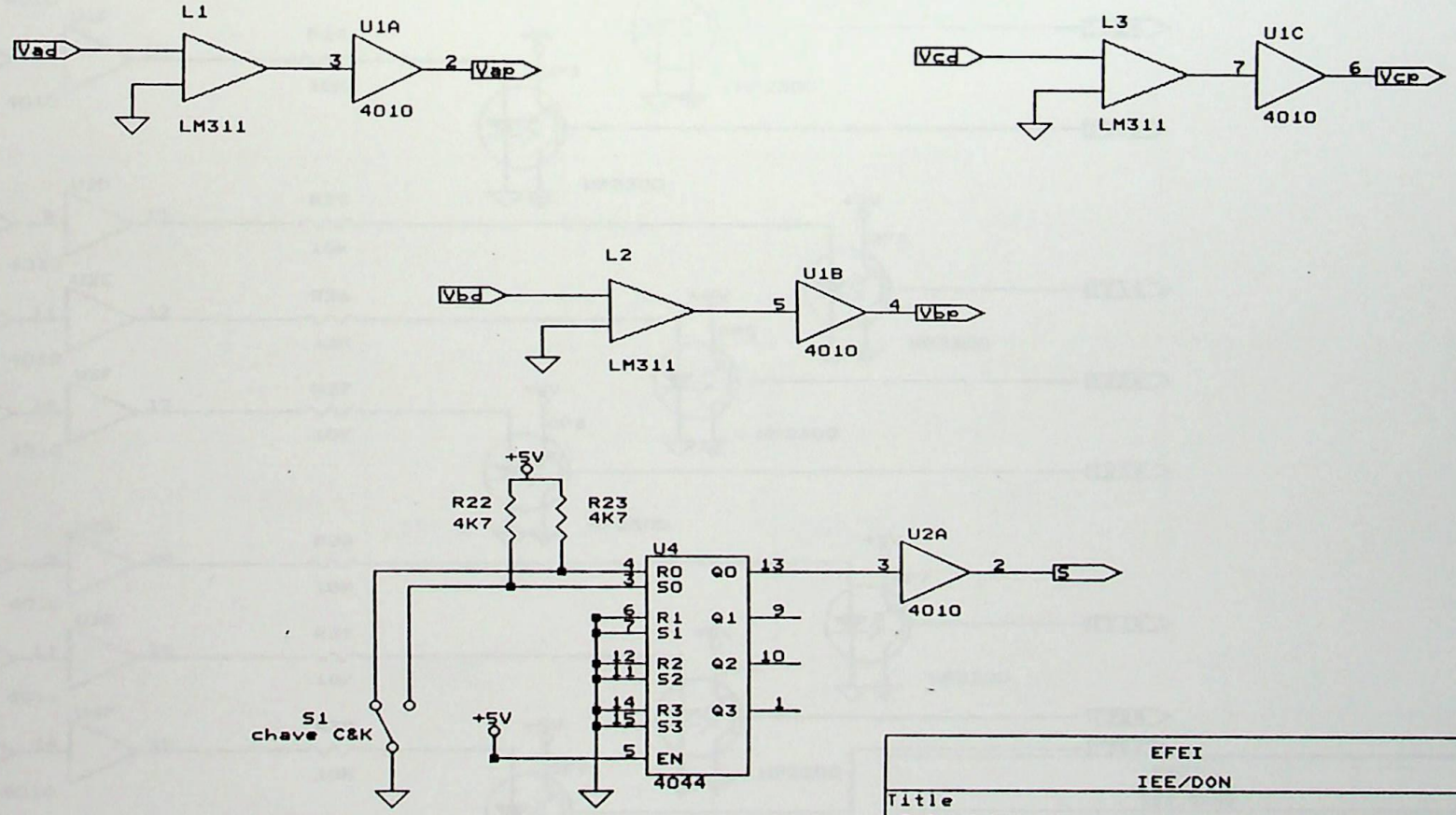
APÊNDICE III

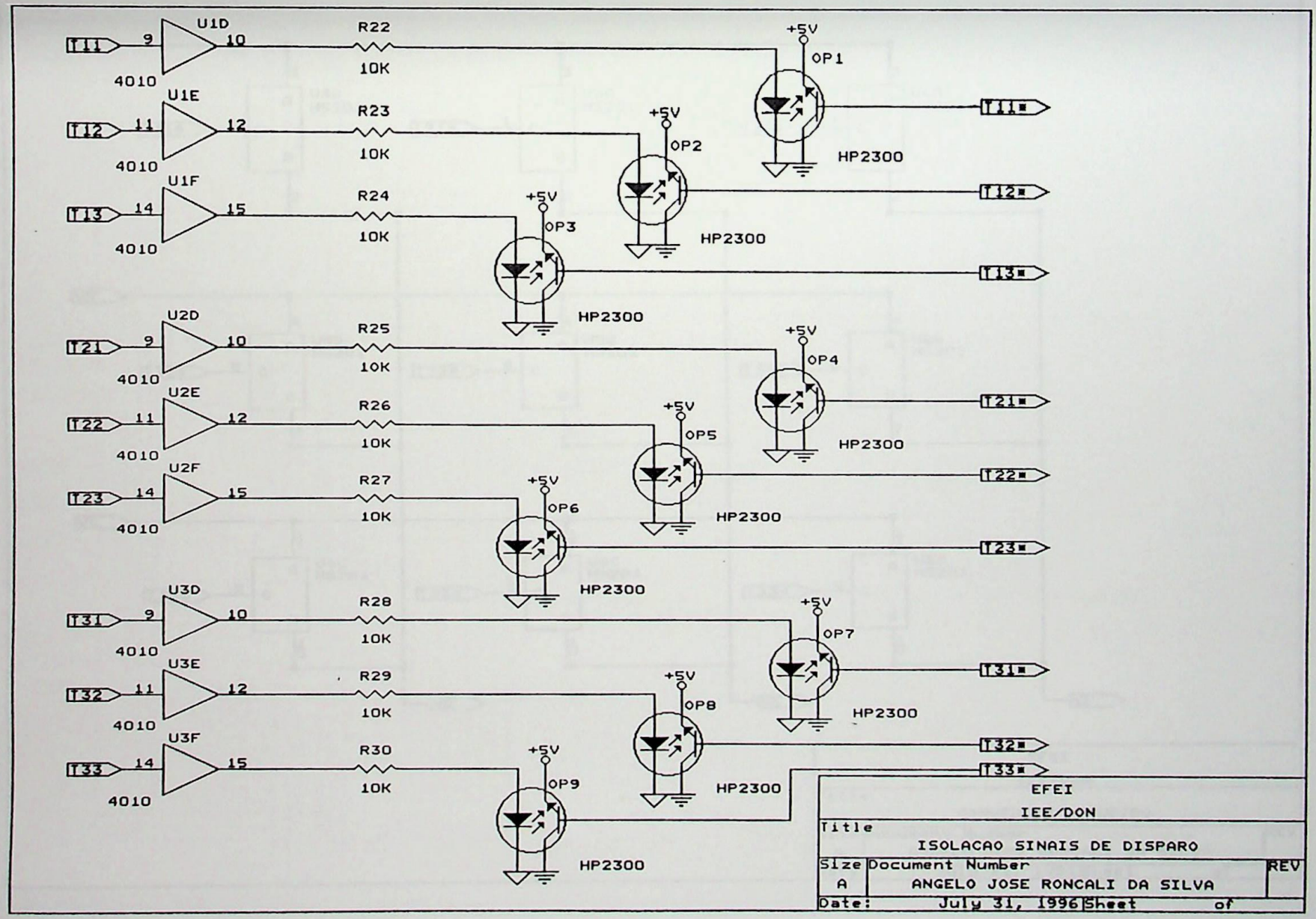
ESQUEMAS DOS CIRCUITOS DE INTERFACE E CONDICIONAMENTO

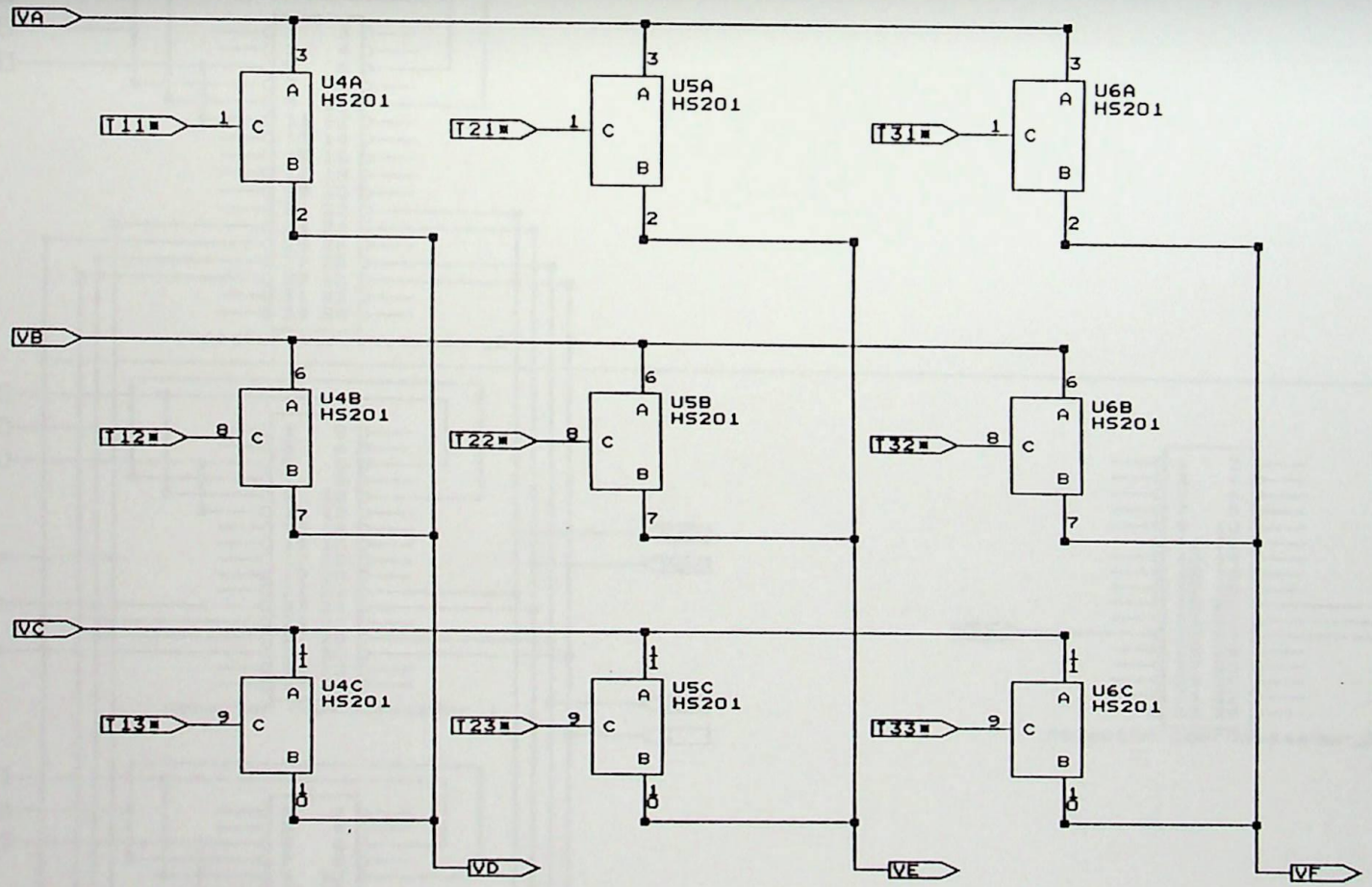

amp_isolator1
ISO122
amp_isolator2
ISO122
amp_isolator3
ISO122
T1
127/7
T2
127/7
T3
127/7
A
127V
B
127V
C
127V
VA
VB
VC
Va
Vb
Vc
EFEI
IEE/DON
Title
OBTENCAO DE Va, Vb E Vc
Size
A
Document Number
REV
ANGELO J. R. DA SILVA
Date:
July 31, 1996
Sheet
of


EFEI
IEE/DON
Title
DEFASAMENTO DE Va, Vb E Vc
Size
A
Document Number
ANGELO J. R. DA SILVA
REV
Date:
July 31, 1996
Sheet
of
Va
Vb
Vc
Vad
Vbd
Vcd
R1 10K
C1 1uF
R2 10K
R3 22K
P1 10K
R4 10K
C2 1uF
R5 10K
R6 22K
P2 10K
R7 10K
C3 1uF
R8 10K
R9 22K
R33 10K
0-15V
4 OP1
741
+15V
0-15V
4 OP3
741
+15V

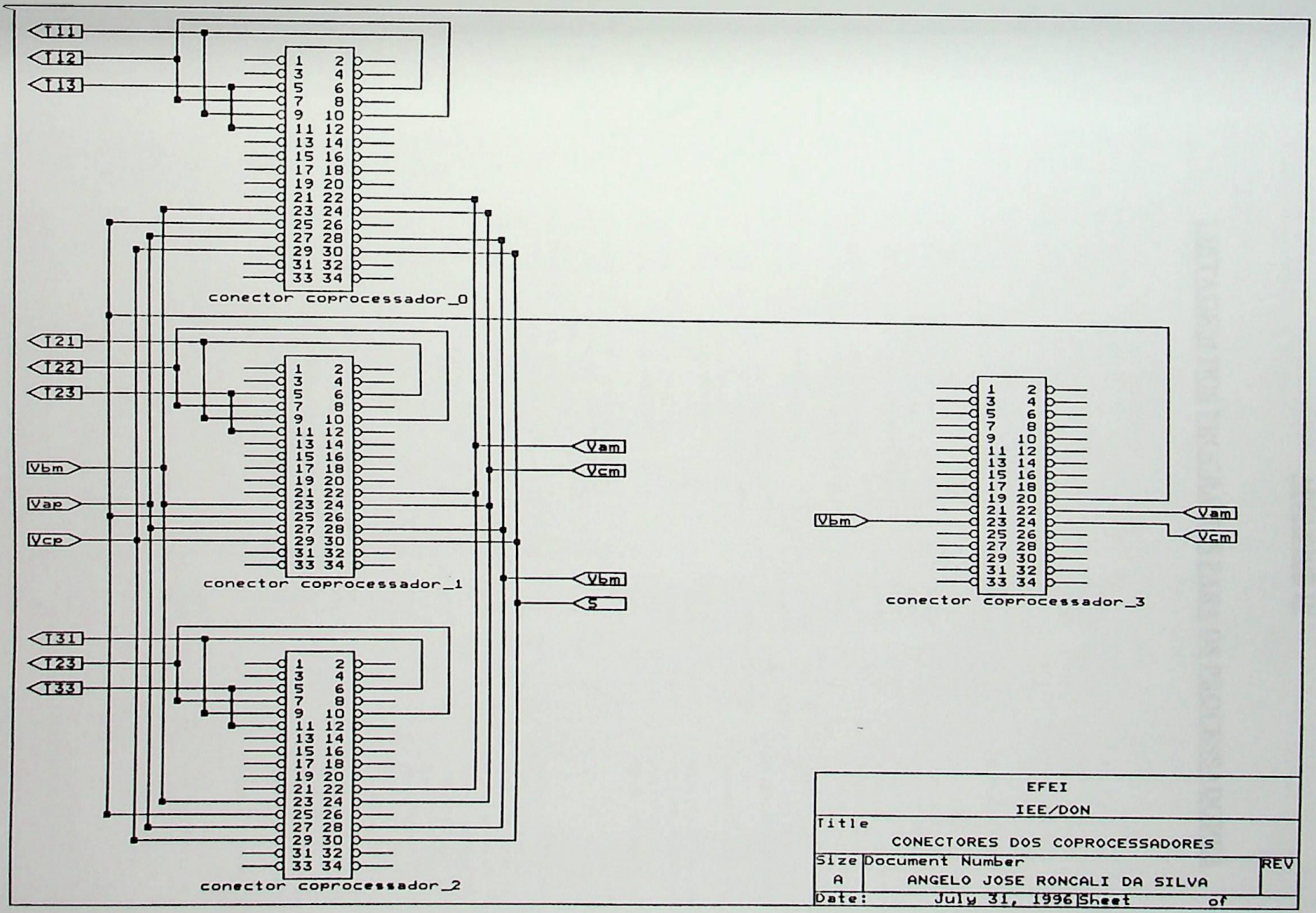

EFEI
IEE/DON
Title
RETIFICACAO DAS SENOIDES
Size
A
Document Number
REV
ANGELO JOSE RONCALI DA SILVA
Date: August 6, 1996 Sheet of
Vad
Vbd
Vcd
Vam
Vbm
Vcm
R10
R12
R13
R14
R15
R16
R17
R18
P4
P5
P6
P7
P8
P9
D1
D2
D3
D4
D5
D6
OP4
OP5
OP6
OP7
OP8
OP9
741
1N4148
20K
-15V
+15V


L1
Vad
LM311
U1A
3
2
Vap
4010
L3
Vcd
LM311
U1C
7
6
Vcp
4010
L2
Vbd
LM311
U1B
5
4
Vbp
4010
+5V
R22
4K7
R23
4K7
U4
R0
S0
Q0
13
R1
S1
Q1
9
R2
S2
Q2
10
R3
S3
Q3
1
EN
4044
U2A
3
2
S
4010
S1
chave C&K
+5V
EFEI
IEE/DON
Title
Polaridade tensoes e sinal parar conversao
Size
A
Document Number
ANGELO J. R. DA SILVA
REV
Date:
July 31, 1996
Sheet
of


T11
9
U1D
10
4010
R22
10K
+5V
OP1
T11*
T12
12
U1E
11
4010
R23
10K
+5V
OP2
HP2300
T12*
T13
14
U1F
15
4010
R24
10K
+5V
OP3
HP2300
T13*
T21
9
U2D
10
4010
R25
10K
+5V
OP4
HP2300
T21*
T22
12
U2E
11
4010
R26
10K
+5V
OP5
HP2300
T22*
T23
14
U2F
15
4010
R27
10K
+5V
OP6
HP2300
T23*
T31
9
U3D
10
4010
R28
10K
+5V
OP7
HP2300
T31*
T32
12
U3E
11
4010
R29
10K
+5V
OP8
HP2300
T32*
T33
14
U3F
15
4010
R30
10K
+5V
OP9
HP2300
T33*
EFEI
IEE/DON
Title
ISOLACAO SINAIS DE DISPARO
Size
A
Document Number
ANGELO JOSE RONCALI DA SILVA
REV
Date:
July 31, 1996
Sheet
of



EFEI		
IEE/DON		
Title: CONVERSOR REDUZIDO		
Size: A	Document Number: ANGELO JOSE RONCALI DA SILVA	REV
Date: July 31, 1996	Sheet of	


T11
T12
T13
conector coprocessador_0
T21
T22
T23
Vbm
Vap
Vcp
Vam
Vcm
Vbm
5
conector coprocessador_1
T31
T23
T33
conector coprocessador_2
Vbm
Vam
Vcm
conector coprocessador_3
EFEI
IEE/DON
Title
CONECTORES DOS COPROCESSADORES
Size
A
Document Number
ANGELO JOSE RONCALI DA SILVA
REV
Date:
July 31, 1996
Sheet
of

APÊNDICE IV

LISTAGEM DOS PROGRAMAS PARA OS PROCESSADORES

```
;Programa para o controle de três interruptores de um ramo de
;saída de um conversor direto de freqüência, utilizando como
;método de controle o Algoritmo Escalar
;
CPU "M450.TBL"
HOF "INT8"
;
;Endereços do 37450
;
P3:       EQU $D6       ;registro P3
P3D:      EQU $D7       ;registro de direcao P3
P4:       EQU $D8       ;registro P4
P5:       EQU $DA       ;registro P5
P5D:      EQU $DB       ;registro de direcao P5
P6:       EQU $DC       ;registro P6
P6D:      EQU $DD       ;registro de direcao P6
;
MISRG1: EQU $DE         ;registro de miscelanea 1
MISRG2: EQU $DF         ;registro de miscelanea 2
;
DA1R:     EQU $E0       ;registro do DAC 1
DA2R:     EQU $E1       ;registro do DAC 2
ADR:      EQU $E2       ;registro do ADC
ADCR:     EQU $E3       ;registro de controle do ADC
;
IBR:      EQU $E4       ;registro do data bus
IBSTAT:  EQU $E5        ;registro de estado do data bus
;
RTBR:     EQU $E6       ;registro do buffer de
;recepcao/transmissao
SIOS:     EQU $E7       ;registro de estado da I/O serial
SIOC:     EQU $E8       ;registro de controle da I/O serial
UACON:  EQU $E9         ;registro de controle UART
BRG:      EQU $EA       ;gerador taxa baud
;
PWML:     EQU $EB       ;registro PWM low
PWMH:     EQU $EC       ;registro PWM high
;
T1C:      EQU $ED       ;registro de controle timer 1
T2C:      EQU $EE       ;registro de controle timer 2
T3C:      EQU $EF       ;registro de controle timer 3
T1L:      EQU $F0       ;registro timer 1 low
T1H:      EQU $F1       ;registro timer 2 high
T1LL:     EQU $F2       ;latch timer 1 low
T1LH:     EQU $F3       ;latch timer 1 high
T2L:      EQU $F4       ;registro timer 2 low
T2H:      EQU $F5       ;registro timer 2 high
T2LL:     EQU $F6       ;latch timer 2 low
T2LH:     EQU $F7       ;latch timer 2 high
T3L:      EQU $F8       ;registro timer 3 low
T3H:      EQU $F9       ;registro timer 3  high
T3LL:     EQU $FA       ;latch 3 low
T3LH:     EQU $FB       ;latch 3 high
;
IRQ1:     EQU $FC       ;registro requisicao interrupcao 1
IRQ2:     EQU $FD       ;registro requisicao interrupcao 2
ICON1:    EQU $FE       ;registro controle interrupcao 1
ICON2:    EQU $FF       ;registro controle interrupcao 2
;
;
;Constantes e outros enderecos
;
ORG       $08
carry5:   DFS 1*1       ;valor do carry1
carry4:   DFS 1*1       ;valor do carry2
carry3:   DFS 1*1       ;valor do carry3
w8:       DFS 1*1       ;valor de w8(multiplicacao de 16
;bits)
carry2:   DFS 1*1       ;valor do carry4
carry1:   DFS 1*1       ;valor do carry5
w6:       DFS 1*1       ;valor de w1(multiplicacao de 16
;bits)
w4:       DFS 1*1       ;valor de w4(multiplicacao de 16
;bits)
w2:       DFS 1*1       ;valor de w6(multiplicacao de 16
;bits)
;       b2 b1
;     x b4 b4
;     --------
;       w2 w1
;   w4 w3 00
;   w6 w5 00
;w8 w7 00 00
;
ORG       $50
;
ftimer1:  DFS 1*1       ;fim timer 1
ftimer2:  DFS 1*1       ;fim timer 2
ftimer3:  DFS 1*1       ;fim timer 3
mascara:    DFS 1*1     ;mascara do carry
Tmaxlow:  DFS 1*1       ;tempo maximo do
;contador(low)(500 us)
Tmaxhigh: DFS 1*1       ;tempo maximo do
;contador(high)(500 us)
result1l: DFS 1*1       ;byte1 resultado Tl(desprezado)
result2l: DFS 1*1       ;byte2 resultado Tl(desprezado)
result3l: DFS 1*1       ;byte3 resultado Tl(low)
result4l: DFS 1*1       ;byte4 resultado Tl(high)
result1k: DFS 1*1       ;byte1 resultado Tk(desprezado)
result2k: DFS 1*1       ;byte2 resultado Tk(desprezado)
result3k: DFS 1*1       ;byte3 resultado Tk(low)
result4k: DFS 1*1       ;byte4 resultado Tk(high)
result3m:   DFS 1*1     ;byte low Tm
result4m:   DFS 1*1     ;byte high Tm
stack:    DFS 1*1       ;posicao do stack para
;[Vx*(Vm+-Vo)]*348
stack1:   DFS 1*1       ;posicao do stack para
;Vx*(Vm+-Vo)
stacks1:  DFS 1*1       ;stack para entrada na subrotina
stacks2:  DFS 1*1       ;stack para saida da subrotina
zero:     DFS 1*1       ;valor zero
fstop:    DFS 1*1       ;flag de sincronismo entre os
;micro
;
ORG   $80
;
Va:     DFS 1*1         ;modulo da tensao Va
sigva:  DFS 1*1         ;sinal da tensao Va
Vb:     DFS 1*1         ;modulo da tensao Vb
sigvb:  DFS 1*1         ;sinal da tensao Vb
Vc:     DFS 1*1         ;modulo da tensao Vc
sigvc:  DFS 1*1         ;sinal da tensao Vc
Vo:     DFS 1*1         ;modulo da tensao Vo
sigvo:  DFS 1*1         ;sinal da tensao Vo
Vm:     DFS 1*1         ;modulo da tesao Vm
sigvm:  DFS 1*1         ;sinal da tensao Vm
Vl:     DFS 1*1         ;modulo da tensao Vl
Vk:     DFS 1*1         ;modulo da tensao Vk
tm:     DFS 1*1         ;tempo tm
Kl:     DFS 1*1         ;constante de normalizacao(low)
Kh:     DFS 1*1         ;constante de normalizacao
;(high)
y:      DFS 1*1         ;variavel de primeira vez
;
ORG   $90
mul1l:  DFS 1*1         ;resultado de
;(Vm+-Vo)*Vx_low
mul1h:    DFS 1*1       ;resultado de
;(Vm+-Vo)*Vx_high
;
ORG   $A0
freq:   DFS 1*1         ;valor convertido da frequencia,
;representa o passo de amostragem da tabela (enviado pelo
;PC)
ampl:   DFS 1*1         ;valor convertido da amplitude
;(enviado pelo PC)
stop1:  DFS 1*1         ;limite superior para retorno na
;tabela
```

```
v1:     DFS 1*1             ;variavel para armazenamento
;temporario de dados
Vi:     DFS 1*1             ;valor de pico
;
;tabela com os valores do seno entre 0 e 90 graus
ORG    $1000
seno:DFB $00,$02,$03,$05,$06,$08,$09,$0b,$0d,$0e
     DFB $10,$11,$13,$14,$16,$18,$19,$1b,$1c,$1e
     DFB $1f,$21,$22,$24,$26,$27,$29,$2a,$2c,$2d
     DFB $2f,$30,$32,$33,$35,$37,$38,$3a,$3b,$3d
     DFB $3e,$40,$41,$43,$44,$46,$47,$49,$4a,$4c
     ;
     DFB $4d,$4f,$50,$52,$53,$55,$56,$58,$59,$5b
     DFB $5c,$5e,$5f,$60,$62,$63,$65,$66,$68,$69
     DFB $6b,$6c,$6d,$6f,$70,$72,$73,$74,$76,$77
     DFB $79,$7a,$7b,$7d,$7e,$80,$81,$82,$84,$85
     DFB $86,$88,$89,$8a,$8c,$8c,$8d,$8f,$91,$92
     ;
     DFB $93,$95,$96,$97,$98,$9a,$9b,$9c,$9d,$9f
     DFB $a0,$a1,$a2,$a4,$a5,$a6,$a7,$a8,$a9,$ab
     DFB $ac,$ad,$ae,$af,$b0,$b2,$b3,$b4,$b5,$b6
     DFB $b7,$b8,$b9,$ba,$bb,$bc,$bd,$be,$c0,$c1
     DFB $c2,$c3,$c4,$c5,$c6,$c7,$c8,$c9,$ca,$cb
     ;
     DFB $cb,$cc,$cd,$ce,$cf,$d0,$d1,$d2,$d3,$d4
     DFB $d5,$d5,$d6,$d7,$d8,$d9,$da,$da,$db,$dc
     DFB $dd,$de,$de,$df,$e0,$e1,$e1,$e2,$e3,$e4
     DFB $e4,$e5,$e6,$e6,$e7,$e8,$e8,$e9,$ea,$ea
     DFB $eb,$eb,$ec,$ed,$ed,$ee,$ee,$ef,$ef,$f0
     ;
     DFB $f0,$f1,$f2,$f2,$f3,$f3,$f3,$f4,$f4,$f5
     DFB $f5,$f6,$f6,$f7,$f7,$f7,$f8,$f8,$f8,$f9
     DFB $f9,$F9,$fa,$fa,$fa,$fb,$fb,$fb,$fb,$fc
     DFB $fc,$fc,$fc,$fd,$fd,$fd,$fd,$fd,$fe,$fe
     DFB $fe,$fe,$fe,$fe,$fe,$ff,$ff,$ff,$ff,$ff
     ;
     DFB $ff,$ff,$ff,$ff,$ff,$ff
;
;tabela com as constantes multiplicativas
const:DFB  $03,$47,$03,$4d,$03,$54,$03,$5b,$03,
      DFB  $62,$03,$69,$03,$70,$03,$77,$03,$7e,
      DFB  $03,$85,$03,$8d,$03,$94,$03,$9c,$03,
      DFB  $a3,$03,$ab,$03,$b3,$03,$bb,$03,$c3,
      DFB  $03,$cb,$03,$d3,$03,$dc,$03,$e4,$03,
      DFB  $ed,$03,$f5,$03,$fe,$04,$07,$04,$10,
      DFB  $04,$19,$04,$23,$04,$2c,$04,$35,$04,
      DFB $3f,$04,$49,$04,$53,$04,$5d,$04,$67,
      DFB $04,$71,$04,$7c,$04,$86,$04,$91,$04,
      DFB $9c,$04,$a7,$04,$b2,$04,$be,$04,$c9,
      DFB $04,$d5,$04,$e1,$04,$ed,$04,$f9,$05,
      DFB $05,$05,$12,$05,$1f,$05,$2c,$05,$39,
      DFB $05,$46,$05,$54,$05,$61,$05,$6f,$05,
      DFB $7e,$05,$8c,$05,$9b,$05,$a9,$05,$b8
      DFB $05,$c8,$05,$d7,$05,$e7,$05,$f7,$06,
      DFB $07,$06,$18,$06,$29,$06,$3a,$06,$4b,
      DFB $06,$5d,$06,$6f,$06,$81,$06,$94,$06,
      DFB $a7,$06,$ba,$06,$cd,$06,$e1,$06,$f5,
      DFB $07,$0a,$07,$1f,$07,$34,$07,$4a,$07,
      DFB $60,$07,$76,$07,$8d,$07,$a4,$07,
      DFB $bc,$07,$d4,$07,$ec,$08,$05,$08,$1f,
      DFB $08,$39,$08,$53,$08,$6f,$08,$89,$08,
      DFB $a5,$08,$c2,$08,$df,$08,$fc,$09,$1c,
      DFB $08,$39,$09,$59,$09,$79,$09,$99,
      DFB $09,$bb,$09,$dd,$0a,$00,$0a,$23,$0a,
      DFB $47,$0a,$6c,$0a,$92,$0a,$b8,$0a,$e0,
      DFB $0b,$08,$0b,$31,$0b,$5b,$0b,$86,$0b,$b2
;
;Programa
;
ORG    $4000
;
;-----------------------------------------------
;inicio do programa por reset
;
start1:  clt                ;clear o bit t
         cld                ;clear o bit d
         ldm #00000110b,MISRG2 ;stack on page 0 e uP
;mode
         clb 2,IBSTAT
;-----------------------------------------------
;
;inicializacao dos registros
;portas
;

         ldm #00000111b,P3D  ;bits 0-2 como saida e ;3-7
;como entrada da porta 3
         clb 0,P3            ;clear bits 0, 1 e 3 de P3
         clb 1,P3
         clb 2,P3
         ldm #00001000b,P6D  ;bits 0-2 como entrada da
;porta ;6
;
;variaveis
;
start3:  ldm #$48,kl         ;carrega kl com 48h
         ldm #$03,kh         ;carrega kh com 04h
         ldm #$00,ftimer1    ;carrega ftimer1 com 0h
         ldm #$00,ftimer2    ;carrega ftimer2 com 0h
         ldm #$00,ftimer3    ;carrega ftimer3 com 0h
         ldm #$01,mascara    ;carrega mascara com 1
         ldm #$e1,Tmaxlow    ;carrega Tmaxlow com E1h
;(500 us)
         ldm #$04,Tmaxhigh   ;carrega Tmaxhigh com 04h
;(500 us)
         ldm #$10,stack      ;carrega stack com 10h, para
;multiplicacao 16bits
         ldm #$91,stack1     ;carrega stack1 com 91h, para
;multiplicacao por 348h
         ldm #$20,stacks1    ;carrega stacks1 com 20h,
;para entrada na subrotina
         ldm #$1e,stacks2    ;carrega satcks2 com 1Eh,
;para saida da subrotina
         ldm #$00,zero       ;carrega zero com 0h
         ldm #$00,y          ;carrega y com 0h
         ldm #$00,fstop      ;carrega fstop com 0h
;desabilitacao dos contadores, senao os contadores continuam
;contando, mesmo com a chave desligada
         ldm #$01,ICON1      ;desabilita interrupcoes
         ldm #$00,ICON2      ;desabilita interrupcoes
;escrita nos registros de latch e contagem dos contadores
         ldm #$06,T1C        ;contador 1 em one-shot
;generation mode
         ldm #$06,T2C        ;contador 2 em one-shot
;generation mode
         ldm #$00,T3C        ;contador operacao normal
         ldm #$00,MISRG1     ;possibilta escrita nos
;contadores
         ldm #$ff,T1LL       ;inicializa registro timer1
;latch low
         ldm #$ff,T1LH       ;inicializa registro timer1
;latch high
         ldm #$ff,T1L        ;inicializa registro timer1 low
         ldm #$ff,T1H        ;inicializa registro timer1 high
         ldm #$ff,T2LL       ;inicializa registro timer2
;latch low
         ldm #$ff,T2LH       ;inicializa registro timer2
;latch high
         ldm #$ff,T2L        ;inicializa registro timer2 low
         ldm #$ff,T2H        ;inicializa registro timer2 high
         ldm #$ff,T3LL       ;inicializa registro timer3
;latch low
         ldm #$ff,T3LH       ;inicializa registro timer3
;latch high
         ldm #$ff,T3L        ;inicializa registro timer3 low
         ldm #$ff,T3H        ;inicializa registro timer3 high
;
         ldm #$40,MISRG1     ;habilita contagem timer3
         sei                 ;desabilita interrupcoes
```

```
        ldx #$22                ;carrega stack pointer com
;22h
        txs
;
;obtencao de Vo
;
;registro Y e o contador para a tabela
        ldm #$33,ampl   ;amplitude igual a 2 volts
        ldm #$1e,freq   ;frequencia igual a 60 Hz
        lda #$ff        ;faz stop1= FFh - freq
        sec
        sbc freq
        sta stop1
        ldm #$00,sigvo  ;sinal da tensao negativo
        ldy #$00        ;tensao a 0 graus
;
p:      lda sigvo       ;inverte sinal da tensao
        eor #00010000b
        sta sigvo
        lda seno,y      ;ler valor da tabela
        jsr cal_amp     ;jump para rotina de calculo da
;amplitude
        jsr escalar     ;jump para rotina de
;implemantacao do algoritmo escalar
p1:     clc             ;calcula posicao do novo valor da
;tabela
        tya
        adc freq
        tay
        lda seno,y      ;ler valor da tabela
        jsr cal_amp     ;jump para rotina de calculo da
;amplitude
        jsr escalar     ;jump para rotina de
;implementacao do algoritmo escalar
        cpy stop1       ;compara Y com stop1
        bcc p1          ;se menor vai para p1
        lda #$ff        ;faz a=FFh-Y
        sty v1
        sec
        sbc v1
        sta v1
        lda freq        ;faz b=freq-a
        sec
        sbc v1
        sta v1
        lda #$ff        ;faz c=FFh-b,que e o novo valor de
;Y
        sec
        sbc v1
        tay
        lda seno,y      ;ler valor da tabela
        jsr cal_amp     ;jump para rotina de calculo da
;amplitude
        jsr escalar     ;jump para rotina de
;implementacao do algoritmo escalar
p2:     sec             ;calcula posicao do novovalor da
;tabela
        tya
        sbc freq
        tay
        lda seno,y      ;ler valor da tabela
        jsr cal_amp     ;jump para rotina de calculo da
;amplitude
        jsr escalar     ;jump para rotina de implemantacao
;do AE
        cpy freq        ;compara Y com freq
        bcs p2          ;se maior vai para p2
        sty v1          ;faz a=freq-Y, que e o novo valor de
;Y
        lda freq
        sec
        sbc v1
        tay
        jmp p           ;jump para p
;
;
;rotina de calculo da amplitude
;
cal_amp:  ldx zero      ;multiplica o valor da tabela por ampl
        mul ampl,x
        pla             ;coloca valor do TOS no A
        sta Vo          ;armazema valor em Vo
        rts             ;retorno
;
;rotina de implementacao do algoritmo escalar
;
escalar:
;
;leitura da chave
;
start2: lda P4          ;verifica se chave esta em on
        and #$80        ;se estiver prossegue va para cont
        bne cont        ;se nao estiver va prossegue
        lda y           ;y=0?
        beq start2      ;se for, va para start2
        cli
t1:     lda ftimer3     ;timer2 terminou contagem?
        beq t1          ;se nao, va para t1
        ldm #$1d,ICON1  ;habilita INTs
        ldm #$00,T1C    ;para timer1
        ldm #$00,T2C    ;para timer2
t2:     lda ftimer1     ;timer3 terminou contagem?
        beq t2          ;se nao, va para t2
        ldm #$00,T3C    ;para timer3
        ldm #$06,T1C    ;timer1 em one shot generation
        ldm #$06,T2C    ;timer2 em one shot generation
        ldm #$00,y
        sei
;
start4: lda P4          ;verifica se chave esta em on
        and #$80        ;se estiver prossegue programa
        beq start4      ;se nao estiver va para start4
        ldm #$00,MISRG1;possibilita escrita nos contadores
        ldm #$00,fstop
        ldm #$00,T3LL   ;inicializa registro timer3 latch low
        ldm #$05,T3LH   ;inicializa registro timer3 latch
;high
        ldm #$00,T3L    ;inicializa registro timer3 low
        ldm #$05,T3H    ;inicializa registro timer3 high
        clb 7,IRQ1      ;clear pedido interrupcao timer3
        ldm #$40,MISRG1;habilita contagem timer3

;leitura de Va,sigva,Vb,sigvb,Vc,sigvc,Vo,sigvo
;leitura de Va e sigva
cont:   ldm #$00,ADCR   ;start no conversor AD para le Va
        lda P4          ;le porta3 (sinal de Va)
        and #$10        ;and do acumulador com 4h
        sta sigva       ;store sinal de Va em sigva
l1:     lda IRQ2        ;load acumulador com registro de
;requisicao de int.
        and #$20        ;mascara bits de IRQ2
        beq l1          ;verifica se conversao terminou
        clb 5,IRQ2      ;clear pedido interrupcao A/D
        lda ADR         ;load acumulador com modulo de
;Va
        sta Va          ;store modulo de Va em Va
;
;leitura de Vb e sigvb
;
        ldm #$01,ADCR   ;start no conversor AD para le Vb
        lda P4          ;le porta3 (sinal de Vb)
        and #$20        ;and do acumulador com 10h
        lsr A           ;deslocamento do bit de sinal para
;posicao 3
        sta sigvb       ;store sinal de Vb em sigvb
l2:     lda IRQ2        ;load acumulador com registro de
;requisicao de int.
        and #$20        ;mascara bits de IRQ2
        beq l2          ;verifica se conversao terminou
        clb 5,IRQ2      ;clear pedido interrupcao A/D
```

```
        lda ADR          ;load acumulador com modulo de
;Vb
        sta Vb           ;store modulo de Vb em Vb
;
;leitura de Vc e sigvc
;
        ldm #$02,ADCR    ;start no conversor AD para ler
;Vc
        lda P4           ;le porta3 (sinal de Vc)
        and #$40         ;and do acumulador com 20h
        lsr A            ;deslocamento do bit de sinal
;para posicao 4
        lsr A            ;deslocamento do bit de sinal
;para posicao 3
        sta sigvc        ;store sinal de Vc em sigvc
l3:     lda IRQ2         ;load acumulador com registro
;de requisicao de int.
        and #$20         ;mascara bits de IRQ2
        beq l3           ;verifica se conversao terminou
        clb 5,IRQ2       ;clear pedido interrupcao A/D
        lda ADR          ;load acumulador com modulo
;de Vc
        sta Vc           ;store modulo de Vc em Vc
;
;leitura do valor de pico
;
        ldm #$03,ADCR    ;start no conversor AD para le
;Vi
l4:     lda IRQ2         ;load acumulador com registro
;de requisicao de int.
        and #$20         ;mascara bits de IRQ2
        beq l4           ;verifica se conversao terminou
        clb 5,IRQ2       ;clear pedido interrupcao A/D
        lda ADR          ;load acumulador com modulo
;de Vi
        sta Vi           ;store modulo de Vi em Vi
        lda #$ff
        sec
        sbc Vi           :ler tabela 2.(FF-Vi)
        asl A
        tax
        lda const,x
        sta Kh           :primeiro byte
        inx
        lda const,x
        sta Kl           ;segundo byte
;
;Determinacao de Vk,Vl,Vm
;Calculo dos tempos tk,tl,tm
;
        lda sigva        ;carrega no acumulador sinal de Va
        cmp sigvb        ;sinal de Va igual sinal de Vb?
        beq frente1      ;se for va para frente1
        jmp frente1a     ;se nao for va para frente1a
frente1:lda Vc
        sta Vm           ;Vm igual a Vc
        lda sigvc
        sta sigvm        ;sinal de Vm igual a sinal de Vc
        lda Va           ;carrega no acumulador modulo de
;Va
        cmp Vb           ;Va maior ou igual a Vb?
        bcs frente2      ;se for, va para frente2
        jmp frente2a     ;se nao, va para frente2a
frente2:sta Vl           ;Vl igual a Va
        lda Vb
        sta Vk           ;Vk igual a Vb
        lda sigvm        ;carrega A com sinal de Vm
        cmp sigvo        ;sinal de Vm igual a sinal de Vo?
        beq frente20     ;se for va para frente20
        jsr calcmais     ;va para rotina calculo dos tempos
        jsr cont1        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
;
frente20: jsr calcmenos  ;va para subrotina calculo dos
;tempos
        jsr cont1        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
;
frente2a: sta Vk         ;Vk igual a Va
        lda Vb
        sta Vl           ;Vl igual a Vb
        lda sigvm        ;sinal de Vm e negativo?
        cmp sigvo        ;sinal de Vm igual a sinal de
;Vo?
        beq frente40     ;se for va para frente40
        jsr calcmais     ;va para subrotina calculo dos
;tempos
        jsr cont2        ;va para rotina de
;carregamento dos contadores
        rts
frente40: jsr calcmenos  ;va para subrotina calculo dos
;tempos
        jsr cont2        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
;
frente1a: lda sigva      ;carrega no acumulador sinal de
;Va
        cmp sigvc        ;sinal de Va igual sinal de Vc?
        beq frente5      ;se for, va para frente5
        jmp frente5a     ;se nao, va para frente5a
frente5: lda Vb
        sta Vm           ;Vm igual a Vb
        lda sigvb
        sta sigvm        ;sinal de Vm igual a sinal de Vb
        lda Va           ;carrega no acumulador modulo
;de Va
        cmp Vc           ;Va maior ou igual a Vc?
        bcs frente6      ;se for, va para frente6
        jmp frente6a     ;se nao for, va para frente6a
frente6: sta Vl          ;Vl igual a Va
        lda Vc
        sta Vk           ;Vk igual a Vc
        lda sigvm        ;sinal de Vm e negativo?
        cmp sigvo        ;sinal de Vm igual a sinal de
;Vo?
        beq frente60     ;se for, va para frente60
        jsr calcmais     ;va para subrotina calculo dos
;tempos
        jsr cont3        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
frente60: jsr calcmenos  ;va para subrotina calculo dos
;tempos
        jsr cont3        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
;
frente6a: sta Vk         ;Vk igual a Va
        lda Vc
        sta Vl           ;Vl igual a Vc
        lda sigvm        ;sinal de Vm e negativo?
        cmp sigvo        ;sinal de Vm igual a sinal de
;Vo?
        beq frente80     ;se for, va para frente80
        jsr calcmais     ;va para subrotina calculo dos
;tempos
        jsr cont4        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
;
frente80: jsr calcmenos  ;va para subrotina calculo dos
;tempos
        jsr cont4        ;va para rotina de carregamento
;dos contadores
        rts              ;retorno rotina escalar
;
```

```
frente5a:  lda Va              ;carrega no acumulador sinal de
Va
        sta Vm                 ;Vm igual a Va
        lda sigva
        sta sigvm              ;sinal de Vm igual a sinal de Va
        lda Vb                 ;carrega no acumulador modulo
;de Vb
        cmp Vc                 ;Va maior ou igual a Vc?
        bcs frente9            ;se for, va para frente9
        jmp frente9a           ;se nao, va para frente9a
frente9:   sta Vl              ;Vl igual a Vb
        lda Vc
        sta Vk                 ;Vk igual a Vc
        lda sigvm              ;sinal de Vm e negativo?
        cmp sigvo              ;sinal de Vm igual a sinal de Vo?
        beq frentea0           ;se for, va para frentea0
        jsr calcmais           ;va para subrotina calculo dos
;tempos
        jsr cont5              ;va para rotina de carregamento
;dos contadores
        rts                    ;retorno rotina escalar
;
frentea0:  jsr calcmenos       ;va para subrotina calculo dos
;tempos
        jsr cont5              ;va para rotina de carregamento
;dos contadores
        rts                    ;retorno rotina escalar
;
frente9a:  sta Vk              ;Vk igual a Vb
        lda Vc
        sta Vl                 ;Vl igual a Vc
        lda sigvm              ;sinal de Vm e negativo?
        cmp sigvo              ;sinal de Vm igual a sinal de Vo?
        beq frentec0           ;se for, va para frentec0a
        jsr calcmais           ;va para subrotina calculo dos
;tempos
        jsr cont6              ;va para rotina de carregamento
;dos contadores
        rts                    ;retorno rotina escalar
;
frentec0:  jsr calcmenos       ;va para subrotina calculo dos
;tempos
        jsr cont6              ;va para rotina de carregamento
;dos contadores
        rts                    ;retorno rotina escalar
;
;rotinas de calculo dos tempos
;calt+
calcmais:
;calculo de tl
        ldx stack1         ;carrega X com valor do
;stack=posicao de mul1h
        txs                ;carrega o stach pointer
        ldx zero           ;carrega X com zero
        clc                ;clear carry para adicao
        lda Vm             ;carrega acumulador com Vm
        adc Vo             ;Vm+Vo
        bcc bits8l         ;branch se nao houve carry
        jmp bits9l         ;jump se soma possui 9 bits
bits8l: mul Vl,x           ;(Vm+Vo)*Vl
        sta mul1l          ;armazena produto_low em mul1l
;(Vm+Vo)Vl*384
        ldx stack          ;carrega X com valor do stack
        txs                ;carrega o stack pointer
        ldx zero           ;carrega X
        lda mul1l          ;carrega o A com multiplicando low
        mul Kl,x           ;multiplica o A pelo multiplicador
;low
        sta result1l       ;armazema byte 1 resultado
        lda mul1h          ;carrega A com multiplicando high
        mul Kl,x           ;multiplica o A pelo multiplicador
;low
        sta result2l       ;armazema w3
        lda mul1l          ;carrega A com multiplicando low
        inx                ;incrementa X
        mul Kl,x           ;multiplica o A pelo mutiplicador
;high
        clc                ;clear carry
        adc result2l       ;adiciona w5 com w3
        php                ;salva o valor do carry1
        clc                ;clear o carry
        adc w2             ;adiciona w2 com o resultado de
;(w5+w3)
        php                ;salva o valor do carry2
        sta result2l       ;armazena byte 2 resultado
        lda mascara        ;carrega A com mascara
        and carry1         ;mascara carry1
        sta carry1         ;salva carry1
        lda mascara        ;carrega A com mascara
        and carry2         ;mascara carry2
        sta carry2         ;salva carry2
        clc                ;clear carry
        pla                ;carrega A com valor do carry2
        adc carry1         ;soma carry1 com carry2
        pha                ;salva carry1+carry2
        clc                ;clear carry
        lda mul1h          ;carrega A com mutiplicando high
        mul Kl,x           ;multiplica o A pelo multiplicador
;high
        adc w6             ;soma w7 com w6
        php                ;salva carry3
        clc                ;clear carry
        adc w4             ;soma w4 com (w7+w6)
        php                ;salva carry4
        clc                ;clear carry
        adc carry2         ;soma (w4+w6+w7) com
;(carry1+carry2)
        php                ;salva carry5
        sta result3l       ;armazena byte 3 resultado
        lda mascara        ;carrega A com mascara
        and carry3         ;mascara carry3
        sta carry3         ;salva carry3
        lda mascara        ;carrega A com mascara
        and carry4         ;mascara carry4
        sta carry4         ;salva carry4
        lda mascara        ;carrega A com mascara
        and carry5         ;mascara carry5
        sta carry5         ;salva carry5
        clc                ;clear carry
        pla                ;carrega A com carry5
        adc carry4         ;soma carry4 com carry5
        adc carry3         ;soma carry3 com (carry4+carry5)
        adc w8             ;soma w8 com soma dos carry
        sta result4l       ;armazema byte 4 resultado
        jmp tk1            ;va para calculo de tk
bits9l: mul Vl,x           ;(Vm+Vo)*Vl
        sta mul1l          ;armazena produto low em mul1l
;vl*100
        clc                ;clear carry
        lda mul1h          ;carrega A com byte mais
;significativo
        adc Vl             ;adiciona com Vl
        bcc bits16l        ;se nao houver carry, branch para
;16bits
        jmp bits17l        ;jump se soma for de 17 bits
bits16l:sta mul1h          ;armazena soma em mul1h
;(Vm+Vo)Vl*384
        ldx stack          ;carrega X com valor do stack
        txs                ;carrega o stack pointer
        ldx zero           ;carrega X
        lda mul1l          ;carrega o A com multiplicando
;low
        mul Kl,x           ;multiplica o A pelo multiplicador
;low
        sta result1l       ;armazema byte 1 resultado
        lda mul1h          ;carrega A com multiplicando high
        mul Kl,x           ;multiplica o A pelo multiplicador
;low
        sta result2l       ;armazema w3
        lda mul1l          ;carrega A com multiplicando low
```

```
        inx             ;incrementa X
        mul Kl,x        ;multiplica o A pelo mutiplicador
;high
        clc             ;clear carry
        adc result2l    ;adiciona w5 com w3
        php             ;salva o valor do carry1
        clc             ;clear o carry
        adc w2          ;adiciona w2 com o resultado de
;(w5+w3)
        php             ;salva o valor do carry2
        sta result2l    ;armazena byte 2 resultado
        lda mascara     ;carrega A com mascara
        and carry1      ;mascara carry1
        sta carry1      ;salva carry1
        lda mascara     ;carrega A com mascara
        and carry2      ;mascara carry2
        sta carry2      ;salva carry2
        clc             ;clear carry
        pla             ;carrega A com valor do carry2
        adc carry1      ;soma carry1 com carry2
        pha             ;salva carry1+carry2
        clc             ;clear carry
        lda mul1h       ;carrega A com mutiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;high
        adc w6          ;soma w7 com w6
        php             ;salva carry3
        clc             ;clear carry
        adc w4          ;soma w4 com (w7+w6)
        php             ;salva carry4
        clc             ;clear carry
        adc carry2      ;soma (w4+w6+w7) com
;(carry1+carry2)
        php             ;salva carry5
        sta result3l    ;armazena byte 3 resultado
        lda mascara     ;carrega A com mascara
        and carry3      ;mascara carry3
        sta carry3      ;salva carry3
        lda mascara     ;carrega A com mascara
        and carry4      ;mascara carry4
        sta carry4      ;salva carry4
        lda mascara     ;carrega A com mascara
        and carry5      ;mascara carry5
        sta carry5      ;salva carry5
        clc             ;clear carry
        pla             ;carrega A com carry5
        adc carry4      ;soma carry4 com carry5
        adc carry3      ;soma carry3 com (carry4+carry5)
        adc w8          ;soma w8 com soma dos carry
        sta result4l    ;armazema byte 4 resultado
        jmp tk1         ;va para calculo de tk
bits17l:sta mul1h       ;armazena soma em mul1h
;(Vm+Vo)Vl*384
        ldx stack       ;carrega X com valor do stack
        txs             ;carrega o stack pointer
        ldx zero        ;carrega X
        lda mul1l       ;carrega o A com multiplicando low
        mul Kl,x        ;multiplica o A pelo multiplicador
;low
        sta result1l    ;armazema byte 1 resultado
        lda mul1h       ;carrega A com multiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;low
        sta result2l    ;armazema w3
        lda mul1l       ;carrega A com multiplicando low
        inx             ;incrementa X
        mul Kl,x        ;multiplica o A pelo mutiplicador
;high
        clc             ;clear carry
        adc result2l    ;adiciona w5 com w3
        php             ;salva o valor do carry1
        clc             ;clear o carry
        adc w2          ;adiciona w2 com o resultado de
;(w5+w3)
        php             ;salva o valor do carry2
        sta result2l    ;armazena byte 2 resultado
        lda mascara     ;carrega A com mascara
        and carry1      ;mascara carry1
        sta carry1      ;salva carry1
        lda mascara     ;carrega A com mascara
        and carry2      ;mascara carry2
        sta carry2      ;salva carry2
        clc             ;clear carry
        pla             ;carrega A com valor do carry2
        adc carry1      ;soma carry1 com carry2
        pha             ;salva carry1+carry2
        clc             ;clear carry
        lda mul1h       ;carrega A com mutiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;high
        adc w6          ;soma w7 com w6
        php             ;salva carry3
        clc             ;clear carry
        adc w4          ;soma w4 com (w7+w6)
        php             ;salva carry4
        clc             ;clear carry
        adc carry2      ;soma (w4+w6+w7) com
;(carry1+carry2)
        php             ;salva carry5
        sta result3l    ;armazena byte 3 resultado
        lda mascara     ;carrega A com mascara
        and carry3      ;mascara carry3
        sta carry3      ;salva carry3
        lda mascara     ;carrega A com mascara
        and carry4      ;mascara carry4
        sta carry4      ;salva carry4
        lda mascara     ;carrega A com mascara
        and carry5      ;mascara carry5
        sta carry5      ;salva carry5
        clc             ;clear carry
        pla             ;carrega A com carry5
        adc carry4      ;soma carry4 com carry5
        adc carry3      ;soma carry3 com (carry4+carry5)
        adc w8          ;soma w8 com soma dos carry
        sta result4l    ;armazema byte 4 resultado
;384*10000
        clc             ;clear carry
        lda result3l    ;carrega A com result3
        adc kl          ;soma com kl
        sta result3l    ;armazena em result3
        lda result4l    ;carrega A com result4
        adc kh          ;soma com kh
        sta result4l    ;armazema em result4
;
;calculo de tk
tk1:
        ldx stack1      ;carrega X com valor do stack=posicao
;de mul1h
        txs             ;carrega o stach pointer
        ldx zero        ;carrega X com zero
        clc             ;clear carry para adicao
        lda Vm          ;carrega acumulador com Vm
        adc Vo          ;Vm+Vo
        bcc bits8k      ;branch se nao houve carry
        jmp bits9k      ;jump se soma possui 9 bits
bits8k:mul Vk,x         ;(Vm+Vo)*Vk
        sta mul1l       ;armazena produto_low em mul1l
;(Vm+Vo)Vm*384
        ldx stack       ;carrega X com valor do stack
        txs             ;carrega o stack pointer
        ldx zero        ;carrega X
        lda mul1l       ;carrega o A com multiplicando low
        mul Kl,x        ;multiplica o A pelo multiplicador low
        sta result1k    ;armazema byte 1 resultado
        lda mul1h       ;carrega A com multiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador low
        sta result2k    ;armazema w3
        lda mul1l       ;carrega A com multiplicando low
        inx             ;incrementa X
        mul Kl,x        ;multiplica o A pelo mutiplicador high
```

```
        clc             ;clear carry
        adc result2k    ;adiciona w5 com w3
        php             ;salva o valor do carry1
        clc             ;clear o carry
        adc w2          ;adiciona w2 com o resultado de
;(w5+w3)
        php             ;salva o valor do carry2
        sta result2k    ;armazena byte 2 resultado
        lda mascara     ;carrega A com mascara
        and carry1      ;mascara carry1
        sta carry1      ;salva carry1
        lda mascara     ;carrega A com mascara
        and carry2      ;mascara carry2
        sta carry2      ;salva carry2
        clc             ;clear carry
        pla             ;carrega A com valor do carry2
        adc carry1      ;soma carry1 com carry2
        pha             ;salva carry1+carry2
        clc             ;clear carry
        lda mul1h       ;carrega A com mutiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;high
        adc w6          ;soma w7 com w6
        php             ;salva carry3
        clc             ;clear carry
        adc w4          ;soma w4 com (w7+w6)
        php             ;salva carry4
        clc             ;clear carry
        adc carry2      ;soma (w4+w6+w7) com
;(carry1+carry2)
        php             ;salva carry5
        sta result3k    ;armazena byte 3 resultado
        lda mascara     ;carrega A com mascara
        and carry3      ;mascara carry3
        sta carry3      ;salva carry3
        lda mascara     ;carrega A com mascara
        and carry4      ;mascara carry4
        sta carry4      ;salva carry4
        lda mascara     ;carrega A com mascara
        and carry5      ;mascara carry5
        sta carry5      ;salva carry5
        clc             ;clear carry
        pla             ;carrega A com carry5
        adc carry4      ;soma carry4 com carry5
        adc carry3      ;soma carry3 com (carry4+carry5)
        adc w8          ;soma w8 com soma dos carry
        sta result4k    ;armazema byte 4 resultado
        jmp tm1         ;va para calculo de tm
;
bits9k:mul Vk,x         ;(Vm+Vo)*Vk
        sta mul1l       ;armazena produto low em mul1l
;vk*100
        clc             ;clear carry
        lda mul1h       ;carrega A com byte mais significativo
        adc Vk          ;adiciona com Vl
        sta mul1h       ;armazena soma em mul1h
;(Vm+Vo)Vk*384
        ldx stack       ;carrega X com valor do stack
        txs             ;carrega o stack pointer
        ldx zero        ;carrega X
        lda mul1l       ;carrega o A com multiplicando low
        mul Kl,x        ;multiplica o A pelo multiplicador low
        sta result1k    ;armazema byte 1 resultado
        lda mul1h       ;carrega A com multiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador low
        sta result2k    ;armazema w3
        lda mul1l       ;carrega A com multiplicando low
        inx             ;incrementa X
        mul Kl,x        ;multiplica o A pelo mutiplicador
;high
        clc             ;clear carry
        adc result2k    ;adiciona w5 com w3
        php             ;salva o valor do carry1
        clc             ;clear o carry
        adc w2          ;adiciona w2 com o resultado de
;(w5+w3)
        php             ;salva o valor do carry2
        sta result2k    ;armazena byte 2 resultado
        lda mascara     ;carrega A com mascara
        and carry1      ;mascara carry1
        sta carry1      ;salva carry1
        lda mascara     ;carrega A com mascara
        and carry2      ;mascara carry2
        sta carry2      ;salva carry2
        clc             ;clear carry
        pla             ;carrega A com valor do carry2
        adc carry1      ;soma carry1 com carry2
        pha             ;salva carry1+carry2
        clc             ;clear carry
        lda mul1h       ;carrega A com mutiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;high
        adc w6          ;soma w7 com w6
        php             ;salva carry3
        clc             ;clear carry
        adc w4          ;soma w4 com (w7+w6)
        php             ;salva carry4
        clc             ;clear carry
        adc carry2      ;soma (w4+w6+w7) com
;(carry1+carry2)
        php             ;salva carry5
        sta result3k    ;armazena byte 3 resultado
        lda mascara     ;carrega A com mascara
        and carry3      ;mascara carry3
        sta carry3      ;salva carry3
        lda mascara     ;carrega A com mascara
        and carry4      ;mascara carry4
        sta carry4      ;salva carry4
        lda mascara     ;carrega A com mascara
        and carry5      ;mascara carry5
        sta carry5      ;salva carry5
        clc             ;clear carry
        pla             ;carrega A com carry5
        adc carry4      ;soma carry4 com carry5
        adc carry3      ;soma carry3 com (carry4+carry5)
        adc w8          ;soma w8 com soma dos carry
        sta result4k    ;armazena byte 4 resultado
tm1:
;calculo de tm
        sec             ;seta carry
        lda Tmaxlow     ;carrega A com E1h
        sbc result3k    ;E1h-tkl
        sta result3m    ;armazena resultado em result3m
        lda Tmaxhigh    ;carrega A com 04h
        sbc result4k    ;04h-tkh
        sta result4m    ;armazena resultado em result4m
        sec             ;seta carry
        lda result3m    ;carrega A com (E1h-tkl)
        sbc result3l    ;(E1h-tkl)-tll
        sta result3m    ;armazena resultado em result3m(tml)
        lda result4m    ;carrega A com (04h-tkl)
        sbc result4l    ;(04h-tkl)-tlh
        sta result4m    ;armazena resultado em result4m
;(tmh)
        ldx stacks2     ;carrega X com endereco de retorno
;da subrotina
        txs             ;trasnfere endereco para stack pointer
        rts             ;retorna da subrotina
;
;calt-
calcmenos:
;calculo de tl
        ldx stack1      ;carrega X com valor do
;stack=posicao de mul1h
        txs             ;carrega o stach pointer
        ldx zero        ;carrega X com zero
        sec             ;seta carry para subtracao
        lda Vm          ;carrega acumulador com Vm
        sbc Vo          ;Vm-Vo
```

```
        mul Vl,x        ;(Vm-Vo)*Vl
        sta mul1l       ;armazena produto_low em mul1l
;(Vm+Vo)Vl*384
        ldx stack       ;carrega X com valor do stack
        txs             ;carrega o stack pointer
        ldx zero        ;carrega X
        lda mul1l       ;carrega o A com multiplicando low
        mul Kl,x        ;multiplica o A pelo multiplicador
;low
        sta result1l    ;armazema byte 1 resultado
        lda mul1h       ;carrega A com multiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;low
        sta result2l    ;armazema w3
        lda mul1l       ;carrega A com multiplicando low
        inx             ;incrementa X
        mul Kl,x        ;multiplica o A pelo mutiplicador high
        clc             ;clear carry
        adc result2l    ;adiciona w5 com w3
        php             ;salva o valor do carry1
        clc             ;clear o carry
        adc w2          ;adiciona w2 com o resultado de
;(w5+w3)
        php             ;salva o valor do carry2
        sta result2l    ;armazena byte 2 resultado
        lda mascara     ;carrega A com mascara
        and carry1      ;mascara carry1
        sta carry1      ;salva carry1
        lda mascara     ;carrega A com mascara
        and carry2      ;mascara carry2
        sta carry2      ;salva carry2
        clc             ;clear carry
        pla             ;carrega A com valor do carry2
        adc carry1      ;soma carry1 com carry2
        pha             ;salva carry1+carry2
        clc             ;clear carry
        lda mul1h       ;carrega A com mutiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;high
        adc w6          ;soma w7 com w6
        php             ;salva carry3
        clc             ;clear carry
        adc w4          ;soma w4 com (w7+w6)
        php             ;salva carry4
        clc             ;clear carry
        adc carry2      ;soma (w4+w6+w7) com
;(carry1+carry2)
        php             ;salva carry5
        sta result3l    ;armazena byte 3 resultado
        lda mascara     ;carrega A com mascara
        and carry3      ;mascara carry3
        sta carry3      ;salva carry3
        lda mascara     ;carrega A com mascara
        and carry4      ;mascara carry4
        sta carry4      ;salva carry4
        lda mascara     ;carrega A com mascara
        and carry5      ;mascara carry5
        sta carry5      ;salva carry5
        clc             ;clear carry
        pla             ;carrega A com carry5
        adc carry4      ;soma carry4 com carry5
        adc carry3      ;soma carry3 com (carry4+carry5)
        adc w8          ;soma w8 com soma dos carry
        sta result4l    ;armazema byte 4 resultado
;
;calculo de tk
        ldx stack1      ;carrega X com valor do
;stack=posicao de mul1h
        txs             ;carrega o stach pointer
        ldx zero        ;carrega X com zero
        sec             ;seta carry para subtracao
        lda Vm          ;carrega acumulador com Vm
        sbc Vo          ;Vm-Vo
        mul Vk,x        ;(Vm-Vo)*Vk
        sta mul1l       ;armazena produto_low em mul1l
;(Vm+Vo)Vk*384
        ldx stack       ;carrega X com valor do stack
        txs             ;carrega o stack pointer
        ldx zero        ;carrega X
        lda mul1l       ;carrega o A com multiplicando
;low
        mul Kl,x        ;multiplica o A pelo multiplicador
;low
        sta result1k    ;armazema byte 1 resultado
        lda mul1h       ;carrega A com multiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;low
        sta result2k    ;armazema w3
        lda mul1l       ;carrega A com multiplicando low
        inx             ;incrementa X
        mul Kl,x        ;multiplica o A pelo mutiplicador
;high
        clc             ;clear carry
        adc result2k    ;adiciona w5 com w3
        php             ;salva o valor do carry1
        clc             ;clear o carry
        adc w2          ;adiciona w2 com o resultado de
;(w5+w3)
        php             ;salva o valor do carry2
        sta result2k    ;armazena byte 2 resultado
        lda mascara     ;carrega A com mascara
        and carry1      ;mascara carry1
        sta carry1      ;salva carry1
        lda mascara     ;carrega A com mascara
        and carry2      ;mascara carry2
        sta carry2      ;salva carry2
        clc             ;clear carry
        pla             ;carrega A com valor do carry2
        adc carry1      ;soma carry1 com carry2
        pha             ;salva carry1+carry2
        clc             ;clear carry
        lda mul1h       ;carrega A com mutiplicando high
        mul Kl,x        ;multiplica o A pelo multiplicador
;high
        adc w6          ;soma w7 com w6
        php             ;salva carry3
        clc             ;clear carry
        adc w4          ;soma w4 com (w7+w6)
        php             ;salva carry4
        clc             ;clear carry
        adc carry2      ;soma (w4+w6+w7) com
;(carry1+carry2)
        php             ;salva carry5
        sta result3k    ;armazena byte 3 resultado
        lda mascara     ;carrega A com mascara
        and carry3      ;mascara carry3
        sta carry3      ;salva carry3
        lda mascara     ;carrega A com mascara
        and carry4      ;mascara carry4
        sta carry4      ;salva carry4
        lda mascara     ;carrega A com mascara
        and carry5      ;mascara carry5
        sta carry5      ;salva carry5
        clc             ;clear carry
        pla             ;carrega A com carry5
        adc carry4      ;soma carry4 com carry5
        adc carry3      ;soma carry3 com (carry4+carry5)
        adc w8          ;soma w8 com soma dos carry
        sta result4k    ;armazema byte 4 resultado
;calculo de tm
        sec             ;seta carry
        lda Tmaxlow     ;carrega A com E1h
        sbc result3k    ;E1h-tkl
        sta result3m    ;armazena resultado em result3m
        lda Tmaxhigh    ;carrega A com 04h
        sbc result4k    ;04h-tkh
        sta result4m    ;armazena resultado em result4m
        sec             ;seta carry
        lda result3m    ;carrega A com (E1h-tkl)
        sbc result3l    ;(E1h-tkl)-tll
```

```
        sta result3m        ;armazena resultado em
;result3m(tml)
        lda result4m        ;carrega A com (04h-tkl)
        sbc result4l        ;(04h-tkl)-tlh
        sta result4m        ;armazena resultado em result4m(tmh)
        ldx stacks2         ;carrega X com endereco de retorno da
;subrotina
        txs                 ;transfere endereco para stack pointer
        rts                 ;retorno da subrotina
;
cont1:
        cli                 ;habilita interrupcoes
        lda y               ;e a primeira vez ?(y=0)
        bne n_inicioa       ;se nao for va para nao inicioa
        ldm #$81,ICON1;habilita interrupcao timer3 e buffer
;cheio
g1:     lda fstop           ;fstop=00? Se sim, fica em loop
        beq g1
        ldm #$01,ICON1      ;desabilita interrupcao timer3
        lda result3l        ;carrega T1LL
        sta T1LL
        lda result4l        ;carrega T1LH
        sta T1LH
        lda result3k        ;carrega T2LL
        sta T2LL
        lda result4k        ;carrega T2LH
        sta T2LH
        lda result3m        ;carrega T3LL
        sta T3LL
        lda result4m        ;carrega  T3LH
        sta T3LH
        ldm #$ff,y          ;y=ffh, nao e a primeira vez
        ldm #$70,MISRG1 ;ativa contagem dos timers
        ldm #$19,ICON1  ;habilita int2 e int3
        seb 2,P3            ;provoca um pulso em P60
        clb 2,P3
        ldm #$06,T3C    ;timer 3 em one shot
        ldm #$1d,ICON1  ;habilita todas as INTs
        sei
        rts                 ;fim rotina
;
n_inicioa: lda result3l     ;carrega T1LL
        sta T1LL
        lda result4l        ;carrega T1LH
        sta T1LH
;
f2a:    lda ftimer2         ;timer1 terminou contagem?
;(ftimer=FFh)
        beq f2a             ;se nao, va para f2a
        lda result3k        ;carrega T2LL
        sta T2LL
        lda result4k        ;carrega T2LH
        sta T2LH
        ldm #$00,ftimer2    ;faz ftimer2=zero
;
f3a:    lda ftimer3         ;timer2 terminou contagem?
;(ftimer=FFh)
        beq f3a             ;se nao, va para f3a
        lda result3m        ;carrega T3LL
        sta T3LL
        lda result4m        ;carrega T3LH
        sta T3LH
        ldm #$00,ftimer3    ;faz ftimer3=zero
f4a:    lda ftimer1         ;timer3 terminou contagem?
;(ftimer=FFh)
        beq f4a             ;se nao va para f4a
        ldm #$00,ftimer1    ;faz ftimer1=zero
        sei
        rts                 ;fim rotina
;
;
cont2:
        cli                 ;habilita interrupcoes
        lda y               ;e a primeira vez ?(y=0)
        bne n_inicioe       ;se nao for va para nao inicioe
        ldm #$81,ICON1  ;habilita interrupcao timer3 e buffer
;cheio
g2:     lda fstop           ;fstop=00? Se sim, fica em loop
        beq g2
        ldm #$01,ICON1  ;desabilita interrupcao timer3
        lda result3k        ;carrega  T1LL
        sta T1LL
        lda result4k        ;carrega  T1LH
        sta T1LH
        lda result3l        ;carrega  T2LL
        sta T2LL
        lda result4l        ;carrega  T2LH
        sta T2LH
        lda result3m        ;carrega  T3LL
        sta T3LL
        lda result4m        ;carrega  T3LH
        sta T3LH
        ldm #$ff,y          ;y=FFh, nao e a primeira vez
        ldm #$70,MISRG1 ;ativa contagem dos timers
        ldm #$19,ICON1  ;ativa INT 2 e 3
        seb 2,P3            ;provoca um pulso em P60
        clb 2,P3
        ldm #$06,T3C    ;timer3 em one shot
        ldm #$1d,ICON1  ;habilita todas as INTs
        sei
        rts                 ;fim rotina
;
n_inicioe: lda result3k     ;carrega T1LL
        sta T1LL
        lda result4k        ;carrega T1LH
        sta T1LH
;
f2e:    lda ftimer2         ;timer1 terminou contagem?
;(ftimer=FFh)
        beq f2e             ;se nao, va para f2e
        lda result3l        ;carrega T2LL
        sta T2LL
        lda result4l        ;carrega T2LH
        sta T2LH
        ldm #$00,ftimer2    ;faz ftimer2=zero
;
f3e:    lda ftimer3         ;timer2 terminou contagem?
;(ftimer=FFh)
        beq f3e             ;se nao, va para f3e
        lda result3m        ;carrega T3LL
        sta T3LL
        lda result4m        ;carrega T3LH
        sta T3LH
        ldm #$00,ftimer3    ;faz ftimer3=zero
f4e:    lda ftimer1         ;timer3 terminou contagem?
;(ftimer=FFh)
        beq f4e             ;se nao, va para f4e
        ldm #$00,ftimer1    ;faz ftimer1=zero
        sei
        rts                 ;fim da rotina
;
cont3:
        cli                 ;habilita interrupcoes
        lda y               ;e a primeira vez ?(y=0)
        bne n_inicioi       ;se nao for va para nao inicioi
        ldm #$81,ICON1      ;habilita interrupcao timer3 e
;buffer cheio
g3:     lda fstop           ;fstop=00? Se sim, fica em loop
        beq g3
        ldm #$01,ICON1      ;desabilita interrupcao timer3
        lda result3l        ;carrega  T1LL
        sta T1LL
        lda result4l        ;carrega T1LH
        sta T1LH
        lda result3m        ;carrega T2LL
        sta T2LL
        lda result4m        ;carrega T2LH
        sta T2LH
        lda result3k        ;carrega T3LL
        sta T3LL
```

```
        lda result4k            ;carrega T3LH
        sta T3LH
        ldm #$ff,y              ;y=FFh, nao e a primeira vez
        ldm #$70,MISRG1         ;ativa contagem dos timers
        ldm #$19,ICON1          ;ativa INT 2 e 3
        seb 2,P3                ;provoca um pulso em P60
        clb 2,P3
        ldm #$06,T3C            ;timer3 em one shot
        ldm #$1d,ICON1          ;habilita todas as INTs
        sei
        rts                     ;fim rotina
;
n_inicioi: lda result3l         ;carrega T1LL
        sta T1LL
        lda result4l        ;carrega T1LH
        sta T1LH
;
f2i:    lda ftimer2         ;timer1 terminou contagem?
;(ftimer=FFh)
        beq f2i             ;se nao, va para f2i
        lda result3m        ;carrega T2LL
        sta T2LL
        lda result4m        ;carrega T2LH
        sta T2LH
        ldm #$00,ftimer2    ;faz ftimer2=zero
;
f3i:    lda ftimer3         ;timer2 terminou contagem?
;(ftimer=FFh)
        beq f3i             ;se nao, va para f3i
        lda result3k        ;carrega T3LL
        sta T3LL
        lda result4k        ;carrega T3LH
        sta T3LH
        ldm #$00,ftimer3    ;faz ftimer3=zero
f4i:    lda ftimer1         ;timer3 terminou contagem?
;(ftimer-;FFh)
        beq f4i             ;se nao, va para f4i
        ldm #$00,ftimer1    ;faz ftimer1=zero
        sei
        rts                 ;fim rotina
;
cont4:
        cli                 ;habilita interrupcoes
        lda y               ;e a primeira vez ?(y=0)
        bne n_iniciom       ;se nao for va para nao iniciom
        ldm #$81,ICON1      ;habilita interrupcao timer3 e
;buffer cheio
g4:     lda fstop           ;fstop=00? Se sim, fica em loop
        beq g4
        ldm #$01,ICON1      ;desabilita interrupcao timer3
        lda result3k        ;carrega T1LL
        sta T1LL
        lda result4k        ;carrega T1LH
        sta T1LH
        lda result3m        ;carrega T2LL
        sta T2LL
        lda result4m        ;carrega T2LH
        sta T2LH
        lda result3l        ;carrega T3LL
        sta T3LL
        lda result4l        ;carrega T3LH
        sta T3LH
        ldm #$ff,y          ;y=FFh,nao e a primeira vez
        ldm #$70,MISRG1     ;ativa contagem dos timers
        ldm #$19,ICON1      ;habilita INT 2 e 3
        seb 2,P3            ;provoca um pulso em P60
        clb 2,P3
        ldm #$06,T3C        ;timer3 em one shot
        ldm #$1d,ICON1      ;habilita todas as INTs
        sei
        rts                 ;fim rotina
;
n_iniciom: lda result3k     ;carrega T1LL
        sta T1LL
        lda result4k        ;carrega T1LH
        sta T1LH
;
f2m:    lda ftimer2         ;timer1 terminou contagem?
;(ftimer=FFh)
        beq f2m             ;se nao, va para f2m
        lda result3m        ;carrega T2LL
        sta T2LL
        lda result4m        ;carrega T2LH
        sta T2LH
        ldm #$00,ftimer2    ;faz ftimer2=zero
f3m:    lda ftimer3         ;timer2 terminou contagem?
;(ftimer=FFh)
        beq f3m             ;se nao, va para f3m
        lda result3l        ;carrega T3LL
        sta T3LL
        lda result4l        ;carrega T3LH
        sta T3LH
        ldm #$00,ftimer3    ;faz ftimer3=zero
f4m:    lda ftimer1         ;timer3 terminou contagem?
;(ftimer=FFh)
        beq f4m             ;se nao, va para f4m
        ldm #$00,ftimer1    ;faz ftimer1=zero
        sei
        rts                 ;fim rotina
;
cont5:
        cli                 ;habilita interrupcoes
        lda y               ;e a primeira vez ?(y=0)
        bne n_inicioq       ;se nao for va para nao inicioq
        ldm #$81,ICON1      ;habilita interrupcao timer3 e
;buffer cheio
g5:     lda fstop           ;fstop=00? Se sim, fica em loop
        beq g5
        ldm #$01,ICON1      ;desabilita interrupcao timer3
        lda result3m        ;carrega  T1LL
        sta T1LL
        lda result4m        ;carrega T1LH
        sta T1LH
        lda result3l        ;carrega T2LL
        sta T2LL
        lda result4l        ;carrega T2LH
        sta T2LH
        lda result3k        ;carrega T3LL
        sta T3LL
        lda result4k        ;carrega T3LH
        sta T3LH
        ldm #$ff,y          ;y=FFh, nao e a primeira vez
        ldm #$70,MISRG1     ;ativa contagem dos timers
        ldm #$19,ICON1      ;habilita INT 2 e 3
        seb 2,P3            ;provoca um pulso em P60
        clb 2,P3
        ldm #$06,T3C        ;timer3 em one shot
        ldm #$1d,ICON1      ;habilita todas as INTs
        sei
        rts                 ;fim rotina
;
n_inicioq: lda result3m     ;carrega T1LL
        sta T1LL
        lda result4m        ;carrega T1LH
        sta T1LH
f2q:    lda ftimer2         ;timer1 terminou contagem?
;(ftimer=FFh)
        beq f2q             ;se nao, va para f2q
        lda result3l        ;carrega T2LL
        sta T2LL
        lda result4l        ;carrega T2LH
        sta T2LH
        ldm #$00,ftimer2    ;faz ftimer=zero
f3q:    lda ftimer3         ;timer2 terminou contagem?
;(ftimer=FFh)
        beq f3q             ;se nao, va para f3q
        lda result3k        ;carrega T3LL
        sta T3LL
        lda result4k        ;carrega T3LH
        sta T3LH
```

```
        ldm #$00,ftimer3        ;faz ftimer3=zero
f4q:    lda ftimer1             ;timer3 terminou contagem?
;(ftimer=FFh)
        beq f4q                 ;se nao, va para f4q
        ldm #$00,ftimer1        ;faz ftimer=zero
        sei
        rts                     ;fim rotina
;
cont6:
        cli             ;habilita interrupcoes
        lda y           ;e a primeira vez ?(y=0)
        bne n_iniciox   ;se nao for va para nao iniciox
        ldm #$81,ICON1  ;habilita interrupcao timer3 e
;buffer cheio
g6:     lda fstop       ;fstop=00? Se sim, fica em loop
        beq g6
        ldm #$01,ICON1  ;desabilita interrupcao timer3
        lda result3m    ;carrega T1LL
        sta T1LL
        lda result4m    ;carrega T1LH
        sta T1LH
        lda result3k    ;carrega T2LL
        sta T2LL
        lda result4k    ;carrega T2LH
        sta T2LH
        lda result3l    ;carrega T3LL
        sta T3LL
        lda result4l    ;carrega T3LH
        sta T3LH
        ldm #$ff,y      ;y=FFh, nao e a primeira vez
        ldm #$70,MISRG1 ;ativa contagem dos timers
        ldm #$19,ICON1  ;habilita INT 2 e 3
        seb 2,P3        ;provoca um pulso em P60
        clb 2,P3
        ldm #$06,T3C    ;timer3 em one shot
        ldm #$1d,ICON1  ;habilita todas as INTs
        sei
        rts             ;fim rotina
;
n_iniciox: lda result3m ;carrega T1LL
        sta T1LL
        lda result4m    ;carrega T1LH
        sta T1LH
;
f2x:    lda ftimer2     ;timer1 terminou contagem?
;(ftimer=FFh)
        beq f2x         ;se nao, va para f2x
        lda result3k    ;carrega T2LL
        sta T2LL
        lda result4k    ;carrega T2LH
        sta T2LH
        ldm #$00,ftimer2 ;faz ftimer2=zero
;
f3x:    lda ftimer3     ;timer2 terminou contagem?
;(ftimer=FFh)
        beq f3x         ;se nao, va para f3x
        lda result3l    ;carrega T3LL
        sta T3LL
        lda result4l    ;carrega T3LH
        sta T3LH
        ldm #$00,ftimer3 ;faz ftimer3=zero
f4x:    lda ftimer1     ;timer3 terminou contagem?
;(ftimer=FFh)
        beq f4x         ;se nao, va para f4x
        ldm #$00,ftimer1 ;faz ftimer1=zero
        sei
        rts             ;fim rotina
;
;rotina de interrupcao timer1
;
int1:
        ldm #$ff,ftimer1 ;carrega ftimer1 com FFh
        rti
;
;rotina de interrupcao timer2
;
int2:
        ldm #$ff,ftimer2 ;carrega ftimer2 com FFh
        rti
;
;rotina de interrupcao timer3
;
int3:
        ldm #$ff,ftimer3 ;carrega ftimer3 com FFh
        rti
;
;rotina de interrupcao de buffer de interface cheio
;
cheio:  bbs 3,IBSTAT,amp ;se dado recebido for amplitude,
;va para amp
        pha             ;salva A
        lda IBR         ;carrega A com valor da
;frequencia
        sta freq        ;armazema valor em freq
        lda #$ff        ;faz stop1 = FFh - freq
        sec
        sbc freq
        sta stop1
        pla             ;recupera A
        rti
amp:    pha             ;salva A
        lda IBR         ;carrega A com valor da
;amplitude
        sta ampl        ;armazema valor em ampl
        pla             ;recupera A
        rti
;
iTimer3: dm #$ff,fstop
        rti
;vetores de interrupca
;
ORG   $ffe0
;
    DWL start1          ;instrucao BRK
    DWL start1          ;flag de conversao AD completa
    DWL start1          ;transmissao serial
    DWL start1          ;recepcao serial
    DWL start1          ;EV3
    DWL start1          ;EV2
    DWL start1          ;EV1
    DWL iTimer3         ;Timer3
    DWL start1          ;Timer2
    DWL start1          ;Timer1
    DWL int3            ;INT3
    DWL int2            ;INT2
    DWL int1            ;INT1
    DWL start1          ;buffer de saida vazio
    DWL cheio           ;buffer de entrada cheio
    DWL start1          ;reset
;
    END
```

```
;Programa que obtem o valor de pico das senoides de entrada e
;o envia para os outros coprocessadores
;
CPU "M450.TBL"
HOF "INT8"
;
;Enderecos do 37450
;
P3:       EQU $D6       ;registro P3
P3D:      EQU $D7       ;registro de direcao P3
P4:       EQU $D8       ;registro P4
P5:       EQU $DA       ;registro P5
P5D:      EQU $DB       ;registro de direcao P5
P6:       EQU $DC       ;registro P6
P6D:      EQU $DD       ;registro de direcao P6
MISRG1:   EQU $DE       ;registro de miscelanea 1
MISRG2:   EQU $DF       ;registro de miscelanea 2
DA1R:     EQU $E0       ;registro do DAC 1
DA2R:     EQU $E1       ;registro do DAC 2
ADR:      EQU $E2       ;registro do ADC
ADCR:     EQU $E3       ;registro de controle do ADC

IBR:      EQU $E4       ;registro do data bus
IBSTAT:   EQU $E5       ;registro de estado do data bus

RTBR:     EQU $E6       ;registro do buffer de
;recepcao/transmissao
SIOS:     EQU $E7       ;registro de estado da I/O serial
SIOC:     EQU $E8       ;registro de controle da I/O serial
UACON:    EQU $E9       ;registro de controle UART
BRG:      EQU $EA       ;gerador taxa baud

PWML:     EQU $EB       ;registro PWM low
PWMH:     EQU $EC       ;registro PWM high

TC1:      EQU $ED       ;registro de controle timer 1
TC2:      EQU $EE       ;registro de controle timer 2
TC3:      EQU $EF       ;registro de controle timer 3
T1L:      EQU $F0       ;registro timer 1 low
T1H:      EQU $F1       ;registro timer 2 high
T1LL:     EQU $F2       ;latch timer 1 low
T1LH:     EQU $F3       ;latch timer 1 high
T2L:      EQU $F4       ;registro timer 2 low
T2H:      EQU $F5       ;registro timer 2 high
T2LL:     EQU $F6       ;latch timer 2 low
T2LH:     EQU $F7       ;latch timer 2 high
T3L:      EQU $F8       ;registro timer 3 low
T3H:      EQU $F9       ;registro timer 3  high
T3LL:     EQU $FA       ;latch 3 low
T3LH:     EQU $FB       ;latch 3 high

IRQ1:     EQU $FC       ;registro requisicao interrupcao 1
IRQ2:     EQU $FD       ;registro requisicao interrupcao 2
ICON1:    EQU $FE       ;registro controle interrupcao 1
ICON2:    EQU $FF       ;regidtro controle interrupcao 2
;
ORG    $4000
;
reset:  clt             ;clear o bit t
        cld             ;clear o bit d
        clb 0,P3D
        lda #$ff        :coloca 5 V em DA2R
        sta DA2R
loop:   clb 0,IRQ2      ;espera sinal de sincronismo
l0:     bbc 0,IRQ2,l0
        clb 0,IRQ2
f0:     bbc 0,IRQ2,f0
        ldy #6d         ;espera 30 graus
b:      ldx #71d
a:      dex
        bne a
        dey
        bne b
s:      ldm #$00,ADCR   ;ler Va
        clb 5,IRQ2
c1:     lda IRQ2
        and #$20
        beq c1
        clb 5,IRQ2
        lda ADR
        sta DA2R        ;envia Vp
;
        ldy #7d         ;espera 60 graus
d:      ldx #174d
c:      dex
        bne c
        dey
        bne d
        ldm #$02,ADCR   ;ler Vc
c2:     lda IRQ2
        and #$20
        beq c2
        clb 5,IRQ2
        lda ADR
        sta DA2R        ;envia Vp
;
        ldy #7d         ;espera 60 graus
f:      ldx #174d
e:      dex
        bne e
        dey
        bne f
        ldm #$01,ADCR   ;ler Vb
c3:     lda IRQ2
        and #$20
        beq c3
        clb 5,IRQ2
        lda ADR
        sta DA2R        ;envia Vp
;
        ldy #7d         ;espera 60 graus
h:      ldx #174d
g:      dex
        bne g
        dey
        bne h
        ldm #$00,ADCR   ;ler Va
c4:     lda IRQ2
        and #$20
        beq c4
        clb 5,IRQ2
        lda ADR
        sta DA2R        ;envia Vp
;
        ldy #7d         ;espera 60 graus
j:      ldx #174d
i:      dex
        bne i
        dey
        bne j
        ldm #$02,ADCR   ;ler Vc
c5:     lda IRQ2
        and #$20
        beq c5
        clb 5,IRQ2
        lda ADR
        sta DA2R        ;envia Vp
;
        ldy #7d         ;espera 60 graus
m:      ldx #174d
l:      dex
        bne l
        dey
        bne m
        ldm #$01,ADCR   ;ler Vb
c6:     lda IRQ2
        and #$20
        beq c6
        clb 5,IRQ2
        lda ADR
```

```
        sta DA2R        ;envia Vp
;
        jmp loop
;
;vetores de interrupcao

ORG    $ffe0

      DWL  reset        ;instrucao BRK
      DWL  reset        ;flag de conversao AD completa
      DWL  reset        ;transmissao serial
      DWL  reset        ;recepcao serial
      DWL  reset        ;EV3
      DWL  reset        ;EV2
      DWL  reset        ;EV1
      DWL  reset        ;Timer1
      DWL  reset        ;Timer2
      DWL  reset        ;Timer3
      DWL  reset        ;INT3
      DWL  reset        ;INT2
      DWL  reset        ;INT1
      DWL  reset        ;buffer de saida vazio
      DWL  reset        ;buffer de entrada cheio
      DWL  reset        ;reset

      END
```

Rotinas *Forth* para os processadores

a) Sincroniza os coprocessadores

```
: sinc
  dec all-off all-on ;
```

b) Envia o valor da amplitude das tensões de saída aos coprocessadores

```
: ampl
  u>d d>f
  fdup 48, f<= if else 48, then
  fdup 2, f<= if  2,      then
  100, f/ 256, f*
  f>d dup dup
  0 >cp'
  1 >cp'
  2 >cp'
;
```

c) Envia o valor da freqüência das tensões de saída aos coprocessadores

```
: freq
  2 / dup dup
  0 >cp
  1 >cp
  2 >cp
;
```

d) Envia o valor da freqüência e da amplitude aos coprocessadores

```
: seno
  2 / dup dup
  0 >cp
  1 >cp
  2 >cp
  u>d d>f
  fdup 48, f<= if else 48, then
  fdup 2, f<= if  2,      then
  100, f/ 256, f*
  f>d dup dup
  0 >cp'
  1 >cp'
  2 >cp'
;
```

FOTOS APÊNDICE V

FOTOS DO SISTEMA DE CONTROLE