GERADOR DE FORMA DE ONDA ARBITRÁRIA POR SÍNTESE DIGITAL

Fernando António Castilho Mamede dos Santos



Faculdade de Engenharia da Universidade do Porto

Porto, Julho 1991

Bes. 590 ElecV

GERADOR DE FORMA DE ONDA ARBITRÁRIA POR SINTESE DIGITAL

Fernando António Castilho Mamede dos Santos
Licenciado em Engenharia Electrotécnica pela Faculdade de Engenharia da Universidade do Porto

Tese submetida para satisfação parcial dos requisitos do programa de Mestrado em Engenharia Electrotécnica e de Computadores

UNIVERSIDADE DO PORTO
Faculdade de Engenharia
BIBLIOTECA M
N.º 20651-gf
CDU 681.3(043)
Data 5/6/1995
N.N.34787

Faculdade de Engenharia da Universidade do Porto

Porto, Julho 1991

043 M
S 235 g
ex. 2

Tese realizada sob a supervisão do Doutor Artur Capelo Cardoso, Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

RECONHECIMENTO

Por que o homem é um ser social que deve viver simbiose, levava-nos a pensar, até à pouco tempo, da razão pela qual num trabalho profissional se podia recordar a memória do Pai, do exemplo da Mãe, a compreensão da Mulher, a desculpa dos filhos, ou a gratidão para com terceiros.

No entanto, sob pena de cometer o erro de omissão, quero deixar expresso os meus agradecimentos:

Aos colegas da Escola Superior de Tecnologia de Viseu,
Engenheiros,

Francisco Ferreira Francisco

e

Luís António Ferreira do Amaral

Ao Sr. Presidente da Comissão Instaladora,
Dr. António Soares de Sousa

Ao Sr. Técnico Adjunto Especialista de Física,
José Gaspar

Aos colegas da Universidade do Porto,
António Manuel Lucas Soares e Raúl Teixeira de Oliveira

Ao Professor Doutor José Carlos Marques dos Santos

Em especial ao Sr. Professor Doutor Artur Capelo Cardoso, por ter supervisionado este trabalho.

ÍNDICE

Índice I

Lista de figuras III

Lista de tabelas V

Lista de programas VI

Bibliografias VII

Catálogos IX

Manuais X

"Standards" X

Preâmbulo 1

1- Aplicações dos geradores de onda arbitrários 2

2 - Frequência de amostragem 4

2.1 - Introdução 4

2.2 - Definição do número de amostras por ciclo de sinal. 4

3 - Interface digital-analógica 8

3.1 - Introdução 8

3.2 - Conversão digital analógica 9

3.2.1 - Métodos interpolativos 10

3.2.2 - Elementos do circuito 14

3.2.3 - Modo de funcionamento 15

3.3 - Controlo de variação de amplitude 18

3.4 - Filtragem 19

3.4.1 - Escolha do tipo de filtro a utilizar 19

3.4.2 - Obtenção de relógio de comutação do filtro 20

3.4.3 - Dimensionamento do filtro 22

4 - "Interface" entre o microprocessador e o BUS IEEE 488 24
4.1 - Introdução 24
4.2 - Elementos do circuíto 25
4.3 - Modo de funcionamento 27
4.4 - Software de controlo 28
4.4.1 - Rotina de inicialização 28
4.4.2 - Rotina de Interrupção do Z80 30
4.4.3 - Rotina de transferência de dados entre o GPIB e Z80 31

5 - Software do gerador de forma de onda 33
5.1 - Introdução 33
5.2 - Software 33
5.3 - Estudo da temporização do programa 35
5.3.1 - Estudo da temporização sem atraso 35
5.3.2 - Estudo da temporização com atraso 35
5.3.3 - Verificação experimental da temporização com atraso 36

6 - Interface com o utilizador 37
6.1 - Introdução 37
6.2 - Interface entre a aplicação e o GPIB 37
6.3 - Interface entre o utilizador e a aplicação 39

Apêndice A: Tabelas 50
Apêndice B: Esquemas eléctricos 56
Apêndice C: Características das placas de interface 58
Apêndice D: Programas 77

LISTA DAS FIGURAS

Figuras:

P.1: Diagrama de blocos do gerador 1

1.1(a): Ciclo de batimento cardíaco 3

1.1(b): Reconstrução do ciclo de batimento cardíaco 3

2.1: Cálculo do número de amostras 5

2.2: Construção de uma sinusoide com $f_a = f_{sinal}$ 6

2.3: Precisão da conversão versus número de amostras 7

3.1: Diagrama de blocos da interface digital analógica 8

3.2: Esquema do método interpolativo básico 10

3.3(a): Onda sinusoidal obtida pelo método SWS 11

3.3(b): Onda sinusoidal obtida pelo método interpolativo básico 12

3.3(c): Onda sinusoidal obtida pelo método interpolativo desenvolvido 13

3.4: Diagrama de blocos do sintetizador de formas de onda 14

3.5: Formato dos dados 15

3.6: Estrutura do controlo da entrada de dados no AD 7548 16

3.7(a): Circuito conversor de corrente tensão 17

3.7(b): Circuito de offset 17

3.8: Esquema eléctrico do controlo de atenuação 18

3.9(a): Sinal de relógio disponível no conversor 21

3.9(b): Sinal de relógio com duty cicle de 50% 21

3.10(a): Onda sinusoidal 22

3.10(b): Onda triangular 23

4.1: Diagrama de blocos da arquitectura da "interface" Micro, Bus IEEE 488 25

4.2: Desenho esquemático do "interface" Micro, Bus IEEE 488 26

4.3: Fluxograma da rotina de inicialização do TMS 29

4.4: Fluxograma da rotina de interrupção do Z80 30

4.5: Fluxograma da rotina de transferência de dados entre o GPIB e Z80 32

5.1: Fluxograma da rotina geradora de formas de onda 34

6.1: Estrutura do software NI-488 38

6.2: Menu do utilizador 39

6.3: Função leitura 40

6.4: Quadro de valores da função logaritmo 41

6.5: Função Zoom 42

6.6: Função setup 44

6.7: Menu de opcções da forma de onda 48

6.8: Mensagem de erro de transmissão 49

B.1 - Esquema eléctrico do conversor D/A 56

B.2 - Esquema eléctrico do filtro 57

C.1 - Endereços relativos da base de I/O 2B8 61

C.2 - Arquitectura interna do TMS 62

C.3 - Terminais 63

LISTA DE TABELAS

TABELAS

A.1 - Código de offset 50

A.2 - Código de controlo de variação de amplitude 50

A.3 - Valores dos registos para iniciliazação do TMS 50

A.4 - Valores digitais de formas de onda predefinidas 51

A.5 - Relação experimental entre os registos C e D,
e os valores de frequência obtidos 53

A.6 - Registos do TMS 55

LISTA DOS PROGRAMAS

D.1 - Programa de inicialização do TMS 77

D.2 - Programa de interrupção do Z80 78

D.3 - Programa de transferência de dados entre o GPIB e Z80 79

D.4 - Software gerador de forma de onda 80

D.5 - Programa de interacção 84

BIBLIOGRAFIA

1 - Aaron M. Tenenbaum, Moshe J. Augenstein, "Data Structures Using Pascal", Mac Graw Hill, 1986.

2 - Constantin H. Houpis, Gary B. Lamont, "Digital Control Sistems. Theory", Hardware and Software - Mac Graw Hill, 1985.

3 - Brian Brighouse, George Loveday, "Microprocessors in Enginneering Sistems", Pitman, 1987.

4 - D. P. Burton, A. L. Dexter, "Microprocessor Systems Hand Book", Analog Devices, 1979.

5 - David F. Stout, "Hand Book of Operational Amplifiers Circuit design", Mac Graw Hill, 1976.

6 - David Lalond, "The 8080, 8085, and Z80. Hardware, software, programming, interfacing and troubleshouting", Prentice Hall, 1988.

7 - David R. Barsctow, Moward E. Shrobe, Erick Sandewall, "Interactive Programming Environments", International Editor, 1986.

8 - Donald D. Givone, Robert P. Poesser, "Microprocessors, Microcomputers: An Introduction", Mac Graw Hill, 1980.

9 - Donald Hearn, M. Pauline Baker, "Graphiques Sur Votre Microordinateur", Prentice Hall, 1984.

10 - George Loveday, "Microprocessor Source Book", Pitman, 1986.

11 - Herbert Schildt, Advance Turbo Pascal Programming and Techniques", Mac Graw Hill, 1986.

12 - Jim Welsh, John Elder, "Introduction to Pascal", Prentice Hall Internactional, 1982.

13 - Joaquim Moura Relvas, "Introdução aos Microcomputadores", Figueirinhas, 1983.

14 - Kris Jamsa, Steven Nameroff, "Turbo Pascal Programmer's Library", Mac Graw Hill, 1987.

15 - R. Dony, "Graphisme Cientifique sur Microordinateure", Masson, 1988.

16 - Sergio Franco, " Design With Operational Amplifiers, and Analog Integrated Circuits", Mac Graw Hill, 1988.

17 - Stephen K. O'Brien, " Turbo Pascal: the complete reference", Mac Graw Hill 1989.

18 - Steven Harringhon, "Computer Graphics: A Programming Approach", Mac Graw Hill, 1985.

19 - Steven Wood, "Using Turbo Pascal", Mac Graw Hill, 1988.

20 - Tobey, Graem, Hulsman, "Operational Amplifiers, Design and Applications", Mac Graw Hill, 1986.

21 - Tom Rugg, Phil Feldman, "Turbo Pascal, Tips, Triks, and Traps", Que TH Corporation, 1987.

CATÁLOGOS

1 - Analog Integrated Circuits - Volume 10 - PMI

2 - Analog product (Industrial/Comercial - 1989) - Harris

3 - Analog Product (military - 1989) - Harris

4 - Integrated Circuits - Volume 33 - Burr Brown

5 - Linear 2 - National Semiconductor Corporation.

6 - Linear Circuits - Volume 1 - Texas Instruments.

7 - Linear Circuits - Volume 2 - Texas Instruments.

8 - LS/S/TLL - National Semiconductor Corporation.

9 - Non - Volatile Memory - National Semiconducter Corporation.

10 - TTL - Volume 1 - Texas Intruments

11 - TTL - Volume 2 - Texas Instruments

12 - 1990 Linear - Linear Technology

MANUAIS

1 - Data Aquisition and conversion Handbook - Intersil.

2 - NI-488 MS-DOS Software Reference Manual - National Instruments Corporation.

3 - Technical Reference Manual - Amstrad.

4 - TMS 9914 General prupose Interface Bus (GPIB) Controler Data Manual - Texas Instrumens.

5 - User Instructions - Amstrad.

6 - The Handbook of Linear IC applications Burr-Brown

STANDARD

1 - Standard IEEE 488.1

2 - Standard IEEE 488.2

PREÂMBULO

O presente trabalho correspondeu à execução de um gerador de onda arbitrária por síntese digital, o qual funciona de forma autónoma para formas de onda predefinidas, ou por controlo remoto utilizando um PC. Através do controlo remoto, que é interactivo com o utilizador, é possível definir qualquer tipo de forma de onda, ou seleccionar uma das formas de onda predefenidas. O diagrama de blocos da figura P.1 representa o gerador.

Tendo presente os objectivos acima descritos e o modo de transferência de dados realizado pelo "Bus Standard IEEE 488", desenvolveu-se:

A nível de Hardware, uma placa de interface entre o gerador de onda e o "Bus IEEE 488" e uma placa de interface digital-analógica.

A nível de software, um programa em Turbo Pascal, que faz o interface entre o PC e utilizador e um programa em "assembler" que faz a gestão do microprocessador do gerador de onda.

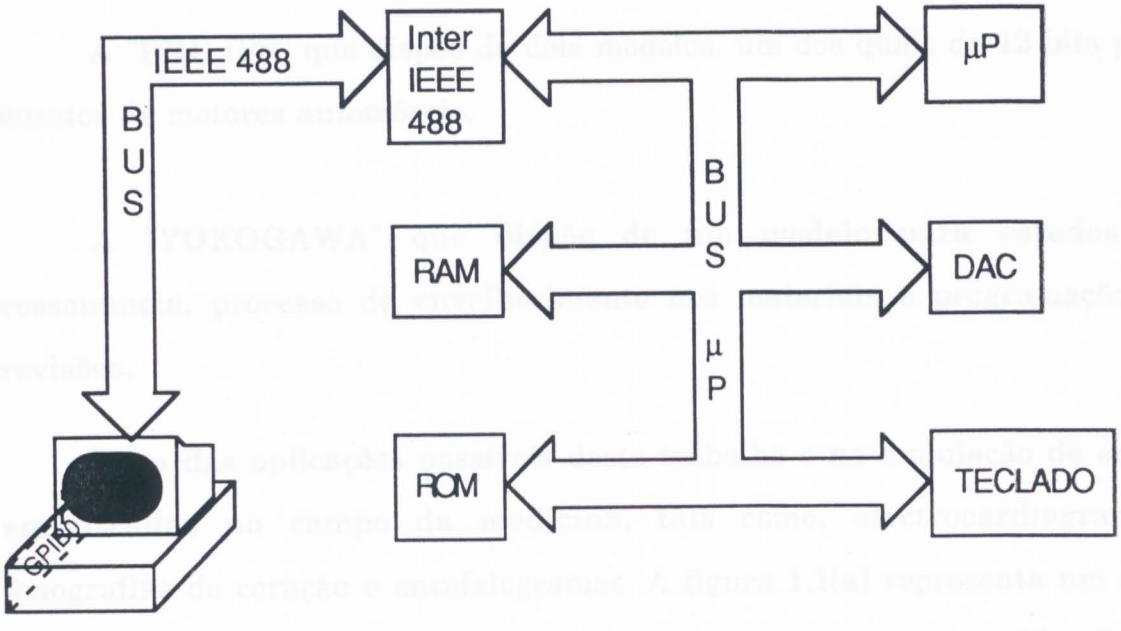


Figura P.1: Diagrama de blocos do gerador

1- APLICAÇÕES DOS GERADORES DE ONDA ARBITRÁRIA

Esta aplicação começa a despertar interesse em alguns fabricantes, destacando--se os seguintes:

A "WAVETEK" que dispõe de dois modelos de 12 bits com aplicação em medicina, física e informática.

Na medicina em testes de "pacemakers", na simulação do batimento cardíaco e gráficos cerebrais.

Na física a nível de propagação sonar, radar e ultrasons, ensaios de materiais, simulação da comutação de motores, solenoides e relés.

Na informática em periféricos, tais como, "drives de discketes" e impressoras de alta velocidade.

A "TECTRONIX" que desenvolveu um carta com uma resolução de 10 bits, para estudo da fadiga nos materiais.

A "PHILIPS" que dispõe de dois modelos, um dos quais de 12 bits para ensaios de motores automóveis.

A "YOKOGAWA" que dispõe de um modelo para estudos de ressonância, processo de envelhecimento nos materiais e programação de revisões.

Uma das aplicações possíveis deste trabalho é na simulação de ondas encontradas no campo da medicina, tais como, electrocardiogramas, fonografias do coração e encefalogramas. A figura 1.1(a) representa um ciclo do batimento cardíaco sendo esta onda um dos sinais reconstruidos, figura 1.1(b).

Nesta figura a curva superior representa o sinal reconstruido sem filtragem, à saída do DAC e a onda inferior representa o mesmo sinal depois de filtrado.

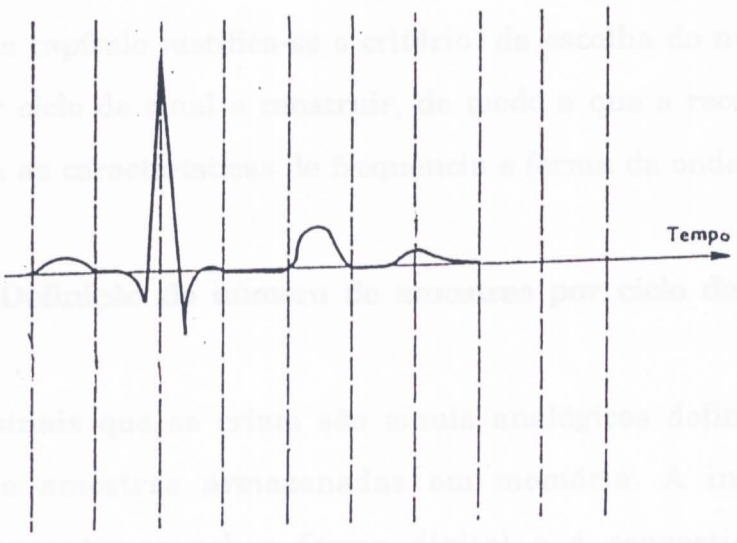


Figura 1(a): Ciclo de batimento cardíaco

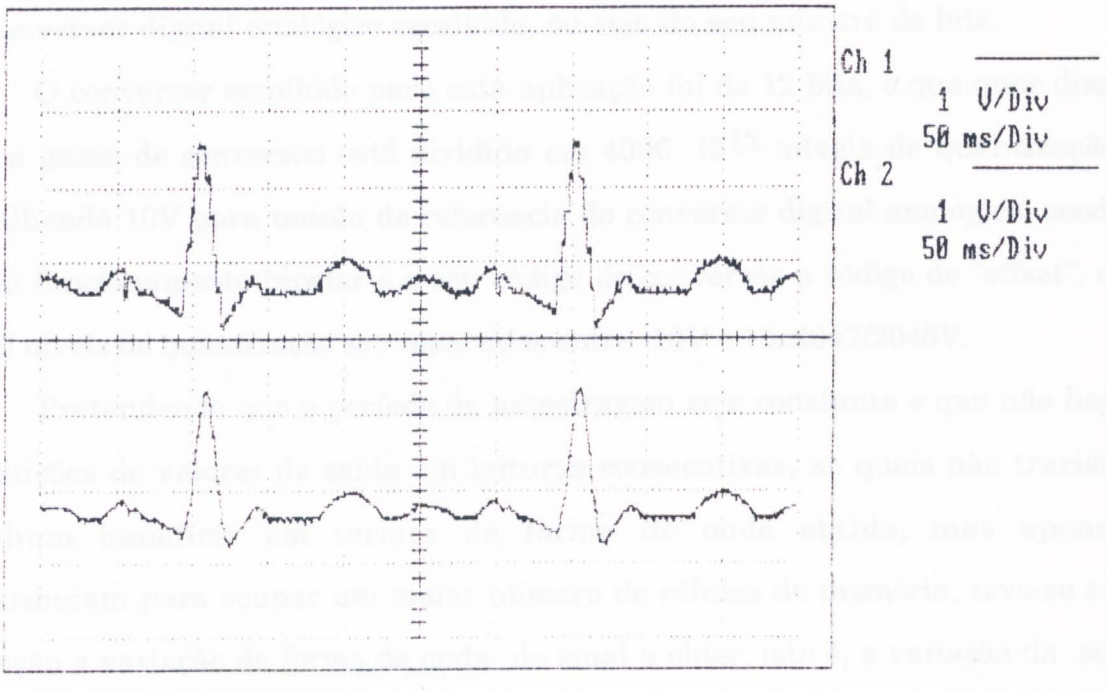


Figura 1(b): Reconstrução do ciclo do batimento cardíaco

2 - FREQUÊNCIA DE AMOSTRAGEM

2.1 - Introdução

Neste capítulo justifica-se o critério da escolha do número de amostras usadas por ciclo de sinal a construir, de modo a que a reconstrução esteja de acordo com as características de frequência e forma da onda.

2.2 - Definição do número de amostras por ciclo de sinal

Os sinais que se criam são sinais analógicos definidos por um certo número de amostras armazenadas em memória. A informação de cada amostra encontra-se sob a forma digital e é convertida em informação analógica, usando uma frequência de conversão de acordo com a frequência a gerar.

O número de amostras distintas possíveis está dependente da resolução do conversor digital analógico escolhido, ou seja do seu número de bits.

O conversor escolhido para esta aplicação foi de 12 bits, o que quer dizer que a gama de conversão está dividida em 4096 (2^{12})níveis de quantização. Escolhendo 10V para tensão de referência do conversor digital analógico, sendo o seu funcionamento bipolar e o seu código de conversão o código de "offset", os 4096 níveis de quantização são repartidos entre -10V e 10x2047/2048V.

Pretendendo que o período de amostragem seja constante e que não haja repetições de valores de saída em leituras consecutivas, as quais não trariam nenhum benefício em termos de forma de onda obtida, mas apenas contribuiam para ocupar um maior número de células de memória, teve-se em atenção a variação da forma de onda do sinal a obter, isto é, a variação da sua derivada.

Como um dos sinais a ser gerado onde a variação da derivada é mais significativa é a função sinusoidal, representou-se esta função na figura 2.1, na qual se estabeleceram as condições anteriores.

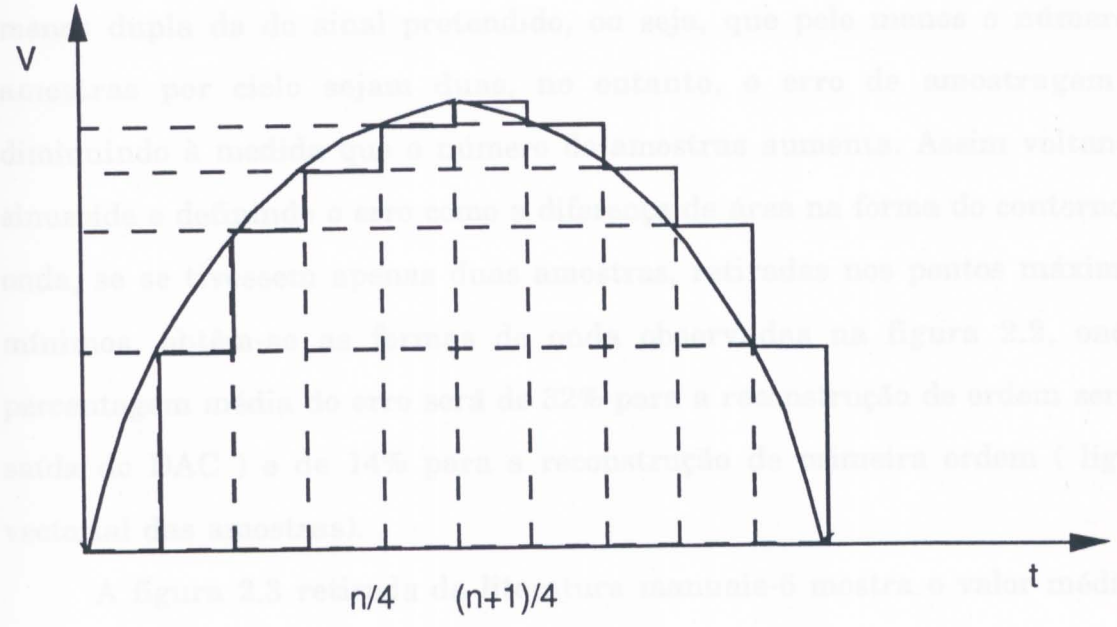


Figura 2.1: Cálculo do número de amostras n

Sendo a zona de menor variação da derivada do seno na vizinhança do seu máximo, é onde pode existir a primeira repetição de valores em duas amostras consecutivas, pelo que:

$$\text{sen}(2\pi/n \times n/4) - \text{sen}(2\pi/n \times (n+1)/4) >= 2/2^{12} \quad <=>$$

$$1 - \cos(2\pi/n) >= 2^{-11} \quad <=>$$

$$n <= 200$$

de onde se conclui que 200 é o número máximo de amostras possível de usar sem que haja repetições.

Por outro lado interessa saber o número mínimo de amostras para construir o sinal:

O teorema da amostragem de Nyquist afirma que na (re)construção de um sinal de frequência f não haverá o erro de aparecimento de um sinal falso (fenómeno de " aliasing") desde que a frequência de amostragem f_a seja pelo menos dupla da do sinal pretendido, ou seja, que pelo menos o número de amostras por ciclo sejam duas, no entanto, o erro de amostragem vai diminuindo à medida que o número de amostras aumenta. Assim voltando à sinusoide e definindo o erro como a diferença de área na forma do contorno da onda, se se tivessem apenas duas amostras, retiradas nos pontos máximos e mínimos, obtêm-se as formas de onda observadas na figura 2.2, onde a percentagem média do erro será de 32% para a reconstrução de ordem zero (à saída do DAC) e de 14% para a reconstrução de primeira ordem (ligação vectorial das amostras).

A figura 2.3 retirada da literatura manuais-6 mostra o valor médio da precisão do sinal reconstruido à medida que o número de amostras aumenta.

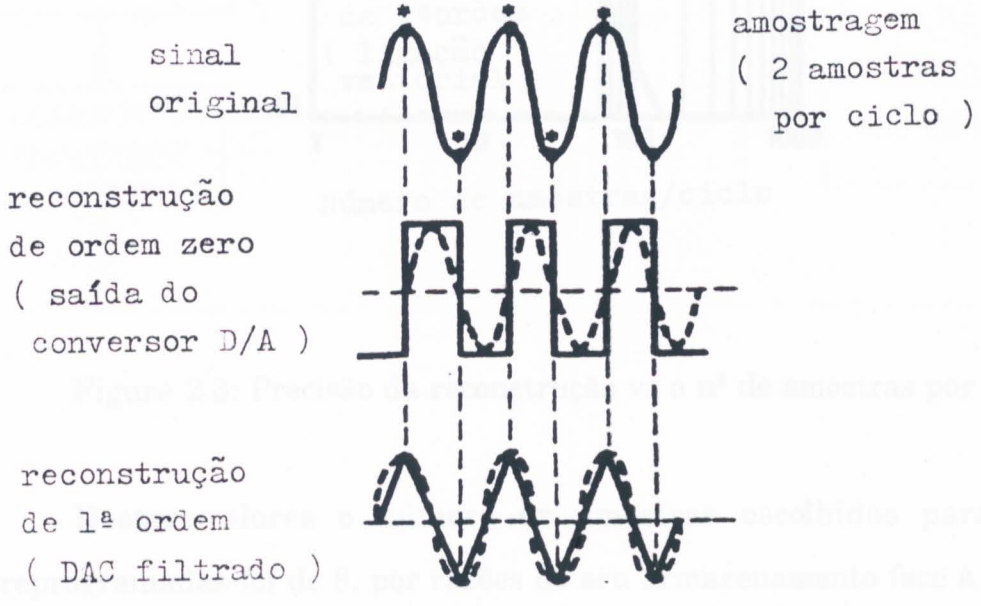


Figura 2.2: Construção de uma sinusoide com $f_a = f_{sinal}$

Da figura 2.3 verifica-se que num conversor de ordem zero, à saída do DAC, são necessárias mais de 20 amostras para um precisão média de 90% ou mais. Por este facto é colocado um filtro de 2ª ordem, capaz de com um número de amostras entre sete e dez, número este que é o corrente como regra prática na industria em aparelhagem de síntese digial, caso dos osciloscópios de memória digital da Philips, por ciclo a (re)construir, satisfazer os objectivos da aplicação.

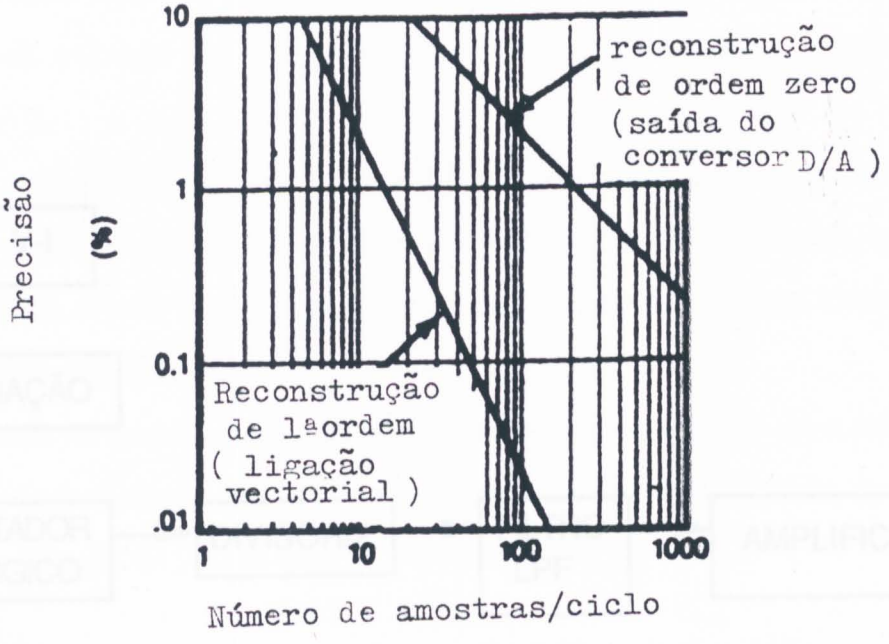


Figura 2.3: Precisão da reconstrução vs o nº de amostras por ciclo

Destes valores o número de amostras escolhidas para as ondas preprogramadas foi de 8, por razões do seu armazenamento face à organização da memória, isto é, a frequência de amostragem para as ondas preprogramados é 8 vezes a frequência do sinal a construir.

3 - INTERFACE DIGITAL - ANALÓGICA

3.1 - Introdução

Neste capítulo faz-se uma descrição do tratamento de informação realizado por esta interface que consiste na conversão de informação binária digital para informação analógica e acondicionamento do sinal, o qual é contituído pela filtragem e variação da amplitude. Na figura 3.1 pode-se observar o diagrama de blocos correspondente a este interface.

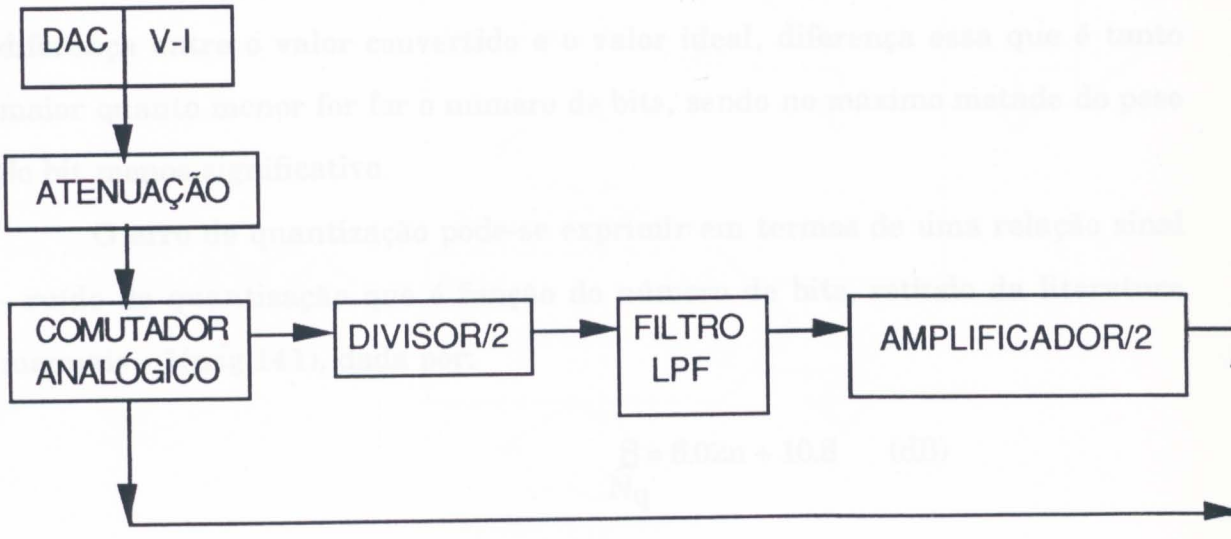


Figura 3.1: Diagrama de blocos da interface digital analógica

3.2 - Conversão digital analógica

Os conversores digitais analógicos têm uma vasta gama de aplicações: Tais como em detectores de pico, fontes de alimentação programáveis, "drivers" de motores passo a passo, compressão e expansão de linguagem. No presente trabalho estes são utilizados para síntese digital de onda.

O princípio que presidiu à síntese digital dos sinais foi um processo que é designado por síntese dos sinais em degrau (SWS, " Staircase Waveform Synthesis"), nome que deriva da saída não ser de variação continua, mas sim por salto, sendo o seu nível definido pelo código binário colocado na entrada do conversor digital analógico (DAC, " Digital Analog Converter ").

Ao número de amostras está ainda associado o erro de quantização, diferença entre o valor convertido e o valor ideal, diferença essa que é tanto maior quanto menor for for o número de bits, sendo no máximo metade do peso do bit menos significativo.

O erro de quantização pode-se exprimir em termos de uma relação sinal - ruído de quantização que é função do número de bits, retirado da literatura manuais - 1(pág.141), dada por:

$$\frac{S}{N_q} = 6.02n + 10.8 \quad (dB)$$

Onde:

n é o número de bits,

S o sinal,

N_q o ruído de quantização.

O produto 6,02n designa-se por gama dinâmica (" Dynamic Range ", $20\log 2^n$).

3.2.1 - Métodos interpolativos

Poder-se-ia utilizar um outro método de síntese digital alternativo, designado por método interpolativo básico, o qual usa dois DACs, sendo um para estabelecer um ponto inicial e o outro o ponto final de um segmento correspondente ao sector de onda a criar. A figura 3.2 mostra um esquema do circuito deste método.

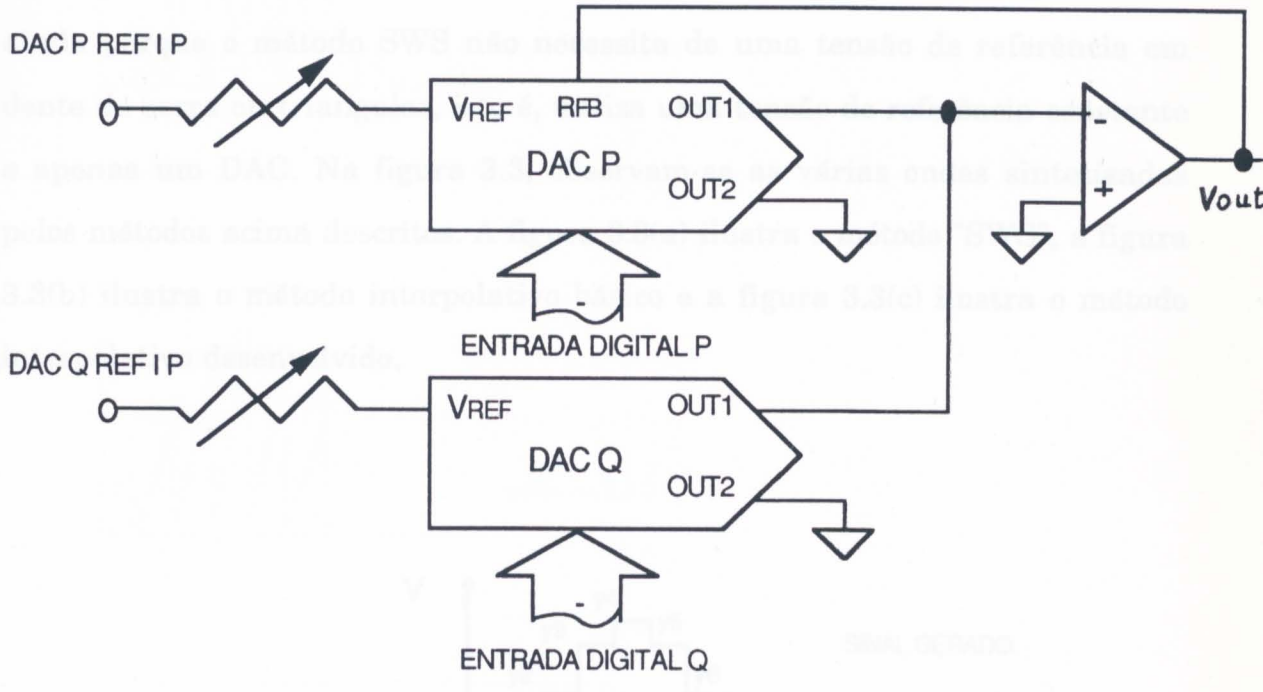


Figura 3.2 - Esquema do método interpolativo básico

O método interpolativo relativamente ao método SWS tem as seguintes vantagens: uma melhor precisão, menor ruído de quantização, melhor tempo de resposta e redução de "glitches", tendo, no entanto, as desvantagens de ser necessária uma tensão de referência em dente de serra e a necessidade de se utilizarem dois DACs. Existe ainda uma forma desenvolvida deste método que é conhecido por método interpolativo desenvolvido, destinguindo-se um do outro pela forma de onda da tensão de referência, que neste último é triangular, o que permite uma alternância dos DACs na definição dos pontos inicial e final do sector de onda a construir. A razão da não utilização destes métodos foi devida, às boas características do DAC utilizado, nomeadamente por ter uma resolução de 12 bits e possuir "double buffering" reduzindo os "glitches" e ainda porque o método SWS não necessita de uma tensão de referência em dente de serra ou triangular, isto é, utiliza uma tensão de referência constante e apenas um DAC. Na figura 3.3, observam-se as várias ondas sintetizadas pelos métodos acima descritos. A figura 3.3(a) ilustra o método "SWS", a figura 3.3(b) ilustra o método interpolativo básico e a figura 3.3(c) ilustra o método interpolativo desenvolvido.

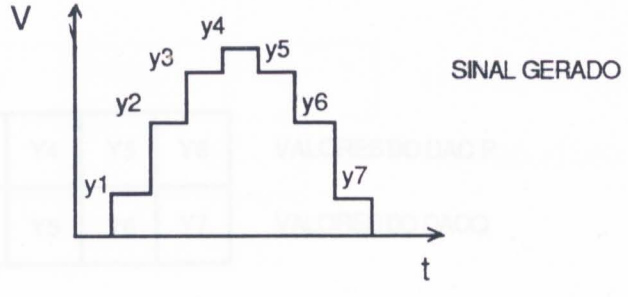
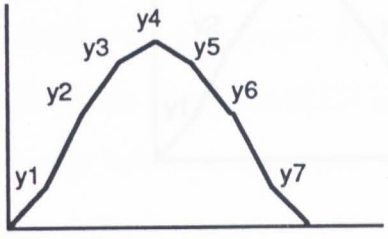
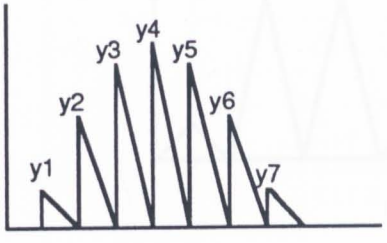
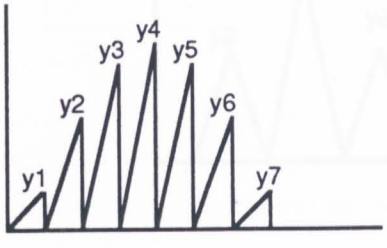


Figura 3.3(a): Onda obtida pelo método SWS







0	Y1	Y2	Y3	Y4	Y5	Y6	VALORES DO DAC P
Y1	Y2	Y3	Y4	Y5	Y6	Y7	VALORES DO DACQ

Figura 3.3(b): Método interpolativo básico

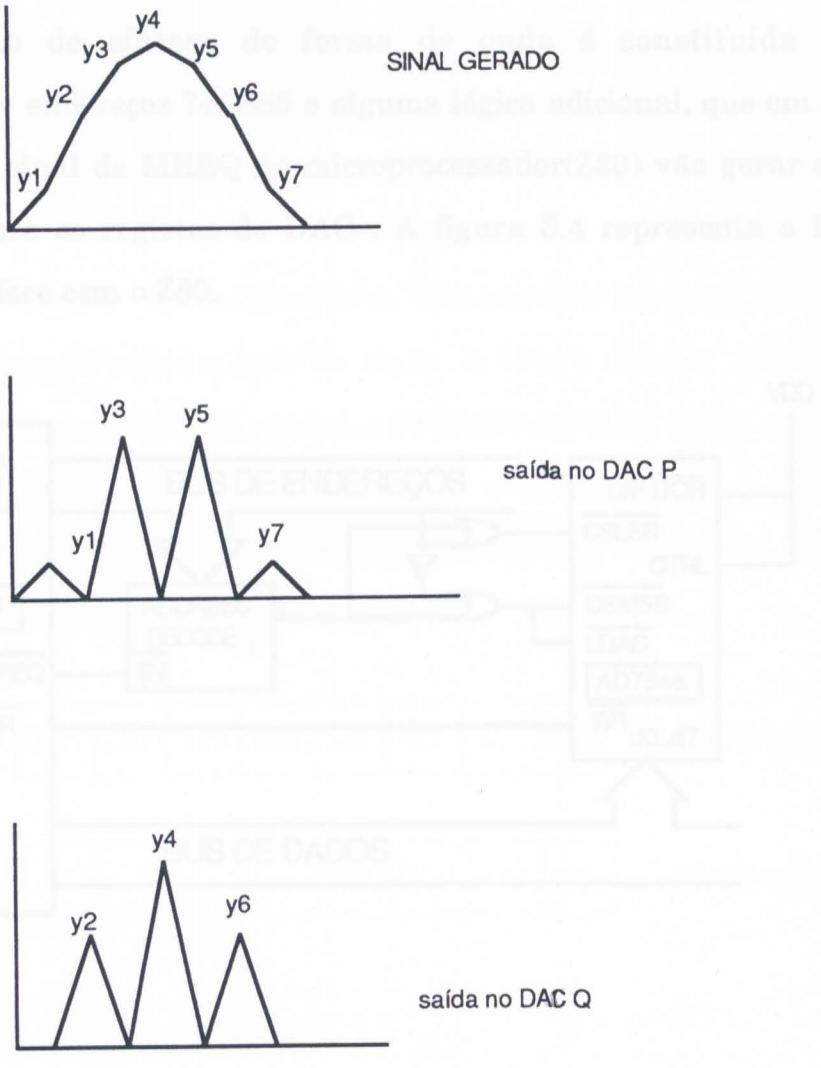


Figura 3.3(c): Método interpolativo desenvolvido

3.2.2 - Elementos do circuito

O circuito de síntese de forma de onda é constituida por um descodificador de endereços 74LS85 e alguma lógica adicional, que em conjunto com a linha de sinal de MREQ do microprocessador(Z80) vão gerar os sinais que actuam sobre os registos do DAC . A figura 3.4 representa o DAC e a respectiva interface com o Z80.

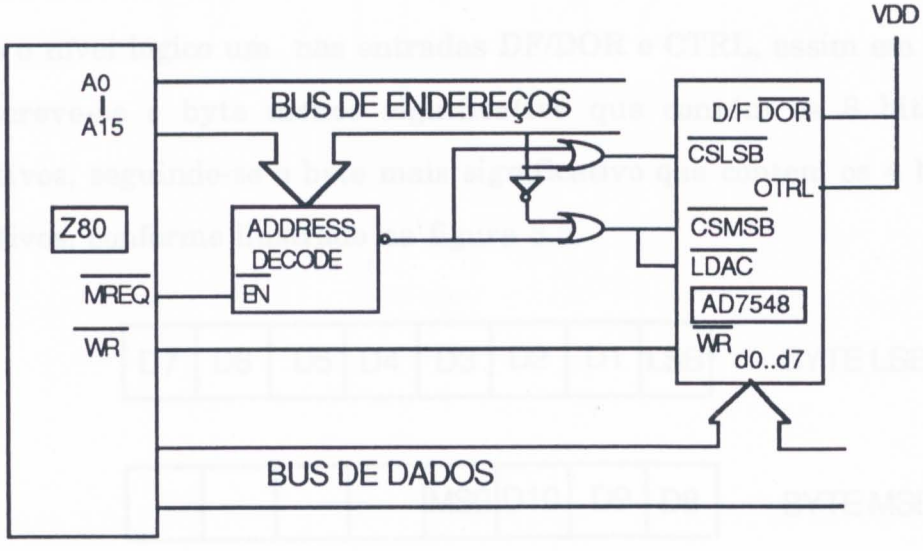


Figura 3.4: Diagrama de blocos do sintetizador de forma de onda

Escolheu-se para esta aplicação o conversor D/A da Analog Devices, ref AD7548 com as seguintes características:

resolução de 12 bits,

compatibilidade directa com bus de 8 bits

"setling time" de 1µs

uma baixa deriva de ganho (menor que 5 ppm/^{o}C)

monotonicidade ao longo dos doze bits garantida para toda a gama de temperatura

multiplicador de quatro quadrantes

envólucro pequeno de 20 pinos

alimentação compreendida entre 5 e 15 volt

3.2.3 - Modo de funcionamento

Como os a dados contêm doze bits, é necessário escrever no DAC dois Bytes consecutivos. O alinhamento de dados é feito à direita, seleccionando para isso o nível lógico um nas entradas DF/DOR e CTRL, assim em primeiro lugar escreve-se o byte menos significativo que contém os 8 bits menos significativos, seguindo-se o byte mais significativo que contem os 4 bits mais significativos, conforme ilustrado na figura 3.5.

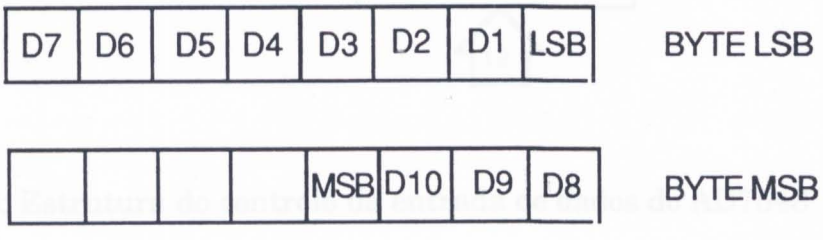


Figura 3.5: Formato dos dados

O 1º byte é armazenado num registo de entrada do DAC aplicando um nível lógico zero nas entradas CSLSB e WR, não tendo efeito imediato na saída, o segundo byte é armazenado num outro registo de entrada do DAC aplicando um nível lógico zero nas entradas CSMSB e WR, sendo a sua informação acrescentada à anteriormente armazenada. Aplicando um nível lógico zero nas entradas LDAC e WR é carregado um registo interno do DAC, com toda a informação armazenada anteriormente, sendo de seguida enviado para a saída conforme ilustrado na figura 3.6.

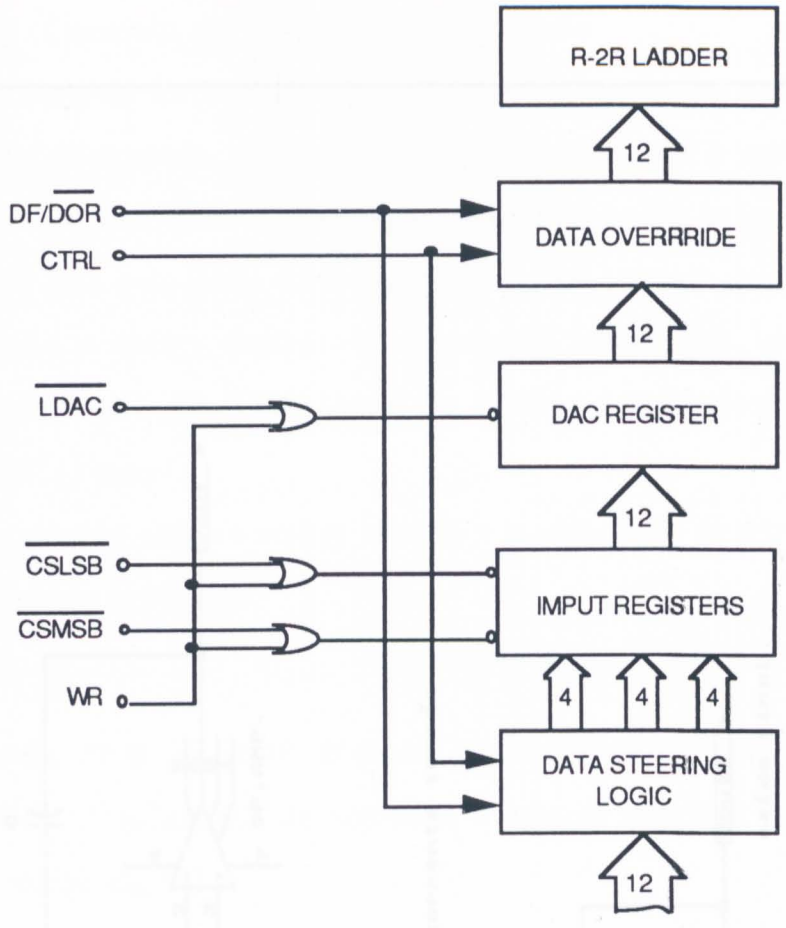


Figura 3.6: Estrutura do controlo da entrada de dados do AD7548

Denotar que neste caso específico as entradas CSMSB e LDAC são activadas simultaneamente, como pode ser observado na figura 3.4.

Visto que a saída do DAC é em corrente foi necessário executar um circuito que faça a conversão de corrente em tensão, a qual foi conseguida utilizando o circuito ilustrado na figura 3.7 (a). Visto que a tensão de saída do conversor de corrente tensão varia de 0V a Vref, para obtermos um sinal que varia de -Vref a +Vref utiliza-se o circuito ilustrado na figura 3.7 (b), que permite usar o código de entrada binário do tipo "offset", tabela A1 do apêndice A.

O esquema eléctrico dos circuitos descritos em cima encontra-se nas figuras B1 do apêndice B.

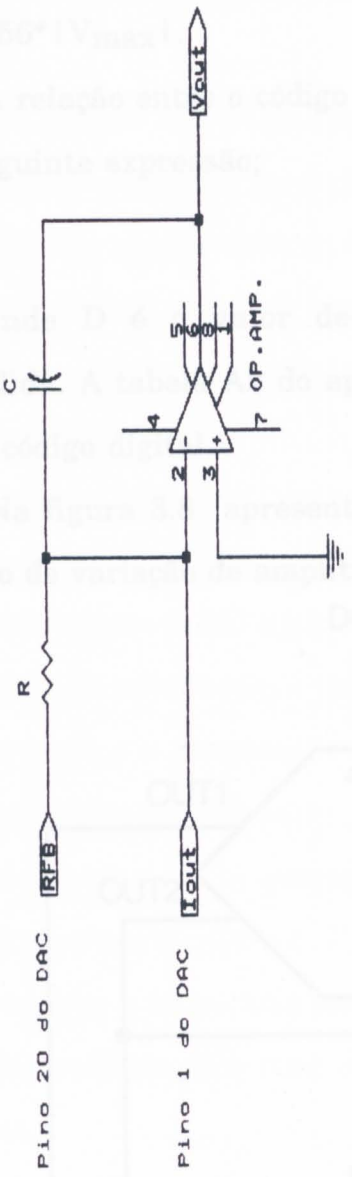


Figura 3.7(a): Conversor de corrente tensão

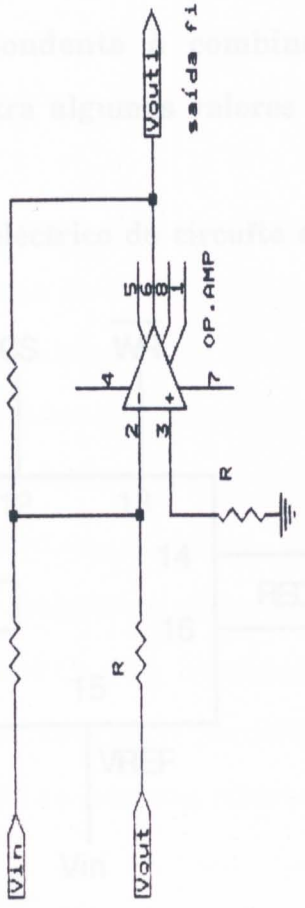


Figura 3.7(b): Conversor código de offset

3.3 - Controlo de variação de amplitude

O controlo da variação da amplitude é implementado introduzindo uma malha de atenuação programável em cascata com o módulo de conversão digital analógica. Para a realização desta função foi seleccionado o DAC PM-7524 de 8 bits pelo facto de possuir um registo interno de entrada que permite armazenar o código digital da atenuação pretendida, até se desejar uma alteraração desta. Os 8 bits permitem variar a amplitude desde $0/256 * |V_{max}|$ a $255/256 * |V_{max}|$.

A relação entre o código binário e a atenuação pretendida em dB é dada pela seguinte expressão;

$$\text{Atenuação (dB)} = -20 \log_{10}(D/256)$$

onde D é o valor decimal correspondente à combinação binária pretendida. A tabela A2 do apêndice A mostra algumas valores de atenuação versus código digital.

Na figura 3.8 apresenta o esquema eléctrico do circuíto que executa o controlo de variação de amplitude.

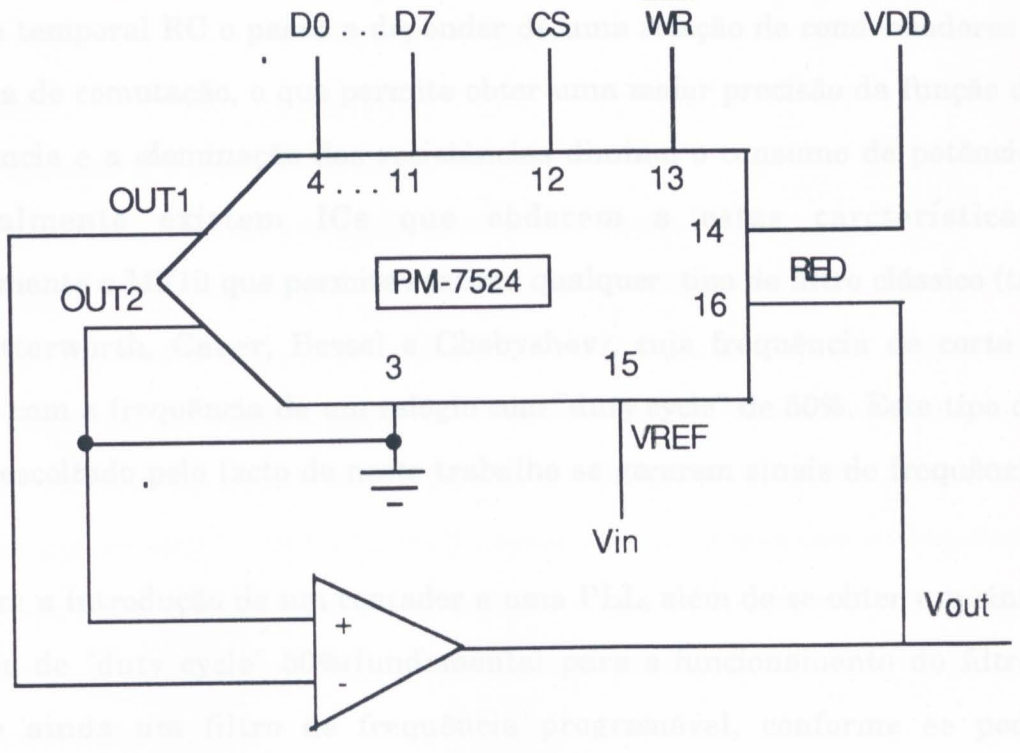


Figura 3.8: Circuito controlador de atenuação

3.4 - Filtragem

Dado que no método SWS a onda obtida é uma aproximação em escada do sinal pretendido, para que este seja recuperado é necessário proceder à filtragem da onda obtida através do SWS.

3.4.1 - Escolha do tipo de filtro a utilizar

Os filtros de capacidade comutável são considerados sistemas de amostragem analógica de dados e são construidos apenas com amplificadores operacionais, condensadores e comutadores analógicos. Se as frequências do sinal são muito menores que a frequência de comutação dos comutadores analógicos, estes constituêm uma alternativa aos filtros activos RC, com as seguintes vantagens:

O filtro pode ser fabricado monoliticamente, utilizando tecnologia MOS de alta densidade de componentes, a frequência de corte deixa de depender da constante temporal RC e passa a depender de uma relação de condensadores e frequência de comutação, o que permite obter uma maior precisão da função de transferência e a eleminação das resistências diminui o consumo de potência. Comercialmente existem ICs que obdecem a estas carcterísticas, nomeadamente o MF10 que permite realizar qualquer tipo de filtro clássico (tal como Butterworth, Cauer, Bessel e Chebyshev), cuja frequência de corte é ajustável com a frequência de um relógio com "duty cycle" de 50%. Este tipo de filtro foi escolhido pelo facto de neste trabalho se gerarem sinais de frequência variável.

Com a introdução de um contador e uma PLL, além de se obter um sinal de relógio de "duty cycle" 50%(fundamental para o funcionamento do filtro) obtem-se ainda um filtro de frequência programável, conforme se pode observar a seguir.

3.4.2 - Obtenção do relógio de comutação do filtro

Dado que o relógio a aplicar ao filtro deve ser periódico, proporcional ao número de amostras, e com "duty cycle" de 50% e visto que o único relógio disponível no módulo conversor digital analógico é duplo da frequência de amostragem, não periódico e com "duty cycle" diferente de 50%, como se pode observar na figura 3.9(a) em cima, para a obtenção do sinal periódico faz-se uma divisão por dois do já referido sinal de relógio, utilizando o contador 74LS161, sendo o sinal obtido o ilustrado na figura 3.9(a) em baixo. Pelo facto de a frequência do relógio de comutação do filtro ter que ser bastante superior à frequência do sinal e do "duty cycle ter que ser de 50% a frequência do relógio anterior é multiplicado por um factor N, por forma a obtermos a frequência de corte desejável. Este processo é realizado pela PLL 4046 juntamente com um contador BCD 40102 e representa-se na figura 3.9(b).

Resulta disto que a frequência de relógio é dada pela seguinte expressão;

$$f_{clk} = N.n.f_{sinal}$$

Onde:

n representa o número de amostras

N representa o factor multiplicativo obtido com a PLL

f_{sinal} representa a frequência do sinal a reconstruir

Desta expressão conclui-se que por um lado a frequência de corte varia com a frequência do sinal a gerar e por outro lado é possível programar a frequência de corte de acordo com o tipo de sinal a gerar programando o valor de N, pois por exemplo, para uma onda sinusoidal interessa que a frequência de corte seja igual à frequência do sinal e para uma onda triangular esta deve ser ligeiramente superior a fim de permitir a passagem dos harmónicos de maior frequência.

Na figura B.2 do apêndice B representa-se o esquema eléctrico do filtro.

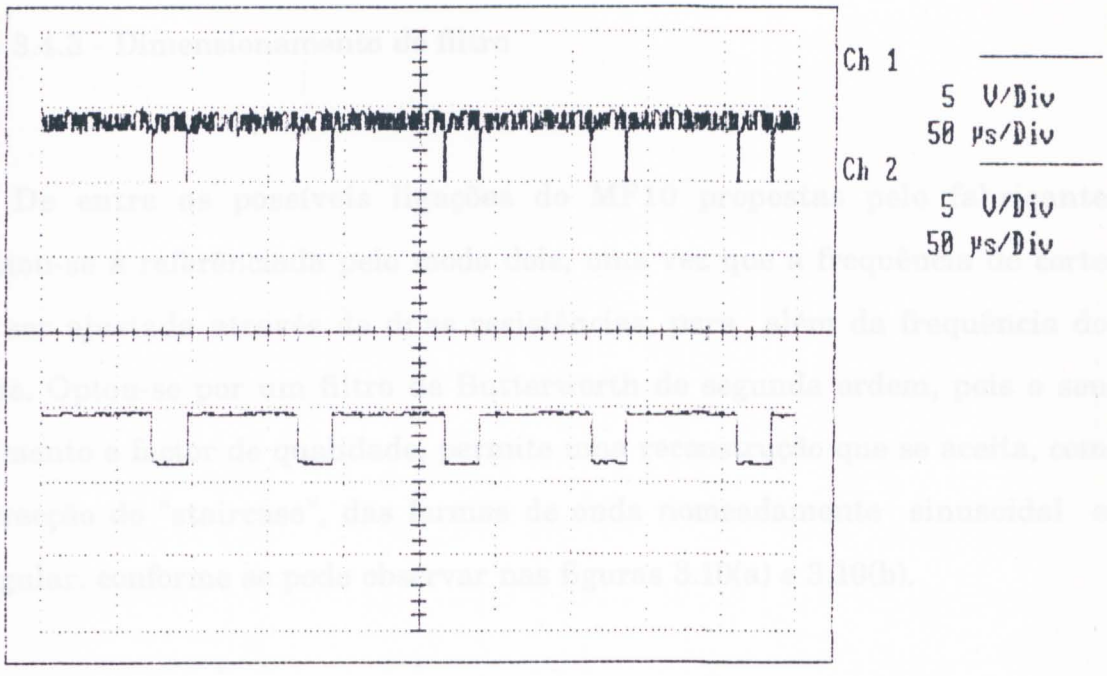


Figura 3.9(a): Relógio disponível, em cima no conversor D/A e em baixo o relógio obtido depois da divisão por 2.

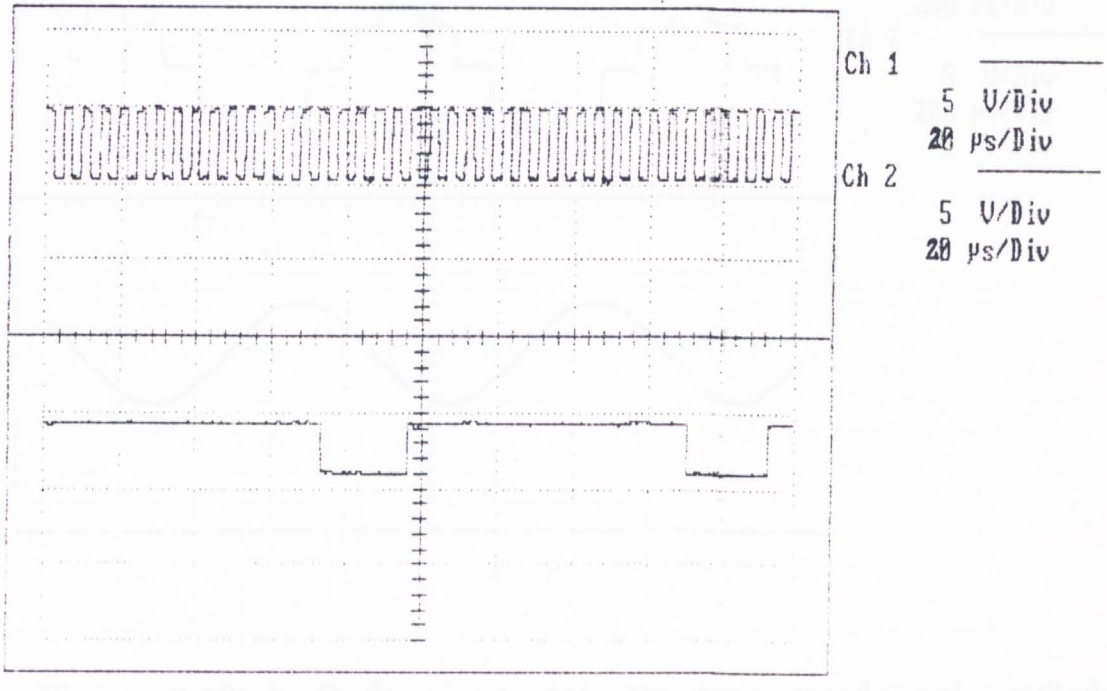


Figura 3.9(b): Relógio com "duty cicle de 50%, em baixo o relógio de que se parte e em cima o relógio corrigido

3.4.3 - Dimensionamento do filtro

De entre as possíveis ligações do MF10 propostas pelo fabricante executou-se a referênciada pelo modo dois, uma vez que a frequência de corte pode ser ajustada através de duas resistências, para além da frequência do relógio. Optou-se por um filtro de Butterworth de segunda ordem, pois o seu decaímento e factor de qualidade, permite uma reconstrução que se aceita, com a correcção do "staircase", das formas de onda nomeadamente sinusoidal e triângular. conforme se pode observar nas figuras 3.10(a) e 3.10(b).

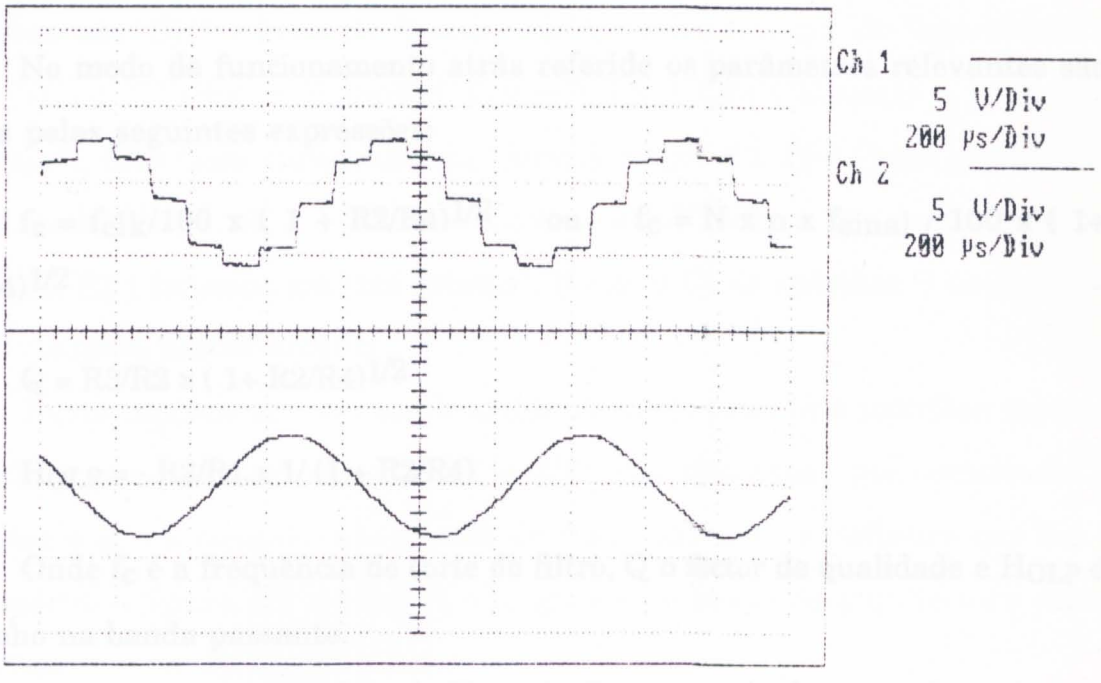


Figura 3.10(a): Onda sinusoidal, em cima gerada pelo método "staircase" e em baixo depois de filtrada.

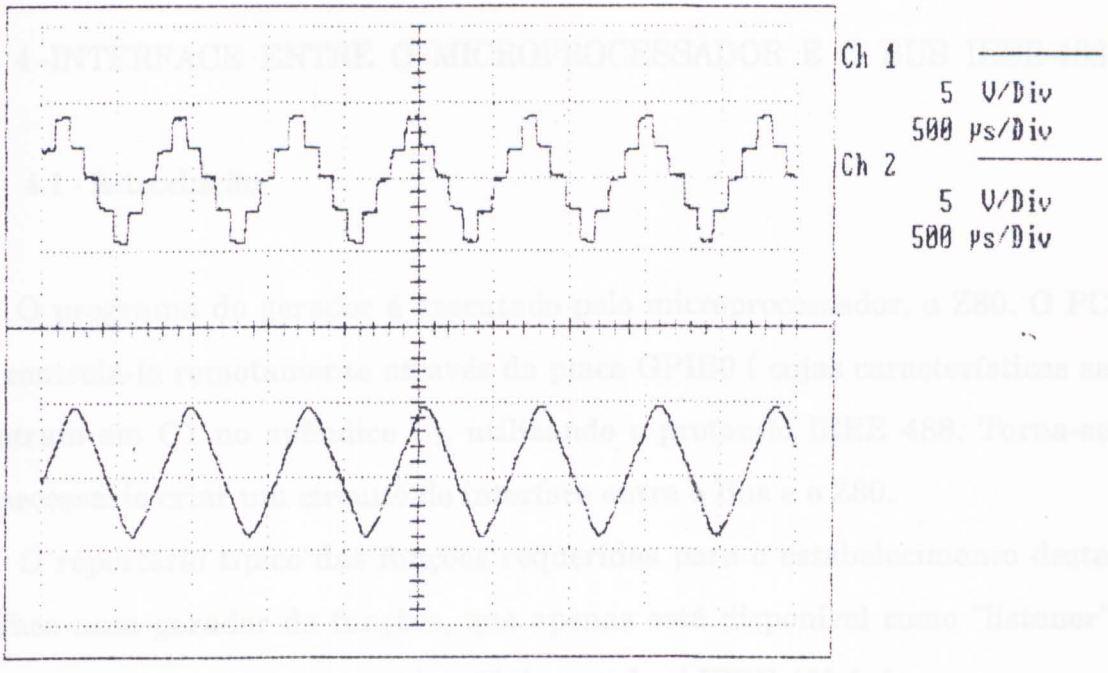


Figura 3.10(b): Onda triangular, em cima gerada pelo método "staircase" e em baixo depois de filtrada.

No modo de funcionamento atrás referido os parâmetros relevantes são dados pelas seguintes expressões:

$f_c = f_{clk}/100 \times (1 + R2/R4)^{1/2}$ ou $f_c = N \times n \times f_{sinal} / 100 \times (1+ R2/R4)^{1/2}$

$Q = R3/R2 \times (1+ R2/R4)^{1/2}$

$H_{OLP} = - R2/R1 \times 1/ (1 + R2/R4)$

Onde f_c é a frequência de corte do filtro, Q o factor de qualidade e H_{OLP} é o ganho na banda passante.

Visto que se pretende um filtro de Butterworth de segunda ordem, o factor de qualidade é de 0.707, e com n=8, N = 20, e ainda uma frequência de corte igual a 1,6 vezes a frequência do sinal, valor este que foi deduzido experimentalmente. Para estas características obtêm-se os seguintes valores de resistências, R4 = 10Kohm, R2 = 5,6Kohm, R3 = 3,3Kohm e R1 = 2,9Kohm.

4 -INTERFACE ENTRE O MICROPROCESSADOR E O BUS IEEE-488

4.1 - Introdução

O programa do gerador é executado pelo microprocessador, o Z80. O PC pode controlá-lo remotamente através da placa GPIB0 (cujas características se encontram em C1 no apêndice C), utilizando o protocolo IEEE 488. Torna-se pois necessário criar um circuito de interface entre o Bus e o Z80.

O reportório típico das funções requeridas para o estabelecimento desta interface num gerador de funções, que apenas está disponível como "listener" são, conforme se apresenta a página 68 do standard IEEE 488.1 de:

AH ("Accept Handshake"), L ("Listener"), RL ("Remote Local") e DT ("Device Trigger"). No entanto, o subconjunto das funções implementadas pela interface são: SH1 (fonte de "handshake"), AH1(aceitador de "handshake"), T5 e TE5 (" Talker e Extended Talker"), L3 e LE3 ("Listener e Extended Listener"), SR0 (sem capaciadde de "Service Request"), PP0 ("sem capacidade de "Parallel Poll"), RL1 ("Remote/ Local control"), C0 (é não controlador do sistema), E2 (funciona em três estados). No item C2 do apêndice C descreve-se o sub conjunto destas funções.

Neste capítulo faz-se uma descrição do modo como esta interface opera a transferência de informação do PC e Z80, dos elementos que constituem o Hardware que a compõe, o seu modo de funcionamento e Software que lhe é necessário. A figura 4.1 apresenta o diagrama de blocos da arquitectura desta interface.

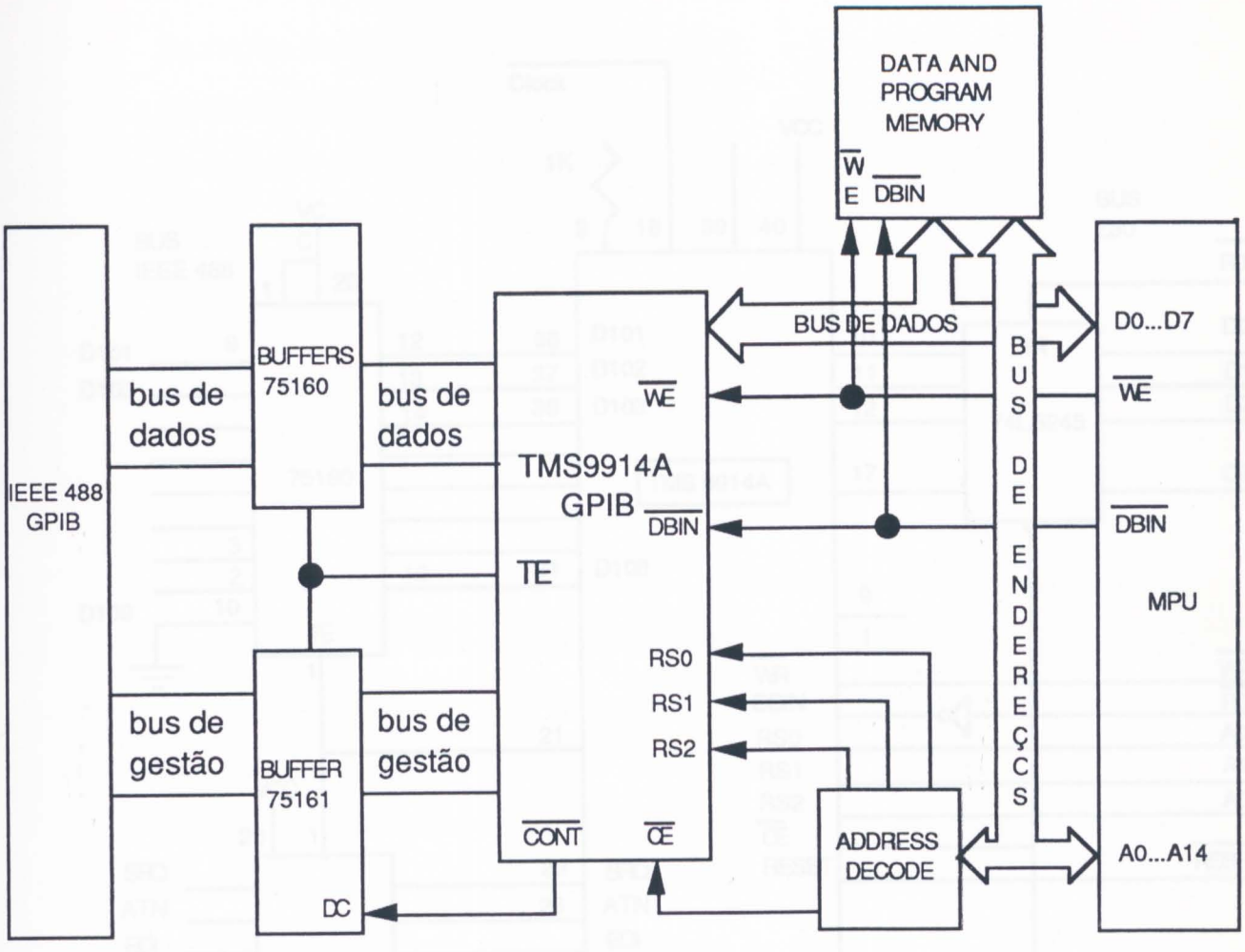


Figura 4.1: Diagrama de blocos do circuito de "interface"

4.2 - **Elementos do circuíto**

Este circuito é composto por um descodificador de endereços, o 74LS85 e alguma lógica adicional, que juntamente com o sinal da linha activa a zero IORQ do Z80 gera um sinal que actua "transceiver octal" 74LS245 e na linha activa a zero CE do periférico TMS 9914A. Este circuíto é constituido ainda por dois "buffers" 75160 e 75161. Se se pretendesse que esta interface tivesse a função de ser placa controladora o Buffer 75161 deveria ser substituido pelo 75162, segundo notas do fabricante. A figura 4.2 apresenta o desenho esquemático do circuíto implementado com estes componentes.

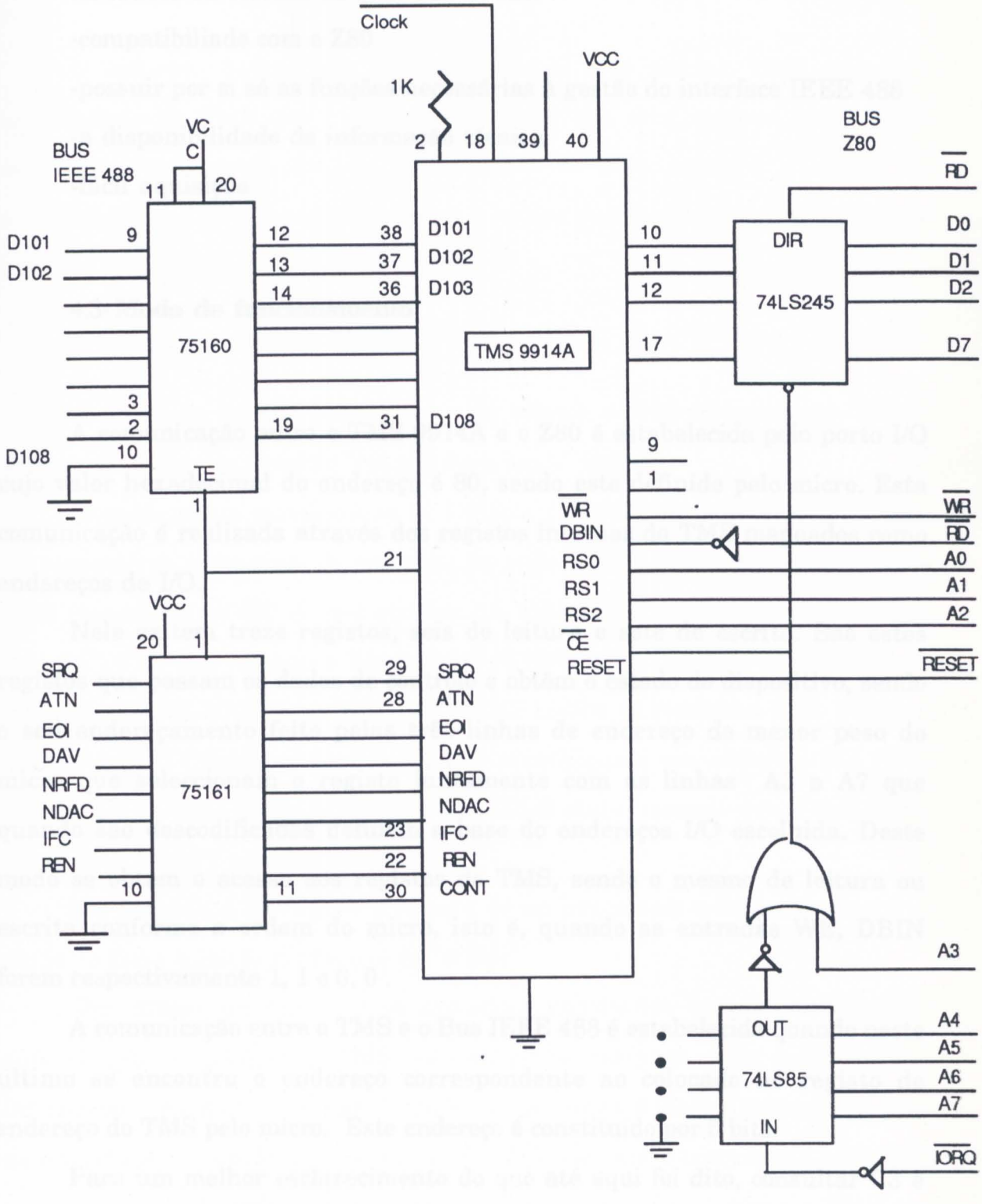


Figura 4.2: Desenho esquemático do circuito de interface

As razões da escolha do TMS 9914A são:

-compatibiliade com o Z80

-possuir por si só as funções necessárias à gestão do interface IEEE 488

-a disponibilidade de informação técnica

-fácil aquisição

4.3- Modo de funcionamento

A comunicação entre o TMS 9914A e o Z80 é estabelecida pelo porto I/O cujo valor hexadecimal do endereço é 80, sendo este definido pelo micro. Esta comunicação é realizada através dos registos internos do TMS mapeados como endereços de I/O.

Nele exitem treze registos, seis de leitura e sete de escrita. São estes registos que passam os dados de controlo e obtêm o estado do dispositivo, sendo o seu endereçamento feito pelas três linhas de endereço de menor peso do micro que seleccionam o registo juntamente com as linhas A3 a A7 que quando são descodificadas definem a base do endereços I/O escolhida. Deste modo se obtem o acesso aos registos do TMS, sendo o mesmo de leitura ou escrita conforme a ordem do micro, isto é, quando as entradas WE, DBIN forem respectivamente 1, 1 e 0, 0 .

A comunicação entre o TMS e o Bus IEEE 488 é estabelecida quando neste último se encontra o endereço correspondente ao colocado no registo de endereço do TMS pelo micro. Este endereço é constituido por 5 bits.

Para um melhor esclarecimento do que até aqui foi dito, consultar C3 e C4 do apêndice C.

4.4 - Software de controlo

O software de controlo é constituido por três rotinas. Uma rotina de inicialização do TMS que consiste em colocar nos seus registos internos as condições necessárias ao seu funcionamento. Uma outra é a rotina de interrupção do Z80 e finalmente temos a rotina que realiza a transferência de bytes de informação entre o Z80 e o GPIB.

4.4.1- Rotina de inicialização do TMS

A rotina de inicialização começa por colocar o TMS no estado de "IDLE" através do comando auxiliar "set software reset" (swrst).

Seguidamente carregam-se os registos INT.MASK0 e INT.MASK1 com as respectivas máscaras. A máscara do registo INT.MASK0 permite que o TMS comunique ao Z80 de que recebeu um byte de informação através bit BI(" byte In") e de que recebeu o último byte de informação através do bit END. A máscara do INT.MASK1 permite que o TMS receba os comandos DCL ("Device Clear") e SDC ("Selected Device Clear"), identifique o seu endereço MA ("My Adress") e permite ainda o regresso ao "IDLE state" sempre que a linha IFC ("Interface Clear") se torne activa.

De seguida é carregado o registo de endereço com o respectivo valor e ainda se faz a inibição da função de "talker".

Por último enviam-se três comandos auxiliares; lon ("listener only") cuja função é colocar o TMS apenas como receptor, isto é, só recebe informação. "Holdoff on all data" (hdfa), cuja função é fazer a retenção do byte de informação recebido, permitindo que o Z80 que faça o tratamento adequado ao byte recebido, após o que executa o "reset" do hdfa, ficando o TMS em condições de receber novo byte. "Clear software reset" (swrst) que faz com que o TMS fique activo, isto é, faz com que o GPIB o reconheça como elemento activo do sistema.

A sequência da inicialização representa-se na tabela A3 do apêndice A.

A figura 4.3 mostra o fluxograma da rotina de inicialização do TMS, em D1 do apêndice D apresenta-se o programa correspondente.

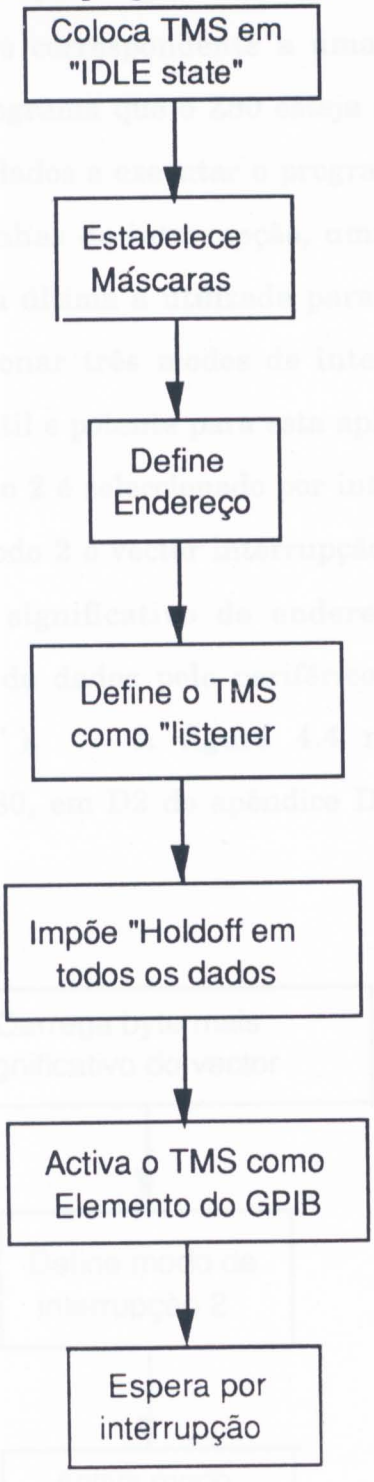


Figura 4.3: Fluxograma da rotina de inicialização do TMS

4.4.2 - Rotina de interrupção do Z80

Com a rotina de interrupção é possível, sempre que se deseje realizar a transferência de informação correspondente a uma forma de onda a gerar, causar uma paragem no programa que o Z80 esteja a executar, para passar a realizar a transferência de dados e executar o programa principal.

O Z80 possui duas linhas de interrupção, uma linha não mascarável e uma mascarável, sendo esta última a utilizada para gerar a interrupção, pois com esta é possível seleccionar três modos de interrupção, onde o modo de interrupção 2 é o mais versátil e potente para esta aplicação.

O modo de interrupção 2 é seleccionado por intermédio da instrução IM2 ("Interrupt mode 2"). No modo 2 o vector interrupção é formado pelo registo I que contém o byte mais significativo do endereço, sendo o byte menos significativo posto no bus de dados pelo periférico após o conhecimento da interrupção ("acknowledge"). A figura 4.4 mostra o fluxograma da rotina de interrupção do Z80, em D2 do apêndice D apresenta-se o programa correspondente.

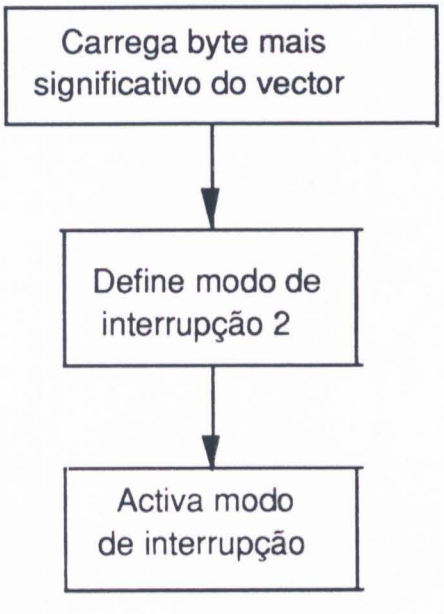


Figura 4.4: Fluxograma da rotina de interrupção do Z80

4.4.3 - Rotina de transferência de dados entre GPIB e o Z80

Ao surgir uma interrupção no Z80 significa que o GPIB tem dados para transmitir, devendo ser recebidos e armazenados em memória (RAM) pelo Z80.

Os dados a receber e armazenar são: valor da atenuação, características do filtro, frequência da onda a gerar, o número de amostras e o endereço correspodente à primeira amostra.

A figura 4.5 apresenta o fluxograma da rotina de transferência de dados entre o GPIB e Z80.

Em D3 do Apêndice D apresenta-se o programa correspondente.

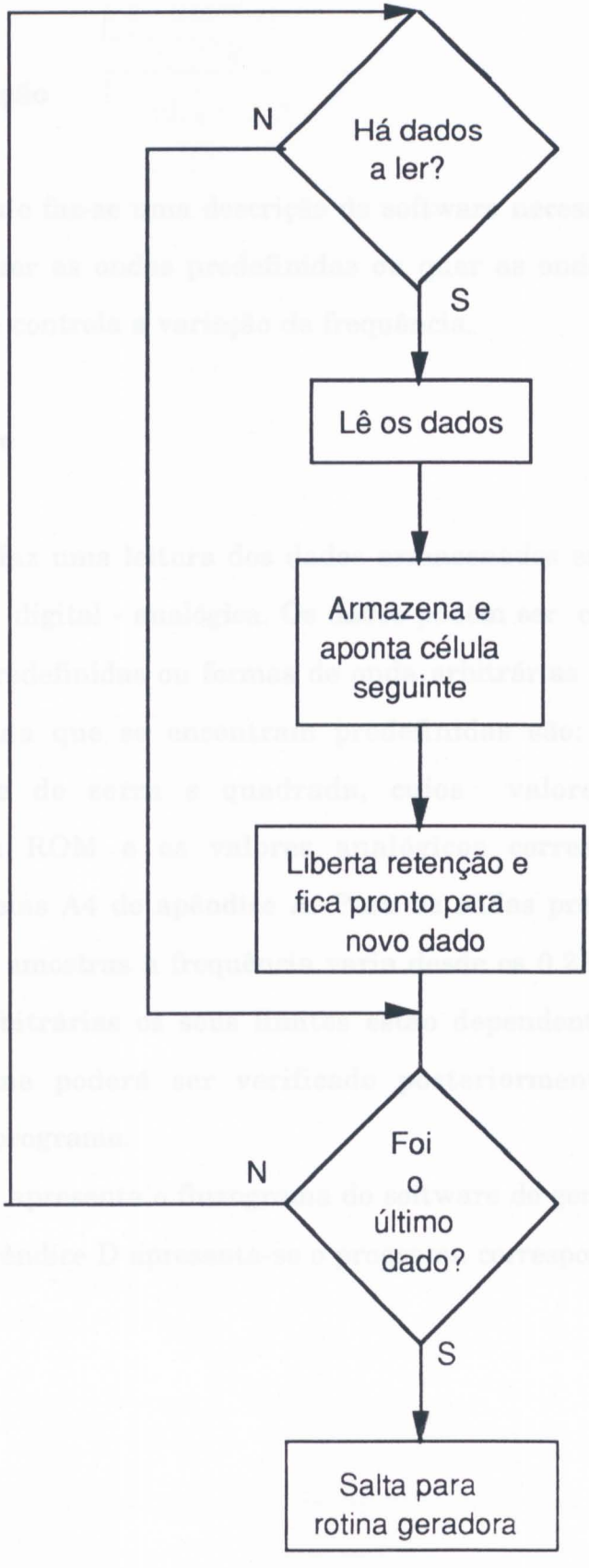


Figura 4.5: Fluxograma da rotina de transferência de dados

5 - SOFTWARE DO GERADOR DE FORMA DE ONDA

5.1 - **Introdução**

Neste capítulo faz-se uma descrição do software necessário para gerar a forma de onda, quer as ondas predefinidas ou quer as ondas arbitrárias e o modo de como este controla a variação de frequência.

5.2 - **Software**

O software faz uma leitura dos dados armazenados enviando-os para a placa de interface digital - analógica. Os dados podem ser correspondentes às formas de onda predefinidas ou formas de onda arbitrárias enviados pelo PC. As formas de onda que se encontram predefinidas são: onda sinusoidal, triangular, dente de serra e quadrada, cujos valores digitais estão armazenados em ROM e os valores analógicos correspondentes estão ilustrados na tabelas A4 do apêndice A. Para as ondas predefinidas que são definidas por oito amostras a frequência varia desde os 0.2Hz e os 1.25KHz e para as ondas arbitrárias os seus limites estão dependentes do número de amostras conforme poderá ser verificado posteriormente no estudo da temporização do programa.

A figura 5.1 apresenta o fluxograma do software do gerador de formas de onda em D4 do apêndice D apresenta-se o programa correspondente.

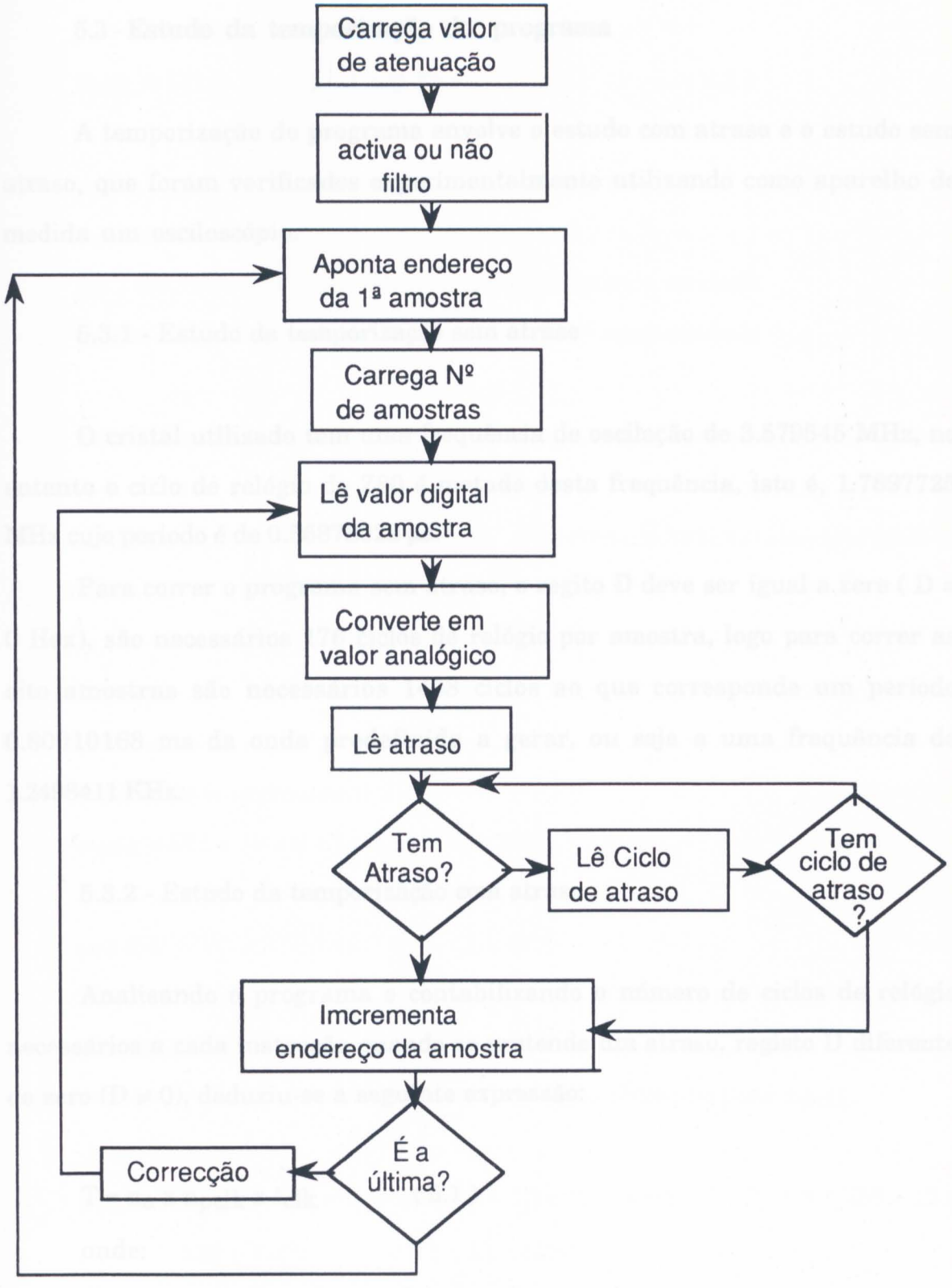


Figura 5.1:Fluxograma da rotina geradora das formas de onda

5.3 - Estudo da temporização do programa

A temporização do programa envolve o estudo com atraso e o estudo sem atraso, que foram verificados experimentalmente utilizando como aparelho de medida um osciloscópio.

5.3.1 - Estudo da temporização sem atraso

O cristal utilizado tem uma frequência de oscilação de 3.579545 MHz, no entanto o ciclo de relógio do Z80 é metade desta frequência, isto é, 1.7897725 MHz cujo periodo é de 0.55873022 μs.

Para correr o programa sem atraso, o regito D deve ser igual a zero (D = 0 Hex), são necessários 176 ciclos de relógio por amostra, logo para correr as oito amostras são necessários 1408 ciclos ao que corresponde um período 0.80010168 ms da onda predefinida a gerar, ou seja a uma frequência de 1.2498411 KHz.

5.3.2 - Estudo da temporização com atraso

Analisando o programa e contabilizando o número de ciclos de relógio necessários a cada instrução quando se pretende um atraso, registo D diferente de zero ($D \neq 0$), deduziu-se a seguinte expressão:

$$T = n_a \times n_{pclk} \times t_{clk} \qquad (5.1)$$

onde:

T é o período da onda gerada

n_a é o número de amostras do sinal

t_{clk} é o período do relógio

n_{pclk} é o número de ciclos de relógio por amostra, sendo dado por:

$$n_{pclk} = 171 + 12 \times n_{d-1} + n_d(39 + 16 \times n_{c-1}) \qquad (5.2)$$

onde:

n_d é o valor do registo D, em hexadécimal

n_{d-1} é o valor do registo D decrementedo de uma unidade

n_{c-1} é o valor do registo C decrementado de uma unidade

5.3.3 - Verificação experimental da temporização com atraso

A fórmula (5.2) foi verificada experimentalmente considerando 8 amostras .

Exemplo 1

D = 1 e C = 0

Aplicando a expressão 5.2

$n_{pclk} = 171 + 12 \times 0 + 1 \times (39 + 16 \times 255)$

$= 4290$

por 5.1

T = 19.175621

f = 52.15 Hz

contando os ciclos de relógio nesta condição obtem-se para n_{pclk}.

$n_{pclk} = 20 + 17 + 19 + 13 + 10 + 19 + 13 + 17 + 4 + 4 + 7 + 17 + 4 \times 256 + 12 \times 255 + 7 + 4 + 7 + 10 + 8 + 10 = 4290$

O que confirma a expressão 5.2 pois o valor de n_{pclk} é igual nas duas situações e como é de esperar o valor de f obtido será 52.15 Hz.

O valor observado no osciloscópio foi de aproximadamente de 52.63 Hz.

Exemplo 2

D = 1 e C = 1

Aplicando as expressões 5.1 e 5.2

$n_{pclk} = 210$

f = 1.0654 KHz

O valor observado no osciloscópio foi de 1.0869 KHz.

A pequena diferença entre o valor observado experimentalmente e o valor obtido teoricamente, resulta do facto de na observação do período no osciloscópio não poder ser muito precisa, pois cometem-se erros de leitura devido à visão humana não ser suficientemente precisa e como tal fazem-se aproximações que fazem com que os valores observados defiram dos valores calculados teóricamente.

A variação dos registos D e C afectam o valor de frequência obtido da seguinte forma: O registo D establece a gama de frequências que se pretende obter e o registo C permite percorrer as frequências que se encontram dentro da gama definida por D. A tabela A5 do apêndice A establece a relação obtida experimentalmente entre os registos D e C com a frequência da onda gerada.

6 - INTERFACE COM O UTILIZADOR

6.1 - **Introdução**

Neste capítulo faz-se uma descrição da "interface entre a aplicação e o GPIB e interface entre a aplicação e o utilizador.

6.2 - **Interface entre a aplicação e o GPIB**

Para esta aplicação a linguagem de alto nível utilizada para realizar a interface com o utilizador foi o Turbo Pascal, uma vez que dentro das que se podiam utilizar propostas pelo fabricante, esta era uma das disponíveis.

A figura 6.1 mostra a estrutura do software possível de ser usado, tomando como base o software operador da carta de interface (GPIB0), introduzida no PC, o NI-488. Este software está instalado como parte do sistema operativo e é acedido pelo programa do utilizador através de comandos standard I/O da "Universal Language Interface" (ULI).

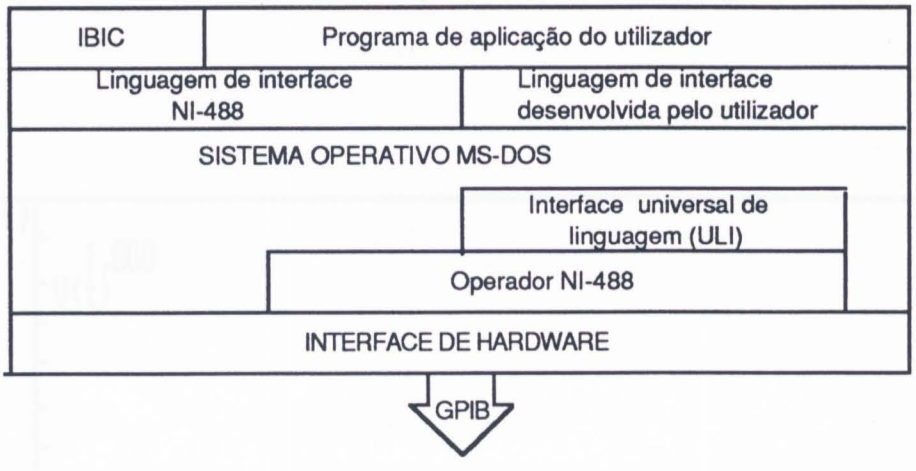


Figura 6.1: Estrutura do software

O ULI é um utilitário terminal e residente que faz a interface entre o MS-DOS e o GPIB.COM do operador NI-488. O acesso do turbo pascal a esta linguagem é feita pela utilização dos ficheiros DOS, chamados através dos comandos I/O standard do turbo pascal e desta forma elimina-se a necessidade de uma linguagem de interface. O acesso através dos Comandos I/O standard é possível, devido ao facto do Turbo Pascal considerar que todas as entradas e saídas ser transferências de ficheiros ou cadeias de caracteres provenientes da "drive", do teclado ou dirigindo-se para as impressoras ou visores.

6.3 - **Interface entre o utilizador e a aplicação**

A interface entre o utilizador e a aplicação é conseguida pela visualização no ecran das opções possíveis, que são: A função de leitura de ficheiros, a função de guardar ficheiros, a função de estabelecer as tabelas dos valores para funções arbitrárias, a função de Zoom, a função setup, e a função de gerador. Estas funções são seleccionadas premindo as teclas F1, F2, F3, F4, F5 e F6 respectivamente, conforme se pode visualizar na figura 6.2.

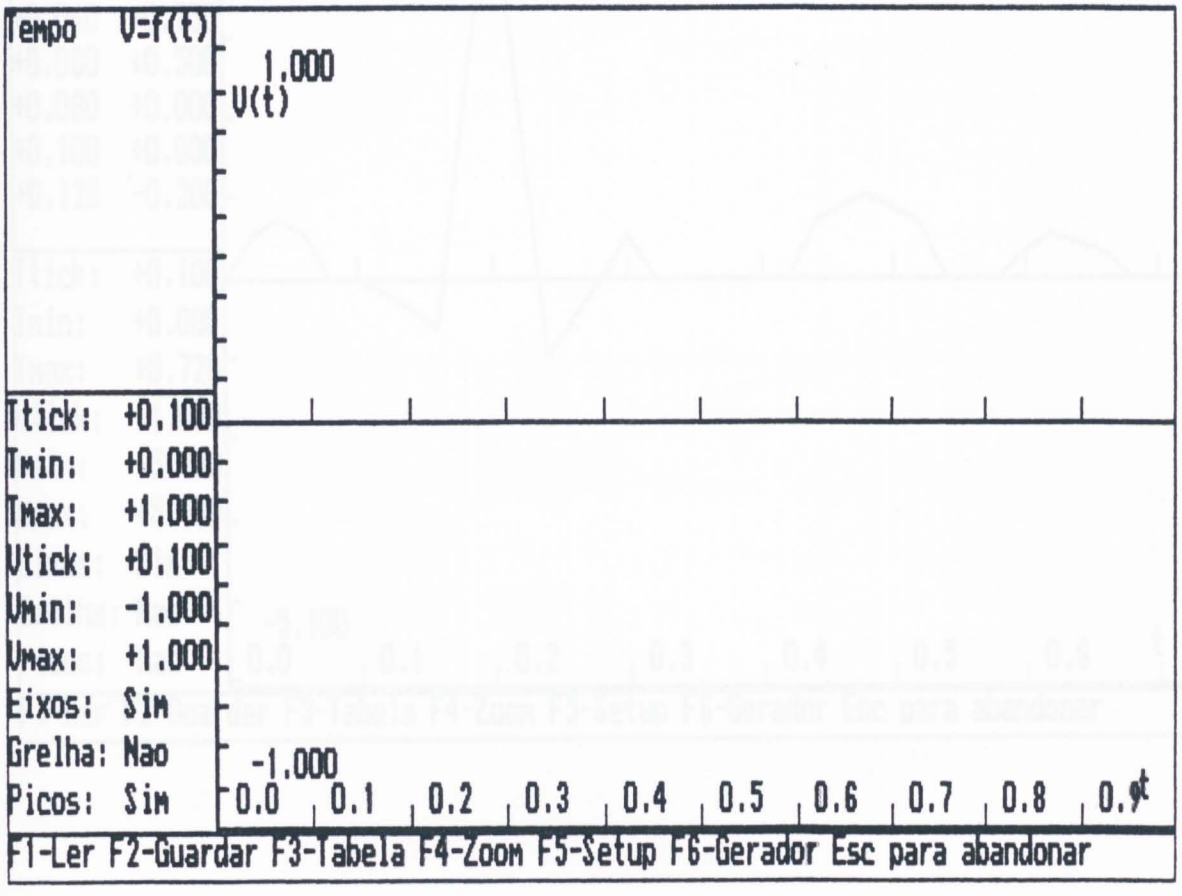


Figura 6.2: Menu do utilizador

Premindo a tecla F1 temos acesso à função de leitura, a qual permite colocar no visor qualquer forma de onda previamente armazenada em ficheiro. A figura 6.3 mostra a forma de onda armazenada no ficheiro "coração" acedido por F1.

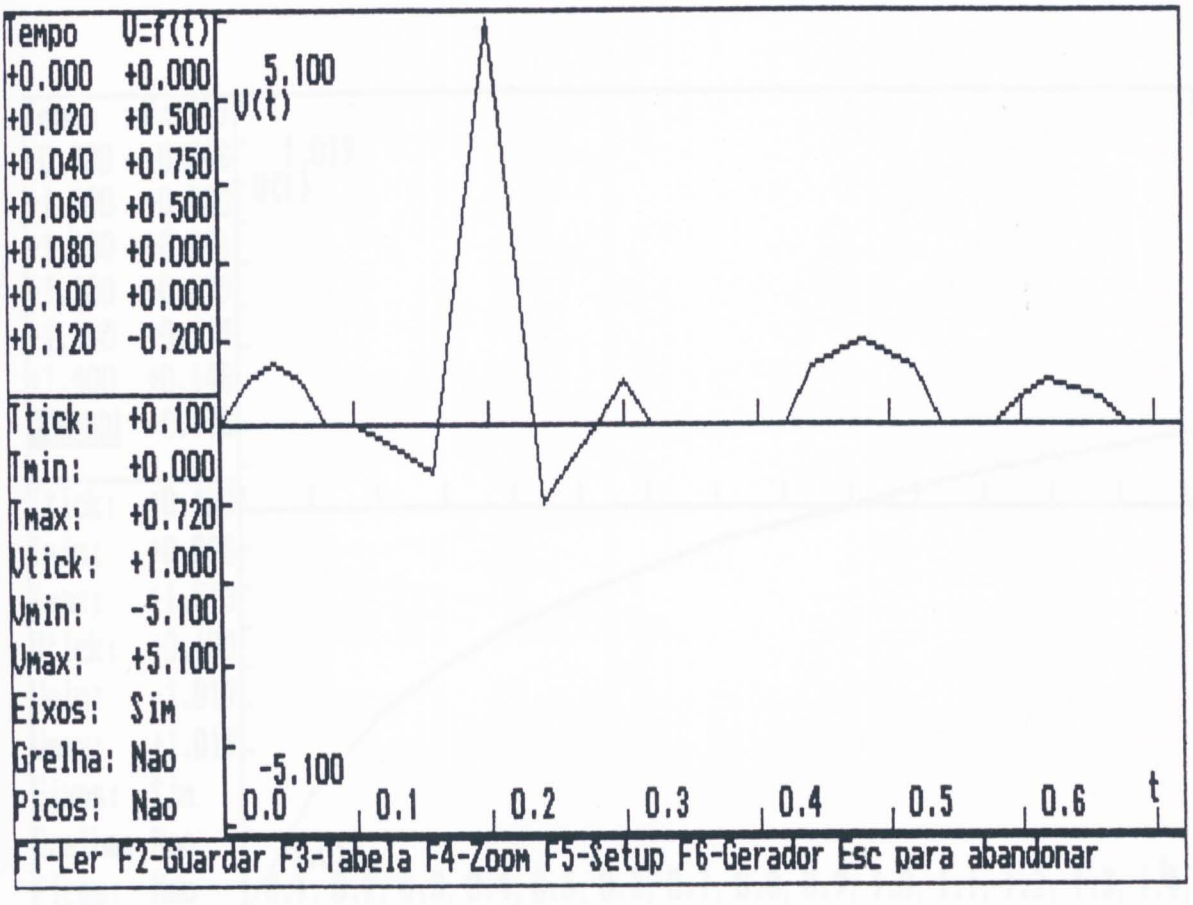


Figura 6.3: Função de leitura

Premindo a tecla F2 pode-se armazenar em ficheiro qualquer forma de onda criada. O ficheiro "coração" foi guardado através de F2 depois de gerado.

Premindo a tecla F3 é possível construir o quadro de valores que define a forma de onda a gerar, valores estes que são tensão versus tempo. A figura 6.4 apresenta o quadro de valores para a função logaritmo.

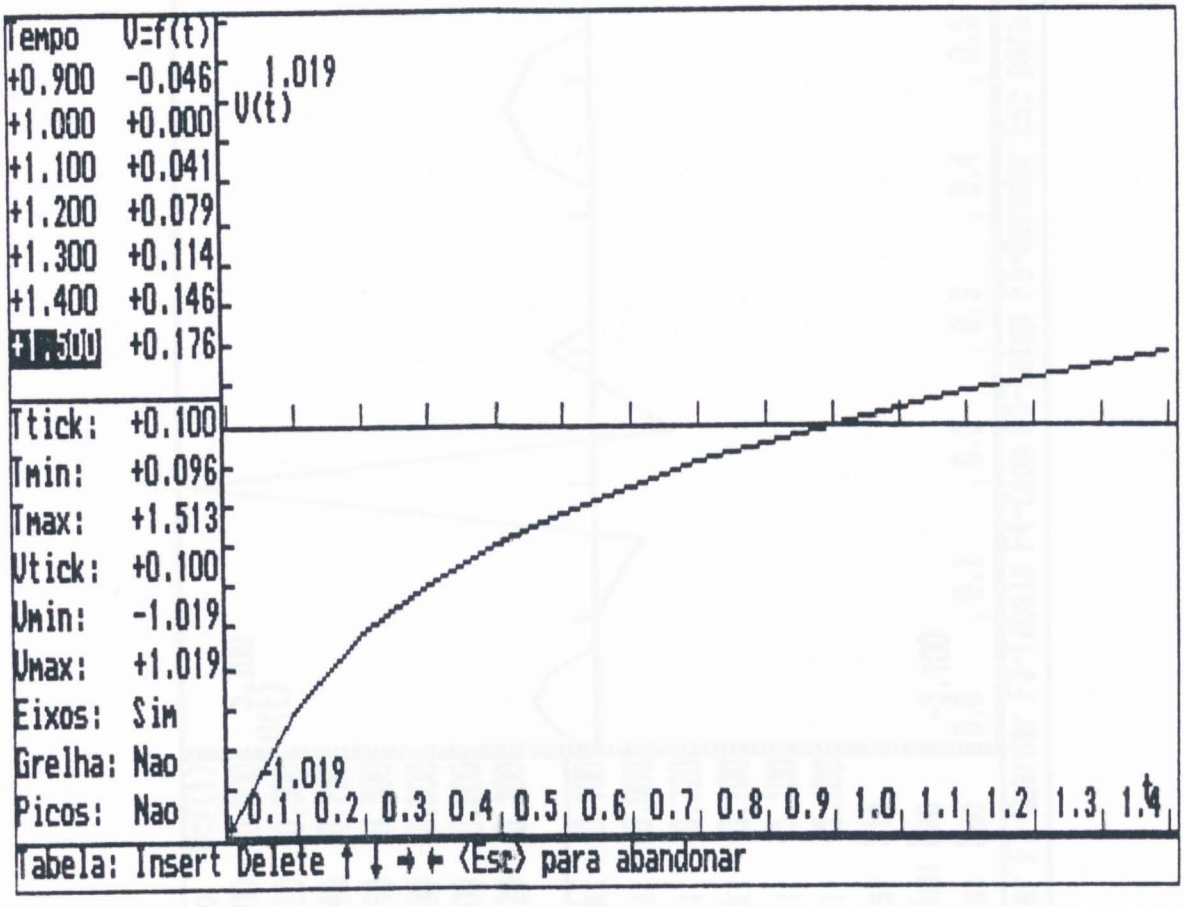


Figura 6.4: Quadro de valores da função logaritmo

Premindo a tecla F4 é possível fazer um Zoom da forma de onda gerada, conforme se pode visualizar nas figuras 6.5(a) e 6.5(b).

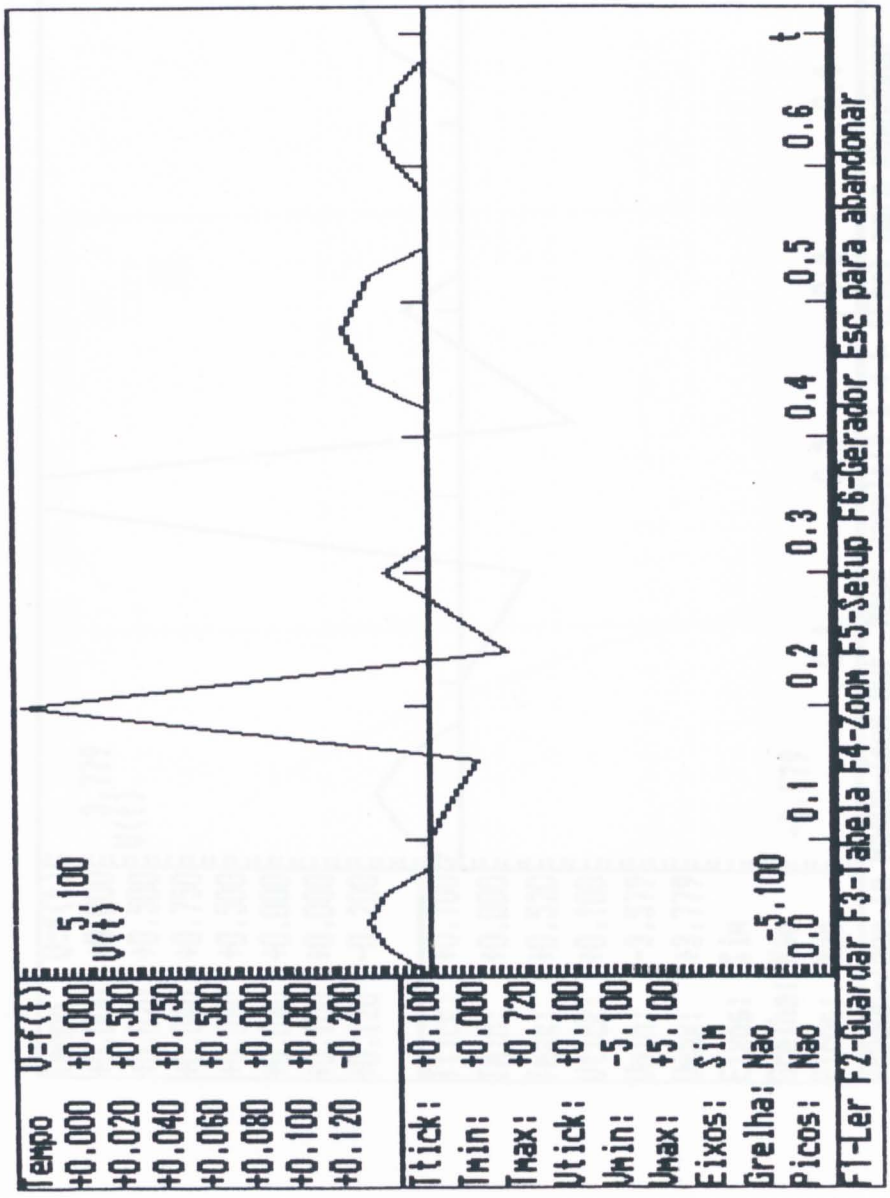


Figura 6.5(a): Função Zoom, normal

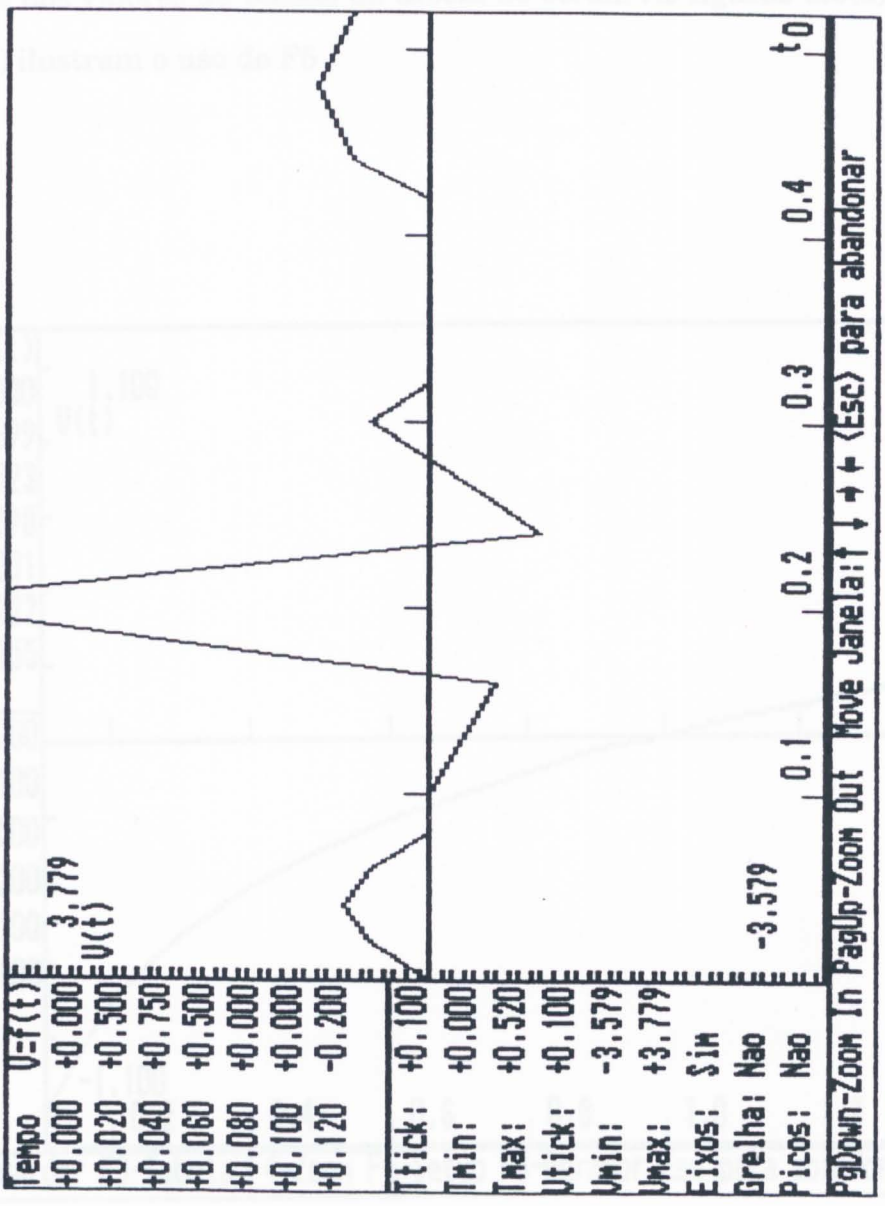


Figura 6.5(b): Função Zoom, amplificado

Premindo a tecla F5 é possível definir os espaçamentos do tempo, o seu valor inicial e final, define-se ainda o espaçamento das unidades da tensão, e a gama de tensões onde a forma de onda vai estar inserida. Nesta tecla ainda são possíveis as seguintes opções: Definir os eixos, establecer uma grelha e o aparecimento dos valores de tensão da tabela no ecran. As figuras 6.6(a), 6.6(b), 6.6(c) e 6.6(d) ilustram o uso de F5

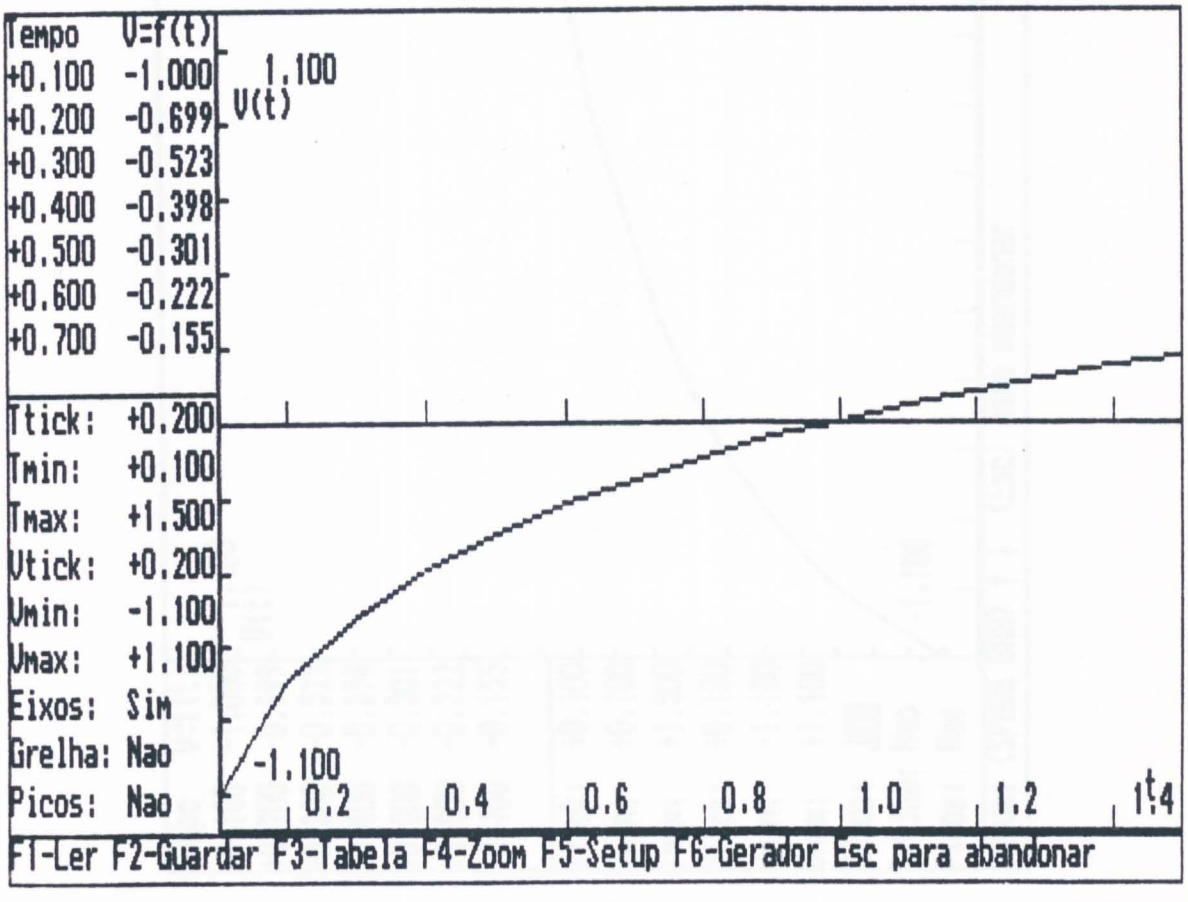


Figura 6.6(a):Função setup, espaçamentode unidades

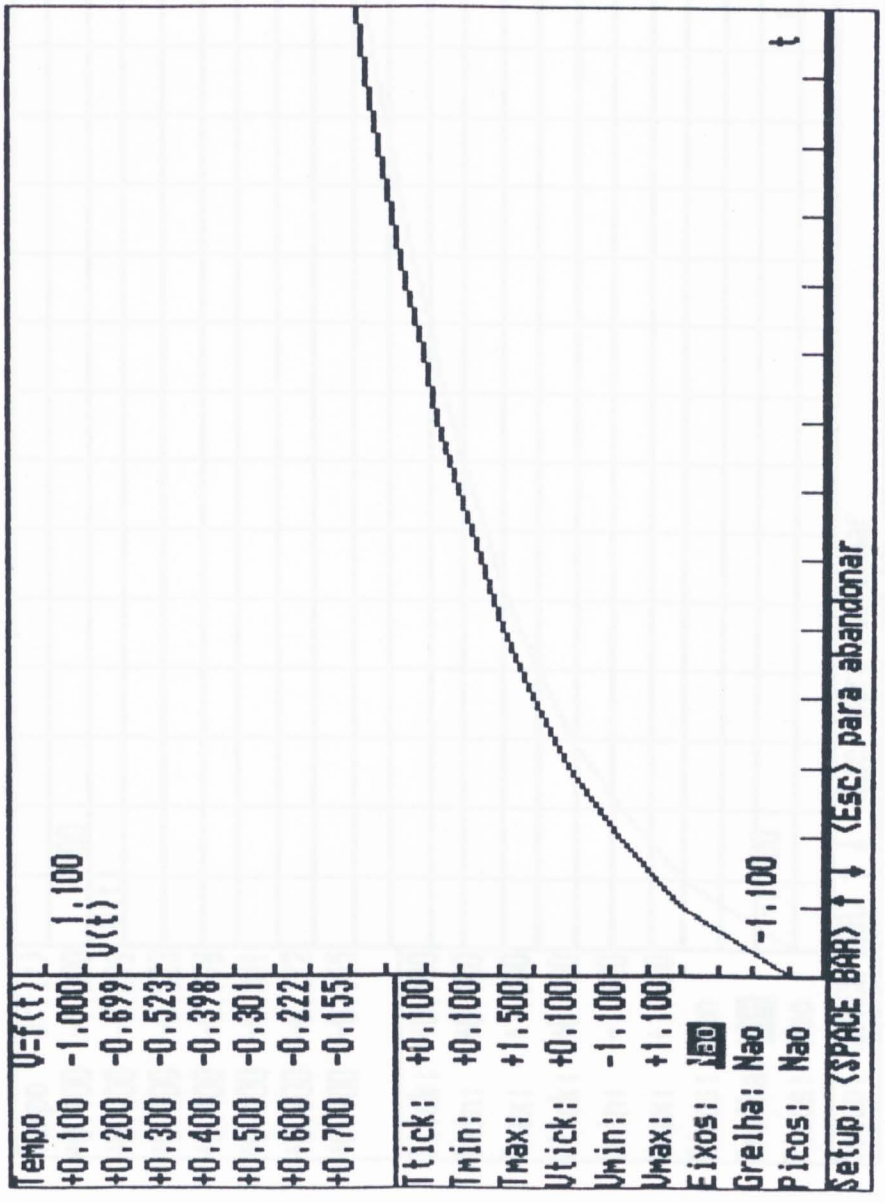


Figura 6.6(b): Função setup, definição de eixos

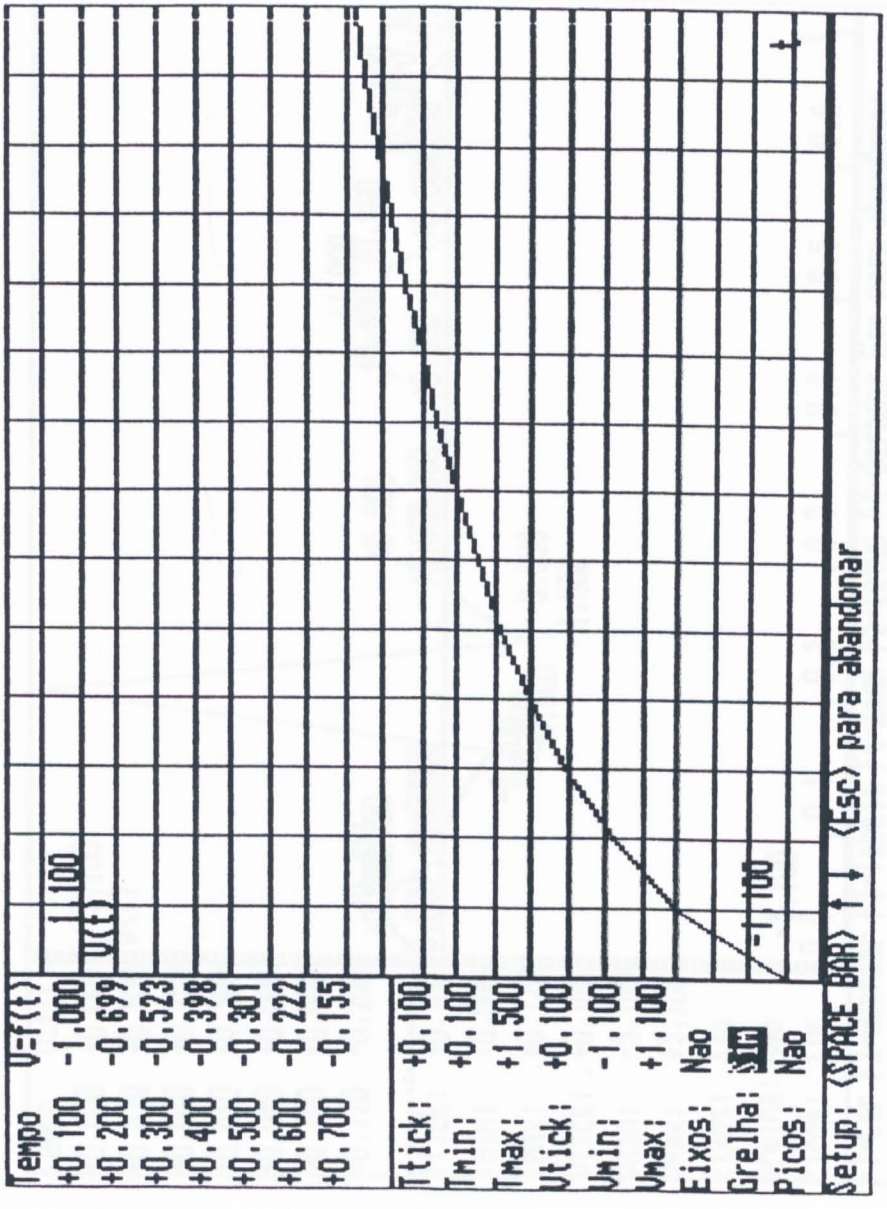


Figura 6.6(c): Função setup, definição da grelha

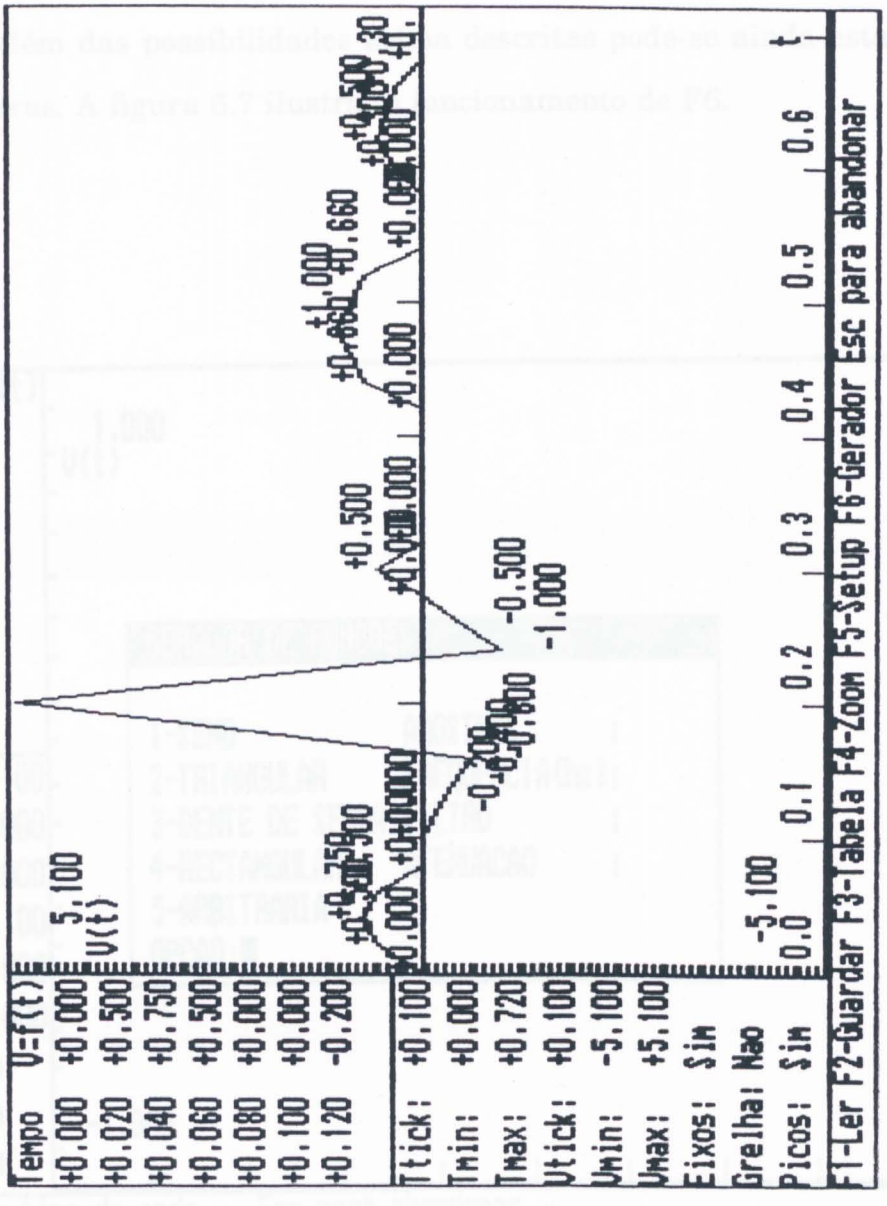


Figura 6.6(d): Função setup, definição de valores de tensão no ecran

Premindo F6 é possível ter acesso à janela do gerador de funções, que é constituida por 5 opções, quatro para ondas predefinidas e uma arbitrária.

Nas onda predefinidas é possivel definir a frequência da onda a gerar, a sua amplitude e se se pretende que esta seja filtrada ou não, para as ondas arbitrárias, além das possíbilidades acima descritas pode-se ainda estabelecer o nº de amostras. A figura 6.7 ilustra o funcionamento de F6.

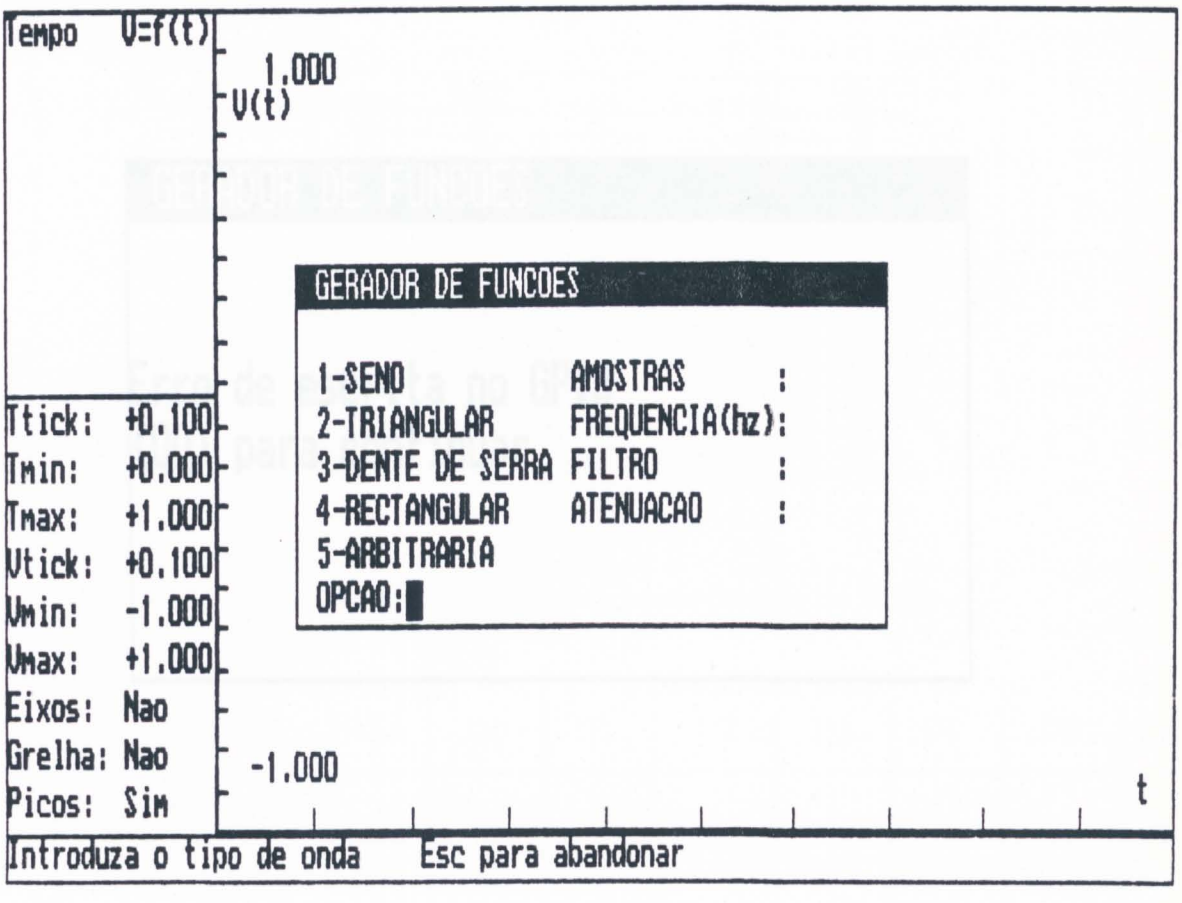


Figura 6.7: Menu de opções de formas de onda

Para abandonarmos a aplicação prime-se a tecla ESC.

No caso de ocorrência de erro na transferência de informação aparece no ecran uma mensagem de erro, figura 6.8.

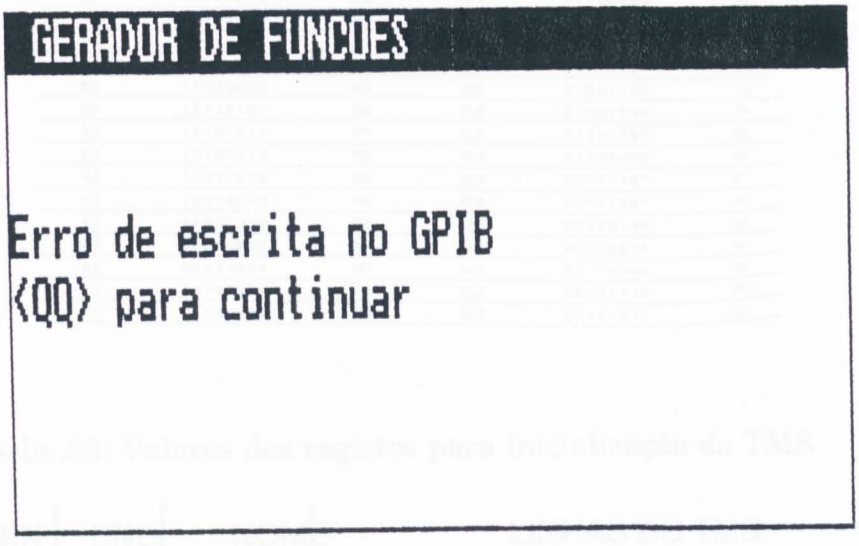


Figura 6.8: Mensagem de erro de transmição

O software em turbo pascal que realiza esta aplicação encontra-se em D5 do apêndice D.

APÊNDICE A

Tabela A1: Código de offset

MSB	LSB	
1111	1111 1111	$+V_{IN}(\frac{2047}{2048})$
1000	0000 0001	$+V_{IN}(\frac{1}{2048})$
1000	0000 0000	0 V
0111	1111 1111	$-V_{IN}(\frac{1}{2048})$
0000	0000 0000	$-V_{IN}(\frac{2048}{2048})$

Tabela A2: Código para controlo de variação de amplitude

ATTN. dB	DAC INPUT CODE	CODE IN DECIMAL	ATTN. dB	DAC INPUT CODE	CODE IN DECIMAL
0	11111111	255	8.0	01100110	102
0.5	11110010	242	8.5	01100000	96
1.0	11100100	228	9.0	01011011	91
1.5	11010111	215	9.5	01010110	86
2.0	11001011	203	10.0	01010001	81
2.5	11000000	192	10.5	01001100	76
3.0	10110101	181	11.0	01001000	72
3.5	10101011	171	11.5	01000100	68
4.0	10100010	162	12.0	01000000	64
4.5	10011000	152	12.5	00111101	61
5.0	10010000	144	13.0	00111001	57
5.5	10001000	136	13.5	00110110	54
6.0	10000000	128	14.0	00110011	51
6.5	01111001	121	14.5	00110000	48
7.0	01110010	114	15.0	00101110	46
7.5	01101100	108	15.5	00101011	43

Tabela A3: Valores dos registos para inicialização do TMS

WE	DBIN	END REG.	NOME REG	LINHAS DO TMS D0	D1	D2	D3	D4	D5	D6
0	0	011	AUX. C	1	X	X	0	0	0	0
0	0	000	I.MASKO	X	X	1BI	0 B0	1 END	0SPAS	0RLC
0	0	001	I.MASK1	0GET	0ERR	OVNC	0APT	1DCAS	1MA	0SRQ
0	0	100	ADRESS	OEDPA	0DAL	1DAT	0	1	0	1
0	0	011	AUX. C	1	X	X	0	1	0	0
0	0	011	AUX. C	1	X	X	0	0	0	1
0	0	011	AUX. C	0	0	0	0	0	0	0

Tabela A4:Valores digitais das formas de onda predefinidas

Tabela A4.1: Correspondência entre valores digitais e analógicos da função seno

AMOSTRA	GRAUS	RAD.	FUNÇÃO	CÓDIGO BINÁRIO	CÓDIGO HEXAD.	ENDEREÇO DA AMOSTRA (IX)
1	0	0	0	1000 0000 0000	0800 H	(2000)
2	45	Π/4	0.7071	1101 1010 1000	0D98 H	(2002)
3	90	Π/2	1	1111 1111 1111	0FFF H	(2004)
4	135	3/4Π	0.7071	1101 1010 1000	0D98 H	(2006)
5	180	Π	0	1000 0000 0000	0800 H	(2008)
6	225	5/4Π	-0.7071	0010 0101 1000	0258 H	(200A)
7	270	3/2Π	-1	0000 0000 0000	0000 H	(200C)
8	315	7/4Π	-0.7071	0010 0101 1000	0258 H	(200E)

Tabela A4.2: Correspondência entre os valores digitais e analógicos da função triangular

AMOSTRA	GRAUS	RAD.	FUNÇÃO	CÓDIGO BINÁRIO	CÓDIGO HEXAD.	ENDEREÇO DA AMOSTRA (IX)
1	0	0	0	1000 0000 0000	0800 H	(2000)
2	45	Π/4	0.5	1100 0000 0000	0C00 H	(2002)
3	90	Π/2	1	1111 1111 1111	0FFF H	(2004)
4	135	3/4Π	0.5	1100 0000 0000	0C00 H	(2006)
5	180	Π	0	1000 0000 0000	0800 H	(2008)
6	225	5/4Π	-0.5	0100 0000 0000	0400 H	(200A)
7	270	3/2Π	-1	0000 0000 0000	0000 H	(200C)
8	315	7/4Π	-0.5	0100 0000 0000	0400 H	(200E)

Tabela A4.3: Correspondência entre valores digitais e analógicos da função dente de serra

AMOSTRA	GRAUS	RAD.	FUNÇÃO	CÓDIGO BINÁRIO	CÓDIGO HEXAD.	ENDEREÇO DA AMOSTRA (IX)
1	0	0	0	1000 0000 0000	0800 H	(2000)
2	45	π/4	1/8	1001 0000 0000	0900 H	(2002)
3	90	π/2	2/8	1010 0000 0000	0A00 H	(2004)
4	135	3/4π	3/8	1011 0000 0000	0B00 H	(2006)
5	180	π	4/8	1100 0000 0000	0C00 H	(2008)
6	225	5/4π	5/8	1101 0000 0000	0D00 H	(200A)
7	270	3/2π	6/8	1110 0000 0000	0E00 H	(200C)
8	315	7/4π	7/8	1111 0000 0000	0F00 H	(200E)

Tabela A4.4: Correspondência entre os valores digitais e analógicos da função quadrada

AMOSTRA	GRAUS	RAD.	FUNÇÃO	CÓDIGO BINÁRIO	CÓDIGO HEXAD.	ENDEREÇO DA AMOSTRA (IX)
1	0	0	1	1111 1111 1111	0FFF H	(2000)
2	45	π/4	1	1111 1111 1111	0FFF H	(2002)
3	90	π/2	1	1111 1111 1111	0FFF H	(2004)
4	135	3/4π	1	1111 1111 1111	0FFF H	(2006)
5	180	π	-1	0000 0000 0000	0000 H	(2008)
6	225	5/4π	-1	0000 0000 0000	0000 H	(200A)
7	270	3/2π	-1	0000 0000 0000	0000 H	(200C)
8	315	7/4π	-1	0000 0000 0000	0000 H	(200E)

Tabela A5: Relação experimental entre os valores dos registos C, D e o valor de frequência obtido

D	C	F
1	01	1087,00 HZ
1	02	1000,00 HZ
1	03	925,00 HZ
1	04	862,00 HZ
1	05	819,00 HZ
1	06	781,00 HZ
1	07	735,00 HZ
1	08	694,00 HZ
1	09	666,00 HZ
1	0A	641,00 HZ
1	0B	609,75 HZ
1	0C	580.40 HZ
1	0D	555.00 HZ
1	0E	526.00 HZ
1	0F	500.00 HZ
1	10	500.00 HZ
1	11	476.00 HZ
1	12	454.00 HZ
1	13	454.00 HZ
1	14	435.00 HZ
1	15	416.60 HZ
1	16	416.00 HZ
1	17	400,00 HZ
1	18	384.00 HZ
1	19	370.00 HZ
1	1A	370.00 HZ
1	1B	357,00 HZ
1	1C	345,00 HZ
1	1D	333.30 HZ
1	1E	333.00 HZ
1	1F	323.00 HZ
1	2F	238.00 HZ
1	3F	185.00 HZ

D	C	F
1	4F	156,00 HZ
1	5F	132.00 HZ
1	6F	116.00 HZ
1	7F	102,00 HZ
1	8F	90,90 HZ
1	9F	83,33 HZ
1	AF	75.75 HZ
1	BF	69.44 HZ
1	CF	64,10 HZ
1	DF	59.50 HZ
1	EF	55.55 HZ
01	0	52,00 HZ
02	0	27,03 HZ
03	0	17.85 HZ
04	0	13.15 HZ
05	0	10.86 HZ
06	0	8.92 HZ
07	0	7,81 HZ
08	0	6,75 HZ
09	0	5.95 HZ
0A	0	5.13 HZ
0B	0	4.93 HZ
0C	0	4.54 HZ
0D	0	4,16 HZ
0E	0	3,84 HZ
0F	0	3,57 HZ
10	0	3,44 HZ
11	0	3,25 HZ
12	0	3,03 HZ
13	0	2,85 HZ
14	0	2,73 HZ
15	0	2,56 HZ
16	0	2,51 HZ
17	0	2,38 HZ

Tabela A5 (continuação)

D	C	F
18	0	2,27 HZ
19	0	2,17 HZ
1A	0	2,08 HZ
1B	0	2,00 HZ
1C	0	1,92 HZ
1D	0	1,85 HZ
1E	0	1,78 HZ
1F	0	1,72 HZ
.	.	.
.	.	.
.	.	.
FF	0	0,21 HZ

A.6 - Registos do TMS

Registos de leitura

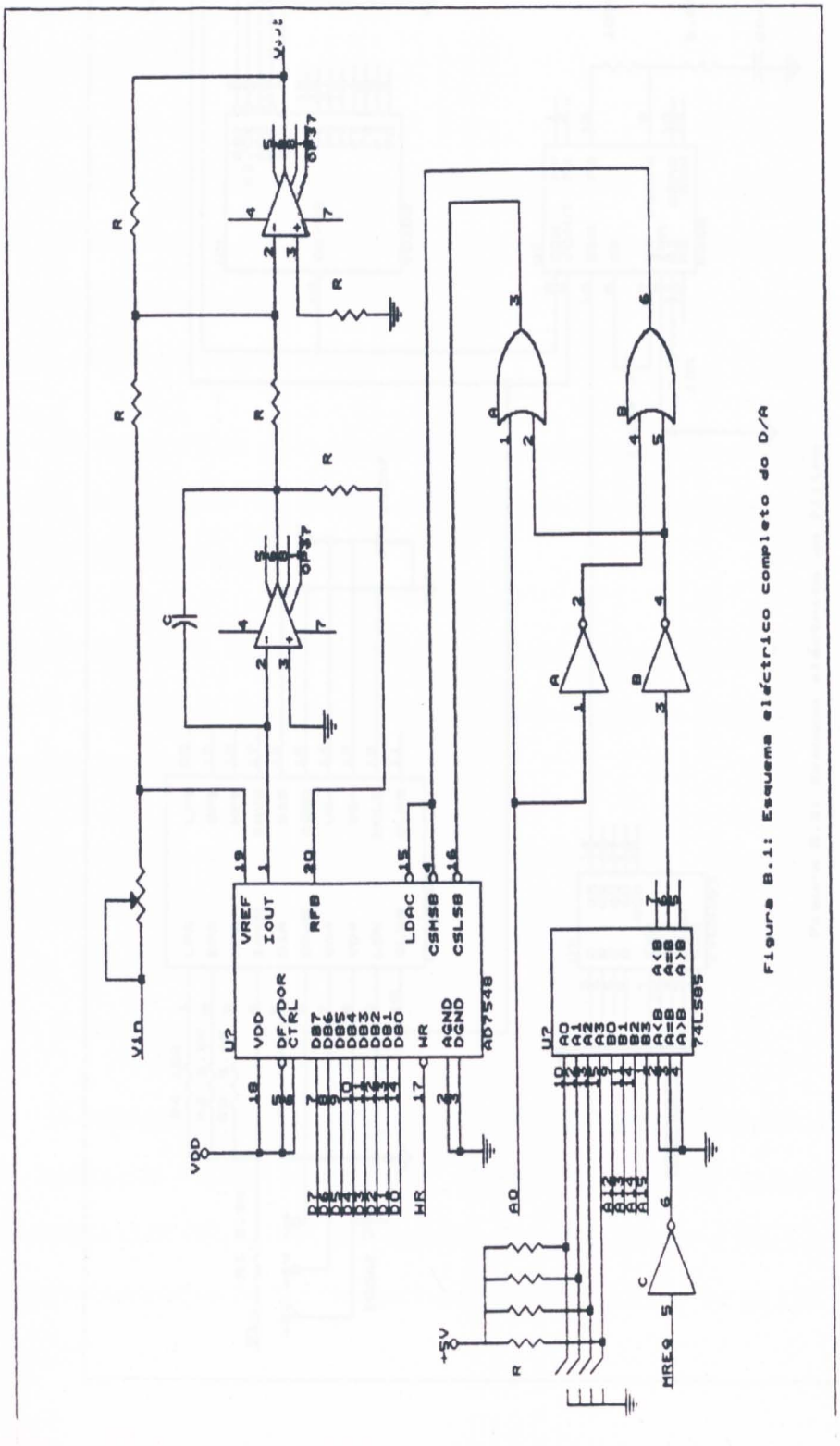
Endereço RS2	RS1	RS0	nome	linha D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	Int Status 0	INT0	INT1	BI	BO	END	SPAS	RLC	MAC
0	0	1	Int Status 1	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC
0	1	0	Address Status	REM	LLO	ATN	LPAS	TPAS	LADS	TADS	ulpa
0	1	1	Bus Status	ATN	DAV	NDAC	NRFD	EOI	SRQ	IFC	REN
1	0	0	*								
1	0	1	*								
1	1	0	Cmd Pass Thru	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
1	1	1	Data In	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1

Registos de escrita

Endereço RS2	RS1	RS0	nome	linha D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	Int Mask 0			BI	BO	END	SPAS	RLC	MAC
0	0	1	Int Mask 1	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC
0	1	0	*	xx	xx	xx	xx	xx	xx	xx	xx
0	1	1	Auxiliary Cmd	cs	xx	xx	f4	f3	f2	f1	f0
1	0	0	Address	edpa	dal	dat	A5	A4	A3	A2	A1
1	0	1	Serial Poll	S8	rsvl	S6	S5	S4	S3	S2	S1
1	1	0	Parallel Poll	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1
1	1	1	Data Out	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1

APÊNDICE B

B1: Esquema eléctrico da placa de conversão de informação digital em informação analógica.



B2: Esquema eléctrico do filtro

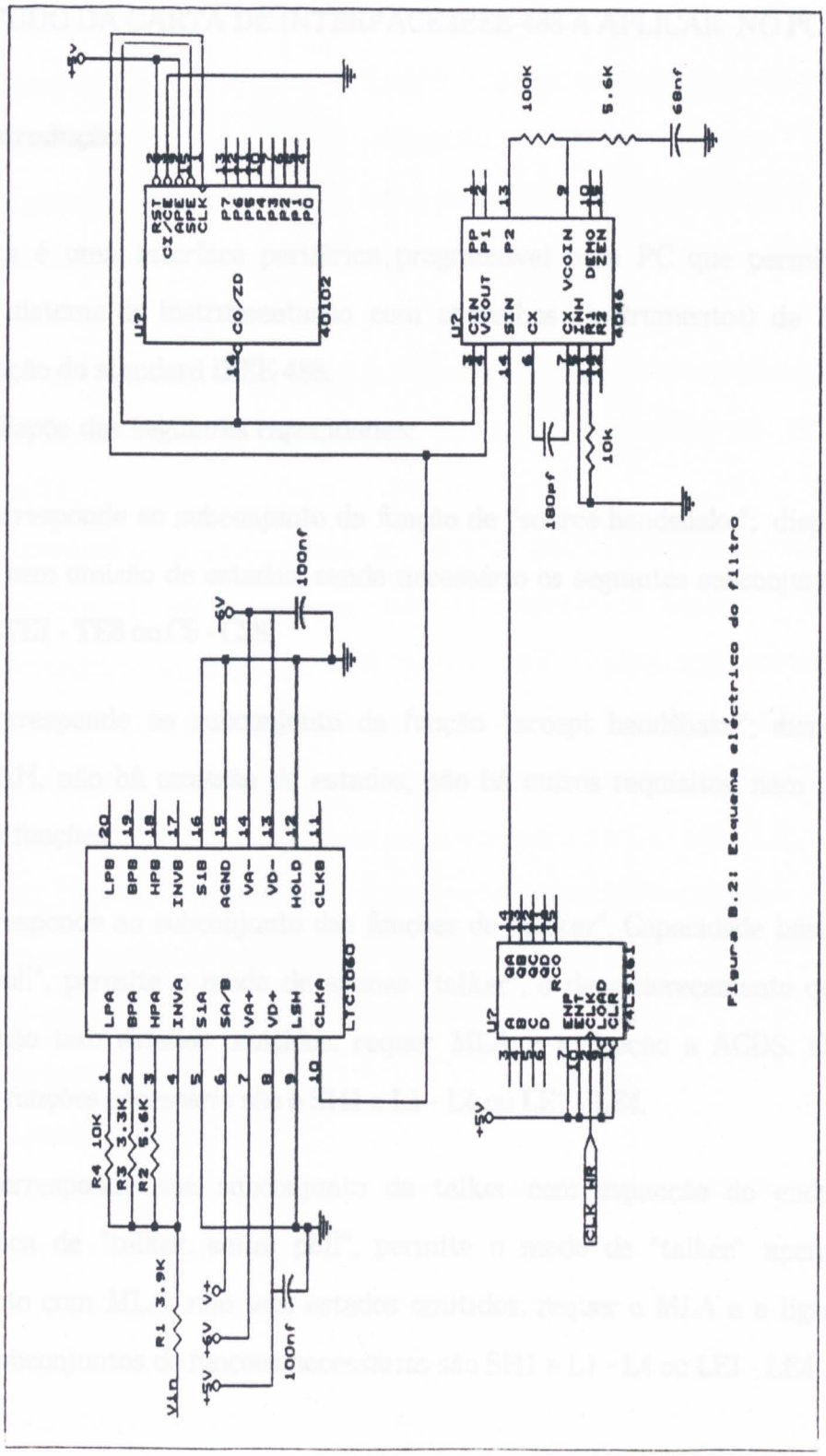


Figura B.2: Esquema eléctrico do filtro

APÊNDICE C

C.1 - ESTUDO DA CARTA DE INTERFACE IEEE-488 A APLICAR NO PC

C.1.1 - Introdução

Esta carta é uma interface periférica programável para PC que permite um controlo de um sistema de instrumentação com aparelhos (instrumentos) de acordo com a especificação do standard IEEE-488.

A carta dispõe das seguintes capacidades:

SH1 - Corresponde ao subconjunto da função de "source handshake"; dispõe da capacidade SH, sem omisão de estados, sendo necessário os seguintes subconjuntos de funções T1 - T8, TE1 - TE8 ou C5 - C28.

AH1 - Corresponde ao subconjunto da função "accept handshake"; dispõe da capacidade de AH, não há omissão de estados, não há outros requisitos, nem outros subconjuntos de funções.

T5 - Corresponde ao subconjunto das funções de "talker". Capacidade básica de "talker, serial poll", permite o modo de apenas "talker", o desendereçamento quando surge o MLA; não tem estados omitidos, requer MLA e a ligação a ACDS. Outros subconjuntos de funções necessário são o SH1 e L1 - L4 ou LE1 - LE4.

TE5 - Corresponde aoa subconjunto de talker com expanção do endereço. Capacidade básica de "talker, serial poll", permite o modo de "talker" apenas, o desendereçamento com MLA; não tem estados omitidos, requer o MLA e a ligação a ACDS. Outros subconjuntos de funções necessárias são SH1 e L1 - L4 ou LE1 - LE4.

L3 - Corresponde ao conjunto da função "listener". Capacidade básica de "listener", modo de apenas "listener", desendereçamento com MTA. Não tem estados omitidos, requer o MTA e a ligação a ACDS. Outros subconjuntos de funções necessárias são AH1 e T1 - T8 ou TE1 - TE8

LE3 - Corresponde ao subconjunto da função "listener com expanção de endereço. Capacidade básica de "listener" com expanção, modo de apenas "listener", desendereçável perante MSA e TPAS; não tem estados omitidos, requer MSA e a ligação a TPAS e ACDS. Outros subconjuntos de funções necessárias são AH1 e T1 - T8 ou TE1 - TE8

SR1 - Corresponde ao subconjunto SR. Capacidade completa de pedido de serviço. Não tem estados omitidos. Outros subconjuntos de funções nesseçárias são: T1, T2. T5, T6, TE1, TE5, ou T6.

PP1 - Corresponde ao subconjunto de "parallel poll".Configuração remota, não tem estados omitidos, não são necessários outros subconjuntos de funções.

PP2 - Corresponde ao subconjunto de "parallel poll". Configuração remota. Estão omitidos os estados PUCS e PACS. Não necessita de outros subconjuntos de funções.

RL1 - Subconjunto das funções de comando remoto ("remote local"). Capacidade completa, não tem estados omitidos, necessita dos subconjuntos L1 - L4 ou LE1 - LE4.

C1, C2, C3, C4 e C5 - Pertencem ao subconjunto da função de controlo.

C1 - Tem a capacidade de controlar o sistema.

C2 - Pode enviar "interface clear" e tomar o controlo.

C3 - Pode enviar REN.

C4 - Responde ao SRQ.

C5 - Envia mensagem de interface, recebe o controlo, passa o controlo, o "parallel poll, toma o controlo sincronamente.

E1 e E2 - Subconjunto das caracteristicas eléctricas.

E1 - "Drivers" de colector aberto.

E2 - "Drivers three-state".

C.1.2 - **Configuração do hardware**

Antes da utilização da carta torna-se necessário que o canal de D.M.A., o "interrupt" e os endereços da base de entradas/saída sejam devidamente colocados.

C.1.2.1 - Base de endereços de Entrada/Saída

Existe um selector, que é usado para seleccionar o porto de endereço entrada/saída desde 000H até 3F8H com saltos de 8. A base de endereços ocupa os oito endereços contínuos na porta I/O para ser possível seleccionar um dos regitos do µPD7210.

A figura C.1 mostra os endereços relativos da base de endereço de I/O 2B8H e os respectivos registos do µPD7210.

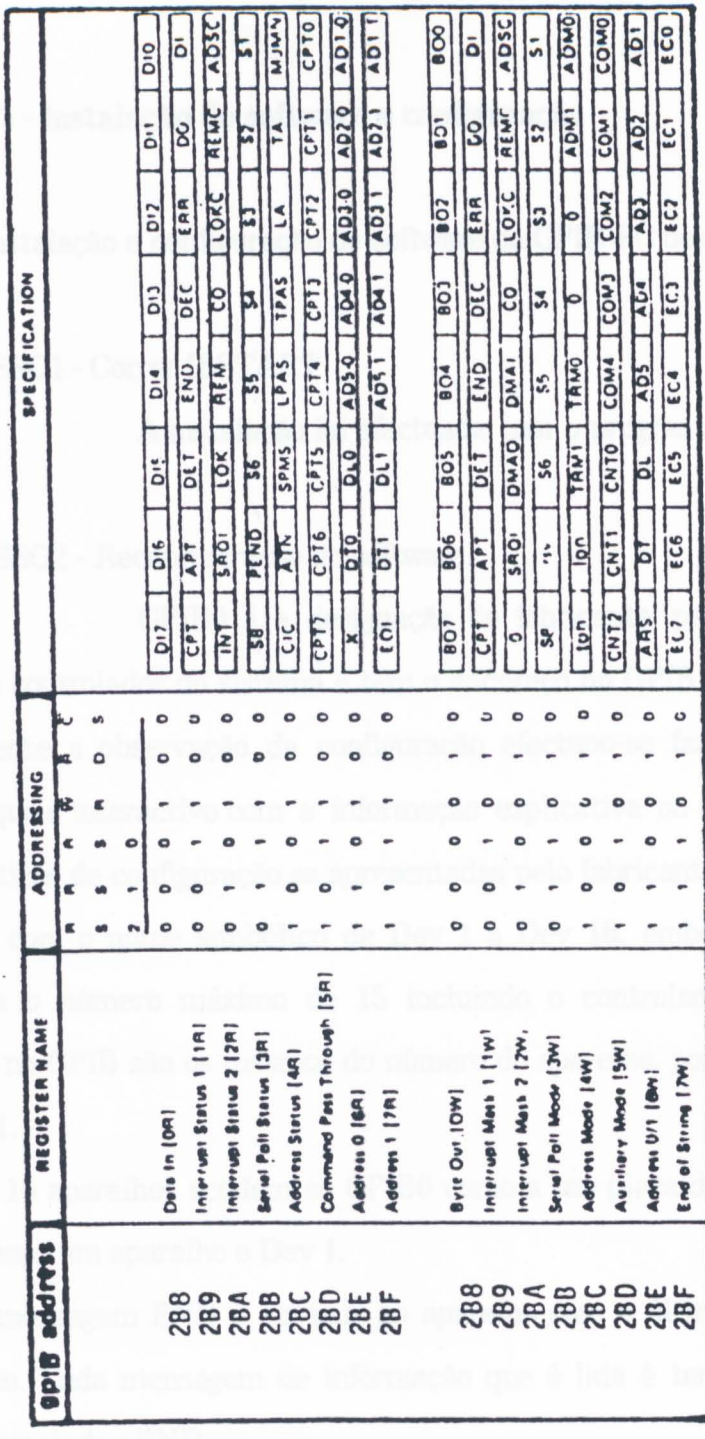
C.1.2.2 - Selecção do canal DMA

Através de uma ponte ("jumper") referenciado por JP1 e dos contactos 7 e 8 do selector SW2 é possivel seleccionar o canal de DMA.

Nesta aplicação optou-se por manter o canal 1 de entre o 1, 2 e 3, uma vez que era o que vinha estabelecido de fábrica. Verificou-se no manual de referência técnica do PC que este canal correspondia a uma reserva no BUS de expanção usado pela porta série externa SDLC ("Synchronous Data Linck Control").

C.1.2.3 - Interrupção

A linha de interrupção de TLC foi seleccionada escolhendo IRQ7, que mais uma vez era o proposto como ponto de partida pelo fabricante. Verificou-se novamente no manual de referência técnica do PC estar disponivel no BUS de expanção para a porta porta de impressora paralela.



GPIB address	REGISTER NAME	ADDRESSING						SPECIFICATION								
		A 2	A 1	A 0	R/W	RS										
2B8	Data In (0R)	0	0	0	1	0	0	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	
2B9	Interrupt Status 1 (1R)	0	0	1	1	0	0	CPT	APT	DET	END	DEC	ERR	DO	DI	
2BA	Interrupt Status 2 (2R)	0	1	0	1	0	0	INT	SRQI	LOK	REM	CO	LOKC	REMC	ADSC	
2BB	Serial Poll Status (3R)	0	1	1	1	0	0	S8	PEND	S6	S5	S4	S3	S2	S1	
2BC	Address Status (4R)	1	0	0	1	0	0	CIC	ATN	SPMS	LPAS	TPAS	LA	TA	MJMN	
2BD	Command Pass Through (5R)	1	0	1	1	0	0	CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	
2BE	Address 0 (6R)	1	1	0	1	0	0	X	DT0	DL0	AD5.0	AD4.0	AD3.0	AD2.0	AD1.0	
2BF	Address 1 (7R)	1	1	1	1	0	0	EOI	DT1	DL1	AD5.1	AD4.1	AD3.1	AD2.1	AD1.1	
2B8	Byte Out (0W)	0	0	0	0	1	0	BO7	BO6	BO5	BO4	BO3	BO2	BO1	BO0	
2B9	Interrupt Mask 1 (1W)	0	0	1	0	1	0	CPT	APT	DET	END	DEC	ERR	DO	DI	
2BA	Interrupt Mask 2 (2W)	0	1	0	0	1	0	0	SRQI	DMAO	DMAI	CO	LOKC	REMC	ADSC	
2BB	Serial Poll Mode (3W)	0	1	1	0	1	0	S8	rsv	S6	S5	S4	S3	S2	S1	
2BC	Address Mode (4W)	1	0	0	0	1	0	ton	lon	TAM1	TAM0	0	0	ADM1	ADM0	
2BD	Auxiliary Mode (5W)	1	0	1	0	1	0	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0	
2BE	Address 0/1 (6W)	1	1	0	0	1	0	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1	
2BF	End of String (7W)	1	1	1	0	1	0	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0	

Figura C1: Endereços relativos da base de I/O 2B8H

C.1.3 - **Instalação do software e configuração**

A instalação e configuração de software do GPIB-PC desenvolve-se por 4 passos:

PASSO1 - Correr IBSTART

A instalação foi efectuada com o programa IBSTART que instala o software.

PASSO2 - Reconfiguração do software.

GPIB0 é a designação do fabricante para a placa GPIB-PC II. GPIB0 é o controlador do sistema e tem o endereço no GPIB de 0. A reconfiguração ou simplesmente a observação da configuração efectuou-se fazendo correr o programa IBCONF que é interactivo com a informação explicativa no ecran. Tomaram-se como características de configuração as apresentadas pelo fabricante. É possível disopor de 16 aparelhos com o nome simbólico de Dev 1 a Dev 16, embora o standard IEEE 488 estabeleça o número máximo de 15 incluindo o controlador. Os endereços destes aparelhos no GPIB são os mesmos do número do aparelho, por exemplo o Dev 1 está no endereço 1.

Os 16 aparelhos acedem ao GPIB0 como a sua placa de acesso. Nesta aplicação temos apenas um aparelho o Dev 1.

A mensagem END é enviada ao aparelho com o último byte de mensagem de informação. Cada mensagem de informação que é lida é terminada automaticamente quando é recebido o END.

Nenhum modo EOS é activo, isto é, não há "string" a informar o fim da mensagem. O tempo limite de espera das funções e I/O é aproximadamente de 10s. A base de endereçamento do GPIB0 é 2B8, usa DMA no canal 1 e linha de interrupção TLC IRQ7. como já foi dito anteriormente. Seguidamente faz-se uma ilustração desta configuração e as características do aparelho.

CONFIGURAÇÃO DA PLACA DO GPIB0

Endereço primário do GPIB ---------- 0

Endereço secundário ---------- Não tem

Tempo de saída ---------- 10s

Byte de EOS ---------- 00

Fim de leitura com EOS ---------- Sem

Colocação na escrita de EOI com EOS ---------- Sem

Comparação no EOS ---------- 7 bits

Colocação do EOI com o último byte de escrita ---------- Sim

Modelo do GPIB-PC ---------- PC2

Placa controladora do siatema ---------- Sim

Bloqueamento do comando local dos aparelhos ---------- Sim

Inibição auto "polling serie" ---------- Sim

Inibição de aparelhos não endereçados ---------- Não

Tempo de velocidade alta ---------- Não

Colocação de "jumpers" de interrupção ---------- Não tem

Endereço base do I/O ---------- 2B8(H)

Canal do DMA ---------- 1

Frequência de relógio ---------- 8 MHz

CARACTERíSTICAS DO APARELHO

Aparelho ---------- TMS 9914A

Acesso ---------- Através do GPIB0

Endereço primário para o GPIB ---------- 10

Endereço secundário para o GPIB ---------- Não tem

Tempo de colocação ---------- 10s

Byte de EOS ---------- 00 (H)

Leitura terminada em EOS ---------- Não

Colocação de EOI com EOS na escrita ---------- Não

Colocação de EOI com o último byte de escrita ---------- Sim

PASSO3 - Reinício do sistema de funcionamento.

Teve de se reiniciar o sistema operativo para o DOS carregar o programa operador do GPIB-PC.

PASSO4 - Verificação da instalação.

Fez-se correr o programa IBTEST. Assim a instalação utiliza uma série de ficheiros que constituem o "package" do software do GPIB-PC que dispõe de uma série de ficheiros principais:

GPIB.COM - É o ficheiro operador do "drive", é carregado através do sistema de arranque pelo sistema operativo DOS.

BIB.M - É o ficheiro de "interface" de linguagem que permite a um programa de aplicação em BASIC ou QUICKBASIC, aceder ao operador GPIB-PC.

QBIB*.OBJ - É um ficheiro de interface de linguagem que permite a um programa de aplicação em QUICKBASIC , aceder ao operador do GPIB-PC.

DECL.BAS - É um ficheiro de declaração que contem os códigos a serem colocados no princípio de programas de aplicação em BASIC ou QUICKBASIC.

QBDECL.BAS - É um ficheiro de declaração que contem os códigos a serem colocados no princípio de programas de aplicação em QUICKBASIC.

E programas e ficheiros adicionais tais como:

APPMON.COM - É um programa monitor, residente, útil na sequência de "debugging" das chamadas do GPIB e uma aplicação pretendida.

IBTRAP.EXE - É um programa para configurar a aplicação do monitor.

IBSTART.BAT - É um ficheiro usado para a instalação e arranque. É um programa com vários fins que permitem a instalação do software. Copia ficheiros, modifica o ficheiro de configuração do sistema DOS e ensaia o hardware.

IBDIAG.EXE - É um programa de ensaio do hardware a usar antes que o software do GPIB tenha sido configurado e instalado.

IBCONF.EXE - É um programa de configuração de software que permite alterar os parâmetros e outros dados usados pelo operador.

IBIC.EXE - É um programa de controlo interactivo que permite controlar o sistema de instrumentação a partir do teclado. Ensina a usar as funções, a programar os instrumentos ou outro aparelho GPIB e a desenvolver os programas de aplicação.

De seguida dão-se alguns exemplos de programas para BASIC, QUICKBASIC E IBIC.

DBSAMP.BAS, DQSAMP.BAS, DIBSAMP.BAS, BIBSAMP.BAS E BBSAMP.BAS

C.2 - ESTUDO DAS CAPACIDADES DA "INTERFACE" ENTRE O MICRO E O BUS

Esta "interface" apresenta um subconjunto de funções já descritas em C1 para o GPIB: SH1, AH1, T5, TE5, L3, LE3 e RL1. No entanto destingue-se do GPIB no seguinte subconjunto de funções:

SR0 - Corresponde ao subconjunto da função SR("Service Request"). Sem capacidade de serviço. Tem todos os estados omitidos. Não necessita de outros subconjuntos de funções.

PP0 - Corresponde ao subconjunto de "parallel poll". Sem capacidade. Todos os estados estão omitidos. Não são necessários outros subconjuntos de funções.

DT0 - Não tem capacidade de entrar em funcionamento em grupo com outros aparelhos.

C0 - Corresponde ao subconjunto das funções de controlo, sem capacidade de ser controlador do sistema.

E2 - Corresponde ao subconjunto de caracteristicas eléctricas E2, que é caracterizada por "drivers three state".

C.3 - ARQUITECTURA DO TMS 9914A

A arquitectura interna do TMS é apresentada na figura C2. Como se pode observar este tem treze registos, seis de leitura e sete de escrita, os quais permitem operar as comunicações entre o BUS IEEE 488 e o μP.

Cada registo é acedido colocando os respectivos endereços RS0, RS1 e RS2 e executando uma leitura ou escrita de memória (WE=1 DBIN=1 e WE=0 DBIN=0 respectivamente). As tabelas A6 mostram os registos, o seu endereço e os seus bits correspondentes. Seguidamente faz-se um breve descrição destes registos.

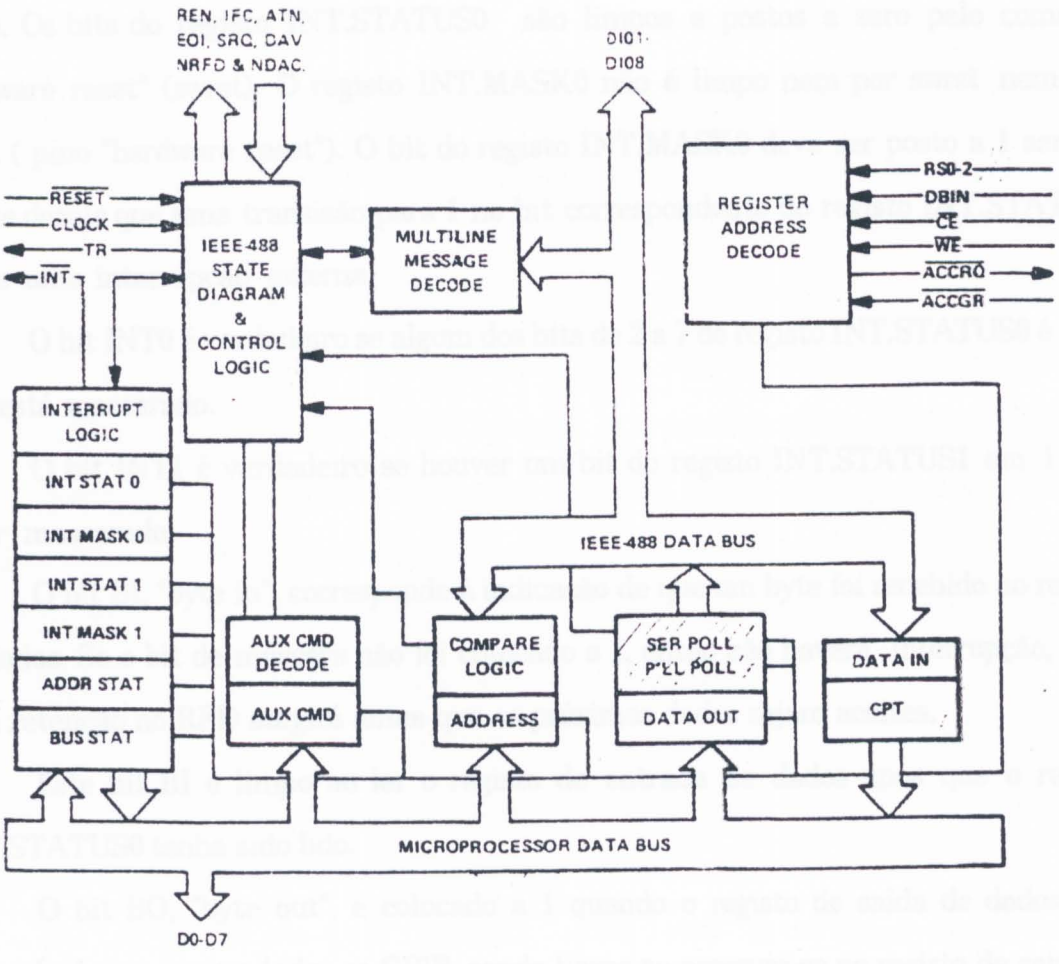


Figura C2: Arquitectura interna do TMS

Registos INT. MASK0 E INT.STATUS0 - Estes registos são acedidos pelo código 000, operando independentemente um do outro.

Os bits de estado de interrupção são colocados a 1 quando o respectivo acontecimento ocorre e tem em atenção o bit de máscara correspondente. Com exepção dos bits INT0 e INT1, os restantes armazenam informação do BUS e são limpos logo após o correspondente registo de interrupção de estado ser lido pelo microprecessador (Z80). Os bits do registo INT.STATUS0 são limpos e postos a zero pelo comando "software reset" (swrst). O registo INT.MASK0 não é limpo nem por swrst nem por reset (pino "hardware reset"). O bit do registo INT.MASK0 deve ser posto a 1 sempre que se deseje que uma transisão para 1 no bit correspondente do registo INT.STATUS0 cause uma interrupção externa.

O bit INT0 é verdadeiro se algum dos bits de 2 a 7 do registo INT.STATUS0 é um e não está mascarado.

O bit INT1 é verdadeiro se houver um bit do registo INT.STATUS1 em 1 sem estar mascarado.

O bit BI, "byte in", corresponde à indicação de que um byte foi recebido no registo de dados. Se o bit de máscara não foi colocado a 1, então não haverá interrupção, mas uma retenção no RFD surgirá antes que os próximos dados sejam aceites.

Este bit BI é limpo ao ler o registo de entrada de dados após que o registo INT.STATUS0 tenha sido lido.

O bit BO, "byte out", é colocado a 1 quando o registo de saída de dados está disponível para enviar dados ao GPIB, sendo limpo ao escrever-se no registo de saída de dados.

O bit "END", indica que o byte acabado receber pelo "listener" foi o último do conjunto, ou seja o recebido com a linha EOI verdadeira e é colocado em simultâneo com o BI.

O bit SPAS indica que o TMS9914 pediu serviço através dos bits NSV1 e NS2, respectivamente no registo "serial poll" ou comando auxiliar colocado em "serial poll".

O bit RLC mudança de local/remoto, é colocado por qualquer transição entre o estado local e remoto.

O bit MAC, mudança do meu endereço, indica que um comando do GPIB foi recebido de que resulta um endereço no estado do endereço do TMS.

Registos INT.MAK1 e INT.STATUS1 - Estes tegistos são acedidos pelo código RS2=0, RS1=0 e RS0=1. Os bits de estado são limpos após serem lidos e através de "swrst".

O registo INT.STATUS1 tem um grupo de interrupts: GET, UNC, APT, DCAS, MA, que respondem a comandos vindos do BUS IEEE 488. Se estiverem desmascarados ocorrerá uma retenção de "data accepted" que poderá ser libertado pelo comando auxiliar "dacr". Deste modo o:

Bit GET é colocado a 1 se um comando de disparo de execução em grupo é recebido. Surgirá uma retenção de DAC se o bit está desmascarado. O ponto TR tornar-se-á alto enquanto a retenção DAC persistir. Se este bit estiver mascarado o pino TR torna-se alto durante 5 ciclos de relógio.

Bit ERR é colocado a 1 se SH setorna activo e encontra as linhas NDAC e NRFD simultâneamente altas, indicando que os dados não são aceites no BUS.

Bit UNC é colocado a 1 perante um comando desconhecido do TMS9914A; haverá umDAC holdoff se este estiver desmascarado; comandos secundários causam este "interrupt" se o comando auxiliar PTS foi previamente recebido.

Bit APT, "adress pass throught, serve para permitir o endereçamento secundário se este estiver desmascarado.

Bit DCAS, "device clear active state" é activado quando o comando "device clear" (DLC) é recebido ou se é recebido o comando "selected device clear" (SDC) pelo TMS9914 quando em LADS. Aparecerá uma retenção em DAC se o bit estiver desmascarado.

Bit SRQ, "service request" é activado quando quando a linha SRQ for verdadeira e pedido serviço ao controlador.

Bit MA, "My Adress", é activado quando o TMS reconhece o seu endereço primário. Causa uma retenção no DAC se estiver desmascarado.

Bit IFC, "interface clear", é activado quando a linha IFC se torna verdadeira e indica que o aparelho regressou ao "idle state".

Registo ADRESS STATUS - Este registo é acedido pelo código RS2=0, RS1=1, RS0=0 e é um registo de leitura cujos bits são:

REM - O aparelho está no estado remoto

LLO - O fecho local está em funcionamento

ATN - A linha de atenção é verdadeira

LPAS - O TMS encontra-se no endereço primário de "listener"

LADS ou LACS - O TMS foi endereçado para ouvir

TADS ou TACS - O TMS foi endereçado para "talker"

ULPA - Este bit mostra o LSB do último endereço reconhecido pelo TMS

Registo ADRESS - Este registo é descrito e acedido por RS2=1, RS1=0 e RS0=0. Os seus bits são:

edpa - Permite o endereçamento primário dual

dal - Inibe a função de "listener"

dat - Inibe a função "talker"

A5 - A1 - Endereço primário do TMS

Registo AUXILIAR COMMAND - Este registo é acedido por RS2=0, RS1=1 e RS0=1. Usado para permitir e inibir a maioria das funções seleccionáveis no TMS e para inicializar muitas das acções dos aparelhos. Esses comandos são:

"Software reset" (swrst) - Que quando activado permite ao TMS regressar ao estado "IDLE", durante o qual não toma qualquer actividade no GPIB. É neste estado que se deve configurar o TMS, nomeadamente escrever o endereço no registo "ADRESS", colocar a máscara nos registos "INT.MASK0" e "INT.MASK1", seleccionar as operações no registo "AUXILIAR COMMAND.

Release DAC holdoff (dacr) - Liberta a retenção da linha DAC, a qual permite ao microprocessador responder a um comando desconhecido, endereço secundário comando de disparo ("device clear"). A libertação dá-se quando a acção é realizada.

Release RFD holdoff (rhdf) - Permite a libertação da retenção da linha RFD causada pelos comandos hdfa e hdfe.

Holdoff on All Data(hdfa) - Causa a retenção em RFD em todos os bytes de dados.

Holdoff on end (hdfe) - Causa a retenção em RFD quando uma mensogem de fim de cadeia de dados é recebida através da "interface".

Set new byte available false (nbaf), suprime um byte que não foi transmitido devido a uma interrupção e que não é mais necessário.

Force group execute trigger (fget) - Actua sobre a linha TR do TMS.

Return local (rtl) - Permite que o microprecessador responda ao controlo feito pelo painel do aparelho.

Force end or identify (feoi) - Permite uma mensagem de EOI a ser enviada no proximo byte.

Listener only (lon) - Activa o estado de "listener" até que um novo comando o desactive.

Talker only (ton) - Activa o estado de "talker" até que um novo comando o desactive.

Go to standby (gts) - é activo pelo controlador colocando a linha ATN falsa.

Take control synchronously (tcs) - permite que o controlador activo retome o controlo sem perda ou corrupção da informação.

Request parallel poll (rpp) - é executado pelo controlador activo para enviar o comando de "parallel poll" para a " interface".

Take control asynchronously (tca) - é usado pelo controlador activo para tomar de emediato o comando, podendo ocorrer perda ou corrupção de transmissão.

Send interface clear (sic) - é colocado pelo controlador do sistema tornando a linha IFC verdadeira.

Send remote enable (sre) - é colocado pelo controlador do sistema tornando a linha REN verdadeira.

Request control (rqc) - Comando enviado pelo microprecessador em resposta ao comando TCT que foi identificado através do comando " pass through", passando o TMS ao estado CACS (controler active estate).

Realese control (ric) - Comando enviado após um TCT no sentido de libertar a linha ATN e passar o controlo a outro aparelho.

Disable all interrupt (dai) - desactiva a linha de "interrupt", no entanto os registos de interrupção e qualquer retenção não são afectados.

Pass through next secondary (pts) - é usado para obter uma configuração remota do parallel poll. O comando de configuração de "parallel poll"(PPC) passa através do TMS como um comando de endereço não conhecido e é identificado pelo microprecessador.

Set T1 delay (stdl) - Permite colocar o tempo de atraso em seis ciclos de relógio se este comando for enviado.

Shadow handshake (shdw) - Permite que o controlador activo obtenha "listener handshake" sem participar na transferência de dados.

Very short delay (vstd1) - Se este comando for enviado o "settling time" t1 será reduzido a três ciclos de relógio ao segundo byte e seguintes.

Request service bit2 (rsv2) - Este comando utiliza a mesma função de rsv1, mas permite um meio de solicitar serviço que é independente do registo "serial poll".

Registo BUS STATUS - É um registo de leitura com RS2=0, RS1=1 e RS0=1. Este permite examinar o GPIB e é contituido pelos seguintes bits: ATN, DAV, NDAC, NRFD, EOI, SRQ, IFC e REN.

Registo SERIAL POLL - É um registo de escrita acedido por RS2=1, RS1=0 e RS0=1, quando o aparelho é endereçado em "serial poll" coloca os bits do registo no GPIB.

Registo COMMAND PASS THROUGH - É um registo de leitura e é acedido com RS2=1,RS1=1 e RS0=0. Permite a observação directa das linhas do GPIB. É ainda usado para o controlador activo ler os resultados de um "parallel poll".

Registo PARALLEL POLL - É um registo de escrita e é acedido com RS2=1,RS1=1 e RS0=0. Põe o seu conteudo no BUS IEEE 488 quando o controlador inicia o "parallel poll".

Registo DATA IN - É um registo de leitura e é acedido com RS2=1, RS1=1 e RS0=1. Este registo armazena os dados do TMS quando "listener".

Registo DATA OUT - É um registo de escrita e é acedido com RS2=1, RS1=1 e RS0=1. É usado por um "talker" para enviar mensagens de "interface" ou mensagens dependentes de aparelhos.

A implementação das funções de "interface" descritas pelo standard IEEE 488 são obtidas por meio de blocos de estado.

A informação é recebida do BUS IEEE e dos registos internos combinado com o estado actual do aparelho (por exemplo, "Listener Active State", LACS) para produzir o sinal de controlo a fim de carregar os registos, operar o handshake ou gerar as linhas do BUS.

C.4 - TERMINAIS DE ACESSO DO TMS 9914A

O standard IEEE 488 convenciona a utilização da lógica negativa para as linhas do GPIB. O estado falso (0) é representado por um nivil de tensão alto, superior a 2V e o estado verdadeiro (1) é representado pelo nível de tensão baixo, inferior a 0.8V. O TMS 9914A está de acordo com esta convensão e por exemplo se data valid é verdadeira (1) a linha de DAV é puxada para baixo pelo aparelho. Ao ligar-se o TMS ao microprecessador teve-se em atenção que este trabalha com lógica positiva, logo foi necessário corrigir a inversão de linhas. A figura C3 a e b mostra-nos os terminais do TMS e respectivas funções.

O TMS é ligado ao BUS IEEE socorrendo-se de dois "bufferes", os transceivers SN75160e SN75161, A sua aplicação resultou da correspondência dos seus terminais com o TMS, o que permitiu uma completa interface com o GPIB, dispondo de uma pequena área do circuito.

O SN75160 é um integrado de 20 pinos própio para ser "buffer" de linhas de dados DIO (8 - 1) em todas as aplicações. O sentido do "buffer" é controlado através da saída "talk enable" (TE) do TMS. Este sinal é activa a alto e torna-se verdadeiro sempre que há uma função de interface do TMS enviando uma mensagem no DIO, não nula (NUL), o que quer dizer se este se encontrasse nos estados TACS, CACS, SPAS ou PPAS. Através da entrada activa em alto do sinal "Pull-up Enable" (PE) do SN75160 seleccionou-se um funcionamento em "push- pull" dispondo de maior velocidade de dados.

O SN75161 é um integrado de 20 pinos usado como "buffer" das linhas de gestão. O sentido das linhas de handshake NRFD, NDAC e DAV são controladas também pelo sinal TE. Contudo os "buffers" de SRQ, ATN, REN e IFC são controlados pelo pela entrada DC do SN75161 que se encontra ligada à saída CONT ("controller active") do TMS. Esta saída, CONT, torna-se baixa quando o TMS é o controlador activo, ou seja se não está CIDS ou CADS. O SN75161 permite ainda o controlo do sentido do "buffer" de EOI, que depende do sinal TE quando ATN é falso (alto) e o sinal da DC quando ATN é verdadeiro (baixo).

As suas características encontram-se no apêndice C do Data Manual do TMS.

linha	pino	I/O	descrição
CE	3	I	CHIP ENABLE: $\overline{CE}$ low allows access of read and write registers. If $\overline{CE}$ is high, D0-D7 are in high impedance unless $\overline{ACCGR}$ is low.
$\overline{WE}$	4	I	WRITE ENABLE: when active (low), indicates to the TMS9914A that data is being written to one of its registers.
DBIN	5	I	DATA BUS IN: an active (high) state indicates to the TMS9914A that a read is about to be carried out by the MPU.
$\overline{INT}$	9	O(o/d) (no pullup)	INTERRUPT: sent to the MPU to cause a branch to a service routine.
$\overline{ACCRQ}$	1	O(p/p)	ACCESS REQUEST: this pin becomes active (low) to request a direct memory access.
$\overline{ACCGR}$	2	I	ACCESS GRANTED: when received from the direct memory access control logic this enables the byte onto the data bus. ACCGR must be high when not participating in DMA transfer.
$\overline{RESET}$*	19	I	INITIALIZES the TMS9914A at power-on.
TR	39	O(p/p)	TRIGGER: activated when the GET command is received over the interface or the fget command is given by the MPU.
$\overline{\Phi}$	18	I	CLOCK input: 500 kHz to 5 MHz. Need not be synchronous to system clock.
V_{SS}	20		Ground reference voltage.
V_{CC}	40		Supply voltage (+ 5 V nominal).

Figura C3(a): Terminais e funções do TMS

linha	pino	I/O	descrição
DIO8	31	I/O(p/p)	DIO8 through DIO1 are the
DIO7	32	I/O(p/p)	data input/output lines
DIO6	33	I/O(p/p)	on the GPIB side. These
DIO5	34	I/O(p/p)	pins connect to the IEEE-
DIO4	35	I/O(p/p)	488 bus via non-inverting
DIO3	36	I/O(p/p)	transceivers.
DIO2	37	I/O(p/p)	
DIO1	38	I/O(p/p)	
DAV	26	I/O(p/p)	DATA VALID: handshake line controlled by source to show acceptors when valid data is present to the bus.
NDAC	24	I/O(p/p)	NOT DATA ACCEPTED: handshake line. Acceptor sets this false (high) when it has latched the data from the I/O lines.
NRFD	25	I/O(p/p)	NOT READY FOR DATA: handshake line. Sent by acceptor to indicate readiness for the next byte.
ATN	28	I/O(p/p)	ATTENTION: sent by controller in charge. When true (low), interface commands are being sent over the DIO lines. When false (high), these lines carry data.
REN	22	I/O(o/d)	REMOTE ENABLE: sent by system controller to select control either from the front panel or from the IEEE bus.
IFC	23	I/O(o/d)	INTERFACE CLEAR: sent by the system controller to set the interface system into a known quiescent state. The system controller becomes the controller in charge.
SRQ	29	I/O(p/p)	SERVICE REQUEST: set true (low) by a device to indicate a need for service.
EOI	27	I/O(p/p)	END OR IDENTIFY: if ATN is false (high), this indicates the end of of a message block. If ATN is true (low), the controller is requesting a parallel poll.
$\overline{\text{CONT}}$	30	O(p/p)	Indicates if a device is controller in charge. It is used to control direction of SRQ and ATN in pass control systems. Logically, it is (CIDS + CADS).
TE	21	O(p/p)	TALK ENABLE: controls the direction of the transfer of the line transceivers. Logically, it is:(CACS + TACS + EIO.ATN.(CIDS + CADS). $\overline{\text{SWRST}}$)
D0	17	I/O(p/p)	Data transfer lines on the MPU side
D1	16	I/O(p/p)	of the device
D2	15	I/O(p/p)	
D3	14	I/O(p/p)	
D4	13	I/O(p/p)	
D5	12	I/O(p/p)	
D6	11	I/O(p/p)	
D7	10	I/O(p/p)	
RS0	6	I	REGISTER SELECT LINES: determine
RS1	7	I	which register is addressed by the
RS2	8	I	MPU during a read or write operation

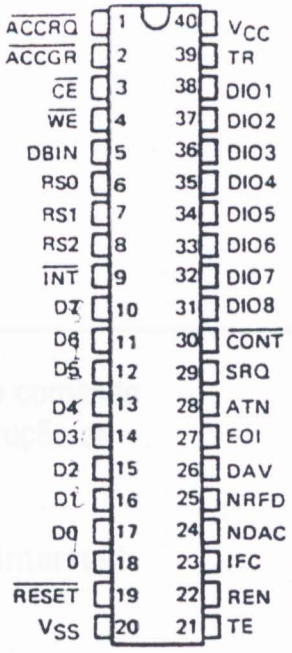


Figura C3(b): Terminais e funções do TMS

APÊNDICE D

D1: Rotina de inicialização do TMS

ENDEREÇO DO PC	CÓDIGO HEXAD.	INSTRUC. ASSEMBL	COMENTÁRIOS
06F0	3E	LDA, 80	Carrega o registo de comando
06F1	80		auxiliar com a instrução de
06F2	D3	OUT(83), A	"software reset"
06F3	83		
06F4	3E	LDA, 20	Carrega o registo "interrupt
06F5	20		MASK0"
06F6	D3	OUT(80), A	
06F7	80		
06F8	3E	LDA, 0D	Carrega o registo "interrupt
06F9	0D		MASK1"
06FA	D3	OUT(81), A	
06FB	81		
06FC	3E	LDA, 0A	Carrega o registo de endereço
06FD	0A		
06FE	D3	OUT(84), A	
06FF	84		
0700	3E	LDA, 89	Carrega o registo de comando
0701	89		auxiliar com a instrução de
0702	D3	OUT(83), A	"listener only"
0703	83		
0704	3E	LDA, 83	Carrega o registo de comando
0705	83		auxiliar com a instrução
0706	D3	OUT(83), A	"Holdoff All Data"
0707	83		
0708	3E	LDA, 00	Carrega o registo de comando
0709	00		auxiliar com a instrução de
070A	D3	OUT(83), A	"clear software reset"
070B	83		

D2: Rotina de interrupção do Z80

ENDEREÇO DO PC	CÓDIGO HEXAD.	INSTRUC. ASSEMBL	COMENTÁRIOS
070C	3E	LDA, 24	Definição do byte de endereço
070D	24		mais significatido
070E	ED	LDI, A	
070F	47		
0710	ED	IM2	Definição do modo de
0711	5E		interrupção 2
0712	FB	EI	Activa modo de interrupção
0713	C9		

D3: Rotina de transferência de dados

ENDEREÇO DO PC	CÓDIGO HEXAD.	INSTRUC. ASSEMBL	COMENTÁRIOS
24FF	01		Definição do enderço inicial da
2500	25		rotina de interrupção de leitura
2501	FD	LDIY, 1800	Carrega o registo indexado IY
2502	21		com 1800
2502	00		
2503	18		
2505	DB	INA, (80)	Lê os dados do registo
2506	80		"interrupt status0"
2507	47	LDB, A	Carrega o registo B com os dados
2508	CB	BIT 5, A	O bit 5 do acumulador é posto na
2509	6F		"flag" Z
250A	CA	JZ 2514	Se BI (bit 5) = 0 salta para a
250B	14		posição 2514
250C	25		
250D	DB	INA, (87)	Se BI=1 faz a leitura do registo
250E	87		de dados
250F	FD	LDIY, A	Carrega os dados a partir de 1800
2510	77		
2511	00		
2512	FD	INC IY	Aponta a posição de memória do
2513	23		dado que se segue
2514	3E	LDA, 02	
2515	02		
2516	D3	OUTA, (83)	Liberta a prisão do RFD
2517	83		
2518	CB	BIT 3, B	Põe o bit 3 do registo "interrupt
2519	58		status" na "flag" Z
251A	CA	JZ 2505	Se o bit 3(end) é zero salta para
251B	05		a posição 2505 para continuar a
251C	25		lêr
251D	FB	EI	
251E	C3	JP 20A0	Se o bit 3 = 1, todos os dados
251F	A0		foram lidos, activa "interupt" e
2520	20		salta para a posição 20A0

D.4 - ROTINA GERADORA DE FORMAS DE ONDA

ENDEREÇO DO µP	LING. MAQ. HEXAD.	PERIODOS DE RELÓG.	LABEL	LING. ASSEMBLY	COMENTÁRIOS
20A0	3A			LDA, 1800	Põe a atenuação no DAC2
20A1	00				
20A2	18				
20A3	32			LD 1C02, A	
20A4	02				
20A5	1C				
20A6	3A			LDA. 1801	Põe filtro e factor de escala
20A7	01				
20A8	18				
20A9	32			LD 1C03, A	
20AA	03				
20AB	1C				

ENDEREÇO DO µP	LING. MAQ. HEXAD.	PERIODOS DE RELÓG.	LABEL	LING. ASSEMBLY	COMENTÁRIOS
20AC	DD	20	LOOP	LD IX,(1805)	Escolha da onda
20AD	2A				
20AE	05				
20AF	18				
20B0	3A	13		LDA, (1804)	Fixação do nº de amostras
20B1	04				
20B2	18				
20B3	47	4			
20B4	DD	19	LOOP1	LDA, (IX)	Litura do byte menos significativo
20B5	7E				
20B6	00				
20B7	32	13		LD (1C00), A	Carrega o byte no 1º resisto do DAC
20B8	00				
20B9	1C				
20BA	DD	19		INC IX	Aponta o proximo endereço
20BB	23				
20BC	DD	19		LDA, (IX)	Leitura do byte mais significativo
20BD	7E				
20BE	00				
20BF	32	13		LD (1C00), A	Carrega o byte mais significativo no 2º resisto do DAC
20C0	01				
20C1	1C				
20C2	3A	13		LDA, (1802)	Atraso no registo D
20C3	02				
20C4	18				

ENDEREÇO DO µP	LING. MAQ. HEXAD.	PERIODOS DE RELÓG.	LABEL	LING. ASSEMBLY	COMENTÁRIOS
20C5	57	4		LDD, A	Vê se tem atraso
20C6	AF	4		XOR A	
20C7	BA	4		CPD	
20C8	28	12/7		JRZ STAR	Se não tem atraso toma proxima amostra
20C9	0A				
20CA	3A	13	ATRASO	LDA, (1803)	Anel de atraso do registo C
20CB	03				
20CC	18				
20CD	4F	4		LDC, A	
20CE	0D		LOOP2	DEC C	
20CF	20	12/7		JRNZ LOOP2	
20D0	FD				
20D1	15	4		DEC D	
20D2	20	12/7		JRNZ ATRASO	Se Houve atraso e já chegou fim, toma nova amostra
20D3	FD				
20D4	DD	10	STAR	INC IX	Verifica se já enviou a ultima amostra, se sim, então inicia novo ciclo
20D5	23				
20D6	10	8/13		DJNZ COR	
20D7	03				
20D8	C3	10		JUMP LOOP	
20D9	A0				
20DA	20				registo do DAC
20DB	00	4	COR	NOP	Ciclo de compensação de tempo por amostra
20DC	00				
20DD	00				

ENDEREÇO DO µP	LING. MAQ. HEXAD.	PERIODOS DE RELÓG.	LABEL	LING. ASSEMBLY	COMENTÁRIOS
20DD	00				
20DE	00				
20DF	00				
20E0	00				
20E1	00				
20E2	C3	10		JUMP LOOP1	Nova amostra
20E3	A8				
20E4	20				

.D5:

```
program GeradorFuncoes;

uses Graph,crt;

const TituloTab ='Tempo    V=f(t)';
      DigitosTab='123456  123456';
      NlinhasSetup=9;
      DefLargTick=5;
      Ndigitos=6;
      TInsert=82;
      TDelete=83;
      TEnd=79;
      THome=71;
      TPgUp=73;
      TPgDown=81;
      TUp=72;
      TDown=80;
      TLeft=75;
      TRight=77;
      TEsc=27;
      TReturn=13;
      Sair=Tesc;

Type TipoStr=string;
     PtLista=^ElmtLista;
     ElmtLista=record
         Tempo,Tensao:real;
         Prox:PtLista;
         end;

var graphdriver :integer;
    graphmode   :integer;
    errorcode   :integer;
    Xoffset,Yoffset:integer;
    HMsg:Integer;
    Xmax,Ymax:Integer;
    Vmax,Vmin:real;
    Tmax,Tmin:real;
    Ttick,VTick:real;
    Ysetup:Integer;
    NmaxTab:Integer;
    xtempo,xtensao,ytempo,ytensao:integer;
    Lista,Pedit,Pinit:PtLista;
    ficheiro:text;
    grelha,eixos,picos:boolean;
    Ndec:Integer;
    TeclaEsp:Integer;
    Kpos:Integer;
    Straux:string;

procedure apagastr(X,Y,N,cor:Integer);
var Vport:ViewPortType;
    str:string;
begin
setfillstyle(solidfill,cor);
```

```
bar(X,Y,X+(N*TextWidth('1')),Y+TextHeight('1'));
setfillstyle(solidfill,white);
end;

procedure mensagem(str:tipostr);
begin
outtextxy(1,Ymax+2,str);
end;

procedure clearmensagem;
begin
Apagastr(1,Ymax+2,Getmaxx div (TextWidth('1')),black);
end;

procedure esperatecla;
begin
while not keypressed do;
end;


procedure som;
var k:integer;
begin
sound(1000);
for k:=1 to 10000 do;
Nosound;
end;

Function LeStrxy(X,Y,Nchar:Integer;var valido:boolean):string;
var str:string;
    continua:boolean;
    ch:char;
    Vch:Integer;
    k:integer;
begin
str:='';
for k:=1 to Nchar do str:=concat(str,' ');
apagastr(X,Y,Nchar,white);
setcolor(black);
outtextxy(x,y,str);
str:='';
continua:=true;
valido:=true;
while (continua) and (length(str)< NChar) do
  begin
  OutTextxy(x,y,str);
  ch:=readkey;
  Vch:=Ord(Ch);
  if Vch=27 then begin
                 continua:=false;
                 valido:=false;
                 end
```

```
  else
  if Vch=13 then continua:=false
     else
     if Vch=8 then
        begin
        apagastr(X,Y,Length(str),white);
        str:=copy(str,1,Length(str)-1);
        end
        else
        if (Vch>=32) and (Vch<=126) then str:=concat(str,ch)
           else
           begin
           som;
           while keypressed do
             ch:=readkey;
           end;
  end;
apagastr(X,Y,Nchar,black);
setcolor(white);
outtextxy(x,y,str);
Lestrxy:=str;
while keypressed do
  ch:=readkey;
end;

Function LeIntxy(X,Y,Nchar:Integer;var valido:boolean):Integer;
var str:string;
    continua:boolean;
    ch:char;
    Vch:Integer;
    k,auxint:integer;

begin
str:='';
for k:=1 to Nchar do str:=concat(str,' ');
apagastr(X,Y,Nchar,white);
setcolor(black);
outtextxy(x,y,str);
str:='';
continua:=true;
valido:=true;
while (continua) and (length(str)< NChar) do
  begin
  OutTextxy(x,y,str);
  ch:=readkey;
  Vch:=Ord(Ch);
  if vch=27 then begin
                 continua:=false;
                 valido:=false;
                 end
  else
  if Vch=13 then continua:=false
     else
```

```
    if Vch=8 then
       begin
       apagastr(X,Y,Length(str),white);
       str:=copy(str,1,Length(str)-1);
       end
       else
       if (Vch>=48) and (Vch<=57) then str:=concat(str,ch)
          else
          begin
          som;
          while keypressed do
            ch:=readkey;
          end;
  end;
apagastr(X,Y,Nchar,black);
setcolor(white);
Outtextxy(x,y,str);
if length(str)=0 then str:='0';
val(str,auxint,k);
leintxy:=auxint;
while keypressed do
  ch:=readkey;
end;

Function LeRealxy(X,Y,Nchar:Integer;var valido:boolean):Real;
var str:string;
    continua:boolean;
    ch:char;
    Vch:Integer;
    k:integer;
    auxreal:real;
    ponto:boolean;
begin
ponto:=false;
auxreal:=0;
str:='';
for k:=1 to Nchar do str:=concat(str,' ');
apagastr(X,Y,Nchar,white);
setcolor(black);
outtextxy(x,y,str);
str:='';
continua:=true;
valido:=true;
while (continua) and (length(str)< NChar) do
  begin
  OutTextxy(x,y,str);
  ch:=readkey;
  Vch:=Ord(Ch);
  if vch=27 then begin
                 continua:=false;
                 valido:=false;
                 end
```

```
  else
  if Vch=13 then continua:=false
     else
     if Vch=8 then
        begin
        if str[length(str)]='.' then ponto:=false;
        apagastr(X,Y,Length(str),white);
        str:=copy(str,1,Length(str)-1);
        end
        else
        if vch In [48,49,50,51,52,53,54,55,56,57] then str:=concat(str,ch)
          else
          if (vch=46)and(not ponto) then
             begin
             str:=concat(str,ch);
             ponto:=true;
             end
          else
          begin
          som;
          while keypressed do
            ch:=readkey;
          end;
  end;
apagastr(X,Y,Nchar,black);
setcolor(white);
Outtextxy(x,y,str);
if length(str)=0 then str:='0';
auxreal:=0;
if valido then
   begin
   val(str,auxreal,k);
   valido:=(k=0);
   end;
LeRealxy:=AuxReal;
while keypressed do
  ch:=readkey;
end;

Function ConvRstr(R:real; Ndec:Integer):string;
var Pint,Pdec:Longint;
    str:string[15];
    digito,j:Integer;
    Pot10:Longint;
    Negativo:Boolean;
begin
 Pot10:=1;
 for j:=1 to Ndec do Pot10:=Pot10*10;
 str:='';
 if R<0 then begin
             R:=R*(-1);
             Negativo:=true;
```

```
            end
            else Negativo:=False;
R:=Round(R*(Pot10))/Pot10;
Pint:=trunc(R);
Pdec:=round((R-Pint)*Pot10);
Pdec:=Pot10+Pdec;

repeat
  digito:=Pdec mod 10;
  Pdec:=Pdec div 10;
  str:=concat(chr(digito+48),str);
until (Pdec=1);

str:=concat('.',str);
repeat
  digito:=PInt mod 10;
  PInt:=PInt div 10;
  str:=concat(chr(digito+48),str);
until (PInt=0);
if Negativo then str:=concat('-',str)
            else str:=concat('+',str);
ConvRstr:=str;
end;

function readreal(x,y:integer;var r:real):integer;
var str:string[15];
    feito,ok,continua,processa:boolean;
    sit,k,limitint:integer;
    pot10:longint;
    ch,chaux:char;
begin

str:=ConvRstr(r,Ndec);
{muda cor}
setcolor(black);
apagastr(x-(textwidth('1')*Ndigitos),y,Ndigitos,white);
for k:=length(str) downto 1 do
    outtextxy(x-(textwidth('1')*(length(str)-k+1)),y,str[k]);
continua:=true;
sit:=3; {0-vazio 1-Com sinal 2-parte inteira 3-Parte decimal}
while continua do
 begin
 if length(str)>=Ndigitos then
    begin
    ch:=readkey;
    if ord(ch)=0 then ch:=readkey;
    while not (ord(ch) in [Treturn,Tesc,8,TInsert,TDelete,TEnd,THome,TUp,TDow
       begin
       som;
       ch:=readkey;
       if ord(ch)=0 then ch:=readkey;
       end;
    if ord(ch) in [TInsert,TDelete,TEnd,THome,TUp,TDown,TRight,TLeft] then
```

```
      begin
      readreal:=ord(ch);
      continua:=false;
      end;
    end
    else
    ch:=readkey;
if continua then
 begin
 ok:=true;
 feito:=false;
 case ord(ch) of
 TEsc:begin
     feito:=true;
     readreal:=TEsc;
     continua:=false;
     end;
   0:begin
     feito:=true;
     ch:=readkey;
     if ord(ch) in [TInsert,TDelete,TEnd,THome,TUp,TDown,TRight,TLeft] then
                begin
                readreal:=ord(ch);
                continua:=false;
                end
                else som;
     end;

   8:if length(str)>0 then
      begin
      if str[length(str)] in ['+','-'] then sit:=0
         else if str[length(str)]='.' then sit:=2;
      apagastr(x-(textwidth('1')*Ndigitos),y,Ndigitos,white);
      delete(str,length(str),1);
      feito:=true;
      end;
  13:begin
     continua:=false;
     if keypressed then ch:=readkey;
     readreal:=13;
     feito:=true;
     end;
  end;
  if not feito then
    case sit of
      0:if ch in ['+','-'] then
            begin
            str:=ch;
            sit:=1;
            end
            else
            if ch in ['0','1','2','3','4','5','6','7','8','9'] then
               begin
```

```
            str:=concat('+',ch);
            sit:=2;
            end
            else
            if ch='.' then
               begin
               str:='+0.';
               sit:=3;
               end
               else ok:=false;
    1:if ch='.' then
        begin
        if length(str)=1 then str:=concat(str,'0',ch)
                         else str:=concat(str,ch);
        sit:=3;
        end
        else
        if ch in ['0','1','2','3','4','5','6','7','8','9'] then
           str:=concat(str,ch)
           else ok:=false;

     2:if ch='.' then
         begin
         str:=concat(str,ch);
         sit:=3;
         end
         else
         if ch in ['0','1','2','3','4','5','6','7','8','9'] then
            str:=concat(str,ch)
            else ok:=false;
     3:if ch in ['0','1','2','3','4','5','6','7','8','9'] then
         str:=concat(str,ch)
         else ok:=false;
   end; { end case}
  if not ok then som
      else
      begin
      apagastr(x-(textwidth('1')*Ndigitos),y,Ndigitos,white);
      for k:=length(str) downto 1 do
            outtextxy(x-(textwidth('1')*(length(str)-k+1)),y,str[k]);
      end;
  end;
end; {while continua}

{conversao para real}
r:=0.0;
if pos('.',str)=0 then limitint:=length(str)
                  else limitint:=pos('.',str)-1;
pot10:=1;
for k:=limitint downto 2 do
   begin
   r:=r+(ord(str[k])-48)*pot10;
   pot10:=pot10*10;
```

```
   end;
Pot10:=10;
for k:=LimitInt+2 to length(str) do
   begin
   r:=r+(ord(str[k])-48)/pot10;
   pot10:=pot10*10;
   end;
if str[1]='-' then r:=r*(-1);
str:=convRstr(r,Ndec);
apagastr(x-(textwidth('1')*Ndigitos),y,Ndigitos,black);
setcolor(white);
for k:=length(str) downto 1 do
   outtextxy(x-(textwidth('1')*(length(str)-k+1)),y,str[k]);
end;


Function ConvertV(V:real):integer;
begin
ConvertV:=round((Ymax*(V-Vmax))/(Vmin-Vmax));
end;

Function ConvertT(T:real):integer;
begin
ConvertT:=round(((Xmax-Xoffset)*(T-Tmin))/(Tmax-Tmin));
end;

procedure layout;
var Vinit,Tinit:real;
    LargTickT,LargTickV:Integer;
    st:string;
begin
rectangle(0,0,Xoffset-1,Ysetup);
rectangle(0,Ysetup,Xoffset-1,Ymax+1);
rectangle(0,Ymax+1,Xmax,getmaxy);
str(Vmax:7:3,st);
OutTextxy(Xoffset+10,10,st);
str(VMin:7:3,st);
OutTextxy(Xoffset+10,Ymax-textheight('1')-10,st);
Outtextxy(Xoffset+10,5+textheight('1')+5,'V(t)');
Outtextxy(Xmax-20,Ymax-TextHeight('1')-5,'t');
setcolor(green);
rectangle(Xoffset,0,Xmax,Ymax);
setviewport(Xoffset,0,Xmax,Ymax,True);
if eixos then
        begin
        {eixo T}
        Moveto(ConvertT(Tmin),ConvertV(0));
        Lineto(ConvertT(Tmax),ConvertV(0));
        {eixo V}
        Moveto(ConvertT(0),ConvertV(Vmax));
        Lineto(ConvertT(0),ConvertV(Vmin));
```

```
        end;
{Tickness}
If grelha then begin
               LargTickV:=Xmax;
               LargTickT:=Ymax;
               end
          else begin
               LargTickV:=DefLargTick;
               LargTickT:=DefLargTick;
               end;
{eixo V}
Vinit:=0;
While VInit<Vmax do
      begin
      Line(ConvertT(Tmin),ConvertV(Vinit),ConvertT(Tmin)+LargTickV,ConvertV(Vinit
      Vinit:=Vinit+VTick;
      end;
Vinit:=(-1)*VTick;
While VInit>Vmin do
      begin
      Line(ConvertT(Tmin),ConvertV(Vinit),ConvertT(Tmin)+LargTickV,ConvertV(Vinit
      Vinit:=Vinit-VTick;
      end;
{eixo T}
Tinit:=0;
if Tmin>0 then
   while Tinit<Tmin do Tinit:=Tinit+Ttick;
While TInit<Tmax do
      begin
      Line(ConvertT(Tinit),ConvertV(Vmin)-LargTickT,ConvertT(Tinit),ConvertV(Vmir
      if eixos then
         begin
         Line(ConvertT(Tinit),ConvertV(0)-LargTickT,ConvertT(Tinit),ConvertV(0));
         str(Tinit:4:1,st);
         outtextxy(ConvertT(Tinit)+2,ConvertV(Vmin)-2-textheight('1'),st);
         end;
      Tinit:=Tinit+TTick;
      end;
Tinit:=(-1)*TTick;
While TInit>Tmin do
      begin
      Line(ConvertT(Tinit),ConvertV(Vmin)-LargTickT,ConvertT(Tinit),ConvertV(Vmir
      if eixos then
         begin
         Line(ConvertT(Tinit),ConvertV(0)-LargTickT,ConvertT(Tinit),ConvertV(0));
         str(Tinit:4:1,st);
         outtextxy(ConvertT(Tinit)+2,ConvertV(Vmin)-2-textheight('1'),st);
         end;
      Tinit:=Tinit-TTick;
      end;
setcolor(white);
setviewport(0,0,getmaxx,getmaxy,false);
```

```
outtextxy(1,1,TituloTab);
end;

procedure mostrasetup;
var y:integer;
    Xsimnao:Integer;
    Xvalor:Integer;
    str:string[15];
begin
setviewport(1+1,ysetup+1,xoffset-2,ymax-1,false);
clearviewport;
setviewport(0,0,getmaxx,getmaxy,true);
y:=ysetup+2;
Xsimnao:=1+textwidth('12345678');
Xvalor:=1+textwidth('12345678');
outtextxy(1,y,'Ttick:');
str:=convRstr(Ttick,Ndec);
outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
y:=y+textheight('1')+3;
outtextxy(1,y,'Tmin:');
str:=convRStr(Tmin,Ndec);
outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
y:=y+textheight('1')+3;
outtextxy(1,y,'Tmax:');
str:=convRStr(Tmax,Ndec);
outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
y:=y+textheight('1')+3;
outtextxy(1,y,'Vtick:');
str:=convRStr(Vtick,Ndec);
outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
y:=y+textheight('1')+3;
outtextxy(1,y,'Vmin:');
str:=convRStr(Vmin,Ndec);
outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
y:=y+textheight('1')+3;
outtextxy(1,y,'Vmax:');
str:=convRStr(Vmax,Ndec);
outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
y:=y+textheight('1')+3;
outtextxy(1,y,'Eixos:');
if Eixos then outtextxy(Xsimnao,y,'Sim')
         else outtextxy(Xsimnao,y,'Nao');
y:=y+textheight('1')+3;
outtextxy(1,y,'Grelha:');
if Grelha then outtextxy(Xsimnao,y,'Sim')
          else outtextxy(Xsimnao,y,'Nao');
y:=y+textheight('1')+3;
outtextxy(1,y,'Picos:');
if Picos then outtextxy(Xsimnao,y,'Sim')
          else outtextxy(Xsimnao,y,'Nao');
end;
```

```
procedure setup;
{ define valores a visualizar}
var Xsimnao,Xvalor,Tecla,PosSetup,y:integer;
    r:reaL; saida:boolean;
    str:string[15];
begin
y:=ysetup+2;
Xsimnao:=1+textwidth('12345678');
Xvalor:=1+textwidth('12345678');
tecla:=0;
PosSetup:=1;
while tecla<>Tesc do
  begin
  clearmensagem;
  mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
  y:=Ysetup+2+((PosSetup-1)*(textheight('1')+3));
  Case PosSetup of
   1:begin
      repeat
      mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
      saida:=false;
      r:=Ttick;
      Tecla:=0;
      while not (tecla in [Tesc,Tup,Tdown,Treturn]) do
         Tecla:=readreal(Xvalor+(textwidth('1')*Ndigitos),y,r);
      if r<0 then begin
                     clearmensagem;
                     mensagem('Erro: Valor de Ttick inferior ou igual a 0.0 <SP
                     som;
                     esperatecla;
                     clearmensagem;
                     end
                     else
                     begin
                      saida:=true;
                      Ttick:=r;
                     end
      until saida or (tecla in [Tesc,Tup,TDown,Treturn]);
      str:=convRstr(Ttick,Ndec);
      apagastr(XValor,y,Ndigitos,black);
      outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
     end;
   2:begin
      repeat
      mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
      saida:=false;
      r:=Tmin;
      Tecla:=0;
      while not (tecla in [Tesc,Tup,Tdown,Treturn]) do
         Tecla:=readreal(Xvalor+(textwidth('1')*Ndigitos),y,r);
      if r>=Tmax then begin
                     clearmensagem;
                     mensagem('Erro: Valor de Tmin superior ou igual a Tmax <SP
```

```
                som;
                esperatecla;
                clearmensagem;
                end
                else
                begin
                 saida:=true;
                 Tmin:=r;
                end
  until saida or (tecla in [Tesc,Tup,TDown,Treturn]);
  str:=convRstr(Tmin,Ndec);
  apagastr(XValor,y,Ndigitos,black);
  outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
 end;
3:begin
  repeat
  mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
  saida:=false;
  r:=Tmax;
  Tecla:=0;
  while not (tecla in [Tesc,Tup,Tdown,Treturn]) do
     Tecla:=readreal(Xvalor+(textwidth('1')*Ndigitos),y,r);
  if r<=Tmin then begin
                clearmensagem;
                mensagem('Erro: Valor de Tmax inferior ou igual a Tmin <SP
                som;
                esperatecla;
                clearmensagem;
                end
                else
                begin
                 saida:=true;
                 Tmax:=r;
                end
  until saida or (tecla in [Tesc,Tup,TDown,Treturn]);
  str:=convRstr(Tmax,Ndec);
  apagastr(XValor,y,Ndigitos,black);
  outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
 end;
4:begin
  repeat
  mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
  saida:=false;
  r:=Vtick;
  Tecla:=0;
  while not (ord(tecla) in [Tesc,Tup,Tdown,Treturn]) do
     Tecla:=readreal(Xvalor+(textwidth('1')*Ndigitos),y,r);
  if r<0 then begin
                clearmensagem;
                mensagem('Erro: Valor de Vtick inferior ou igual a 0.0 <SP
                som;
                esperatecla;
```

```
                clearmensagem;
                end
                else
                begin
                 saida:=true;
                 Vtick:=r;
                end
  until saida or (tecla in [Tesc,Tup,TDown,Treturn]);
  str:=convRstr(Vtick,Ndec);
  apagastr(XValor,y,Ndigitos,black);
  outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
 end;
5:begin
  repeat
  mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
  saida:=false;
  r:=Vmin;
  Tecla:=0;
  while not (tecla in [Tesc,Tup,Tdown,Treturn]) do
     Tecla:=readreal(Xvalor+(textwidth('1')*Ndigitos),y,r);
  if r>=Vmax then begin
                clearmensagem;
                mensagem('Erro: Valor de Vmin superior ou igual a Vmax <SP
                som;
                esperatecla;
                clearmensagem;
                end
                else
                begin
                 saida:=true;
                 Vmin:=r;
                end
  until saida or (tecla in [Tesc,Tup,TDown,Treturn]);
  str:=convRstr(Vmin,Ndec);
  apagastr(XValor,y,Ndigitos,black);
  outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
 end;
6:begin
  repeat
  mensagem(concat('Setup: ',chr(24),' ',chr(25),' ',' <Esc> para abandonar'));
  saida:=false;
  r:=Vmax;
  Tecla:=0;
  while not (tecla in [Tesc,Tup,Tdown,Treturn]) do
     Tecla:=readreal(Xvalor+(textwidth('1')*Ndigitos),y,r);
  if r<=Vmin then begin
                clearmensagem;
                mensagem('Erro: Valor de Vmax inferior ou igual a Vmin <SP
                som;
                esperatecla;
                clearmensagem;
                end
                else
```

```
                    begin
                     saida:=true;
                     Vmax:=r;
                    end
 until saida or (tecla in [Tesc,Tup,TDown,Treturn]);
 str:=convRstr(Vmax,Ndec);
 apagastr(XValor,y,Ndigitos,black);
 outtextxy(XValor+(textwidth('1')*(Ndigitos-length(str))),y,str);
end;
7:begin
  clearmensagem;
  mensagem(concat('Setup: ','<SPACE BAR> ',chr(24),' ',chr(25),' ',' <Esc> p
  repeat
  apagastr(Xsimnao,y,3,white);
  setcolor(black);
  if Eixos then outtextxy(Xsimnao,y,'Sim')
           else outtextxy(Xsimnao,y,'Nao');
  Tecla:=Ord(readkey);
  case tecla of
    32:Eixos:=not(Eixos);
     0:Tecla:=Ord(readkey);
  end;
  until tecla in [Treturn,Tup,Tesc,TDown];
  apagastr(Xsimnao,y,3,black);
  setcolor(white);
  if Eixos then outtextxy(Xsimnao,y,'Sim')
           else outtextxy(Xsimnao,y,'Nao');
  end;
8:begin
  clearmensagem;
  mensagem(concat('Setup: ','<SPACE BAR> ',chr(24),' ',chr(25),' ',' <Esc>
  repeat
  apagastr(Xsimnao,y,3,white);
  setcolor(black);
  if Grelha then outtextxy(Xsimnao,y,'Sim')
            else outtextxy(Xsimnao,y,'Nao');
  Tecla:=Ord(readkey);
  case tecla of
    32:Grelha:=not(Grelha);
     0:Tecla:=Ord(readkey);
  end;
  until tecla in [Treturn,Tup,Tesc,TDown];
  apagastr(Xsimnao,y,3,black);
  setcolor(white);
  if Grelha then outtextxy(Xsimnao,y,'Sim')
            else outtextxy(Xsimnao,y,'Nao');
  end;
9:begin
  clearmensagem;
  mensagem(concat('Setup: ','<SPACE BAR> ',chr(24),' ',chr(25),' <Esc> para
  repeat
  apagastr(Xsimnao,y,3,white);
  setcolor(black);
```

```
      if Picos  then outtextxy(Xsimnao,y,'Sim')
                else outtextxy(Xsimnao,y,'Nao');
      Tecla:=Ord(readkey);
      case tecla of
        32:Picos:=not(Picos);
         0:Tecla:=Ord(readkey);
      end;
      until tecla in [Treturn,Tup,Tesc,TDown];
      apagastr(Xsimnao,y,3,black);
      setcolor(white);
      if Picos  then outtextxy(Xsimnao,y,'Sim')
                else outtextxy(Xsimnao,y,'Nao');
      end;
  end;
  case tecla of
    Tup:if Possetup>1 then PosSetup:=PosSetup-1
                      else PosSetup:=9;
    TDown,Treturn:if PosSetup<9 then PosSetup:=PosSetup+1
                      else PosSetup:=1;
  end;
 end;
end;


procedure VisuFuncao(p:PtLista);

Var Desloc,X,Y:integer;
     q:Ptlista;
    Tant:real;
    Correcto:boolean;
begin
setviewport(Xoffset,0,Xmax,Ymax,True);
q:=p;
if q<>nil then
   begin
   Y:=ConvertV(q^.tensao);
   X:=ConvertT(q^.tempo);
   Tant:=q^.tempo;
   Moveto(X,Y);
   If q^.tensao>=0 then Desloc:=-8
                   else Desloc:=+8;
   If picos then
      if (X>=0)and(X<=Xmax) then
         if (Y>=0) and ((Y+Desloc)<=Ymax) then
                Outtextxy(X,Y+Desloc,ConvRstr(q^.tensao,Ndec));

   Moveto(X,Y);
   q:=q^.prox;
   correcto:=true;
   while (q<>nil) and correcto do
      begin
```

```
     if q^.tempo>=Tant then
        begin
        Tant:=q^.tempo;
        X:=ConvertT(q^.tempo);
        Y:=ConvertV(q^.tensao);
        Lineto(X,Y);
        If q^.tensao>=0 then Desloc:=-8
                        else Desloc:=+8;
        if picos then
          if (X>=0)and(X<=Xmax) then
             if (Y>=0) and ((Y+Desloc)<=Ymax) then
                    Outtextxy(X,Y+Desloc,ConvRstr(q^.tensao,Ndec));
        Moveto(X,Y);
        q:=q^.prox;
        end
        else
        correcto:=false;
      end;
   end;
setviewport(0,0,getmaxx,getmaxy,false);
if not correcto then begin
                     clearmensagem;
                     mensagem('Erro: Tabela de valores de t nao crescente <QQ> pa
                     esperatecla;
                     esperatecla;
                     clearmensagem;
                     end;
end;


procedure escreveficheiro;
var ficheiro:text;
    Nomefich:string;
    valido:boolean;
    q:PtLista;

begin
   clearmensagem;
   mensagem('Guarda no ficheiro:');
   nomefich:=LeStrxy(1+TextWidth('Guarda no ficheiro:'),Ymax+2,11,valido);
   clearmensagem;
   if (length(nomefich)>0)and(valido) then
    begin
    assign(ficheiro,nomefich);
    {$I-}
    rewrite(ficheiro);
    {$I+}
    if IOresult=0 then
```

```
      begin
      q:=Lista;
      while q<>nil do
        begin
        writeln(ficheiro,q^.tempo:10:3,' ',q^.tensao:10:3);
        q:=q^.prox;
        end;
      close(ficheiro);
      end
      else
      begin
      mensagem('Erro na escrita em ficheiro <QQ> para continuar');
      esperatecla;
      clearmensagem;
      end;
    end;
end;

procedure leficheiro;
var ficheiro:text;
    Nomefich:string;
    erro:Integer;
    q,t,p:PtLista;
    Tempo,Tensao:real;
    valido:boolean;
    maxv,maxt,mint:real;
begin
   clearmensagem;
   mensagem('Leitura do ficheiro:');
   nomefich:=LeStrxy(1+TextWidth('Leitura do ficheiro:'),Ymax+2,11,valido);
   clearmensagem;
   if (length(nomefich)>0)and(valido) then
   begin
   t:=Nil;
   erro:=0;
   assign(ficheiro,nomefich);
   {$I-}
   reset(ficheiro);
   {$I+}
   if IOresult<>0 then erro:=1;
   if erro=0 then
      begin
      {$I-}
      readln(ficheiro,tempo,tensao);
      maxv:=abs(tensao);
      minT:=tempo;
      maxT:=tempo;
      {$I+}
      if IOresult<>0 then erro:=2;
      if erro=0 then
         begin
         New(t);
         t^.prox:=nil;t^.tempo:=tempo;t^.Tensao:=Tensao;
```



```
      q:=t;
      while (not eof(ficheiro)) and (erro=0) do
         begin
         {$I-}
         readln(ficheiro,tempo,tensao);
         {$I+}
         if abs(tensao)>maxv then maxv:=abs(tensao);
         maxT:=tempo;
         if IOresult<>0 then erro:=2;
         if erro=0 then
            begin
            new(q^.prox);
            q:=q^.prox;q^.tempo:=tempo;q^.tensao:=tensao;
            q^.prox:=nil;
            end;
         end;
      end;
   end;
if erro<>1 then close(ficheiro);
if erro=0 then begin
                 while lista<>nil do
                   begin
                   p:=Lista;
                   Lista:=Lista^.prox;
                   dispose(p);
                   end;
                 Lista:=t;
                 Pinit:=t;
                 PEdit:=t;
                 Kpos:=0;
                 Vmax:=trunc(maxv)+Vtick;
                 Vmin:=(-1)*Vmax;
                 Tmax:=maxt;
                 Tmin:=mint;
                 end
          else
          begin
          while t<>nil do
            begin
            q:=t; t:=t^.prox; dispose(q);
            end;
          case erro of
           1:mensagem('Erro na abertura do ficheiro  <QQ> para continuar');
           2:mensagem('Erro na leitura dos dados     <QQ> para continuar');
           end;
          esperatecla;
          clearmensagem;
          end;
  end;

end;

Function LeTeclaEsp:Integer;
```

```
var ch:Char;
    ok:boolean;
begin
  ok:=false;
  while not ok do
    begin
    Ch:=readkey;
    if  (Ord(ch)=0) and (Keypressed) then
                begin
                Ch:=ReadKey;
                Ok:=True;
                end
                else
                if (ord(ch)=27) then ok:=true
                   else som;
    end;
 LeTeclaEsp:=Ord(Ch);
end;

procedure InitGraficos;
var k,x,y,j,h:integer;
    ch:char;
begin
  graphdriver:=detect;
  detectgraph(graphdriver,graphmode);
  initgraph(graphdriver,graphmode,'');
  errorcode:=graphresult;
  if errorcode <> grOk then
  begin
    writeln('Erro: ',grapherrormsg(errorcode));
    writeln('Programa interrompido ...');
    readln;
    Halt(1);
  end
  else
  begin
  settextstyle(TriplexFont,HorizDir,3);
  setfillstyle(solidfill,white);
  outtextxy((getmaxx-textwidth('GERADOR DE FUNCOES')) div 2,50,'GERADOR DE FUNCO
  settextstyle(triplexfont,horizdir,2);
  outtextxy((getmaxx-textwidth('REALIZADO POR:')) div 2,75,'REALIZADO POR:');
  outtextxy((getmaxx-textwidth('FERNANDO MAMEDE')) div 2,100,'FERNANDO MAMEDE');
  outtextxy((getmaxx-textwidth('I.P.V. - E.S.T.')) div 2,125,'I.P.V. - E.S.T.');
  outtextxy((getmaxx-textwidth('SUPERVISIONADO POR:')) div 2,150,'SUPERVISIONADO
  outtextxy((getmaxx-textwidth('PROF. DR. ARTUR CAPELO CARDOSO')) div 2,
  175,'PROF. DR. ARTUR CAPELO CARDOSO');
  outtextxy((getmaxx-textwidth('F.E.U.P.')) div 2,200,'F.E.U.P.');
  Settextstyle(GothicFont,HorizDir,2);
  x:=textwidth('Escola Superior de Tecnologia');
  rectangle(getmaxx div 2,getmaxy-10-10-(2*textheight('E')),(getmaxx div 2)+x,ge
  Outtextxy(getmaxx div 2,getmaxy-10-10-(2*textheight('E')),'Instituto Politecni
  Outtextxy(getmaxx div 2,getmaxy-10-textheight('E'),'Escola Superior de Tecnol
  settextstyle(Defaultfont,HorizDir,1);
  settextjustify(lefttext,Toptext);
```

```
  setfillstyle(solidfill,white);
  outtextxy(0,getmaxy-20,'Premir uma tecla para continuar');
  ch:=readkey;
  if ((ord(ch)=0) and keypressed) then ch:=readkey;
  clearviewport;
  HMsg:=textHeight('1')+3;
  Ymax:=getmaxy-HMsg-1;
  Xmax:=getmaxx;
  Xoffset:=textWidth(DigitosTab)+3;
  Yoffset:= Ymax div 2;
  Xtempo:=1+(textwidth('1')*Ndigitos);
  Xtensao:=1+((textwidth('1')*(2*Ndigitos+2)));
  Ytempo:=1+2+textheight('1');
  Ytensao:=1+2+textheight('1');
  end;
end;

procedure VisuTabela;
var p:PtLista;
    k,y,x:Integer;
    str:string[Ndigitos];
    Vport:ViewPortType;
begin
if Pinit<>nil then
   begin
   setfillstyle(solidfill,black);
   bar(1,1+2+Textheight('1'),xtensao,1+((2+Textheight('1'))*(NMaxTab+1)));
   setfillstyle(solidfill,white);
   {visualiza tabela corrente}
   p:=PInit;
   y:=1+2+textheight('1');
   k:=0;
   while (p<>nil) and (k<NmaxTab) do
     begin
     str:=convRstr(p^.tempo,Ndec);
     x:=xtempo-(textWidth('1')*length(str));
     outtextxy(x,y,str);
     str:=convRstr(p^.tensao,Ndec);
     x:=xtensao-(textWidth('1')*length(str));
     outtextxy(x,y,str);
     k:=k+1;
     y:=y+2+textheight('1');
     p:=p^.prox;
     end;
   end;
end;


procedure lertabela;
```

```
var p:PtLista;
    Y,Valer,tecla:integer;
    continua,Visutab:boolean;
procedure reajusta(q:Ptlista);
begin
if q<>Pedit then begin
                 reajusta(q^.prox);
                 Kpos:=Kpos-1;
                 if Kpos=0 then Pinit:=q;
                 end
                 else Pinit:=q;
end;

begin
clearmensagem;
mensagem(concat('Tabela: Insert Delete ',chr(24),' ',chr(25),' ',chr(26),' ',chr
ValLer:=0; {tempo=0 tensao=1}
VisuTab:=true;
continua:=true;
while continua do
 begin
 if Pedit=nil then begin
                   new(Pedit);
                   Pedit^.tempo:=0.0;
                   Pedit^.tensao:=0.0;
                   Pedit^.prox:=nil;
                   Lista:=Pedit;
                   VisuTab:=true;
                   end;
 Y:=Kpos;
 reajusta(Lista);
 Kpos:=Y;
 if VisuTab then Visutabela;
 VisuTab:=False;
 Y:= 1+((2+textheight('1'))*(Kpos+1));
 if Valer=0 then tecla:=readreal(xtempo,Y,Pedit^.tempo)
            else tecla:=readreal(xtensao,Y,Pedit^.tensao);
 case tecla of
   Tesc:continua:=false;
   TReturn:begin
           if Valer=0 then Valer:=1
              else
              begin
              Valer:=0;
              if Pedit^.prox=nil then
                 begin
                 New(Pedit^.prox);
                 Pedit^.prox^.tempo:=Pedit^.tempo+Ttick;
                 Pedit:=Pedit^.prox;
                 Pedit^.tensao:=0.0;
                 Pedit^.prox:=Nil;
                 if Kpos=(NmaxTab-1) then visutab:=true;
                 end
```

```
            else
            Pedit:=Pedit^.prox;
            if Kpos<(NmaxTab-1) then Kpos:=Kpos+1;
          end;
       end;
Tinsert:begin
        Valer:=0;
        VisuTab:=true;
        p:=Pedit^.prox;
        New(Pedit^.prox);
        Pedit^.prox^.tempo:=Pedit^.tempo+Ttick;
        Pedit:=Pedit^.prox;
        Pedit^.tensao:=0.0;
        Pedit^.prox:=p;
        if Kpos<(NmaxTab-1) then Kpos:=Kpos+1;
        VisuTab:=true;
        end;
 Tup:begin
     if Pedit<>Lista then
        begin
        p:=Lista;
        while p^.prox<>Pedit do  p:=p^.prox;
        Pedit:=p;
        if Kpos>0 then Kpos:=Kpos-1
                  else VisuTab:=true;
        end;
     end;
  TDown:if Pedit^.prox<>nil then
           begin
           Pedit:=Pedit^.prox;
           if Kpos<(NmaxTab-1) then Kpos:=Kpos+1
                               else VisuTab:=true;
           end;
  Tleft:if Valer=1 then Valer:=0;
  Tright:if Valer=0 then Valer:=1;
  Tdelete:begin
          if Pedit=lista then
             begin
             Lista:=Pedit^.prox;
             dispose(Pedit);
             Pedit:=Lista;
             end
             else
             begin
             p:=Lista;
             while p^.prox<>Pedit do  p:=p^.prox;
             p^.prox:=Pedit^.prox;
             dispose(pedit);
             Pedit:=p;
             if Kpos>0 then Kpos:=Kpos-1;
```

```
                end;
            VisuTab:=true;
            end;
     end;
 end;
clearmensagem;
end;

procedure movewindow(Tecla:integer);
begin
setviewport(Xoffset+1,1,Xmax-1,Ymax-1,false);
clearViewport;
setviewport(0,0,getmaxx,getmaxy,true);
case tecla of
  Tright:begin
         Tmin:=Tmin+TTick;
         TMax:=Tmax+Ttick;
         end;
  Tleft:begin
         Tmin:=Tmin-TTick;
         TMax:=Tmax-Ttick;
         end;
  TDown:begin
         Vmin:=Vmin-VTick;
         VMax:=Vmax-Vtick;
         end;
    Tup:begin
         Vmin:=Vmin+VTick;
         VMax:=Vmax+Vtick;
         end;
  end;
end;

procedure zoom;
var tecla:integer;
    Vzoom,Tzoom:real;
    straux:string;
begin
clearmensagem;
straux:='PgDown-Zoom In PagUp-Zoom Out  Move Janela:';
mensagem(concat(straux,chr(24),' ',chr(25),' ',chr(26),' ',chr(27),' <Esc> para
Tecla:=0;
while tecla<>Tesc do
  begin
  while not(Tecla in [Tesc,Tup,Tdown,Tright,Tleft,TPgUp,TPgDown]) do
    begin
    Tecla:=Ord(readkey);
    if tecla=0 then Tecla:=Ord(readkey);
    end;
  if tecla<>Tesc then
     begin
```

```
      case Tecla of
       Tup,Tdown,Tright,Tleft:movewindow(Tecla);
       TPgDown:begin
                Vzoom:=(Vmax-Vmin)/10;
                Tzoom:=(Tmax-Tmin)/10;
                Vmin:=Vmin+Vzoom;
                Vmax:=Vmax-Vzoom;
                Tmin:=Tmin+Tzoom;
                Tmax:=Tmax-Tzoom;
                end;
       Tpgup: begin
                Vzoom:=(Vmax-Vmin)/10;
                Tzoom:=(Tmax-Tmin)/10;
                Vmin:=Vmin-Vzoom;
                Vmax:=Vmax+Vzoom;
                Tmin:=Tmin-Tzoom;
                Tmax:=Tmax+Tzoom;
                end;
      end;
      setviewport(Xoffset+1,1,Xmax-1,Ymax-1,false);
      clearViewport;
      setviewport(0,0,getmaxx,getmaxy,true);
      layout;
      visufuncao(lista);
      mostrasetup;
      tecla:=0;
      end;
  end;
clearmensagem;
end;

procedure redraw;
begin
setviewport(0,0,getmaxx,getmaxy,false);
clearviewport;
Layout;
mostrasetup;
VisuFuncao(Lista);
VisuTabela;
end;

procedure gerador;
const Tciclo=0.5587302E-06;
{frequencia do sinal 0.21 hz a 1087 hz}
var bd:text;
    filtro,atenuacao,amostras:integer;
    frequencia:real;
    ch:char;
    x1,x2,y1,y2:integer;
    tipoonda:string;
    comando,dados:string;
```

```
    continua,valido:boolean;
    D,C:byte;
    j:integer;

procedure amostragem(var dados:string;var amostras:integer);
var T0,T1:Real;
    V0,V1:Real;
    Ttick,Tmax:Real;
    q0,q1:PtLista;
    declive:real;
    v,t:real;
    vc,lb,mb:integer;
begin
dados:='';
if lista<>Nil then
 begin
 q1:=lista;
 T0:=lista^.tempo;
 while q1^.prox<>nil do q1:=q1^.prox;
 Tmax:=q1^.tempo;
 Ttick:=(Tmax-T0)/amostras;
 amostras:=0;
 t:=T0;
 while t<=Tmax do
  begin
  q0:=Lista;
  q1:=Lista^.prox;
  while (t >= q1^.Tempo) do
   begin
   q0:=q1;
   q1:=q1^.prox;
   end;
  V0:=q0^.tensao;
  V1:=q1^.tensao;
  T0:=q0^.tempo;
  T1:=q1^.tempo;
  declive:=(v1-v0)/(t1-t0);
  v:=v0+declive*(t-t0);
  vc:=round((4096*(v+10))/20);
  lb:=vc mod 255;
  if ((lb=10)or(lb=13)) then lb:=lb+1;
  mb:=vc div 255;
  if ((mb=10)or(mb=13)) then mb:=48+mb;
  dados:=concat(dados,chr(lb),chr(mb));
  t:=t+Ttick;
  end;
 end;
 end;
```

```
procedure contadores(var frequencia:real;amostras:integer;var d,c:byte);
var periodo:real;
    Nciclo,Ncic:longint;
    Dc,Cc,Cc1:longint;
begin
 if frequencia=1250 then
   begin
   D:=0;
   C:=0;
   end
   else
   begin
   periodo:=1/frequencia;
   Nciclo:=round(periodo/(amostras*Tciclo));
   Dc:=1;
   Ncic:=0;
   while (Ncic<Nciclo) do
      begin
      Ncic:=171+12*(Dc-1)+Dc*(39+16*255);
      Dc:=Dc+1;
     end;
   Dc:=Dc-1;
   Cc:=1;
   Ncic:=171+12*(Dc-1)+Dc*(39);
   while (Ncic<=Nciclo) do
      begin
      if Cc=0 then Cc1:=255
              else Cc1:=Cc-1;
      Ncic:=171+12*(Dc-1)+Dc*(39+16*(Cc1));
      Cc:=Cc+1;
     end;
   Cc:=Cc-1;
   if Dc>255 then D:=255
             else D:=Dc;
   if Cc>255 then C:=255
             else C:=Cc;
   if C=0    then  Cc1:=255
             else  Cc1:=C-1;
   Ncic:=171+12*(D-1)+D*(39+16*(Cc1));
   frequencia:=1/(amostras*Ncic*Tciclo);
   end;
end;

procedure caixa;
begin
x1:=(getmaxx div 2)-(20*textwidth('1'));
x2:=(getmaxx div 2)+(20*textwidth('1'));
y1:=(getmaxy div 2)-(5*textheight('1'));
y2:=(getmaxy div 2)+(5*textheight('1'));
moveto(x1-1,y1-1);
```

```
lineto(x2+1,y1-1);
lineto(x2+1,y2+1);
lineto(x1-1,y2+1);
lineto(x1-1,y1-1);
setviewport(x1,y1,x2,y2,false);
clearviewport;
setviewport(0,0,getmaxx,getmaxy,true);
apagastr(x1,y1,40,white);
setcolor(black);
outtextxy(x1+10,y1,'GERADOR DE FUNCOES');
setcolor(white);
end;

begin
caixa;
x1:=x1+10;
x2:=x1+10+16*textwidth('1');
outtextxy(x1,y1+5+2*textheight('1'),'1-SENO');
outtextxy(x1,y1+15+2*textheight('1'),'2-TRIANGULAR');
outtextxy(x1,y1+25+2*textheight('1'),'3-DENTE DE SERRA');
outtextxy(x1,y1+35+2*textheight('1'),'4-RECTANGULAR');
outtextxy(x1,y1+45+2*textheight('1'),'5-ARBITRARIA');
outtextxy(x1,y1+55+2*textheight('1'),'OPCAO:');
outtextxy(x2,y1+5+2*textheight('1'), 'AMOSTRAS      :');
outtextxy(x2,y1+15+2*textheight('1'),'FREQUENCIA(hz):');
outtextxy(x2,y1+25+2*textheight('1'),'FILTRO        :');
outtextxy(x2,y1+35+2*textheight('1'),'ATENUACAO     :');
clearmensagem;
mensagem('Introduza o tipo de onda    Esc para abandonar');
continua:=true;valido:=true;
while continua and valido do
 begin
 tipoonda:=lestrxy(x1+6*textwidth('1'),y1+55+2*textheight('1'),1,valido);
 continua:=not((tipoonda>'0')and(tipoonda<='5'));
 end;
if valido then
 begin
 x2:=x2+15*textwidth('1');
 { leitura numero de amostras}
 if tipoonda='5' then
    begin
    continua:=true;
    clearmensagem;
    while continua and valido  do
     begin
     mensagem('Introduza o numero de amostras (1..200)    Esc para abandonar');
     amostras:=leintxy(x2,y1+5+2*textheight('1'),3,valido);
     continua:=not((amostras>0)and(amostras<=200));
     end;
    end
    else
```

```
    begin
    amostras:=8;
    outtextxy(X2,y1+5+2*textheight('1'),'8');
    end;
continua:=true;
clearmensagem;
while continua and valido  do
  begin
  mensagem('Introduza a frequencia da onda   Esc para abandonar');
  frequencia:=leRealxy(x2,y1+15+2*textheight('1'),6,valido);
  continua:=not((frequencia>0)and(frequencia<=1250));
  end;
continua:=true;
clearmensagem;
while continua and valido  do
    begin
    mensagem('Filtra a onda (0->Nao 1->Sim)  Esc para abandonar');
    filtro:=leintxy(x2,y1+25+2*textheight('1'),1,valido);
    continua:=not((filtro=0)or(filtro=1));
    end;
continua:=true;
clearmensagem;
while continua and valido  do
    begin
    mensagem('Atenuacao (0..100) Esc para abandonar');
    atenuacao:=leintxy(x2,y1+35+2*textheight('1'),3,valido);
    continua:=not((atenuacao>=0)and(atenuacao<=100));
    end;
end;
if valido then
begin
comando:='OUTPUT 10;';
contadores(frequencia,amostras,D,C);
comando:=concat(comando,chr(atenuacao),chr(filtro),chr(d),chr(c),chr(amostras));
if tipoonda[1] in ['1','2','3','4'] then
  comando:=concat(comando,chr(16*(ord(tipoonda[1])-49)),chr(32))
  else
  begin
  comando:=concat(comando,chr(7),chr(24));
  amostragem(dados,amostras);
  dados[15]:=chr(amostras);
  comando:=concat(comando,dados);
  end;
setcolor(black);
assign(bd,'gpib0');
rewrite(bd);
{$I-}
writeln(bd,comando);
{$I+}
setcolor(white);
if ioresult<>0 then
    begin
```

```
    clearviewport;
    caixa;
    outtextxy(x1,y1+15+2*textheight('1'),'Erro de escrita no GPIB');
    outtextxy(x1,y1+25+2*textheight('1'),'<QQ> para continuar');
    end
    else
    begin
    clearmensagem;
    mensagem('Transmissao de dados efectuada <QQ> para continuar');
    end;
 ch:=readkey;
 if ((ord(ch)=0) and keypressed) then ch:=readkey;
 close(bd);
 end;
redraw;
end;

begin
   InitGraficos;
   Ysetup:=Ymax-1-(NlinhasSetup*(textheight('1')+3));
   NmaxTab:=(Ysetup-1-2-textheight('1')) div (TextHeight('1')+2);
   Ndec:=3;
   Tmin:=0;
   Tmax:=1;
   Vmax:=1;
   Vmin:=-1;
   Eixos:=true;
   Grelha:=False;
   Picos:=True;
   Vtick:=0.1;
   TTick:=0.1;
   Lista:=nil;
   Pedit:=nil;
   Pinit:=nil;
   Kpos:=0;
   redraw;
   repeat
     straux:=concat('F1-Ler F2-Guardar F3-Tabela F4-Zoom F5-Setup F6-Gerador');
     mensagem(concat(straux,' Esc para abandonar'));
     TeclaEsp:=0;
     while not (TeclaEsp In [Tesc,59,60,61,62,63,64]) do TeclaEsp:=LeTeclaEsp;
     case TeclaEsp of
        59:begin
             Leficheiro;
             redraw;
           end;
        60:escreveFicheiro;
        61:begin
           lertabela;
           redraw;
```

BIBLIOTECA F.E. PORTO

```
            end;
         62:zoom;
         63:begin
            setup;
            redraw;
            end;
         64:begin gerador end;
      end;
   until TeclaEsp=sair;
   closegraph;
end.
```

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA
0000034787