UNIVERSIDADE FEDERAL FLUMINENSE

CENTRO TECNOLÓGICO - ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA DE TELECOMUNICAÇÕES

PLACA ANALISADORA DE ESPECTRO PARA IBM-PC

AUTOR : FÁBIO DAROS DE FREITAS

ORIENTADORES : _____
ALBERTO FERREIRA DE SOUZA , Eng.

_____
JOSÉ BIANCHINI LATGÉ , Eng.

TET

NOVEMBRO 1991

03-01-2006

D E D I C A Ç Ã O

Dedico este trabalho aos meus pais Paulo e Nilma por tudo o que representam para mim e aos amigos pela compreensão e amizade.

A G R A D E C I M E N T O S

Em especial ao amigo e co-orientador Alberto Ferreira de Souza pela total dedicação e disponibilidadde em todas as etapas deste projeto.

Ao Eng. Gustavo Antônio de B. Garnier, orientador na primeira fase do projeto, pela disponibilidade e pelas contribuições no Filtro de FI.

Ao Eng. José Bianchini Latigê pelas contribuições no conversor fly-back.

Ao Eng. Wellington de Souza pela doação dos conversores.

Ao amigo Humberto Figueiró Cocco pela disponibilidade e ajuda em vários ítems deste projeto.

A Rádio Cristais do Brasil S.A. na pessoa do Eng. Antônio Carlos Sales pela confecção e doação dos cristais utilizados neste projeto.

Ao Laboratório de Comunicações Ópticas da Escola de Engenharia da Universidade Federal Fluminense pela utilização de equipamentos necessários ao desenvolvimento deste projeto.

A Eletrônica Tesla Ltda. (Vitória - E.S.) pela utilização de equipamentos e materiais necessários ao desenvolvimento e implementação do circuito analógico.

Por fim a todos colegas e professores do curso de Engenharia de Telecomunicações da Escola de Engenharia da Universidade Federal Fluminense que incentivaram ou contribuiram de alguma forma para a execução deste projeto.

R E S U M O

Este trabalho consiste no projeto e implementação do hardware e software de um analisador de espectro do tipo por varredura (*Scanning Spectrum Analyser - SSA*) para a faixa de 300 Hz a 80 KHz com FI em 200 KHz e filtro à cristal, que utiliza um microcomputador IBM-PC para geração de varredura e apresentação visual do espectro.

ÍNDICE

INTRODUÇÃO pag. 1

CAPÍTULO I - ANÁLISE ESPECTRAL pag 3

I.1 - Sinais e Ortogonalidade pag 3

I.2 - Análise de Fourier pag 6

CAPÍTULO II - ANALISADORES DE ESPECTRO pag 10

II.1 - Analisadores de Espectro em Tempo Real pag 11

II.2 - Analisadores de Espectro por Varredura pag 12

II.3 - Arquitetura Proposta pag 16

CAPÍTULO III - DESCRIÇÃO DO HARDWARE pag 18

III.1 - Filtro Passa-Baixas de Entrada pag 18

III.2 - Multiplicador pag 19

III.3 - Oscilador de Varredura pag 20

III.4 - Filtro de Frequência Intermediária pag 22

III.5 - Amplificador de FI pag 26

III.6 - Detetor pag 29

III.7 - Bloco Conversor Digital/Analógico pag 30

III.8 - Bloco Conversor Analógico/Digital pag 32

III.9 - Interface Digital pag 32

CAPÍTULO IV - IMPLEMENTAÇÃO DO HARDWARE pag 34

IV.1 - Implementação do Cicuito Digital pag 34

IV.2 - Implementação do Circuito Analógico pag 34

IV.3 - Circuitos Auxiliares pag 37

CAPÍTULO V - DESCRIÇÃO DO SOFTWARE pag 40

CONCLUSÃO pag 42

APÊNDICE I - DIAGRAMAS ESQUEMÁTICOS pag 44

APÊNDICE II - LISTAGEM DO SOFTWARE pag 53

INTRODUÇÃO

A análise de sinais no domínio da frequência é um objeto de bastante interesse para engenheiros e outros profissionais das áreas de eletrônica e telecomunicações. O Analisador de Espectro tornou-se uma ferramenta bastante importante para os setores de projeto e desenvolvimento de sistemas e equipamentos, e nos setores de gerenciamento de espectro esta ferramenta é fundamental.

Os projetos de Analisadores de Espectro variam muito devido às várias filosofias de construção e especificações a serem atendidas. A arquitetura dos Analisadores de espectro é intrinsicamente dependente da faixa de frequências que se pretende cobrir, o que leva a uma não generalização da arquitetura de equipamentos projetados sob uma mesma filosofia.

Este trabalho segue a filosofia dos *Scannig Spectrum Analysers* (SSA) descrita por Carlson [1] e Stremler [2], e teve suas especificações estabelecidas mantendo um compromisso com as condições de realizabilidade inerentes a um projeto de graduação em engenharia.

O capítulo I trata da teoria de análise espectral, apresentando a Análise de Fourier para sinais periódicos e não periódicos, levando à visualização de sinais no domínio da frequência. Finalizando, é apresentada o conceito de distribuição espectral de potência de um sinal.

O capítulo II apresenta as principais filosofias de implementação dos analisadores de espectro, mostrando suas vantagens e desvantagens, finaliza com a apresentação da

arquitetura proposta.

O capítulo III trata da descrição do hardware mostrando os circuitos de cada um dos blocos e os pricípios de funcionamento.

O capítulo IV relata a fase de implementação do hardware, apresentando os circuitos auxiliares, os recursos e as técnicas utilizadas.

O capítulo V descreve o Software básico, explicando seu funcionamento e operacionaliade.

O capítulo VI apresenta as conclusões e o roteiro para a continuação deste trabalho.

Os diagramas esquemáticos e a listagem do software são apresentados nos apêndices. Por fim são apresentadas as referências bibliográficas consultadas para a elaboração deste projeto.

C A P Í T U L O I

- ANÁLISE ESPECTRAL:

I.1 - Sinais e Ortogonalidade:

I.1 - 1: Classificação de Sinais

Sinais elétricos são quantidades variantes no tempo, tipicamente uma tensão ou uma corrente, que representam de alguma forma algum fenômeno físico.

Algumas formas de classificação de sinais, segundo Carlson [1], são:

- Sinais de Energia:

São aqueles que existem em apenas um intervalo finito de tempo, ou ao menos tenha a maior parte de sua energia concentrada em um intervalo finito de tempo. Os sinais de energia são aperiódicos por definição. A potência média forneciada por estes sinais encontra contribuição apenas no intervalo de existência do sinal. Assim:

$$P = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |f(t)|^2 \, dt$$

onde t_1 e t_2 são os tempos de início e término do sinal respectivamente.

- Sinais de Potência:

São sinais periódicos nos quais a contribuição de energia por

ciclo é finita e não nula, então a potência média pode ser calculada levando em consideração apenas um ciclo. Esta potência média também é finita e não nula. Assim:

$$P = 0 < \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} |f(t)|^2 \, dt < \infty$$

Onde T é o período do sinal, ou seja:

$$f(t + T) = f(t)$$

I.1 - 2: Sinais no domínio da Frequência

Considere um sinal $v(t) = V \cos(2\Pi f_o t + \varphi)$. A forma usual de representação deste sinal é no domínio do tempo, ou seja, em um par de coordenadas bidimensionais v x t onde a amplitude do sinal varia em função do tempo. Embora este sinal exista fisicamente no tempo, podemos representa-lo graficamente em um sistema de coordenadas bidimensionais, no qual a amplitude é função da sua frequência de oscilação. A este sistema de coordenadas do tipo v x f dá-se o nome de Domínio da Frequência.

Então, a representação de v(t) no Domínio da Frequência seria apenas uma reta paralela ao eixo v, de amplitude V e situada no ponto $f = f_o$ Hz.

Um exemplo é apresentado na figura I.1. Este exemplo mostra um sinal do tipo $v(t) = V_1 \cos(2\Pi f_1 + \varphi_1) + V_2 \cos(2\Pi f_2 + \varphi_2)$. Em (a) temos um sistema tridimensional de coordenadas mostrando o tempo, frequência e amplitude. Em (b) temos o sinal representado no domínio do tempo, onde vê-se v(t) como a soma dos dois cossenos componentes do sinal, e finalmente em (c) temos o sinal representado no domínio da frequência, onde cada um dos dois

cossenos componentes do sinal têm uma representação distinta.

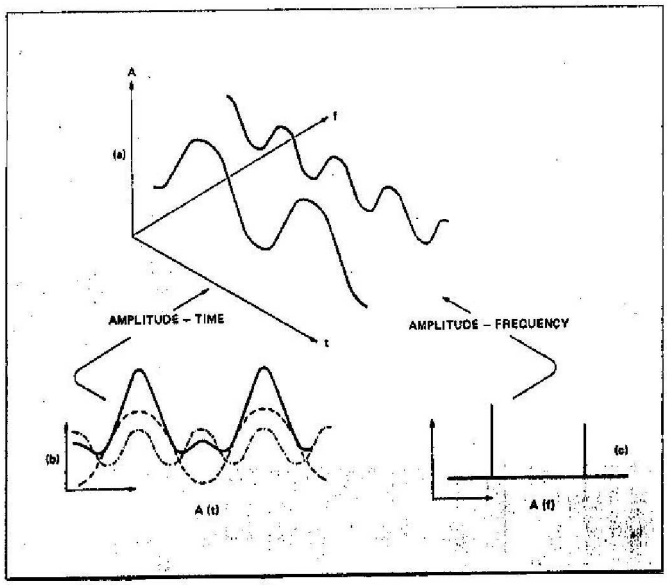


figura I.1

I.1 - 3: Ortogonalidade de Funções

O conceito de ortogonalidade de funções é de extrema importância para o entendimento da Análise de Fourier, que será vista adiante.

Matematicamente, duas funções são ditas ortogonais em um determinado intervalo de seus domínios, segundo Stremler [2], se:

$$\int_{t1}^{t2} f_1(t)\, f_2(t)^* \, dt = \int_{t1}^{t2} f_1(t)^* \, f_2(t) \, dt = 0$$

onde $f_1(t)$ e $f_2(t)$ são funções complexas para o caso genérico, e o operador * indica o conjugado da função.
Estendendo este conceito a um conjunto de funções $f_i(t)$, temos:

$$\text{Se} \quad \int_{t_1}^{t_2} f_n(t)\, f_m(t)^* \, dt = \begin{cases} 0 & ; \; n \neq m \\ K_n & ; \; n = m \end{cases} ,$$

então estas funções são mutualmente ortogonais neste intervalo.

Conceitualmente, a idéia de ortogonalidade de funções é análoga à de ortogonalidade de vetores, o que é razoavel uma vez que o integrando da expressão que verifica a ortogonalidade de funções é análogo a um produto escalar entre vetores. Estendendo-se esta analogia, a um certo conjunto de funções $f_i(t)$ que atende às condições de ortogonalidade, tem-se propriedades análogas às de um espaço vetorial, ou seja, qualquer nova função do tipo $g(t)$ dentro do intervalo de ortogonalidade do conjunto $f_i(t)$ poderia ser representada como uma combinação linear das funções integrantes do conjunto.
Esta idéia é a base da Análise de Fourier.

Para finalizar, é interessante citar dois conjuntos de funções ortogonais:

- $\cos(\theta)$ e $\operatorname{sen}(\theta)$
- $\exp(jn\theta)$, onde $j \triangleq \sqrt{-1}$ e n é um inteiro qualquer.

I.2 - Análise de Fourier

A Análise de Fourier é a ferramenta matemática a qual se lança

mão para se expressar sinais no Domínio da Frequência. Esta transformação de domínio é feita por dois mecanismos básicos: A Série de Fourier, que trata de sinais de potência (sinais periódicos) e a Transformada de Fourier, que trata de sinais de energia (sinais aperiódicos). Estes mecanismos utilizam um conjunto de funções ortogonais para realizar as transformações necessárias.

I.2 - 1: Série de Fourier e Espectro Discreto

O conjunto de funções ortogonais que permite uma maior facilidade de visualização é o das funções seno e cosseno, chamada de Série de Fourier Trigonométrica, embora seja muito comum a utilização da Série de Fourier Exponencial, como em Stremler [2] por exemplo.

Sendo assim, se f(t) é real em (t_1,t_2), temos:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(2\Pi n f_0 t) + \sum_{n=1}^{\infty} b_n \operatorname{sen}(2\Pi n f_0 t)$$

onde:

$$a_0 = \frac{1}{(t_2 - t_1)} \int_{t_1}^{t_2} f(t)\, dt$$

$$a_n = \frac{2}{(t_2 - t_1)} \int_{t_1}^{t_2} f(t) \cos(2\Pi n f_0 t)\, dt$$

$$b_n = \frac{2}{(t_2 - t_1)} \int_{t_1}^{t_2} f(t) \operatorname{sen}(2\Pi n f_0 t)\, dt$$

sendo f_0 constante e n um inteiro qualquer. Este somatório nos leva a um espectro de frequências discreto, com componentes em nfo Hz.

I.2 - 2: Transformada de Fourier e Espectro Contínuo

A Transformada de Fourier é o mecanismo utilizado para a obtenção do espectro de sinais de energia (sinais aperiódicos), embora também sirva para sinais de potência. O seu desenvolvimento é uma extensão da Série de Fourier quando o período dos sinais periodicos tende para o infinito, e é dada pela expressão:

$$F\,[v(t)] \triangleq \int_{-\infty}^{\infty} v(t)\,\exp(-j2\Pi ft)\,dt$$

Onde: f = Frequência em Hz

v(t) = sinal analisado,

pela natureza integral da expressão da Transformada de Fourier, observa-se que o espectro de frequências resultante é de natureza contínua. Doravante vamos utilizar como objeto de referência a Transformada de Fourier, já que a Série de Fourier pode ser considerada como um caso particular da Transformada de Fourier.

I.3 - Densidade Espectral de Energia

Na prática, o objeto alvo na análise de sinais no domínio da frequência é a Densidade Espectral de Energia, que é um conceito extendido aos sinais de potência. A Densidade Espectral de

Energia de um sinal v(t) é dada pela expressão:

$$DEE\ [v(t)] = |V(f)|^2 \quad , \quad \text{onde} \quad V(f) \rightarrow F\ [v(t)]$$

Logo, a energia total deste sinal é obtida integrando-se DEE de $-\infty$ a $+\infty$. O espectro visto no display dos Analisadores de Espectro é a DEE do sinal de entrada.

C A P Í T U L O II

- ANALISADORES DE ESPECTRO:

As duas filosofias básicas para a implementação de Analisadores de Espectro, segundo [3], são: Em Tempo Real e por Varredura e Sintonizado. Varios autores como Carlson [1] e Stremler [2] denominam os Analisadores de Espectro por Varredura e Sintonizado de simplesmente Analisadores de Espectro por Varredura (*Scanning Spectrum Analyzers - SSA*). Embora não seja muito rigoroso a nível de nomeclatura, já que os analisadores em tempo real também utilizam circuitos de varredura, esta será a nomeclatura adotada doravante devido ao fato de ser a mais comum. Outra filosofia de análise espectral são os Analizadores de Fourier, que realizam a computação matemática da Transformada de Fourier. Foram implementados equipamentos analógicos deste tipo mas sem muito sucesso devido a problemas inerentes à complexidade dos circuitos. Também foram utilizadas configurações com computadores interfaceados com placas conversoras Análogico/Digital que faziam a aquisição do sinal durante um intervalo de tempo e computavam a Transformada Rápida de Fourier (*Fast Fourier Transform - FFT*), que é um método numérico interativo para se obter a Transformada de Fourier de funções. Estas configurações não obtiveram sucesso, até o passado próximo, devido à sua depêndencia intrínseca do desempenho da máquina que computa a FFT. Atualmente, já se dispõe de máquinas, as máquinas RISC (*Reduced Instruction Set Computers*) por exemplo, com

desempenho suficiente para efetuar esta tarefa.

Nos últimos anos, devido à grande evolução e consequente aumento de realizabilidade dos circuitos integrados VLSI (*Very Large Scale Integration*) surgiram os Processadores Digitais de Sinais (*Digital Signal Processors - DSP*) que são chips que realizam, entre outras tarefas, a FFT de sinais, fazendo ressurgir o Analizadores de Fourier por Hardware. Obtem-se com esta configuração Analisadores de Espectro bastante versáteis e de excelentes recursos gráficos. Em Carlson [1] tem-se introdução razoável sobre a FFT.

II.1 - Analisadores de Espectro em Tempo Real (RTSA)

Os Analisadores de Espectro em Tempo Real, ou Multifunal segundo [3], consistem de um arranjo filtros passa-faixa, um para cada frequência a ser analisada. Na figura II.1 (a) vê-se o diagrama de blocos.

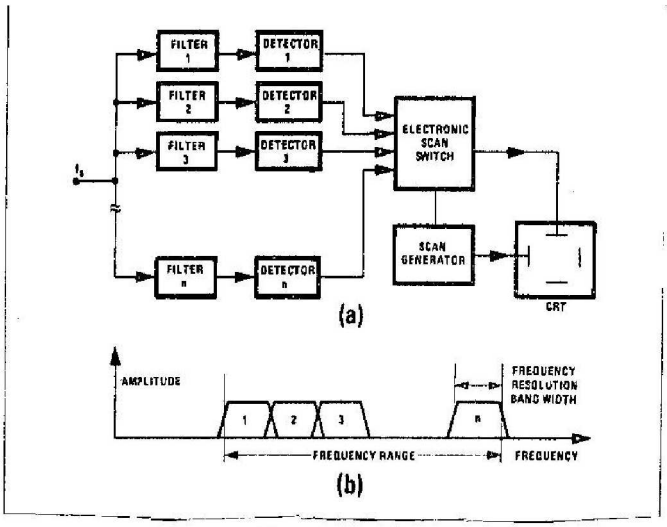


figura II.1

O sinal é injetado simultâneamente em todos os filtros, após cada filtro a componente, de respectiva frequência do filtro, passa por um detetor a diodo para ter seu valor RMS extraído. De posse dos valores RMS de cada componente, o circuito de controle faz a varredura do cinescópio (CRT) sincronizada com a componente a ser injetada na deflexão vertical.

A figura II.1 (b) mostra o efeito de cobertura de espectro deste arranjo. Obviamente o range de frequências a ser cobertas depende do número e banda passante dos filtros, o que torna esta arquitetura proibitiva para determinados ranges. Como a utilização de filtros muito estreitos é inviável devido ao acréscimo do número destes filtros para cobrir determinado range, estes analisadores são bastante imprecisos levando a serem denominados de Analisadores de Banda. As características dos filtros também se diferem bastante, aumentando ainda mais a imprecisão destes instrumentos.

Uma variação interessante desta arquitetura é obtida trocando o arranjo de filtros por um filtro de características variáveis por meio de um circuito de controle e utilizando um só detetor após o filtro. Apesar desta arquitetura ser mais racional que a Multicanal, a imprecisão continua devido ao filtro e o detetor não manterem as mesmas características ao longo da faixa de frequências de operação.

II.2 - Analisadores de Espectro por Varredura (SSA)

O Analisador de Espectro por Varredura (*Scannig Spectrum Analyser - SSA*) é a arquitetura que obteve maior sucesso devido às

vantagens apresentadas. Apesar de terem limitações devido ao fato de não serem de tempo real, inviabilizando de análise de transientes por exemplo, estes analisadores têm performance bastante superior aos analizadores em tempo real.

O diagrama de blocos é mostrado na figura II.2.

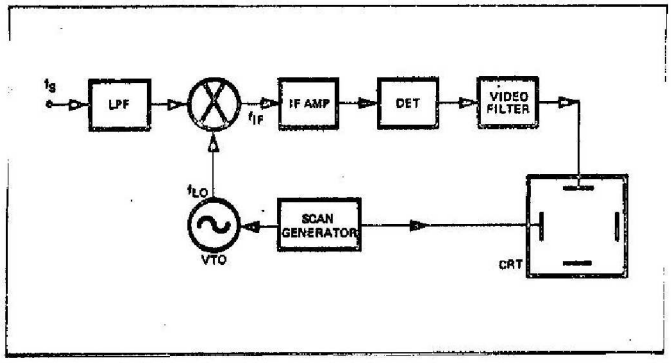


figura II.2

O analisador é sintonizado através de um oscilador de varredura controlado por tensão VCO (*Voltage Controlled Oscillator*), também chamado de VTO (*Voltage Tuned Oscillator*), que é mixado com o sinal de entrada gerando frequências iguais à soma e a diferença das frequências dos sinais mixados. A frequência intermediária FI, frequência central do filtro passa-faixa de FI, é escolhida acima da máxima frequência a ser analisada fs_{max}. Na figura II.2 este filtro encontra-se dentro do bloco do Amplificador de FI. O VCO

oscila entre a frequência de FI e FI + fs_{max}. Somente haverá sinal na saída do filtro de FI quando a frequência de saída do VCO (fvco) for igual à soma da frequência do sinal de entrada (fs) com a FI, ou seja, quando fvco - fs = FI (quando se utiliza a diferença entre as frequências, que é a maioria dos casos). A saída do filtro de FI passa por um amplificador de FI e pelo detetor a diodo para se obter seu valor RMS que é injetado na deflexão vertical do CRT. A deflexão horizontal do CRT é gerada a partir do gerador de tensão de varredura do VCO, havendo assim uma correspondência bi-unívoca entre cada frequência a ser analisada e a sua posição no eixo horizontal do CRT, obtendo-se assim o espectro de frequências do sinal de entrada mostrado no CRT.

Vamos a um exemplo numérico:

Vamos utilizar uma FI de 100 MHz e injetar um sinal senoidal de 100 KHz na entrada. A varredura do VCO começa em 100 MHz, a saída do multiplicador então é 100 MHz - 100 KHz = 99.9 MHz que faz com que a saída do filtro FI seja nula . A esta frequência inicial de saída do VCO está relacionada uma tensão mínima de varredura e consequentemente uma tensão de deflexão mínima, o que posiciona a "mosca" na extremidade esquerda do CRT correspondendo à posição de 0 Hz. A tensão de varredura vai aumentando fazendo com que fvco aumente e com que a "mosca" ande para a direita no CRT. Quando a tensão de varredura for tal que gere fvco de 100 MHz + 100 KHz, a diferença fvco - fs será igual à FI, levando à presença do sinal na saída do filtro de FI. Após se extrair o valor RMS deste sinal através de um detetor à diodo, teremos na saída deste detetor o nível correspondente ao valor RMS do sinal de entrada, então haverá uma deflexão vertical proporcional na posição do eixo

horizontal correspondente a 100KHz. A varredura prosseguirá até a frequência máxima escolhida que corresponderá à extremidade direita do CRT, e então irá reiniciar pela frequência mínima. Agora, se injetarmos na entrada um sinal composto da soma de duas senóides de frequências diferentes, obteremos pelo mesmo processo as suas amplitudes RMS plotadas nas respectivas posições correspondentes ao eixo de frequências no CRT. Expandindo o raciocínio ao domínio da Análise de Fourier, obteremos as amplitudes das componentes e consequentemente o espectro de frequências de qualquer sinal injetado na entrada.

Se o sinal de entrada conter frequências maiores que FI, irão aparecer frequências espúrias na frequência de FI que levarão a falsas componentes do espectro do sinal plotados no CRT. Para eliminar este problema utiliza-se um filtro passa-baixas antes do sinal ser injetado no multiplicador.

Denomina-se ciclo de varredura a cada vez que o VCO gera todas as frequências desde $fvco_{min}$ a $fvco_{max}$. A este ciclo de varredura está associado um período de varredura que é o tempo gasto para se realizar um ciclo de varredura. Este período de varredura mantém um compromisso com a banda passante do filtro de FI, como será visto posteriormente.

As vantagens desta arquitetura são inerentes ao fato desta ser uma arquitetura superheterodina, ou seja, como os cicuitos que tratam cada componente operam com uma frequência fixa, a FI, eles são mais fáceis de serem projetados e mantém suas características estáveis para todas as componentes do sinal. Maiores detalhes serão vistos adiante quando se tratar do projeto dos circuitos.

II.3 - Arquitetura Proposta

A arquitetura proposta neste trabalho, consiste em utilizar um microcomputador interfaceado com conversores digital/analógico e analógico/digital para gerar as tensões de varredura e apresentar, de forma gráfica, o espectro do sinal de entrada. O software é o encarregado de gerar os números binários que serão injetados na entrada do conversor D/A para gerar as tensões de varredura e fazer a aquisição da amplitude RMS de cada componente através do conversor A/D, plotando-a no local apropriado no monitor do microcomputador. O digrama de blocos é mostrado na figura II.3.

As especificações escolhidas são:

- Range: 300 Hz a 80 KHz.
- Faixa dinâmica: 36 dB.
- FI: 200 KHz.
- Fitro de FI: do tipo à cristal.
- Escala vertical: logarítmica.
- Escala horizontal linear.

Lembrando que estas especificações foram escolhidas de acordo com os propósitos deste projeto.

Os blocos componentes serão vistos separadamente no próximo capítulo.

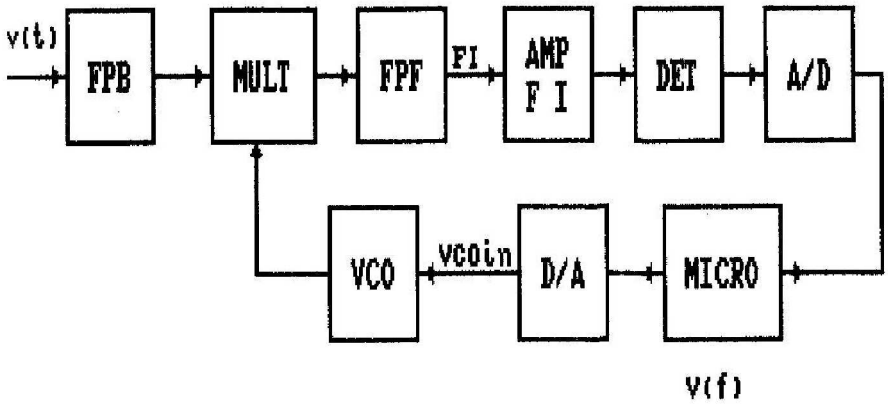


figura II.3 - Diagrama de Blocos da Arquitetura Proposta

figura II.3

C A P Í T U L O III

- DESCRIÇÃO DO HARDWARE:

A descrição do hardware será feita a nível de descrição dos blocos componentes segundo a figura II.3. A escolha dos componentes foi feita de acordo com os propósitos e especificações deste projeto. Utilizou-se, sempre que possível, componentes integrados devido a sua relação custo/performance.

III.1 - Filtro Passa-Baixas de Entrada (FPB)

Ao se transladar o espectro do sinal a ser analisado há o risco do produto da portadora e seus harmônicos com os componentes imagem de mais alta ordem do sinal, componentes de frequência negativa, cairem na FI, gerando assim falsos harmônicos plotados no espectro do sinal. Para evitar este risco, limita-se o sinal a ser analisado em faixa através de filtros.

Neste projeto, foi utilizado um filtro passa-baixas de entrada do tipo RC que satisfez plenamente os propósitos acima citados.

Este filtro foi implementado com a etapa moduladora, e é composto por R1 e C1 do Diagrama 01. Os valores de R e C foram obtidos da frequência de corte dos filtros RC:

$$f_c = \frac{1}{2\Pi\ RC}$$

A frequência de corte obtida foi de 86,8 KHz, a qual o filtro

atenua o sinal em 3 dB.

Sendo o "worst case" quando a frequência da portadora estiver em 280 KHz, ou seja, quando a Banda Lateral Inferior Imagem (BLII) estiver mais próxima das frequências reais, teremos: A maior frequência da BLII interferente será igual à FI (200 KHz). A componente das frequências reais que deu origem a esta componente imagem é máxima frequência que pode conter o sinal de entrada, e é dada por:

$$f_{max.} = FI + (280 \text{ KHz} - 80 \text{ KHz})$$

$$f_{max.} = 400 \text{ KHz}$$

Ou seja, a menor frequência do sinal de entrada que irá gerar harmônicos falsos é 400 KHz. Logo, a atenuação dada pelo FPB a esta componente é a menor rejeição de frequência imagem do equipamento. Para o FPB implementado esta rejeição é de 13,5 dB.

III.2 - Multiplicador (MULT)

O bloco multiplicador é na verdade um modulador AM-DSB-SC implementado com o modulador balanceado integrado MC 1496. O circuito é mostrado no Diagrama 01.

A impedância de entrada é dada pelos resistores R4, R5, R6 e R7; sendo:

$$R_{in} = R5 + R4 // R6 // R7$$

no circuito temos:

$$R_{in} \simeq 82 \ \Omega .$$

O potenciômetro POT1 é para correção do índice de modulação, e serve para zerar o nível de saída da portadora. Este circuito garante uma supressão de portadora tipicamente de 65 dB.

Os níveis de tensão pico a pico do sinal de entrada, da portadora e da saída são:

$$V_{a_{max}} = 1 \text{ V}$$

$$V_c = 800 \text{ mV}$$

$$V_o = 1{,}8 \text{ V}$$

III.3 - Oscilador de Varredura (VCO)

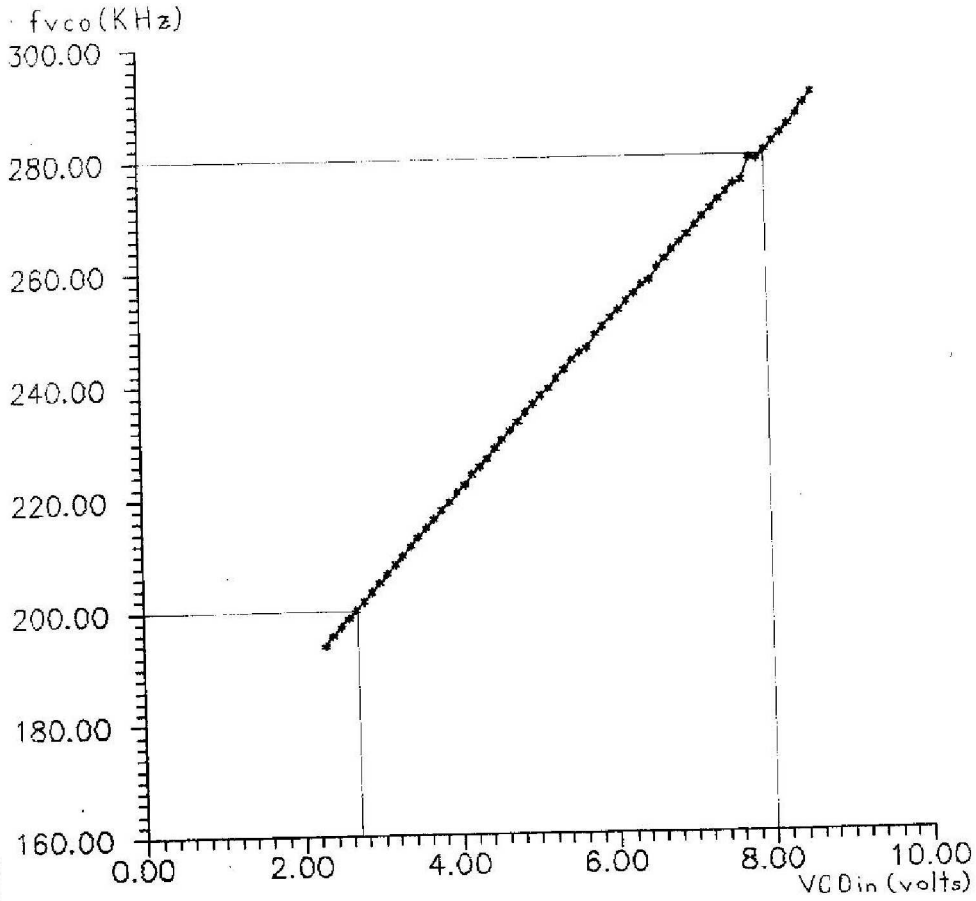
O Oscilador de Varredura foi implementado utilizando o VCO do circuito integrado CD 4046, que é um PLL CMOS. O circuito utilizado é mostrado no Diagrama 01.

A frequência central de oscilação é dada pelos resistores R16, R17 e pelo capacitor C9. A frequência central escolhida foi no entorno de 230 KHz. Esta frequência foi escolhida mantendo-se o compromisso entre o nível de off-set da tensão de controle VCO_{in} e a frequência central ideal, que é 240 KHz. A escolha dos resistores e do capacitor de oscilação teve uma parcela experimental, já que a equação do VCO fornecida pelo manual do componente só é uma boa aproximação para a configuração em malha fechada.

Os resistores R14 R15 e POT2 são para ajustar o nível de tensão de entrada da portadora, e o capacitor C6 é um dc-block.

Foi levantada experimentalmente a curva do VCO, que é mostrada na figura III - 1. Através desta curva acha-se uma inclinação do VCO:

$$\frac{\partial f_{vco}}{\partial VCO_{in}} = \frac{80 \text{ KHz}}{5220 \text{ mV}} \simeq 15{,}32 \text{ Hz/mV}$$


fvco(KHz)
300.00
280.00
260.00
240.00
220.00
200.00
180.00
160.00
0.00
2.00
4.00
6.00
8.00
10.00
VCOin (volts)

21

03-01-2006

III.4 - Filtro de Frequência Intermediária (FPF)

O Filtro de FI é o responsável pela seleção da componente a ser analizada. Consequentemente, ele é o responsável pela precisão do discernimento das componetes. Quanto mais estreito for o filtro, mais a sua resposta se aproximará de uma componente pura. Porém, há uma relação de compromisso entre a banda de passagem do filtro de FI e a taxa de varredura. Este compromisso é a principal limitação dos SSA. Segundo Carlson [1] e Stremler [2]:

Se:

S_r = Taxa de varredura em Hz/segundo

B = Banda do filtro

f_{min} = Frequência inferior da faixa analizada

f_{max} = Frequência superior da faixa analizada

T_B = Tempo gasto para percorrer B Hz

T = Tempo gasto para cada varredura

δ = Resolução

e:

$$S_r = \frac{f_{max} - f_{min}}{T}$$

$$\delta = \frac{f_{max} - f_{min}}{B}$$

Da teoria dos filtros, temos que a Banda mínima para se passar um pulso de largura τ através de um Filtro passa Baixas é:

$$B = \frac{1}{\tau}$$

Segundo Carlson [1] e Stremler [2], este resultado também se aplica ao filtro de FI de analisadores de espectro por varredura, o que é intuitivo, pois efeito da varredura ao passar por uma componente que sintoniza na FI é análogo à resposta de um FPF a um pulso de "burst", e este modelo por sua vez corresponde a um pulso modulado na frequência fo aplicado a um FPF centrado em fo. Logo por intuição espera-se obter o mesmo resultado.

Então:

$$T_B \geq \tau$$

$$T_B \geq \frac{1}{B}$$

Mas:

$$T_B = \frac{T}{\delta}$$

então:

$$\frac{T}{\delta} \geq \frac{1}{B}$$

$$\frac{T\,B}{f_{max} - f_{min}} \geq \frac{1}{B}$$

logo:

$$\frac{f_{max} - f_{min}}{T} \leq B^2$$

$$S_r \leq B^2$$

Daí, tiramos e relação entre o tempo de varredura para determinada faixa e a Banda do FPF:

$$T \geq \frac{f_{max} - f_{min}}{B^2} .$$

Para o nosso caso:

f_{max} = 280 KHz

f_{min} = 200 KHz

B = 170 Hz (Medido em laboratório)

então:

$T \geq 2,77$ s.

Segundo Engelson [4], o tempo de varredura ótimo é dado é 0,44 T. Para o nosso caso, seria de 1,21 s.

O Filtro de FI utilizado é mostrado no Diagrama 02. Este filtro consiste de um Filtro Passa Baixas seguido de um Filtro Passa Faixa.

Os resistores R18 e R20 formam um divisor de tensão que reduzem a

amplitude do sinal de entrada de 1,8 Vpp para 0,2 Vpp.

O resistor R19, o capacitor C10 e o trimmer TRIM 1 formam o FPB. A frequência de corte é em 43 MHz. Este filtro serve para eliminar os harmônicos de mais alta frequência da portadora, e suas respectivas Bandas Laterais, que passariam pelo FPF através da capacitância parasita do cristal, mesmo com a presença do circuito para minimizar os efeitos desta capacitância.

O circuito do FPF é formado pelo transitor Q1, o cristal X1 e o capacitor variável TRIM 2. O cristal X1 é o elemento de filtragem, e TRIM 1, Q1 e seu respectivo circuito de polarização formam um circuito para eliminar a capacitância parasita presente no cristal. Esta capacitância é devido aos terminais metálicos de fixação do cristal, e encontra-se eletricamente em paralelo com o cristal. O circuito funciona da seguinte maneira:

Faz-se passar por TRIM 1 o sinal que passa por X1 defasado de 180°. Se o valor de TRIM 1 for igual ao valor da capacitância parasita do cristal, então as mesmas componentes que passam pela capacitância parasita do cristal passarão também, com as mesmas características, por TRIM 1. Ao se somarem na saída do filtro, estas componentes se anulam devido à diferença de fase de 180° entre si. Então, obtem-se desta forma a resposta do filtro dependente apenas das características ressonantes do cristal.

Na prática esta capacitâcia não é totalmente eliminada, devido à presença de outras reatâncias, mas atinge-se bons resultados, tão melhores quanto melhor a precisão de sintonia de TRIM 1.

Os sinais defasados são conseguidos por transformadores com derivação central no secundário, mas neste circuito prefiriu-se obter estes sinais através das saídas de coletor e emissor de um

amplificador Classe A a transistor. Como estes sinais devem ter a mesma amplitude, o transistor Q1 é polarizado para ter ganho unitário com RC e RE iguais, no circuito R23 e R24. Devido à frequência de operação, os efeitos da capacitância base - coletor prejudicam a resposta em frequência deste amplificador. Este efeito é acentuado à medida que cresce o valor do resistor de coletor, já que o aumento do valor deste resistor causa o aumento das reatâncias parasitas no resistor, que se somam à capacitância base coletor deteriorando ainda mais a resposta em frequência do amplificador. Quando não se utiliza circuitos de compensação, a solução é manter o valor do resistor de coletor pequeno o suficiente. Achou-se por experimentos que valores acima de 5K6 Ω deterioravam muito a resposta para o circuito proposto, foi utilizado então o valor de 1 KΩ para R23, e consequentemente R24.

O resistor R25 é apenas uma carga para o circuito e os capacitores C11 e C101 são dc-blocks.

III.5 - Amplificador de FI (AMP FI)

O Amplificador de FI tem a finalidade de elevar o sinal a níveis apropriados para se extrair o seu nível RMS. A presença deste amplificador antes do circuito detetor ainda tem a vantagem de se melhorar os níveis de ruído do circuito. Sendo a Figura de Ruído de um sistema um parâmetro utilizado para se medir a a Relação Sinal/Ruído deste sistema, Carlson [1] mostra que em um sistema de Banda limitada, que é o nosso caso, a maior contribuição para a deterioração da Figura de Ruído total do

sistema provém do primeiro estágio. Esta relação é dada por:

$$F_T = F_1 + \frac{F_2 - 1}{g_1} + \frac{F_3 - 1}{g_1 \, g_2} + \ldots$$

Onde:

F_T : Figura de Ruído Total do Siatema

F_n : Figura de Ruído do n-ésimo estágio

g_n : Ganho do n-ésimo estágio

Esta fórmula é conhecida como Fórmula de Friis, e conclui-se que quanto menor a Figura de Ruído e maior o ganho do primeiro estágio de um sistema, melhor será a Figura de Ruído Total desta sistema. Entende-se por uma Figura de Ruído melhor, quanto menor é o seu valor.

Como no nosso caso pode-se considerar a saída do Filtro de FI como fonte de sinal para um sistema de Banda limitada, o primeiro estágio deste sistema deve ser um estágio de ganho e de baixa Figura de Ruído.

O Amplificador de FI foi implementado através de um amplificador de três estágios, e é mostrado no Diagrama 03.

Devido às características de ruído, e à necessidade de se ter um circuito de alta impedância na saída do filtro, utilizou-se como primeiro estágio um amplificador Classe A com FET (*Field Effect Transistor*). Este estágio é composto por Q2, que foi polarizado para V_{GS} = - 2,5 V. O ganho de tensão deste estágio é aproximadamente:

$$A_{v1} \simeq \frac{R_D}{R_S} = \frac{R27}{R28} \simeq 4$$

O segundo estágio é um Seguidor de Emissor com acoplamento direto implementado com o transistor Q3, que serve de driver para o terceiro estágio. Utilizou-se a tensão de Dreno de Q2 para polarizar a base de Q3. Esta tensão V_b é de aproximadamente 4 V. Este estágio tem ganho de tensão aproximadamente unitário.

O terceiro estágio é um amplificador Classe A implementado com Q4 ecom ganho de tensão dado por:

$$A_{v3} \simeq \frac{R_C}{R_E} = \frac{R32}{R33} \simeq 3,6$$

O ganho de tensão total é:

$$A_{VT} = A_{v1} \times A_{v2} \times A_{v3} = 4 \times 1 \times 3,6 = 14,4$$

Vale à pena ressaltar que R27 e R32 são da ordem de grandeza de R23 pelos motivos citados na descrição do circuito do Filtro de FI.

Observando-se a particularidade de que nos interessa apenas a frequência de FI, foi acoplado à saída do Amplificador de FI um circuito LC tanque, que é um Filtro Passa Faixa, para diminuir a amplitude das componentes espúrias. A frequência de ressonância deste circuito deve ser, obviamente, a FI. Esta frequência é dada por:

$$f_{LC} = \frac{1}{2\pi\sqrt{LC}}$$

Utilizou-se L = 100 μH e C = 6,8 nF, que são valores disponíveis comercialmente. Assim obteve-se $f_{LC} \simeq 193$ KHz. A atenuação proporcionada pelo LC em 200 KHz é praticamente 0 dB.
Com a utilização deste filtro, obteve-se uma melhora considerável do ruído de fundo na saída do detetor, bem como suprimiu-se as componentes cujo batimento sintonizava na segunda frequência de ressonância do cristal. Esta frequência é de aproximadamente 760 KHz.

III.6 - Detetor (DET)

O Detetor é o elemento responsável pela extração do valor RMS da componente sintonizada na FI. Devido à frequência de operação, os detetores comuns à diodo não funcionam bem. A solução foi utilizar um detetor de precisão. O circuito implementado é mostrado no diagrama 03, e é composto pelo CI 1, R34, R35 e D1.
Utilizou-se um diodo de germânio devido ao seu pequeno potencial de polarização direta, da ordem de 0,1 V.
O funcionamento é o seguinte: Da teoria dos Amplificadores Operacionais, segundo Millman [5], a tensão entre as entradas inversora e não inversora tem que ser nula. Assim, quando esta tensão tender a variar devido à tensão gerada pela saída do Amplificador de FI, a saída do Operacional irá gerar uma corrente para "matar" esta tensão. Acontece, que esta corrente passa por caminhos distintos conforme a sua direção. Isto leva a uma tensão de saída do Operacional variando segundo o ganho do amplificador inversor, Av = R35/R34 = 10, quando a corrente flui no sentido o qual ela passa apenas por R35 (diodo D1 inversamente

polarizado), e ou a tensão de saída fica praticamente nula, quando a corrente flui no sentido o qual ela passa pelo diodo D1 (diodo D1 diretamente polarizado).

Após o retificador, é inserido o integrador formado por R36 e C20 que vai manter a tensão no capacitor em um nível igual ao valor RMS do sinal na saída do detetor, que é o valor RMS da componente sintonizada.

III.7 - Bloco Conversor Digital/Analógico (D/A)

O D/A é o responsável pela geração do sinal de varredura, logo, ele e seus circuitos adjacentes são elementos críticos no projeto deste equipamento.

Foi utilizado o conversor D/A MC 1408, que é um conversor de 8 bits e tem especificações que atenderam às necessidades deste projeto. O circuito implementado é mostrado no Diagrama 04.

Devido a problemas de ruído, torna-se necessário a utilização de pontos de terra distintos para o circuito digital e o circuito analógico. Estes terras se conectam em apenas um ponto, e a simbologia destes dois terras é mostrada em destaque no Diagrama 04, bem como em todos os outros diagramas que utilizam estes dois terras.

O resistor R38, os diodos D1, D2 e D3 são utilizados para a tensão

de referência de CI2, já que a saída deste D/A é função do número binário da entrada, do resistor R37 e desta tensão de referência. Assim, amarrou-se esta tensão de referência pelos três diodos, o que dá uma tensão estável de aproximadamente 3 x 0,65 V = 1,95 V.
O capacitor C21 é filtragem desta tensão de referência.
A alimentação analógica do CI2 é um ponto crítico e foi estabilizada através de R42 e Z1 em 13,7 V.
O driver do D/A foi feito com CI4, que é o Operacional TL 071.
O resistor R40 serve para transformar a saída do D/A, que é uma saí da em corrente, em uma tensão. Esta tensa corrente varia de 0 a - I, o que levou à necessidade de configurar o Operacional como um amplificador inversor. Os resistores de ganho são R41 e POT5. O capacitor C23 é para filtraagem.
Devido à presença de um off-set no range de tensões de VCOin, a tensão correspondente à mínima frequência não corresponde a 0 V, foi utilizado uma tensão de off-set na entrada não inversora do Operacional ajustada através de POT4 de forma que se tenha o off-set necessário na saída do OP-AMP. Devido à ordem de grandeza deste off-set necessário, aproximadamente 3 V, não foi possíval utilizar o ajuste de off-set do próprio OP-AMP.

Desta forma, o ajuste das frequências mínima e máxima do VCO é feito por POT4 e POT5 respectivamente.
Veremos adiante que este ajuste serve para maximizar a resolução do D/A, pois a calibração é feita por software através de um circuito que gera frequências padrão.

III. 8 - Bloco Conversor Analógico/Digital (A/D)

Este bloco é o responsável pela leitura do valor RMS da componente sintonizada, e resume-se basicamente no conversor Analógico/Digital. O conversor utilizado foi o AD 572, que é um conversor A/D de 12 bits. O circuito é mostrado no Diagrama 05.
O AD 572 tem todo o circuito de driver e buffer para o sinal de entrada no próprio chip, o que simplificou ao extremo o circuito.
Os únicos componentes externos necessários são os potenciômetros para ajuste de zero e ajuste de ganho, que são POT7 e POT6. Estes ajustes servem para se ter a máxima precisão na leitura do sinal, maximizando o número de níveis representativos para a máxima amplitude do sinal.
Os capacitores C24, C25 e C26 são para filtragem da alimentação.
Neste circuito também foi necessário, por motivos já citados, utilizar terra analógico e terra digital distintos.

III.9 - Interface Digital

Este circuito é o responsável pelo interfaceamento da placa com o IBM-PC, sendo implícito ao Bloco do microcomputador (MICRO) no Diagrama de Blocos da figura II.3. Esta interface é responsável pela decodificação dos endereços pelos quais o IBM-PC irá se comunicar com a placa, bem como pela leitura e escrita nos conversores A/D e D/A. O circuito foi desenvolvido com lógica TTL e segundo Eggebrecht [6], e é mostrado no Diagrama 06.

Devido às características desta interface, cinco endereços contíguos atendem às necessidades. Utilizou-se então decodificador 74138 (B3 do Diagrama 06), que é um decodificador para oito endereços. O endereço inicial escolhido foi h300. Logo, os endreços os quais se quer decodificar vão de h300 a h307. A decodificação dos três bits menos significativos é feita por B3, e os bits restantes são decodificados pelas portas A3A, A2A, A2B, A3D e A3C. A1 é um buffer bidirecional de 8 bits, e foi utilizado para permitir isolamento do barramento de dados do IBM-PC. A direção do fluxo de dados é dada pelo sinal $\overline{PCIOR}$. O isolamento do barramento da dados é decodificado por B2B.

C A P Í T U L O I V

- IMPLEMENTAÇÃO DO HARDWARE:

O hardware foi montado em uma placa padrão para IBM-PC, que se encaixa em um dos slots internos da main board do micro. Apesar desta característica peculiar, os procedimentos para a implementação dos circuitos digitais e analógicos foram distintos. Por este motivo os temas serão expostos separadamente.

IV.1 - Implementação do Circuito Digital:

O circuito digital, que se resumiu basicamente à interface digital, foi implementado utilizando-se a técnica de "Wire-Wrap", que foi apropriada para este protótipo.
A montagem foi assistida por um software para auxílio à fiação de placas desenvolvido no Laboratório de Sistemas de Computação da C.O.P.P.E./U.F.R.J., o que facilitou bastante esta etapa. Primeiro foi feito o "placement" dos chips, e então utilizou-se a listagem de saída deste software para o roteiro de fiação.
Após a implementação foi feito um programa teste para a depuração.

IV.2 - Implemntação do Circuito Analógico:

Esta etapa foi bastante árdua. Logo de início foi necessário o projeto de um conversor DC-DC para se isolar magneticamente a alimentação dos circuitos analógico e digital, satisfazendo assim a exigência de terras analógico e digital distintos.

Foi projetado e montado um conversor chaveado controlado por um modulador PWM (*Pulse Width Modulation*), que é mostrado no Diagrama 07 . Utilizou-se o modulador SG 3524 (CI5), que é amplamente utilizado para este fim. O funcionamento é o seguinte:
O CI5 gera uma onda quadrada com duty-cicle dependendo da tensão na entrada do comparador de controle, pinos 1 e 2. Esta onda é aplicada à base de um transistor para deixá-lo em corte ou em saturação. Este transitsor é ligado à alimentação através de um transformador. Quando o transistor entra em saturação, a corrente de coletor vai subindo linearmente até um valor máximo e o primário do transformador acumula energia neste período. Ao se cortar o transistor, a corrente no primário do transformador cai abruptamente, o que faz com que apareca uma tensão reversa bastante alta entre seus terminais. Neste instante ocorre a transferência de energia para o secundário. No secundário há um diodo que só deixa circular corrente neste sentido, e esta corrente então carrega um capacitor durante este período. Quando o transistor entra em saturação novamente, a carga do capacitor se descarrega em cima do cicuito acoplado a ele. Se não hover nenhum circuito de carga acoplado ao capacitor, a tensão no capacitor tende ao infinito. Na prática ela chega a valores altíssimos limitados pelas perdas no capacitor.
O conversor foi dimensionado para conseguir fornecer o dobro da potência necessária. Segundo estas especificações, o valor da corrente de coletor de pico é de 2 A. Como o β na saturação para transistores de média potência é da ordem de 10, precisou-se de uma corrente de base de cerca de 200 mA. O CI5 não consegue fornecer esta corrente, por isso utilizou-se Q5 em configuração

Darlington com Q6. Os resistores R49 e R50 são para descarregar a capacitância de base diminuir a corrente de base de Q5 e Q6. O transformador T1 tem núcleo de ferrite devido à frequência de operação. Esta frequência é dada por:

$$f = \frac{1,15}{R_T C_T}$$; onde R_T e C_T são R44 e C27.

Logo: $f \cong 20$ KHz.

A tensão de controle é gerada pelo enrolamenteo de controle do secundário de T1 e é ajustada por POT8. Esta tensão é comparada a uma tensão de referência dada pelo divisor resistivo formado por R45 e R46, que é de 1,22 V. Quando a tensão de controle tende a ficar maior que a referência, o duty cicle da onda quadrada é diminuido, fazendo assim que o primário de T1 acumule menos energia, o secundário recebe menos energia fazendo que a tensão se regule em um nível determinado. Quando a tensão de controle tende a aumenter, ocorre o processo contrário. O nível de regulação destes conversores é dado pelo ganho do comparador, que determina a rapidez de resposta do circuito de regulação. O nível de regulação obtido foi excelente. A desvantagem destes conversores é que devido ao chaveamanto do transistor, a fonte se torna bastante ruidosa. Utilizando-se filtragem para alta frequência este ruído é minimizado. Isto foi feito em todos os blocos, através de um capacitor de 0,1 μF na alimentação, servindo de by-pass para estas frequências.

Para a montagem dos blocos, foram confeccionadas manualmente as placas de circuito impresso para todos os circuitos, que foram acopladas à placa principal. Optou-se por fazer placas separadas

para cada bloco, exceto FPB, MOD e VCO, que foram montados numa mesma placa para maior facilidade. A interconecção entre as placas foi feita com fio coaxial para maior imunidade a induções. A alimentação foi feita por quatro barramentos respectivos a +15 V, -15 V, +9 V e o terra. As tensões de +15 V e -15 V foram obtidas diretamente da fonte, e a +9 V foi obtida através de um regulador com diodo Zener. As orientações para aterramento e a montagem em geral sob ponto de vista de minimiação de ruído foram seguidas segundo Ott [7].

IV.3 - Circuitos Auxiliares:

Devido às imprecisões inerentes à calibração de frequência ser feita por potencômetros e outras variantes, decidiu se projetar um circuito para fornecer frequências padrão para efetuar a calibração pelo software. Este circuito é mostrado no Diagrama 08. O funcionamemto é o seguinte: Obteve-se a frequência de referência através de um oscilador a cristal implementado com o integrado CD 4069, que é um inversor CMOS com 6 unidades. Este oscilador é composto por X2 e CI6. O resistor R56 é o resistor de ganho e R55 é para diminuir a tensão em cima do cristal. Os capacitores C34 e C35 são para eliminar as oscilações em alta frequência que tendem a aparecer devido ao alto ganho do circuito e às capacitâncias parasitas. Escolheu-se a frequência de 200 KHz para este oscilador por ser a mais apropriada dentro das possibilidades disponíveis.

O próximo passo é gerar as frequências de calibração dentro da faixa. Isto foi feito através de um divisor de frequência

implementado com o integrado CD 4040, que é um contador CMOS. Escolheu-se utilizar as seguintes frequências de saída:

- Frequência Inferior de Calibração:

Para esta frequência utilizou-se a saída Q7 (pino 4) que corresponde á frequência de entrada dividida por 2^7, assim:

$$FIC = \frac{200\ KHz}{128} = 1562,5\ Hz$$

- Frequência Superior de Calibração:

Para esta frequência utilizou-se a saída Q3 (pino 6) que corresponde ao dividendo de 2^3. Assim:

$$FSC = \frac{200\ KHz}{8} = 25\ KHz$$

O motivo destes valores é o seguinte:

Como foi utilzado um D/A de 8 bits, então consegue-se 2^8 níveis de tensão para VCOin . Como a Banda que se pretende analisar é de aproximadamente 80 KHz, temos uma resolução de 312,5 Hz por nível. Então temos:

- FIC:

$$RFIC = \frac{1562,5\ Hz}{312,5\ Hz} = 5$$

Logo, o número presente na entrada do D/A quando este sinal for sintonizado diminuido de 5 será o número o qual sintonizará a frequência de 312,5 Hz, que será o número correspondente a frequência inferior da escala. Assim:

$$N_{FIE} = N_{FIC} - RFIC$$

- FSC:

Como estes sinais são em onda quadrada 50-50, temos o terceiro harmônico em 75 KHz, e 80 KHz - 75 KHz = 5 KHz, então:

$$RFSC = \frac{5000 \text{ Hz}}{312,5 \text{ Hz}} = 16$$

Logo, após detetar este terceiro harmônico temos por analogia:

$$N_{FSE} = N_{FSC_{3^{o}\ harmonico}} + 16$$

que é o número correspondente a 80 KHz.

O software é quem se encarrega de fazer estas seleções através do controle das chaves que ligam estes sinais ao circuito, que são as chaves analógicas CI8A CI8C do integrado CD 4016, e o relé RL1. Os transistores Q7 e Q8 são para driver para o sinal de saída e o sinal de controle para acionar RL1 respectivamente. O potenciômetro POT9 é para regular o nível do sinal calibração, que é utilizado como referência de amplitude pelo software.

C A P Í T U L O V

- DESCRIÇÃO DO SOFTWARE:

Terminadas as fases de implementação do hardware, partiu-se para o projeto e implementação da versão básica do Software.
Como consequência da arquitetura utilizada, a rotina operacional do Software não exigiu complexidade. O funcionamento desta rotina é explicado a seguir.

Tomando como referência o pricípio de funcionamento dos SSA visto na seção II.2, vimos que cada valor de tensão na entrada do VCO (VCOin) corresponde a uma determinada frequência no eixo horizontal do CRT. Assim, o Software começa escrevendo o número correspondente à frequência inferior a ser analisada no barramento de dados do PC sob o endereço h302, que é o endereço do clock do primeiro latch do D/A ,que corresponde ao sinal $\overline{\text{DACK1}}$. Em segida, este número é transferido para para o segundo latch do D/A através da escrita do número h0 no endereço do clock do segundo latch do D/A, que corresponde ao sinal $\overline{\text{DACK2}}$ e é decodificado em h304. Temos assim o número devidamente bufferizado na entrada do D/A, o que deixa o barramento de dados livre para trafegarem os dados de saída do A/D. Este é o motivo da presença destes latches.
O próximo passo é ler o valor da eventual componente sintonizada através do A/D. Para isso, precisamos dizer ao A/D para inicializar o processo de conversão, já que este A/D é um Conversor por Aproximação Sucessiva, que tem boa referência em Polmman [8]. Isto é feito através de um pulso positivo no sinal CVST, que é o sinal do Convert Start do A/D. Este pulso é gerado

através da escrita da sequência 0, 1 e 0 no latch D2 do Diagrama 04. Este latch também é endereçado por $\overline{DACK2}$. Então, após o tempo de conversão do A/D, da ordem de 25 μs, temos o número na saída do A/D. Este número é bufferizado pelos latches E1 e E2 do Diagrama 05. Por ser um número de 12 bits, sendo que o barramento de dados tem 8 bits, primeiro são lidos os 4 bits mais significativos através da leitura de E2. Faz-se uma operação AND do número lido com h0 para zerar os bits menos significativos de E2, que foram deixados em "1" para melhor performance do chip. Em seguida este número é deslocado de 4 espaços para a esquerda e é armazenado em uma variável de memória. Então, lê-se os 8 bits restantes através de E1 e os soma a esta variável, obtendo assim o número de 12 bits desejado. As leituras de E1 e E2 são feitas através do endereço h306 e h307, que correspondem aos sinais $\overline{ADOEL}$ e $\overline{ADOEH}$.

Então, de posse do valor da eventual componente sintonizada, plotamos este valor na tela do monitor do PC e repetimos o processo até a frequência superior ser lida.

Esta rotina operacional fica dentro de um loop que pode ser interrompido para mudança de parâmetros através da escolha de opções em uma janela, como o Tempo de Varredura, que é implementado através de uma rotina de espera de tempo variável no final da leitura de cada componete, e o range observado, que é mudado através dos limites inferior e superior dos números de escrita no D/A.

O Software foi implementado utilizando a linguagem C, que se concluiu ser a mais apropriada. O Apêndice II mostra a listagem do Software e as telas de operação dos menus de escolha de opções para esta versão inicial.

C O N C L U S Ã O

Os resultados obtidos satisfizeram os propósitos iniciais, embora alguns pontos devam ser revistos para aumentar a performance do equipamento, dando margem a uma continuidade deste trabalho. O problema mais crítico que apareceu, é o fato de que o ruído na entrada do VCO modula este em frequência, e como o FPF é muito estreito, o valor lido pelo A/D em várias varreduras para uma mesma componente varia. Este problema pode ser minimizado blindando-se o circuito, mas um resultado satisfatório só é conseguido se for utilizado um VCO melhor e um design mais eficiente.

Estes pontos não poderam ser revistos por problemas de disponibilidade de componentes e tempo hábil para as modificações, e alguns são enumerados a seguir:

- Melhorar o FPB de entrada, utilizando um fitro ativo do tipo Butterworth de no mínimo terceira ordem.
- Utilizar um D/A de 12 bits. Este era o objetivo inicial desta versão e não foi alcançado devido a problemas em se conseguir o componente. O latches necessários para esta configuração já se encontram implementados nesta versão, e o processo operacional é análogo ao do A/D. Esta é a modificação mais importante a ser realizada, pois o número de 256 níveis não é suficiente para conseguir a resolução necessária.

A variação de range implementada nesta versão é apenas ilustrativa, já que a resolução de VCO_{in} não permanece costante pois não se utiliza todos os níveis do D/A fora do range máximo.

Para esta resuloção permanecer constante, seria necessário ter

um off-set para cada range escolhido através de chaveamento dos potenciômetros de off-set, utilizando assim o todos os níveis do D/A em cada range escolhido. Optou-se por não implementar este cicuito devido à espectativa de resultados medíocres comparados com o D/A de 12 bits.

Logo, este novo D/A também eliminaria o circuito de off-set implementado, pois o número de níveis é muito maior que o necessário, eliminando as imprecisões inerentes a este circuito.

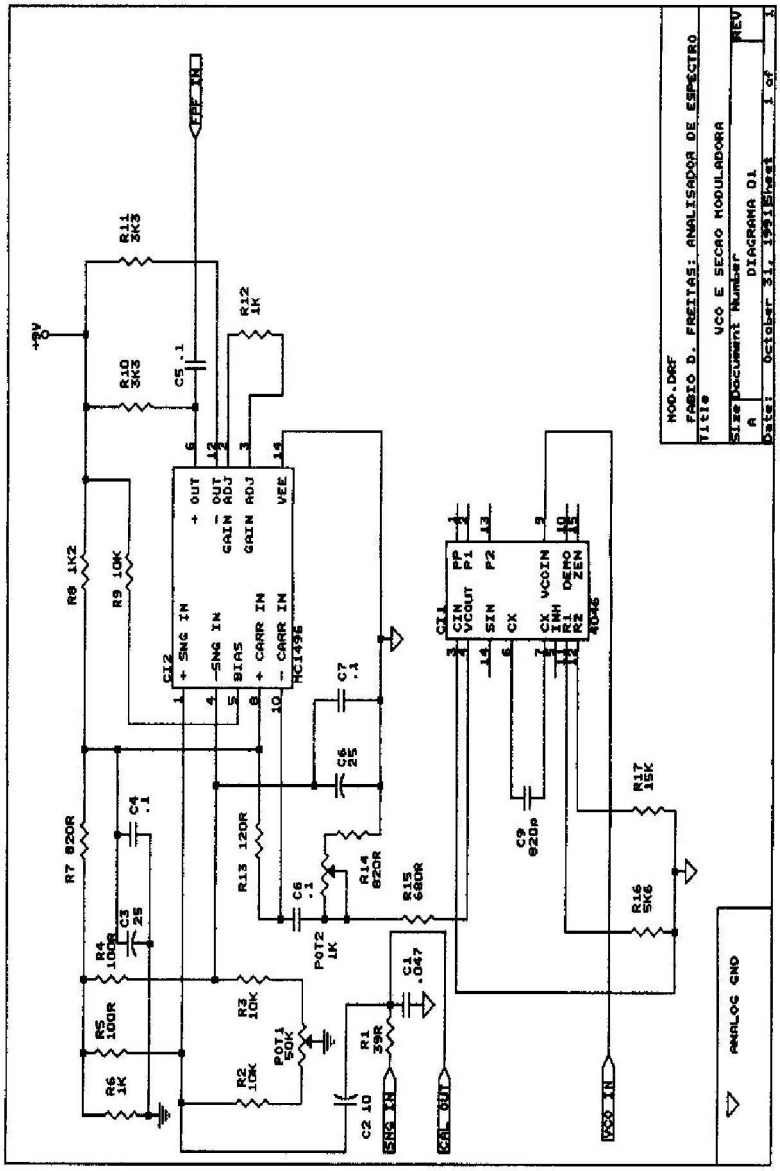
- Utilizar outros FPF de bandas um pouco maiores junto com o FPF implementado, para se variar a resolução. Utilizando assim bandas do FPF apropriadas para a resolução escolhida dentro do range selecionado para se ter maiores velocidades de varredura. Desta forma, o filtro a cristal seria utilizado quando se desejasse a máxima resolução dentro de um determinado range.

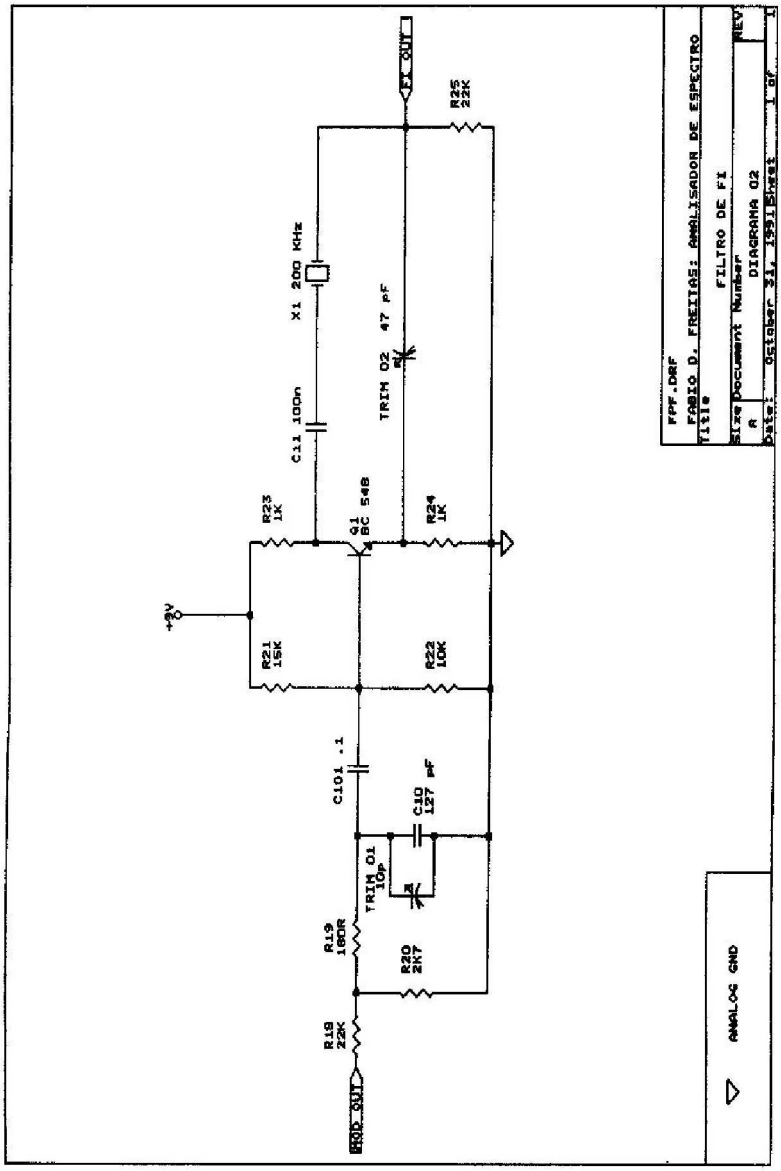
- Compactar o conversor DC-DC utilizando um transformador toroidal de pequenas dimensões, o que também não se conseguiu dispor.

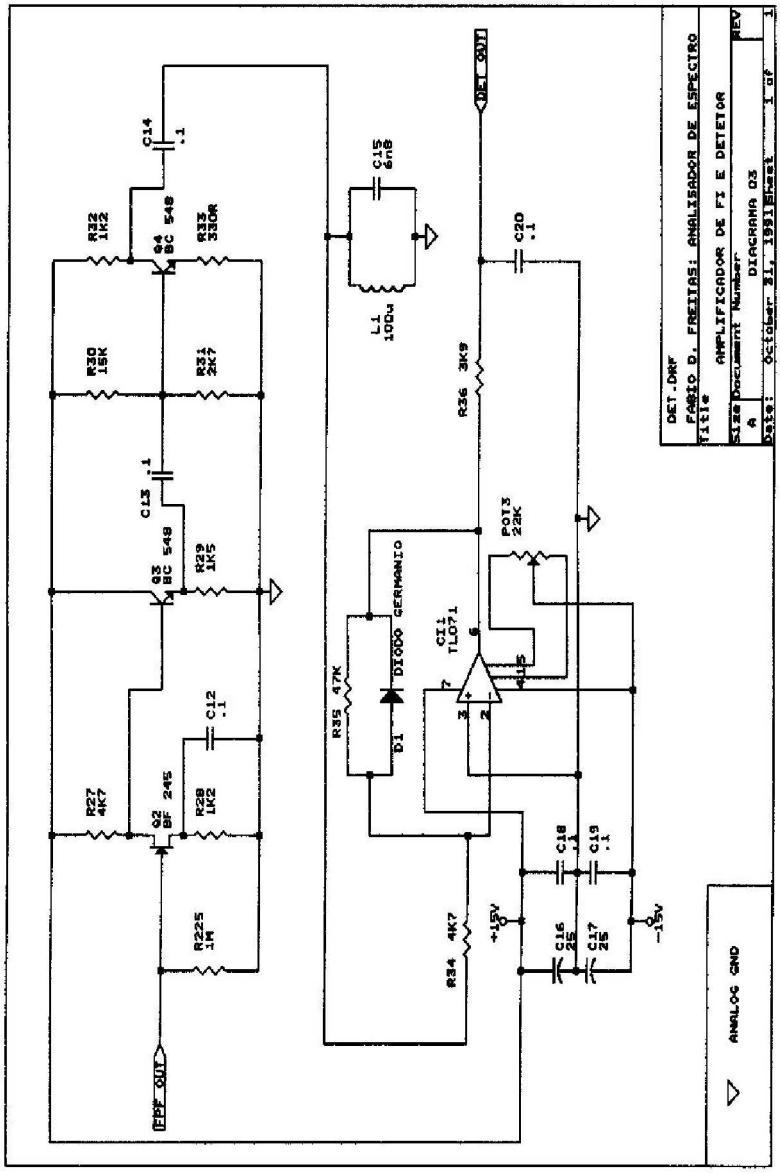
- E por último, como um estágio final deste projeto, fazer o lay-out de uma placa definitiva para ser produzida comercialmente, sendo implícita a adaptação e o melhoramento do Software para estas modificações.

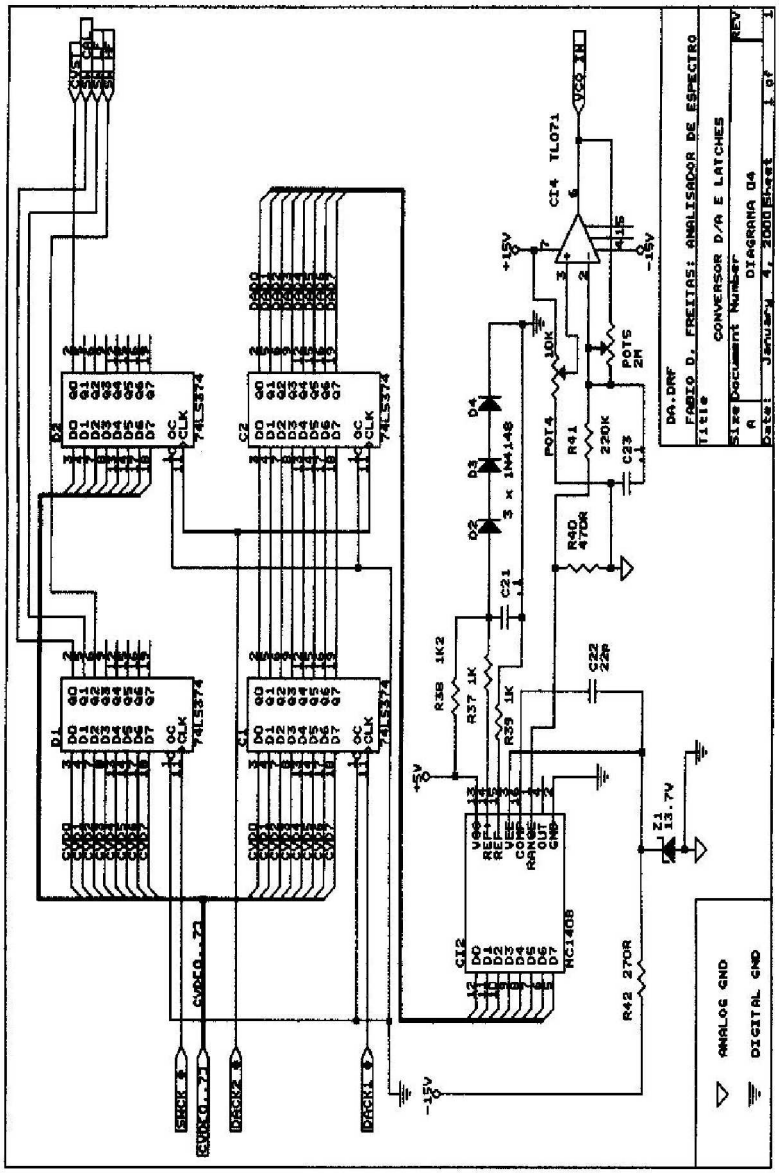
APENDICE I

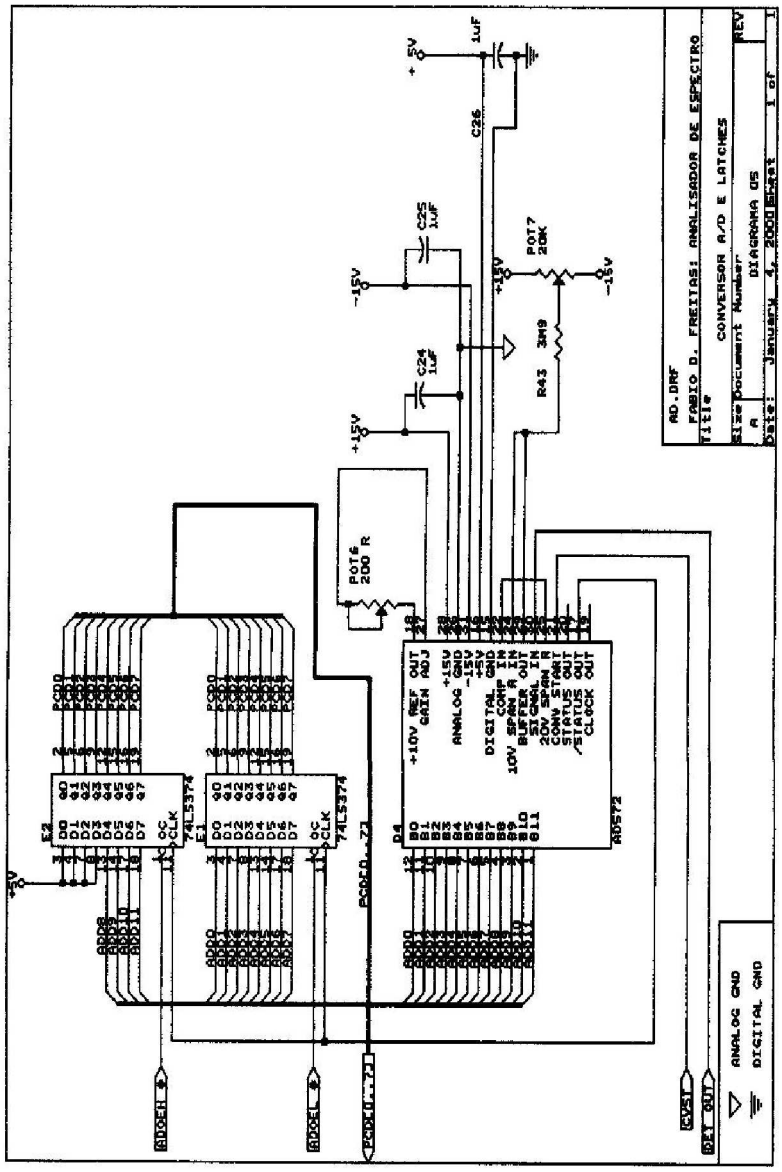
DIAGRAMAS ESQUEMARIOS

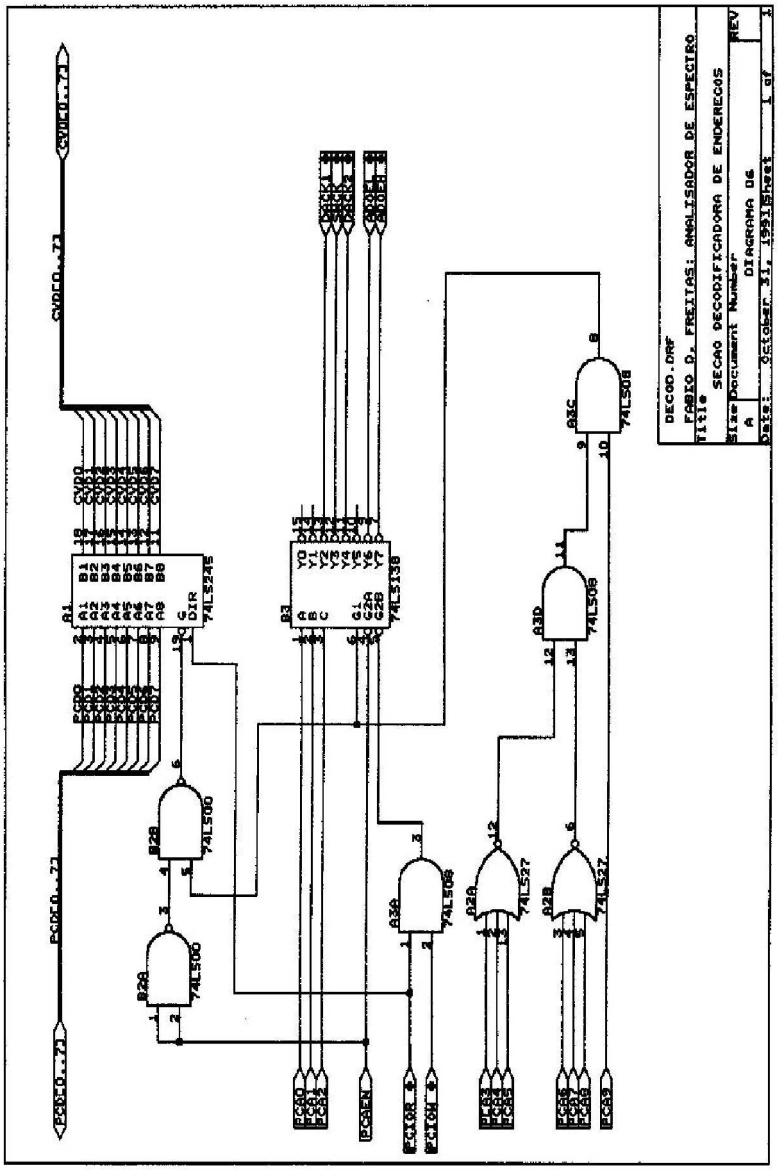

MOD.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title
VCO E SECAO MODULADORA
Size Document Number
A
DIAGRAMA D1
Date: October 31, 1991 Sheet 1 of 1
REV
1
+8V
RF IN
R11 3K3
R12 1K
R10 3K3
C5 .1
R8 1K2
R9 10K
IC2
+ SNG IN
-SNG IN
BIAS
+ CARR IN
- CARR IN
+ OUT
- OUT
GAIN ADJ
GAIN ADJ
VEE
RC1496
C7 .1
C6 25
R7 820R
C4 .1
C3 25
R4 100R
R5 100R
R6 1K
R3 10K
R2 10K
POT1 50K
R13 120R
C6 .1
POT2 1K
R14 820R
R15 680R
C1 .047
R1 39R
C2 10
SNG IN
CAR OUT
IC1
CIN
VCOUT
SIN
CX
CK
INH
R1
R2
PP
P1
P2
VCOIN
DEMO
ZEN
4046
C9 820p
R17 15K
R16 5K6
VCO IN
ANALOG GND

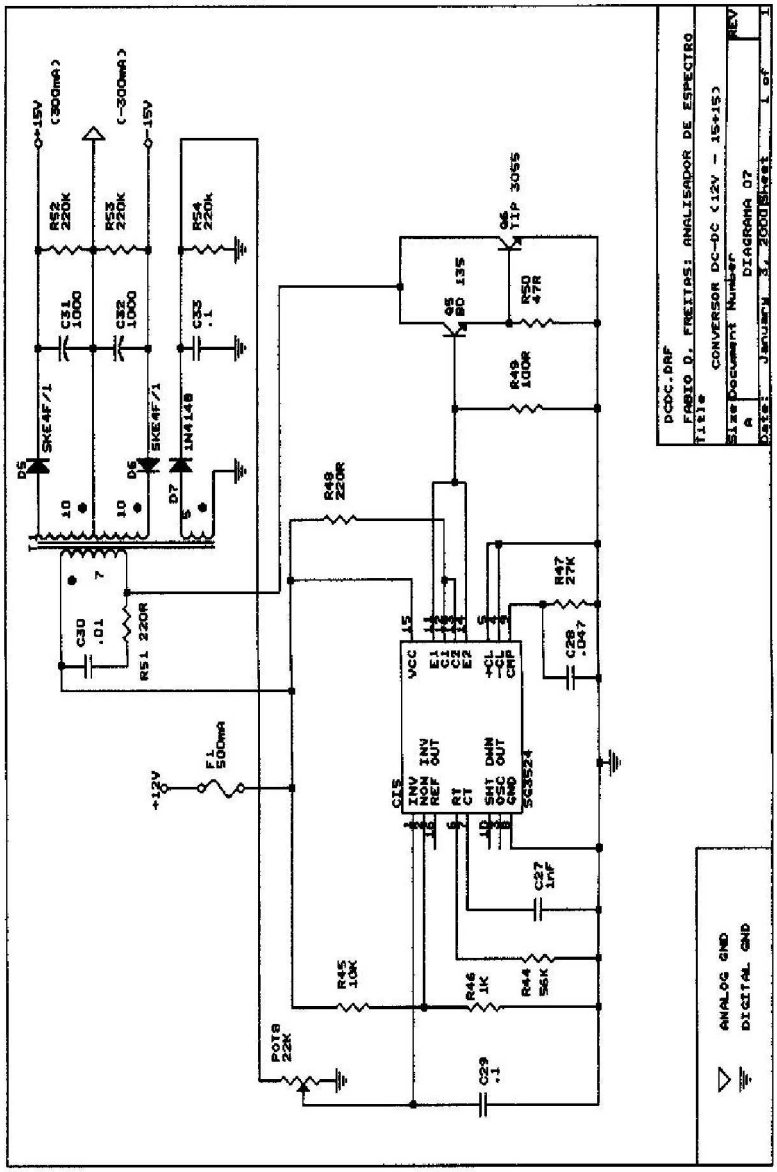

FPF.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title
FILTRO DE FI
Size
Document Number
REV
A
DIAGRAMA 02
Date: October 31, 1991
Sheet 1 of 1
FI OUT
R25 22K
X1 200 KHz
C11 100n
TRIM 02 47 pF
R23 1K
Q1 BC 548
R24 1K
+9V
R21 15K
R22 10K
C101 .1
C10 127 pF
TRIM 01 10p
R19 180R
R20 2K7
R18 22K
FROD OUT
ANALOG GND

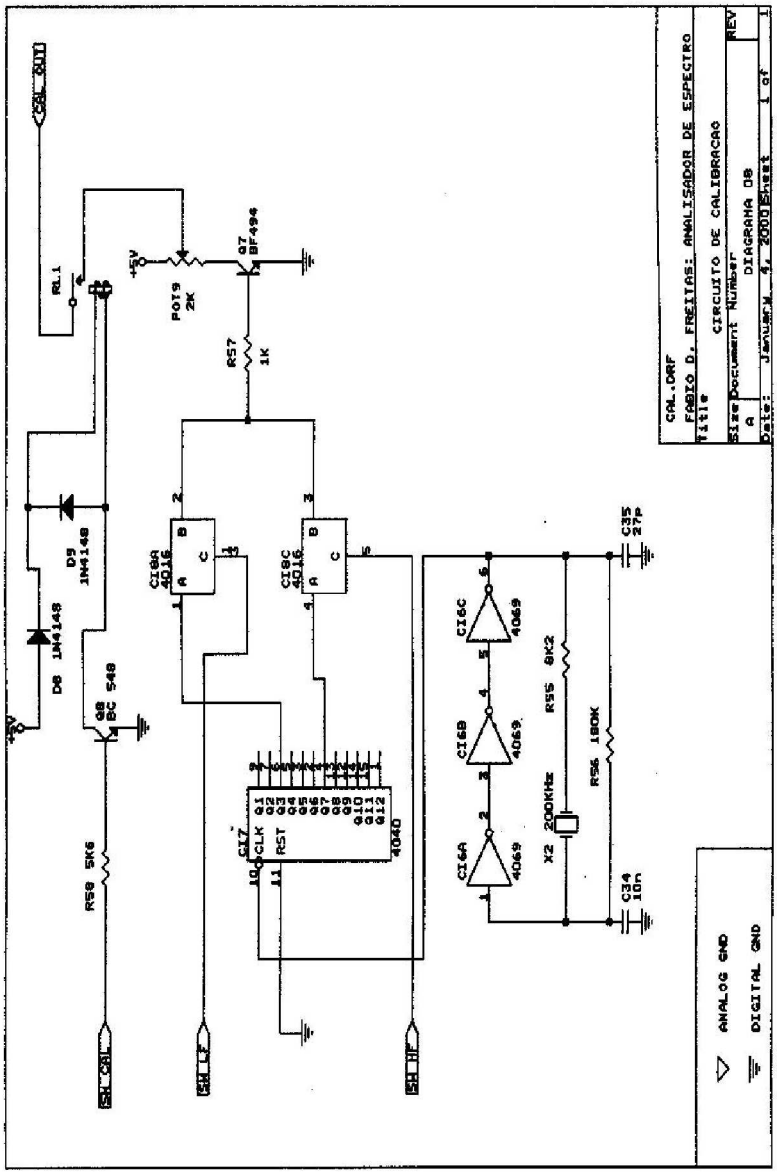

DET.DWF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title
AMPLIFICADOR DE FI E DETETOR
Size
A
Document Number
DIAGRAMA 03
REV
1
Date: October 21, 1991
Sheet 1 of
BPF OUT
DET OUT
ANALOG GND
R225 1M
R27 4K7
Q2 BF 245
R28 1K2
C12 .1
R29 1K5
Q3 BC 548
C13 .1
R30 15K
R31 2K7
R32 1K2
Q4 BC 548
R33 330R
C14 .1
C15 6n8
L1 100u
R35 47K
D1 DIODO GERMANIO
CI1 TL071
POT3 22K
R36 3K9
C20 .1
R34 4K7
+15V
-15V
C16 25
C17 25
C18 .1
C19 .1


DA.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title
CONVERSOR D/A E LATCHES
Size
Document Number
REV
A
DIAGRAMA 04
Date: January 4, 2000
Sheet 1 of 1
D1
D2
C1
C2
74LS374
CI2
MC1408
CI4 TL071
VCC_IN
CVDC0..7
DACK1
DACK2
DACK3
+15V
-15V
+5V
POT4
POTS
R41
220K
R40
470R
R42 270R
R38 1K2
R37 1K
R39 1K
C21
C22
C23
D2
D3
D4
3 x 1N4148
Z1
13.7V
ANALOG GND
DIGITAL GND


AD.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title
CONVERSOR A/D E LATCHES
Size
Document Number
REV
A
DIAGRAMA 05
Date: January 4, 2000
Sheet 1 of 1
+5V
1uF
C26
C25
1uF
C24
1uF
-15V
+15V
POT7
20K
R43
3K9
POT6
200 R
E2
E1
74LS374
D4
AD572
+10V REF OUT
GAIN ADJ
ANALOG +15V
GND
-15V
+5V
GND
DIGITAL
COMP IN
POWER
10V SPAN IN
BUFFER OUT
SIGNAL IN
20V SPAN
START
STATUS OUT
/STATUS OUT
CLOCK OUT
ADDR
ADDR0
ADDR1
ADDR2
ADDR3
ADDR4
ADDR5
ADDR6
ADDR7
ADDR8
ADDR9
ADDR10
ADDR11
ADDEH
ADDEL
PCDFD..7]
CVST
DET OUT
ANALOG GND
DIGITAL GND


DECOD.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title SECÃO DECODIFICADORA DE ENDEREÇOS
Size Document Number REV
A DIAGRAMA D6
Date: October 31, 1991 Sheet 1 of 1
CVDEO..7
CVDEO..7
PCDEO..7
PCDEO..7
B1
74LS245
B3
74LS138
B2A
74LS00
B2B
74LS00
B3A
74LS08
B3B
74LS27
B3C
74LS08
B3D
74LS08
B3E
74LS27
PCEN
PCIOR
PCIOW


DCDC.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title
CONVERSOR DC-DC (12V - 15+15)
Size
A
Document Number
DIAGRAMA 07
REV
1
Date: January 31, 2000
Sheet 1 of 1
+15V
(300mA)
(-300mA)
-15V
R52 220K
R53 220K
R54 220K
C31 1000
C32 1000
C33 .1
D5 SKE4F/1
D6 SKE4F/1
D7 1N4148
C30 .01
R51 220R
+12V
F1 500mA
POT8 22K
R45 10K
R46 1K
R44 56K
C29 .1
C27 1nF
R48 220R
R49 100R
R50 47R
Q5 BD 135
Q6 TIP 3055
R47 27K
C28 .047
U15 SG3524
VCC
E1
C1
C2
E2
+CL
-CL
CMP
INV
NON INV
OUT
REF
RT
CT
SHT DWN
OSC OUT
GND
ANALOG GND
DIGITAL GND


CAL.DRF
FABIO D. FREITAS: ANALISADOR DE ESPECTRO
Title CIRCUITO DE CALIBRACAO
Size Document Number REV
A DIAGRAMA DB
Date: January 4, 2000 Sheet 1 of 1
CON_BOT
RL1
+5V
POT9 2K
Q7 BF494
R57 1K
D8 1N4148
D9 1N4148
Q8 BC 548
R58 5K6
CI8A 4016
CI8C 4016
CI7 4040
CLK Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 RST
CI6A 4069
CI6B 4069
CI6C 4069
R55 8K2
X2 200KHz
R56 180K
C34 10n
C35 27p
SH_CAL
SH_LF
SH_HF
ANALOG GND
DIGITAL GND

APÊNDICE II

LISTAGEM DO SOFTWARE

```
/*********************************************************************
*******
*                SOFTWARE DA PLACA ANALIZADORA DE ESPECTRO
      *
*
      *
*                 Autor: Fabio Daros de Freitas
      *
*                 Data : Novembro de 1991
      *
*********************************************************************
******/

#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#define        UP             72
#define        DOWN           80
#define        LEFT           75
#define        RIGHT          77
#define        PGUP           73
#define        PGDN           81
#define        HOME           71
#define        END            79
#define        DEL            83
#define        ENTER          13
#define        ENDBUFFER      0x302
#define        CKTRANSF       0x304
#define        INADH          0x307
#define        INADL          0x306
#define        CKTRCAL        0x303
#define        GANHO          0.15
#define        NUMY           8
#define        XETELA         70
#define        XDTELA         582
#define        YTTELA         15
#define        YFTELA         159


void inicializa ( void ) ;
void moldura ( void ) ;
void opcao ( void ) ;
void janelahstop ( void ) ;
void janelahstart ( void ) ;
```

```
void janelavr ( void ) ;
void janelavs ( void ) ;
void movimentoh ( void ) ;
void movimentovr ( void );
void movimentovs ( void );
void analisa ( int, int ) ;
void testa_stop ( void ) ;
void saida ( void ) ;
void Wait ( void) ;
int  calibral (void) ;
int  calibrah (void) ;


int          localh, localvr, localvs, tecla, flesc, CALAMP, nfl, nfh, ste
px ;
int          valory [ 500 ], ypla [500], yplpa [500] ;
char         status[10], range[10], sweep[10] ;
void         *ponteirojhl, *ponteirojvl, *ponteiroopch, *ponteiroopcv ;
void         *ponteiroi, *ponteirojvinv, *ponteiroraia, *ponteirotl ;
unsigned     num_bytesjhl, num_bytesjvl, num_bytesopch, num_bytesopcv ;
unsigned     num_bytesraia, num_bytestl ;

int num [300] ;
int ypl [300] ;

void inicializa ( void )

{
        int     i, j, graphdriver = DETECT, graphmode ;


        for ( j = 0; j < 300; j++ )
        {
                ypl [j] = YFTELA ;
                num [j] = j ;
        }
        nfl = 1 ;
        nfh = 256 ;
        strcpy (status,"stop") ;
        strcpy (range, "full") ;
        localh = localvr = localvs = 1 ;
        for ( i = 0; i < 500; i ++ )
                valory [ i ] = ypla [i] = yplpa [i] = 20 ;
        flesc = 0 ;
        randomize () ;
        initgraph ( &graphdriver, &graphmode, "c:\\fabio\\tc\\bgi\\" ) ;

        num_bytesjhl = imagesize ( 1, 180, 638, 198 ) ;
        ponteirojhl = farmalloc ( num_bytesjhl ) ;      /* JANELA HOR. L
IMPA */
        getimage ( 1, 180, 638, 198, ponteirojhl ) ;
```

```
        num_bytesjvl = imagesize ( 443, 115, 623, 193 ) ;
        ponteirojvl = farmalloc ( num_bytesjvl ) ;       /* JANELA VER. L
IMPA */
        getimage ( 443, 115, 623, 193, ponteirojvl ) ;

        num_bytesopch = imagesize ( 20, 183, 100, 195 ) ;
        ponteiroopch = farmalloc ( num_bytesopch ) ;      /* JAN. OPC. HO
R. */
        getimage ( 20, 183, 100, 195, ponteiroopch ) ;

        num_bytesopcv = imagesize ( 452, 126, 624, 139 ) ;
        ponteiroopcv = farmalloc ( num_bytesopcv ) ;
        getimage ( 452, 126, 624, 139, ponteiroopcv ) ;   /* JAN. OPC. V
ER. */

        ponteiroi = farmalloc ( num_bytesjvl ) ;      /* IMAGEM GUARDADA
*/
        getimage ( 443, 115, 623, 193, ponteirojvl ) ;

        ponteirojvinv = farmalloc ( num_bytesjvl ) ;      /* JAN. VERT. I
NV. */
        getimage ( 443, 115, 623, 193, ponteirojvinv ) ;

        num_bytesraia = imagesize ( 80, 16, 82, 158 ) ;
        ponteiroraia  = farmalloc ( num_bytesraia ) ;

        num_bytestl = imagesize ( 71, 16, 629, 158 ) ;
        ponteirotl = farmalloc ( num_bytestl ) ;
        getimage ( 71, 16, 629, 158, ponteirotl ) ;

        strcpy ( range, "Range1" ) ;
        strcpy ( sweep, "Sweep1" ) ;
}


void moldura ()
{


        int  x, y, i, j, passox, passoy, altura, largura ;
        char *texto [ 10 ], *eixoy [ 10 ], *eixox [ 11 ]  ;


        rectangle ( 0, 0, 639, 199 ) ;
        line ( 0, 179, 639, 179 ) ;
        rectangle (XETELA, YTTELA, XDTELA, YFTELA );
        passoy = ( YFTELA - YTTELA ) / 9 ;

        eixoy [ 0 ] = "   0" ;
        eixoy [ 1 ] = "-  4" ;
        eixoy [ 2 ] = "-  8" ;
        eixoy [ 3 ] = "- 12" ;
```

```
        eixoy [ 4 ] = "- 16" ;
        eixoy [ 5 ] = "- 20" ;
        eixoy [ 6 ] = "- 24" ;
        eixoy [ 7 ] = "- 28" ;
        eixoy [ 8 ] = "- 32" ;
        eixoy [ 9 ] = "- 36" ;
        setlinestyle ( USERBIT_LINE, 0x8888, NORM_WIDTH ) ;
        settextstyle ( SMALL_FONT, HORIZ_DIR, 0 ) ;
        altura = textheight ( "XXXX" ) ;
        largura = textwidth ( "XXXX" ) ;
        for ( i = YTTELA, j = 0; i <= YFTELA, j < 10; i += passoy, j++ )
        {
                outtextxy ( XETELA - largura - 5, i - altura / 2, eixoy
[j] );
        }


        eixox [ 0 ] = "0,3";
        eixox [ 1 ] = "10" ;
        eixox [ 2 ] = "20" ;
        eixox [ 3 ] = "30" ;
        eixox [ 4 ] = "40" ;
        eixox [ 5 ] = "50" ;
        eixox [ 6 ] = "60" ;
        eixox [ 7 ] = "70" ;
        eixox [ 8 ] = "80" ;
        eixox [ 9 ] = " 4" ;

        setlinestyle ( USERBIT_LINE, 0x8888, NORM_WIDTH ) ;
        settextstyle ( SMALL_FONT, HORIZ_DIR, 0 ) ;
        altura = textheight ( "XXXX" ) ;
        largura = textwidth ( "XXXX" ) ;
        passox = (XDTELA - XETELA) / 8 ;

        for ( i = XETELA, j = 0; i <= XDTELA, j < 10; i += passox, j++ )
        {
                outtextxy ( i - largura/2 + 10, YFTELA + 5, eixox [j] )
;
        }





        texto [ 1 ] = "ANALISADOR DE ESPECTRO -  versao 1.0  FDF" ;

        largura = textwidth ( texto [ 1 ] ) ;
        altura = textheight ( texto [ 1 ] ) ;
        x = XETELA + ( XDTELA - XETELA - largura ) / 2 ;
        outtextxy ( x, YTTELA - altura - 3, texto [ 1 ] ) ;
```

```
        outtextxy ( 10, 5, "dB" ) ;
        altura = textheight ( "Hz" ) ;
        y = 179 - (179 - YFTELA - altura - 1 ) ;
        outtextxy ( 610 , y, "KHz" ) ;
        getimage ( 80, 15, 82, 158, ponteiroraia ) ;

        settextstyle ( DEFAULT_FONT, HORIZ_DIR, 1 ) ;

        setlinestyle ( SOLID_LINE, 0xffff, NORM_WIDTH ) ;

}



void janelahstop ( void )

{
        int         altura, largura, xe, xd, yt, yf, x, y, i ;
        char        *tituloh[4] ;

        putimage ( 1, 180, ponteirojhl, COPY_PUT ) ;
        setlinestyle ( USERBIT_LINE, 0xffff, NORM_WIDTH ) ;
        i = 0 ;
        xe = 17 ;
        xd = 103 ;
        yt = 181 ;
        yf = 197 ;
        for ( i = 1 ; i <= 4 ; i ++ )
                {
                rectangle ( xe, yt, xd, yf ) ;
                rectangle ( xe + 3, yt + 2, xd - 3, yf - 2 ) ;
                xe += 100 ;
                xd += 100 ;
                }

        tituloh [ 0 ] = "Start" ;
        tituloh [ 1 ] = "Range" ;
        tituloh [ 2 ] = "Sweep" ;
        tituloh [ 3 ] = "Quit" ;
        xe = 20 ;
        xd = 100 ;
        yt = 183 ;
        yf = 195 ;
        for ( i = 0 ; i <= 3 ; i ++ )
                {
                setviewport ( xe, yt, xd, yf, 0 ) ;
                altura = textheight ( tituloh [i] ) ;
                largura = textwidth ( tituloh [i] ) ;
                x = ( 100 - 20  - largura ) / 2 ;
                y = ( 195 - 183 - altura ) / 2 ;
                outtextxy ( x, y, tituloh [i] ) ;
```

```
                xe += 100 ;
                xd += 100 ;

                }

        setviewport ( 0, 0, 639, 199, 0 ) ;
        getimage ( 20, 183, 100, 195, ponteiroopch ) ;
        putimage ( 20, 183, ponteiroopch, NOT_PUT ) ;


}


void janelahstart ( void )

{

        int       altura, largura, xe, xd, yt, yf, x, y ;



        setlinestyle ( USERBIT_LINE, 0xffff, NORM_WIDTH ) ;
        putimage ( 1, 180, ponteirojhl, COPY_PUT ) ;

        xe = 17 ;
        xd = 103 ;
        yt = 181 ;
        yf = 197 ;
        rectangle ( xe, yt, xd, yf ) ;
        rectangle ( xe + 3, yt + 2, xd - 3, yf - 2 ) ;
        xe = 20 ;
        xd = 100 ;
        yt = 183 ;
        yf = 195 ;
        setviewport ( xe, yt, xd, yf, 0 ) ;
        altura = textheight ( "Stop" ) ;
        largura = textwidth ( "Stop" ) ;
        x = ( 100 - 20  - largura ) / 2 ;
        y = ( 195 - 183 - altura ) / 2 ;
        outtextxy ( x, y, "Stop" ) ;
        setviewport ( 0, 0, 639, 199, 0 ) ;
        getimage ( xe, yt, xd, yf, ponteiroopch ) ;
        putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
        settextstyle ( SMALL_FONT, HORIZ_DIR, 0 ) ;
        outtextxy ( 450, 188, "Range : " ) ;
        outtextxy ( 500, 188, range ) ;
        outtextxy ( 450, 179, "Sweep Time : " ) ;
        outtextxy ( 530, 179, sweep ) ;
        settextstyle ( DEFAULT_FONT, HORIZ_DIR, 0 ) ;


}
```

```
void janelavr ()

#define    LARGURA   180
#define    ALTURA    78

{
        int   x, y, i, largura, altura, xe, xd, yt, yf ;
        char  *titulov [4] ;


        xe = 443 ;
        xd = xe + LARGURA ;
        yt = 115 ;
        yf = yt + ALTURA ;
        setlinestyle ( USERBIT_LINE, 0xffff, NORM_WIDTH ) ;
        getimage ( xe, yt, xd, yf, ponteiroi ) ;
        putimage ( xe, yt, ponteirojvl, COPY_PUT ) ;
        rectangle ( xe + 3, yt + 4, xd - 3, yf - 4 ) ;
        rectangle ( xe + 6, yt + 6, xd - 6, yf - 6 ) ;
        titulov [ 0 ] = "20 div" ;
        titulov [ 1 ] = "80 div" ;
        titulov [ 2 ] = "150 div" ;
        titulov [ 3 ] = "256 div" ;

        altura = ( yf - yt ) / 5 ;
        y = yt - 2 ;
        for ( i = 0 ; i <= 3 ; i ++ )
                {
                largura = textwidth ( titulov [ i ] ) ;
                x = 448 + ( 628 - 448 - largura ) / 2 ;
                y += altura ;
                outtextxy ( x, y, titulov [ i ] ) ;

                }



        rectangle ( xe, yt, xd, yf ) ;
        getimage ( xe, yt, xd, yf, ponteirojvinv ) ;
        putimage ( xe, yt, ponteirojvinv, NOT_PUT ) ;

        yt = 121 + 3 + 3 + ( localvr -1 ) * 15 ;
        yf = yt + 9 ;
```

```
        getimage ( xe + 10, yt, xd - 10, yf, ponteiroopcv ) ;
        putimage ( xe + 10, yt, ponteiroopcv, NOT_PUT ) ;

        xe = 453 ;
}



void janelavs ()


#define     LARGURA   180
#define     ALTURA    78

{
        int   x, y, i, largura, altura, xe, xd, yt, yf ;
        char  *titulov [4] ;


        xe = 443 ;
        xd = xe + LARGURA ;
        yt = 115 ;
        yf = yt + ALTURA ;
        setlinestyle ( USERBIT_LINE, 0xffff, NORM_WIDTH ) ;
        getimage ( xe, yt, xd, yf, ponteiroi ) ;
        putimage ( xe, yt, ponteirojvl, COPY_PUT ) ;
        rectangle ( xe + 3, yt + 4, xd - 3, yf - 4 ) ;
        rectangle ( xe + 6, yt + 6, xd - 6, yf - 6 ) ;
        titulov [ 0 ] = "Sweep1" ;
        titulov [ 1 ] = "Sweep2" ;
        titulov [ 2 ] = "Sweep3" ;
        titulov [ 3 ] = "Sweep4" ;

        altura = ( yf - yt ) / 5 ;
        y = yt - 2 ;
        for ( i = 0 ; i <= 3 ; i ++ )
                {
                largura = textwidth ( titulov [ i ] ) ;
                x = 448 + ( 628 - 448 - largura ) / 2 ;
                y += altura ;
                outtextxy ( x, y, titulov [ i ] ) ;

                }



        rectangle ( xe, yt, xd, yf ) ;
        getimage ( xe, yt, xd, yf, ponteirojvinv ) ;
        putimage ( xe, yt, ponteirojvinv, NOT_PUT ) ;
```

```
        yt = 121 + 3 + 3 + ( localvs -1 ) * 15 ;
        yf = yt + 9 ;

        getimage ( xe + 10, yt, xd - 10, yf, ponteiroopcv ) ;
        putimage ( xe + 10, yt, ponteiroopcv, NOT_PUT ) ;

        xe = 453 ;
}


void movimentoh ( void )

#define PASSOH      100
#define LARGURAOH    80
#define ALTURAOH     12
{
        int    xe, yt, xd, yf  ;
        char   caracter ;

        xe = 20 + ( localh - 1 ) * PASSOH ;
        xd = xe + LARGURAOH ;
        yt = 183 ;
        yf = yt + ALTURAOH ;

        caracter = getch () ;

        while ( ( (caracter==ENTER) || (caracter==0) ) && (strcmp (statu
s,"stop"
        {
         if((caracter==ENTER)&&((localh==1)||(localh==4)||(localh==2)||(
localh==
                {
                switch ( localh )
                        {
                        case 1 :
                                strcpy (status,"start") ;
                                break ;

                        case 2 :
                                strcpy (status,"range") ;
                                break ;

                        case 3 :
                                strcpy (status, "sweep") ;
                                break ;

                        case 4 :
                                strcpy (status,"quit") ;
                                break ;
```

```
                    }
            }
        else

            switch ( caracter = getch () )
            {
            case UP:
                    break;

            case DOWN:

                    break;

            case RIGHT :

              switch ( localh )
                {
                case 1 :


                  putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                  xe += PASSOH ;
                  xd += PASSOH ;
                  getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                  putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
                  localh = 2 ;
                  break ;

                case 2 :

                  putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                  xe += PASSOH ;
                  xd += PASSOH ;
                  getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                  putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
                  localh = 3 ;
                  break ;


                case 3 :

                  putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                  xe += PASSOH ;
                  xd += PASSOH ;
                  getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                  putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
                  localh = 4 ;
                  break ;
```

```
                    case 4 :

                     putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                     localh = 1 ;
                     xe = 20 + ( localh - 1 ) * PASSOH ;
                     xd = xe + LARGURAOH ;
                     getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                     putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;

                     break ;


                    }
                    break ;

                case LEFT :

                  switch ( localh )
                    {
                    case 1 :

                     putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                     localh = 4 ;
                     xe = 20 + ( localh - 1 ) * PASSOH ;
                     xe = 320 ;
                     xd = xe + LARGURAOH ;
                     getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                     putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;

                     break ;


                    case 2 :

                     putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                     xe -= PASSOH ;
                     xd -= PASSOH ;
                     getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                     putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
                     localh = 1 ;
                     break ;


                    case 3 :
                     putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                     xe -= PASSOH ;
                     xd -= PASSOH ;
                     getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                     putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
```

```
                    localh = 2 ;
                    break ;


                   case 4 :

                    putimage ( xe, yt, ponteiroopch, COPY_PUT ) ;
                    xe -= PASSOH ;
                    xd -= PASSOH ;
                    getimage ( xe, yt, xd, yf, ponteiroopch ) ;
                    putimage ( xe, yt, ponteiroopch, NOT_PUT ) ;
                    localh = 3 ;
                    break ;

                   }
                   break ;


              }


        }


}


void movimentovr ( void )


#define   PASSOV      15
#define   ALTURAOV     9
#define   LARGURAOV  160


{
        char      caracter ;
        int       xe, xd, yt, yf, seta_saida, localvaux, fltl ;

        localvaux = localvr ;
        xe = 453 ;
        xd = xe + LARGURAOV ;
        yt = 121 + 3 + 3 + ( localvaux -1 ) * PASSOV ;
```

```
        yf = yt + ALTURAOV ;

        fltl = 0 ;
        seta_saida = 0 ;
        caracter = getch () ;
        while (((caracter != ENTER) || (caracter == 0)) && (seta_saida==
0) )
        {


                switch (caracter = getch() )
                  {

                  case ENTER :

                   seta_saida = 1 ;
                   if ( localvaux != localvr )
                        fltl = 1 ;
                   localvr = localvaux ;
                   break ;

                  case RIGHT :

                   putimage ( 120, 183, ponteiroopch, COPY_PUT ) ;
                   getimage ( 220, 183, 300, 195, ponteiroopch ) ;
                   putimage ( 220, 183, ponteiroopch, NOT_PUT ) ;
                   localh = 3 ;
                   seta_saida = 1 ;
                   break ;

                  case LEFT :


                   putimage ( 120, 183, ponteiroopch, COPY_PUT ) ;
                   getimage ( 20, 183, 100, 195, ponteiroopch ) ;
                   putimage ( 20, 183, ponteiroopch, NOT_PUT ) ;
                   localh = 1 ;
                   seta_saida = 1 ;
                   break ;

                  case UP :
                   switch ( localvaux )
                    {
                    case 1:

                     putimage( xe, yt, ponteiroopcv, COPY_PUT);
                     localvaux = 4 ;
                     strcpy ( range, "Range4") ;
                     yt = 121 + 3 + 3 + ( localvaux -1 ) * PASSOV ;
                     yf = yt + ALTURAOV ;
                     getimage( xe, yt, xd, yf ,ponteiroopcv);
```

```
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                break;

               case 2:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                yt -= PASSOV ;
                yf -= PASSOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                localvaux = 1;
                strcpy ( range, "Range1") ;
                break;

               case 3:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                yt -= PASSOV ;
                yf -= PASSOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                localvaux = 2;
                strcpy ( range, "Range2") ;
                break;

               case 4:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                yt -= PASSOV ;
                yf -= PASSOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                localvaux = 3;
                strcpy ( range, "Range3") ;
                break;


               }
              break ;

             case DOWN:


              switch ( localvaux )

               {
               case 1:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                yt += PASSOV ;
                yf += PASSOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
```

```
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                localvaux = 2;
                strcpy ( range, "Range2") ;
                break;

               case 2:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                yt += PASSOV ;
                yf += PASSOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                localvaux = 3;
                strcpy ( range, "Range3") ;
                break;


               case 3:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                yt += PASSOV ;
                yf += PASSOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
                putimage( xe, yt, ponteiroopcv, NOT_PUT);
                localvaux = 4;
                strcpy ( range, "Range4") ;
                break;

               case 4:

                putimage( xe , yt, ponteiroopcv, COPY_PUT);
                localvaux = 1;
                strcpy ( range, "Range1") ;
                yt = 121 + 3 + 3 + ( localvaux - 1 ) * PASSOV ;
                yf = yt + ALTURAOV ;
                getimage( xe, yt, xd, yf, ponteiroopcv);
                putimage( xe, yt, ponteiroopcv, NOT_PUT);

                break;

               }

              break ;

             }

        }

        putimage ( 443, 115, ponteirojvl, COPY_PUT ) ;
        putimage ( 443, 115, ponteiroi, COPY_PUT ) ;
```

```
                if ( fltl == 1 )
                        putimage ( 71, 16, ponteirotl, COPY_PUT ) ;
}


void movimentovs ( void )


#define   PASSOV      15
#define   ALTURAOV     9
#define   LARGURAOV  160


{
        char     caracter ;
        int      xe, xd, yt, yf, seta_saida, localvaux ;

        localvaux = localvs ;
        xe = 453 ;
        xd = xe + LARGURAOV ;
        yt = 121 + 3 + 3 + ( localvaux -1 ) * PASSOV ;
        yf = yt + ALTURAOV ;


        seta_saida = 0 ;
        caracter = getch () ;
        while (((caracter != ENTER) || (caracter == 0)) && (seta_saida==
0) )
        {


                switch (caracter = getch() )
                  {

                  case ENTER :

                   seta_saida = 1 ;
                   localvs = localvaux ;
                   break ;

                  case RIGHT :

                   putimage ( 220, 183, ponteiroopch, COPY_PUT ) ;
                   getimage ( 320, 183, 400, 195, ponteiroopch ) ;
                   putimage ( 320, 183, ponteiroopch, NOT_PUT ) ;
                   localh = 4 ;
                   seta_saida = 1 ;
                   break ;
```

```
case LEFT :

 putimage ( 220, 183, ponteiroopch, COPY_PUT ) ;
 getimage ( 120, 183, 200, 195, ponteiroopch ) ;
 putimage ( 120, 183, ponteiroopch, NOT_PUT ) ;
 localh = 2 ;
 seta_saida = 1 ;
 break ;

case UP :
 switch( localvaux )
  {
  case 1:

   putimage( xe, yt, ponteiroopcv, COPY_PUT);
   localvaux = 4 ;
   strcpy ( sweep, "Sweep4" ) ;
   yt = 121 + 3 + 3 + ( localvaux -1 ) * PASSOV ;
   yf = yt + ALTURAOV ;
   getimage( xe, yt, xd, yf ,ponteiroopcv);
   putimage( xe, yt, ponteiroopcv, NOT_PUT);
   break;

  case 2:

   putimage( xe , yt, ponteiroopcv, COPY_PUT);
   yt -= PASSOV ;
   yf -= PASSOV ;
   getimage( xe, yt, xd, yf, ponteiroopcv);
   putimage( xe, yt, ponteiroopcv, NOT_PUT);
   localvaux = 1;
   strcpy ( sweep, "Sweep1" ) ;
   break;

  case 3:

   putimage( xe , yt, ponteiroopcv, COPY_PUT);
   yt -= PASSOV ;
   yf -= PASSOV ;
   getimage( xe, yt, xd, yf, ponteiroopcv);
   putimage( xe, yt, ponteiroopcv, NOT_PUT);
   localvaux = 2;
   strcpy ( sweep, "Sweep2" ) ;
   break;

  case 4:

   putimage( xe , yt, ponteiroopcv, COPY_PUT);
   yt -= PASSOV ;
   yf -= PASSOV ;
```

```
                    getimage( xe, yt, xd, yf, ponteiroopcv);
                    putimage( xe, yt, ponteiroopcv, NOT_PUT);
                    localvaux = 3;
                    strcpy ( sweep, "Sweep3" ) ;
                    break;


                  }
                 break ;

                case DOWN:

                 switch ( localvaux )

                  {
                  case 1:

                    putimage( xe , yt, ponteiroopcv, COPY_PUT);
                    yt += PASSOV ;
                    yf += PASSOV ;
                    getimage( xe, yt, xd, yf, ponteiroopcv);
                    putimage( xe, yt, ponteiroopcv, NOT_PUT);
                    localvaux = 2;
                    strcpy ( sweep, "Sweep2" ) ;
                    break;

                  case 2:

                    putimage( xe , yt, ponteiroopcv, COPY_PUT);
                    yt += PASSOV ;
                    yf += PASSOV ;
                    getimage( xe, yt, xd, yf, ponteiroopcv);
                    putimage( xe, yt, ponteiroopcv, NOT_PUT);
                    localvaux = 3;
                    strcpy ( sweep, "Sweep3" ) ;
                    break;


                  case 3:

                    putimage( xe , yt, ponteiroopcv, COPY_PUT);
                    yt += PASSOV ;
                    yf += PASSOV ;
                    getimage( xe, yt, xd, yf, ponteiroopcv);
                    putimage( xe, yt, ponteiroopcv, NOT_PUT);
                    localvaux = 4;
                    strcpy ( sweep, "Sweep4" ) ;
                    break;

                  case 4:

                    putimage( xe , yt, ponteiroopcv, COPY_PUT);
```

```
                    localvaux = i;
                    strcpy ( sweep, "Sweep1" ) ;
                    yt = 121 + 3 + 3 + ( localvaux - 1 ) * PASSOV ;
                    yf = yt + ALTURAOV ;
                    getimage( xe, yt, xd, yf, ponteiroopcv);
                    putimage( xe, yt, ponteiroopcv, NOT_PUT);

                    break;

                  }

                 break ;

                }


        }

        putimage ( 443, 115, ponteirojvl, COPY_PUT ) ;
        putimage ( 443, 115, ponteiroi, COPY_PUT ) ;


}

 void Wait ()

 {
    int i;

        for (i = 1; i < 100; i++)
        {

 }

void analisa ( i, x )

        int i, x ;

{


        int     yple,  valor, valor1, valor2, xpl, n ;
        int     tecla, valormax, li, yplepast ;
```

```
valormax = 0 ;

if ( i == nfl )
{
outportb (CKTRANSF, 0);
Wait () ;
outportb (CKTRANSF, 1);
outportb (CKTRANSF, 0);

valor = inportb (INADH) & 0xf0;
valor = (valor << 4) | (inportb (INADL) & 0xff);
}


outportb ( ENDBUFFER, num [i] ) ;
outportb ( CKTRANSF, 0 ) ;
delay ( 10 ) ;


for ( li = 0; li < 25; li++ )
{

outportb (CKTRANSF, 0);
Wait () ;
outportb (CKTRANSF, 1);
outportb (CKTRANSF, 0);

valor = inportb (INADH) & 0xf0;
valor = (valor << 4) | (inportb (INADL) & 0xff);
if ( valor > valormax )     valormax = valor;

}

if ( valormax < 1 )     valormax = 1 ;
valor1 =  ( log10 ((double) (valormax))) * 350 ;
valor2 = (int) (valor1 * GANHO);


ypl [i] = YFTELA - valor2 ;

if (ypl [i] < YTTELA)
        ypl [i] = YTTELA;
if (ypl [i] > YFTELA )
        ypl [i] = YFTELA ;

xpl = XETELA  + x * stepx;

if ( xpl == XDTELA ) xpl = XDTELA - stepx ;
if ( i == 0 )
{
```

```
                yple = YFTELA ;
                }
                else
                {
                yple = ypl[i - 1];
                }
                setcolor (BLACK) ;
                for ( n = xpl - stepx + 1; n (xpl + 1; n++ )
                {
                        line ( n, YFTELA -1, n, YTTELA + 1) ;
                }

                setcolor (WHITE) ;
                line (xpl - stepx, yple, xpl, ypl [i] ) ;

                line (xpl - stepx, YFTELA + 2, xpl, YFTELA + 2 ) ;
                if ( i == nfh )
                {
                setcolor (BLACK) ;
                line(XETELA, YFTELA + 2, XDTELA, YFTELA + 2 );
                setcolor (WHITE) ;
                }


}


void opcao ()


{


        if ( strcmp (status,"stop") == 0 )
        {
                janelahstop () ;
                while ( strcmp (status,"stop") == 0 )
                {
                        movimentoh () ;

                        if ( strcmp (status,"range") == 0 )
                        {
                                janelavr () ;
                                movimentovr () ;
                                strcpy (status,"stop") ;
                        }
                        if ( strcmp (status, "sweep" ) == 0 )
                        {
                                janelavs () ;
                                movimentovs () ;
                                strcpy (status, "stop") ;
```

```
                        }

                }
        }
        if ( strcmp (status,"start") == 0 )
        {
                janelahstart () ;

        }

}

void testa_stop ()

{
        if ( ( tecla = getch () ) == ENTER )

                strcpy (status,"stop") ;

        if ( tecla == 0 )

                getch() ;

}

void saida ()

{
        closegraph () ;

}


calibral ()


{
        int      nflint, i, j, valor, valormax, xt, yt ;

        valor = valormax = nflint = 0 ;

        outportb ( CKTRCAL, 3 ) ;
```

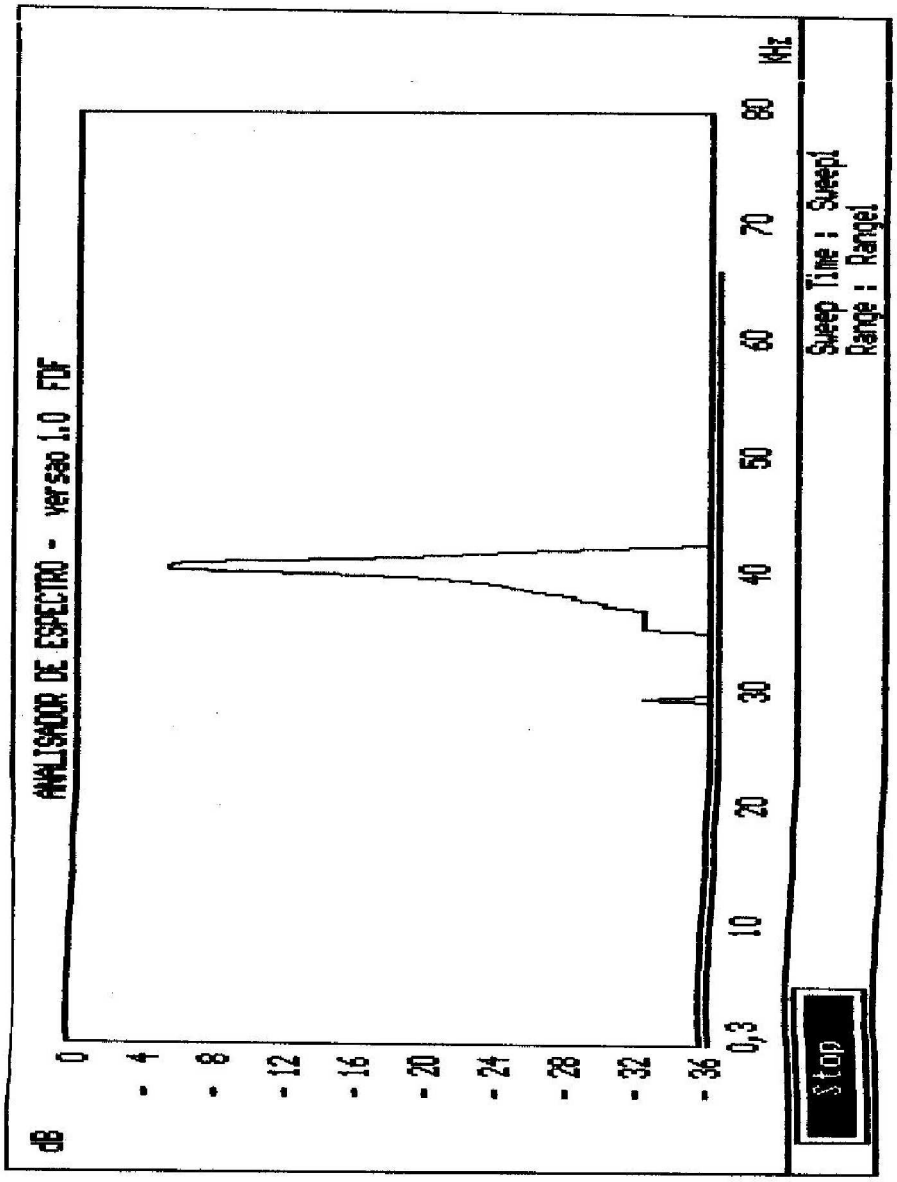
```
        xt = textwidth ("Retire o Sinal de Entrada e Pressione Qualquer
Tecla");
        yt = textheight ("Retire o Sinal de Entrada e Pressione Qualquer
 Tecla")
        xt = (620  - xt) / 2   ;
        yt = 180 + (20 - yt) / 2 ;
        outtextxy (xt,yt,"Retire o Sinal de Entrada e Pressione Qualquer
 Tecla")
        getch() ;
        putimage ( 1, 180, ponteirojhl, COPY_PUT ) ;
        xt = textwidth ("Calibrando") ;
        yt = textheight ("Calibrando") ;
        xt = (620  - xt) / 2   ;
        yt = 180 + (20 - yt) / 2 ;
        outtextxy ( xt, yt, "Calibrando" ) ;

        for ( i = 0; i < 100; i++)
        {
                outportb ( ENDBUFFER,  i  ) ;
                outportb ( CKTRANSF, 0 ) ;

                delay (11) ;

                if ( i == 0 )
                {
                        outportb (CKTRANSF, 0);
                        Wait () ;
                        outportb (CKTRANSF, 1);
                        Wait() ;
                        outportb (CKTRANSF, 0);
                        Wait() ;

                        valor = inportb (INADH) & 0xf0;
                        valor = (valor << 4) | (inportb (INADL) & 0xff);
                }

                for ( j = 0; j < 25; j++ )
                {


                        outportb (CKTRANSF, 0);
                        Wait () ;
                        outportb (CKTRANSF, 1);
                        Wait() ;
                        outportb (CKTRANSF, 0);
                        Wait() ;

                        valor = inportb (INADH) & 0xf0;
                        valor = (valor << 4) | (inportb (INADL) & 0xff);

                        if ( valor > valormax ) valormax = valor ;
```

03-01-2006

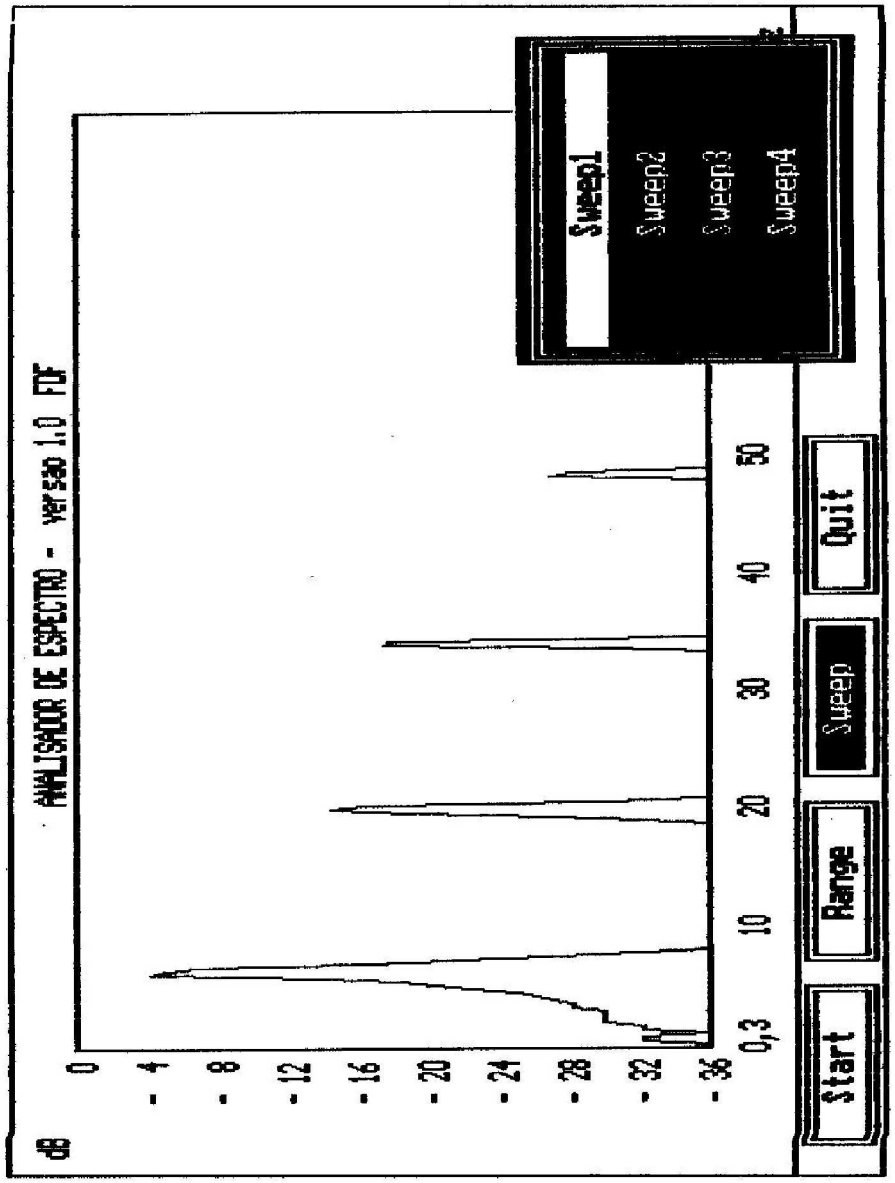
```
                }
                delay (i) ;

                        if (valormax > 390 )
                        {
                                nflint = i - 5;
                                if ( nflint < 0 )  nflint = 0 ;
                                CALAMP = valormax ;
                                i = 1000 ;
                        }


                delay ( 50 ) ;

            }

        outportb ( CKTRCAL, 0 ) ;
          return (nflint) ;

}



calibrah ()


{
        int      nfhint, i, j, valor, valormax ;

        valor = valormax = nfhint  = 0 ;

        outportb ( CKTRCAL, 5 ) ;
        for ( i = 180; i < 256; i++)
        {

                outportb ( ENDBUFFER, i  ) ;
                outportb ( CKTRANSF, 0 ) ;
                delay (ii) ;

                if ( i == 180 )
                {
                        outportb (CKTRANSF, 0);
                        Wait () ;
                        outportb (CKTRANSF, 1);
                        outportb (CKTRANSF, 0);

                        valor = inportb (INADH) & 0xf0;
                        valor = (valor << 4) | (inportb (INADL) & 0xff);
                }
```

```
                for ( j = 0; j < 25; j++ )
                {


                        outportb (CKTRANSF, 0);
                        Wait () ;
                        outportb (CKTRANSF, 1);
                        outportb (CKTRANSF, 0);

                        valor = inportb (INADH) & 0xf0;
                        valor = (valor << 4) | (inportb (INADL) & 0xff);

                        if ( valor > valormax ) valormax = valor ;
                }
                delay (1) ;

                        if (valormax > 200 )
                        {
                                nfhint = i  + 16;
                                if ( nfhint > 256 )  nfhint = 256 ;
                                i = 1000 ;

                        }


            delay ( 50 ) ;
        }

        outportb ( CKTRCAL, 0 ) ;
        return (nfhint) ;

}


/*********************************************************************
********
*******************          MAIN  FINAL          *******************
******/

main()

{
int  scan, k, range_nf, j ;


        inicializa () ;
        moldura () ;

        nfl = calibral () ;
```

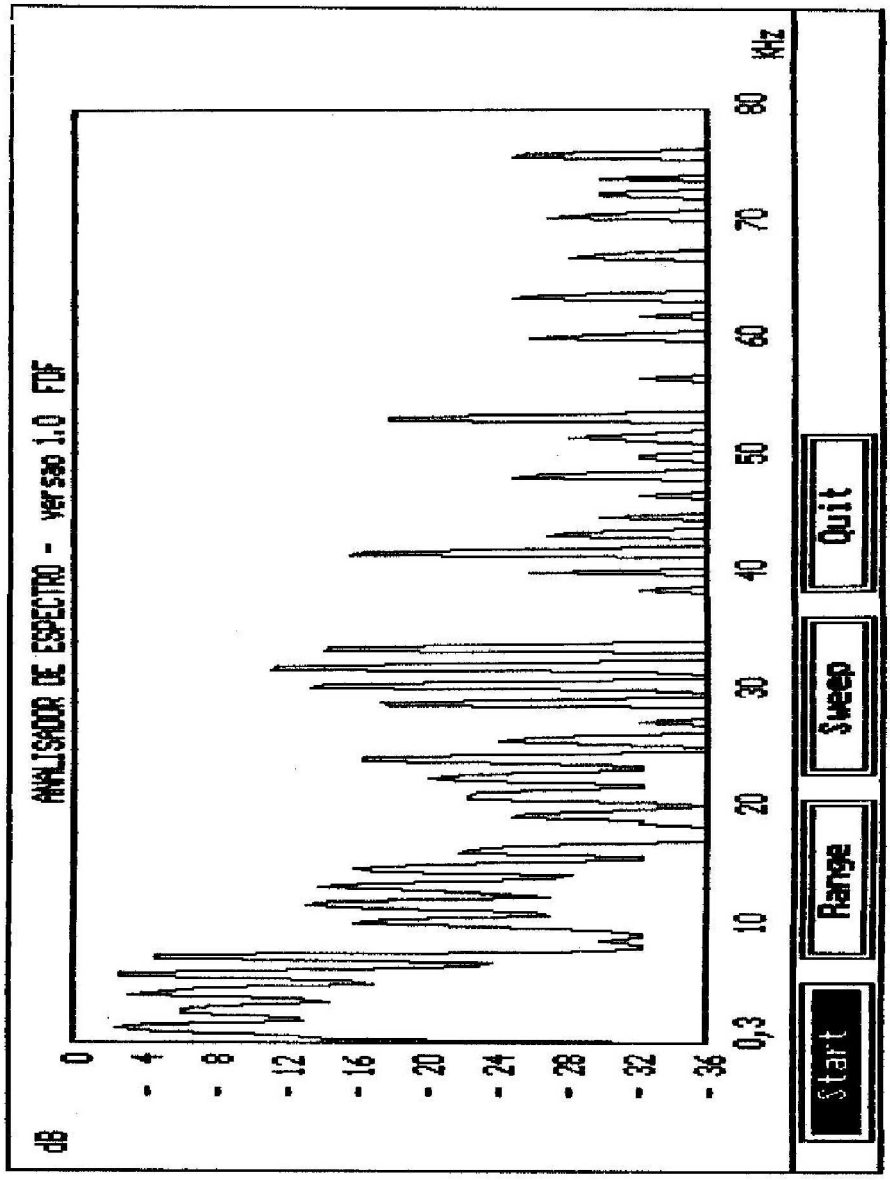
```
        nfh = calibrah () ;
        if ( nfh < 100 ) nfh = 256 ;
        range_nf = nfh - nfl ;
        stepx = (int) ((XDTELA - XETELA) / range_nf );

        opcao () ;

        while (  strcmp (status,"quit") != 0 )
        {
                while ( strcmp (status, "start") == 0 )
                {
                for (scan=nfl, j = 1;scan<nfh + 1; scan++, j++)
                    {

                        analisa (scan, j) ;

                        if ( kbhit () != 0 )

                                testa_stop () ;

                    }

                }

                opcao () ;
        }
        saida () ;
        return (0) ;
}

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


ANALISADOR DE ESPECTRO - versao 1.0 FDF
dB
0
-4
-8
-12
-16
-20
-24
-28
-32
-36
0,3
10
20
30
40
50
60
70
80
MHz
Sweep Time : Sweep1
Range : Range1
Stop

03-01-2006


ANALISADOR DE ESPECTRO - versão 1.0 FDF
dB
0
-4
-8
-12
-16
-20
-24
-28
-32
-36
0,3
10
20
30
40
50
Sweep1
Sweep2
Sweep3
Sweep4
Start
Range
Sweep
Quit

03-01-2006


ANALISADOR DE ESPECTRO - versão 1.0 FDF
dB
0
-4
-8
-12
-16
-20
-24
-28
-32
-36
0,3
10
20
30
40
50
60
70
80
KHz
Start
Range
Sweep
Quit

03-01-2006

B I B L I O G R A F I A

[1] - CARLSON, A. Bruce - "COMMUNICATION SYSTEMS"
McGraw-Hill - 1986 III ed.

[2] - STREMLER, Ferrel G. - "INTRODUCTION TO COMMUNICATION SYSTEMS"
Addison Wesley - 1982 II ed.

[3] - APLLICATION NOTE - 150 - "SPECTRUM ANALYSIS - SPECTRUM ANALYSER BASICS"
Hewlett Packard - 1974

[4] - ENGELSON, Morris - "SPECTRUM ANALYZER CIRCUITS"
Tektronix, Inc. - 1969 I ed.

[5] - MILLMAN, Jacob - "INTEGRATED ELECTRONICS"
McGraw-Hill - 1972

[6] - EGGEBRECHT, Lewis C. - "INTERFACING TO THE IBM PERSONAL COMPUTER"
Howard W. Sams & Co. - 1983 I ed.

[7] - OTT, Henry W. - "NOISE REDUCTION TECHNIQUES IN ELECTRONIC SYSTEMS"
John Wiley & Sons - 1976

[8] - POHLMANN, Ken C. - "PRINCIPLES OF DIGITAL AUDIO"
Howard W. Sams & Co. - 1989 II ed.

ESI / 1º Sem/91

MEC-UFF-CTC
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA DE TELECOMUNICAÇÕES

REGISTRO DE NOTAS OBTIDAS NA DEFESA DE PROJETO DE FIM DE CURSO

TEMA: Placa Analisadora de Espectro para IBM-PC

ORIENTADOR: ALBERTO FERREIRA DE SOUZA

ALUNO: Fábio Daros de Freitas GRAU: 10.0 (DEZ)

ALUNO: GRAU:

ALUNO: GRAU:

PROF.: José Baptista Barão Ribeiro

ALUNO: Fábio Daros de Freitas GRAU: 10.00 (DEZ)

ALUNO: GRAU:

ALUNO: GRAU:

PROF.: José Bianchini Latgé

ALUNO: Fábio Daros de Freitas GRAU: 10,0 (Dez).

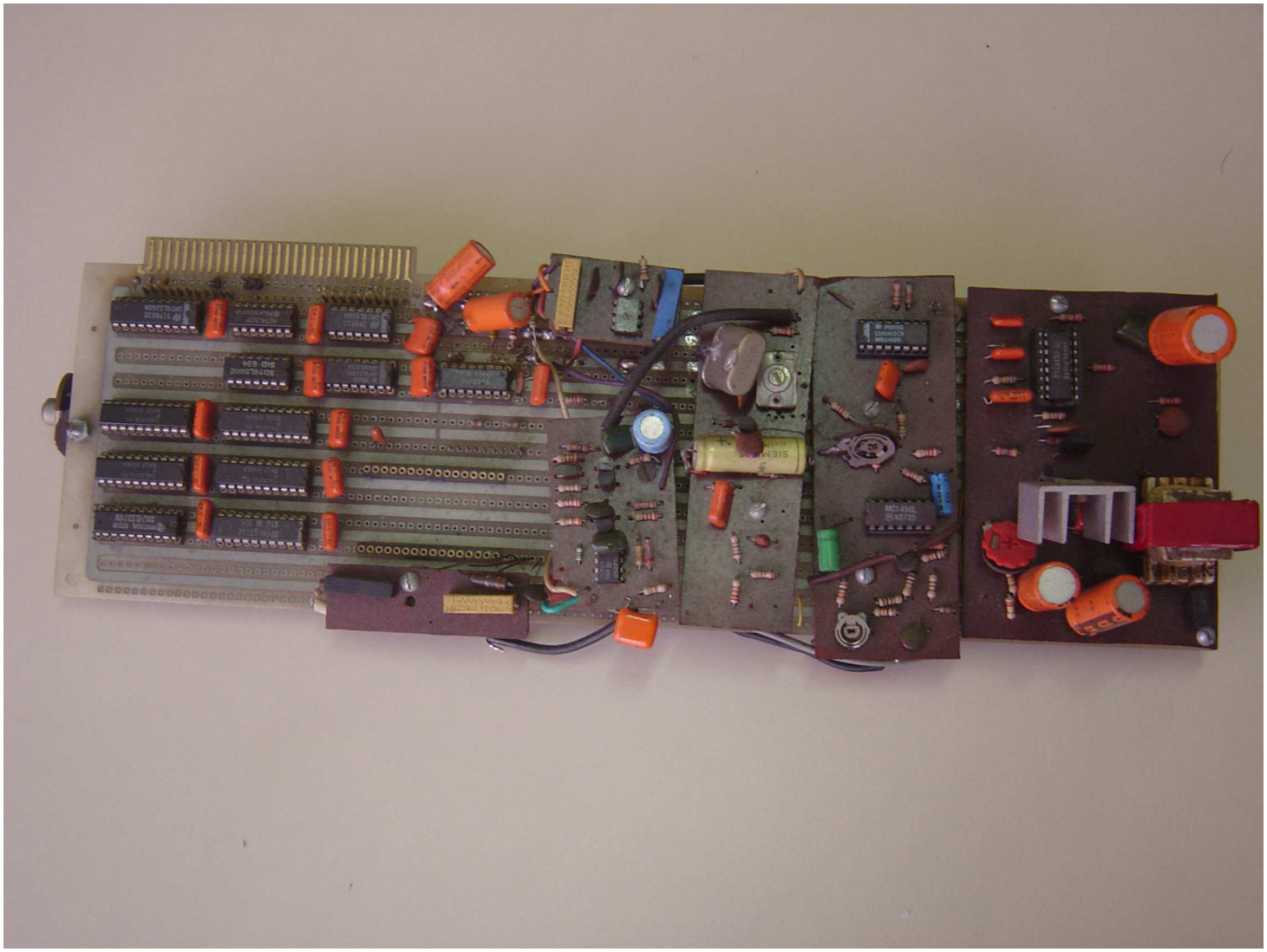
ALUNO: GRAU:

ALUNO: GRAU:

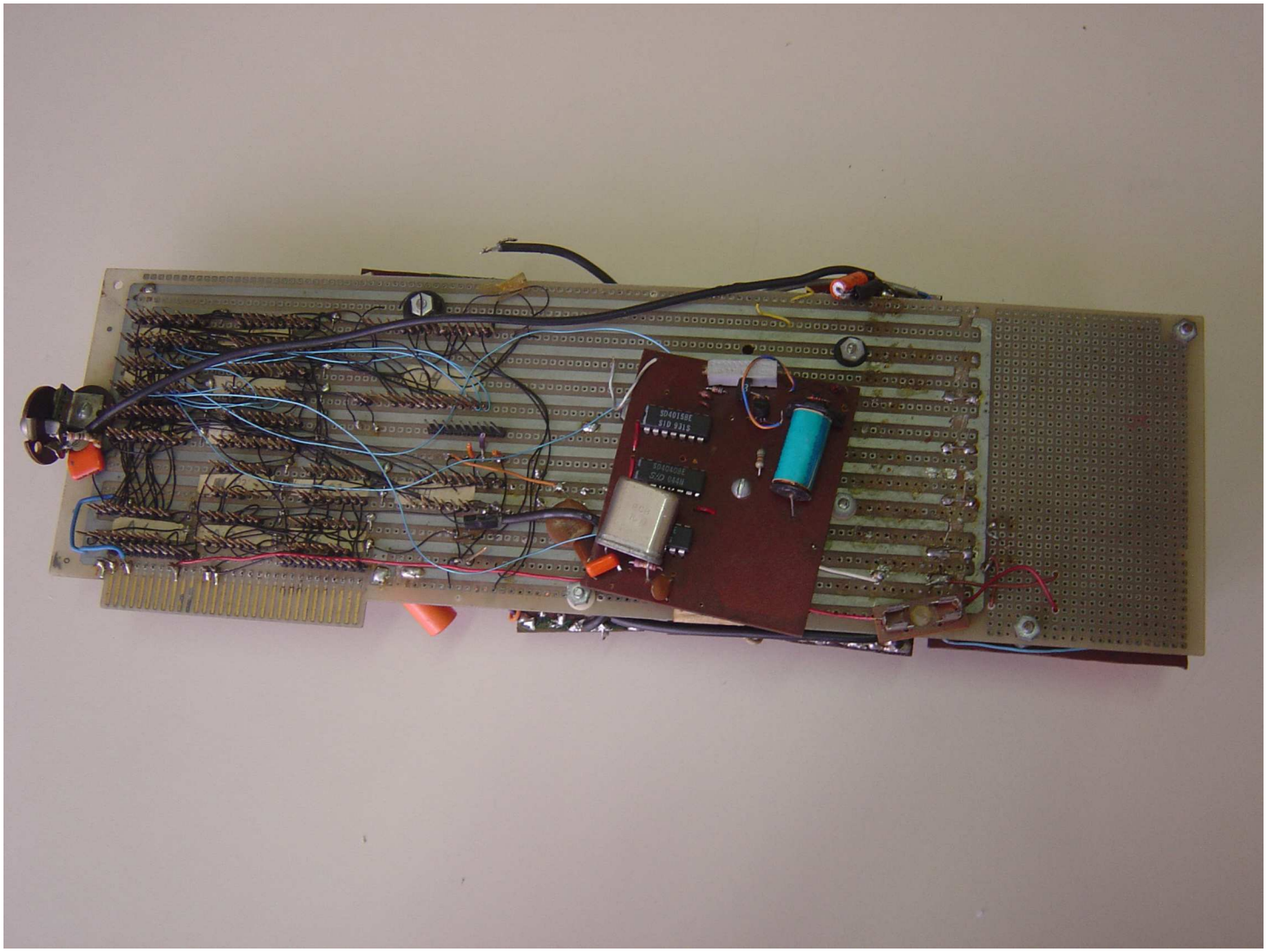
OBS.: CALCULE A MÉDIA E REGISTRE NA "FICHA DE CONTROLE DE TRABALHO FINAL DE CURSO"

Alberto Ferreira de Souza 19/11/91

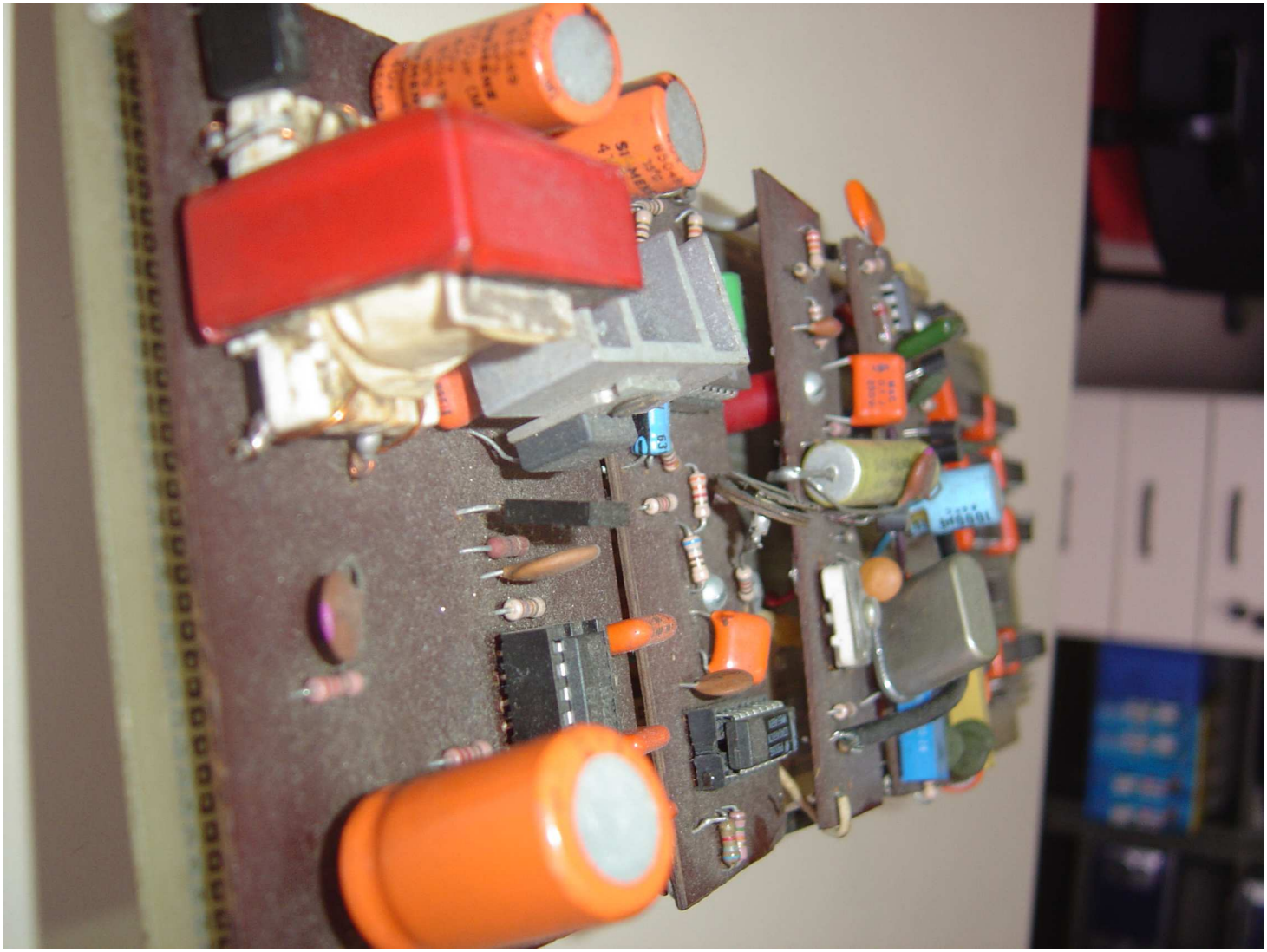
03-01-2006



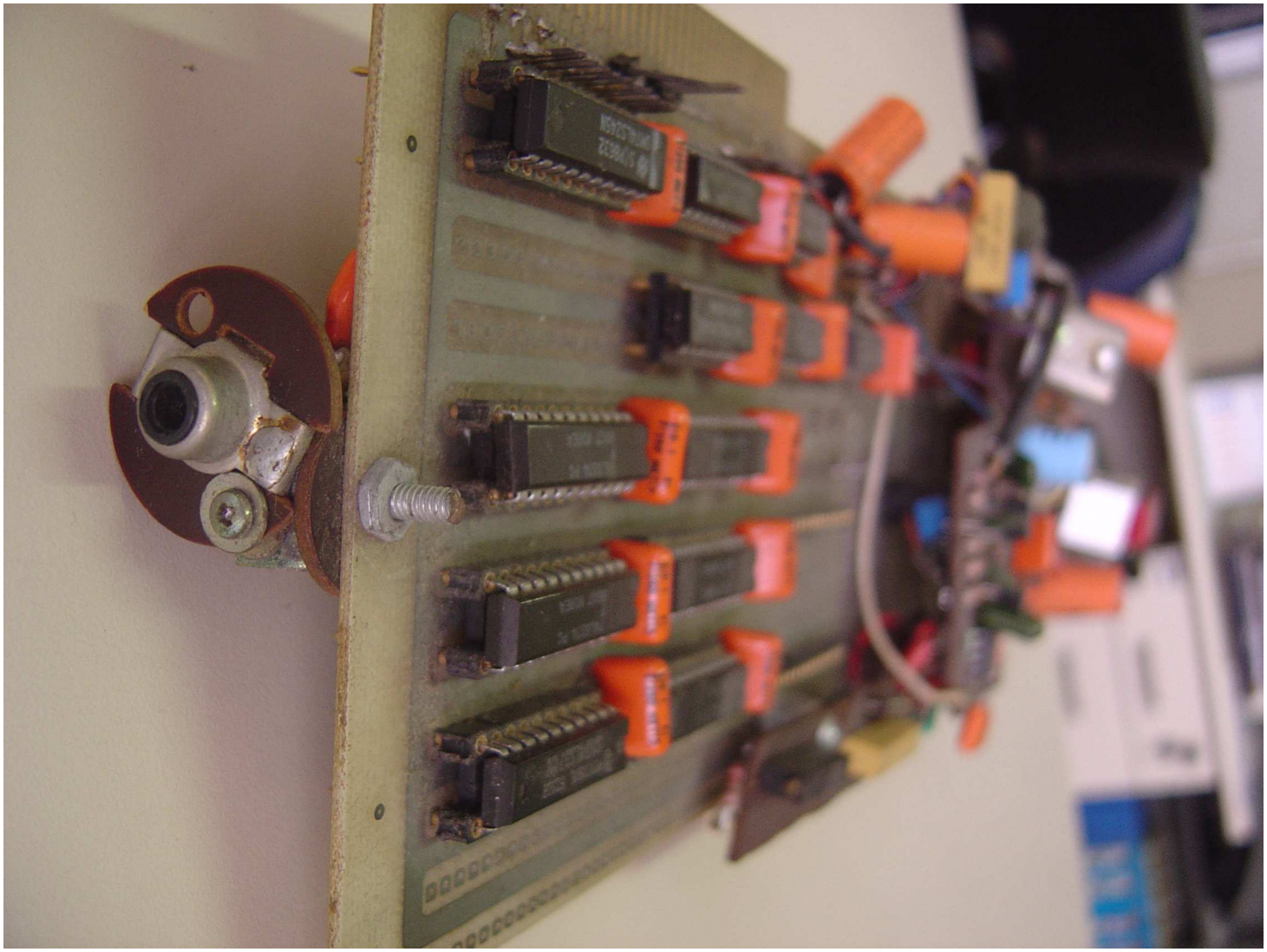
03-02-2006



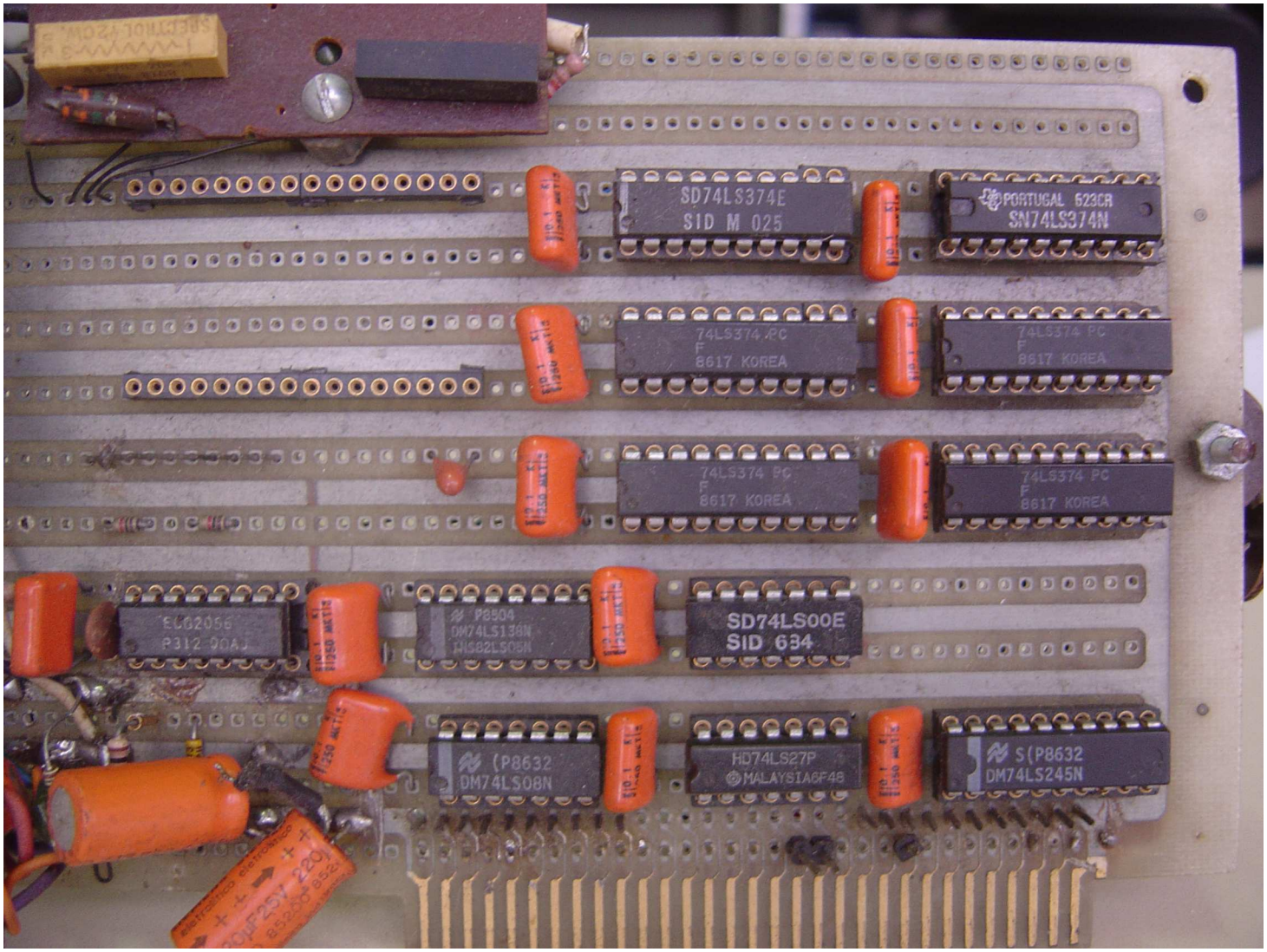
03-02-2006



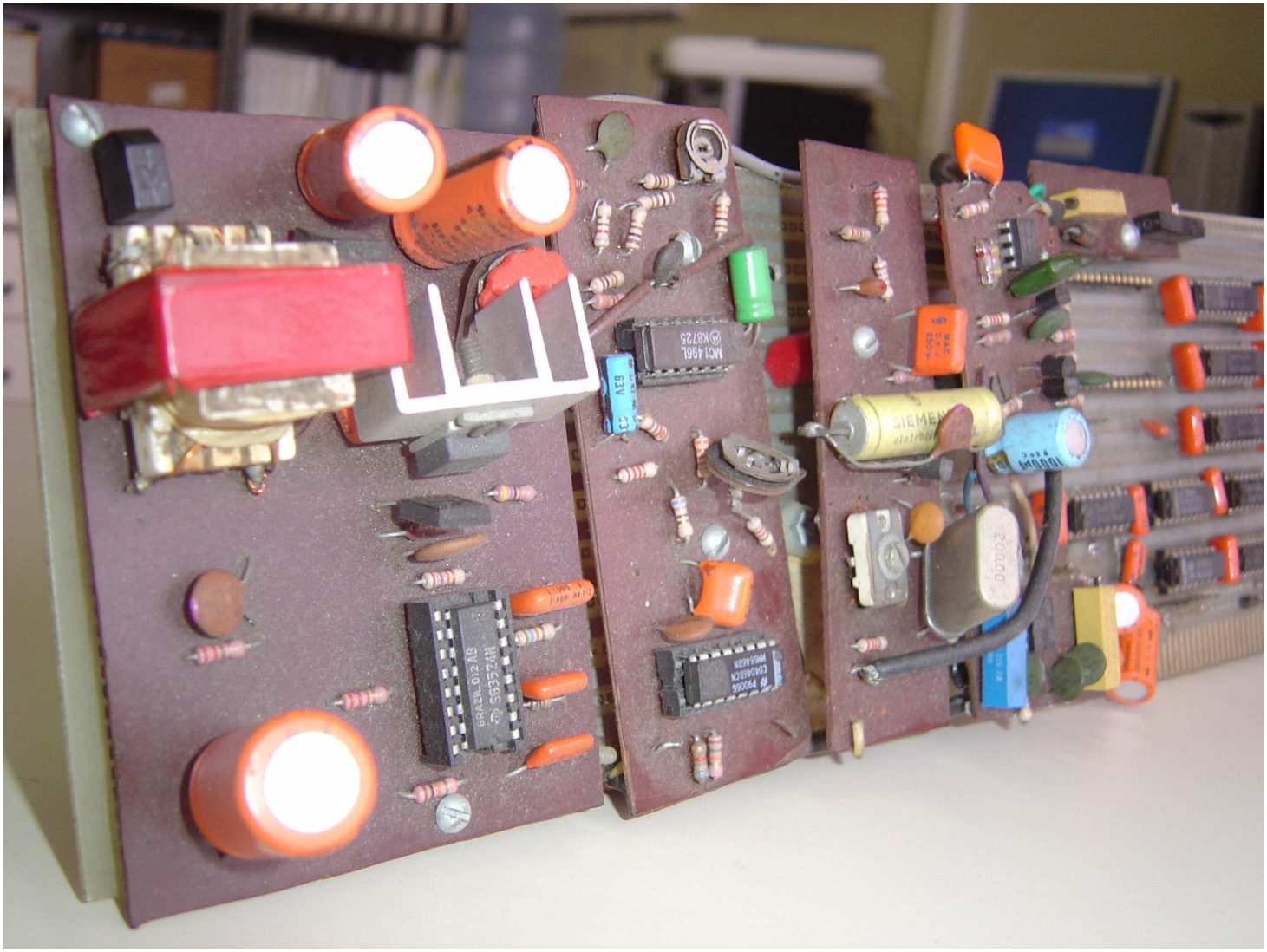
03-02-2006



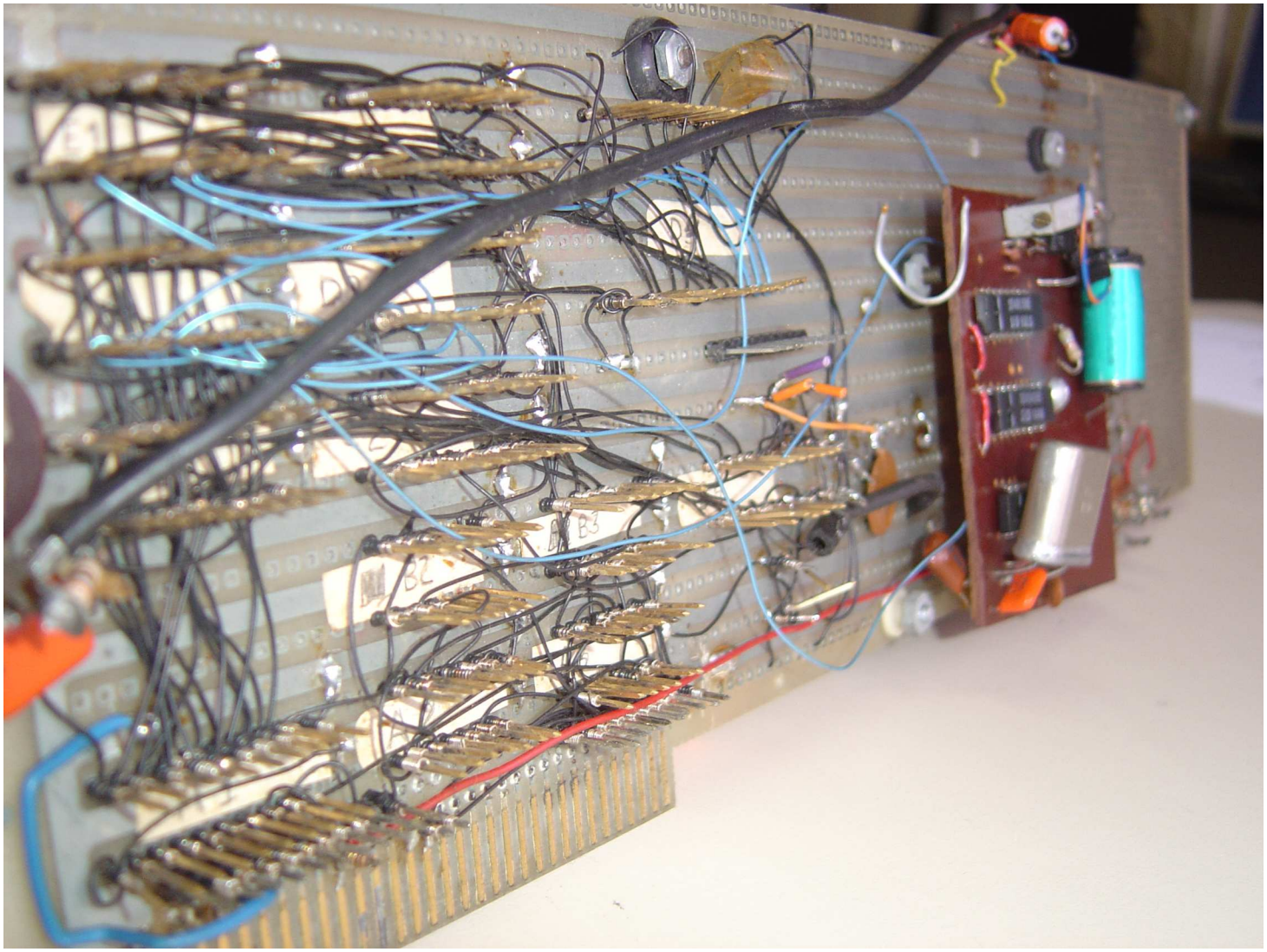
03-02-2006


SD74LS374E
SID M 025
PORTUGAL 523CR
SN74LS374N
74LS374 PC
F
8617 KOREA
74LS374 PC
F
8617 KOREA
74LS374 PC
F
8617 KOREA
74LS374 PC
F
8617 KOREA
ECG2056
P312 90AJ
P8504
DM74LS138N
SD74LS00E
SID 634
(P8632
DM74LS08N
HD74LS27P
MALAYSIA6F48
S(P8632
DM74LS245N

03-02-2006


MC1496L
X6725
6.3V
SIEMENS
10000µ
BRAZIL 012 AB
S6352AN
CMOS-BRICK

03-02-2006



03-02-2006