AQUISIÇÃO DE DADOS EM EXPERIÊNCIAS DE RPE USANDO MINICOMPUTADOR DE TEMPO REAL

Roberto Alves Nogueira

Tese apresentada à Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciências

Junho/90

Agradecimentos

- ao Prof. Geraldo Mathias, pela orientação.
- aos Profs. Manoel, Fábio, Ramayana e Geraldo Alexandre pelas inúmeras discussões e constante incentivo ao nosso trabalho.
- aos Profs. Sampaio, Maria Sylvia, Elmo, Paulo Sérgio e aos estudantes de pós-graduação Alvaro, Carlos Joel, Genivaldo e Rosana, pela indispensável colaboração no laboratório.
- aos Engs. José Geraldo e Gilberto, pelo bom funcionamento da eletrônica e ao Cleber, pelos desenhos.
- à Nelci, pela xerografia, e à Tia Maria pelo reconfortante cafezinho do início da noite.
- às agências FINEP, CNPq, AIEA e FAPEMIG pelo apoio financeiro que permitiu a aquisição e manutenção de vários dos equipamentos usados neste trabalho.
- à minha família, Maria Helena, Daphne e Nano, pelo carinho e compreensão.

Abstract

In this work we developed two real-time data acquisition computer systems, for data acquisition in Electron Paramagnetic Ressonance Experiments. In both, a high degree of freedom was kept in the measuring instruments configuration and programming, whith the aim of making easy a future reassignment to other experimental technics to be installed in the laboratory.

One of the systems is an improved version of the so called Passive Monitoring System and was built in such a way that it can be used in conjunction with the normal operation mode of the spectrometer. The other system incorporates, in addition to the data acquisition facilities, an step motor based gear to drive the sweep of the magnetic field.

Furthermore, a large program was also developed to facilitate the processing of the spectra and their transportation to other computers.

Indice

APENDICES :

APENDICE 1: O Programa MASPE

APENDICE 2: O Programa SMAP

APENDICE 3: O Programa SMOP

APENDICE 4: O Computador HP 1000

APENDICE 5: Outros Programas do Sistema

Introdução

O advento do uso dos computadores nos laboratórios de pesquisa tem contribuído de maneira sensível para a realização de avanços científicos consideráveis e, na maioria das áreas do conhecimento, tornou-se praticamente impossível manter a competitividade de um laboratório que não tenha acesso aos modernos recursos computacionais.

No caso específico da Física, o uso dos computadores data da época da criação destas máquinas e os físicos teóricos têm estado, desde então, entre os grandes usuários desses recursos. Já para os físicos experimentais, até bem pouco tempo, o uso do computador em suas atividades limitava-se, quase que exclusivamente, ao ajuste dos dados obtidos no laboratório e introduzidos no computador por meio de cartões ou fitas de papel.

A partir do início da década de 70, abriram-se novas perspectivas para o trabalho de laboratório pois, em virtude do aparecimento dos mini e micro computadores e dos chamados "instrumentos inteligentes", tornou-se viável a criação de novos sistemas de controle de experiências e aquisição de dados.

As pessoas que se envolveram inicialmente com esses sistemas enfrentaram uma série de dificuldades, decorrentes da falta de padronização nos mecanismos de interconexão dos primeiros equipa-

mentos desta nova geração. Essas dificuldades foram sensivelmente reduzidas a partir da estandardização das interfaces GP-IB (General Purpouse Interface Bus - norma IEEE/488), CAMAC, etc., que vieram permitir a ligação simultânea de vários instrumentos a uma única interface do computador de controle, com um protocolo de comunicação de uso simplificado e padronizado.

Dentro desse espírito de informatização, começaram a aparecer nessa mesma época os "sistemas computadorizados de laboratório", que normalmente trazem embutidos dispositivos de medida e controle, um computador dedicado e um sistema de interação com o usuário.

Por serem dedicados, estes sistemas permitem que suas operações sejam realizadas a um simples *apertar de botão*, o que os torna excelentes aquisições para os laboratórios onde se realizam trabalhos de rotina. Entretanto, para o pesquisador, que frequentemente necessita ajustar o seu equipamento às peculiaridades das suas experiências, eles apresentam várias limitações pois dificilmente podem ser usados em situações experimentais diferentes daquelas para as quais foram projetados. Além disso, os seus programas de controle não são divulgados e, em geral, a interligação de seus componentes é feita por meio de interfaces não padronizadas.

Como uma alternativa para esses sistemas, têm sido descritos na literatura[1] os chamados "***Sistemas de Arquitetura Aberta***", que

são compostos por um computador, geralmente chamado de estação de trabalho, ao qual estão ligados, através de interfaces padronizadas, instrumentos de medição e controle.

A configuração destes sistemas é de total responsabilidade do usuário a quem cabe, não apenas a escolha da estação de trabalho e dos instrumentos, como também a preparação dos programas necessários para o seu funcionamento.

Embora possa ser uma operação demorada, a implantação de um Sistema de Arquitetura Aberta para a realização das tarefas computacionais de um laboratório traz várias vantagens, pois permite uma escolha mais criteriosa dos instrumentos e do computador utilizados. Além disto, ao desenvolver os programas, o pesquisador envolvido no trabalho torna-se apto a redirecionar o sistema para outras técnicas, já que passa a ter um conhecimento detalhado de todas as suas partes.

Neste trabalho, vamos apresentar dois sistemas de medição que montamos para aquisição de dados em experiências de Ressonância Paramagnética Eletrônica, adotando a metodologia da "arquitetura aberta" e usando como base um computador HP 1000 de tempo real, um espectrômetro de EPR montado no próprio laboratório a partir de um eletroimã Varian e vários instrumentos que serão mencionados oportunamente.

No capítulo I, vamos introduzir alguns conceitos básicos sobre técnicas e recursos que foram usados ao longo do nosso trabalho tais como aquisição de dados, redução de ruídos, computação em tempo real, filtros digitais, etc.

No capítulo II, vamos descrever o Sistema Melhorado de Aquisição Passiva, que apesar de ser bem simples e não precisar de mudanças no espectrômetro, possui características que o destacam de sistemas semelhantes, descritos na literatura. A idéia central desse sistema é o redirecionamento dos sinais de ressonância e de intensidade do campo, do registrador X-Y para um conversor analógico-digital ligado ao computador. A aquisição dos dados é feita em tempo real, através de rotinas do HP 1000, mas sob o controle de uma base de tempo externa. O uso do DMA (Acesso Direto à Memória) e de outros recursos do sistema operacional permite que o computador continue disponível para a execução de outras tarefas, mesmo durante a aquisição dos dados de EPR.

No capítulo III, vamos descrever o Sistema do Motor de Passo que é um sistema mais complexo, onde o computador é responsável não somente pela aquisição dos dados, mas também pelo controle da evolução da varredura do campo magnético. Neste caso, a idéia central é a substituição do mecanismo original da varredura por um sistema externo, movido por um motor de passo, cujo comando é feito pelo computador. Graças a esse sistema de varredura, o campo magnético pode ser mantido fixo em um ponto qualquer, possibilitando a realização de várias medições do sinal de ressonância

naquele ponto. Estas medidas podem então ser submetidas a um processamento matemático adequado, tal como o cálculo de médias, filtragem digital, etc. Todo o processo é executado sob o controle do sistema operacional de tempo real e por isso, também nesse caso, o computador continua disponível para outras atividades durante a aquisição dos dados.

No capítulo IV, vamos comentar os resultados obtidos, juntamente com as nossas conclusões finais.

As descrições dos programas de computador, foram agrupadas nos apêndices 1, 2 e 3, com o intuito de facilitar a leitura desse trabalho. Nesses apêndices se encontram comentários sobre o desenvolvimento, o uso e as aplicações dos mesmos.

No apêndice 1, descrevemos o programa MASPE, desenvolvido especialmente para agilizar o processamento dos espectros obtidos com os sistemas dos capítulos II e III e facilitar o transporte dos dados para outros programas instalados no HP1000 , no IBM 4341 ou em qualquer outro computador que disponha de fita magnética de 1600 BPI.

Nos apêndices 2 e 3, descrevemos os programas **SMAP** e **SMOP** que controlam o Sistema Melhorado de Aquisição Passiva e o Sistema do Motor de Passo, respectivamente.

No apêndice 4, apresentamos uma breve descrição do computador HP 1000. E finalmente, no apêndice 5, apresentamos comentários sobre outros programas de interesse.

As listagens e os manuais de operação dos programas foram agrupadas em volume à parte, em virtude de suas extensões, e são mantidos à disposição dos interessados.

Ao desenvolver este trabalho, foram dois os objetivos que procuramos alcançar.

O primeiro foi o de trazer para o Laboratório de Ressonância mecanismos modernos de aquisição de dados, através de sistemas flexíveis que permitam, inclusive, um redirecionamento para outras técnicas experimentais que venham a ser implantadas.

O segundo foi o de reunir em um só trabalho, uma série de informações pormenorizadas sobre programação em tempo real, interconexão de instrumentos, aquisição de dados, etc., pois estas informações encontram-se de uma maneira muito dispersa na literatura e raramente os autores apresentam listagens e exemplos concretos de uso dos programas citados em suas publicações.

Capítulo 1

Aquisição de Dados e Computação em Tempo Real

1.1 Aquisição de Dados

Sempre que surge a necessidade de se submeter os dados obtidos em uma experiência a um processamento computacional qualquer, a primeira providência que deve ser tomada é a conversão dos mesmos em uma forma que possa ser compreendida pelo computador. Uma das maneiras mais simples de se realizar tal conversão é a digitação dos dados através de um teclado ligado ao computador, o que é conveniente apenas para pequenos volumes de dados.

Nas situações em que se manipula um grande número de dados, a digitação torna-se totalmente inadequada, não só pelo tempo consumido no processo, como também pela possibilidade de introdução de erros. Para contornar essa dificuldade, a maioria dos instrumentos de medição e controle hoje existentes no mercado dispõe de um conector apropriado para ligação direta aos sistemas de computação. Através desta ligação, é possível então o estabelecimento de um canal de dados rápido e extremamente confiável entre o instrumento e o computador.

Em várias técnicas experimentais, os resultados são obtidos a partir de sinais analógicos que devem ser convertidos em sinais digitais para que possam ser processados pelo computador. Nos casos em que o computador se encontra ligado diretamente à fonte dos dados, essa conversão se processa através de um dispositivo intermediário, genericamente chamado de "conversor analógico para

digital" - ADC -, cuja função é receber o sinal analógico proveniente da experiência e colocá-lo em uma forma digital que possa ser submetida ao processamento.

Ao implementar essa função, o conversor estará então transformando o sinal analógico de entrada, digamos **s(t)**, em uma sequência de saída, **s(k)**, através de um processo que é chamado *amostragem do sinal*. Os elementos da sequência **s(k)** são representações binárias dos valores do sinal s(t) nos pontos k da variável independente t e têm uma precisão finita, determinada pelo número de bits usados na conversão. Na Fig. 1-1 representamos a situação em que um conversor de 8 níveis (3 bits), usado para amostrar o sinal s(t), gera a sequencia de saida s(k) composta dos valores x_0, $x_1,\ldots,x_5$, amostrados nos instantes $t_0,\ldots,t_5$.

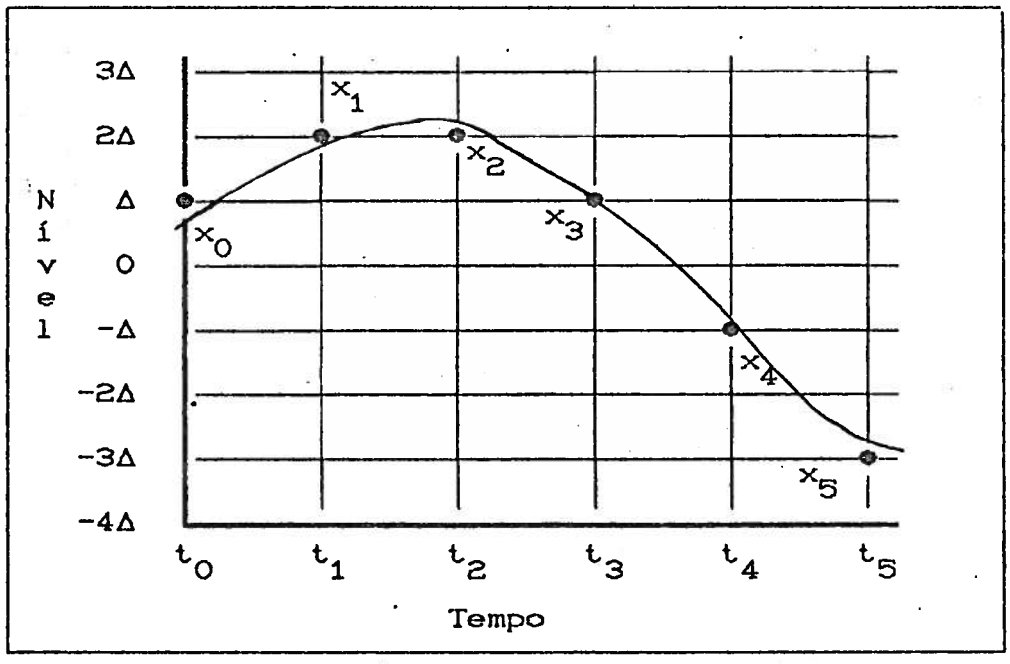


Fig. 1-1: Efeito de discretização em um processo de conversão analógico-digital com o uso de um ADC de 8 níveis (3 bits) com aproximação por arredondamento.

Essa precisão finita, que é inerente ao processo de conversão, acarreta uma discretização do sinal s(t) e introduz nos dados amostrados o chamado "erro de discretização" definido por

$$\Delta s(k) = s(k) - s_q(k) \qquad 1.1$$

onde $s_q(k)$ é o valor do sinal amostrado no instante t = k. Para um ADC de n bits, o bit menos significativo representa uma fração de $1/2^{n-1}$ de seu fim de escala de modo que os limites de $\Delta s(k)$ serão dados por

$$-\frac{b}{2} < \Delta s(k) < \frac{b}{2}, \text{ com } b = 2^{-n+1}$$

Para que esse erro de discretização não comprometa todo um programa de aquisição de dados, é necessário que ele seja suficientemente pequeno para ser ignorado face ao ruído presente no sinal.

1.2 O Teorema da Amostragem

Para produzir os elementos da sequência s(k), o ADC executa uma série de amostragens do sinal s(t) ao longo da variável independente t. Em cada uma dessas amostragens, a amplitude de s(t) é discretizada e colocada em elementos sucessivos de s(k). Na maioria das aplicações, a amostragem do sinal é feita com uma taxa fixa, fazendo assim com que os elementos de s(k) representem valores de s(t), uniformemente espaçados ao longo da variável t. Desta forma, o k-ésimo elemento da sequência será dado por

$$s(k) = P\{ s(kT + to) \} \qquad 1.2$$

$$k = 0, 1, 2, \ldots$$

onde T é o espaçamento e P é um operador que representa o processo de conversão analógico para digital.

Na amostragem do sinal **s(t)**, a escolha correta do espaçamento T, ou da taxa de aquisição, **1/T**, é da maior importância para a obtenção de bons resultados. Uma taxa muito pequena poderá causar a perda de detalhes de s(t), enquanto que uma taxa muito grande não sòmente produzirá desnecessariamente uma sequência com muitos termos, como também necessitará de um ADC mais rápido, o que nem sempre é tecnica e economicamente viável.

Pode-se mostrar que a taxa mínima de aquisição a ser usada na amostragem de um dado sinal **s(t)**, depende, essencialmente, das componentes de frequências do sinal e é determinada com o auxílio do Teorema da Amostragem[2], cujo enunciado é:

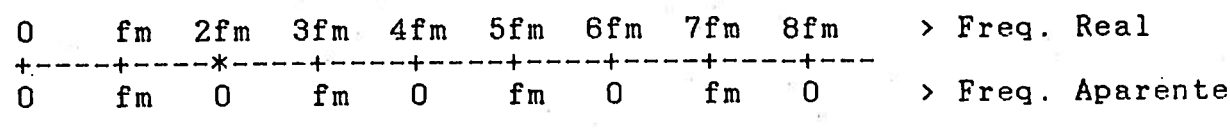
> Um sinal contínuo, que não contenha componentes de frequência acima de fm hertz, pode ser completamente determinado por seus valores amostrados em intervalos uniformes e de separação menor que 1/2fm segundos.

A frequência indicada pelo Teorema da Amostragem é também chamada de "Frequência de Nyquist"[3].

1.3 O Fenômeno de "Aliasing"

O Teorema da Amostragem, embora nos indique a máxima frequencia f_m que pode ser corretamente amostrada por um ADC, não nos dá nenhuma informação sobre o efeito que frequências maiores que f_m podem causar em s(k). Pode-se mostrar[3] que tais frequências, se presentes no espectro de **s(t)**, aparecerão no espectro de **s(k)** ocupando posições falsas, dando origem ao chamado efeito de "aliasing" ou "falseamento de amostragem".

Como consequência desse "falseamento", todas as frequências acima de f_m, presentes em **s(t)**, aparecem no espectro de **s(k)** "rebatidas" no intervalo [0, f_m]. Em particular, as frequências $2f_m$, $4f_m$, ... aparecem em f=0, enquanto as frequências $3f_m$, $5f_m$, ... aparecem em f=f_m conforme se vê no diagrama abaixo



Esse efeito pode ser evitado impedindo-se que frequências superiores a f_m estejam presentes na entrada do ADC, o que pode ser conseguido com o uso de filtros tipo "passa-baixas", com frequência de corte inferior a f_m.

Na Fig. 1-2 apresentamos um exemplo de falseamento de amostragem obtido em uma experiência de laboratório onde um gerador de onda senoidal, ajustado para 725 Hz, foi ligado a um ADC operando a uma taxa de 200 conversões por segundo. Como nessas condições a

maior frequência que pode ser amostrada é de 100 Hz, o sinal de 725 Hz é incorretamente registrado e provoca o aparecimento de uma linha no espectro de potencia na posição 75 Hz (100 - 25).

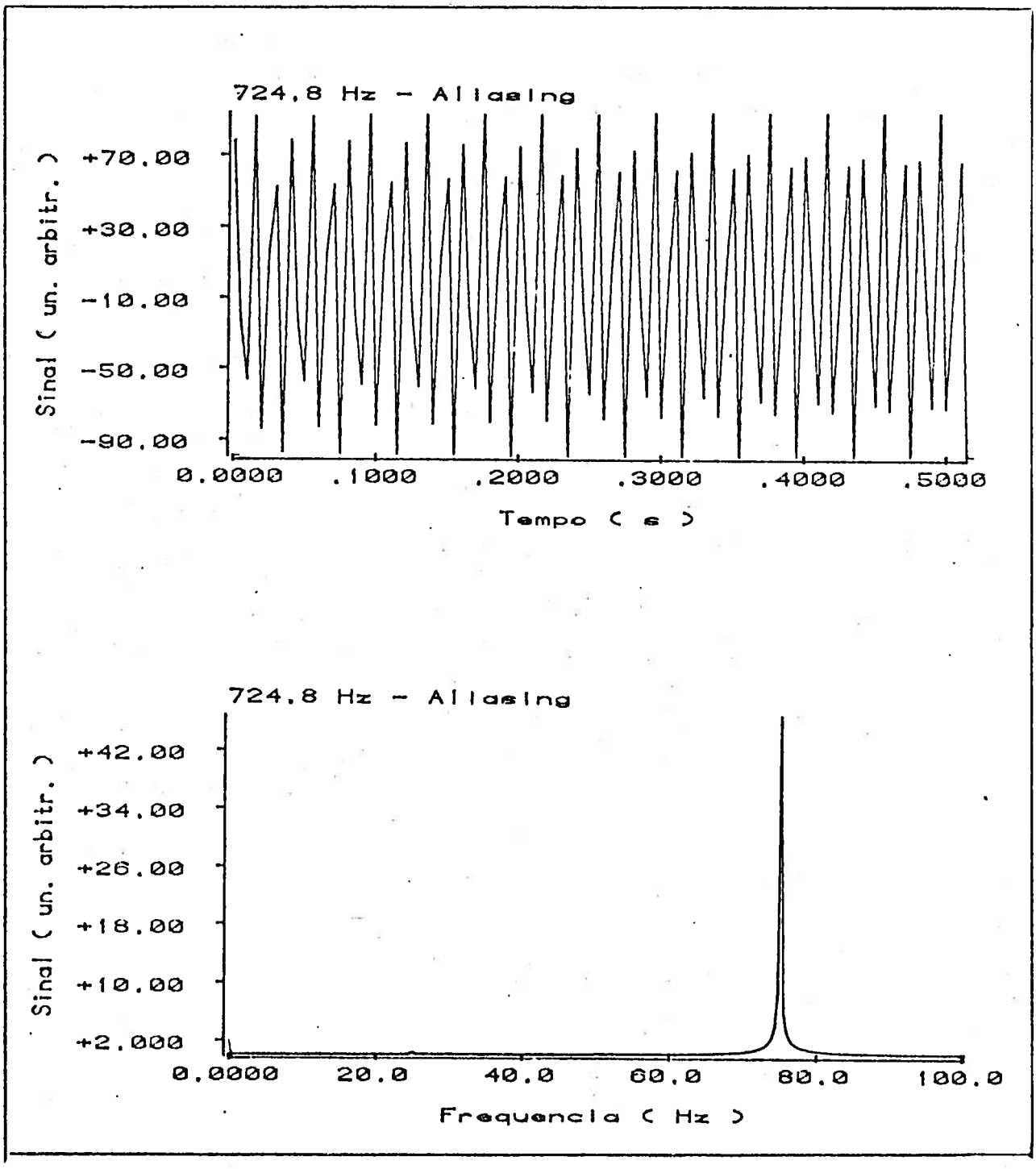


Fig. 1-2: "Aliasing" de um sinal de 725 Hz - Na parte superior, o sinal amostrado e abaixo, o seu espectro de potência, mostrando uma linha intensa em f = 75Hz

1.4 Fontes de Ruídos

Os dados obtidos em uma experiência, sejam eles registrados de forma analógica ou digital, estão sempre acompanhados de algum tipo de ruído. Nas situações em que esses dados são obtidos essencialmente através de instrumentação eletrônica, os ruidos podem ser agrupados da seguinte maneira:

Ruído do tipo 1/f: Este tipo de ruído aparece como consequência de flutuações de corrente nos resistores e é dado por

$$\langle V_n \rangle^2 = AR^2I^2\Delta f/f$$

onde A é uma constante, R a resistência, I a corrente, Δf a faixa de passagem do detetor e f a frequência de sintonia do detetor.

Ruido Johnson: Este tipo de ruído aparece como consequência de flutuações na densidade de elétrons nos resistores e é dado por

$$\langle V_n \rangle^2 = 4kTR\Delta f$$

onde k é a constante de Boltzman, T a temperatura, R a resistência e Δf a faixa de passagem do detetor.

Shot Noise: Esse tipo de ruído aparece como consequência de flutuações na emissão termoiônica (válvulas). Nos dispositivos semicondutores, é associado às flutuações no número de portadores e é dado por

$$\langle I_n \rangle^2 = 2eI_{dc}\Delta f$$

onde e é a carga do elétron, I_{dc} o valor da corrente e Δf a faixa de passagem do detetor.

Além desses tipos, que podem ser considerados intrínsecos à instrumentação eletrônica, várias outras fontes podem contribuir para aumentar o nível de ruído presente em uma experiência, como, por exemplo,

acoplamentos resistivos, indutivos e capacitivos devidos à presença de resistores, indutores e capacitores "parasitas",

efeito termoelétrico devido à presença de junções de metais diferentes e sujeitas a variações de temperatura,

microfonia devida à presença de vibrações mecânicas que possam levar ao aparecimento de ruídos elétricos,

interferências devidas à rede elétrica, emissoras de rádio e TV, fornos de indução, etc.

1.5 Redução de Ruídos: "Signal Averaging"

A técnica de redução de ruído, conhecida como "signal averaging" ou "ensamble averaging", é um método que permite a recuperação de um sinal, mesmo na presença de ruídos intensos, através de uma melhoria da relação sinal/ruído obtida a partir do cálculo da média de várias medições sincronizadas. Esta técnica está baseada no fato de que o nível do sinal de interesse, suposto estável,

cresce linearmente com o número N de medições, enquanto que o do ruído, suposto aleatório, cresce com $\sqrt{N}$ [3].

Na prática, a aplicação deste método consiste em se tomar várias medidas do sinal de interesse e, em seguida, calcular a média dessas medidas. Para o caso de espectros, sejam eles de Ressonância Magnética ou de outra técnica experimental qualquer, o problema da média é um pouco mais complexo, pois temos que tomar a média de todos os pontos da seqüência **s(k)** que representa a amostragem do sinal. Para isto, o que se faz normalmente é registrar o mesmo espectro várias vezes, de modo a se obter um conjunto de varreduras $s_n(k)$, onde **k** é o número do ponto e **n** é o número da varredura.

Uma vez registradas as **n** varreduras, o próximo passo é então somá-las, ponto a ponto, de modo a obter a média dos espectros, dada por

$$\langle s(k) \rangle_{Nv} = \frac{\sum_{1}^{Nv} s_n(k)}{Nv}$$

onde N_v é o número de varreduras efetuadas. Embora essa operação seja muito simples, e possa ser realizada mesmo nos pequenos computadores, ela apresenta duas limitações que são 1) a possibilidade da ocorrência de "overflow" na somatória e 2) o fato de que o resultado só pode ser conhecido após a Nv-ésima varredura.

Para contornar essas limitações, costuma-se dar preferência ao cálculo da <u>média estável</u>, conforme mostraremos a seguir:

Sendo N o número de varreduras realizadas até um dado instante t, temos:

$$\langle s(k)\rangle_{N-1} = \frac{\sum_{1}^{N-1} s_n(k)}{N-1}$$

$$\langle s(k)\rangle_{N} = \frac{\sum_{1}^{N-1} s_n(k) + s_N(k)}{N}$$

$$\langle s(k)\rangle_{N} = \frac{\langle s(k)\rangle_{N-1}\,(N-1) + s_N(k)}{N}$$

$$\langle s(k)\rangle_{N} = \langle s(k)\rangle_{N-1} + \frac{1}{N}\left(s_N(k) - \langle s(k)\rangle_{N-1} \right)$$

Esta última expressão nos dá uma maneira recursiva de calcular a média e evita o aparecimento de "overflow" nos casos em que se processa um número muito grande de dados.

Apesar da simplicidade do método, a operação de média só pode ser empregada em situações experimentais que satisfaçam aos seguintes requisitos:

O sinal a ser processado deve ser repetitivo, embora não necessite ser periódico.

Deve existir um mecanismo muito bem definido, sincronizado com o sinal de interesse, para disparar o início

da amostragem e garantir, assim, uma adição coerente das várias medições.

Do sinal a ser processado, deve-se eliminar ao máximo, por outros métodos, quaisquer ruídos que possam estar sincronizados com a amostragem, para impedir que os mesmos sejam coerentemente adicionados.

1.6 Redução de Ruídos: Filtros Digitais

Uma outra técnica que vem sendo muito empregada no processamento de sinais digitais é a da filtragem de sinais, através dos chamados "filtros digitais". Estes filtros representam, para os sinais digitais, o mesmo papel que o dos filtros analógicos para os sinais contínuos, ou seja, são "dispositivos" capazes de modificar, de uma maneira previsível, o espectro de frequência do sinal.

A implementação dos filtros digitais pode ser efetuada tanto em "hardware", como em "software" e o aumento de suas aplicações em vários ramos da ciência reflete, sem dúvida, a crescente disponibilidade de computadores e instrumentos correlatos nos laboratórios de pesquisa.

Tal como no caso analógico, um filtro digital pode ser descrito por seu comportamento no domínio da frequência (resposta à frequência), ou no domínio do tempo (resposta ao impulso unitário).

Na maioria dos casos, é usada a primeira forma, resposta à frequência, porque ela traduz, de maneira mais evidente, a "habilidade" do filtro em discriminar certas regiões do espectro em favor de outras. A teoria dos filtros digitais é um assunto bastante complexo e extenso[4], mas os princípios básicos que regem seu funcionamento podem ser compreendidos a partir de uma breve revisão de alguns conceitos dos Sistemas Lineares e da Análise de Fourier[2], que faremos a seguir.

- Um *sistema* é uma função ou um algoritmo que, para cada sequência de entrada, x(n), atribui uma única sequência de saida, y(n). Representando o sistema por *S*, podemos escrever

$$S\{x(n)\} = y(n)$$

- Um *sistema L* é dito *linear*, se para ele vale o princípio da superposição, ou seja,

$$L\{a_1x_1(n) + a_2x_2(n)\} = a_1L\{x_1(n)\} + a_2L\{x_2(n)\}$$

onde a_1 e a_2 são constantes arbitrárias.

- Um *sistema linear L* é dito *invariante no tempo*, se

$$L\{x(t - t_0)\} = y(t - t_0)$$

onde

$$L\{x(t)\} = y(t)$$

- Os filtros digitais, aqui considerados, são *sistemas lineares invariantes no tempo* representados por ***L***.

- Se $h(t)$ é a resposta de um sistema linear invariante no tempo, para uma excitação do tipo impulso unitário, $\delta(t)$, ou seja, se

$$h(t) = L\{\delta(t)\}$$

então,

$$h(t - \tau) = L\{(t - \tau)\}$$

- A resposta $f_r(t)$, de um sistema linear invariante no tempo, a uma excitação arbitrária $f_e(t)$, pode ser expressa como a convolução da excitação $f_e(t)$ com a resposta ao impulso unitário do sistema, $h(t)$; isto é,

$$f_r(t) = h(t) * f_e(t)$$

onde o sinal * representa a operação de convolução.

- A transformada de Fourier (F), da resposta ao impulso unitário de um sistema linear, é chamada *função sistema*, isto é,

$$H(\omega) = F\{h(t)\}$$

- Teorema da convolução no tempo: Se $F\{f_1(t)\} = g_1(\omega)$ e $F\{f_2(t)\} = g_2(\omega)$ então,

$$F\{f_1(t) * f_2(t)\} = g_1(\omega)\, g_2(\omega)$$

Como dissemos anteriormente, os filtros digitais podem ser implementados no domínio da frequência ou no domínio do tempo. No pri-

meiro caso, o passo inicial é o cálculo da transformada de Fourier do sinal a ser filtrado, o que se faz normalmente com auxílio de um algorítmo de FFT (Fast Fourier Transform). De posse do espectro do sinal, podemos então ajustar as amplitudes e as fases das várias frequências, de modo a refletir as caracteristicas desejadas para o filtro. Feito este ajuste, o sinal filtrado no domínio do tempo pode ser obtido através de uma FFT inversa. Este processo é de grande flexibilidade, pois requer apenas a multiplicação do espectro do sinal pelas características do filtro, além de ser uma boa escolha nas situações onde se dispõe de recursos eficientes para o cálculo das FFTs.

No segundo caso, todo o processo de filtragem se passa no domínio do tempo e é realizado através da convolução do sinal com a resposta ao impulso unitário do filtro apropriado. Embora o processo no domínio do tempo não nos pareça tão intuitivo quanto no domínio da frequência, ele pode ser compreendido a partir das observações feitas anteriormente sobre a Análise de Fourier e os Sistemas Lineares. Com efeito, de acordo com o teorema da convolução, o produto do espectro do sinal pelo espectro do filtro, que é o ponto central da filtragem do domínio da frequência, é equivalente, no tempo, à convolução do sinal com a resposta ao impulso do filtro.

De um modo geral, o projeto de um filtro digital começa pela escolha de uma função H(w) que represente a resposta de frequência desejada[4]. A partir deste ponto, pode-se implementar o filtro

diretamente no domínio da frequência ou calcular a resposta ao impulso unitário, pära implementação no domínio do tempo. Nas aplicações de tempo real, costuma-se dar preferência aos filtros no domínio do tempo para se evitar o cálculo das FFTs que consome tempo e memória, nem sempre disponíveis.

Neste trabalho, tivemos a oportunidade de usar dois tipos de filtros digitais para atenuação de ruidos de alta frequência. O primeiro foi um algorítimo de alisamento (smoothing), que consiste em substituir o valor de cada ponto de um espectro pela média aritmética dos valores do próprio ponto e de seus N_v vizinhos mais próximos.

O segundo filtro utilizado foi um filtro tipo "passa-baixas", no domínio do tempo, conhecido na literatura[5] como NER ("Nearly Equal Ripple"). Esse filtro apresenta como principal característica a minimização das oscilações em torno da frequência de corte, decorrentes do uso de um número finito de termos para representar a resposta ao impulso unitário (fenômeno de Gibbs)[5].

Essa minimização é conseguida com o uso de uma função peso, ajustável, para modificar de maneira adequada os coeficientes da sequência do filtro. O resultado que se obtém com esta técnica é a "dispersão" das oscilações, então concentradas na frequência de corte, ao longo de toda a faixa de corte e de passagem, com a consequente redução de suas amplitudes. Os parâmetros que caracterizam esse filtro são a frequência de transição (β), a largura

da região de transição (δ) e a tolerância na amplitude das oscilações (ξ), conforme se pode ver na Fig. 1-3. Por uma questão de conveniência, as frequências são medidas em relação à frequência de Nyquist e variam, portanto, de 0 a 1.

O comportamento desses dois filtros, tanto no domínio do tempo como no da frequência, está representado nas Figs. 1-4 e 1-5, onde podemos ver, de maneira clara, a superioridade do filtro NER sobre o método de alisamento no que diz respeito à atenuação das baixas frequências.

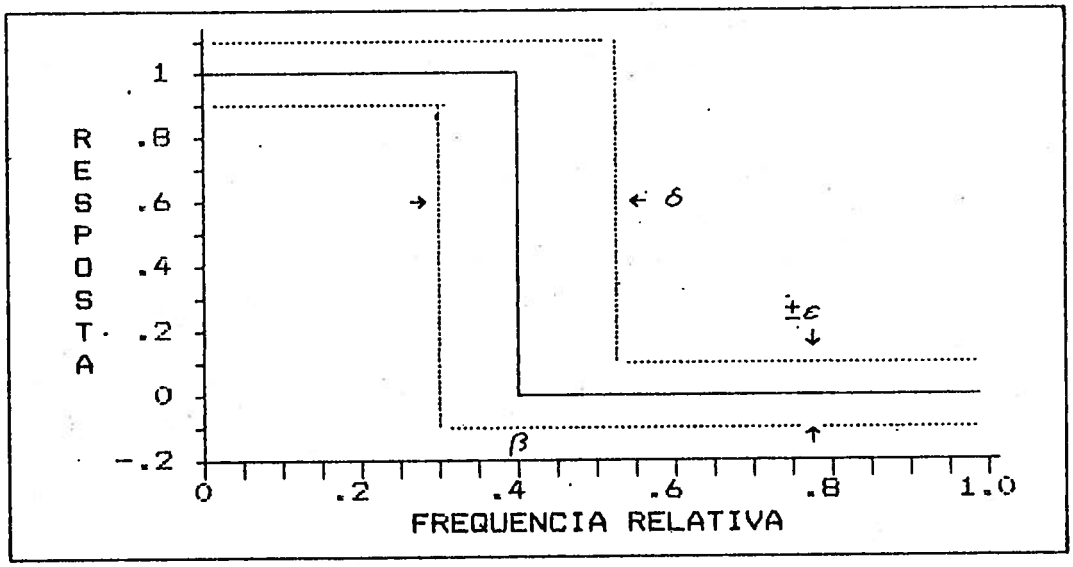


Fig. 1-3: Parâmetros do filtro NER
β - frequência de corte
δ - região de transição
ε - tolerância

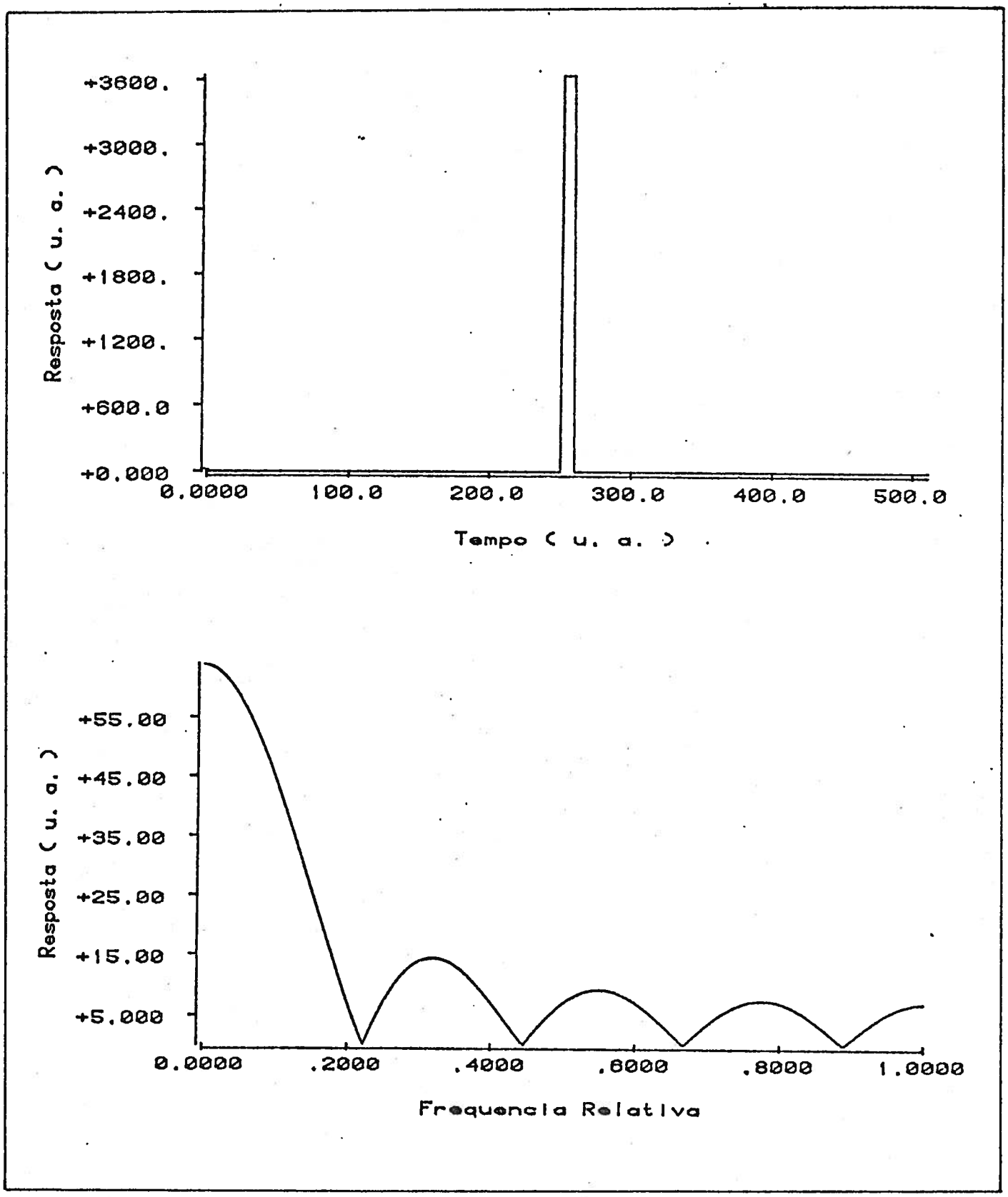


Fig. 1-4: Comportamento do filtro de alisamento. Acima, resposta ao impulso e, abaixo, resposta à frequência.

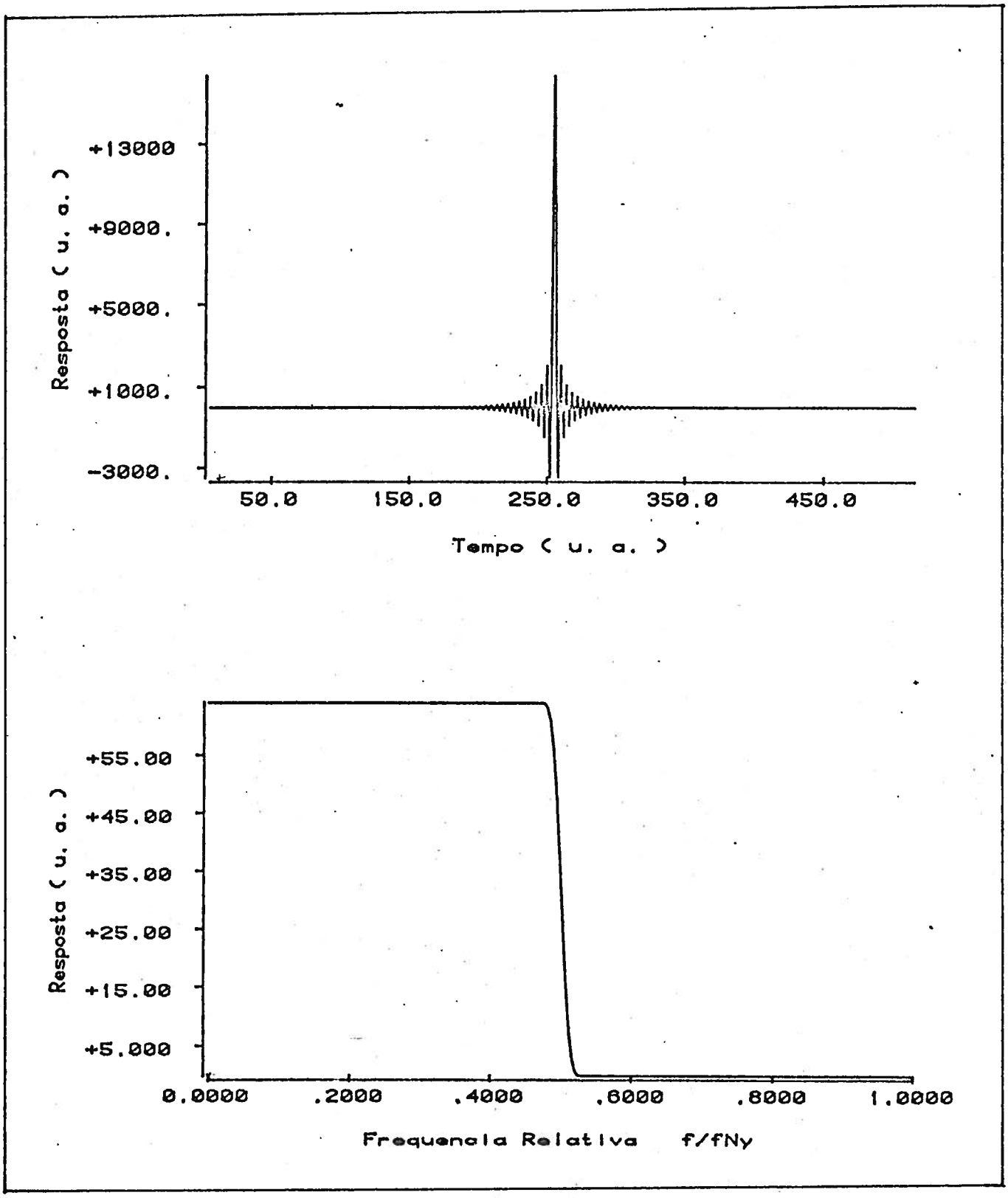


Fig. 1-5: Comportamento do filtro NER. Acima, resposta ao impulso e, abaixo, resposta à frequência.

1.7 Computação em Tempo Real

Como vimos nas seções anteriores, tanto as tarefas de aquisição como as de processamento de dados dependem de um computador de controle. Esse computador, por sua vez, tanto pode ser dedicado exclusivamente a uma tarefa como pode ser uma máquina de uso mais geral, capaz de atender simultâneamente a várias tarefas de vários usuários. De qualquer forma, seja qual for o computador usado, é necessário que ele seja capaz de interagir com o processo sob controle durante sua evolução, ou seja, ele deve ser um *computador de tempo real*

Para que um computador possa ser classificado como sendo "*de tempo real*", é necessário que o sistema operacional que o controla seja capaz de executar, entre outras, duas funções específicas, que são:

1. -executar programas em instantes pré-determinados.
2. -responder prontamente a eventos externos que possam ocorrer em instantes imprevisíveis.

Para que essas especificidades possam ser atendidas, o que se faz na criação do sistema operacional é dar uma atenção especial às rotinas de contagem de tempo, de controle de entrada e saída e à prioridade com que os programas são executados pela máquina.

O computador HP 1000 que usamos neste trabalho é uma máquina de tempo real do tipo multiusuário - multiprogramação, o que signi-

fica que ele pode atender, simultâneamente, a vários usuários que estejam executando várias tarefas diferentes.

Essa simultaneidade na execução das várias tarefas é, na verdade, apenas aparente. O que acontece na realidade é que, como os periféricos de entrada e saida são muito mais lentos que a UCP (Unidade Central de Processamento), existe um "tempo morto", no qual o computador fica esperando o término de uma tarefa de entrada e saida, que pode ser usado por um outro programa dando assim, ao usuário, a impressão de que os programas estão sendo executados ao mesmo tempo.

Todas as atividades do computador HP 1000, são coordenadas por um sistema operacional, baseado em disco, chamado "RTE-6/VM" (Real Time Executive 6 / Virtual Memory)[6] que, além de implementar as funções de tempo real anteriormente mencionadas, cuida do gerenciamento dos arquivos e das várias outras funções típicas de um sistema operacional.

Para o tratamento de eventos com "hora marcada" e manutenção do relógio de tempo real, o RTE-6 usa uma interface, instalada no computador, chamada "Time Base Generator - **TBG**". Esta interface dispõe de um oscilador de 10 MHz, que, através de divisores de frequência programáveis, interrompe a UCP a cada centésimo de segundo. A cada uma destas interrupções, não só a hora do sistema é atualizada, como também é feita uma consulta a uma lista de espera, para verificação da existência de alguma tarefa para ser pro-

cessada naquele instante.

Para a execução das várias tarefas que podem ser submetidas ao computador, o RTE-6 utiliza um esquema de prioridades que funciona da seguinte maneira:

Quando um programa é criado, a ele é atribuida, pelo usuário, uma "prioridade" representada por um parâmetro numérico que pode variar de 1 a 32767 sendo que, quanto menor o número, maior a prioridade. Em tempo de execução, o RTE dispara os programas na ordem de suas prioridades, ou seja, os programas mais prioritários são executados primeiro.

Para evitar que um programa de alta prioridade monopolize a máquina, a escala de prioridades é dividida em duas regiões. Na primeira região, os programas são executados na ordem estrita de suas prioridades. Na segunda, o sistema operacional se encarrega, automaticamente, de dividir o tempo entre programas de mesma prioridade, num processo conhecido como "time slice", dando a cada programa um "quantum" de tempo de 300 de milisegundos, "quantum" este que pode ser ajustado pelo gerente do sistema operacional, de acordo com as necessidades da instalação.

Em uma configuração típica do RTE-6, a divisão na escala de prioridades acontece na prioridade 50. Além disso, para garantir uma boa distribuição dos recursos da máquina, é atribuida a todos os programas submetidos pelos usuários "comuns", uma prioridade de

99 a qual somente o gerente do sistema pode alterar, para atender a necessidades especificas de alguma tarefa.

Dentro desse sistema de atendimento, as atividades de tempo real devem, necessariamente, ocupar a região privilegiada das prioridades, para garantir sua execução no exato instante em que forem necessárias.

O atendimento a "eventos imprevisíveis" é satisfeito pelo RTE-6 através do sistema de interrupção da UCP. Neste caso, a interrupção proveniente de um dado dispositivo de entrada e saída é respondida com a execução de um programa especifico, que satisfaça àquela interrupção.

Como exemplo deste recurso, tomemos o controle de um reservatório que deve ser enchido com um líquido até um dado nível, determinado por uma bóia. Quando o nível é alcançado, a bóia fecha um circuito que envia um pulso de controle para o computador e gera uma interrupção. O RTE atende a esta interrupção, colocando em execução, por exemplo, um programa que fecha a válvula de controle de entrada do líquido.

Como vemos, com esse sistema de atendimento, o computador não precisa de passar todo o tempo monitorando o evento que está por acontecer ou seja, ele pode gastar esse tempo executando outras tarefas de acordo com sua lista de prioridades.

Um outro recurso de grande utilidade existente no RTE-6 são as rotinas de entrada e saida por classe (CLASS I/O). Com elas, o sistema operacional pode ser encarregado da transmissão e/ou recepção de um bloco de dados, deixando o programa do usuário livre para execução de outras atividades. Com auxilio desse recurso, um determinado programa pode, por exemplo, processar um bloco de medidas B_n, enquanto o sistema operacional faz a aquisição do bloco B_{n+1} o que, em muitas aplicações, aumenta significativamente a eficiência do programa.

Do ponto de vista do "hardware", um instrumento de medição e controle que muito contribui para a eficiência dos processos de tempo real é o MULTIPROGRAMADOR. Esse instrumento, que apareceu no mercado há pouco mais de uma década, é constituido basicamente de um microprocessador ao qual estão conectadas várias interfaces - os chamados cartões funcionais - destinadas a efetuar operações específicas relacionadas, principalmente, com as atividades de aquisição de dados e controle de processos.

O multiprogramador segue de perto a filosofia dos sistemas de arquitetura aberta, já mencionados, pois o usuário pode configurá-lo de acôrdo com as suas necessidades, instalando apenas os cartões funcionais que precisar (Fig. 1-6). A quantidade de cartões que pode ser instalada, varia de instrumento para instrumento e em alguns modelos pode chegar a 255. Nos mais modernos existem ainda algumas facilidades adicionais tais como memória local expandida, acionadores de discos rígidos e flexíveis, etc. Os car-

tões mais comuns incluem conversores AD e DA, geradores de base de tempo, contadores, entradas e saídas digitais, reles, etc. O controle do instrumento é feito por comandos, enviados por um computador através de uma linha de comunicação, que permitem o acesso a qualquer um dos cartões funcionais instalados.

A grande vantagem do uso do multiprogramador reside no fato de que, com apenas uma linha de comunicação do computador de controle, o usuário pode ter acesso a um grande número de funções e assim delegar tarefas a processadores auxiliares deixando a CPU principal livre para outras atividades.

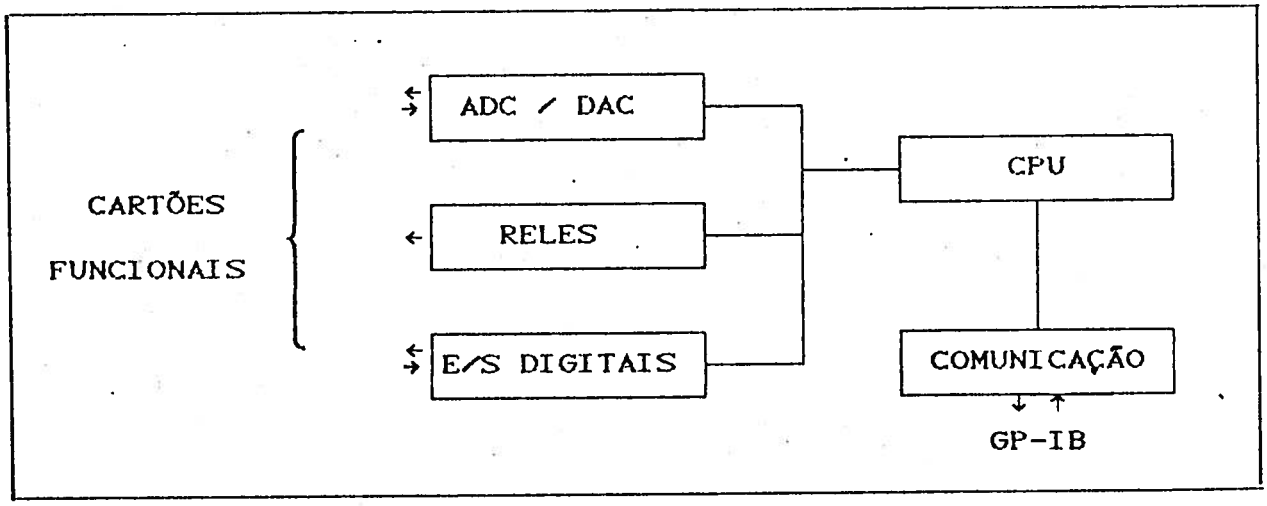


Fig. 1-6: Configuração típica de um multiprogramador

1.8 Os Sistemas de Arquitetura Aberta

Como mencionamos na Introdução deste trabalho, os Sistemas de Arquitetura Aberta representam uma alternativa bem mais econômica e muito mais criativa para a implantação de um sistema de aquisição de dados.

De um modo geral, qualquer que seja a complexidade do ~~sistema~~ do sistema a ser implantado, é fundamental que sejam mantidas a integridade e qualidade dos dados ao longo de todo o processo, para garantir o êxito da análise final desses dados.

Em um sistema típico de aquisição de dados, tal como representado na Fig. 1-7, o critério da "qualidade e integridade" deve ser usado, da melhor maneira possível, não apenas na escolha do computador e dos instrumentos como também na preparação e documentação dos programas de controle.

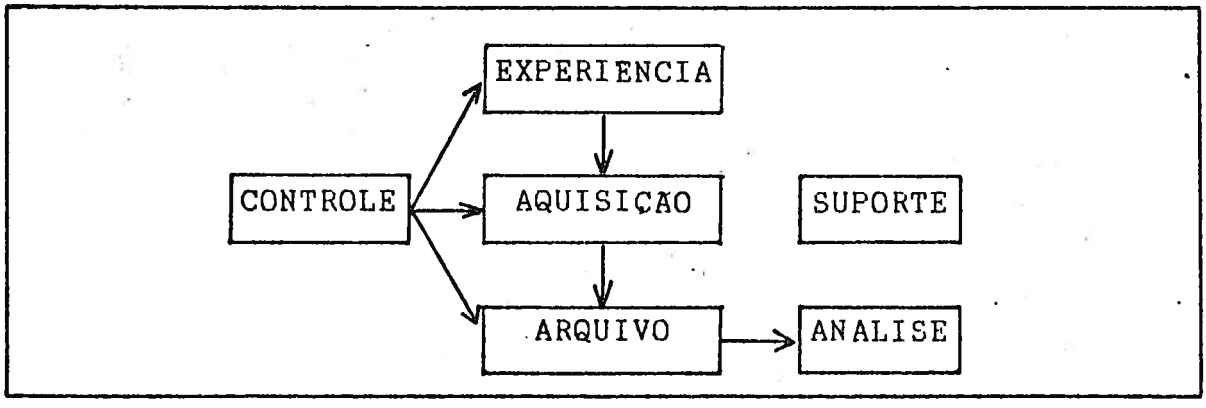


Fig. 1-7: Configuração de um sistema de aquisição de dados. A escolha dos instrumentos e/ou programas relativos a cada bloco deve ser feita de forma a garantir a qualidade dos dados.

Capítulo 2

SMAP: Sistema Melhorado de Aquisição Passiva

2.1 O Método da Aquisição Passiva

Um dos métodos mais simples de se efetuar a aquisição de dados de uma experiência é o Método da Aquisição Passiva. Nele, um instrumento adequado é ligado à saída de dados da montagem, com o objetivo de registrar os dados experimentais mas sem exercer qualquer tipo de controle ou influência sobre o andamento do processo. Um exemplo comum deste tipo de aquisição é a conexão de um termopar a um registrador X-t para produção de curvas de temperatura em função do tempo.

Em um espectrômetro de EPR, a saída de dados consiste basicamente de dois sinais:

1 - sinal de ressonância, que é obtido do detetor sensível à fase - "lock-in".

2 - sinal de intensidade do campo magnético, geralmente na forma de uma rampa de tensão.

Nesse caso, a implementação básica desse método consiste no uso de um registrador X-Y, cujas entradas X e Y são conectadas, respectivamente, ao sinal do campo magnético e ao sinal de ressonância (Fig. 2-1).

Embora amplamente utilizada ao longo dos últimos anos e até mesmo nos dias de hoje, a grande limitação dessa implementação é que, terminada uma série de medidas, o pesquisador tem como resultado uma coleção de gráficos dos quais terá de extrair, através de processos manuais, os dados numéricos de que necessitar.

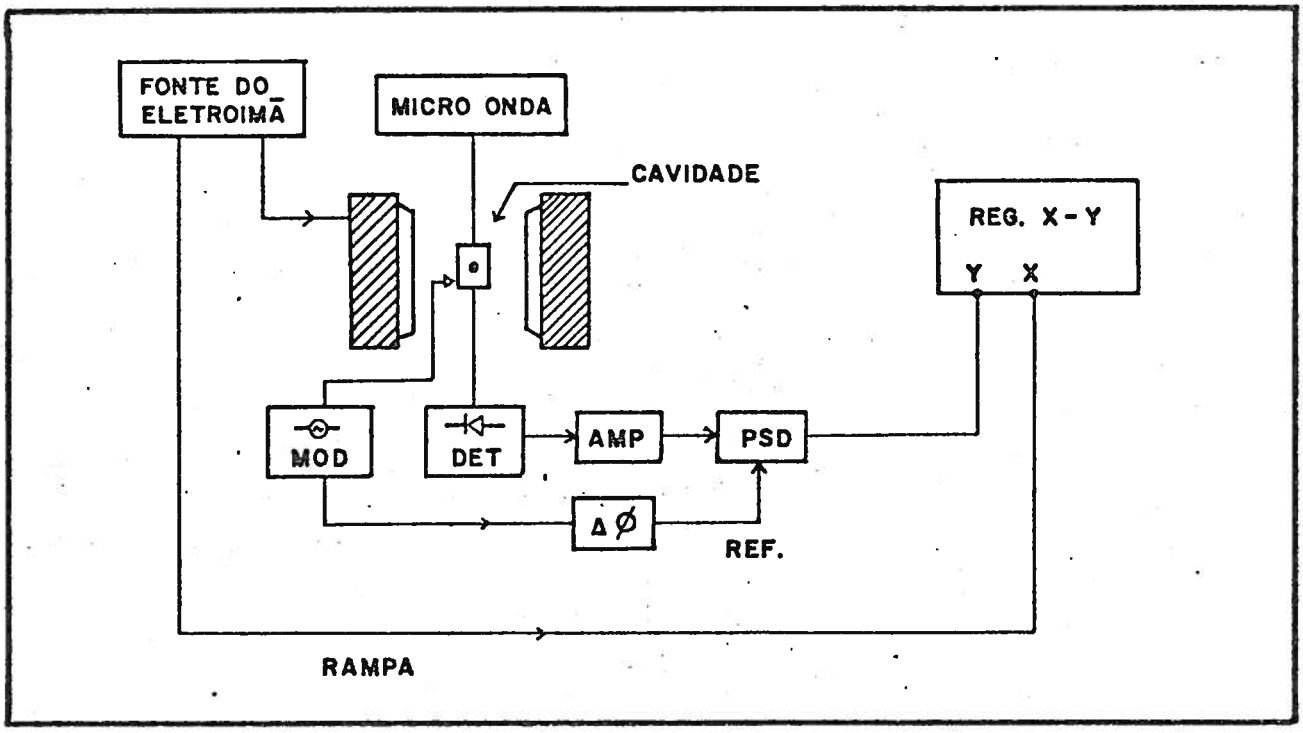


Fig. 2-1: Configuração Básica de um Espectrômetro de EPR

Os primeiros passos para automatizar as medidas de EPR foram dados com a criação de sistemas de aquisição passiva, capazes de registrar os espectros de uma maneira mais eficiente e, antes mesmo do aparecimento dos microcomputadores, analisadores multicanal foram usados, com sucesso, em sua implantação[7].

A idéia básica desses sistemas de aquisição passiva consiste em substituir o registrador X-Y por um computador provido de conversores analógico/digital e de dispositivos de armazenamento e/ou transmissão de dados (Fig. 2-2). Esse computador, através de sinais de controle, faz com que os conversores sejam disparados simultaneamente, em intervalos de tempo regulares e lê, para cada um destes disparos, um par de números que correspondem às intensidades do campo magnético e do sinal de ressonância naquele instante. Terminada a aquisição, esses pares de valores podem ser processados imediatamente ou então ser transmitidos para um computador remoto, de maior porte, para processamentos mais complexos tais como ajustes de forma de linha, simulações, etc.

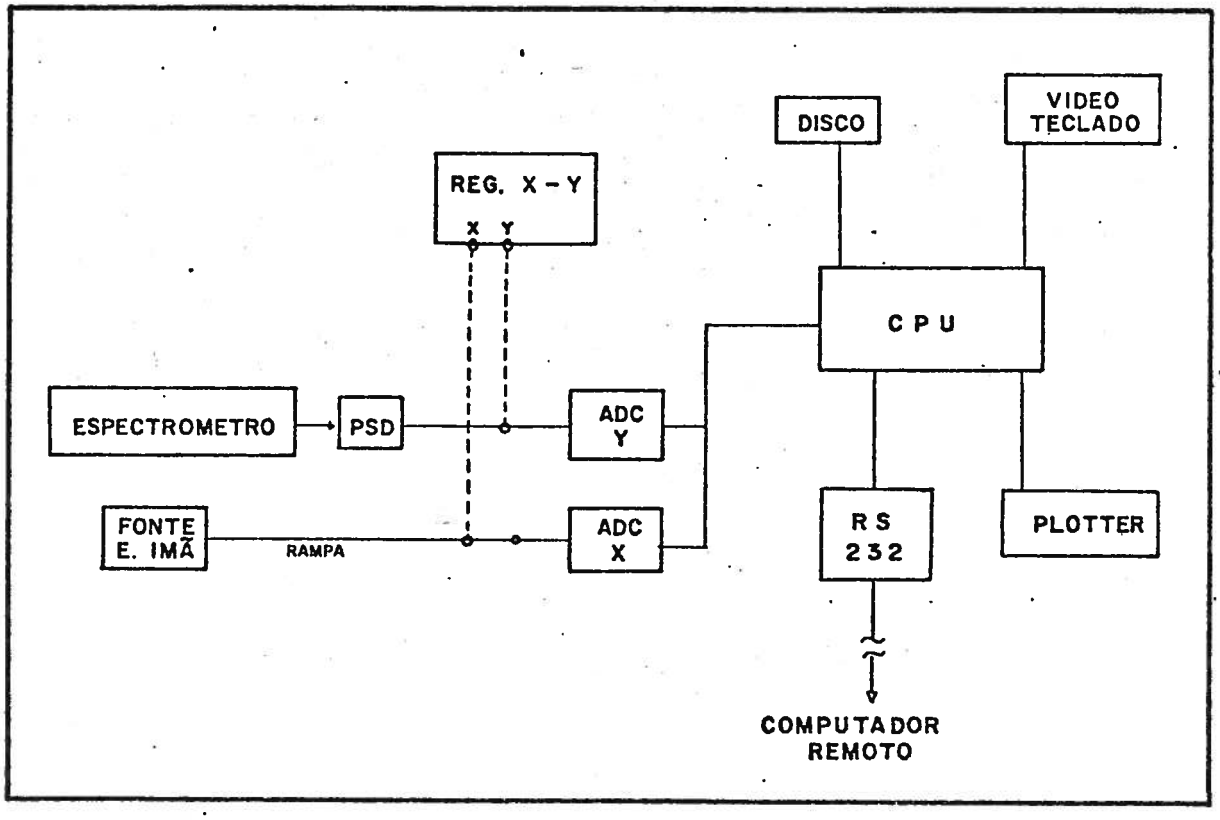


Fig. 2-2: Sistema Básico de Aquisição Passiva

2.2 O Sistema Melhorado de Aquisição Passiva

Um dos sistemas que tivemos a oportunidade de desenvolver para uso no Laboratório de Ressonância Magnética foi o que chamamos de Sistema Melhorado de Aquisição Passiva (SMAP). Esse sistema foi idealizado com base no conceito de arquitetura aberta[1] e procuramos, em seu desenvolvimento, explorar ao máximo as possibilidades oferecidas pelo computador de controle, o HP 1000 e pelo sistema operacional de tempo real RTE 6-VM.

As principais vantagens apresentadas pelo SMAP com relação a outros sistemas de aquisição passiva[5,8] são as seguintes:

- possibilidade de se escolher, de maneira simples e precisa, a região do espectro a ser armazenada na memória do computador.
- possibilidade de se executar o processamento de um espectro (um ajuste de forma de linha, por exemplo), enquanto se faz a aquisição de um outro espectro.
- possibilidade de se controlar uma série de outros instrumentos para aquisição de dados complementares, relativos ao desenvolvimento da experiência.
- facilidade de se usar o programa de controle que, por ter sido escrito nos moldes de um sistema operacional, interage com o operador através de um sistema de comandos mnemônicos.
- facilidade de se incorporar novos recursos ao sistema, decorrente do uso do FORTRAN 77 e de técnicas de programação

estruturada no programa de controle, e do emprego do barramento IEEE-488 na interconexão dos instrumentos.

Para que se tornasse possível a obtenção dessas vantagens, vários instrumentos foram incorporados ao espectrômetro conforme se vê na Tabela 2-1. Todos esses instrumentos foram interligados ao computador de controle através do barramento IEEE-488.

Instrumento	Modelo
Conversor analógico-digital de 4 canais	HP 59313A
Frequencímetro de micro-ondas	HP 5342A
Goniômetro motorizado	Micro Controle IT6D
Multiprogramador	STD 85MP
Multímetro digital	Keithley 195A
Gaussímetro	Varian E 500
Gaussímetro	Metrolab PT 2020
Extensor HP-IB	HP 59301A

Tabela 2-1: Instrumentos usados pelo SMAP

Para se entender o funcionamento do SMAP, é necessário que se faça inicialmente uma breve discussão sobre o funcionamento normal do espectrômetro de EPR, no que diz respeito à obtenção de um espectro no registrador X-Y.

Supondo-se que toda a instrumentação esteja pronta para medir, ou seja, temperatura da amostra estabilizada, cavidade sintonizada, amplificador "lock-in" ajustado, etc., são os seguintes os procedimentos típicos para a obtenção de um espectro:

Procedimento	Controle da fonte(*)
- ajusta-se o valor central do campo	Field Selector
- ajusta-se a faixa de varredura	Sweep Range
- ajusta-se a duração total da varredura	Sweep Time
- ajusta-se o valor inicial da varredura	Sweep
- dispara-se a medida	Sweep Mode

(*) Fonte Varian Fieldial Mark I

Tabela 2-2: Operação do Espectrômetro

Uma vez disparado o processo, dois sinais serão enviados para o registrador X-Y. No eixo dos Y, teremos o sinal de ressonância e no eixo dos X, teremos um sinal proporcional ao valor do "SWEEP". Como o controle de varredura, "SWEEP", é acionado internamente por um motor sincrono, o eixo dos X terá um movimento linear e a figura traçada será então uma representação do sinal de ressonância em função do campo magnético. Na fonte Varian utilizada em nosso sistema, o sinal correspondente à variação do campo é uma rampa de tensão que varia de 0 a -10 volts para variações do "SWEEP" entre -50% e +50% do valor central do campo, tal como representado na Fig. 2-3.

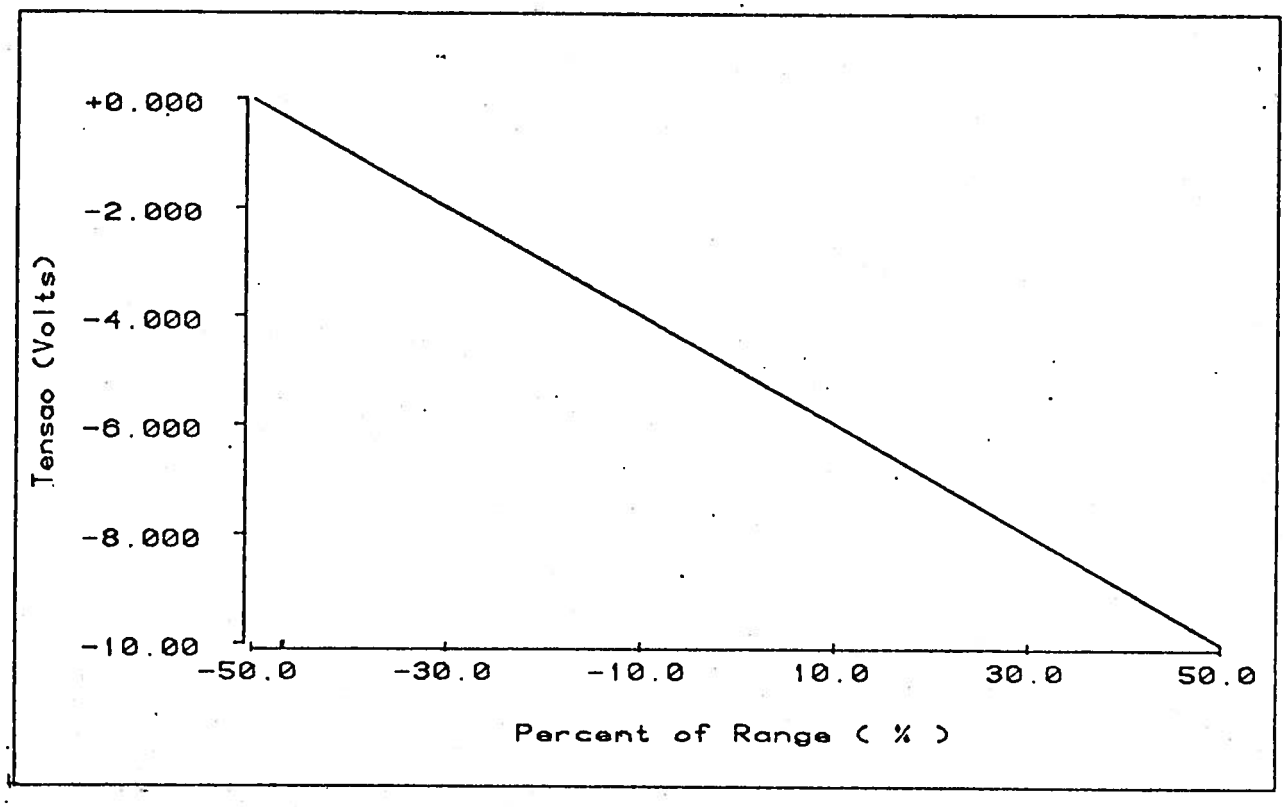


Fig. 2-3: Rampa de Varredura da fonte Varian

Do ponto de vista do sistema de aquisição passiva, o objetivo básico é o registro, da melhor maneira possivel, dos sinais representativos da ressonância e da intensidade do campo magnético mas, no SMAP, incorporamos alguns recursos adicionais para o controle da varredura e para a conexão de outros instrumentos auxiliares.

Para facilitar a compreensão deste sistema como um todo, vamos apresentar inicialmente uma descrição das técnicas usadas para o registro das intensidades do campo magnético e da ressonância e para o controle da varredura.

2.2.1 Registro da Intensidade do Campo

A idéia inicialmente desenvolvida para o registro das intensidades do campo magnético era usar um gaussimetro, conectado diretamente ao computador. Essa solução mostrou-se ineficaz porque os gaussimetros disponiveis, que funcionam a partir da técnica de NMR, não são capazes de acompanhar a varredura do campo, perdendo, frequentemente, o sincronismo com o sinal de NMR. Por isso, tivemos de resolver o problema da medição do campo a partir da tensão da rampa de varredura.

Como mencionamos anteriormente, essa rampa de tensão, gerada pela fonte do eletroimã, varia de 0 a -10 Volts para valores do controle "Percent of Range", compreendidos entre -50% e +50% do valor central do campo. Deste modo, a relação entre a indicação deste controle, PR, e a tensão da rampa, **V**, será dada por

$$PR = -10 * (V + 5) \qquad (2.1)$$

Sendo H_0 o valor central do campo, escolhido no controle "Field Select", o valor do campo para um ponto qualquer do "Sweep Range", **SR**, pode ser obtido de:

$$H = H_0 + (PR / 100) * SR \qquad (2.2)$$

ou, considerando a equação (2.1),

$$H = H_0 - (V+5)*(0,1*SR) \qquad (2.3)$$

A equação (2.3) nos permite conhecer o campo magnético a partir da tensão da rampa e dos valores previamente ajustados para o campo ("Field Select") e para a varredura ("Sweep Range") e, embora nos forneça um caminho bastante simples para a medição do campo magnético, seu resultado não é totalmente confiável em consequência do mecanismo usado para geração da rampa, no qual um potenciômetro de alta precisão e boa linearidade é mecanicamente acoplado, através de engrenagens, ao eixo do motor síncrono que comanda a varredura do campo. Esse potenciômetro, por sua vez, é ligado na forma de um divisor de tensão a uma fonte de referência ajustada para -10 Volts. Quando a varredura é iniciada, o movimento do motor síncrono é transmitido a esse potenciômetro e a rampa é obtida na saída do divisor de tensão (Fig. 2-4).

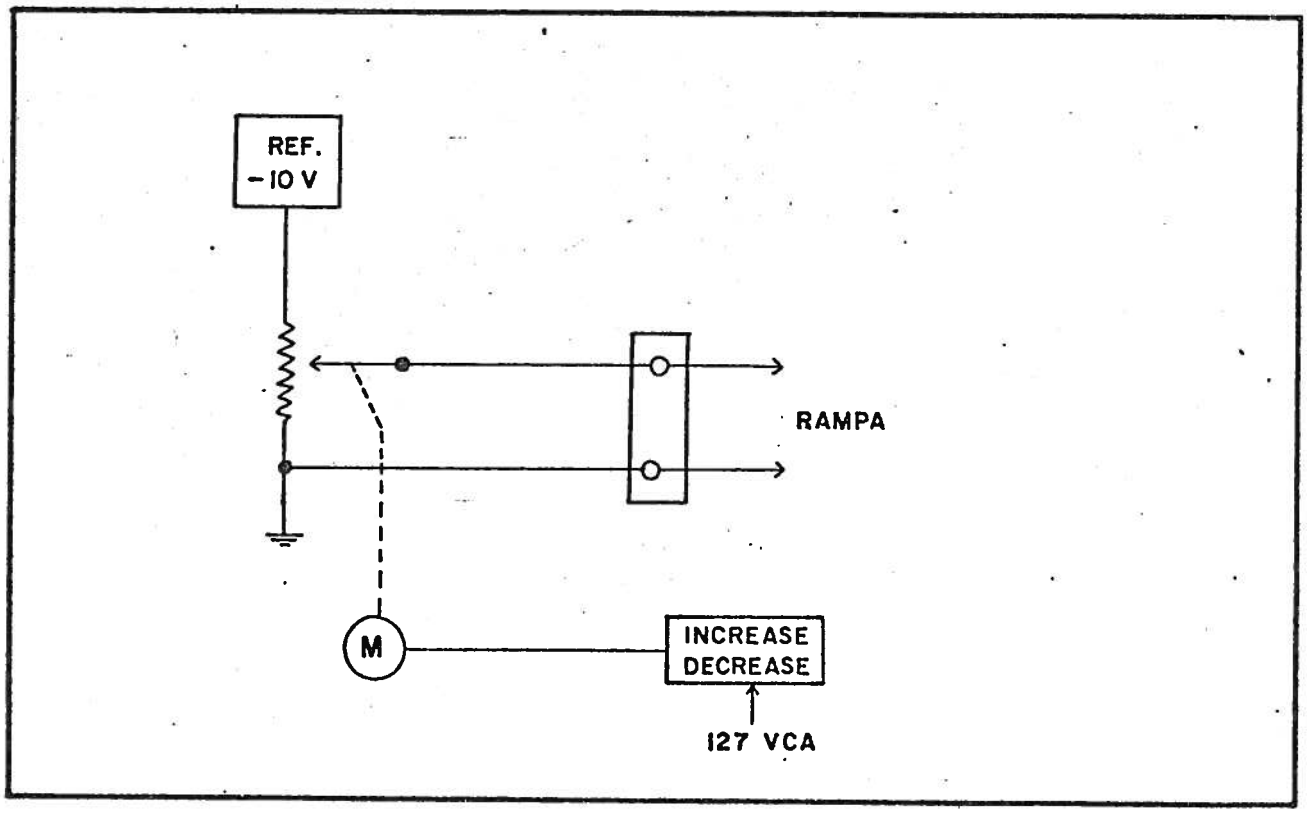


Fig. 2-4: Gerador da rampa.

Devido a folgas no acoplamento, desajustes na fonte de -10V e incertezas existentes em circuitos internos de calibração da fonte, o valor de H dado pela equação (2.3), pode não corresponder à realidade.

Para contornar essas dificuldades e ainda assim usar a tensão da rampa para medir o campo, adotamos no SMAP a seguinte técnica, que chamamos de calibração da rampa:

Antes de iniciar uma série de medidas de espectros, são feitas, com auxílio do gaussimento, duas ou mais medidas do campo e da tensão da rampa em pontos distintos do controle de "sweep range". Para atenuar o efeito de eventuais ruidos na medida da tensão, é tomada a média de 100 leituras consecutivas do ADC.

Os pares de pontos (H_n, V_n), obtidos nestas medidas, são usados então para calcular os parâmetros **a** e **b** da reta $H = aV + b$. Esses parâmetros, uma vez calculados, são mantidos na memória e usados em todos os cálculos subsequentes de **H(V)**, até que se faça uma nova calibração.

Para nos certificarmos da validade desta técnica, foi montado um sistema de medida composto de um ADC e um gaussimetro e controlado pelos programas BURST e RMU01 (Ap. 4).

O programa BURST, que simplesmente registra os valores da tensão da rampa em função do tempo, foi usado com o objetivo de se veri-

ficar a linearidade da rampa. Os dados obtidos por esse programa estão representados na Fig. 2-5. Como se pode observar, a rampa é de boa qualidade e apresenta dois pequenos patamares de tensão constante que não contribuem para a variação do campo, mas são convenientes para o ajuste e verificação das tensões nas extremidades.

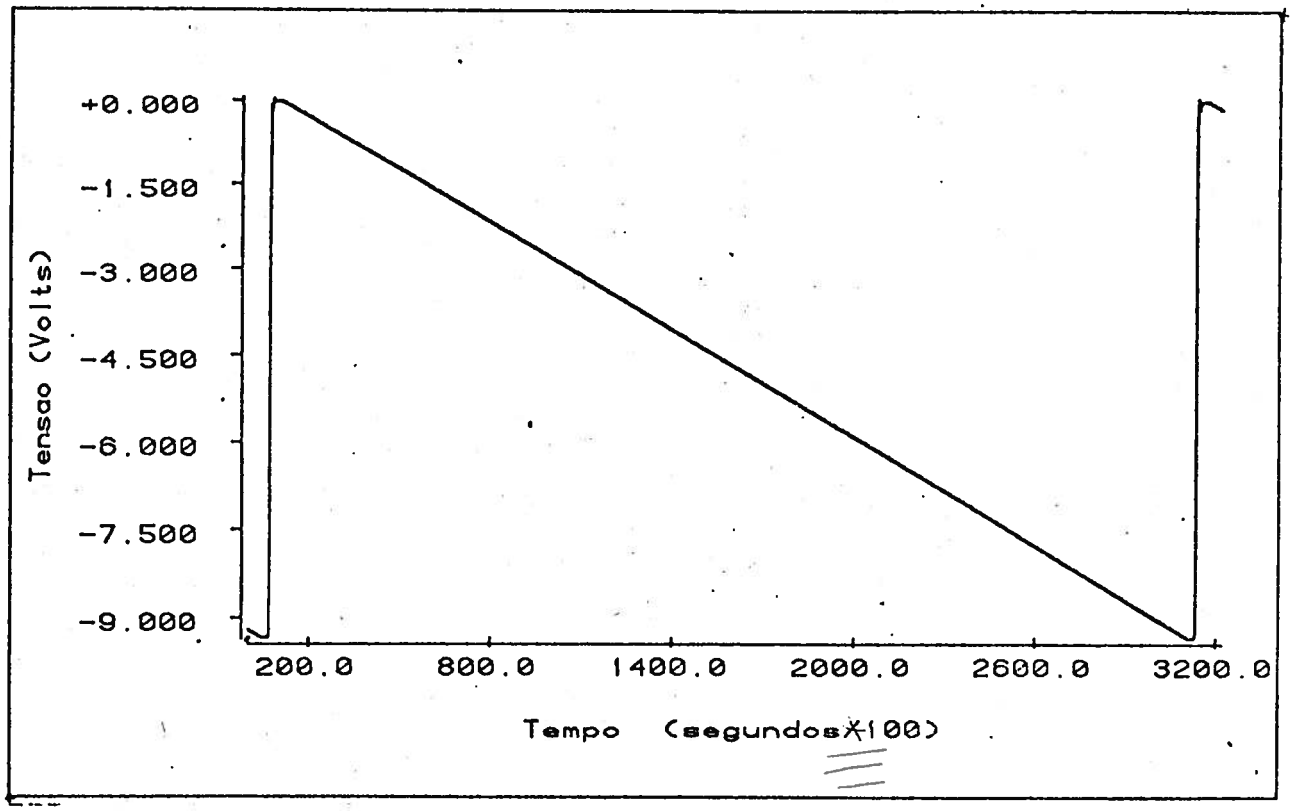


Fig. 2-5: Rampa do Eletroimã

O programa RMU01, por sua vez, permite que o valor do campo medido através da tensão da rampa seja comparado com o valor obtido diretamente pelo gaussimetro.

Para medir o campo através da tensão da rampa, o programa solicita inicialmente que seja feita uma calibração. Para tanto, o controle de "Percent of Range" é colocado sequencialmente em dois pontos distintos de sua escala e, para cada um destes pontos, são feitas medidas da tensão da rampa com o ADC e do campo magnético com o gaussímetro. Neste caso, como o campo é estacionário, não existe o problema da perda de sincronismo do gaussímetro, discutida anteriormente.

Os valores obtidos nestas medidas, (H_1V_1) e (H_2V_2) são usados então para a determinação dos parâmetros **a** e **b** da reta

$$H(V) = aV + b \qquad (2.4)$$

Feita esta calibração, o campo H pode então ser determinado para qualquer ponto do "Percent of Range", a partir de uma medição da tensão da rampa, **V**.

O programa RMU01 permite que se faça até 100 medições em pontos escolhidos aleatoriamente pelo operador.

Terminado o processo, os resultados são impressos em três colunas que representam, respectivamente, o campo medido pelo gaussímetro (HG), o campo medido pela rampa (HR) e a diferença entre as medidas (HR - HG).

Em nosso sistema, fizemos várias medidas deste tipo e um resumo dos resultados é apresentado na Tabela 2-3.

Conforme se pode observar, as diferenças HR - HG são da ordem de 1 parte em 1000, o que consideramos perfeitamente satisfatório para medidas de EPR. O conjunto completo dessas medidas encontra-se no apêndice 5, juntamente com a listagem do programa RMU01.

==========

Região: 2778.2 a 2267.7 gauss

Medida	H_G	H_R	$H_R - H_G$
01	2778,2	2777,5	-0,7
10	2732,2	2733,3	1,1
20	2685,1	2681,3	-3,8
30	2630,5	2631,0	0,5
40	2578,5	2578,4	-0,1
50	2527,5	2527,5	0,0
60	2475,1	2475,5	0,4
70	2423,0	2423,5	0,5
80	2371,2	2371,5	0,3
90	2319,5	2319,5	0,0
100	2627,7	2627,0	-0,7

Resultados (para 100 medidas):

Erro médio:..... -0,101

Desvio padrão:.... 1,098

==========

Tabela 2-3: Resumo das medidas comparativas da intensidade do campo magnético- Programa RMU01

2.2.2 Registro do Sinal de Ressonância

O sinal de ressonância é obtido do espectrômetro através da conexão da saída do amplificador "lock-in" à entrada do ADC que, em nossa montagem é um subsistema HP-IB (HP59313A) e incorpora, além do conversor A/D propriamente dito, os seguintes circuitos auxiliares:

a) amplificadores de entrada
b) gerador de base de tempo
c) multiplexador analógico
d) interface HP-IB (IEEE-488)
e) canal reverso
f) lógica de disparo externo
g) logica de controle.

Os amplificadores de entrada, em número de 4, são responsáveis pelo condicionamento do sinal. Eles dispõem de controles de ganho e de zero através dos quais se faz a calibração do instrumento.

O gerador de base de tempo (10 MHz) prove os sinais de sincronismo para o instrumento. Desse mesmo gerador, através de divisores programáveis, são retirados os pulsos que controlam a taxa de conversão.

O multiplexador analógico, também programável, determina qual dos amplificadores de entrada vai fornecer o sinal para a conversão.

A interface HP-IB implementa a norma IEEE-488 e é o elo de ligação do instrumento com o computador de controle.

O canal reverso é composto de um transistor NPN, montado em configuração de coletor aberto, que pode ser colocado nos estados de corte e condução através de comandos do computador, permitindo o controle de dispositivos externos do tipo liga-desliga.

A lógica de disparo externo, **external start,** é um circuito que permite sincronizar o início das conversões com um evento externo qualquer, que possa ser traduzido por uma transição de níveis TTL.

A lógica de controle sincroniza toda a operação do instrumento. Ela controla a interface HP-IB, decodifica as instruções recebidas do computador, e formata os dados de saída.

As características básicas deste instrumento estão resumidas na tabela abaixo:

Número de entradas:	4, bipolares
Fundo de escala:	+/- 1,0 2,5 5,0 e 10,0 volts
Resolução:	11 bits (10 bits + polaridade)
Tempo de conversão:	4,75 ms
Tempo de amostragem:	2,048 ms
Taxas de conversão:	5, 10, 20, 50, 100 e 200 cps
Formato dos dados:	binário, complemento de 2

Tabela 2.4 - Características Básicas do ADC HP 59313A

A conversão A/D se processa pelo método da rampa dupla que consiste basicamente de dois estágios:

1) A tensão desconhecida V_e é aplicada à entrada de um circuito integrador, cuja tensão inicial de saída V_o é zero, durante um intervalo de tempo fixo **TF** determinado pela base de tempo do conversor. Ao final deste intervalo, a tensão de entrada é desligada e a saida do integrador estabiliza-se em uma tensão dada por

$$V_o(TF) = -(1/RC) \int_0^{TF} V_e(t)dt$$

ou, quando V_e for constante, por

$$V_o(TF) = -(1/RC)V_eTF \qquad (2.5)$$

onde R e C são constantes do integrador.

2) Uma tensão de referência, de valor conhecido V_r e polaridade oposta à tensão V_e, é então aplicada ao integrador e o tempo gasto para que o mesmo retorne à sua condição inicial, **TD**, é armazenado em um contador, ou seja,

$$0 = V_o(TF) - \{ -(1/RC)V_rTD \} \qquad (2.6)$$

Comparando as eq. (2.5) e (2.6) podemos ver que

$$-(1/RC)V_eTF = -(1/RC)V_rTD$$

ou que,

$$V_e = V_r\ (TD/TF)$$

Como a base de tempo do conversor é constante durante todo o processo, o resultado acima pode ser reescrito como

$$V_e = V_r\ (ND/NF) \qquad (2.7)$$

onde NF é o número de contagens do tempo fixo e ND é o número de contagens da fase de descarga do integrador. Como se pode ver, o resultado independe dos valores de R e C o que é uma caracteristica muito importante desse método de conversão.

Com relação ao processo de conversão, devemos notar ainda que, em virtude da integração do sinal, as componentes de "alta frequência" serão praticamente eliminadas do resultado final. Nesse aspecto, o integrador tem um comportamento semelhante a um filtro conforme se pode ver na Fig 2-6, onde mostramos o resultado da integração em função da frequência do sinal de entrada. E justamente esse comportamento que faz com que esse ADC seja relativamente insensível às "altas frequências".

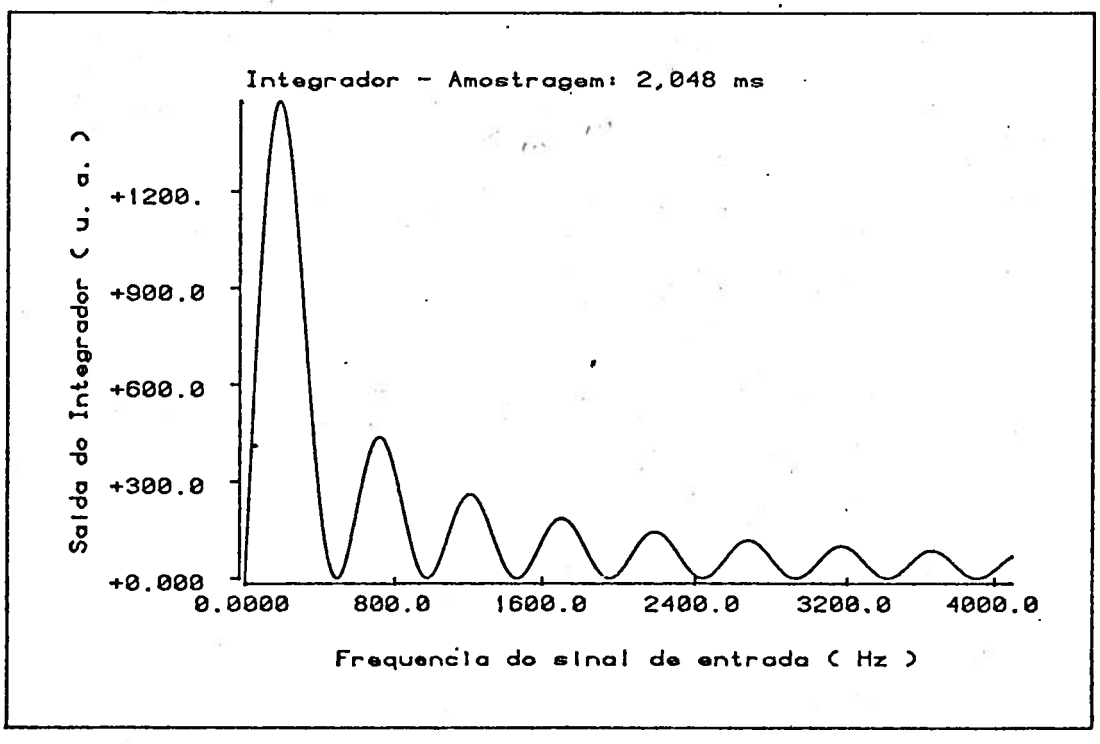


Fig 2-6: Integrador - resposta à frequência

2.2.3 Controle da Varredura do Campo

Como vimos, a varredura do campo magnético é controlada por um motor síncrono. Este motor, além de acionar o potenciômetro de geração da rampa, aciona também um outro potenciômetro que faz parte do circuito de controle do campo e é o componente responsável pela variação do campo (varredura).

A direção de rotação do motor - determinada pelos capacitores presentes no circuito - é controlada por uma chave de duas posições, localizada no painel frontal da fonte, que permite a reversão do movimento, fazendo com que o campo possa ser incrementado ou decrementado ("Increase", "Decrease"), de acordo com as necessidades da experiência.

Para colocar esse circuito sob o controle do computador, foram instalados dois relés, segundo o diagrama da Fig. 2-7. Conforme se pode observar, os contatos dos relés estão em paralelo com os contatos da chave, de modo que o acionamento de RL1 inicia a varredura no sentido "Increase" e o de RL2, no sentido "Decrease".

Os contatos adicionais dos relés são usados para inibir os contatos da chave original, evitando assim que um acionamento acidental da mesma durante uma sessão de medições possa comprometer o funcionamento do sistema.

O acionamento de RL1 e RL2 é comandado por dois relés de baixa potência, contidos em um dos cartões funcionais do multiprogramador. Este sistema de "relé acionando relé", foi usado como uma medida extra de segurança para aumentar a isolação entre o sistema de computação e a fonte do eletro-imã, pois a tensão comutada por RL1 e RL2 é de 127 Vca.

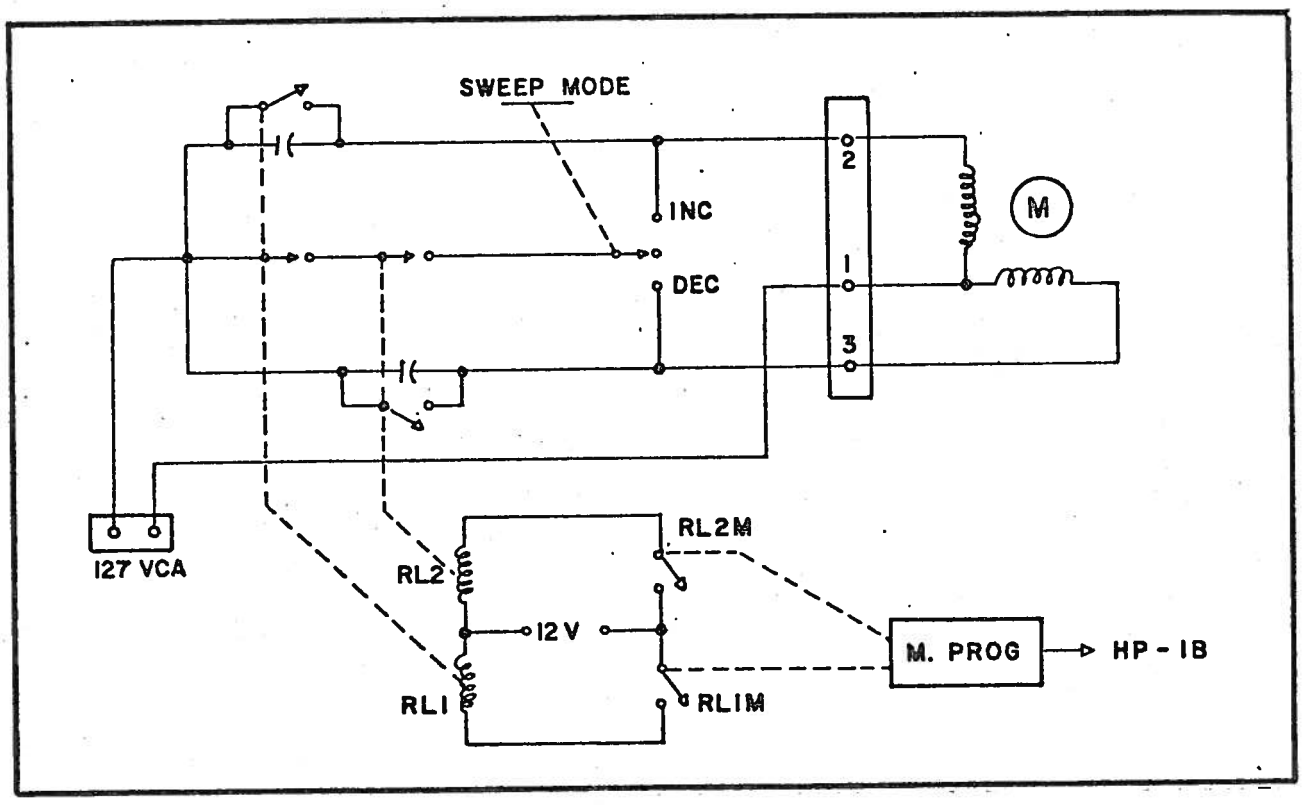


Fig. 2-7: Circuito de Controle da Varredura

A operação dos relés é feita através das rotinas RELE e RELAY (Ap. 4), que foram escritas para esta aplicação mas constituem também um exemplo das técnicas de programação do cartão funcional de relés do multiprogramador.

2.2.4 O Registro dos Espectros

Todo o processo de registro dos espectros é controlado pelo programa SMAP. Este programa, cuja listagem se encontra no apêndice 2, interage com o usuário através de uma linguagem de comandos parametrizados, bem semelhante a um sistema operacional.

Com o uso dos comandos do programa são feitos os ajustes prèliminares do sistema, a aquisição dos dados e também a gravação de arquivos, exibição de gráficos, etc.

A aquisição de dados é feita com base nos métodos já descritos neste capítulo e de acordo com a teoria exposta no capítulo 1.

No tocante aos instrumentos usados, a maior dificuldade que enfrentamos foi devida ao fato de que o ADC HP 59313A, embora dispondo de 4 canais de entrada, só permite a operação de 1 canal em cada conversão, tornando-se impossível, por isso, a aquisição simultânea dos dados relativos às intensidades do campo magnético e do sinal de ressonância. Para contornar esta dificuldade idealizamos uma outra maneira de fazer as medidas, a partir da montagem representada na Fig. 2-8.

Nessa montagem, o sinal da rampa é ligado permanentemente ao canal 4 do ADC. O sinal de ressonância, que vem da saída do amplificador "lock-in", é ligado aos canais 1, 2 ou 3 que são calibrados com sensibilidades diferentes para acomodar sinais de maior ou menor intensidade.

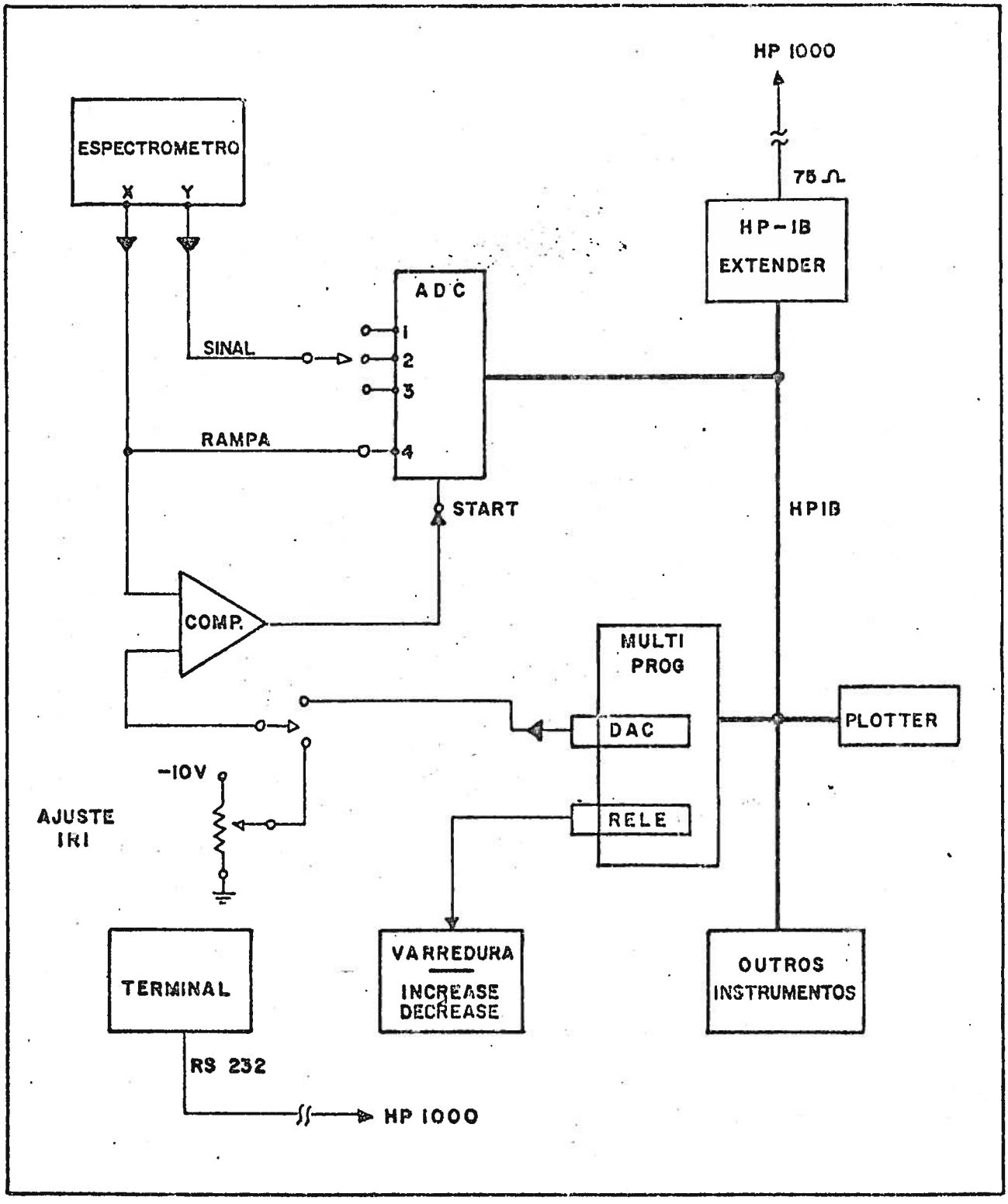


Fig. 2-8: Diagrama Básico do SMAP

Para disparar o processo de medida e assegurar que a aquisição seja iniciada em um valor bem determinado do campo magnético, foi montado um comparador analógico, cuja saída foi conectada ao "external start" do ADC e cujas entradas foram ligadas, respectivamente, ao sinal da rampa (canal 4) e a uma fonte de tensão ajustável de 0 a -10 Volts. Essa fonte de tensão pode ser escolhida pelo operador entre as saídas de um divisor de tensão, baseado em um potenciômetro de 10 voltas, ou de um conversor D/A de 12 bits instalado no multiprogramador.

Um indicador luminoso (LED), ligado à saída do comparador, informa se a tensão da rampa está acima ou abaixo da tensão ajustada pelo operador e e' justamente esta tensão que determina, no sistema SMAP, o que chamamos de Inicio da Região de Interesse (IRI).

Para que essa técnica funcione corretamente, é necessário que antes do inicio da aquisição de um espectro, o controle de "sweep" seja ajustado de forma que a tensão da rampa fique um pouco acima do ponto de transição do comparador, antes portanto do IRI. Depois deste ajuste o processo é iniciado e a chave da varredura é ligada na posição de campo crescente ("Increase") pelo operador ou, opcionalmente, pelos relés do multiprogramador.

Assim que a varredura começa a funcionar, a tensão da rampa diminui no sentido de -10 V. Ao passar pelo ponto ajustado no comparador (IRI), uma transição aparecerá na saída do mesmo, disparando o relógio do ADC que começa a fazer as conversões programadas.

Este procedimento nos permite disparar de uma maneira bem determinada o início das conversões, mas não resolve o problema da medição simultânea do campo e do sinal de ressonância. Dada a impossibilidade (econômica) do uso de um outro ADC, resolvemos então abordar o problema usando a seguinte técnica:

> No início do processo de aquisição, ou seja, na primeira conversão, é feita uma medição da tensão da rampa (canal 4). Imediatamente após esta medição, o canal de entrada do ADC é trocado, de modo a tornar ativo o canal que se encontra ligado à saída do "lock-in", e são realizadas N_p medições da tensão do espectro, onde N_p é um número pré-determinado pelo usuário. Imediatamente após a N_p-ésima medição, o canal 4 é novamente ativado e é feita uma última medição da tensão da rampa.

No final desse processo, teremos um conjunto de N_p+2 pontos, onde os pontos inicial e final referem-se à tensão da rampa e os demais, ao espectro. Como a rampa varia linearmente com o tempo e a taxa de aquisição é uniforme, os N_p+2 pontos estarão uniformemente espaçados no tempo e consequentemente os N_p pontos do espectro estarão uniformemente espaçados com relação ao campo magnético.

A obtenção dos valores do campo para cada ponto do espectro torna-se então um processo bastante simples, conforme vimos na sec. 2.2.1.

Para o desenvolvimento dessas ideias no programa SMAP, tivemos de lançar mão do procedimento especial de entrada de dados, chamado de "CLASS READ" (sec. 1.7), além de técnicas mais comuns como acesso direto à memória, etc., conforme mostraremos a seguir.

Na fase de ajustes e inicialização do programa, são tomadas algumas decisões relacionadas principalmente com os parâmetros da experiência:

- A duração TOTAL da varredura é passada ao programa através do comando TV.
- O comprimento da região de interesse, RI, medido em unidades de "Percent of Range" é passado pelo comando PV.
- A taxa de conversão do ADC, f, é escolhida pelo comando FC.
- Um canal do ADC com sensibilidade adequada ao sinal proveniente do "lock-in" é escolhido com o comando CA.
- A rampa é calibrada através do comando CR tal como descrito na seção I deste capítulo.
- O comparador é ajustado pelo usuário de tal forma que a transição ocorra no incio da região de interesse (IRI).

A partir desse ponto, o programa prepara o ADC para as conversões e calcula o número de conversões a realizar através da relação

$$N = Tv * (RI/100) * f \qquad (2.8)$$

onde Tv = tempo total de varredura ("sweep time")

RI = comprimento da região de interesse (PV)

f = taxa de conversão, em Hz

Após estes cálculos e procedimentos, o programa estará apto a iniciar o processo de aquisição do espectro, o que acontecerá quando lhe for dado o comando **AQ**.

Uma vez recebido o comando AQ, a sequência de eventos que leva à aquisição do espectro é a seguinte: (Figs. 2-9 e 2-10)

- Como mais de um programa pode estar sendo executado pela CPU, a opção de "memory lock" é ativada para evitar que o SMAP seja retirado da memória por outro programa de maior prioridade e que esteja habilitado para tal ação.

- Os circuitos de DMA (Acesso Direto à Memória) são inicializados para garantir um fluxo mais rápido dos dados do ADC para a memória.

- O ADC é programado com os seguintes parâmetros:

  canal ativo: 4 (tensão da rampa)

  start : externo, via saída do comparador

  taxa : f (valor dado pelo comando FC)

Neste instante, ou seja, alguns milisegundos após o recebimento do comando AQ, a varredura já pode ser iniciada através do acionamento da chave "Increase", o que fará com que a tensão da rampa

comece a diminuir. Quando a tensão da rampa atingir o valor previamente ajustado no comparador (IRI), o ADC será disparado, marcando o início das aquisições (t = 0) e ter-se-á uma nova sequência de eventos que é a seguinte:

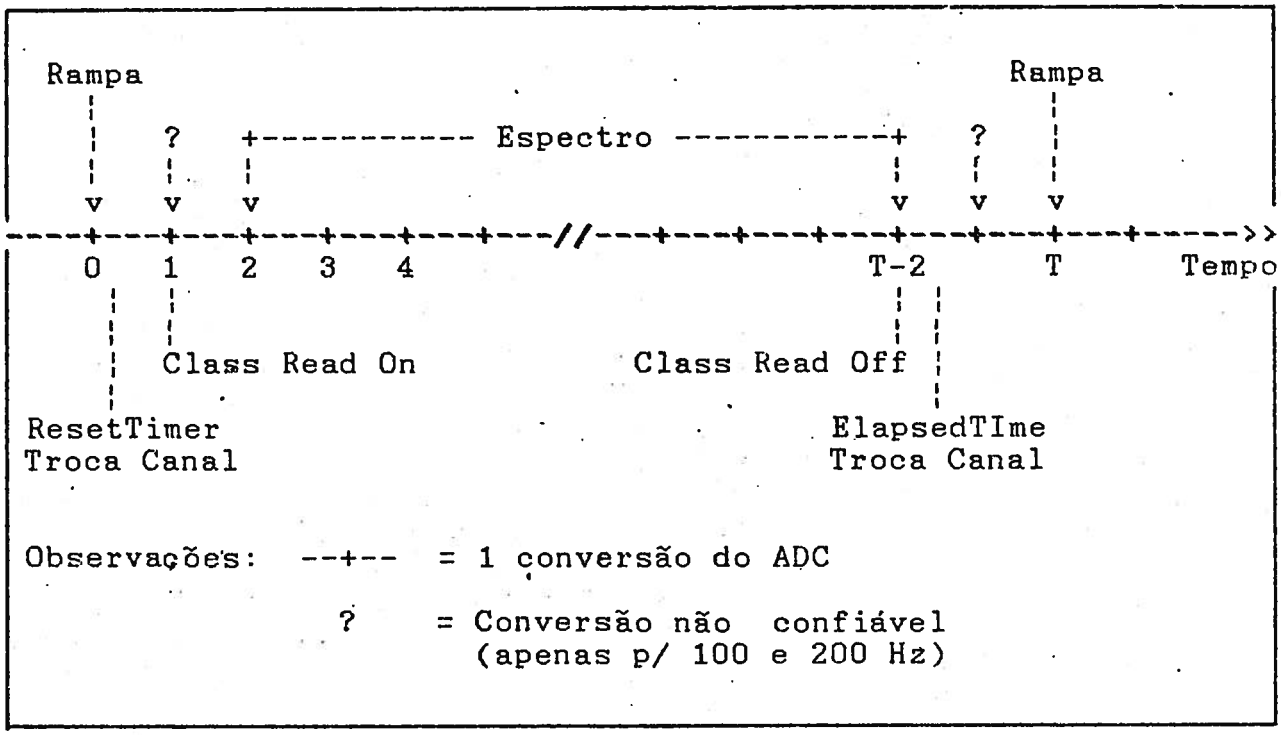


Fig. 2-9: Sequência de Eventos da Aquisição dos Dados

- No instante t=0 o ADC converte a tensão presente no canal 4 (rampa) e transmite o resultado para o computador.
- O computador recebe e salva esse dado e imediatamente reprograma o ADC, tornando ativo o canal que se encontra ligado ao "lock-in" (passado pelo comando CA) e dispara o relógio do sistema operacional para posterior verificação do tempo gasto na aquisição.

- Em seguida o computador chama a rotina de leitura por classe, passando-lhe o número de conversões a relizar. A partir deste ponto, o sistema operacional encarrega-se da leitura dos dados, em intervalos regulares, tendo como base de tempo o relógio do ADC. O programa SMAP é "retirado do ar" e colocado em estado de suspensão até que todos os pontos sejam lidos.

Terminada a leitura, o sistema operacional é avisado e coloca novamente "no ar" o programa SMAP, a partir do ponto em que ele havia sido suspenso. O SMAP ganha novamente o controle da aquisição e toma as seguintes providências:

- Salva os dados lidos pelo sistema operacional pois os mesmos, até este instante, encontram-se em uma região da memória que é de uso comum para todos os programas.
- Salva o conteúdo do relógio do sistema operacional.
- Troca novamente o canal ativo do ADC para o canal 4 e faz mais **duas** leituras da tensão da rampa.

Feitas essas leituras, a opção de "Memory Lock" e o sistema de DMA são desativados e o encerramento da aquisiçao é sinalizado no terminal do operador.

Se o programa estiver também controlando a varredura (modo AUTO) o programa auxiliar INDEC é despachado para acionar os relés do multiprogramador e reposicionar a varredura no ponto **IRI**. Caso a operação esteja ocorrendo no modo MANUAL o operador deve tomar a

iniciativa de desligar a varredura e reposicionar os controles.

Já com os dados colocados sob seu controle e não mais na memória temporária usada na operação de leitura por classe, o programa verifica se o fundo de escala do ADC foi atingido ou seja, se houve saturação, e informa quantas vezes esta situação ocorreu. Caso o fundo de escala não tenha sido atingido, o programa calcula a leitura máxima do ADC e informa ao usuário para que ele possa ter uma idéia da ocupação da faixa dinâmica do ADC. No caso de saturação, cabe ao usuário verificar se existe algo de aproveitável no espectro como, por exemplo, uma linha de baixa intensidade, etc.

Como o processo de leitura por classe só permite a inspecção dos dados após a leitura de todos os pontos, torna-se impossível detectar uma saturação ao longo das medições. Por isso, o usuário deve ser cuidadoso e escolher um canal do ADC que tenha uma sensibilidade suficiente para acomodar a maior intensidade existente no espectro. Para facilitar essa escolha, o SMAP dispõe do comando AD que permite a realização de conversões "avulsas", em qualquer canal, independentes do comando de aquisição.

Para completar o processo de aquisição, falta ainda a informação da intensidade do campo magnético correspondente a cada ponto do espectro. Esta informação é obtida da seguinte maneira:

Conforme se pode observar na Fig. 2-9, são feitas leituras da tensão da rampa nos instantes **t = 0** e **t = T** (as outras leituras serão discutidas posteriormente).

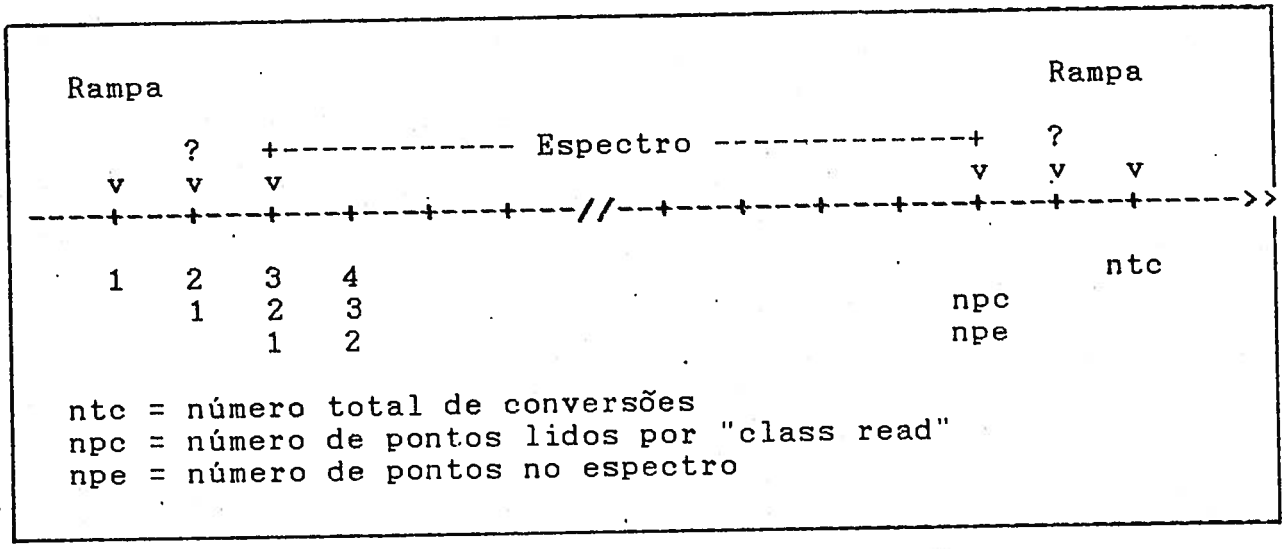


Fig. 2-10: Sequência de Eventos na Aquisição dos Dados II

A partir dos dados obtidos na calibração da rampa (comando CR) o campo nos instantes t=0 e t=T pode ser calculado por

$$B(0) = BI = a * IVI + b \qquad (2.9)$$

$$B(T) = BF = a * IVF + b \qquad (2.10)$$

onde

a, b = parâmetros da rampa

IVI = tensão da rampa em t=0

IVF = tensão da rampa em t=T

BI = campo no início da aquisição (t=0)

BF = campo no final da aquisição (t=T)

Ainda com referência às Figs. 2-9 e 2-10, pode-se observar que as conversões feitas nos instantes **t=1** e **t=T-1** foram assinaladas como "conversões não confiáveis" **(?)**.

O motivo para esta não confiabilidade prende-se ao fato de que nos intervalos (0, 1) e (T-2, T-1) a CPU tem que executar outras tarefas além da troca de canal e por isso não consegue trocar o canal em tempo hábil, fazendo com que as conversões em t=1 e t=T-1 sejam efetuadas antes que a troca de canal se processe.

Como consequência, a leitura em t=1 ainda contém o sinal da rampa e a leitura em t=T-1 ainda contém o sinal de ressonância. Embora esta dificuldade somente apareça quando o conversor trabalha em taxas de 100 e 200 aquisições por segundo, preferimos ignorar essas leituras em todas as taxas de aquisição, para evitar a criação de uma condição de exceção no programa. Deste modo, os pontos do espectro serão representados então pelas conversões efetuadas de t=2 até t=T-2.

Chamando de Nta o número total de conversões efetuadas, teremos que (Fig. 2-10)

$$Npe = Nta - 4 \qquad (2.11)$$

$$Npc = Nta - 3 \qquad (2.12)$$

onde

Npe = número de pontos do espectro

Npc = número de pontos lidos em "class read"

Nta = número total de aquisições

Considerando as eq. (2.8) e (2.9) e chamando de B_inc a variação do campo entre duas conversões consecutivas, teremos que:

$$B\_inc = (BF - BI)/(Nta - 1) \qquad (2.13)$$

e portanto, o campos correspondentes ao primeiro e ao último pontos do espectro, serão dados por:

$$hi = BI + 2 * B\_inc \qquad (2.14)$$

$$hf = BF - 2 * B\_inc \qquad (2.15)$$

Com base nesses resultados, o campo para qualquer ponto do espectro poderá então ser obtido de:

$$h(n) = hi + (n-1) * B\_inc \qquad (2.16)$$

onde n = 1,2,3,...,Npe.

O cálculo de h(n) para todos os pontos do espectro encerra o comando AQ. A partir deste ponto, os dados ficam na memória e podem ser exibidos no vídeo ou no plotter. A Fig. 2-11 mostra um resultado tipico do programa.

Após a exibição, os dados considerados de interesse podem ser gravados em disco ou fita magnética para processamento posterior.

A gravação, tanto na fita como no disco, é feita em arquivos de acesso direto, no formato binário, para poupar tempo e espaço. Estes arquivos podem entretanto ser convertidos em arquivos sequenciais com formato de 80 colunas ("imagem de cartão") no caso de se fazer necessária a transmissão para outros computadores através de fitas magnéticas ou linha serial (RS 232C). O programa MASPE (Ap. 1) dispõe de recursos para esta conversão nos comandos LI (listar) e GR (gravar)

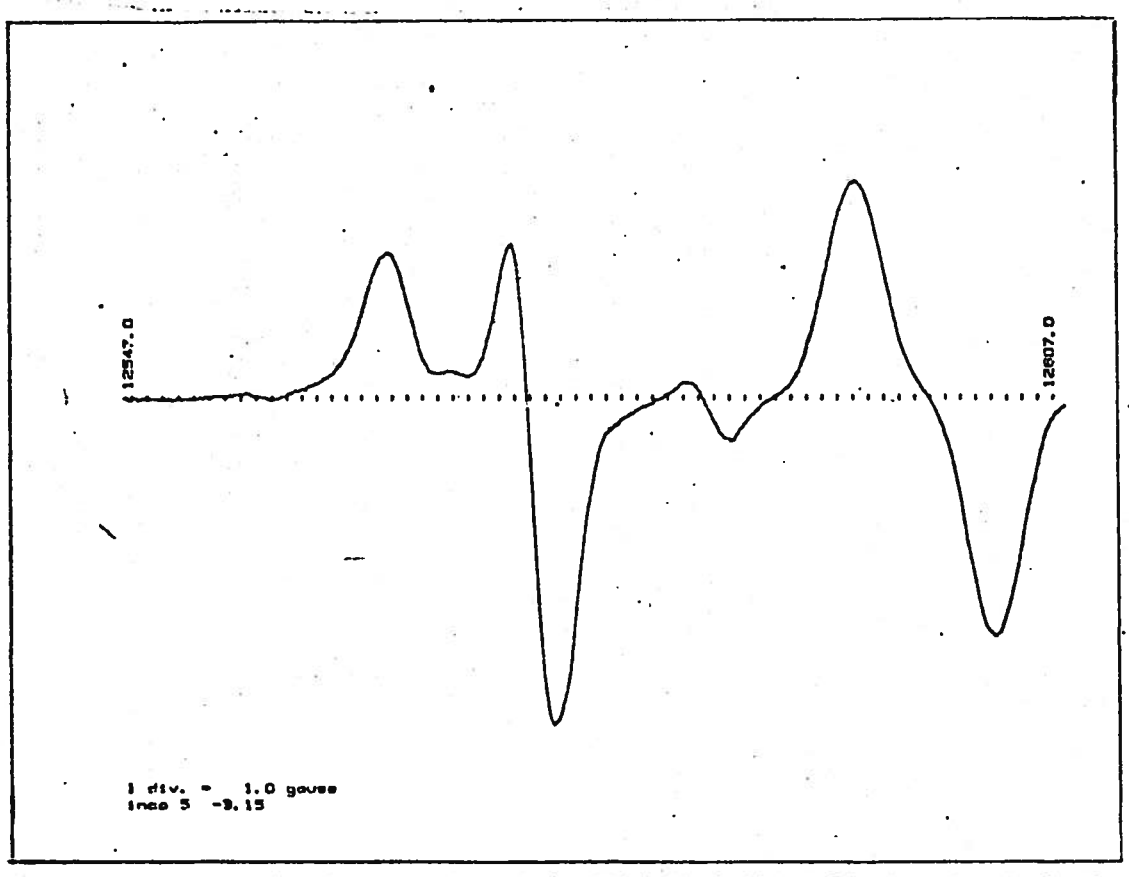


Fig. 2-11: Registro Típico Produzido Pelo SMAP

2.3 Recursos Adicionais do SMAP

Além do registro dos espectros, o programa SMAP pode realizar uma série de outras operações, relacionadas com o desenvolvimento da experiência de EPR, que foram nele colocadas com o objetivo de facilitar o trabalho do pesquisador. Essas operações estão descritas detalhadamente no Apêndice 2 e algumas delas serão apresentadas a seguir, a título de ilustração.

2.3.1 A escolha da taxa de aquisição dos dados.

A taxa de aquisição dos dados, ou seja, a frequência com que são feitas as conversões pelo ADC, é passada ao programa com o comando **FC** e pode ser escolhida entre os valores 5, 10, 20, 50, 100 e 200 cps (conversões por segundo).

Na prática, quando se usa o programa, uma das primeiras indagações que surgem é justamente a respeito de como se escolher essa frequência de aquisição. Para responder a esta questão e estabelecer um critério que facilite a escolha, vamos relembrar o mecanismo da aquisição de dados, discutido na sec. 1.1.

O teorema da amostragem uniforme no dominio do tempo afirma que se uma função f(t) não contém nenhuma componente de frequência mais alta que fm, então esta função pode ser completamente determinada por seus valores situados em intervalos uniformes cuja separação seja menor que 1/2fm.

No aspecto prático, este teorema nos permite criar a seguinte regra:

> Para se amostrar um sinal s(t), cujo conteúdo de frequência se estenda até f_M, devemos ter o ADC operando, a uma taxa maior que $2f_M$ conversões por segundo.

A observância desta regra é fundamental para o sucesso do processo de aquisição de dados porque, como vimos no Cap. 1, se amostrarmos um sinal que contenha, digamos, uma componente de frequência $f_R > f_M$ usando uma taxa de conversão $2f_M$, esta componente aparecerá no espectro como se tivesse uma frequência $f'_R < f_M$ devido ao fenômeno de falseamento da amostragem ("aliasing") e poderá comprometer seriamente a fidelidade dos dados.

Na Fig. 2-12 mostramos um exemplo típico de "aliasing", ocorrido na obtenção de um espectro de EPR de $KCaF_3Mn^{2+}$. Na parte superior da figura, temos um espectro corretamente amostrado, revelando seus vários detalhes. Na parte inferior, temos o mesmo espectro amostrado com uma taxa de conversão 4 vezes menor.

A simples comparação dos dois espectros mostra, claramente, os danos que o fenômeno de "aliasing" pode causar em um processo de aquisição, fazendo com que detalhes importantes, literalmente desapareçam da figura o que pode comprometer seriamente qualquer análise posterior que venha a ser feita a partir desses dados.

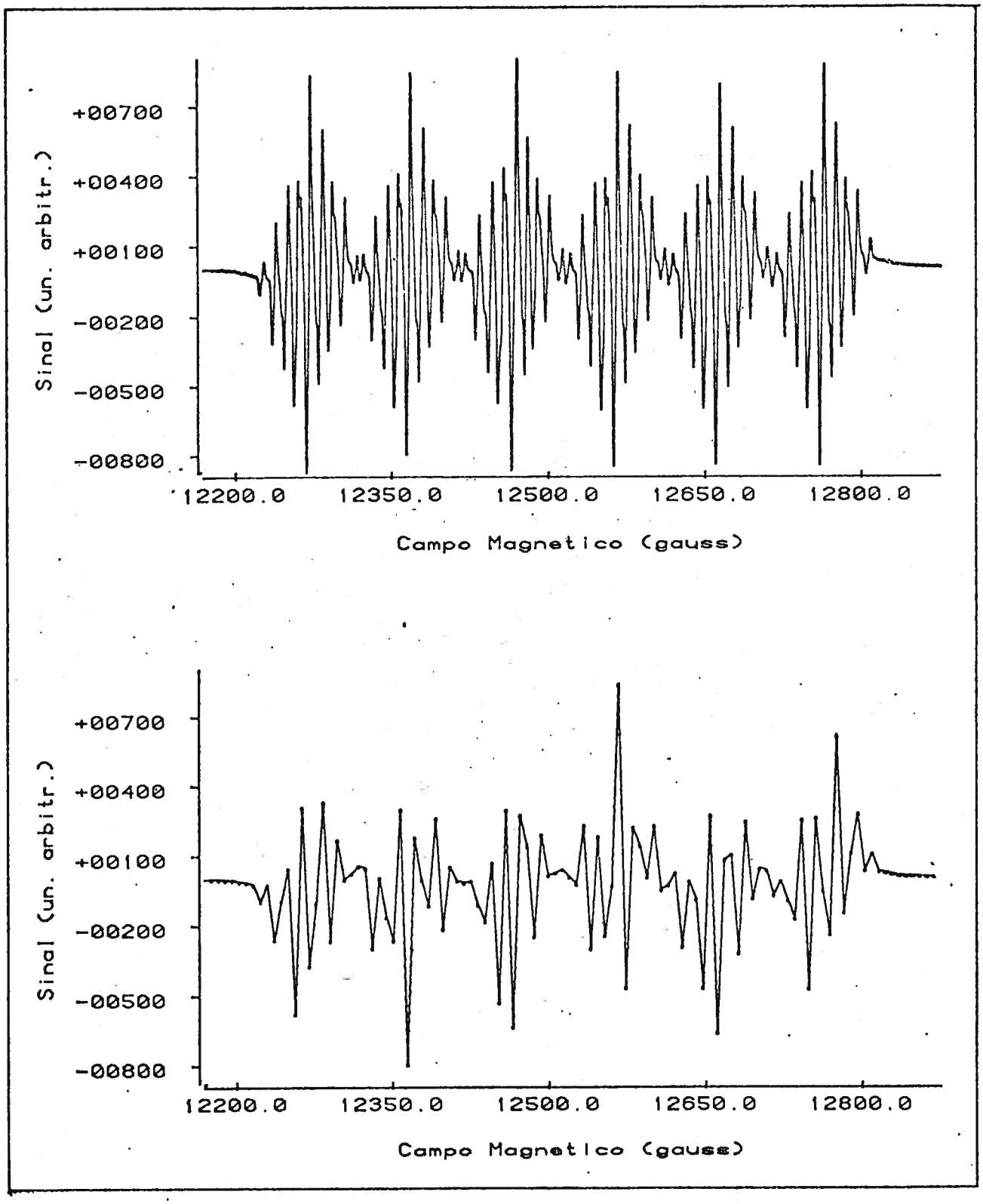


Fig. 2-12: Fenômeno de "aliasing" em um espectro de $KCaF_3Mn^{2+}$
Parte superior: espectro corretamente amostrado
Parte inferior: espectro falsamente amostrado

No caso especifico das medidas de EPR, a aplicação do Teorema da Amostragem nos indica que, para termos uma representação correta de um espectro, devemos amostrar no mínimo dois pontos na linha mais estreita do mesmo.

Assim, se **l** é a largura da linha mais estreita, a densidade de amostragem, (número de pontos/largura da linha mais estreita) d_a, será

$$d_a = 2/l \qquad (2.17)$$

e o número total de pontos em uma varredura de comprimento S será dado por

$$N = (2/l)*S. \qquad (2.18)$$

Chamando de Tv o tempo de varredura ("sweep time") a frequência de aquisição poderá então ser calculada a partir de:

$$f_a = 2S/lT_v \qquad (2.19)$$

No programa SMAP, o comando QF informa o valor da frequência de aquisição para um número **n** de pontos na linha mais estreita usando a relação $f_a = nS/lT_v$ mas, como o ADC só dispõe de algumas frequências de trabalho, o comando informa também a frequência disponível imediatamente acima de f_a.

Deve-se notar que o valor da frequência de aquisição, f_a, dado por (2.19) representa um mínimo para uma amostragem correta.

Para os casos em que os dados vão ser submetidos a processamentos posteriores, certamente o valor de n terá de ser substancialmente maior que 2 não só para atender às exigências dos algoritmos usados em tais processamentos, como também para garantir que alguma linha mais estreita, não conhecida a priori, não venha a passar desapercebida devido ao "aliasing".

2.3.2 O ajuste do "offset"

Normalmente os amplificadores síncronos ("lock-in") dispõem de um ajuste de offset que permite adicionar uma tensão contínua ao sinal de saída, para compatibilização com os ajustes de fundo de escala do registrador X-Y e evitar que o gráfico, ao ser traçado, ultrapasse os limites eletromecânicos do mesmo.

Para espectros que são registrados apenas no papel, o valor dessa tensão de "offset" é irrelevante mas, para os espectros que são digitados para processamento posterior, a presença dessa componente contínua causa problemas nos programas de ajuste e em cálculos de integrais e de Transformadas de Fourier.

No caso das integrais, a presença da tensão de "offset" pode criar um retângulo tão grande abaixo da linha de base que a variação de "área" devida à linha de ressonância praticamente desaparece, conforme se vê nas Figs. 2-13 e 2-14.

Por outro lado, no caso das Transformadas de Fourier, o degrau criado pela tensão de "offset" superpõe-se na forma de um pulso quadrado à forma de onda original e dá origem a frequências espúrias no espectro, conforme se vê nas Figs. 2-15 e 2-16.

O sistema SMAP, por uma questão de filosofia, não permite que os dados de um espectro, uma vez registrados, sejam modificados pelo usuário mas, através do comando **OS**, o valor do "offset" na linha de base pode ser avaliado e registrado juntamente com os dados para uso posterior.

Para se avaliar o valor do "offset", o valor do campo magnético é ajustado manualmente em um ponto qualquer onde o sinal de ressonância seja nulo (linha de base). Neste ponto, são feitas 100 leituras da tensão e a média dessas leituras é tomada como valor do "offset".

De um modo geral, é aconselhável que os espectros sejam registrados com a tensão de "offset" o mais próximo possivel de zero volts. Se isto não for possivel, o programa de manipulação de espectros, MASPE, descrito no apêndice 1, possibilita uma correção precisa do nivel da linha de base, ou seja, do valor de "offset" através de um comando que permite a adição ou subtração de constantes ao espectro. Deste modo, o usuário tem uma certa liberdade para variar o "offset" durante a aquisição o que lhe permite usar melhor os papéis do registrador analógico.

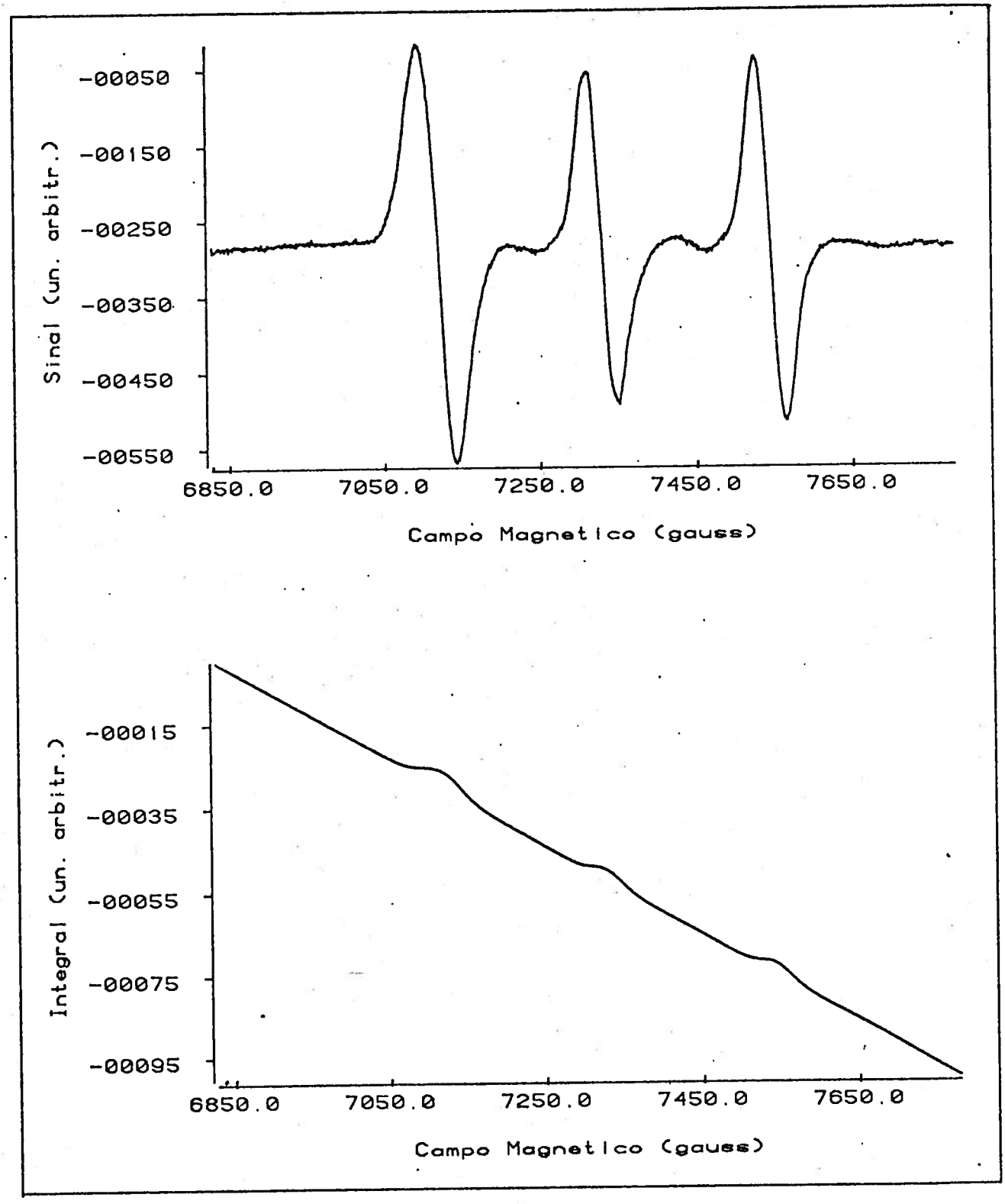


Fig. 2-13: Efeito do "offset" na integração de espectros
Acima: espectro com "offset" diferente de zero
Abaixo: integral do espectro

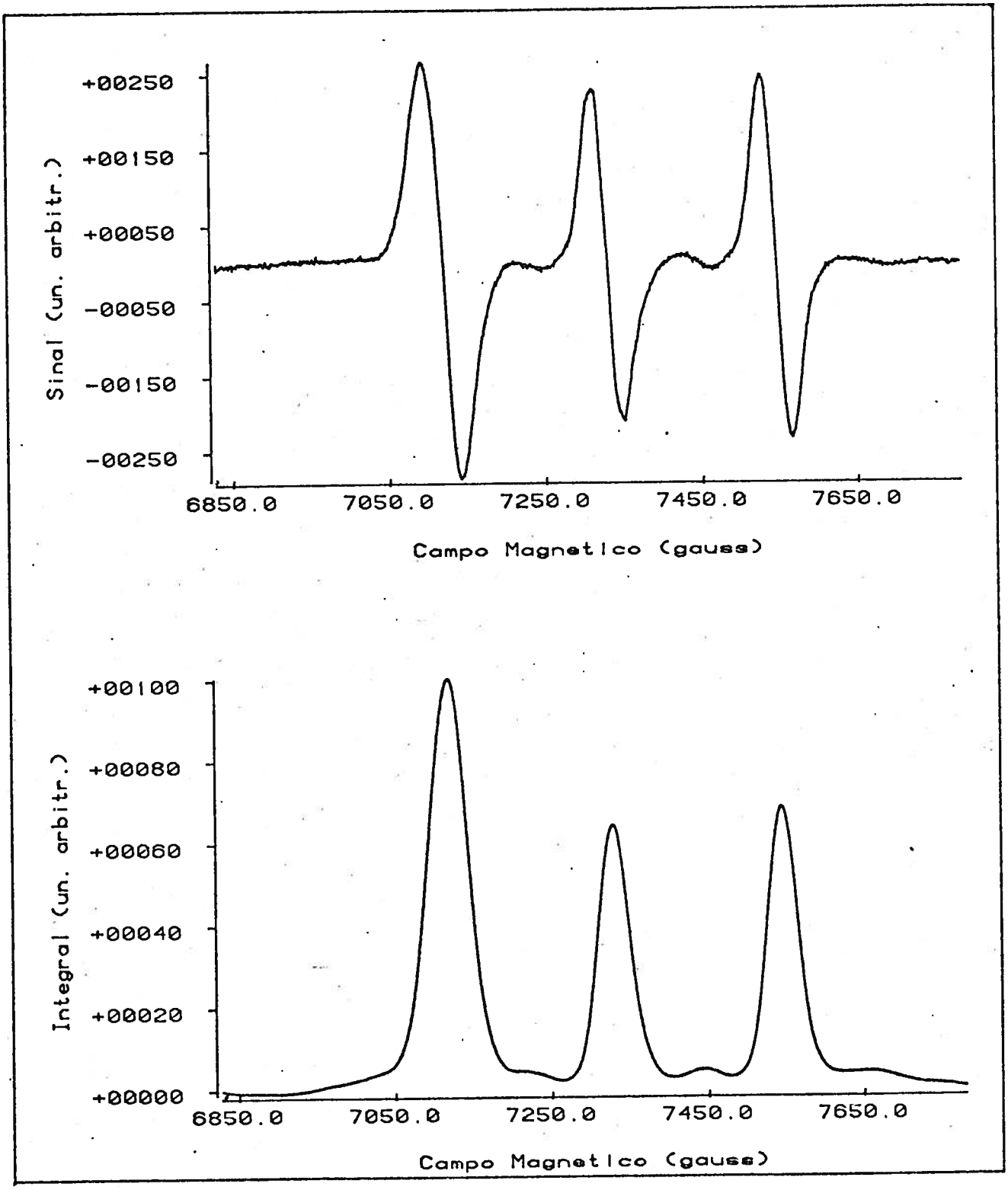


Fig. 2-14: Efeito do "offset" na integração de espectros
Acima: Espectro da fig 2-13, com "offset" removido
Abaixo: Integral do espectro com "offset" removido

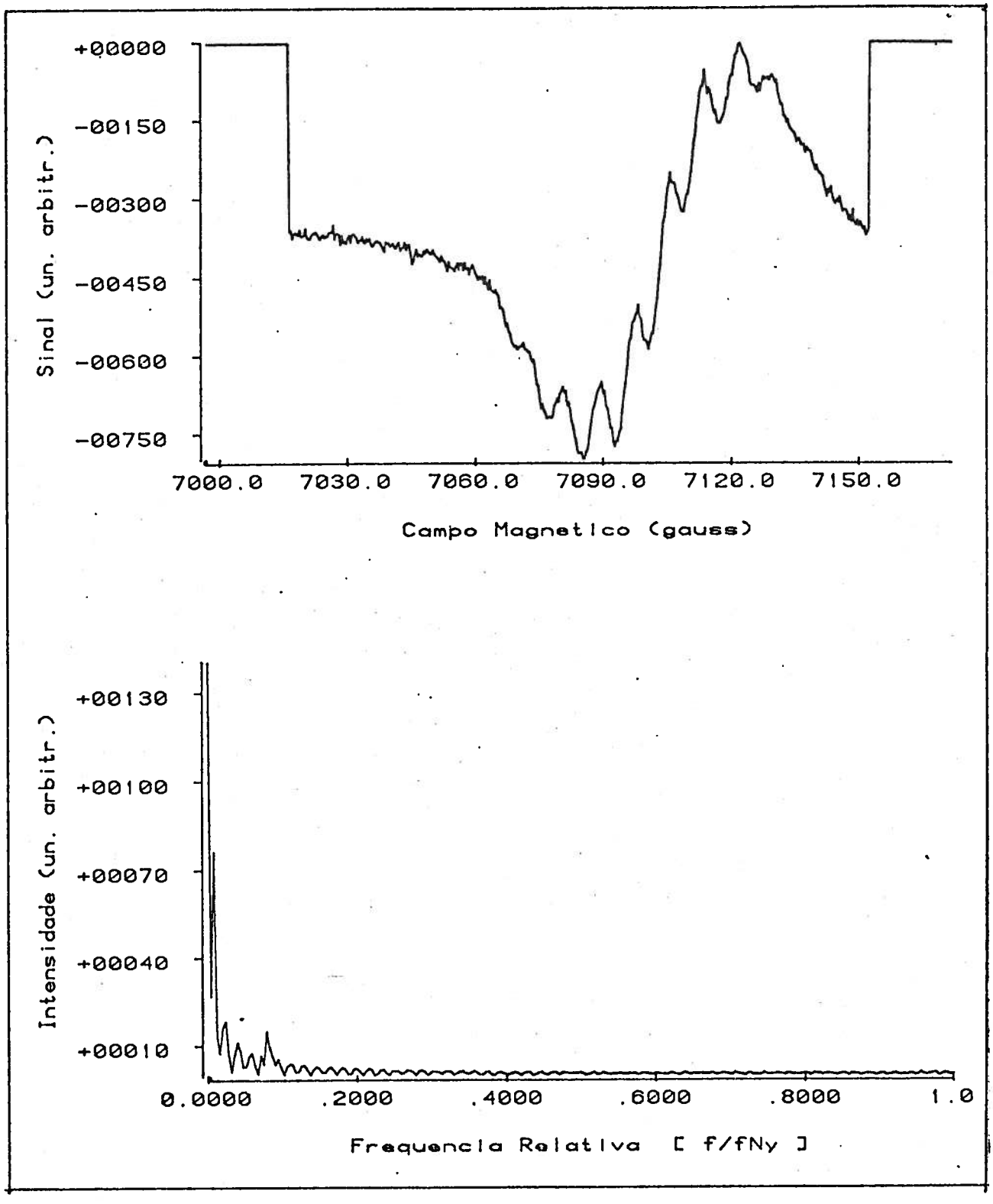


Fig. 2-15: Efeito do "offset" na Transformada de Fourier
Acima: espectro com "offset" diferente de zero
Abaixo: FFT do espectro com "offset"

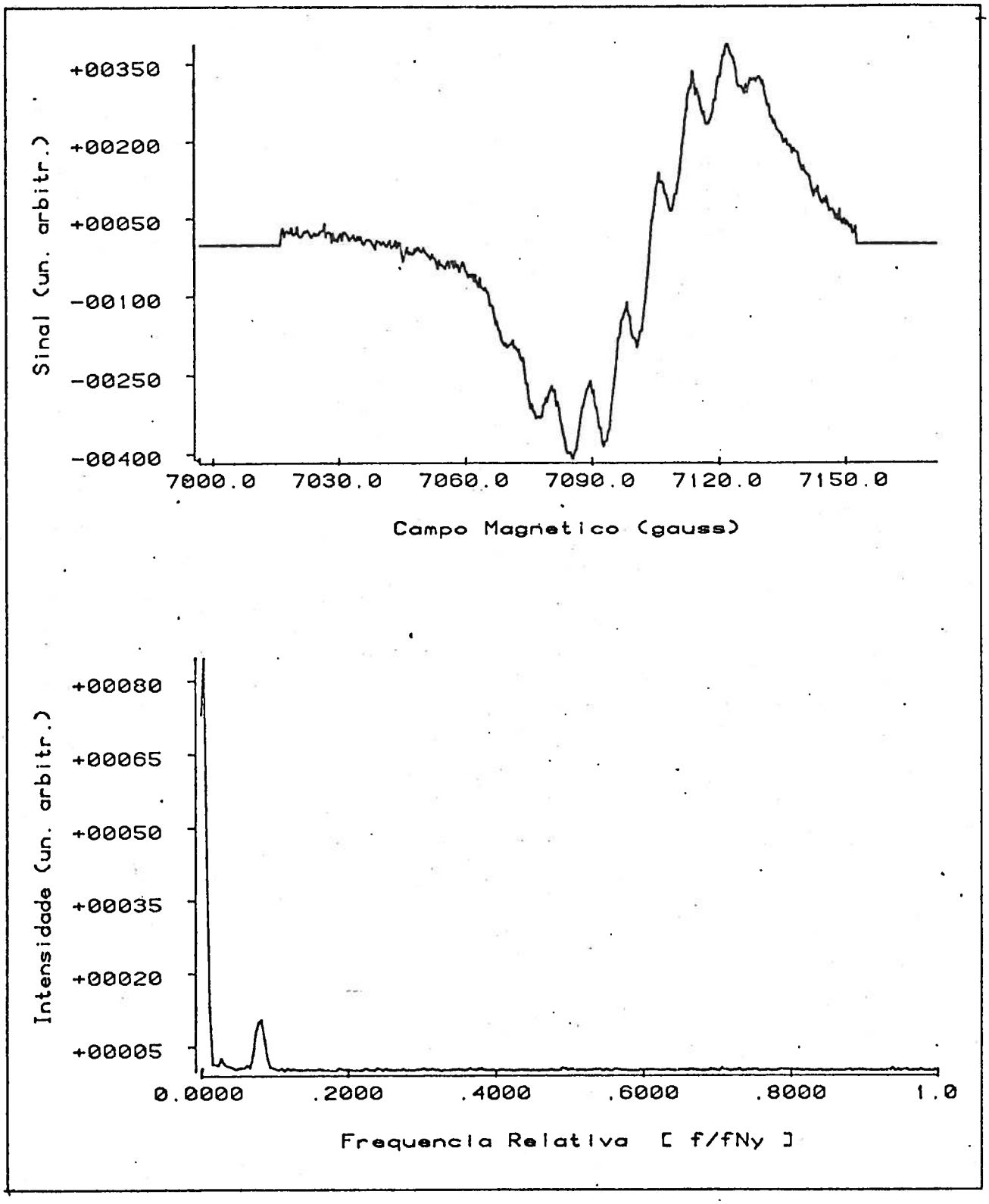


Fig. 2-16: Efeito do "offset" na Transformada de Fourier
Acima: Espectro da Fig. 2-15 com "offset" removido
Abaixo: FFT do espectro sem "offset"

2.3.3 A rotação da amostra

A rotação da amostra em torno do eixo vertical foi automatizada com o acoplamento mecânico entre o porta-amostras e o goniômetro motorizado IT6-DA[9] que se encontra conectado ao sistema através do barramento GP-IB.

O controle do goniômetro foi incorporado ao programa SMAP de modo que, com o uso do comando GN, o operador pode colocar a amostra em qualquer posição accessível ao sistema mecânico com uma resolução de 0,01 graus. A adição do goniômetro motorizado ao porta-amostras simplifica em muito o levantamento das dependências angulares dos espectros pois evita o procedimento, nada prático, de girar o eletroimã.

2.3.4 O uso do barramento IEEE-488 (GP-IB)

A adição de instrumentos GP-IB ao SMAP é uma tarefa que pode ser realizada com relativa facilidade em virtude das caracteristicas universais do barramento IEEE-488 e da estrutura modular adotada nos programas.

Em sua versão atual, o sistema incorpora comandos que permitem o acesso ao frequencímetro de micro ondas HP 5342A e ao multímetro Keythley 195A. Com esse acesso, o usuário pode obter um registro permanente da indicação desses instrumentos, o que é de muita utilidade em situações experimentais que envolvam, por exemplo, o controle da frequência do "klystron" e da temperatura da amostra.

2.4 Aplicações e Resultados

O programa SMAP é um programa de uso geral para as atividades de aquisição de dados em EPR e tem a grande vantagem de poder "coexistir" com a operação tradicional do espectrômetro, coexistência esta que se revelou muito cômoda na fase de ajustes preliminares que sempre precede uma sessão de medidas.

Durante uma sessão, fica a critério do operador fazer ou não a aquisição dos dados para uma dada temperatura, um dado ângulo, etc. Além disso, em virtude da estrutura de comandos utilizada, o programa pode ser usado parcialmente para, por exemplo, controlar a rotação da amostra, enquanto os espectros são registrados da forma analógica habitual.

Para os casos de espectros muito ruidosos, pode-se fazer uma série de varreduras e depois tirar a média dos espectros (signal averaging) desde que seja possivel manter as condições da máquina suficientemente estáveis, durante todo o processo, para garantir a adição coerente dos espectros. Esta estabilidade nem sempre é mantida porque, como a melhoria da relação sinal/ruido no processo de média é proporcional a $\sqrt{N_v}$ (onde N_v é o número de varreduras), o tempo gasto na aquisição pode ser muito grande e acarretar uma desestabilização do espectrômetro. De qualquer forma, o recurso existe e poderá ser usado com proveito sempre que as condições experimentais o permitirem. Além disso, o uso dos filtros digitais pode constituir-se numa opção bem mais rápida para a redução dos ruídos conforme veremos no capítulo 3.

2.5 Resultados do SMAP

O sistema SMAP foi utilizado com pleno sucesso, ao longo de todo o seu desenvolvimento, em vários estudos no Laboratório de Ressonância Magnética[9,11,12,13,14,15].

Nas medidas com ele realizadas concentramos nossa atenção em três pontos fundamentais para uma avaliação de sua confiabilidade.

O primeiro ponto foi uma verificação da capacidade do sistema para registrar espectros ricos em detalhes. Embora pudéssemos avaliar "a priori" as condições ideais para o registro (sec. 2.3.1), havia uma grande expectativa em torno dos resultados, pois estas seriam as primeiras medidas em tempo real a serem feitas em nosso laboratório.

Para essa verificação, fizemos uma série de medidas dos espectros de $KCaF_3Mn^{2+}$ no espectrômetro de banda Q (35 GHz). Nelas, o SMAP foi usado "em paralelo" com o sistema convencional, ou seja, o registrador X-Y (analógico) foi mantido ligado durante a experiência e assim pudemos comparar os resultados obtidos pelos dois métodos e concluir que o sistema de monitoração passiva comportou-se como esperado, registrando corretamente todos os detalhes do espectro, como se pode ver na Fig. 2-17.

Esses mesmos dados, em virtude da riqueza de detalhes, foram também usados no estudo do fenômeno de "aliasing", mostrado na Fig. 2-12.

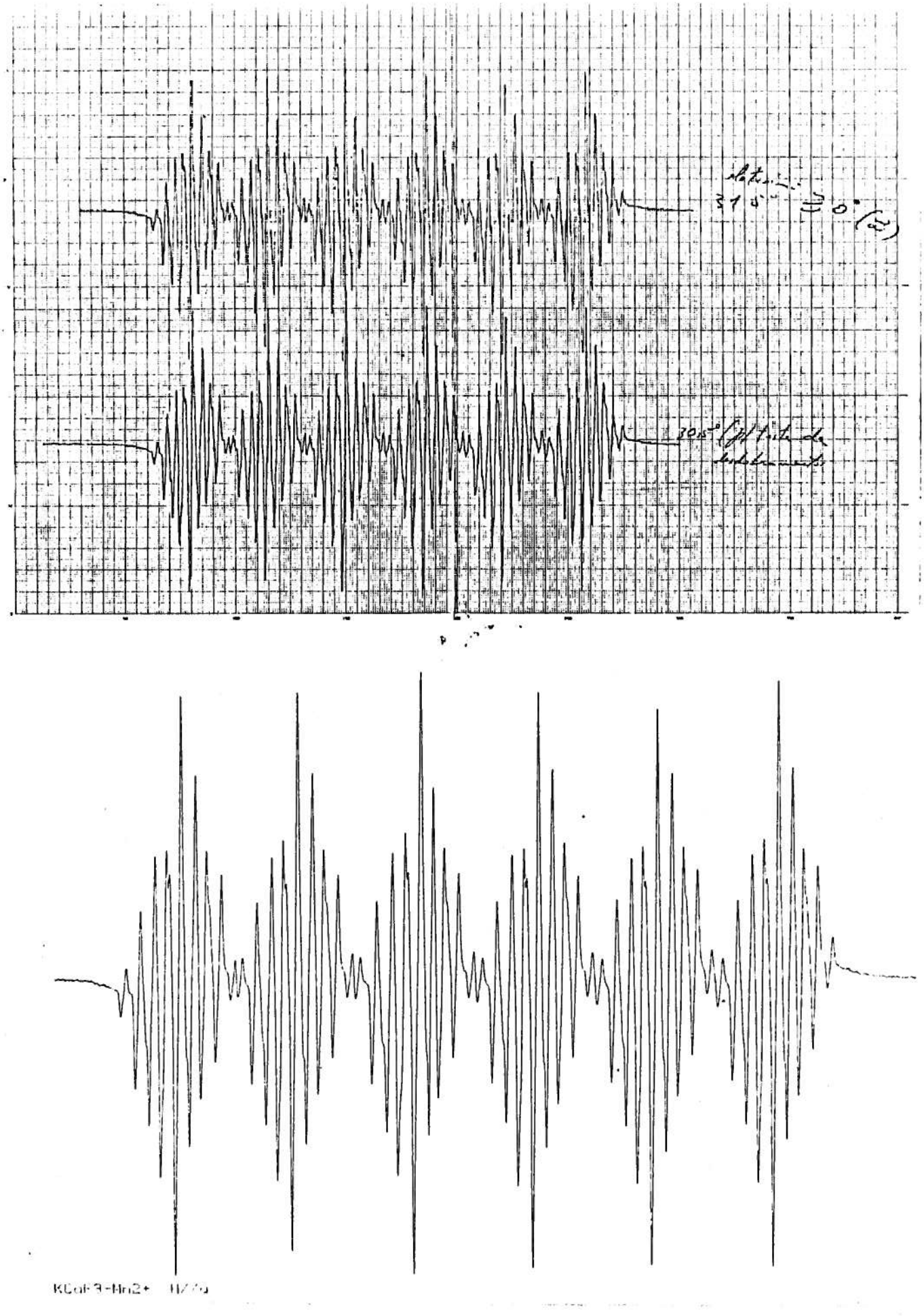


Fig. 2-17: Espectros de $KCaF_3Mn^{2+}$ registrados pelos processos analógico (acima) e digital (abaixo).

O segundo ponto foi uma avaliação da influência de outros programas em execução no HP 1000, no desempenho do SMAP. Para essa avaliação, executamos inicialmente o SMAP com o computador completamente desocupado e anotamos os tempos "previsto" e "realizado" para a aquisição dos dados. Em seguida, colocamos dois outros programas em execução. Um deles, o TTT, calcula indefinidamente o seno(x), onde x é um número qualquer. O outro, o LINK, usa intensamente a unidade de disco. Com esses programas ativados, colocamos o SMAP em execução nas mesmas condições anteriores, e anotamos os tempos "previsto" e "realizado". A comparação dos resultados mostrou-nos que a presença dos outros programas não afetou a execução do SMAP. Os dados obtidos estão apresentados na Fig. 2-18 e os passos seguidos na experiência são os seguintes:

1,2	O programa é colocado no modo de aquisição (CMD = AQ) e solicita providências do operador.
3	O programa inicia a aquisição dos dados e informa no terminal a duração prevista [Tta: 48.3 (segundos)]
4	O programa WHZAT (status) informa que neste instante nenhum programa está sendo executado. (A aquisição dos dados é feita pelo sistema operacional. [sec. 2.2.4])
4+	O programa WHZAT é executado e mostra, neste instante, os programas TTT (terminal 10) e LINK (terminal 17).
5	O fim do processo de aquisição é sinalizado no terminal.
6,6+	O tempo de aquisição é solicitado ao programa (CMD = TI) e exibido juntamente com o tempo previsto.

A comparação de 6 e 6+ mostra que em ambos os casos a aquisição completou-se em 48,31s. A diferença de 0,01s entre os tempos "previsto" e "realizado" aparece porque não se leva em conta o tempo que a CPU gasta na execução das instruções de aquisição.

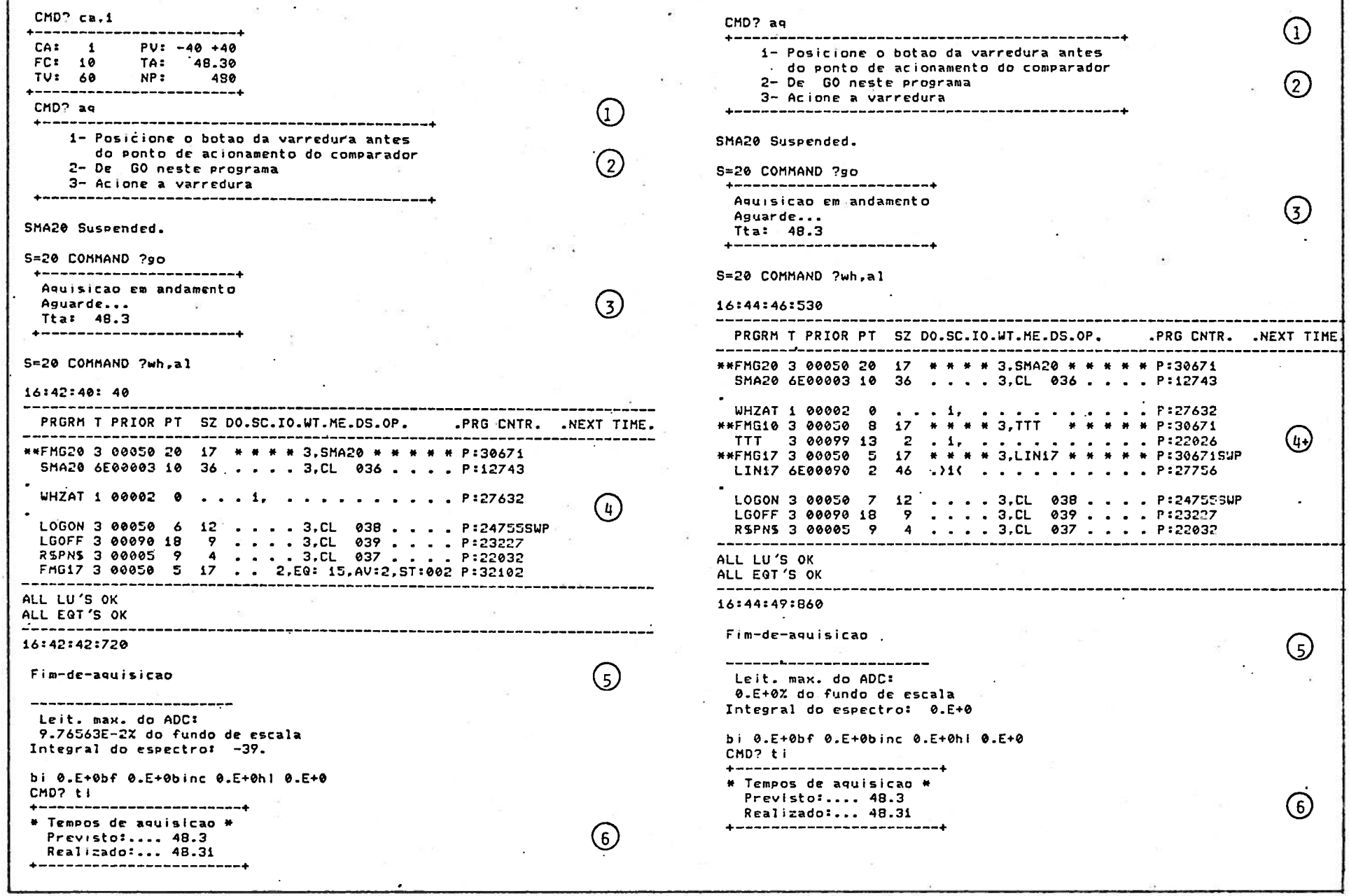


Fig. 2-18: Influência de outros programas no "tempo de aquisicão" do SMAP.

O terceiro ponto foi um teste do "método de calibração da rampa" (sec 2.2.1) para o registro das intensidades do campo magnético. Para realizar este teste, usamos várias séries de registros de espectros de KH_2PO_4 (KDP), KH_2AsO_4 (KDA), KD_2AsO_4 (DKDA), RbH_2AsO_4 (RDA) e $KCaF_3Mn2^+$.

Para cada série de espectros, anotamos os valores inicial e final do campo magnético, calculamos a largura total do espectro, a média e o desvio padrão de cada conjunto de dados e resumimos os resultados nas tabelas 2-5, 2-6, 2-7, 2-8 e 2-9.

Os dados das tabelas 2-5 e 2-6 foram tomados com o ADC "frio", enquanto que os das tabelas 2-7 e 2-8 foram tomados após um período de estabilização de cerca de 2 horas, tal como sugerido pelo fabricante.[10] Comparando as tabelas, podemos verificar que a estabilização do ADC é um fator importante para a redução na dispersão dos valores do campo magnético e por isso, é aconselhável que o tempo de estabilização seja sempre respeitado antes de se iniciar as medições.

A tabela 2-9 mostra os valores do campo no centro da varredura, para uma série de espectros de RDA. Essas medidas foram tomadas após a estabilização térmica de todo o sistema e a rampa foi recalibrada durante a experiência. A dispersão observada neste caso é da ordem de 1 gauss. Este valor é justamente o que podemos esperar como tolerância para a intensidade do campo, em varreduras da ordem de 1000 gauss, com o sistema bem estabilizado.

Programa Data1

KDP - Dep. Angular/Serie #2

Fita Magnetica

No.	Hmin	Hmax	Larg
01	5918.1	7463.2	1545.1
02	5918.1	7457.7	1539.6
03	5920.9	7466.0	1545.1
04	5918.1	7460.5	1542.4
05	5918.1	7463.2	1545.1
06	5915.4	7460.5	1545.1
07	5918.1	7466.0	1547.9
08	5918.1	7466.0	1547.9
09	5923.6	7468.7	1545.1
10	5926.4	7476.9	1550.5
11	5926.4	7468.7	1542.3
12	5923.6	7471.5	1547.9
13	5926.4	7471.5	1545.1
14	5923.6	7474.2	1550.6
15	5923.6	7468.7	1545.1
16	5923.6	7466.0	1542.4
17	5920.9	7466.0	1545.1
18	5918.1	7460.5	1542.4
19	5918.1	7466.0	1547.9
Media	5921.0	7466.4	1545.4
D Padr	3.4024	4.8152	2.8006
DP rel	.0575	.0645	.1812

Tabela 2-5: ADC "frio"

Programa Data1

KDP - Var. da Temperatura #1

Fita Magnetica

No.	Hmin	Hmax	Larg
01	5852.7	7666.4	1813.7
02	5852.7	7658.1	1805.4
03	5852.7	7660.9	1808.2
04	5852.7	7658.1	1805.4
05	5855.4	7660.9	1805.5
06	5855.4	7669.1	1813.7
07	5855.4	7660.9	1805.5
08	5855.4	7666.4	1811.0
09	5855.4	7669.1	1813.7
10	5858.2	7663.6	1805.4
11	5855.4	7666.4	1811.0
12	5858.2	7669.1	1810.9
Media	5855.0	7664.1	1809.1
D Padr	1.8839	4.0190	3.4367
DP rel	.0322	.0524	.1900

Tabela 2-7: ADC estável

Programa Data1

KCaF3Mn2+ H//a+xx 35 GHz

Fita Magnetica

No.	Hmin	Hmax	Larg
01	12153.	12865.	712.0
02	12154.	12870.	716.0
03	12158.	12875.	717.0
04	12158.	12871.	713.0
05	12158.	12871.	713.0
06	12159.	12869.	710.0
07	12158.	12871.	713.0
08	12163.	12877.	714.0
Media	12158.	12871.	713.5
D Padr	2.8696	3.4073	2.0616
DP rel	.0236	.0265	.2889

Tabela 2-6: ADC "frio"

Programa Data1

KDP - avg 16 - Temp. Cte. #1

Fita Magnetica

No.	Hmin	Hmax	Larg
01	5858.2	7666.4	1808.2
02	5858.2	7671.9	1813.7
03	5858.2	7663.6	1805.4
04	5858.2	7663.6	1805.4
05	5858.2	7669.1	1810.9
06	5858.2	7671.9	1813.7
07	5858.2	7663.6	1805.4
08	5858.2	7666.4	1808.2
09	5858.2	7671.9	1813.7
10	5858.2	7666.4	1808.2
11	5860.9	7666.4	1805.5
12	5860.9	7671.9	1811.0
13	5858.2	7666.4	1808.2
14	5858.2	7663.6	1805.4
15	5858.2	7666.4	1808.2
16	5858.2	7669.1	1810.9
Media	5858.5	7667.4	1808.9
D Padr	.8930	3.0675	3.0057
DP rel	.0152	.0400	.1662

Tabela 2-8: ADC estável

Tabelas do método de calibração da rampa

Programa Datai

RDA:T12+ Hi

/ressoi/data/ALVxx.dat

No.	Hmin	Hmax	Larg
01	7052.7	0.0	-7053.
02	7052.2	0.0	-7052.
03	7053.0	0.0	-7053.
04	7050.5	0.0	-7051.
05	7050.6	0.0	-7051.
06	7051.6	0.0	-7052.
07	7051.1	0.0	-7051.
08	7051.0	0.0	-7051.
09	7051.6	0.0	-7052.
10	7051.0	0.0	-7051.
11	7051.0	0.0	-7051.
12	7052.0	0.0	-7052.
13	7050.0	0.0	-7050.
14	7050.0	0.0	-7050.
15	7051.6	0.0	-7052.
16	7051.0	0.0	-7051.
Media	7051.3	0.0	-7051.
D Padr	.8415	0.0000	.8415
DP rel	.0119	******	-.0119

Tabela 2-9: ADC estável, campo no centro da varredura

Nas tabelas que acabamos de apresentar, podemos ainda observar que os valores de campo no fim dos espectros (H_{max}) apresentam, sistematicamente, uma maior dispersão que no início (H_{min}). Este fato está relacionado com duas características intrínsecas ao ADC, que são a incerteza na base de tempo (0,05%) e o atraso no início da conversão (Fig. 2-19) pois, como vimos, a temporização de todo o processo de aquisição é controlada pelo relógio do ADC e não pelo relógio do computador.

Em uma varredura de 1kG, realizada em 60s, a incerteza na base de tempo do ADC pode introduzir erros de até ±0,5 gauss, enquanto que o atraso no início da conversão tem seu valor máximo justa-

mente no final da varredura (tensão mais negativa) e pode introduzir erros de até 0,8 gauss.

Apesar destes erros, podemos concluir do que foi mostrado que o SMAP é capaz de registrar os valores do campo com uma precisão da ordem de 1 parte em 1000, mesmo quando o sistema ainda não se encontra em total estabilidade térmica. Esta capacidade de registrar campo corretamente é uma característica muito importante para aplicações de "signal averaging", pois nelas os erros nos valores do campo podem acarretar sérias distorções na forma de onda resultante.

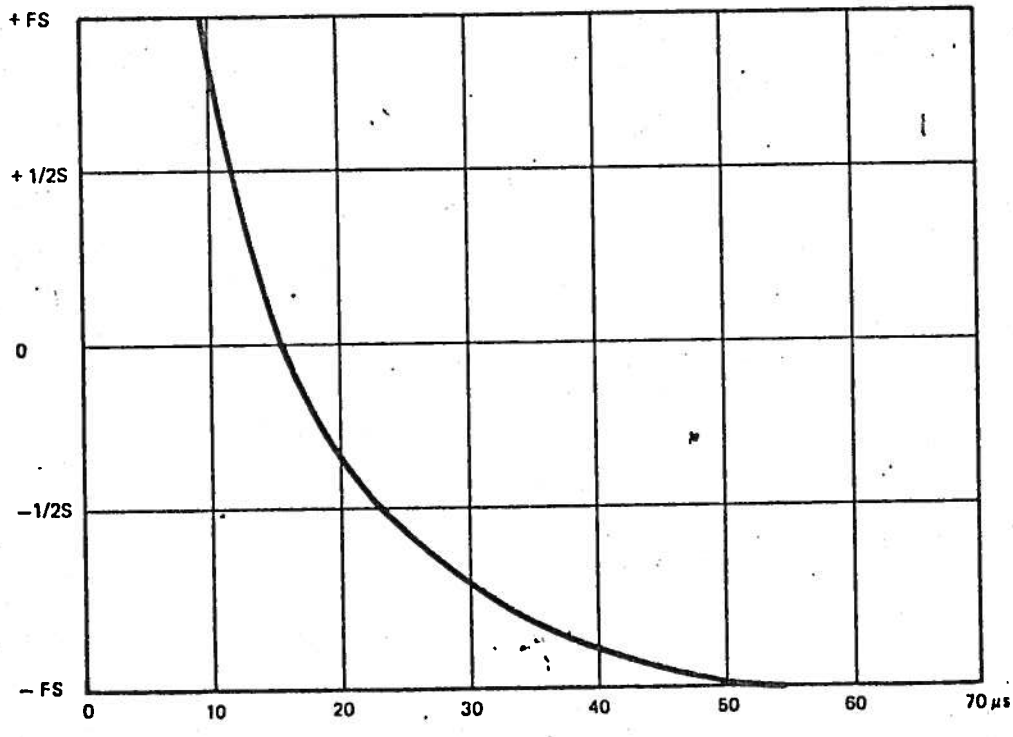


Fig. 2-19: Atraso no início da conversão para o ADC HP 59313A

Além dos erros de base de tempo e início de conversão, o ADC contribui para a incerteza das medidas, tanto da rampa como da intensidade do campo, com outros erros que, de acordo com o fabricante[10], se manifestam na forma de resolução (0,05%), modulação cruzada (<1/2 LSB) e instabilidade no controle de ganho (±0,6%)

Os efeitos de "discretizaçao" e "saturação" do ADC podem também causar sérios transtornos, se não forem levados em conta durante as medições. Esses efeitos são inerentes ao ADC e, embora não possam ser eliminados, podem ser evitados com facilidade, como vimos no cap 1.

O problema da discretizaçao ou "da faixa dinâmica do ADC" aparece quando a faixa de variação do sinal amostrado pelo ADC fica muito próxima de sua resolução. Neste caso, tem-se um espectro com a aparência de uma escada, como se vê na Fig. 2-20.

A saturação acontece quando o sinal amostrado ultrapassa o fundo de escala do ADC e neste caso, perde-se uma região do espectro, como se vê na Fig. 2-21.

Para evitar o aparecimento desses efeitos, basta que o usuário faça uma breve inspeção do espectro, até mesmo com uma varredura "manual", agindo diretamente nos controles da fonte do eletroima, e escolha um canal com a sensibilidade adequada para acomodar o sinal.

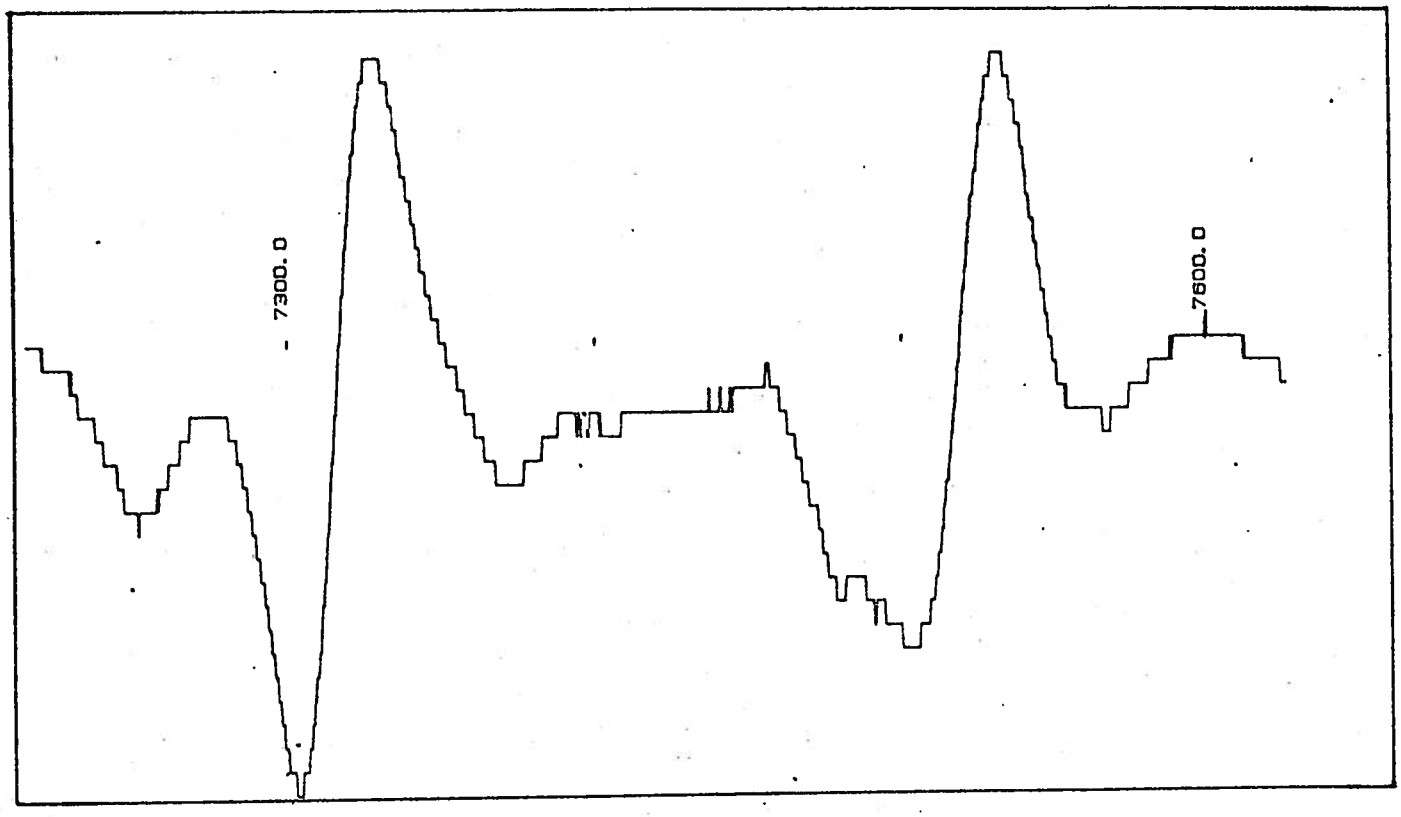


Fig. 2-20: Efeito de discretização

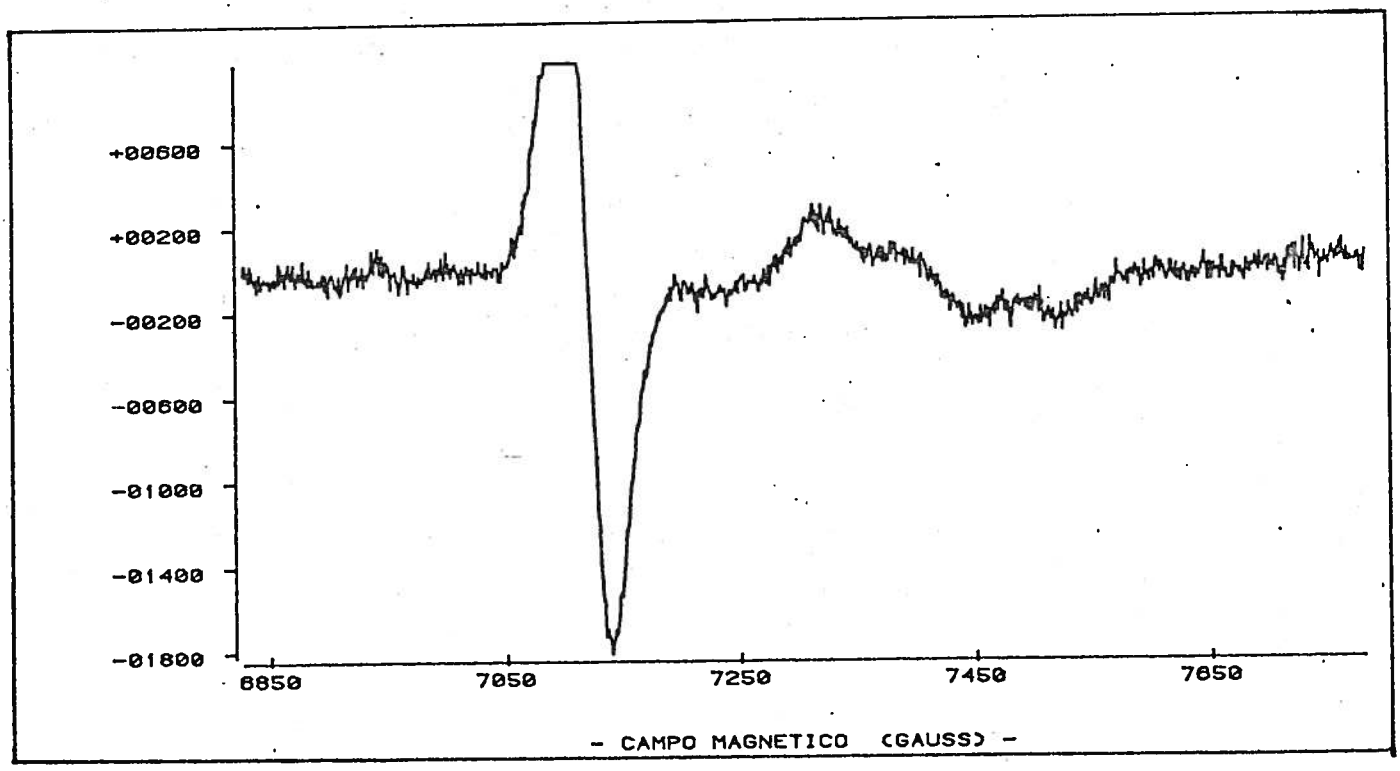


Fig. 2-21: Efeito de saturação

Nas medidas com o KDA, tivemos a oportunidade de testar o método de "signal averaging", fazendo 16 varreduras de uma região do espectro (Fig. 2-22), da qual se desejava um registro mais detalhado.

Essas varreduras foram feitas após um tempo adequado de estabilização térmica do sistema e além disso, outros parâmetros da experiência tais como temperatura da amostra, sintonia do espectrômetro, etc., foram mantidos estáveis, com o objetivo de se atingir, da melhor maneira possível, as condições de aplicabilidade da técnica de "averaging", discutidas no capitulo 1, sec. 1.5.

O cálculo da média dos espectros foi realizado pelo programa MASPE, logo após o término das aquisições. O resultado obtido está na Fig. 2-23, onde se pode ver, claramente, a melhoria na relação sinal-ruido que, de acordo com a teoria ficou multiplicada por um fator de 4 (16 varreduras).

O uso de um outro programa - no caso o MASPE - para o cálculo da média pode parecer, a princípio, um contrasenso mas devemos lembrar que, em virtude das caracteristicas do sistema operacional RTE-6, esta operação não traz o menor problema para o usuário podendo ser inclusive muito oportuna quando duas pessoas operam o sistema pois, neste caso, enquanto uma opera o espectrômetro e cuida da aquisição dos dados, a outra pode dedicar-se ao processamento dos espectros já registrados.

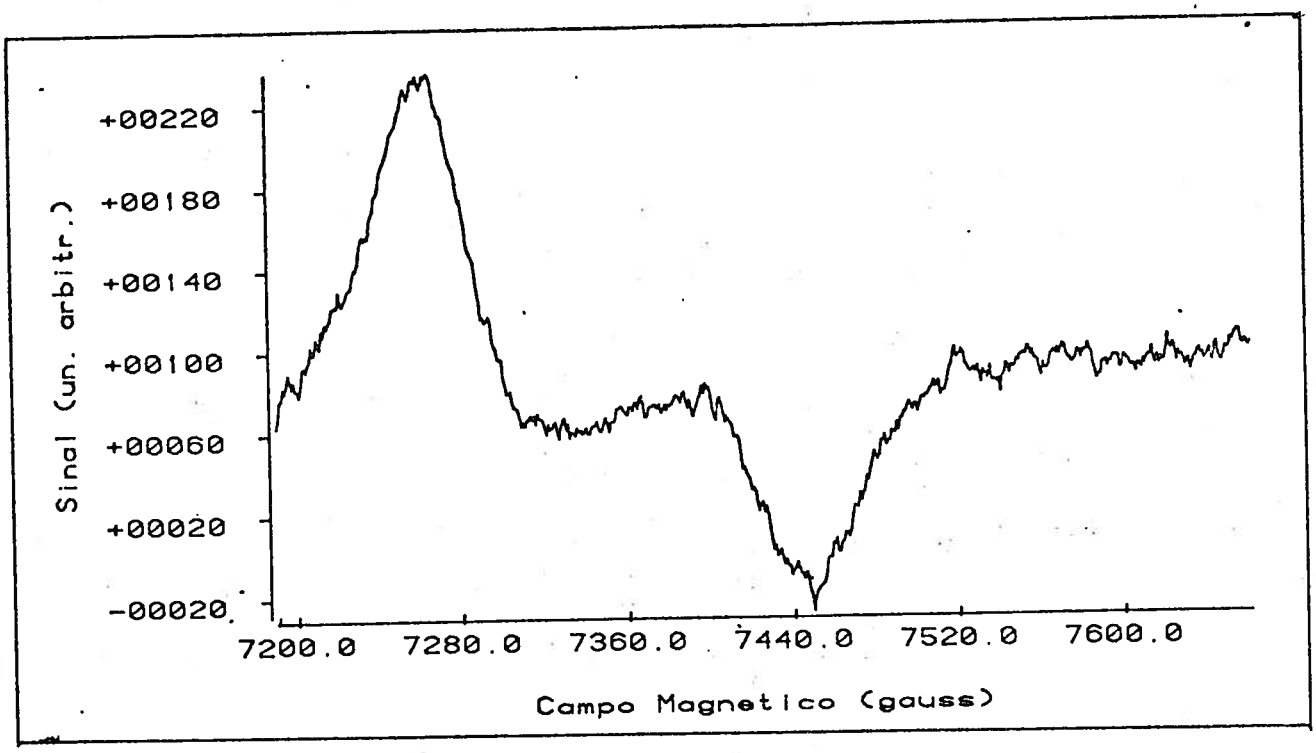


Fig. 2-22: Teste de "Signal Averaging" - Resultado de 1 varredura

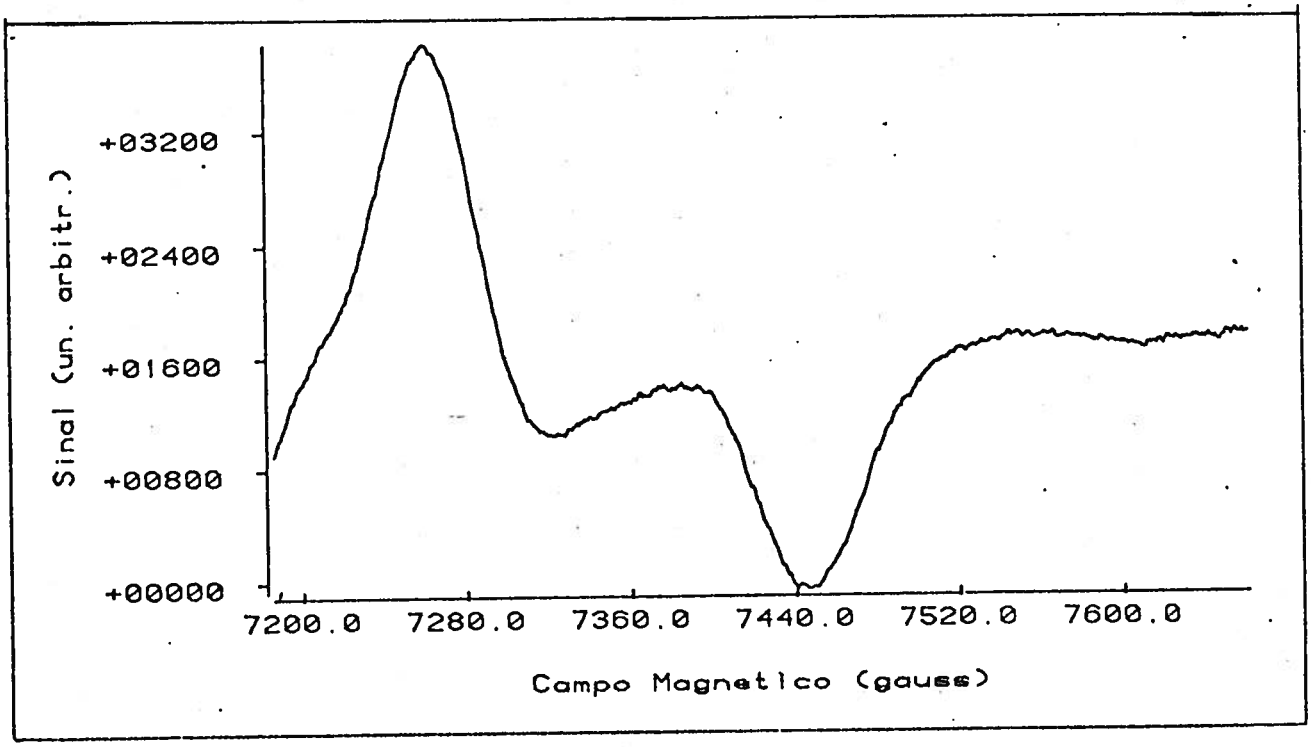


Fig. 2-23: Teste de "Signal Averaging" - Média de 16 varreduras

Um exemplo dos resultados obtidos a partir dos espectros de KDP, usados nos testes do SMAP, está na Fig. 2-24[11]. Com relação a esses espectros, cabe mencionar que eles foram registrados também pelo processo tradicional e que, a comparação sistemática entre os dois processos nos permitiu concluir que o sistema digital apresentou um excelente grau de confiabilidade.

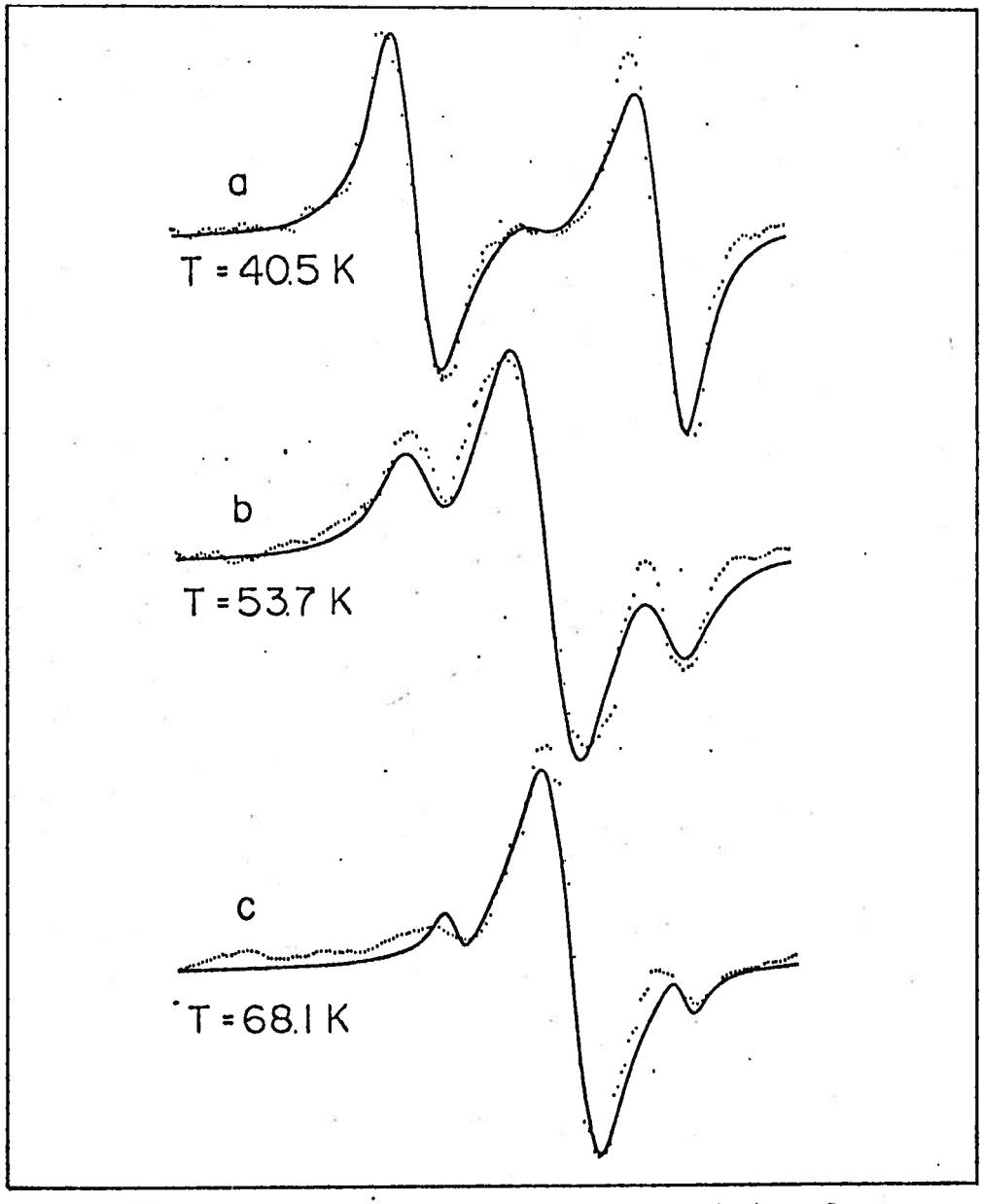


Fig. 2-24: Ajuste de linhas de KDP:TL2+. Os dados foram registrados pelos processos analógico e digital para comparação.

Na Fig. 2-25, apresentamos resultados obtidos a partir de espectros de CsDA (CsH_2AsO_4)[12]. Esses espectros foram digitados a partir de registros analógicos, com auxílio de uma mesa digitadora, e serviram de base para os estudos preliminares do sistema SMAP.

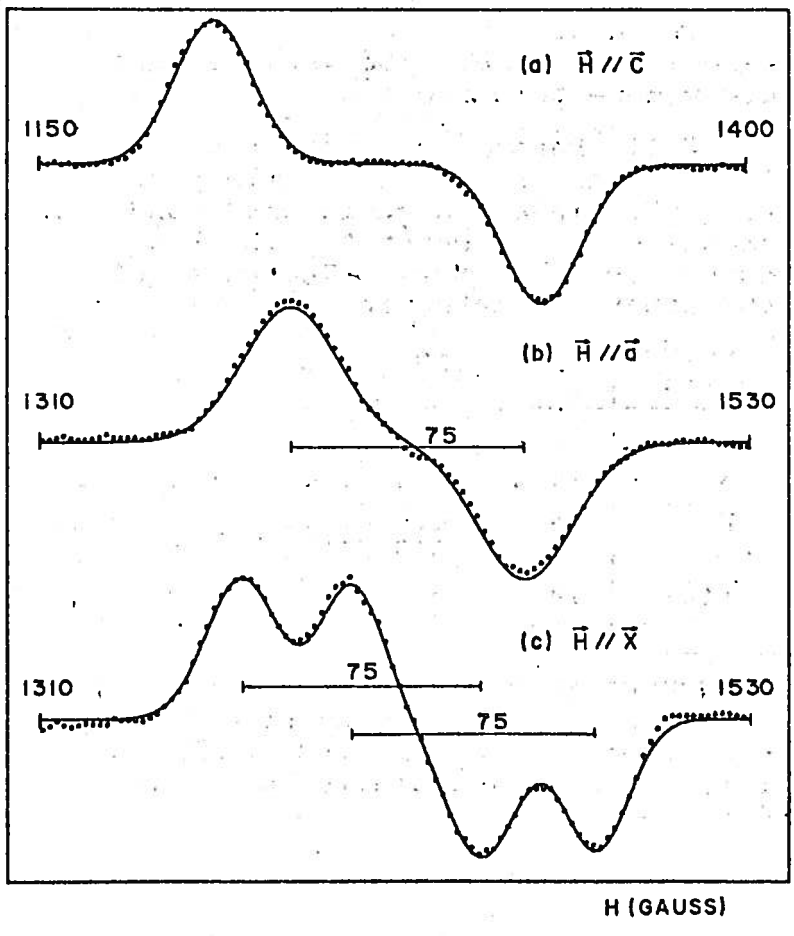


Fig. 2-25: Espectros do centro AsO_4^{4-} em CsDA, registrados através de mesa digitadora, usados nas fases iniciais do desenvolvimento do SMAP e do programa de manipulação de espectros MASPE.

Pelo conjunto desses resultados, pudemos concluir que o SMAP teve um ótimo desempenho, de acordo com a nossa expectativa, e que os dados registrados foram suficientemente confiáveis para levar a resultados físicos significativos.

O valor da dispersão na intensidade do campo, nas extremidades do espectro, não chega a ser um obstáculo para as experiências realizadas atualmente em nosso laboratório. No futuro, se necessário, o desempenho do sistema poderá ser melhorado, neste particular, através do uso de um ADC de maior precisão e melhor base de tempo e da instalação de uma fonte de maior estabilidade para a geração da rampa.

-x-

Capítulo 3

SMOP: Sistema do Motor de Passo

3.1 Introdução

O segundo sistema desenvolvido neste trabalho foi o que chamamos de "Sistema do Motor de Passo" (SMOP), o qual, da mesma forma que o sistema de aquisição passiva, foi implementado a partir do computador HP 1000 seguindo a mesma metodologia da "arquitetura aberta".

Ao contrário do SMAP, o Sistema do Motor de Passo é um sistema ativo de aquisição de dados, ou seja, tem condições de interferir, ainda que parcialmente, no andamento da experiência, através de um mecanismo de controle da evolução da varredura do campo magnético.

O ponto central que procuramos atingir com o SMOP foi a obtenção de um controle mais rigoroso da varredura do campo magnético, com os objetivos de melhorar o desempenho do processo de média do sinal (signal averaging), através do uso da técnica de média ponto a ponto, e preparar o sistema para um futuro redirecionamento que visa a implantação de um programa para aquisição de dados em experiências de ENDOR.

3.2 O Sistema do Motor de Passo

Nos espectrômetros de EPR mais modernos, a automatização do processo de varredura do campo se faz tipicamente através da injeção de uma tensão de controle, gerada por computador, em um terminal apropriado existente na fonte de alimentação do eletroimã.[16,17] Nessas aplicações, esta tensão de controle é obtida a partir de um conversor digital-analógico (DAC), em geral de 12 bits, permitindo assim que a varredura seja dividida em 4096 passos (Fig. 3-1). E comum também a existência de um ADC para leitura da intensidade do campo ou de outros parâmetros do sistema.

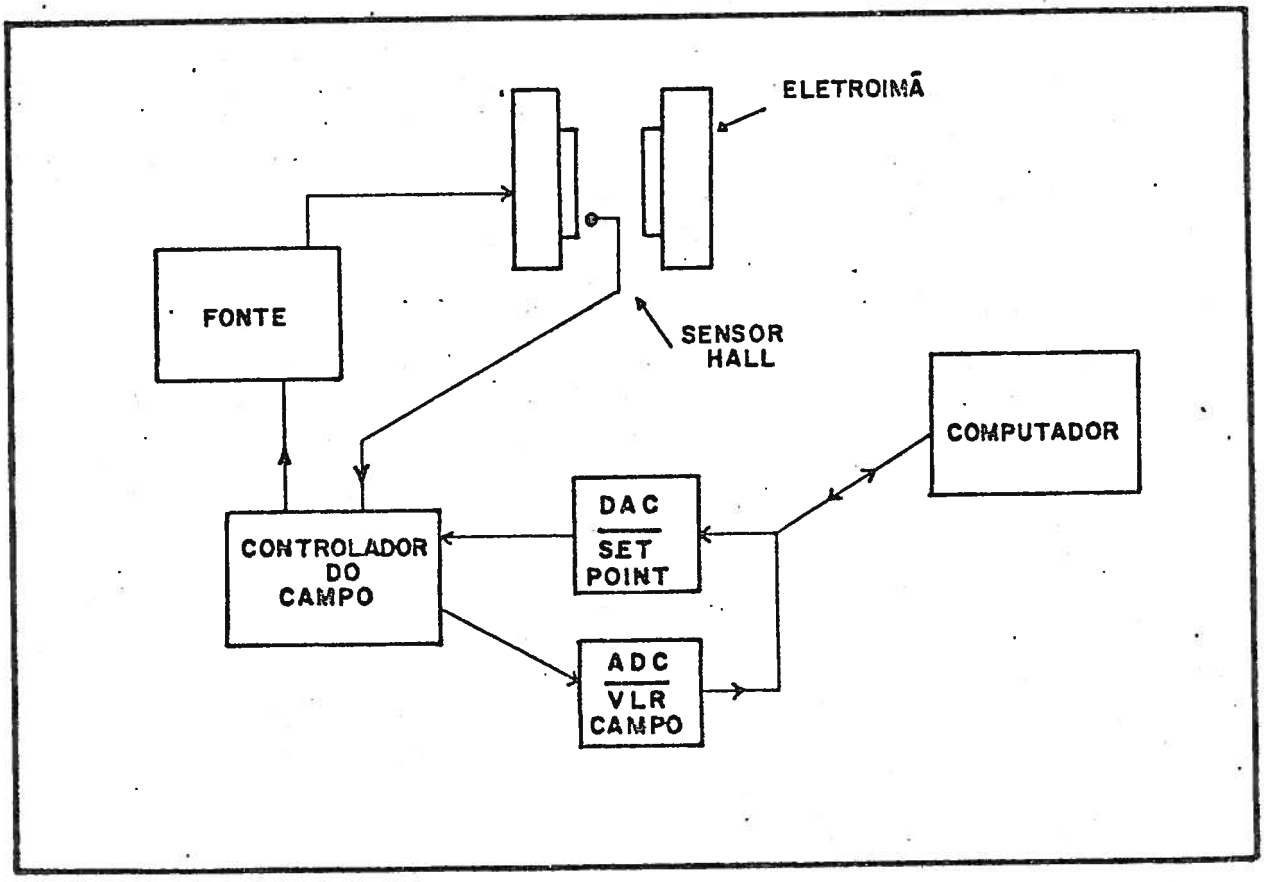


Fig. 3-1: Esquema típico para controle do campo por computador. O valor do campo é ajustado com o DAC e lido com o ADC.

No espectrômetro usado em nosso trabalho, a fonte do eletroimã é uma "Varian Fieldial Mark I" que não dispõe de recursos para controle externo, através de computadores, por pertencer a uma série bem antiga desses instrumentos.

Essa limitação da fonte levou-nos a procurar uma maneira de realizar esse controle externo que não necessitasse de grandes modificações nos circuitos eletrônicos de estabilização do campo, devido às grandes dificuldades técnicas que este procedimento acarretaria. Em vista disso, resolvemos adotar um processo bastante simples, mas que se mostrou adequado ao nosso propósito, e que consiste essencialmente em usar um motor de passo para acionar o eixo do potenciômetro de controle da varredura.

Conforme pode ser visto no diagrama da Fig. 3-2, o controle da varredura é feito por um potenciômetro acionado por um motor síncrono que, por sua vez, tem sua movimentação comandada pela chave "Increase-Decrease". O que fizemos foi substituir o bloco "motor síncrono/potenciômetro" por um bloco "motor de passo/potenciômetro", onde usamos um motor de passo de 200 passos por revolução e um potenciômetro "helipot" de 10 voltas possibilitando, portanto, uma divisão da varredura em 2000 passos.

Para manter a integridade da fonte, esse segundo bloco foi montado externamente e a comutação entre os dois sistemas se faz por um relé, localizado próximo ao potenciômetro original. (Fig. 3-3)

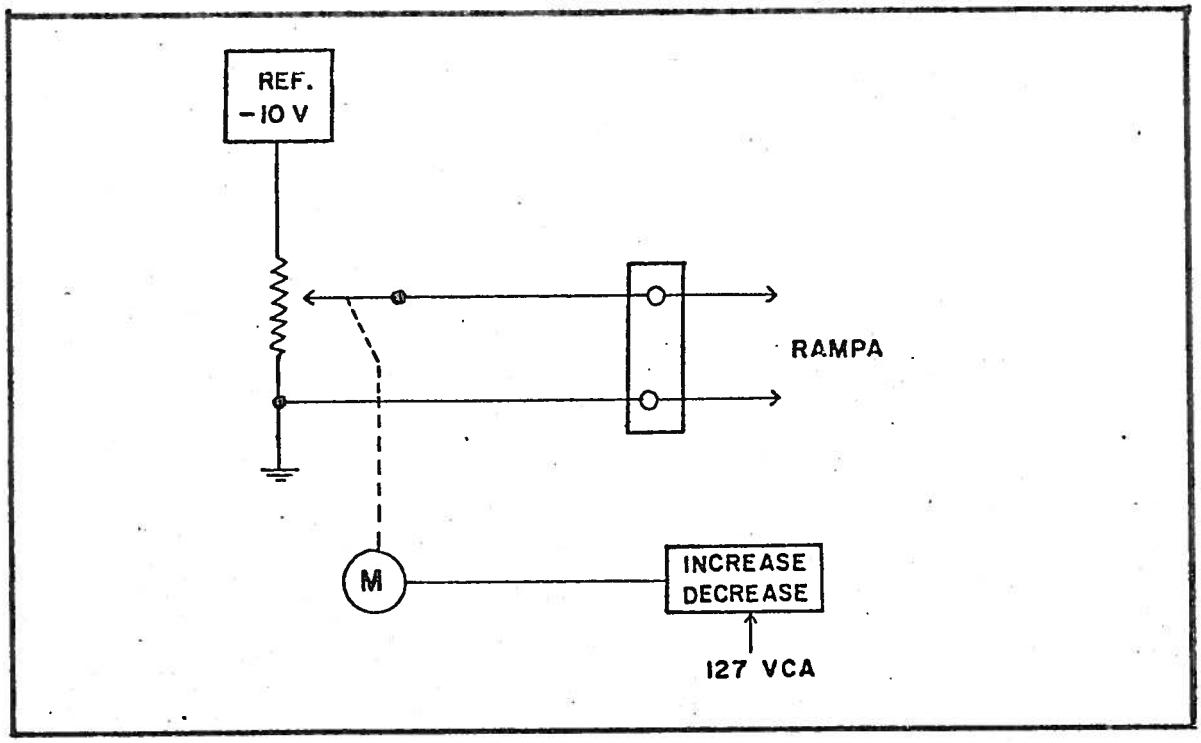


Fig 3-2: Controle da varredura - Diagrama simplificado do sistema existente no controlador Fieldial.

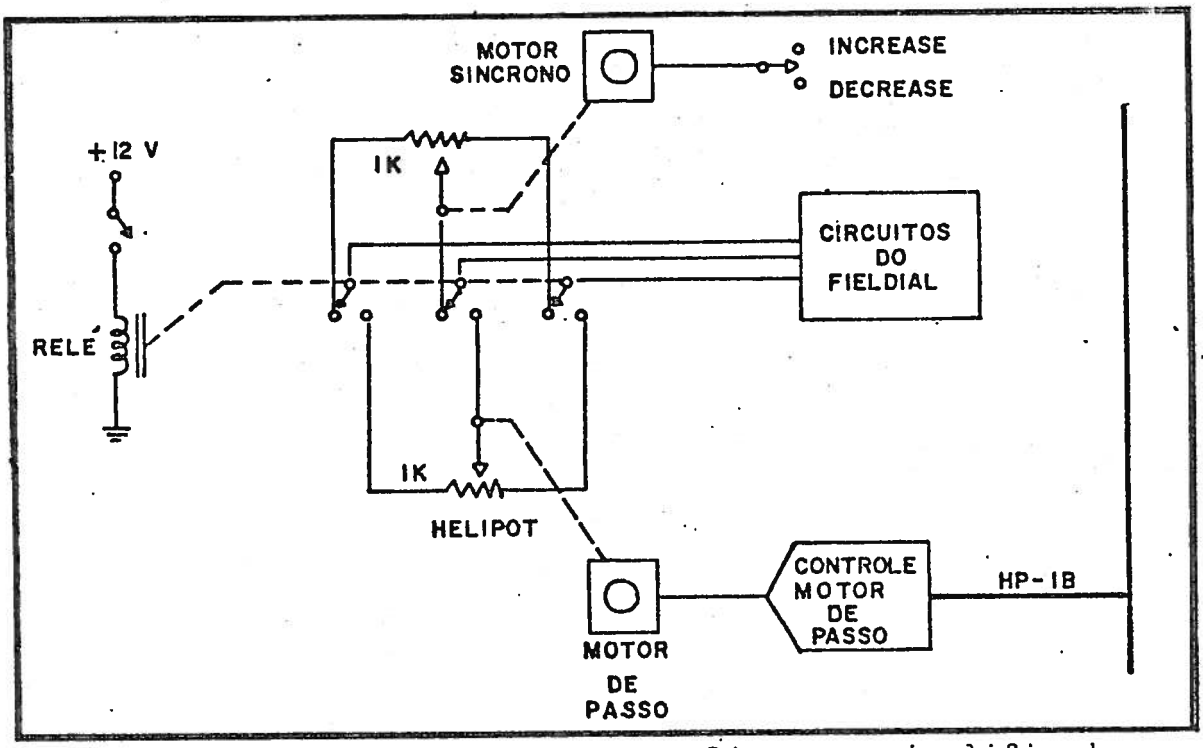


Fig 3-3: Controle da varredura - Diagrama simplificado das modificações introduzidas pelo SMOP.

A menos do motor de passo e do helipot, os instrumentos usados no desenvolvimento do Sistema do Motor de Passo foram exatamente os mesmos do sistema de aquisição passiva (SMAP) e estão listados na tabela 2-1.

O espectrômetro também não sofreu modificações, exceto, é claro, no mecanismo de geração da varredura.

O multiprogramador, devido às suas características de instrumento configurável, desempenha funções diferentes nos dois sistemas. No SMAP, como vimos, foram usados os cartões funcionais de relés, para comandar o acionamento do motor da varredura, e de DACs, para gerar o nível de referência IRI para o comparador. No sistema do motor de passo, por outro lado, foi usado o cartão funcional de saídas digitais para controle do motor de passo como veremos adiante, na seção 3.2.2. Os demais cartões estão sem função na presente configuração do sistema, mas nada impede que eles venham a ser utilizados em futuras aplicações do SMOP ou de outro sistema qualquer que venha a ser desenvolvido no laboratório.

Na Fig. 3-4 temos um diagrama geral das interligações dos instrumentos. Tal como no SMAP, o uso do barramento IEEE-488 (HP-IB) proporciona uma maneira simples e eficiente para o estabelecimento das comunicações com o HP 1000. O multiprogramador, com o cartão de saídas digitais acoplado a circuitos externos, funciona como controle do motor, como veremos mais adiante.

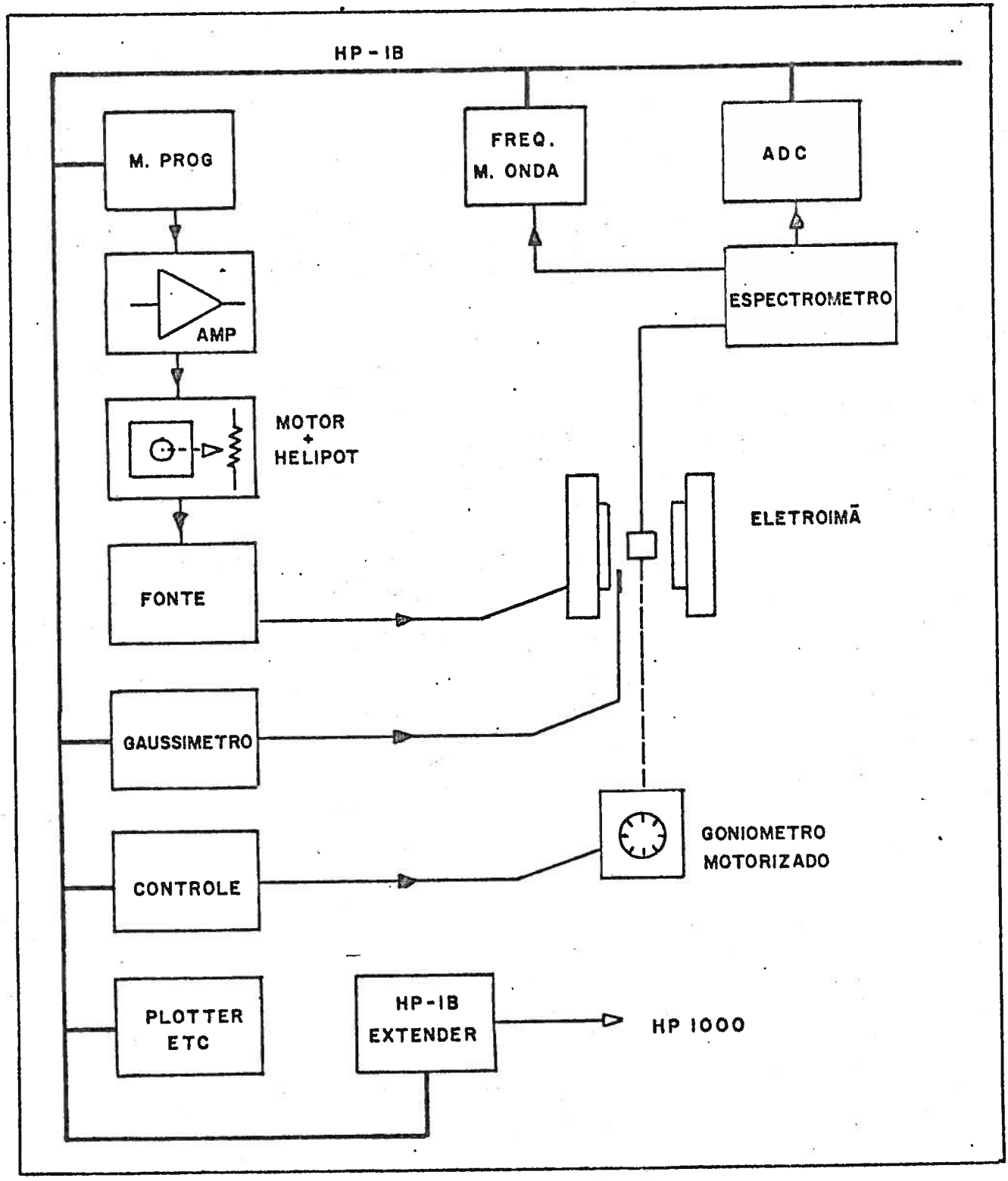


Fig. 3-4: Diagrama geral do Sistema do Motor de Passo.

3.2.1 O bloco motor de passo/helipot

O motor de passo é um tipo especial de motor que tem a propriedade de girar de uma maneira descontínua. Esses motores são empregados em uma variedade de instrumentos, que vão desde as grandes ferramentas e robôs até os goniômetros de alta precisão, acionadores de discos magnéticos, etc.

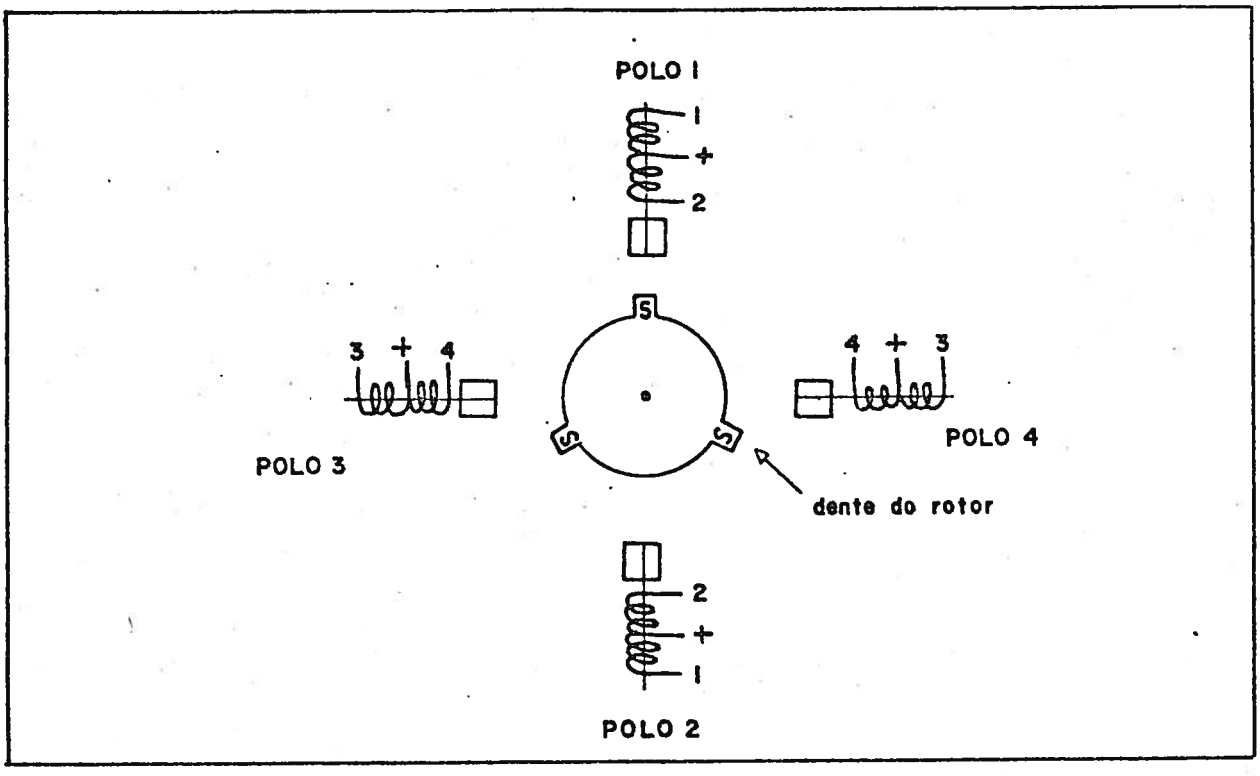


Fig. 3-5: Esquema simplificado de um motor de passo

A fig. 3-5 é um diagrama, bastante simplicado, que ilustra o princípio de funcionamento de um motor de passo. Um rotor, com 3 imãs permanentes, é montado em uma carcaça que contém 4 bobinas de campo simetricamente dispostas e ligadas, através de um cir-

cuito de comutação, a uma fonte de alimentação. Através do controle da direção da corrente em cada uma dessas bobinas, pode-se fazer com que elas apresentem um polo norte ou sul na região do rotor, fazendo com que este mude de posição para uma nova situação de equilíbrio estável.

Nos motores reais, o número de polos do rotor e o número de bobinas de campo determinam o tamanho angular do passo, sendo bastante comuns motores de 500 ou mais passos por revolução. No nosso sistema, usamos um motor de 200 passos, marca Synchro[18], comumente empregado no acionamento de pequenas impressoras de caracteres.

O potenciômetro usado é um helipot de 10 voltas, que tem esse nome pelo fato de o elemento resistivo estar disposto na forma de uma hélice ao longo da qual se desloca o cursor. Este componente é também muito empregado em instrumentos de precisão e está disponível comercialmente com vários valores de resistência. No nosso caso, usamos um helipot de 1000 ohms, tal como o potenciômetro original, com 5% de tolerância e 0,25% de linearidade.

Com essa combinação de motor e helipot, podemos dividir a varredura do campo em 2000 passos, onde cada passo corresponde a uma variação de 0,05% do valor escolhido no controle de "sweep range" da fonte.

3.2.2 O acionamento do motor de passo

Conforme vimos na secão anterior, o acionamento do motor de passo se dá através da comutação da corrente elétrica nas bobinas de campo. Para que se tenha um movimento coerente do motor em um dado sentido, as bobinas têm que ser ligadas e desligadas em uma sequência de configurações, chamadas "fases do motor", que depende do tipo do motor. No nosso caso, em que usamos um motor de 4 fases com enrolamento bifilar, a sequência de ligacão das bobinas está relacionada na tabela 3-1, onde 0 representa bobina desligada e 1 representa bobina ligada. Embora não seja uma regra geral, o acionamento das fases em ordem crescente faz o motor girar no sentido anti-horário.

Fase No.	Bobina 1	2	3	4	Angulo (graus)	No. de passos	Equiv octal
1	1	0	1	0	0	0	12
2	1	0	0	1	1,8	1	11
3	0	1	0	1	3,6	2	05
4	0	1	1	0	5,4	3	06
1	1	0	1	0	7,2	4	12
..		...			...	...	..

Tabela 3-1: Sequência de acionamento do motor de passo

Para controlar a corrente nas bobinas, existem vários circuitos de complexidade variável[19,20] com possibilidade, inclusive, de variar a velocidade e a aceleração do motor. Nesta aplicação, este nivel de sofisticação não se faz necessário e, por isso, todo o controle foi feito através do software do HP 1000 e do hardware do multiprogramador STD 85MP segundo o diagrama da Fig. 3-6.

Como pode ser visto no diagrama, os bits 0, 1, 2 e 3 de uma das linhas de saída da interface paralela do multiprogramador são conectados, respectivamente, aos circuitos de comutação das bobinas 1, 2, 3 e 4 do motor. Esses circuitos de comutação foram desenhados de tal forma que, quando o bit associado a uma dada bobina for 0 (zero), a corrente naquela bobina será nula, permitindo assim que o programa de controle acione qualquer configuração das bobinas, através do envio de sequências adequadas para a interface paralela do multiprogramador.

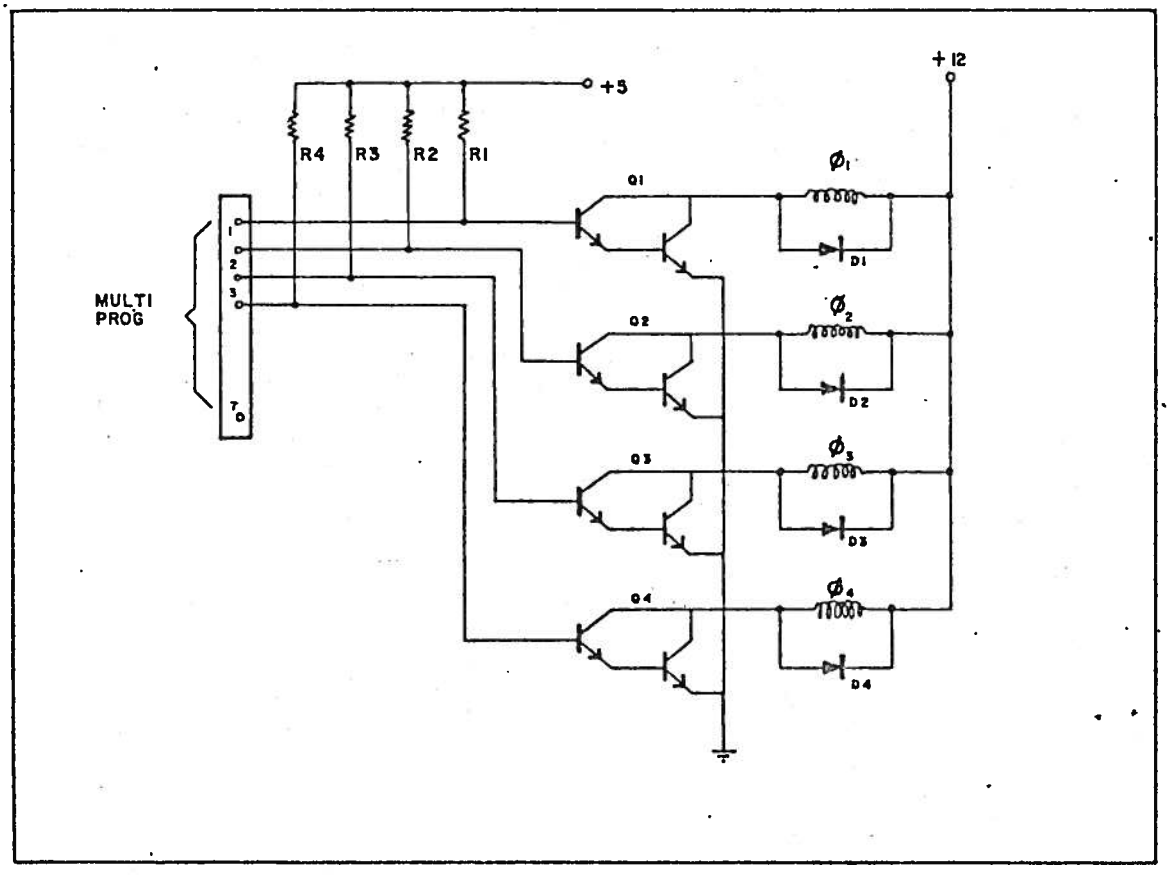


Fig. 3-6: Circuito de comutação para o motor de passo

O envio das sequências para as fases do motor pode ser bastante simplificado se observarmos que o padrão de ligações das bobinas se repete a cada 4 passos e equivale aos números octais 12, 11, 5 e 6, respectivamente (Tabela 3-1).

Com base nessa observação, uma rotina para girar o motor de N passos pode ser escrita em FORTRAN 77 da seguinte maneira:

```
      INTEGER*2    PASSO(0:3)
      DATA PASSO   /12B, 11B, 5B, 6B/
      WRITE(1,*) 'QUANTOS PASSOS ?'
      READ(1,*)  NP
      DO J  = 1, NP
        INDX  = IAND(3, J-1)
        ICONF = PASSO(INDX)
        WRITE(MPG, '("L",@2,"T")') ICONF
      END DO
```

Nessa rotina, a sequência de fases é armazenada em um vetor de 4 posições, numeradas de 0 a 3. Em seguida, após a leitura do número de passos (NP), é criado um "DO-LOOP" que se estende de 1 a NP. Neste loop, cria-se inicialmente um indexador, INDX, aplicando-se uma "máscara" (função lógica E, "IAND") para isolar os dois bits menos significativos do contador de passos J:

```
INDX = IAND(3, J-1)
```

Como consequência da aplicação dessa máscara, à medida que a variável J é incrementada, a variável INDX toma apenas os valores 0, 1, 2 e 3, que são usados para retirar do vetor PASSO as configurações das bobinas do motor:

```
ICONF = PASSO(INDX)
```

Conhecida a configuração correspondente a um passo J qualquer, a rotina simplesmente a envia para a interface paralela do multi-

programador:

```
WRITE(MPG, '("L",@2,"T")') ICONF
```

onde,

MPG = Unidade lógica do multiprogramador

L = Endereço da interface paralela no multiprogramador

@2 = Formato para dois dígitos octais

T = Protocolo de comando do multiprogramador

Devemos lembrar que o motor de passo é um dispositivo essencialmente sequencial e que, portanto, o deslocamento de uma posição para outra qualquer tem que passar, necessariamente, por todos os passos existentes entre elas.

No sistema SMOP, o controle do motor é feito por uma rotina escrita nos moldes da que acabamos de discutir. A mudança na direção de rotação é feita decrementando-se a variável J, de modo a acessar o vetor PASSO na ordem 3,2,1,0. Para desligar o motor e liberar o helipot para ajustes manuais, basta fazer todos os bits de controle iguais a 0. Neste caso, a corrente será nula em todas as bobinas e o rotor ficará livre.

3.2.3 Influência de motor de passo no controle do campo

O motor de passo, devido à natureza de sua construção, movimenta-se ao longo de uma série de posições de equilíbrio estando, portanto, sujeito a oscilar na transição de uma dessas posições para outra (Fig. 3-7)[20]. Por esse motivo, qualquer projeto que envolva sua utilização deve, de alguma forma, levar em conta essa

possibilidade e, se necessário, tomar as providências para o amortecimento dessas oscilações.

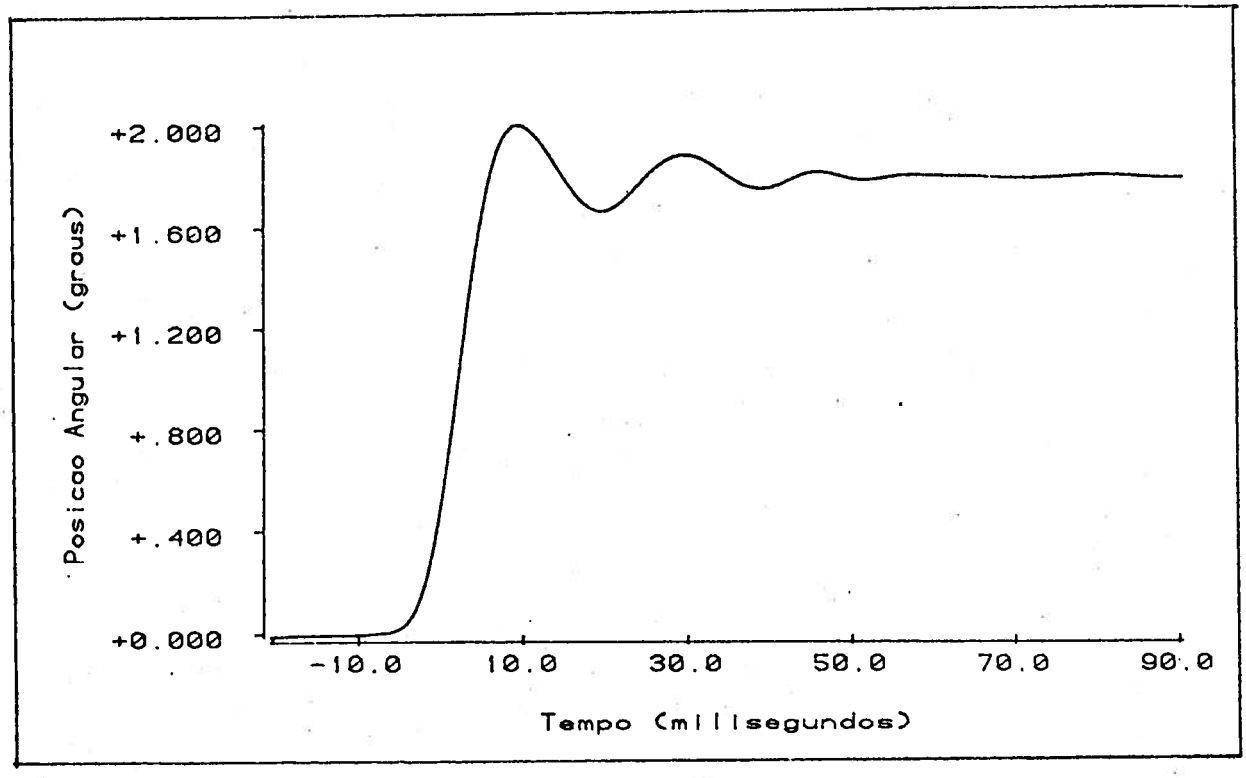


Fig. 3-7: Resposta do motor à função de passo único[18]

Quando se dispõe de todas as características eletromecânicas do motor e demais componentes a ele associados, é possível, pelo menos em princípio, fazer uma previsão do movimento do sistema[20].

Em nossa montagem, como essas características não eram conhecidas, tivemos de montar um sistema de medição para avaliá-las, tal como representado no diagrama da Fig. 3-8.

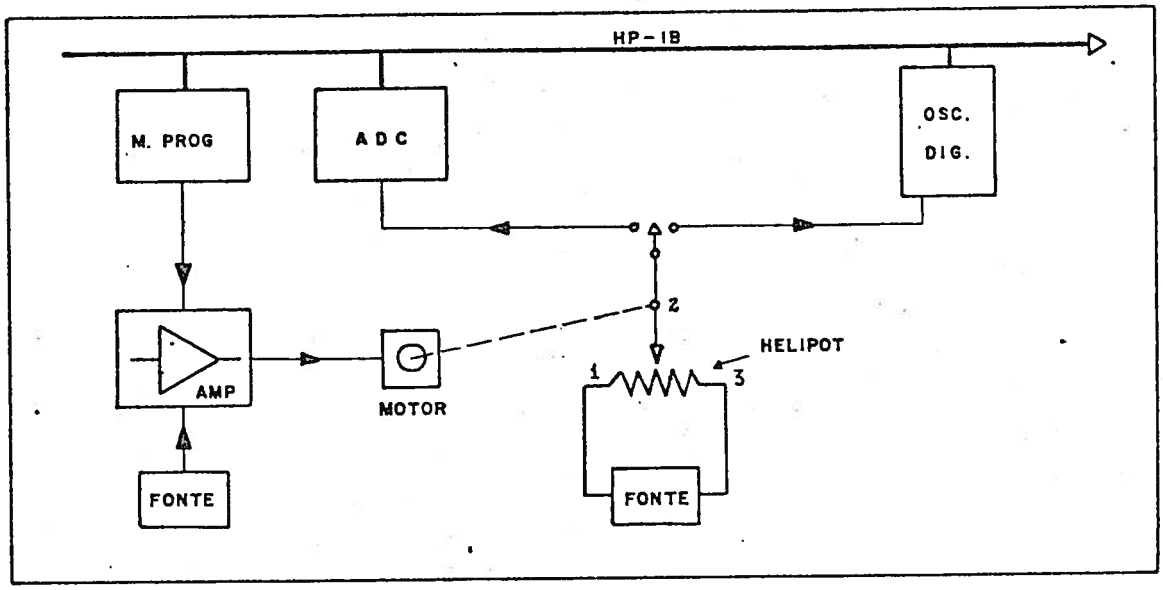


Fig. 3-8: Montagem usada para avaliação do motor de passo

Como se vê na figura, o helipot foi ligado a uma fonte de alimentação na forma de um divisor de tensão, cuja saída (pinos 1 e 2) foi ligada a um osciloscópio de armazenamento digital (Tektronix 7854). Como a tensão na saída é proporcional à resistência entre os pinos 1 e 2 e esta, por sua vez, é proporcional ao deslocamento angular do helipot, a tensão levada ao osciloscópio, Vo, será:

$$V_o = (V_i/R)*k\theta$$

onde

V_i = tensão da fonte

R = resistência do helipot (1000 ohms)

k = constante de proporcionalidade (1000 ohms/3600 graus)

θ = deslocamento angular

Como vemos, a tensão no divisor está diretamente relacionada com o ângulo θ e nos dá a informação desejada sobre a evolução do movimento.

Com essa montagem, realizamos medidas da tensão da rampa (escada) e da forma de onda nas bobinas do motor. Para medir a tensão da rampa, o sinal foi ligado à entrada DC do osciloscópio e a subida do primeiro pulso foi usada como "trigger" para disparar o modo AQS ("Acquire Single Shot") do osciloscópio, no qual os dados são convertidos e armazenados em uma única varredura. A tensão da fonte foi ajustada em 6,0 volts e o controle do motor foi programado para dar um passo a cada 800 milisegundos.

Os dados relativos aos 10 passos iniciais foram registrados e estão representados na Fig. 3-9, onde podemos observar que praticamente não existe "over-shoot" após a subida do degrau, ao contrário do que foi mostrado na Fig. 3-7. Esse amortecimento critico do passo do motor é devido, não só ao atrito existente no sistema mas, também, ao efeito de freio eletromagnético produzido pelos diodos D1,..,D4 (Fig. 3-6).

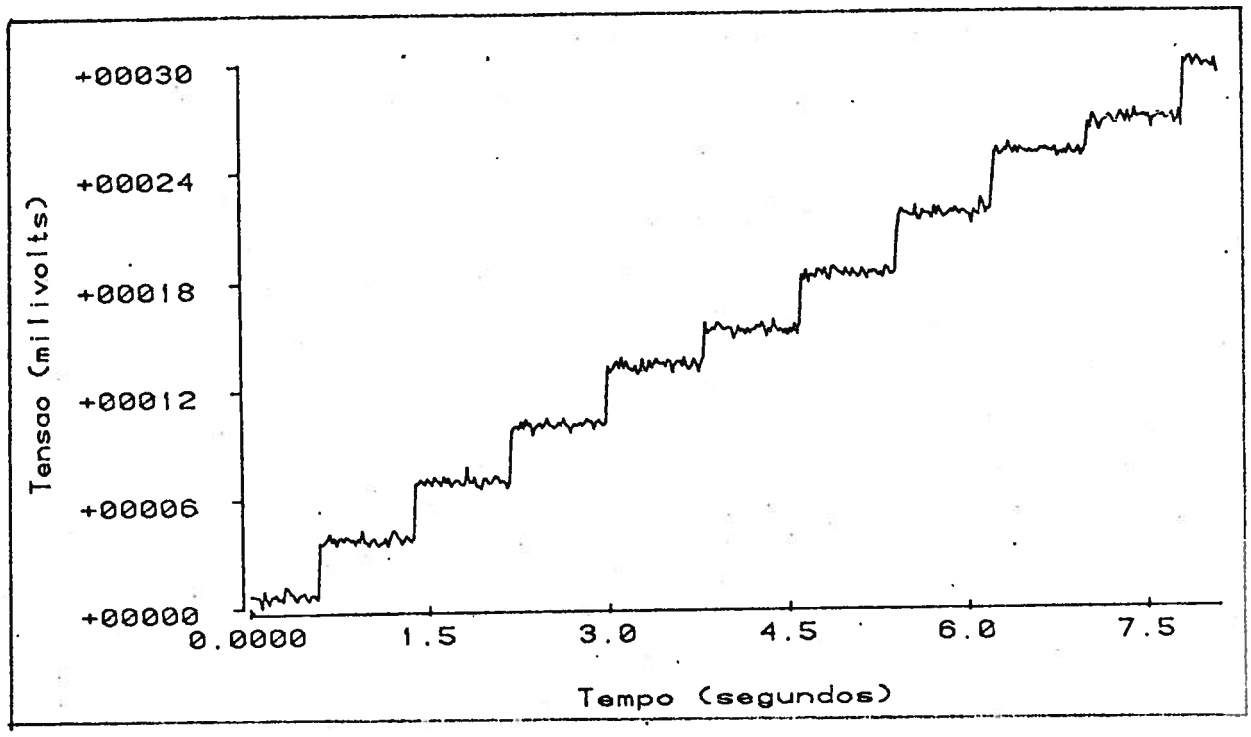


Fig.3-9: Registro direto dos degraus de tensão produzidos pelo sistema motor-de-passo/helipot. O intervalo entre os pulsos foi ajustado em 800 milisegundos. As irregularidades dos degraus 5 e 9 estão discutidas no texto.

Um detalhe que pode ser observado na Fig. 3-9 é a existência de passos irregulares nos "degraus" 5 e 9. Inicialmente, pensamos que este problema estivesse ligado ao movimento do motor mas, testes realizados com a substituição de vários componentes, nos levaram a crer que este problema só poderá ser resolvido com o uso de um helipot de maior linearidade. De qualquer forma, essas irregularidades não causam erro cumulativo e a perturbação local é de cerca de 0,025% do "sweep range", ou seja, 1 parte em 4000, o que é perfeitamente aceitável em espectros de EPR.

Mesmo representando uma pequena fração da varredura, esses dados permitem, como vimos, que se possa tirar conclusões importantes sobre o processo. Um registro com um maior número de passos é possivel mas,neste caso, os dados ficariam tão comprimidos no

gráfico, que seria difícil analisá-los.

Para termos uma visão mais ampla da evolução dos passos do motor, sem o problema da compressão, realizamos outras medidas onde registramos as variações de tensão na saida do divisor, usando a entrada AC do osciloscópio. Como os passos são uniformes, essas variações são iguais e podem, portanto, ser registradas no modo de aquisição repetitiva disponível no osciloscópio.

Nessas medidas, o sistema foi programado para mover o motor a uma taxa de 2 passos por segundo e a base de tempo do osciloscópio foi ajustada em 5 segundos para a acomodar 10 pulsos na tela. O registro foi feito no modo "average", com um total de 100 varreduras, totalizando, portanto, 1000 pulsos do motor (Fig. 3-10).

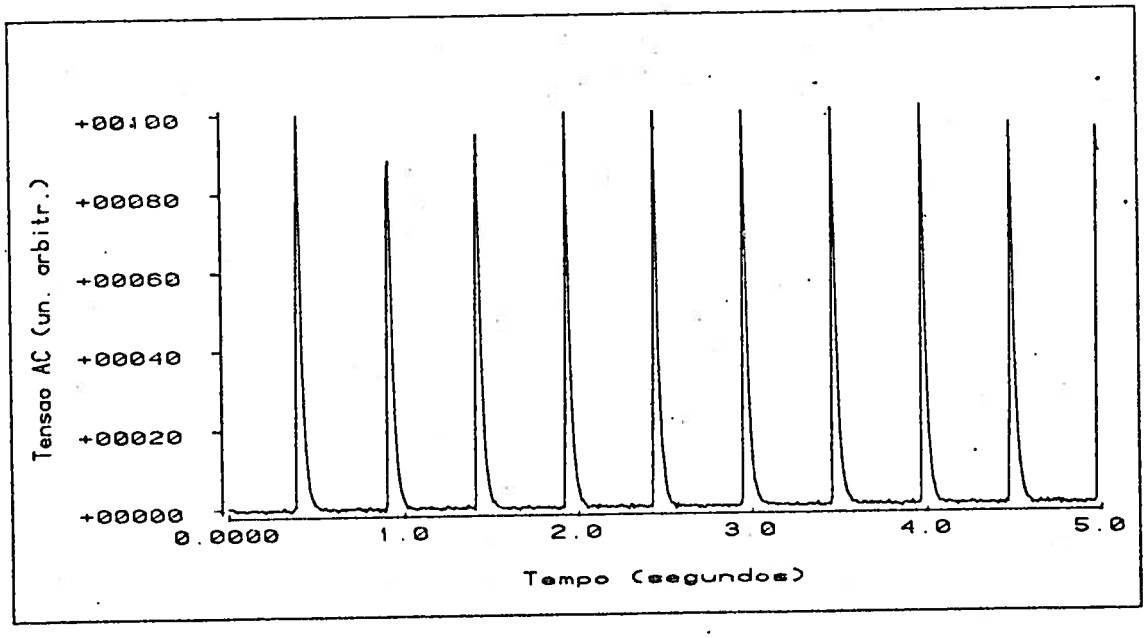


Fig. 3-10: Pulsos do motor, registrados no modo "average"

A evolução da tensão da rampa foi obtida dessas mesmas medidas a partir da integração numérica dos dados dos pulsos (Fig. 3-11).

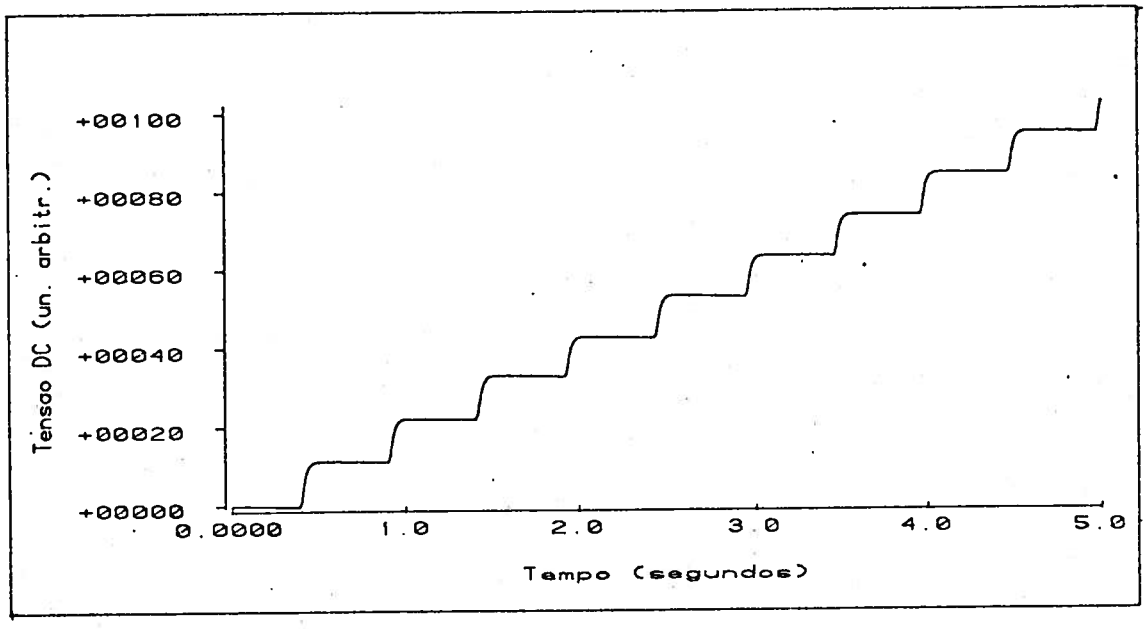


Fig. 3-11: Evolução da tensão da rampa obtida por integração numérica dos dados da Fig. 3-10

Com relação a esses resultados, é importante observar que o uso da entrada AC, embora só nos dê uma aproximação da derivada do sinal -por causa de sua constante de tempo-, nos permite avaliar, com facilidade, o tempo de estabilização do movimento do motor. Conforme se vê na Fig. 3-12, que é o registro de um único pulso em uma das bobinas do motor, este tempo pode ser estimado em 60 milisegundos o que está de acordo, inclusive, com as poucas informações fornecidas pelo fabricante do motor[20].

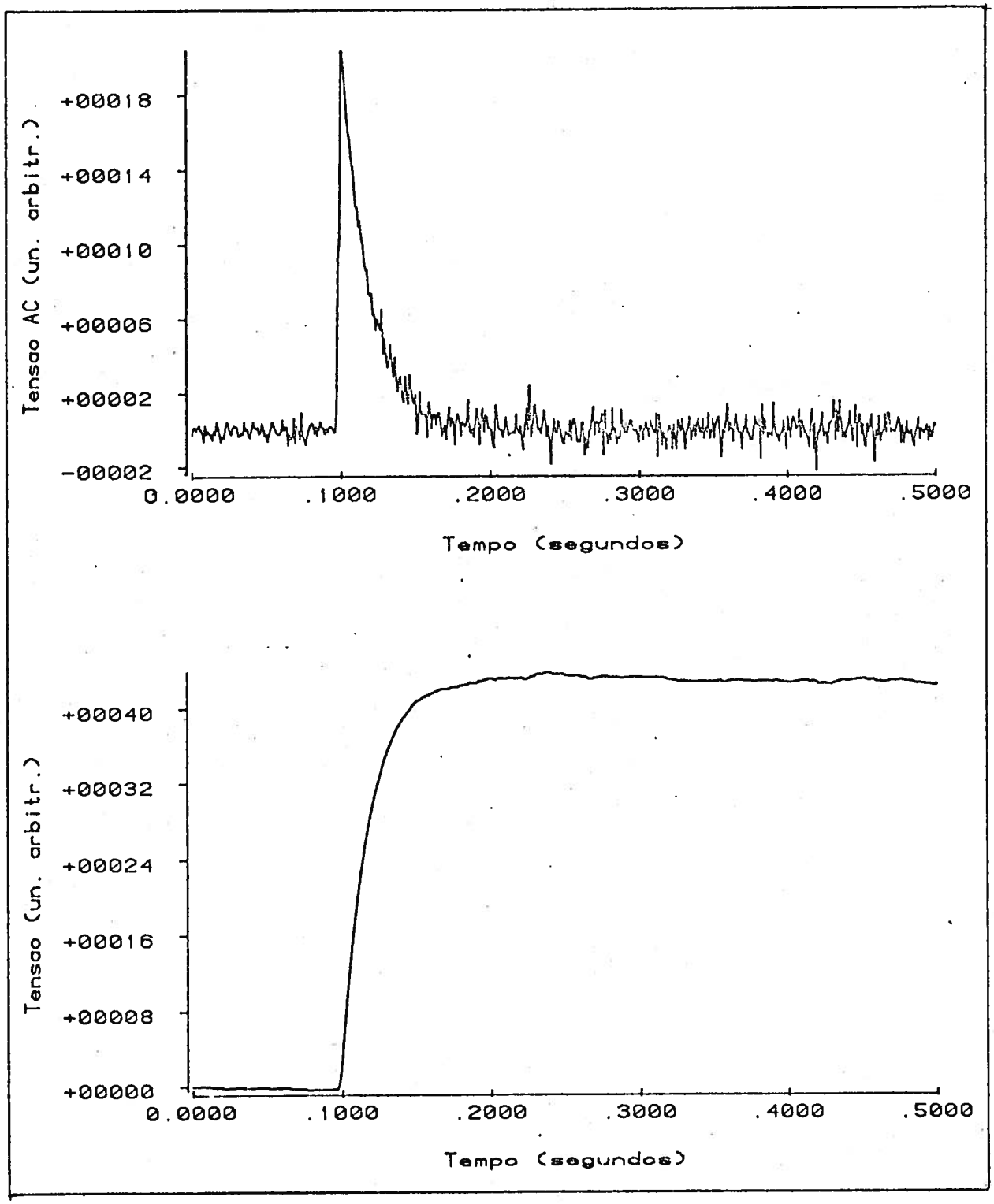


Fig. 3-12: Registro da variação de tensão na saída do divisor para o caso de um pulso. Na metade superior temos o registro do pulso e, na inferior, o "degrau" obtido por integração. O tempo de estabilização pode ser estimado em 60 milisegundos.

Para finalizar os testes de movimentação do motor, fizemos um registro da rampa de tensão gerada ao longo dos 2000 passos. Nesse caso, a saída do divisor foi ligada ao ADC, a fonte ajustada em 10 Volts e o sistema programado para gerar 2000 passos com um atraso de 50 milisegundos após cada passo, para estabilização.

Em cada ponto foram feitas 100 leituras da tensão, a uma taxa de 200 cps, e a média aritmética dessas leituras foi tomada como valor da medida estando os resultados apresentados na Fig. 3-13.

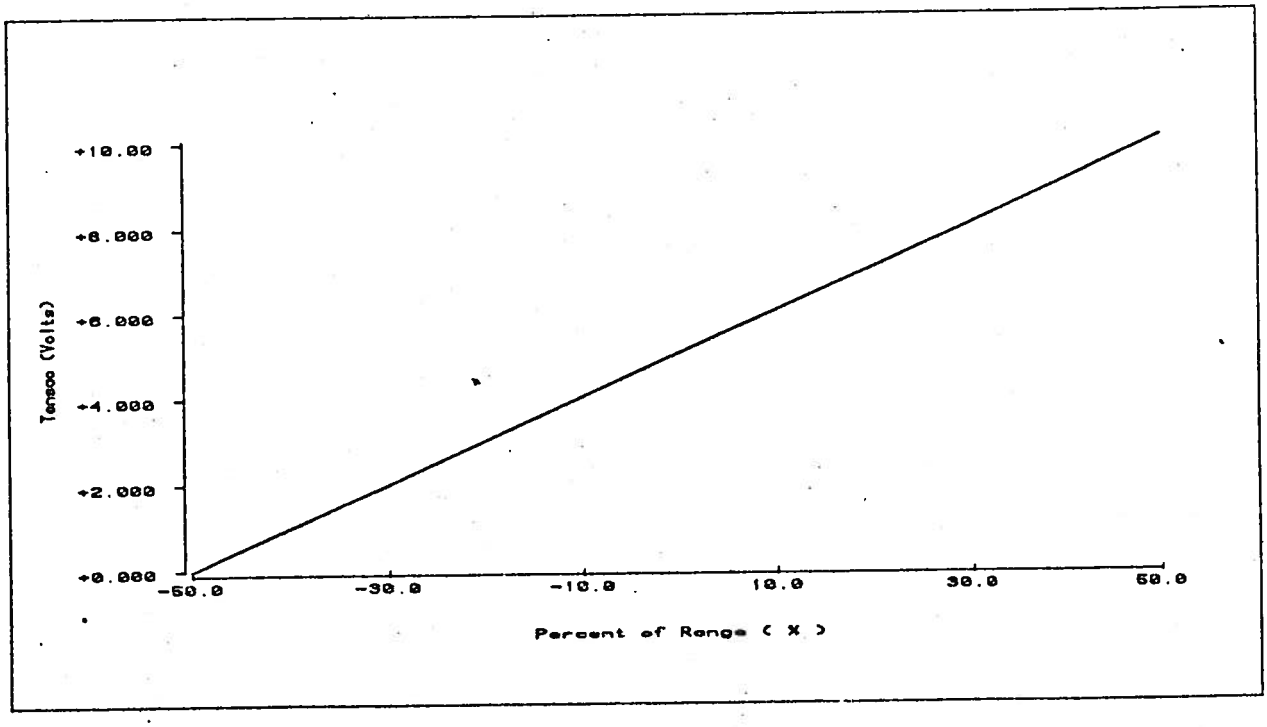


Fig. 3-13: Registro da rampa de tensão produzida pelo helipot acoplado ao motor de passo.

A interação desse sistema de varredura passo a passo com o controlador "Fieldial" não apresentou quaisquer problemas que pudéssemos detetar nos vários espectros que registramos e comparamos com os registros analógicos tradicionais.

Uma verificação direta da variação da intensidade do campo ao longo de um passo do motor foi feita a partir de uma aplicação sofisticada da Lei de Indução de Faraday. Para tanto, uma bobina de 12000 espiras e cerca de 10 cm^2 de secção reta foi colocada no entreferro do eletroimã e o pulso de tensão, induzido pela variação do campo, foi registrado na memória de um osciloscópio digital de amostragem rápida. Depois de amostrado, esse pulso foi transmitido ao computador onde, por integração, foi obtida a curva de variação do campo (Fig. 3-14).

Nas aplicações típicas do SMOP, cada passo do motor resulta em uma variação de cerca de 1 gauss na intensidade do campo. Essa variação produz, na bobina utilizada, uma FEM de poucos milivolts que chega ao osciloscópio bastante corrompida pelo ruído eletromagnético do ambiente e da própria fonte do eletroimã. Para atenuar esse ruído, foi colocado um capacitor de 2,2 uF em paralelo com a bobina e foram feitas dezenas de medidas repetitivas com auxílio do sistema de "averaging" disponível no osciloscópio. O sincronismo das medidas foi obtido com o uso de um sinal de gatilho externo, gerado pelo computador a cada passo do motor conforme pode ser visto na listagem do programa RMU04, apresentada no apêndice 5.

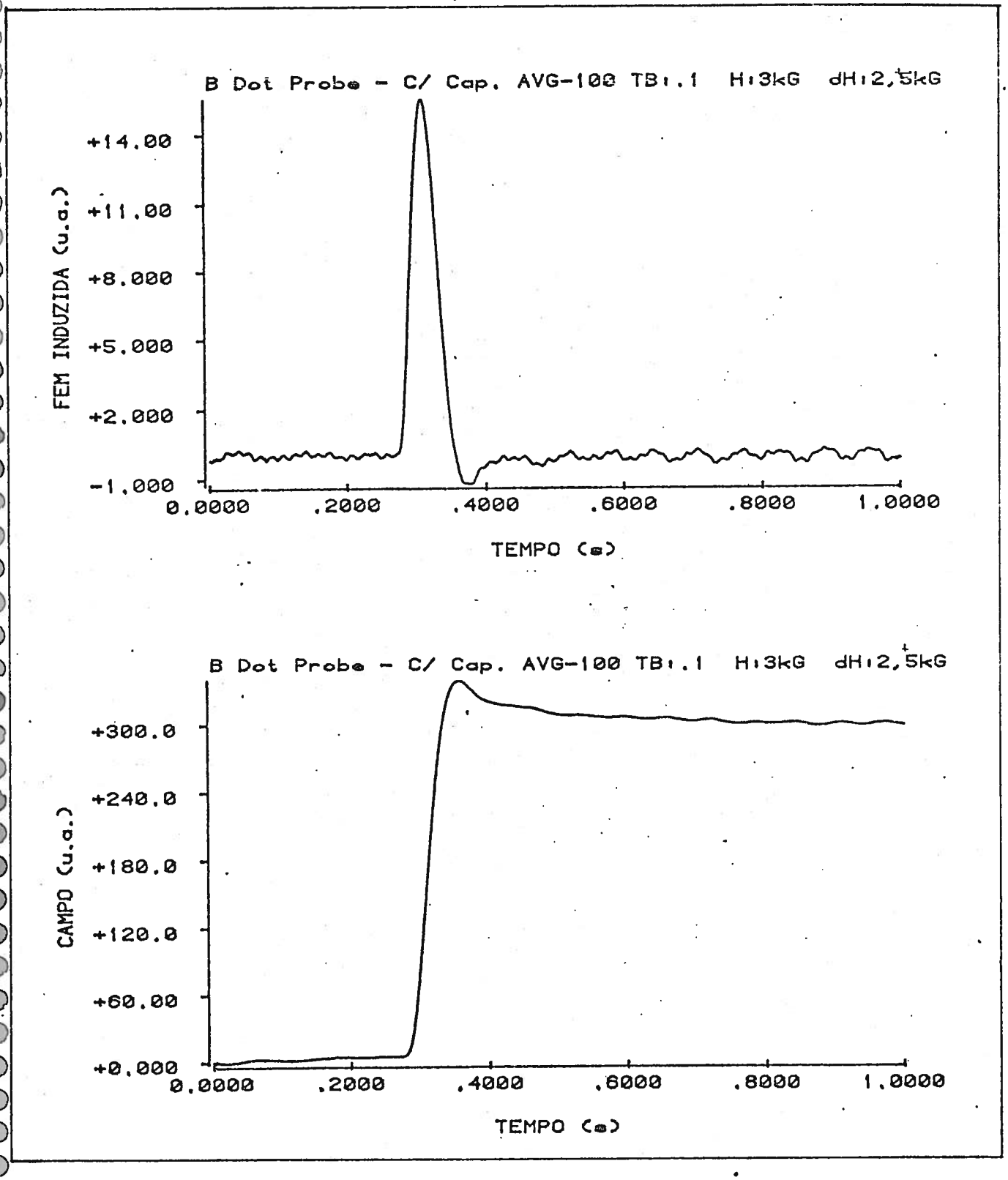


Fig. 3-14: Variação da intensidade do campo, medida pela FEM induzida. Acima, pulso de tensão induzido na bobina - $\mathcal{E}(t) = d\Phi/dt$. Abaixo, variação do campo obtida por integração - $B(t) = \int \mathcal{E}(t)dt$.

Os resultados dessas medidas indicam a existência de um "overshoot" de cerca de 10% da variacão do campo, com um tempo de recuperação de no mínimo 200 ms, para um tempo de subida do pulso de 70 ms o que é bastante coerente com os resultados obtidos para o movimento do motor (Fig. 3-11).

Nas medidas realizadas com o SMOP, verificamos experimentalmente que o tempo de espera pela estabilização do campo podia ser reduzido até 50 ms sem que fosse notada qualquer distorção nos espectros. Essa verificação foi feita através do procedimento sugerido pelo fabricante[21], que consiste em registrar o mesmo espectro em duas varreduras simétricas (increase, decrease).

Neste caso, se o controlador não conseguir acompanhar de perto o "set-point" (ou seja, a rampa gerada pelo helipot), as linhas dos espectros obtidos na subida e na descida do campo aparecerão em posições diferentes, indicando que a varredura está rápida demais para o controlador e que, portanto, o intervalo entre pulsos deve ser aumentado de modo a tornar a varredura mais lenta.

Nas medidas realizadas em cristais de RDA, que serão apresentadas no final deste capitulo, este procedimento foi usado sistematicamente, sem apresentar qualquer problema.

3.3 O registro dos espectros

Tal como no sistema de aquisição passiva, o registro do sinal de ressonância no SMOP é feito através do ADC HP 59313A e as medi-

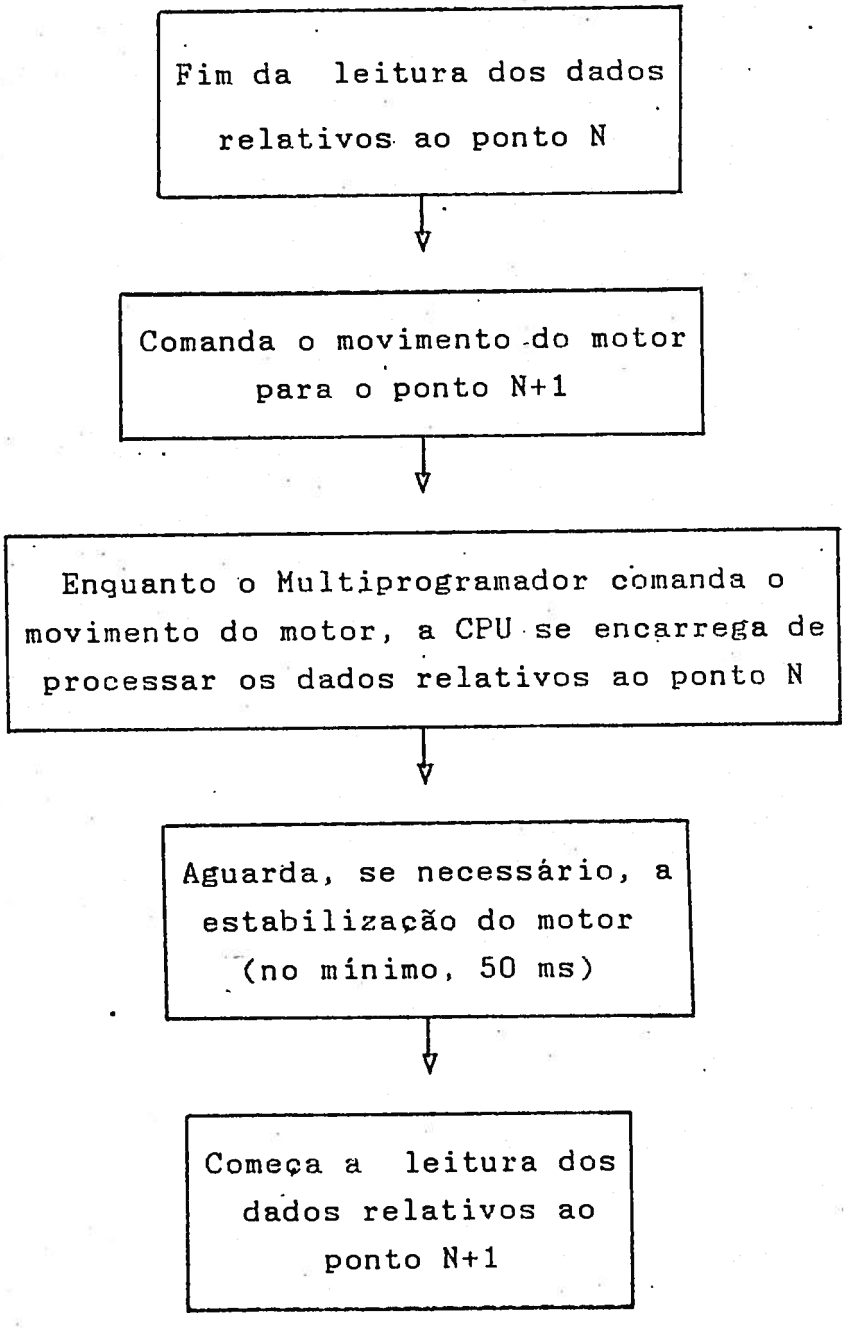
ções da intensidade do campo são feitas com os gaussimetros Varian ou Metrolab. Para evitar que o processo de aquisição tenha de ser interrompido para mudanças de escala ou de pontas de prova do gaussímetro, foi usado um esquema de calibração de posições do helipot, semelhante ao método de calibração da rampa descrito na seção 2.2.1.

Para o registro do sinal de ressonância, o sistema permite o uso da técnica de média ponto a ponto, além do sistema tradicional de média ("signal averaging") baseado na soma coerente dos espectros obtidos em várias varreduras.

A técnica de média ponto a ponto consiste em fazer um bloco de leituras do sinal, mantendo-se fixo o campo magnético, para depois submeter esse bloco a um processamento matemático conveniente como, por exemplo, cálculo de médias, filtros digitais, etc. Embora no momento, o único processamento disponível no sistema seja o cálculo da média aritmética, a inclusão de outras operações é uma tarefa relativamente simples, em virtude da estrutura modular do programa. As limitações dessa técnica serão discutidas na seção 3.4.1.

Como vimos na seção 3.2.3, é de grande importância que o motor esteja estável para que se dê início às conversões relativas a um ponto qualquer do espectro. Por essa razão, o programa de controle deve assegurar a existência de um intervalo de tempo da ordem de 50 milisegundos entre o comando de passo e o comando de medi-

ção. Em um espectro de 1000 pontos, a acumulação desses intervalos resulta em um "tempo morto" da ordem de 1 minuto, tempo este que pode ser aproveitado para fazer os cálculos do próprio programa de aquisição, desde que seja usada a lógica de programação mostrada abaixo:



Conforme se vê, após a estabilização do motor em um dado ponto do campo, é feito um registro do sinal e em seguida, antes de se iniciar o processamento desse registro, é dado um comando de passo para o motor. Terminado o processamento, pode-se ainda dar um atraso de X milisegundos no programa, após o que o processo se repete até que o último ponto de interesse seja medido. Em resumo, o que se faz com essa lógica é processar as medidas relativas ao ponto N, enquanto o motor se desloca para o ponto N+1, aproveitando assim o "tempo morto" decorrente desse deslocamento.

O valor do atraso X a ser colocado após os cálculos vai depender da complexidade dos mesmos e pode ser ajustado experimentalmente ou avaliado "a priori" através de rotinas que simulem o procedimento matemático ao qual o bloco de dados vai ser submetido. No sistema RTE, o atraso na execução é obtido com resolução de 10 milisegundos através do comando de "auto schedule" no qual o próprio programa se coloca em estado de suspensão, após informar ao sistema operacional a hora em que deverá ser "despertado". Depois de reativado, o programa prossegue sua execução a partir do ponto em que foi suspenso.

Toda a operação do Sistema do Motor de Passo é comandada pelo programa SMOP que, tal como no sistema de aquisição passiva, foi escrito no estilo de um sistema operacional onde a interação com o usuário se processa através de comandos. Uma descrição geral do programa e dos comandos encontra-se no Apêndice 3.

3.4 Recursos adicionais do SMOP

O programa SMOP oferece ao usuário uma série de recursos destinados a facilitar a condução da experiência de EPR e que são essencialmente os mesmos disponíveis no sistema de aquisição passiva. Uma descrição desses recursos encontra-se na seção 2.3, razão pela qual não serão reapresentados aqui, com exceção do problema da escolha da taxa de aquisição dos dados, que será discutida em seguida.

3.4.1 A escolha da taxa de aquisição dos dados

Assim como no sistema de aquisição passiva, a frequência de operação do ADC é passada ao programa SMOP com o comando FC. O critério para a escolha dessa frequência, entretanto, é completamente diferente, já que no SMOP as conversões se processam com o campo magnético estacionário e referem-se, portanto, à intensidade do sinal em um único ponto do espectro.

No processo de aquisição, o programa permite que se faça até 1024 conversões para cada ponto do espectro nas frequências de 5, 10, 20, 50, 100 e 200 cps. Os resultados dessas conversões são armazenados em um vetor e o sinal de ressonância é obtido da média aritmética desses resultados. Para que esse processo possa levar a uma medida representativa do sinal é necessário entretanto que a autocorrelação das amostragens seja suficientemente pequena[21] para que haja convergência da média. Caso contrário, a variância da média será sempre da mesma ordem que a variância das amostra-

gens qualquer que seja o número das mesmas.

Com base no que acabamos de expor, podemos concluir que para termos resultados significativos no cálculo da média, é necessário que as conversões tenham um espaçamento tal que cada uma delas não contenha nenhuma "memória" da conversão anterior. Como não podemos determinar "a priori" o tipo de ruído que estará presente em uma dada experiência, torna-se praticamente inviável o estabelecimento de um critério para a escolha da frequência de aquisição, a menos que se faça uma análise do espectro do sinal, o que não é possivel no estágio presente de desenvolvimento do SMOP.

Nos vários testes realizados com este sistema, obtivemos resultados satisfatórios usando uma taxa de 100 cps mas, de um modo geral, a frequência mais apropriada terá de ser determinada, caso a caso, através da inspeção dos resultados.

3.5 A resolução dos espectros no SMOP

Como vimos na seção 2.3.1, a taxa de conversão no sistema de aquisição passiva determina a resolução do espectro ou seja, o número de pontos por gauss. No SMOP a taxa de conversão está relacionada com a qualidade da média do sinal e nada tem a ver com a resolução.

De acordo com a eq. 2.17, a densidade de amostragem de um espectro deve ser no minimo igual a 2/L, onde L é a largura da linha

mais estreita existente. Devido às características eletromecânicas do sistema de varredura passo a passo, a densidade de amostragem no SMOP será sempre dada por

$$d_a = (NPPR * NVH)/SR \qquad (3.1)$$

onde

SR = valor do "sweep range"

NPPR = número de passos por revolução

NVH = número de voltas do helipot

No nosso sistema, NPPR = 200 e NVH = 10 de modo que

$$d_a = 2000/SR \qquad (3.2)$$

Comparando este resultado com a eq. 2.17, podemos escrever então que

$$SR = (2000/2)*L \qquad (3.3)$$

A equação acima nos permite avaliar o valor mínimo do "sweep range", para que o espectro seja corretamente representado do ponto de vista da teoria da amostragem, ou seja, para que o mesmo contenha no mínimo dois pontos na linha mais estreita.

A aplicação pura e simples deste resultado pode nos levar a situações tecnicamente inviáveis, pois a aquisição de linhas mais largas que 15 gauss exigiria um valor de varredura (15000 gauss) impossível de ser conseguido pelo eletroimã do espectrômetro. A solução para este problema consiste simplesmente em trabalhar com um maior número de pontos na linha mais estreita o que, aliás, é desejável, conforme visto na seção 2.3.1., embora possa acarretar um acréscimo no nível de ruído de alta frequência da experiência.

3.6 Aplicações e resultados

O SMOP é um sistema que foi desenvolvido com o objetivo de fazer registros mais precisos de espectros de EPR tendo em vista principalmente os processos de "signal averaging". Dependendo das características do ruído presente no sistema, a média pode ser feita em uma varredura apenas, com uma substancial economia de tempo. Além disto, o programa pode ser usado para fazer medidas de uma varredura com apenas uma conversão por ponto, ou mesmo fazer uma combinação de múltiplas varreduras com múltiplas conversões em cada ponto.

No início das aplicações do SMOP, nossas atenções se concentraram no desempenho do conjunto motor de passo/helipot. Embora já tivéssemos testado o sistema com a geração de rampas de tensão contínua (Fig. 3-13), sabíamos que a sua instalação no "Fieldial Varian" seria um problema diferente, já que neste caso, a tensão a ser controlada, além de ser alternada (1300 Hz), faz parte de um elo de realimentação de um sistema síncrono (PLL). Para evitar interferências, o relé de comutação (Fig. 3-4) foi instalado junto ao potenciômetro original de "sweep", aproveitando-se a blindagem já existente. A ligação do relé com o helipot (que fica localizado externamente à fonte) foi feita com cabos blindados do tipo que se usa em circuitos de áudio. Com a fonte em funcionamento, verificamos que essa instalação não causou qualquer tipo de problema com a estabilidade do campo e pudemos então iniciar a operação do SMOP.

As primeiras medidas foram feitas com o objetivo de se avaliar o comportamento da varredura. Para isto, escrevemos o programa DSMP7 (Ap.4), que aciona o motor de passo ao longo dos 2000 pontos disponíveis e, para cada ponto, registra a intensidade do campo com auxilio de um gaussimetro GPIB (Varian E-500). A partir dos dados coletados por este programa, calculamos a "largura" total da varredura, o valor médio do passo e o desvio padrão do valor do passo. Na tabela 3-2, apresentamos alguns dos resultados obtidos nessa experiência.

Arqv: tgsv8		Arqv: tgsv2	
Nome: Reglao dos 6000 - 15/6/89		Nome:. Teste rampa GSV #2 14/06/89	
NP:.. 2000		NP:.. 2000	
Inicio:.....	6287.9	Inicio:.....	1747.2
Fim:.....	7277.7	Fim:.....	2249.
Varredura total:.....	989.8	Varredura total:.....	501.8
Valor medio do passo:	.49515	Valor medio do passo:	.25103
Desvio padrao:.....	9.85409E-2	Desvio padrao:.....	6.901510-2

Tabela 3-2: Resultados da varredura controlada por motor de passo. NP é o número de pontos e os valores de campo são expressos em gauss.

Os dados acima mostram resultados obtidos nas regiões dos 1000 e dos 6000 gauss e pode-se observar a pequena dispersão no valor médio do passo. As varreduras foram ajustadas para 500 e 1000 gauss, respectivamente, e as pequenas discrepâncias observadas podem ser minimizadas através de uma recalibração da fonte. Outros resultados obtidos nessas medidas são apresentados, graficamente, nas Figs. 3-15, 3-16, e 3-17.

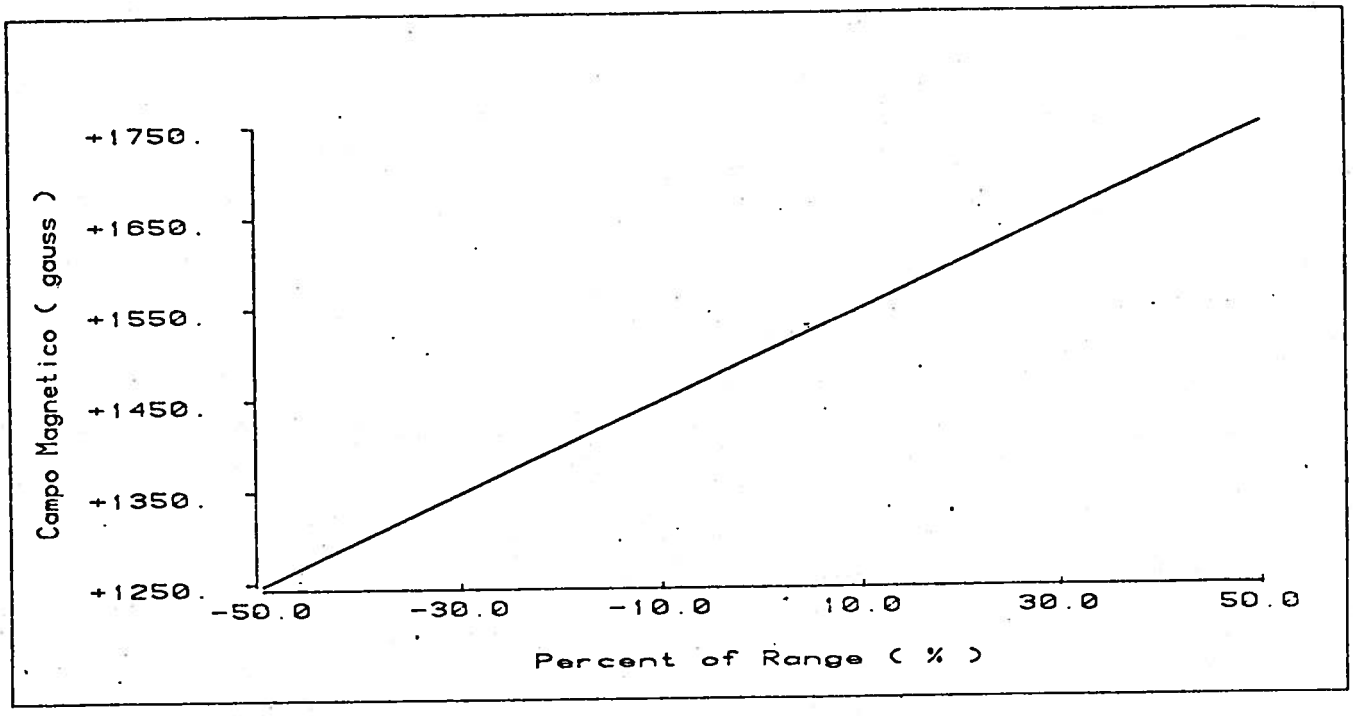


Fig. 3-15: Rampa controlada pelo SMOP - região dos 1000 gauss

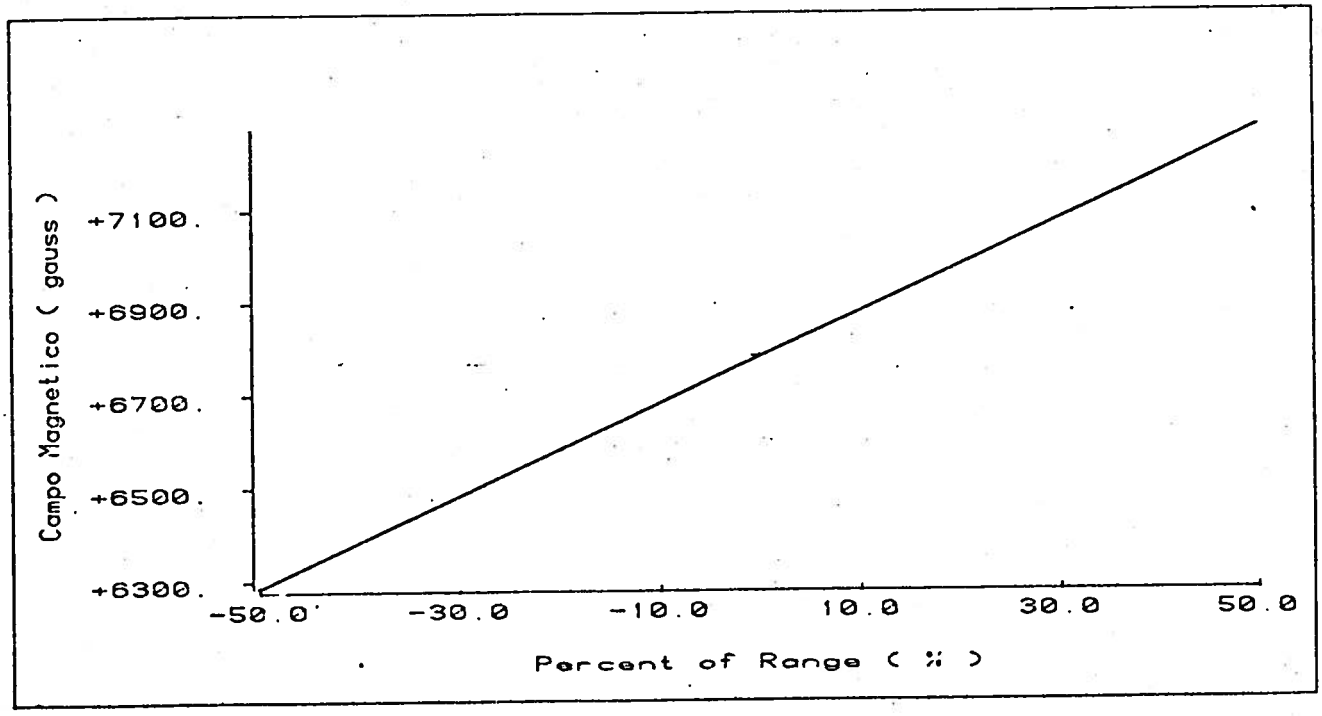


Fig. 3-16: Rampa controlada pelo SMOP - região dos 6000 gauss

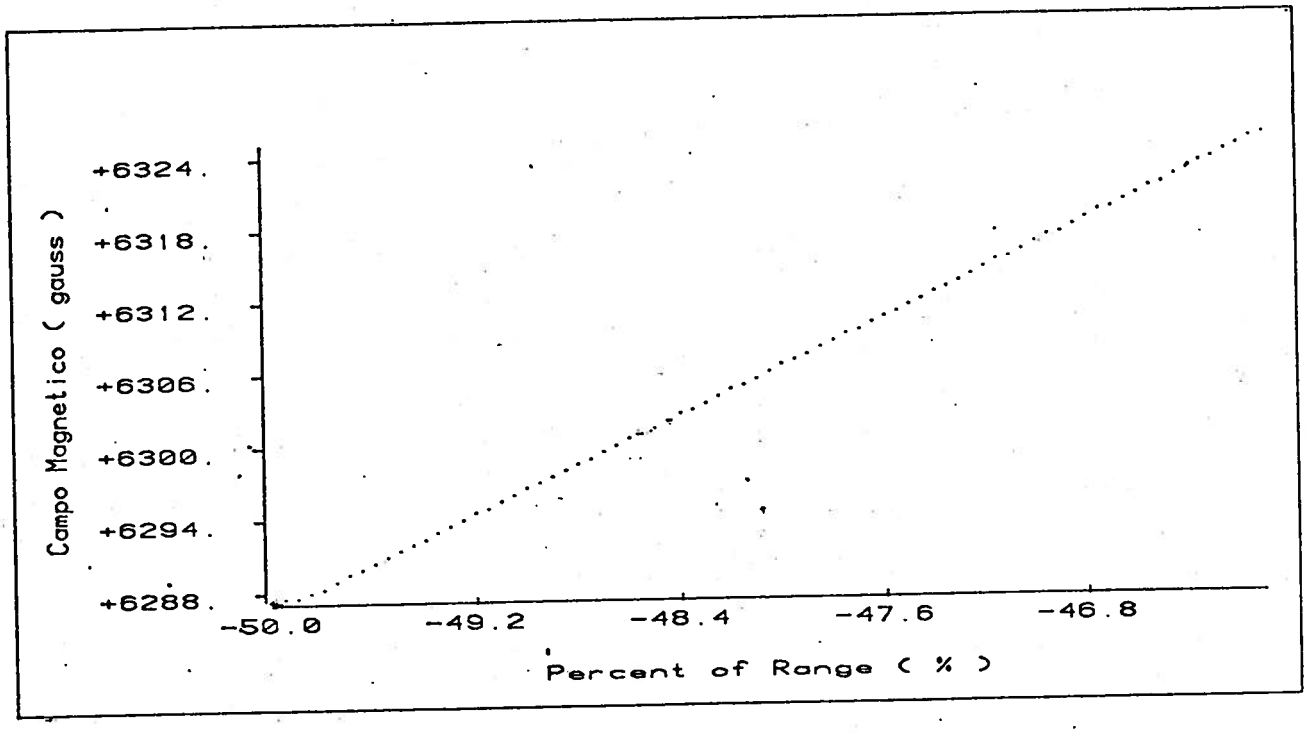


Fig. 3-17: Rampa na região dos 6000 gauss - detalhe
O patamar em -50% é devido ao helipot

Depois de nos certificarmos da confiabilidade da varredura geràda pelo SMOP, através das medidas que acabamos de descrever, concentramos nossa atenção nas experiências de EPR em RDA:Tl^{2+} que estavam sendo realizadas no laboratório[13], pois nelas havia um grande interesse em se estudar o comportamento de uma linha, na região dos 7400 gauss, que aparecia bastante mascarada pelo ruído na linha de base do registro analógico tradicional (Fig. 3-18).

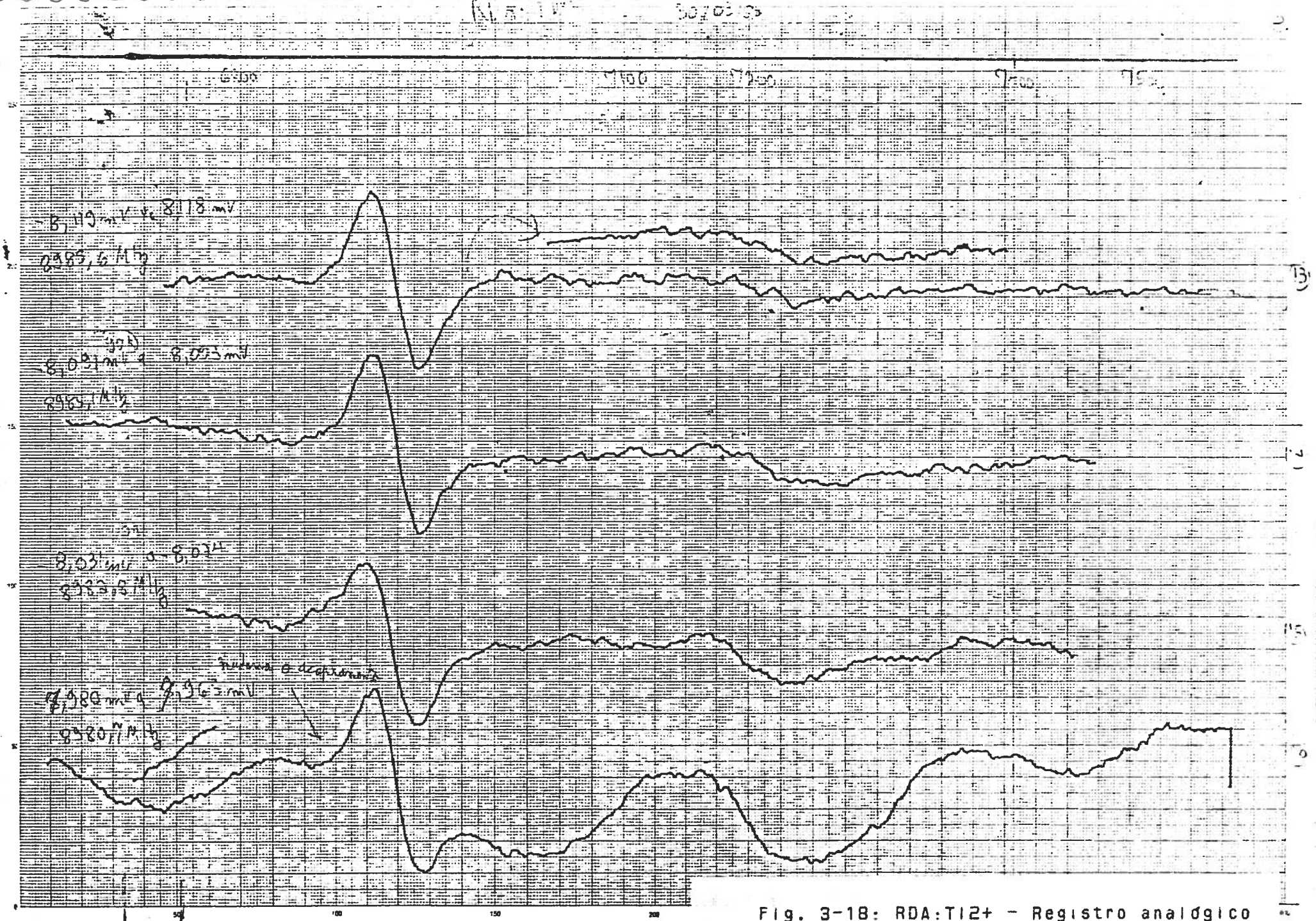
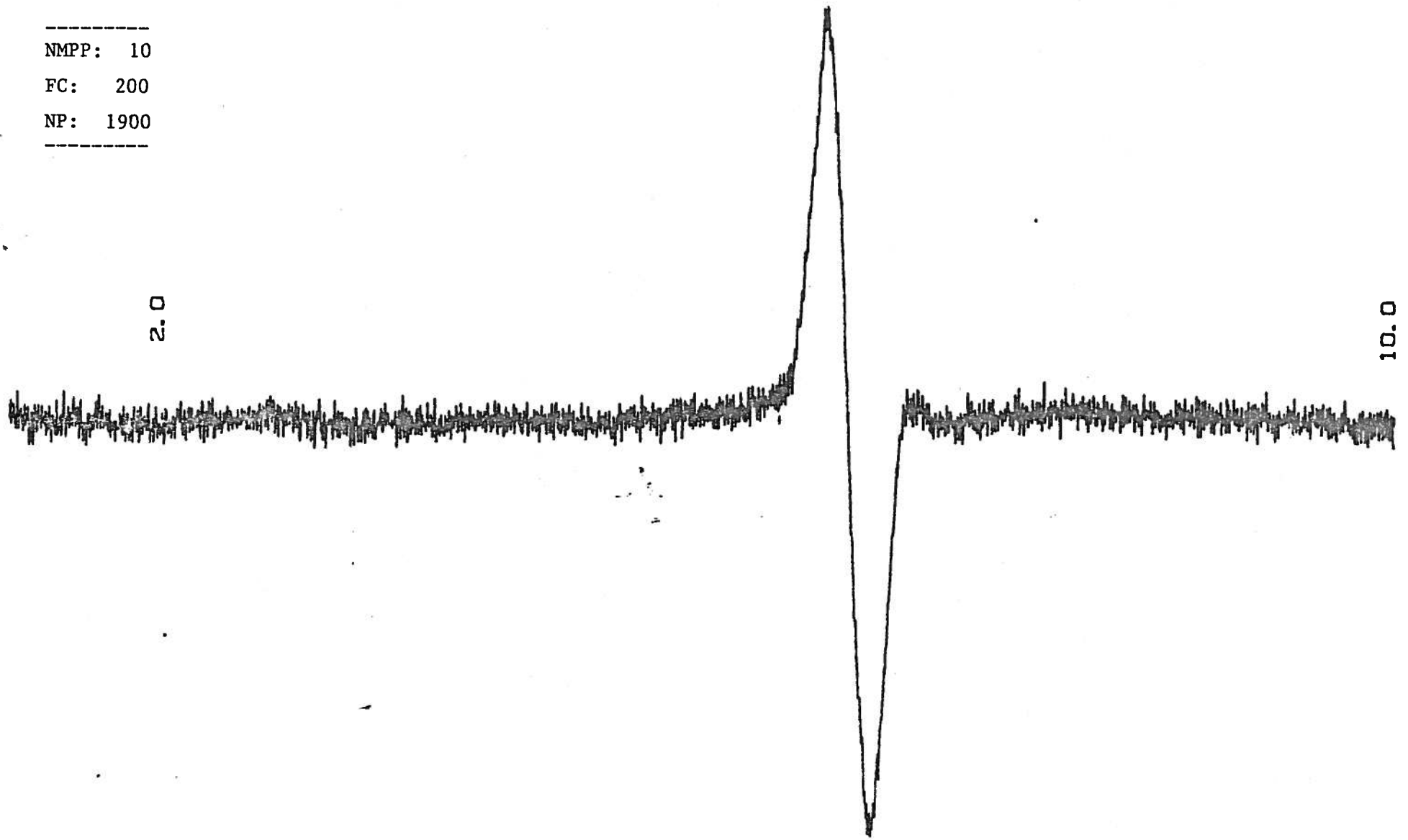


Fig. 3-18: RDA:TI2+ - Registro analógico

O primeiro conjunto de medidas das amostras de RDA:Tl^{2+} teve como objetivo a determinação de valores adequados para a frequência de conversão (FC) e para o número de medidas por ponto (NMPP).
Os resultados obtidos nos indicaram que medidas realizadas a uma taxa de 200 cps (conversões por segundo), com 10 leituras por ponto, já produziam resultados aceitáveis, embora a linha de base ainda aparecesse com um ruido relativamente alto, conforme se vê na Fig. 3-19. Outras medidas, realizadas com a mesma taxa (200 cps), mas com 100 e 500 leituras por ponto, mostraram de maneira inequívoca, a eficiência do método de média ponto a ponto, implementado pelo SMOP, na redução do ruído do espectro (Figs. 3-20 e 3-21).

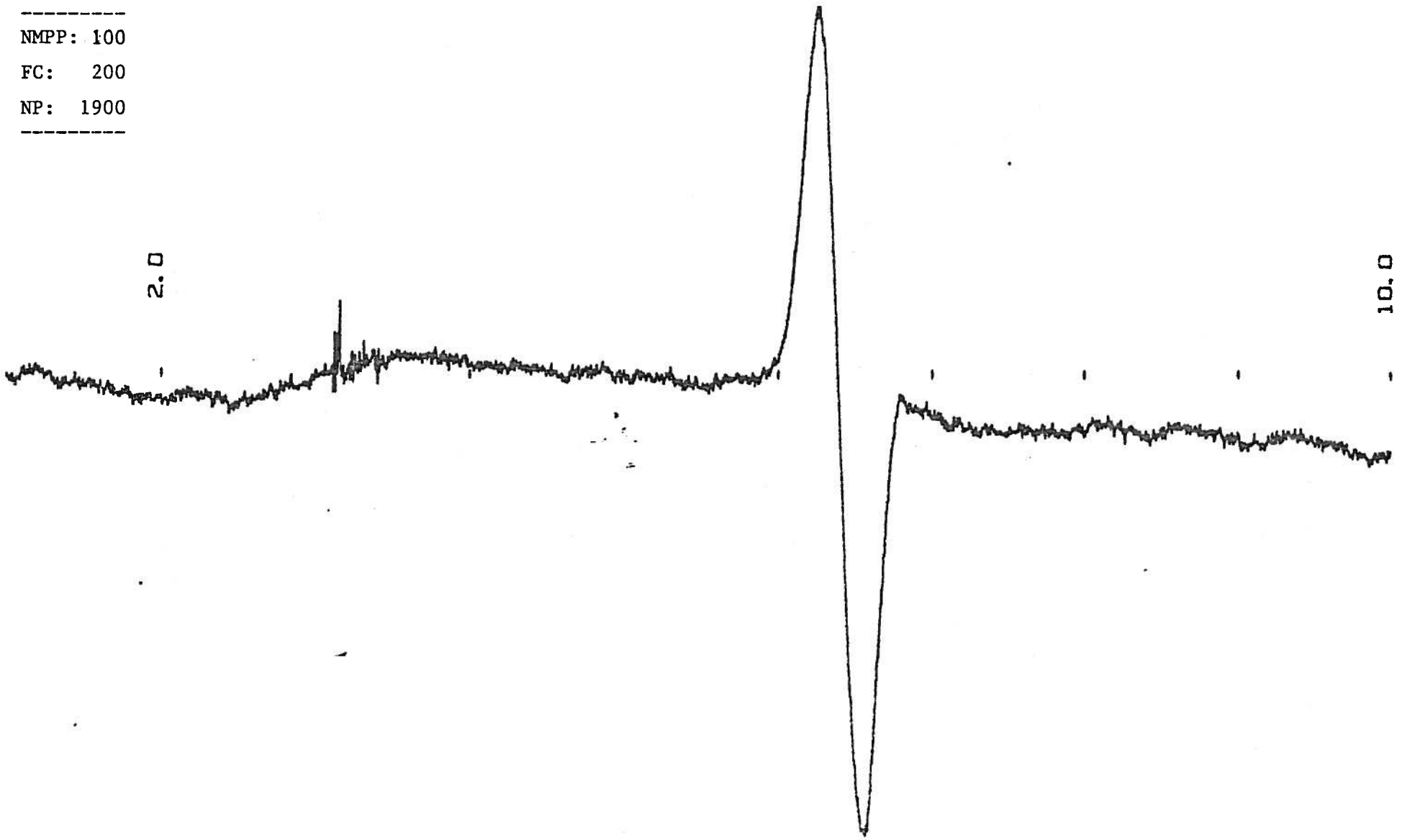
Embora esses resultados tenham sido considerados muito bons, é importante notar que, enquanto o tempo gasto na medida com 10 leituras por ponto foi de 1,5 minutos, as outras medidas gastaram, respectivamente, 16 e 80 minutos. Como se vê, o método é eficiente mas, em contrapartida, pode necessitar de um tempo de amostragem excessivamente longo, às vezes impossível de ser obtido na prática, devido a outros fatores como, por exemplo, a estabilidade da temperatura da amostra.

Embora até o momento não tenha sido testada, uma alternativa que se configura promissora para a redução do tempo de aquisição, é trabalhar com um número menor de pontos e submete-los a um filtro passa-baixas com frequência de corte próxima de zero, já que o sinal de interesse nesse caso é essencialmente constante.



1 div. = 1.0 gauss
test1 do smp1

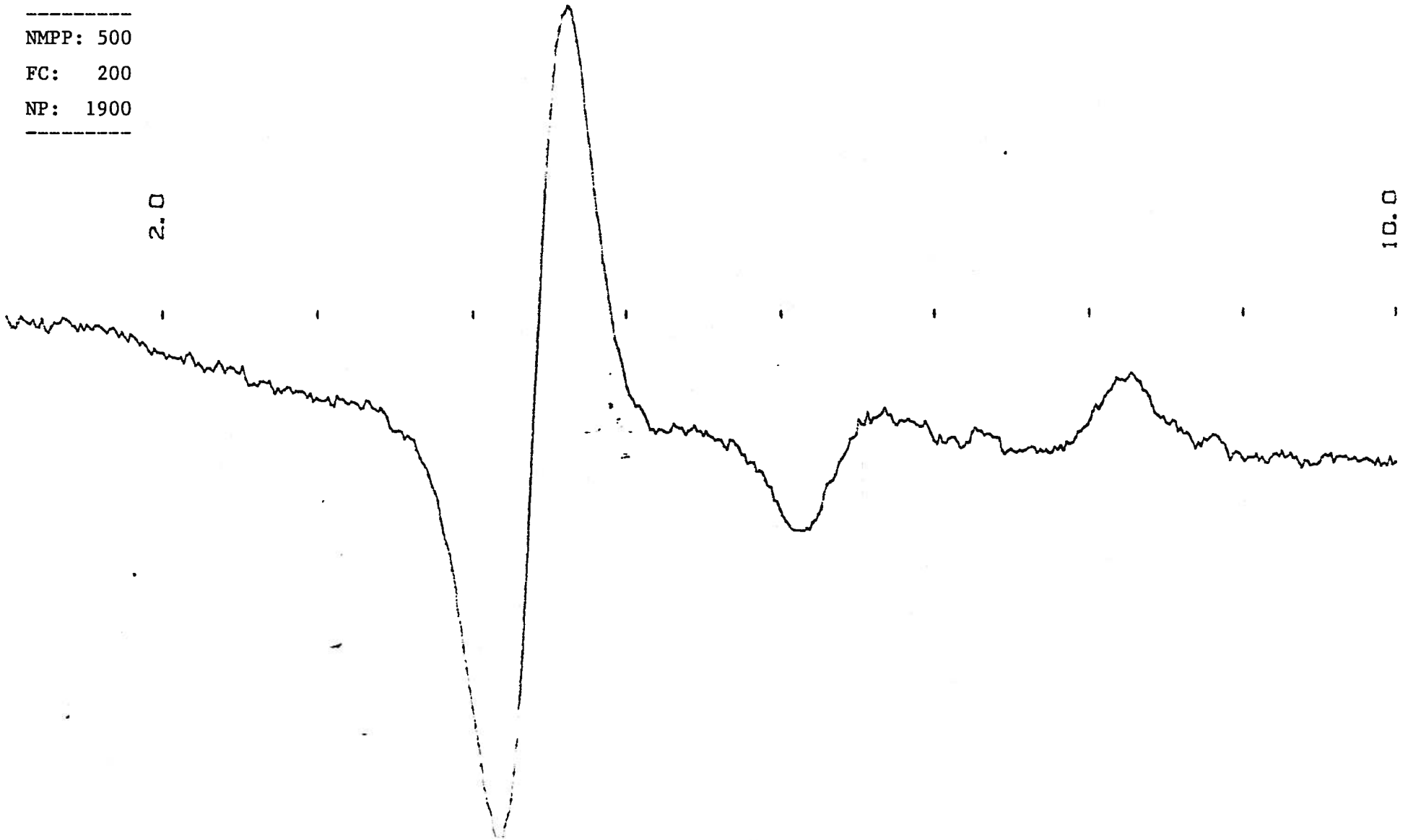
Fig. 3-19: RDA:Ti2+ - Registro com o SMOP
No. de medidas por ponto: 10



1 div. = 1.0 gauss
teste 2 do smp1 - 100 med/ponto - 08/04/88

Fig. 3-20: RDA:TI2+ - Registro com o SMOP
No. de medidas por ponto: 100

(1)



1 div. = 1.0 gauss

Fig. 3-21: RDA:TI2+ - Registro com o SMOP
No. de medidas por ponto: 500

Depois dessas experiências, o SMOP foi usado em um estudo detalhado do RDA:Tl^{2+} [13], onde foram feitas cerca cem medições, nas quais procurou-se estabelecer um compromisso entre a qualidade do sinal e o tempo de aquisição dos espectros.

Para tanto, a taxa de aquisição foi mantida em 200 cps e foram feitas 100 amostragens para cada ponto. Além disso, a confiabilidade do sistema, com relação ao tempo de estabilização do motor, foi testada o tempo todo, fazendo-se varreduras simétricas ("increase", "decrease") em todos os espectros.

Os resultados obtidos foram de muito boa qualidade e permitiram, inclusive, que se tornasse possivel o acompanhamento da linha correspondente ao centro A^+ (Fig. 3-22), o que havia sido considerado inviável com o uso do sistema analógico tradicional.

Alguns desses resultados estão apresentados nas Figs. 3-23, 3-24 e 3-25, onde se pode ver também, nas linhas continuas, o ajuste dos espectros a uma soma de três derivadas de gaussianas, realizado com o auxílio dos programas RESS1[23] e FIT2C[24].

O pré-processamento dos dados, bem como o seu transporte para o programa de ajuste (FIT2C), foi feito com o auxilio do programa MASPE através de fitas magnéticas de 1600 BPI.

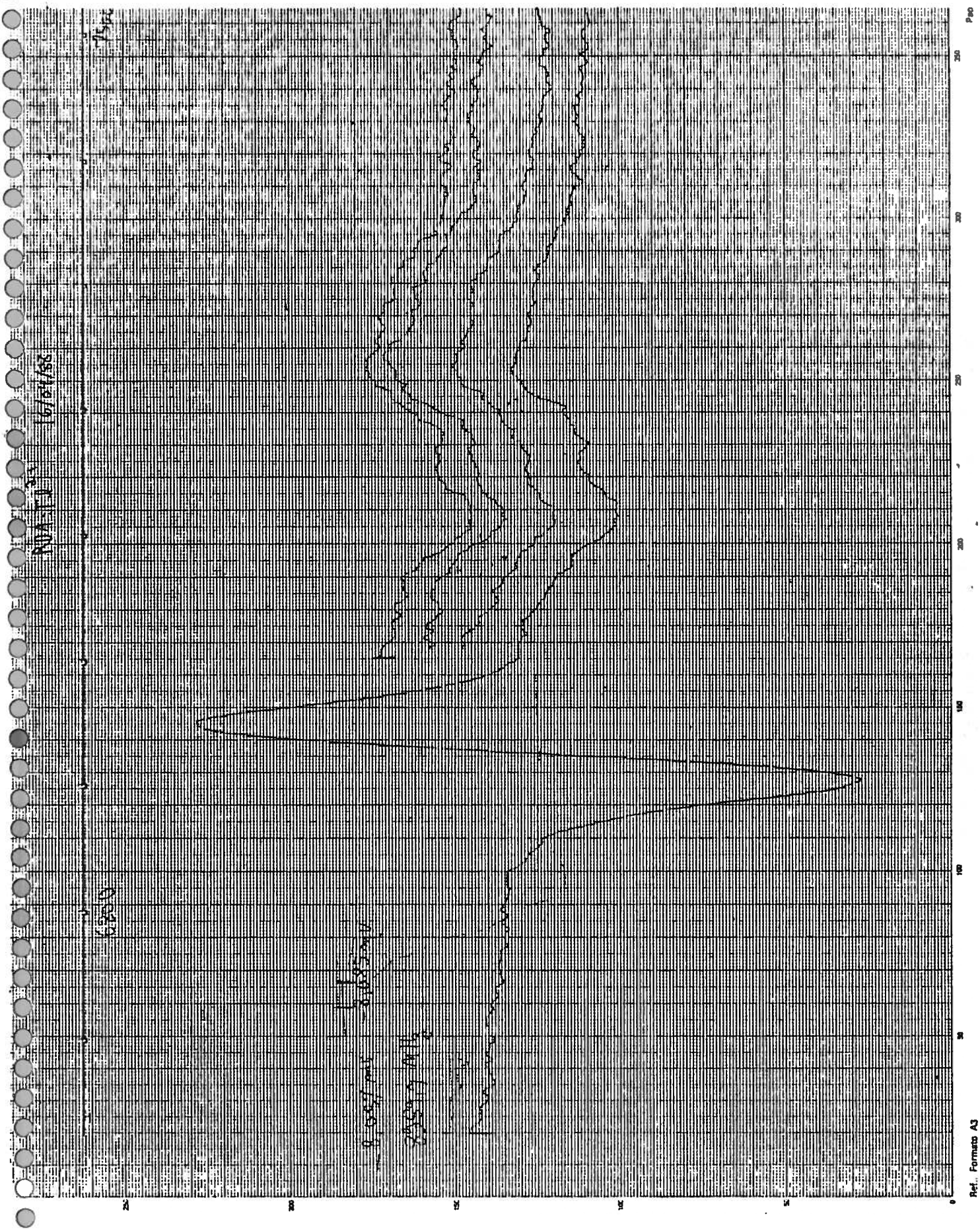


Fig. 3-22: Registros analógicos de espectros de RDA:Tl^{2+} obtidos ao longo dos estudos do centro A^{+} [13]

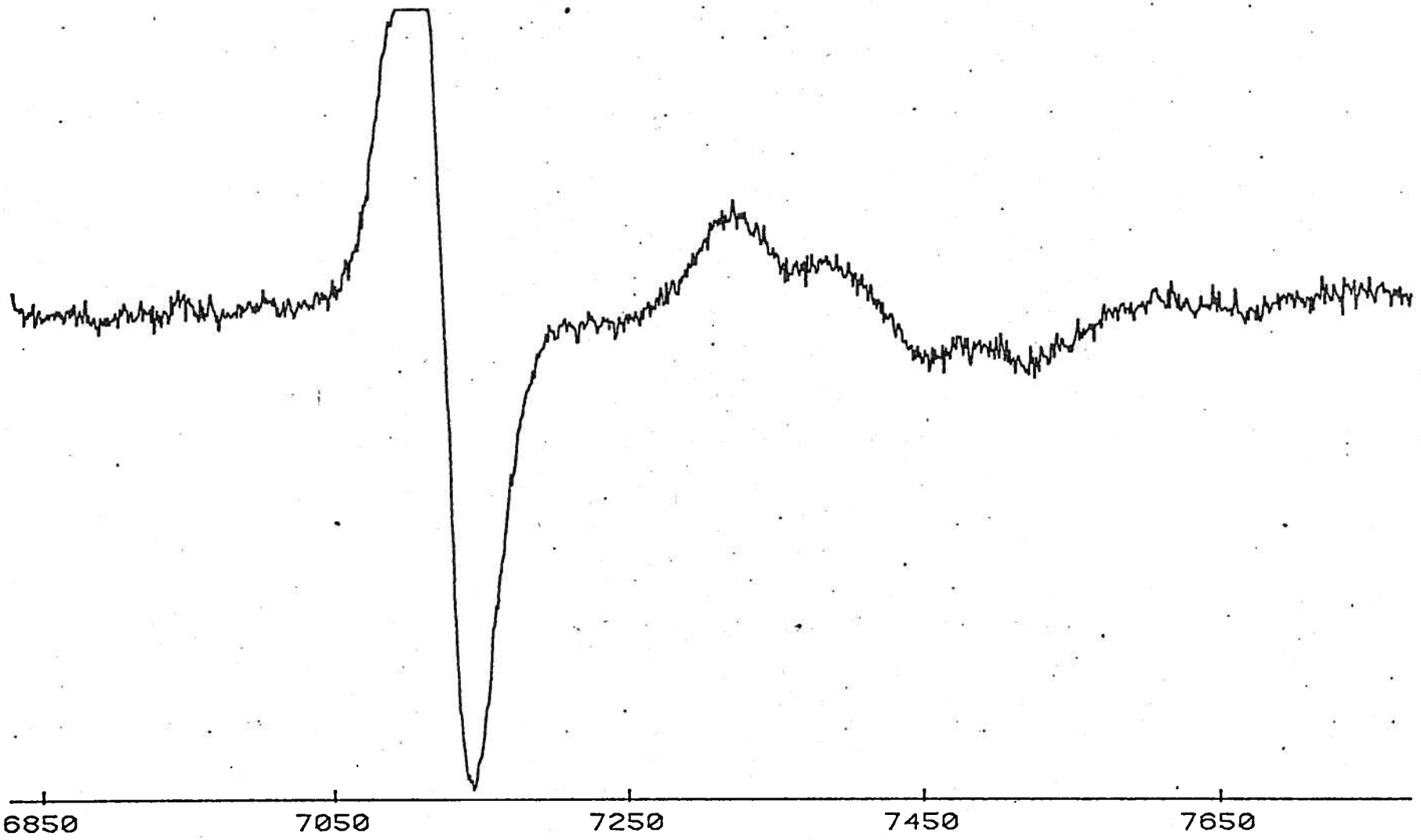
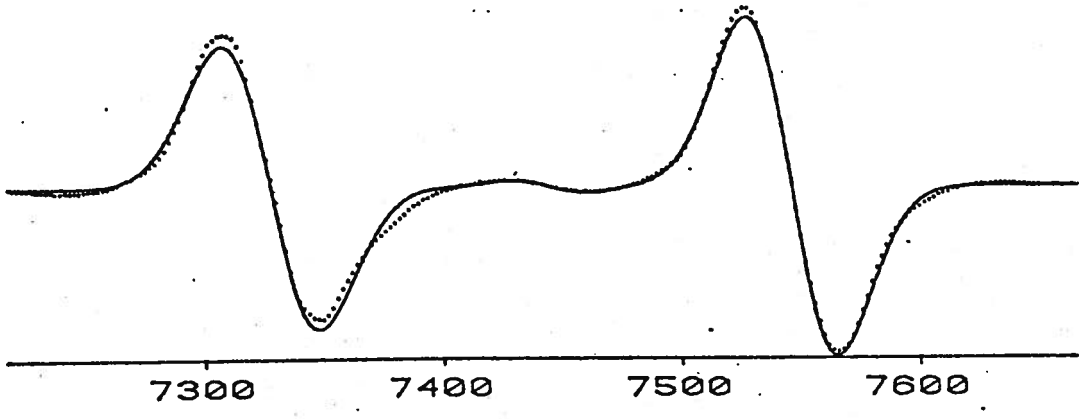
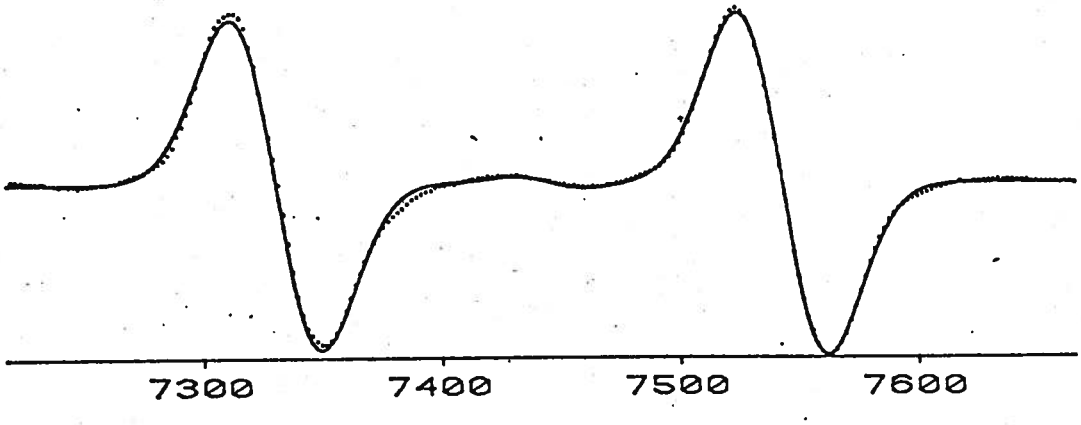


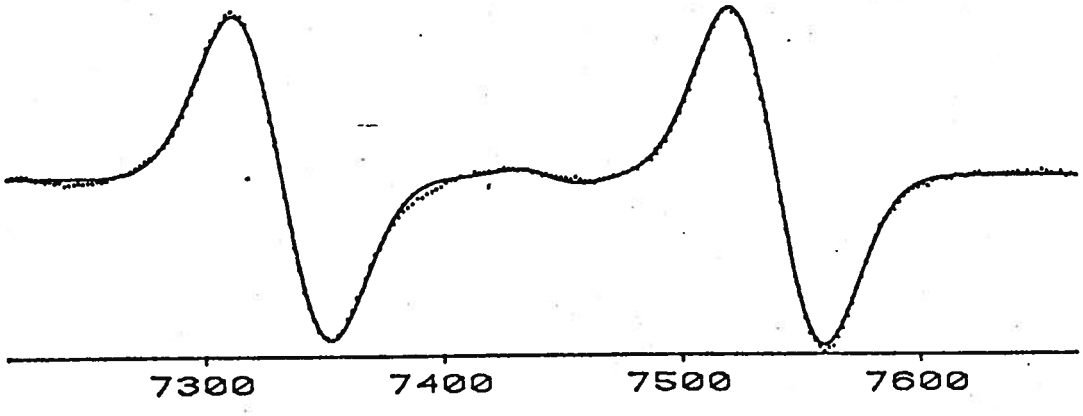
Figura 3-23 - Linhas de campo alto do espectro de RPE de RDA: Tl^{2+} (T = 103.4K, campo magnético no plano ac a 61° de c).



T = 32.6K



T = 62.3K



T = 74.7K

Figura 3-24 - Linhas de campo alto dos centros A''_+, A_+ e A'_+ (Campo magnético no plano ac a 61° de c)

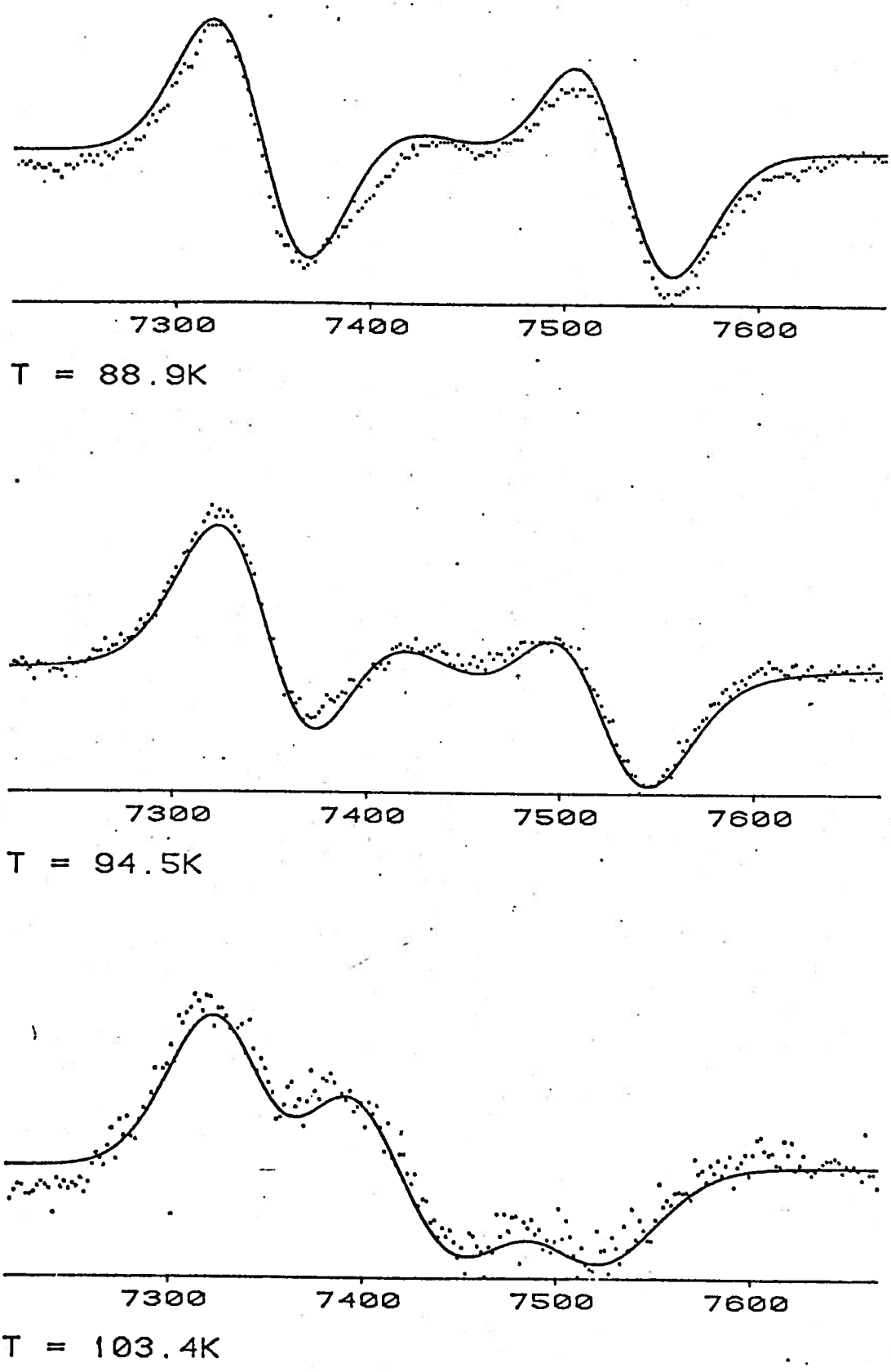


Figura 3-25 - Linhas de campo alto dos centros A''_+, A_+ e A'_+ (campo magnético no plano ac a 61° de c)

3.7 Resultados Adicionais

Ao longo dos estudos do RDA:Tl^{2+}, observou-se[13] a existência de uma estrutura dependente da temperatura, sobre uma das linhas de campo alto, quando o campo magnético era aplicado dentro de uma estreita faixa de orientações (20 a 22°), no plano ab (Fig. 3-26).

Durante a fase de análise dos resultados, aventou-se a hipótese de se fazer um modelo físico para um estudo mais detalhado dessas estruturas, no qual as derivadas dos espectros já armazenados em disco seriam usadas como dados, pois, a realização de uma nova série de medidas acarretaria um grande atraso nas atividades do laboratório.

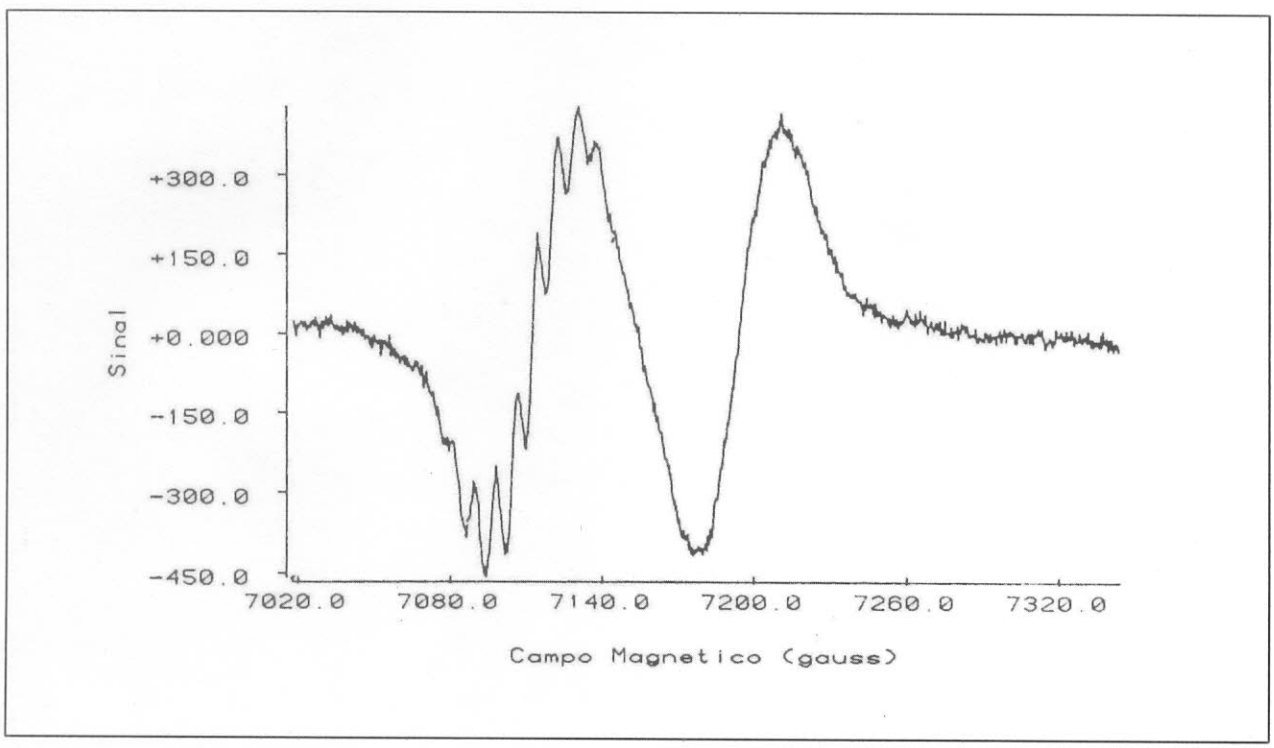


Fig. 3-26: Espectro de RDA c/ estrutura dependente da temperatura

Quando se tem um espectro com uma ótima relação sinal-ruído, o cálculo da derivada não chega a apresentar maiores problemas mas, quando se tem a presença de componentes de "alta frequência" misturadas ao sinal de interesse, o resultado da derivação numérica pode ser bastante desapontador, uma vez que estas componentes geram muito ruído quando derivadas. (Fig. 3-27)

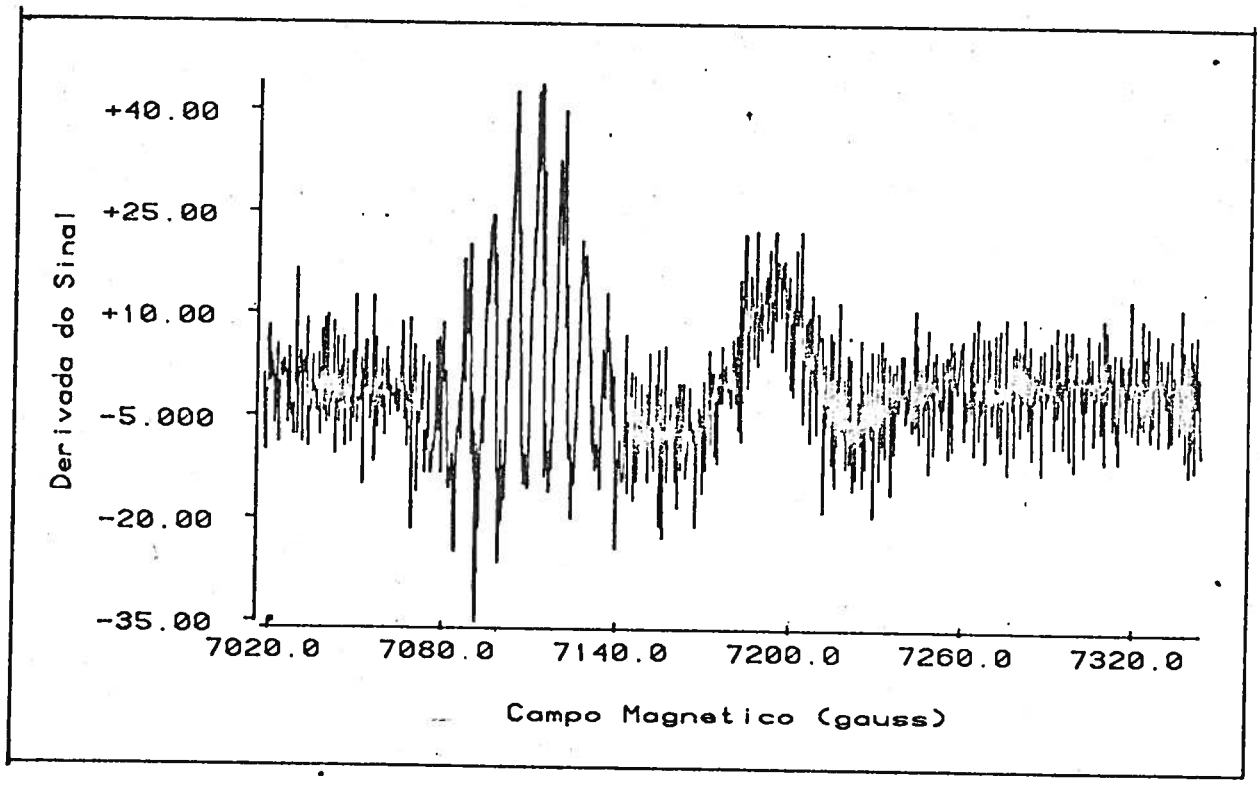


Fig. 3-27: Derivada do espectro de RDA com estrutura

Para contornar essa dificuldade, resolvemos submeter os resultados das derivadas a uma operação de alisamento (smoothing) com o

uso de um algoritmo que, em resumo, substitui cada ponto pela média de vizinhos adjacentes.

Esta operação, embora produza resultados, digamos aceitáveis, apresenta o grande inconveniente de ter uma resposta de frequência um tanto irregular, conforme vimos no capitulo 1, Fig. 1-4.

Por causa desse inconveniente, resolvemos então experimentar a técnica de redução de ruído, baseada em filtros digitais. Para essa experiência, usamos o filtro passa-baixas tipo NER (Nearly Equal Ripple)[5], apresentado no capítulo 1 e que foi implantado como sendo uma das operações disponíveis no programa MASPE (comando FD). O usuário, ao utilizá-lo, fornece a frequência de corte, os limites da ondulação no ganho e a tolerância na frequência de corte.

Para que se possa fazer uma avaliação da frequência de corte, introduzimos, no programa MASPE, uma rotina que calcula a transformada de Fourier (FFT) (comando FT) e exibe na tela o espectro de potência do sinal. Esta rotina usa o algoritmo de Tukey[25] e devolve ao programa principal a transformada na ordem correta ("unscrambled").

Para facilitar a introdução dos dados no filtro digital, a escala de frequências é devolvida em termos da frequência de Nyquist e varia, portanto, de 0 a 1.

Depois da implantação dessas funções, a filtragem das derivadas dos espectros foi feita da seguinte maneira:

- Tomamos inicialmente o espectro e calculamos a derivada numérica ponto a ponto (Figs. 3-26 e 3-27).

- Isolamos no espectro-derivada uma região de interesse, completamos com zeros para que o número de pontos fosse uma potência de 2 e calculamos a FFT (Fig. 3-28). No gráfico da FFT, podemos observar dois picos que, pelas suas frequências, estão relacionados com a linha principal e com a estrutura existente na primeira linha. Para preservar a informação destes picos, escolhemos uma frequência de corte um pouco abaixo de 0,2 e aplicamos o filtro pelo método da convolução no tempo, tal como descrito no capítulo 1.

- A aplicação do filtro resulta em um espectro-derivada totalmente isento de ruído de alta frequência (Fig.3-29). Os ruídos de baixa frequência da linha de base têm larguras muito próximas das larguras das oscilações da derivada e, por isso, não podem ser atenuados.

A FFT do espectro filtrado mostra, de maneira muito clara, o efeito do filtro na atenuação das frequências relativas acima de 0,2 (Fig. 3-29).

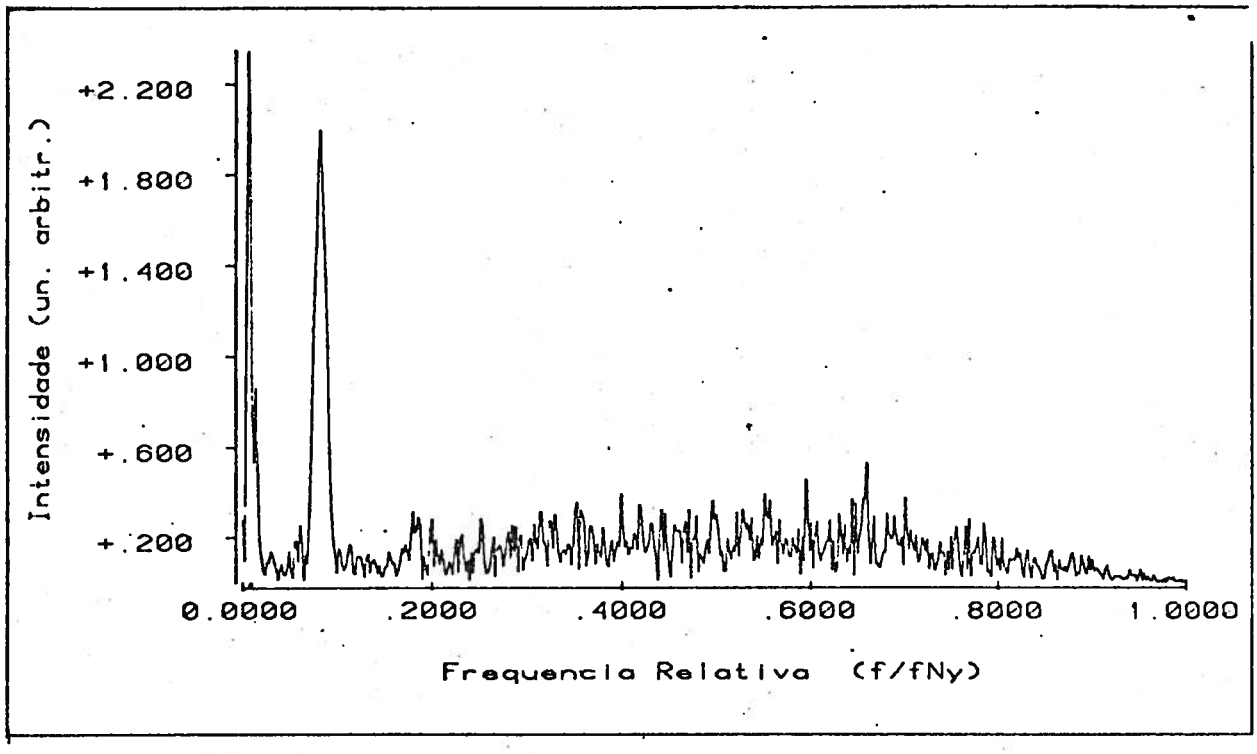


Fig. 3-28: Espectro de potências da derivada do espectro de RDA com estrutura. A frequência de corte para a aplicação do filtro digital foi escolhida um pouco abaixo do ponto 0,2.

De um modo geral, quando se tem de trabalhar com as derivadas de espectros ruidosos, o procedimento correto é filtrar os dados e depois derivá-los. No nosso caso, a ordem inversa foi usada intencionalmente, com o objetivo de realçar a eficiência do filtro.

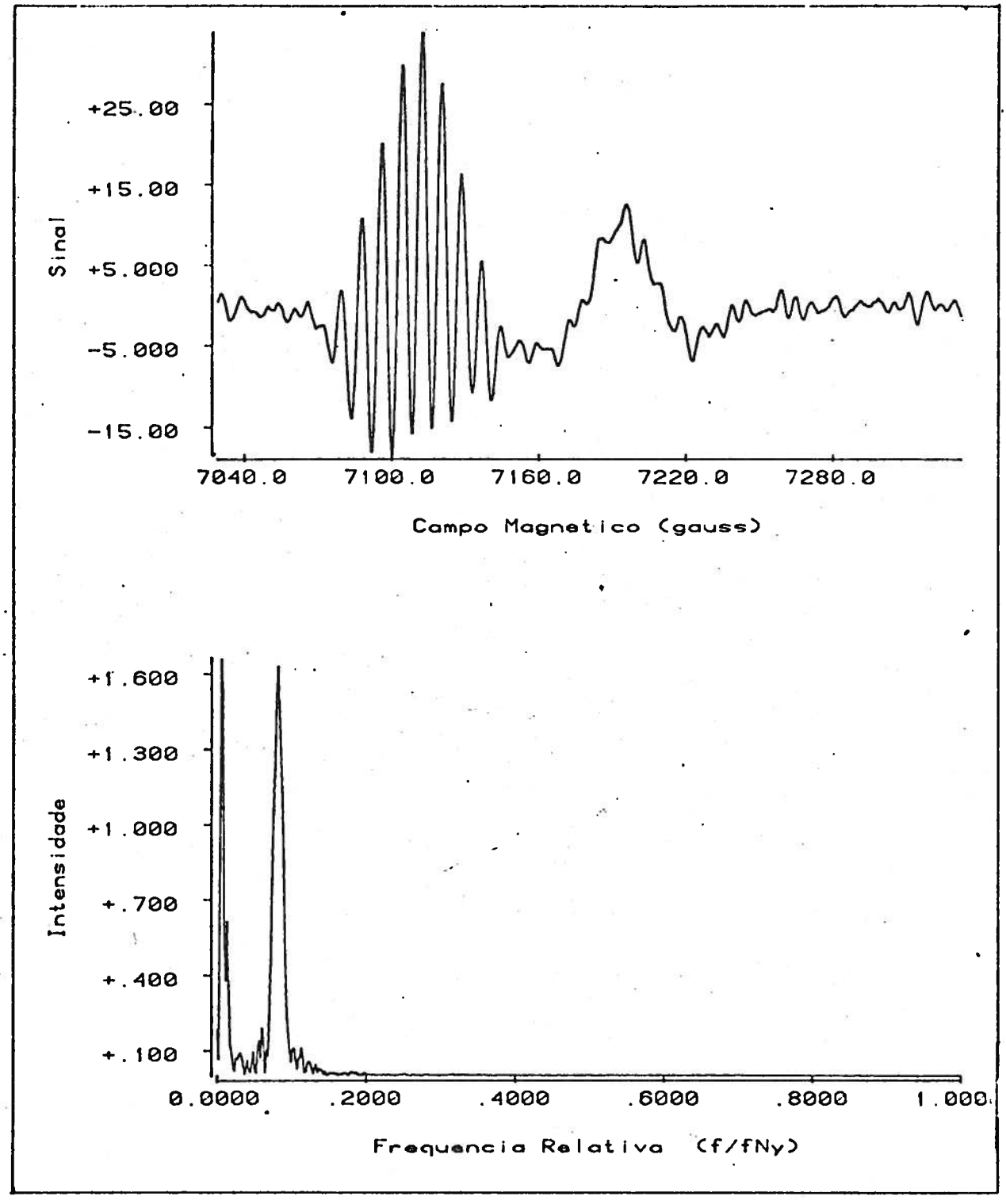


Fig. 3-29: Resultado da filtragem do espectro de RDA. Acima, espectro derivada isento de ruídos e, abaixo, espectro de potência do espectro derivada, mostrando a atenuação das frequências acima de 0,2 (f/f_{Ny}).

Encorajados pelos resultados obtidos com as derivadas, aplicamos com sucesso o filtro NER a vários outros espectros e concluímos que, até mesmo em situações nas quais o sinal não está totalmente mascarado pelo ruído, o filtro digital pode dar um auxílio importante ao pesquisador (Fig. 3-30).

Finalmente, mas ainda com relação aos filtros, gostaríamos de deixar bem claro que, ao usá-los, deve-se ter uma atitude bastante crítica quanto aos resultados para evitar que, ao atenuar as "altas frequências", sejam perdidas informações relevantes do problema físico em questão.

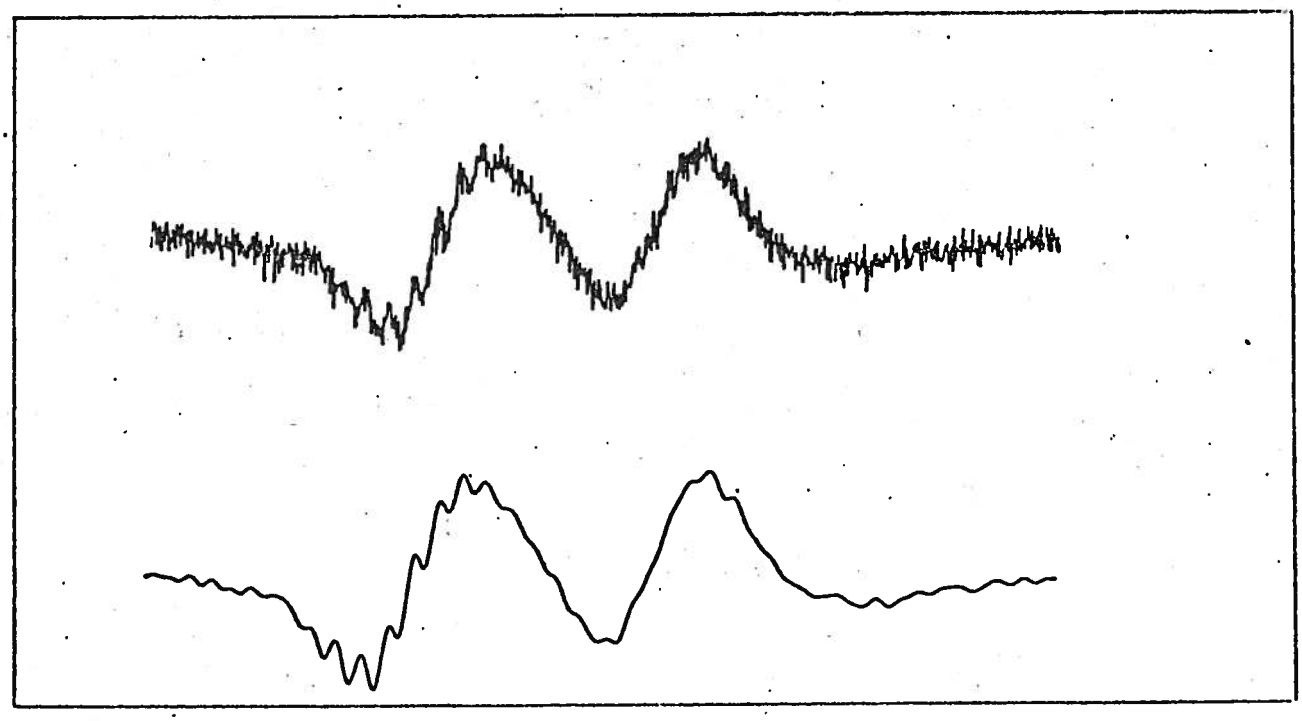


Fig. 3-30: Redução de ruídos obtida em um espectro de RDA através da aplicação de um filtro digital "passa-baixas"

Capítulo 4

Conclusões

Neste trabalho, desenvolvemos dois sistemas para aquisição de dados em experiências de EPR em tempo real, segundo a metodólogia da "arquitetura aberta", tendo como base computacional um sistema HP 1000. Até o momento, mais de 300 espectros já foram registrados, sem que tenhamos tido qualquer problema relacionado com a concepção desses sistemas, o que nos permite concluir que os modelos usados, tanto no Sistema Melhorado de Aquisição Passiva quanto no Sistema do Motor de Passo, foram suficientemente bons e estáveis para permitir um fluxo contínuo dos dados das experiências realizadas no laboratório.

O sistema de aquisição passiva, SMAP, revelou-se extremamente cômodo para o usuário, pois, dada sua total coexistência com o sistema tradicional de registro analógico, a passagem de um sistema para o outro é um processo muito simples e consome muito pouco tempo de aprendizado.

Quanto ao problema da redução do ruído, pudemos verificar que a técnica de "averaging", mesmo sendo um processo lento, funcionou muito bem no sistema de aquisição passiva, graças à amostragem coerente de todas varreduras e ao registro fidedigno das intensidades do campo magnético, o que foi assegurado pelo uso do compa-

rador no circuito da rampa e do ADC com uma base de tempo independente da CPU no registro dos sinais.

O valor da dispersão na intensidade do campo nas extremidades do espectro, como vimos no capitulo 2, não chega a ser um obstáculo para as experiências realizadas atualmente no laboratório. No futuro, se necessário, o desempenho do sistema poderá ser melhorado, neste particular, através do uso de um ADC de maior precisão e da instalação de uma fonte de maior estabilidade para geração da tensão da rampa.

No sistema do motor de passo, o uso do conjunto motor/helipot no acionamento da varredura permitiu-nos mostrar que um equipamento antigo (no caso a fonte Varian-Fieldial Mark II) pode ser atualizado - dentro de limitações, é claro - através de um processo simples, econômico e muito confiável. Graças a essa atualização, pudemos implantar a técnica de "averaging" ponto a ponto que, em muitos casos, pode substituir com vantagens a técnica tradicional de "averaging" varredura a varredura.

Com esse sistema poderemos também facilitar a implantação de mecanismos para a aquisição de dados em experiências de ENDOR, pois nas mesmas é necessário que o campo seja mantido em valores estáveis enquanto se processa a varredura do sinal de rádio-frequência.

Finalmente, gostariamos de apontar dois aspectos operacionais que, mesmo não aparecendo de forma explícita nos resultados obtidos, merecem ser destacados dada a sua relevância no contexto deste trabalho.

O primeiro foi o uso da metodologia da arquitetura aberta que, amparada pelos grandes recursos do FORTRAN 77 e do RTE6-VM, permitiu-nos o desenvolvimento de extensos programas de uma maneira clara e ordenada, o que facilitará, em muito, o redirecionamento dos sistemas para outras técnicas que venham a ser instaladas no laboratório.

O segundo foi a grande interação dos pesquisadores do laboratório com os programas de aquisição ao longo de toda a implantação dos mesmos. Esta interação mostrou-nos que a utilização dos "comandos parametrizados" na operação dos programas foi de rápida assimilação pelos usuários, atingindo plenamente nosso objetivo de criar um sistema "flexível e de uso descomplicado".

Esta assimilação pode ser claramente evidenciada pelo uso do sistema e/ou seus programas em 5 teses desenvolvidas no Laboratório de Ressonância[9,13,14,26,27], uma tese no Laboratório de Semicondutores[27] (redirecionamento do sistema) e em várias publicações recentes[11,14,15,29,30].

Referências Bibliográficas

1 . E. STERNIN, Rev.Sci.Instru,56(11), (1985).

2 . H. P. HSU, "Análise de Fourier", Livros Técnicos e Científicos Editora, Rio de Janeiro.

3 . P. TROUSSON, Rev.Sci.Instrum.55(12), (1984).

4 . P. A. LYNN, "An Introduction to The Analysis and Processing of Signals", Macmillan Press Ltd., London, 1983.

5 . J.F. KAISER and W.A. REED, Rev.Sci.Instrum.,Vol.48,No.11, (1977).

6 . RTE-6/VM Programmer's Reference Manual, Hewllet-Packard Company, Palo Alto, 1983.

7 . M.P. KLEIN and G.W. BARTON, JR, Rev.Sci.Instrum.,Vol.34, No.7, (1963).

8 . F.G. HERRING, J. MAYO and P.S. PHILLIPS, Journal of Magnetic Ressonance, No.34, (1979)

9 . C. J. FRANCO, Tese de Doutoramento, UFMG (em andamento).

10 . HP 59313A ADC Converter Reference Manual, Hewleet-Packard Company, Palo Alto, 1983.

11 . E.S. ALVES, J.F. SAMPAIO, R. GAZZINELLI, A.S. CHAVES and G.M. RIBEIRO, Journal of the Physical Society of Japan, Vol.36,No.12, (1987).

12 . J.F. SAMPAIO, G.M. RIBEIRO, R.A. NOGUEIRA and A.S. CHAVES, Sol.State Comm.,Vol.57,No.12, (1986).

13 . A.J.M. NEVES, Tese de Mestrado, UFMG, 1989.

14 . J.F. SAMPAIO, G.M. RIBEIRO, A.S. CHAVES and R. GAZZINELLI, J.Phys.C: Solid State Phys., No.19, (1986)

15 . M.S.S. DANTAS, A.S. CHAVES, R. GAZZINELLI, A.G. OLIVEIRA, M.A. PIMENTA and G.M. RIBEIRO, Journal of the Physical Society of Japan, Vol.53,No.7, (1984).

16 . J.D.S. DANIELSON and B.H. ROBINSON, Rev.Sci.Instrum., No. 56, (1985).

17 . R.W. QUINE, G.R. EATON and S. EATON, Journal of Magnetic Resonance, No.66, (1986).

18 . R.M. GENET, "Microcomputer in Astronomy", Fairborn Observatory, Fairborn, 1983.

19 . D.M. AUSLANDER, P. SAGUES, "Microprocessors for Measurement and Control", Osborne, Berkeley, 1981

20 . MOTOR DE PASSO - Características e Aplicações, Syncro Eletromecânica Ltda, São Paulo

21 . VARIAN Analytical Instrument Division - Publication 87165 000 - D 566, Palo Alto

22 . W.B. DAVENPORT, JR. E W.L. ROOT, "An Introduction to the Theory of Random Signals and Noise", McGraw-Hill, London, 1958.

23 . J.F. SAMPAIO, Tese de Doutoramento, UFMG, 1986.

24 . E.S. ALVES, Tese de Mestrado, UFMG, 1985.

25 . J.R. JOHNSON, "Introduction to Digital Signal Processing", Prentice-Hall, Englewood Cliffs, (1989).

26 . M.S.S. DANTAS, Tese de Doutoramento, UFMG 1988.

27 . G.J. PERPETUO, Tese de Mestrado, UFMG (em andamento)

28 . R. RODRIGUES, Tese de Mestrado, UFMG, (1990).

29 . A.J.M. NEVES, G.M. Ribeiro, A.S. CHAVES, R. GAZZINELLI J. Phys C: Condens Mater, 2 (1990)

30 . R. RODIRGUES, P.S.S. GUIMARÃES, J.F. SAMPAIO, R.A. NOGUEIRA A.T. OLIVEIRA, I.F.L. DIAS, J.C. BEZERRA, A.G. OLIVEIRA, L. M. SCOLFARO, "Influência da Concentração de Dopantes na Migração de Si em GaAs Com Dopagem Delta", XIII Encontro Nacional de Física da Matéria Condensada, Caxambu (1990)

Apêndice 1

O Programa MASPE

PROGRAMA MASPE

DESCRIÇÃO GERAL DO PROGRAMA

&

DESCRIÇÃO DOS COMANDOS

MASPE: Um programa para manipulação de espectros

1. Introdução

O programa MASPE é um dos programas do sistema de aquisição de dados do Laboratório de Ressonância Magnética e tem como função básica preparar para o ajuste os dados que são registrados em tempo real durante as experiências. Além disto, o MASPE permite também que se faça uma série de operações com os espectros tais como adição de constantes, integração, exibição no vídeo, saida no plotter, cálculo de médias, etc.

A motivação para a criação de um programa para manipulação de espectros está ligada ao fato de que quando se dispõe de mecanismos para aquisição de dados em tempo real consegue-se um grande volume de dados em um curto espaço de tempo. Estes dados entretanto nem sempre estão na forma ideal para que sejam submetidos a um programa de ajuste. Torna-se então necessário dispor de uma ferramenta que facilite uma formatação adequada dos dados disponíveis para que deles se possa retirar as informações desejadas.

O MASPE é um programa interativo, escrito em HP-FORTRAN 77, com os recursos de programação estruturada oferecidos pelo compilador HP. E' executado em um computador HP 1000 sob o sistema operacional RTE-6VM em uma partição de memória de no mínimo 256 kbytes para operações com memória real ou de 128 kbytes para operações com memória virtual.

Por uma questão de eficiência, o MASPE utiliza vários recursos de software, firmware e hardware do computador HP 1000 contidos nos subsistemas FFP (Fast Fortran Processor) e VIS (Vector Instruction Set). Além destes recursos, são usadas também várias extensões da liguagem FORTRAN tais como IF-THEN-ELSE, DO-WHILE, ENCODE-DECODE, BLOCK-DO, etc. O ganho de eficiência proporcionado por estes recursos faz entretanto com que este programa só possa ser executado em máquinas da família HP1000. A sua exportação para outras máquinas constitui tarefa praticamente impossível sem que sejam feitas profundas modificações no código fonte.

Do ponto de vista de hardware, o MASPE necessita para seu funcionamento de um sistema HP 1000 típico que possua além do disco, impressora e fita magnética, um terminal gráfico a cores do tipo HP 2627A, e um plotter do tipo Tektronix 4663. Outros equipamentos gráficos poderão ser utilizados desde que sejam reescritas as rotinas específicas do sistema.

Por ser um programa essencialmente interativo e que faz pouco processamento "pesado", as operações do MASPE se processam na maioria dos casos em poucos milisegundos. Por este motivo, o tempo de CPU gasto em uma sessão no sistema HP 1000 é pequeno. Este fato permite que o programa seja executado com uma prioridade maior que a maioria dos outros programas, garantindo respostas imediatas do sistema mesmo em situações de uso intenso da CPU.

2. Estrutura do programa

O programa MASPE está escrito nos moldes de um pequeno sistema operacional. A interação com o usuário se faz através de comandos parametrizados que levam à execução de uma série de tarefas.

Os comandos são códigos mnemônicos de duas letras seguidos ou não de até sete parâmetros alfa-numéricos. As duas letras indicam a tarefa a ser executada e os parâmetros fornecem informações adicionais necessárias à execução da tarefa. Por exemplo, para leitura sequêncial de dados em fita magnética o comando é: LS, n1, n2. Neste comando, LS significa Leitura Sequencial, n1 é o número de arquivos que se deseja ler e n2 é o número do primeiro arquivo a ser lido.

A disponibilidade do programa para execução de um comando é indicada ao usuário através da exibição da mensagem CMD ?. Sempre que esta mensagem for exibida, o programa estará apto a executar qualquer comando. Uma vez recebido um comando, o programa verifica a validade do mesmo através de um "processador de comandos" e, no caso de comandos válidos inicia imediatamente a execução da tarefa. Terminada a execução o programa volta a exibir a mensagem CMD ?. Para os comandos inexistentes (ou erros e omissões do operador) o programa exibe uma mensagem de advertencia e fica aguardando um novo comando.

Em vários comandos é utilizado o conceito de "default" ou seja, a ausência de um dado parâmetro faz com que o programa use um valor pré-determinado, de conhecimento do usuário.

A execução dos comandos pelo MASPE é feita através da transferência do controle a determinadas partes do programa. Estas "partes", chamadas blocos dos comandos são acessadas através da execução de um GOTO calculado no processador de comandos (Fig. 1). Uma vez dentro de um bloco de comando, o programa executa a tarefa (ou emite uma mensagem de erro, se for o caso) e retorna ao processador de comandos através de uma instrução GOTO 5.

Cada bloco de comando é evidentemente composto por um conjunto de instruções do FORTRAN 77 mas, no contexto deste programa, os blocos são separados em quatro tipos diferentes:

Tipo 1: O bloco é composto por um conjunto de instruções que executam por si próprias a tarefa contida no comando.

Tipo 2: O bloco é composto por um conjunto de instruções que passam o contrôle a uma sub rotina que então executa a tarefa.

Tipo 3: O bloco é composto por um conjunto de instruções que passam o controle a um outro programa, chamado "programa filho", que fica então responsável pela execução da tarefa.

Tipo 4: O bloco é composto por um conjunto de instruções que carregam na memória do sistema um segmento ("overlay") do programa principal e passam o controle a este segmento onde então a tarefa é executada.

Para facilitar futuras modificações no programa, cada bloco foi escrito de modo a ter um único ponto de entrada, acessado pelo processador de comandos, e a convergir todas as saõdas para o rótulo 5, que é a entrada do processador de comandos. Porisso, a a instrução GOTO é pouco usada em outros pontos do programa.

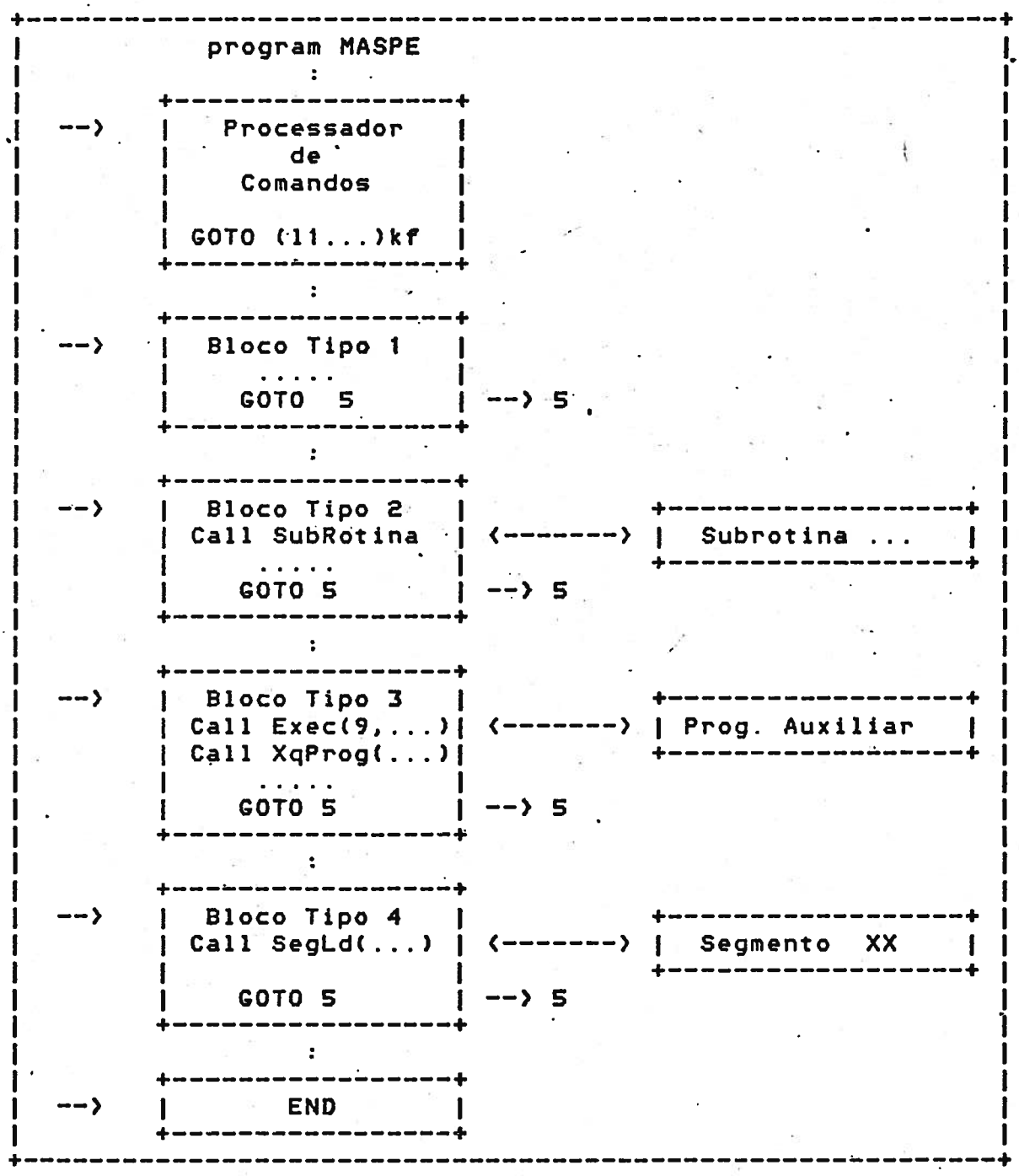


Fig. 1: Diagrama básico do MASPE

Os comandos do MASPE são todos construídos de acordo com o formato

XX [,p1] [,p2] ... [,p7]

onde XX são os dois caracteres que compoem o mnemônico da tarefa e p1,...,p7 são os parâmetros que completam o comando. Conforme será discutido posteriormente, a necessidade dos parâmetros em um dado comando é função do tipo de tarefa a ser executada.

De acordo com este formato, uma sequência de comandos do MASPE pode então ter, por exemplo, o seguinte aspecto:

RW	rebobinar (RW) a fita magnética
PF,6	fazer uma pesquisa na fita (PF) e listar na máquina 6 (impressora) o nome de todos os espectros existentes
LS,27,10	fazer uma leitura sequêncial (LS) de 27 espectros a partir do arquivo de número 10.
MO,ET,1,1,NA	mostrar (MO) o espectro de trabalho (ET) no quadrante 1 da tela grafica, usando a cor número 1 (vermelho) e não apagar (NA) o que porventura ja esteja na tela

Uma vez transmitido ao MASPE, de acordo com o formato acima descrito, o comando é verificado no que diz respeito a existência de um bloco adequado à execução da tarefa solicitada. Os parâmetros, caso presentes, são também testados em função da tarefa a ser executada.

A verificação da validade de um comando é feita em duas etapas. Na primeira, o comando é analisado com uma chamada à rotina PARSE. Esta rotina, que faz parte da biblioteca do sistema RTE-6, organiza o comando e seus eventuais parâmetros na matriz IBUF. Os dois caracteres que definem a tarefa são colocados em IBUF(2) e os parâmetros, se transmitidos, são colocados em outras posições bem determinadas da mesma matriz.

Em seguida, com auxílio de uma instrução IF, o processador de comandos verifica se o dado mnemônico está definido em alguma posição da matriz de comandos MCMD. Se negativo, exibe uma mensagem de erro e aguarda outro comando; se afirmativo, registra na variável KF esta posição e transfere a execução para o bloco de comando adequado através de um GOTO calculado.

A segunda etapa da verificação, se necessária, é feita dentro do bloco de comando. Nesta etapa, situações distintas podem ocorrer:

-se a tarefa não necessita de parâmetros, ela é imediatamente executada e o controle volta para o processador de comandos.

-se a tarefa necessita de parâmetros e os mesmos foram fornecidos de maneira inadequada o programa exibe uma mensagem de erro e devolve o controle para o processador de comandos.

-se a tarefa necessita de parâmetros e os mesmos não foram fornecidos tem-se então duas alternativas, que dependem do comando. Na primeira, o programa usa valores previamente programados para os parâmetros ausentes (esté é o conceito de "default", já mencionado); na segunda, o programa exibe uma mensagem solicitando o fornecimento dos parâmetros. No caso do uso dos valores de "default", a tarefa é imediatamente executada e o controle volta ao processador de comandos. No outro caso, os parâmetros fornecidos pelo usuário são verificados quanto à sua validade, a tarefa é executada (ou uma mensagem de erro é exibida se algum dos parâmetros for inválido) e o controle volta ao processador de comandos.

Além desta verificação que é feita sobre os comandos, o programa mantém uma série de variáveis de controle que podem ser consultadas pelos blocos de comando para detecção de situações anômalas durante a operação. Estas variáveis informam, por exemplo, se a unidade de fita magnética está operacional, qual o tipo de terminal de vídeo que está sendo utilizado, qual o número de espectros existentes na memória, etc.
A inclusão de novas tarefas no programa é um processo bastante simples dada a estrutura de blocos adotada. Basicamente o que se tem a fazer é acrescentar um novo mnemônico à matriz de comandos (MCMD), criar o bloco de comando adequado à nova tarefa e ajustar os parametros de algumas instruções do processador de comandos.

3. Descrição Geral do Programa

O MASPE, como já foi dito, é um programa dedicado à manipulação de espectros. O seu funcionamento está baseado nos conceitos de Pilha de Espectros, Espectro de Trabalho, Sub Espectro e Espectro Resultado.

A Pilha de Espectros é um conjunto de espectros que são lidos da fita magnética ou do disco e armazenados sequencialmente na memória.

O Espectro de Trabalho é uma cópia de um dos espectros da pilha escolhido pelo operador através de comando específico.

O Sub Espectro é uma cópia de uma determinada região do Espectro de Trabalho. Esta região pode abranger todo o Espectro de Trabalho.

O Espectro Resultado é o espectro que se obtém quando se efetua operações matemáticas sobre o Espectro de Trabalho ou sobre o Sub Espectro

No programa, estes conceitos se materializam pelo uso de quatro matrizes através das quais se processa a maior parte do fluxo de dados necessário à implementação das várias tarefas disponíveis. Estas matrizes são as seguintes:

1. ISPEC — Esta é a matriz "pilha de espectros". Ela contém os espectros que são lidos da fita magnética, através dos comandos LS (leitura sequencial) ou LA (leitura aleatória) ou do disco, através do comando LD.

2. YT — Esta é a matriz "espectro de trabalho". Ela contém um dos espectros da pilha, escolhido pelo operador através do comando DT (define espectro de trabalho)

3. YS — Esta é a matriz "sub-espectro". Ela contém uma parte dos dados da matriz YT e normalmente representa uma dada região de interesse do espectro de trabalho. O seu preenchimento se faz com o comando DS (define sub-espectro)

4. YR — Esta é a matriz "espectro resultado". Ela contém os dados resultantes de operações matemáticas efetuadas sobre as matrizes YT e YS

Além destas, existem ainda duas matrizes auxiliares, LINHA1 e LINHA2, que contem os nomes dos espectros e os parâmetros de aquisição, respectivamente. Estes parâmetros de aquisição, conforme descrito nos programas de coleta de dados, trazem as informações relativas ao campo magnético, frequência de amostragem, tamanho da varredura, etc.

Conforme mencionado anteriormente, este programa funciona com o método de comandos e tarefas. Dentro do contexto deste programa, um comando é uma sentença alfa-numérica que lhe é transmitida com o intuito de iniciar a execução de uma tarefa. A tarefa, por sua vez, é um conjunto de procedimentos computacionais que uma vez executados levam a um resultado definido.

Quando em operação, o programa passa por tres fases distintas, que são as seguintes:

Fase 0	Inicialização do programa
Fase 1	Entrada e análise de comandos
Fase 2	Execução de uma tarefa

A Fase 0 é executada apenas uma vez em cada sessão. Nesta fase são feitas as inicializações de algumas variáveis do programa, as condições da fita magnética são testadas, é verificado o tipo do terminal que está sendo usado pelo operador, etc.

A Fase 1 é a fase de entrada dos comandos. Ela é executada toda vez que o operador responde ao sinal CMD ?. Nesta fase é que se faz a verificação da existência do comando e que se transfere o controle para o bloco de comando adequado.

A leitura da sentença de comando é feita com o auxílio da rotina REIO (re-entrant input-output). Esta rotina, que pertence à biblioteca do sistema, permite a leitura de sentenças em ASCII diretamente para um dado vetor na memória e armazena no registro B da CPU o número de caracteres que foram efetivamente transferidos na operação. Este número de caracteres é usado depois na rotina CLCUC, que converte letras minúsculas para maiúsculas e na rotina PARSE, que separa e classifica os parâmetros recebidos.

A Fase 2 é a fase da execução das tarefas. Toda vez que um comando válido é recebido pelo processador de comandos, o controle é transferido para um dos blocos de comando. Este bloco verifica a validade dos parâmetros, executa a tarefa solicitada e transfere o controle de volta à fase 1.

Conforme já mencionado, todos os blocos de comando foram escritos de forma a ter apenas um ponto de entrada e um ou mais pontos de saída. O ponto de entrada é acessado pelo GOTO calculado do processador de comandos e os pontos de saída sempre devolvem o controle ao rótulo 5, que é a entrada do processador de comandos.

A razão de se usar esta técnica de programação foi fazer com que os blocos de comando se tornassem entidades estanques, perfeitamente definidas no corpo do programa. Em nenhum outro ponto foi usada a instrução GOTO. A comunicação entre os blocos, quando necessária, é feita através das matrizes básicas ISPEC, YT, YS, YR, LINHA1 e LINHA2 e de variáveis auxiliares de controle.

Para se ter uma melhor compreensão do uso destas técnicas, vejamos alguns exemplos:

Tomemos inicialmente o comando CP (campo no ponto). Este comando exibe na tela o valor do campo magnético em um ponto qualquer do espectro. O seu o bloco de comando, que é do tipo 1, é o seguinte:

```
-----------------------------------------------
180 WRITE(1,**)MG,'PONTO?',AM,' _'        <<< ENTRADA

    READ(1,*,END=5,IOSTAT=IOS,ERR=900) IPCP
    IF(IPCP.LE.0.OR.IPCP.GT.NPT) THEN
      WRITE(1,*) VM,'PONTO INDEFINIDO'

      GOTO 5                              <<< SAIDA C/ ERRO

    END IF
    CP = HMINT + (IPCP-1) * HINCT
    WRITE(1,.........) VD, IPCP, CP

    GOTO 5                                <<< SAIDA NORMAL
-----------------------------------------------
```

Neste bloco, a tarefa a ser executada é calcular e exibir o valor do campo magnético em um dado ponto do espectro de trabalho. O bloco é acessado pelo processador de comandos através do rótulo 180. A partir deste ponto, o bloco pede ao usuário o valor do ponto através da exibição da mensagem PONTO? e testa a resposta quanto a END e ERR usando os recursos normais do FORTRAN. Em seguida, a resposta é testada quanto à sua validade lógica por comparação com zero e com NPT (número total de pontos do espectro de trabalho) e caso haja erro uma advertência é exibida e o controle volta ao rótulo 5. Caso contrário, o valor do campo é calculado, o resultado exibido e controle devolvido ao processador de comandos. As variaveis MG, AM, VM e VD que aparecem no código acima são do tipo "character" e contêm sequencias de "escape" destinadas a programar as cores do terminal de vídeo. Os recursos de cor do terminal de vídeo são usados intensivamente na maioria dos blocos de comandos deste programa.

O exemplo acima refere-se a bloco do Tipo 1. Os blocos do Tipo 2 são de estruturas semelhantes às do Tipo 1 exceto pelo fato de conterem chamadas a subrotinas. Continuando os exemplos, vejamos agora o caso de um bloco do Tipo 3. Neste tipo de bloco o programa MASPE, através de uma chamada ao sistema operacional, utiliza um programa auxiliar para a execução de uma tarefa. Esta técnica é de grande importância para os programas de tempo real e também permite um melhor aproveitamento da memória disponível no sistema. Por estas razões, relembramos aqui alguns pontos básicos:

Um programa principal ao invocar a execução de um programa auxiliar tem duas opções básicas:

1 - Aguardar a execução total do programa auxiliar para então retomar a sua própria execução.

2 - Iniciar a execução do programa auxiliar e retomar imediatamente a sua própria execução.

No primeiro caso, o programa principal fica em um estado de "espera" enquanto o programa auxiliar é executado podendo inclusive ser retirado da memória para dar espaço a outros usuários. Quando do encerramento do programa auxiliar, o sistema operacional se encarrega de providenciar a continuação das atividades do programa principal.

No segundo caso, o programa principal apenas dispara a execução do auxiliar e não espera pelo seu encerramento. Neste caso, o programa auxiliar é executado de acordo com sua prioridade dentro do sistema.

Em ambos os casos, o programa principal pode passar dados para o programa auxiliar através da memória do sistema ou mesmo através de arquivos no disco. Quando o programa principal espera o fim do programa auxiliar ele pode também receber dados vindos do auxiliar através da memória do sistema.

No programa MASPE este recurso é utilizado nos comandos WH e HE cujas tarefas são, respectivamente, executar o programa WHZAT e exibir uma tela de ajuda (Help).

O programa WHZAT é um programa do sistema operacional que dá várias informções sobre o funcionamento do computador em um dado instante.

A tela de ajuda é uma lista contendo todos os comandos do MASPE com um resumo da tarefa. Os dados para esta tela estão contidos em um arquivo que é exibido pelo programa auxiliar MSHLP

Para execução do WHZAT pode-se optar pela listagem na tela ou em qualquer outro periférico de saída. Quando a opção é pela tela, o MASPE espera o fim da execução do WHZAT para retomar suas atividades. Quando a saída se processa em outro periférico, o MASPE não espera o WHZAT e retoma imediatamente o processamento.

O comando HE, por outro lado, só produz saídas na propria tela do usuário. Por este motivo, o MASPE espera o fim da execução do programa auxiliar MSHLP.

Vejamos agora os "blocos" destes comandos:

Comando WH:

```
110  if(it(1).eq.1.and.ip(1).eq.1) then Opção Tela ?
       write(1,*) clds                  Apaga a Tela
       iwh = 9                          Código " wait "
     else                               Se não é Tela
       iwh = 10                         Código " no wait "
     end if
     call EXEC (iwh, 5hWHZAT, ip(1), ip(2)) Despacha WHZAT
     write(1,*) ' '
     write(1,*) am, 'Retornando ao programa MASPE'
     goto 5                             Retorna ao label 5
```

Comando HE

```
120  write(1,*) cldsp                   Apaga tela e executa
     call EXEC(9, 6hMSHLP )             MSHLP opção "wait"
     goto 5                             Retorna ao label 5
```

Vejamos agora a estrutura dos blocos do Tipo 4. Nestes blocos, a execução da tarefa fica a cargo de um segmento (overlay) do programa principal. A diferença entre um programa auxiliar e um segmento reside principalmente na forma pela qual os mesmos são alocados na memória.

No caso do programa auxiliar, esta alocação se faz em termos de partições, ou seja, o sistema operacional carrega o programa em qualquer partição disponível na memória.

No caso do segmento, este é sempre carregado na mesma partição onde reside o programa principal, na região de memória disponível entre o fim do programa principal e o fim da partição. Neste caso, a comunicação de dados é feita através dos COMMONS declarados em ambos os programas. No programa MASPE, várias tarefas são executadas por meio de segmentos e os seus blocos são essencialmente idênticos.

Vejamos o bloco do comando LS (leitura sequencial)

```
130 call SEGLD( 6hMASLS, ier)
    if(ier.ne.0) write(1,*) 'Erro - SEGLD/MASLS: ', ier
    goto 5
```

Na primeira linha o sistema operacional é chamado através da rotina SEGLD, para carregar e executar o segmento MASLS. Terminada a execução o controle volta para o MASPE que testa a variável

⟨ier⟩ para verificar se houve algum problema com a execução do segmento (ier # 0). E' bom lembrar que os erros eventualmente detetados pela variável ⟨ier⟩ referem-se a problemas do sistema operacional, do disco, etc., e não teem nada a ver com os erros de operação do MASPE.

Os programas de aquisição de dados dos sistemas de aquisição passiva (SMAP) e do motor de passo (SMOP) foram escritos com a mesma técnica do MASPE. Em ambos existe um "processador de comandos" que interpreta os comandos do usuário e transfere o contrôle ao "bloco" adequado.

Nas páginas seguintes encontra-se uma descrição detalhada de todos os comandos do MASPE. Nesta descrição, cada comando é apresentado no formato do quadro abaixo:

```
+-----------------------------------------------------------+
|                                                           |
|                                          --------         |
|                                          CMD = XX         |
|                                          --------         |
|                                                           |
|  Tarefa:       ...........                                |
|                                                           |
|  Sintaxe:      XX  [,p1] ..... [,p7]                      |
|                                                           |
|  Erros:      : ...........                                |
|                                                           |
|  Comentários: ..........                                  |
|                                                           |
+-----------------------------------------------------------+
```

O significado dos campos deste quadro é o seguinte:

Tarefa: E' o procedimento computacional executado pelo comando XX

Sintaxe: E' a forma pela qual o comando XX deve ser submetido ao programa. Os parâmetros p1 ... p7 são sempre separados por vírgulas e podem ser:
- dois caracteres alfabéticos
- um número decimal, inteiro, de 16 bits

Os parâmetros que aparecem entre colchetes são opcionais. Em algus comandos a ausência do parâmetro opcional implica no uso de "defaults". Em outros, esta ausência faz com que o programa peça explicitamente o seu valor ou exiba uma mensagem de erro.

Erros: Este campo informa sobre os erros detectados no comando e suas causas mais prováveis.

Comentários: Neste campo é feita uma descrição do uso do comando, dos parâmetros, etc. Em alguns casos são apresentados exemplos.

Relação dos Comandos do MASPE

AD - Adiciona constante
AF - Define parâmetros para "area fill"
AR - Ajusta reta na linha de base
CE - Reposição de canal espúrio
CP - Exibe campo no ponto
DI - Define incremento para o comando MO
DS - Define Sub-espectro
DT - Define Espectro de Trabalho
DV - Calcula a derivada de um espectro
EG - Retorna ao "editor gráfico"
EN - Encerra o programa
ET - Exibe informações sobre o Espectro de Trabalho
FD - Aplica filtro digital
FT - Calcula Transformada de Fourier (FFT)
GR - Grava dados em fita magnética
HE - Exibe lista de comandos disponíveis
IC - Exibe índice dos espectros na memória
ID - Calcula integral definida
IN - Calcula integral indefinida (todo o espectro)
IY - Inverte Espectro de Trabalho
LA - Leitura aleatória na fita magnética
LD - Leitura no disco
LI - Lista dados de espectro
LK - Prende ou libera a unidade de fita magnética
LO - Controla a impressora acoplada ao terminal
LS - Leitura sequencial na fita magnética
MD - Calcula médias no Espectro de Trabalho
MF - Controla movimentos da fita magnética
MM - Calcula máximos e mínimos
MO - Mostra espectros na tela gráfica
MS - Calcula média de espectros (signal averager)
PA - Coloca o programa em PAUSE
PC - Exibe valor do ponto no campo
PF - Executa uma pesquisa na fita magnética
PL - Executa gráficos no plotter
PP - Executa mudança no sinal de "pronto"
RW - Rebobina a fita magnética
SE - Exibe informações sobre o Sub-espectro
SI - Cria Espectro de Trabalho simulado
SM - Aplica filtro "smooth"
ST - Exibe o "status" do programa
SY - Interface com o sistema operacional RTE-6VM
TF - Programa as teclas de função
TT - Coloca o cursor no topo da tela e apaga
WH - Executa o programa WHZAT
YC - Exibe o sinal no campo
YP - Exibe o sinal no ponto

Data da última edição: <900407.1628>

Apêndice 2

Programas e Comandos do SMAP

PROGRAMA SMAP

DESCRIÇÃO GERAL

&

DESCRIÇÃO DOS COMANDOS

SMAP: Sistema Melhorado de Aquisição Passiva

O SMAP é o programa básico do que chamamos "Sistema Melhorado de Aquisição Passiva". Esse programa foi desenvolvido com a mesma metodologia empregada no programa MASPE, ou seja, é um programa modular, totalmente escrito em FORTRAN 77, que funciona tal como um sistema operacional. A interação com o usuário é feita através de um conjunto de comandos parametrizados que, quando executados, levam à execução de tarefas específicas, relacionadas com a aquisição de dados nas experiências de Ressonância Magnética (EPR).

Por se tratar de um programa para aquisição de dados em tempo real, o SMAP usa alguns recursos do sistema operacional RTE-6 e do equipamento HP 1000 que raramente são empregados em outras situações e que, portanto, merecem um pouco mais de atenção.

O primeiro desses recursos é o uso do barramento HP-IB (ou GP-IB, norma IEEE-488) para a interligação dos instrumentos. Esse barramento, que se encontra convenientemente documentado nos manuais do computador, é uma via paralela de 16 bits que suporta vários instrumentos ligados a uma unidade de controle. Cada um dos instrumentos ligados a essa via tem um endereço que o identifica e pode ser um transmissor ("talker"), um receptor ("listenner") ou mesmo desempenhar ambas as funções.

A ligação física entre os instrumentos é feita através de cabos padronizados, que podem ter até cerca de 15 metros de comprimento e, embora exista a possibilidade de se endereçar até 32 instrumentos em uma via, esse número é limitado a 14 por razões técnicas.

No nosso caso, como o computador se encontra a 50 metros do local de uso dos instrumentos, tivemos que empregar um par de extensores HP-IB para contornar a limitação de distância já mencionada. O extensor é um instrumento, em parte semelhante a um modem, que trabalha com palavras de 16 bits, que são serializadas e transmitidas a uma taxa de 50 kbits/s através de uma linha coaxial de 75 ohms, sendo o seu uso totalmente transparente para o sistema.

Do ponto de vista do sistema operacional, cada um dos instrumentos presentes no barramento HP-IB é uma Unidade Lógica como um outro periférico qualquer. Em particular, na nossa instalação, o instrumento de endereço 1 é a unidade 31, o de endereço 2 é a unidade 32 e assim sucessivamente. Com essa configuração, para se mandar uma mensagem para o intrumento de endereço 12, através de um programa em FORTRAN, basta usar a instrução

```
WRITE(42,*)  '(mensagem)'
```

Da mesma forma, para se receber a resposta desse mesmo instrumento, basta usar a instrução

```
READ(42,'(format)') (resposta)
```

Para os instrumentos que necessitam de comandos mais sofisticados ou que usam cşdigos binários para transmissão dos dados, o sistema operacional dispõe de uma série de rotinas e funções específicas que podem ser usadas pelo programa principal.

Um outro recurso vital para a realização das tarefas desse programa é o processo de entrada e saída (E/S) de dados, disponível no RTE, chamado de "CLASS I/O". Com esse processo, quando um programa necessita fazer um operação de E/S, ele solicita ao sistema operacional a abertura de uma "classe" na memória do sistema e

informa a unidade lógica responsável pela transação e o número de bytes do registro (buffer) a ser transmitido ou recebido. O sistema operacional acolhe a solicitação de E/S e libera o programa para outras tarefas. O término da operação é sinalizado através de "flags" que podem ser consultados pelo programa a qualquer instante.

No caso de leitura de dados em tempo real, o uso da leitura por classe permite que os dados sejam lidos segundo uma base de tempo externa, o que é fundamental para a exatidão do processo. Como exemplo, vejamos o trecho do código fonte do SMAP responsável pela leitura dos dados do ADC, comando AQ:

```
program SMAP
...
...
call CLRQ(ifunc, iclas)          <- abertura de classe
...
call EXEC(17,36,..,npc,..        <- ler 2*npc bytes na LU 36

     A partir desse instante, o programa SMAP fica
     livre para a execução de outras tarefas, en-
     quanto que os dados provenientes do ADC (LU
     36) são lidos pelo sistema operacional e co-
     locados em um buffer temporário, localizado
     em uma região da memória chamada SAM (System
     Avaiable Memory). A taxa com que os dados são
     lidos é determinada pela base de tempo do ADC
     não sendo, portanto, influenciada pela ativi-
     dade porventura existente na CPU. O fim da
     aquisição é testado em seguida...

call EXEC(21,iclas+40000B,..     <- testa o fim da leitura

call EXEC(21,iclas+20000B,..     <- retira os dados de SAM
...
...
end
```

O uso de programas subordinados é também um recurso explorado no SMAP. O programa subordinado é um programa comum que é colocado em execução por um outro programa e não pelo sistema operacional. Ao colocar um programa subordinado em execução o programa principal, também chamado "programa pai", pode optar por continuar funcionando ou aguardar o término da execução do subordinado (filho). Além disso, o programa pai pode passar dados iniciais para o filho e receber dados de volta, caso aguarde o seu término.

No SMAP são usados programas subordinados para gravação de dados em disco, para controle da varredura e para obtenção do status do RTE (WHZAT). Vejamos o caso da gravação em disco, comando GD:

```
...
call EXEC(15,...                 <- reserva uma trilha de serviço
...
call EXEC(2,...                  <- grava dados na trilha reservada
...
call XQPRG(..,.GDISC,..          <- executa GDISC e aguarda término

     A partir desse instante, o programa GDISC
     inicia o seu funcionamento enquanto que o
```

SMAP aguarda seu término. O GDISC lê a trilha de serviço, cujo endereço lhe foi passado pelo SMAP, e grava os dados em um arquivo do usuário. O nome do arquivo do usuário se encontra entre os dados que foram lidos. Ao terminar a gravação, o GDISC libera a trilha de serviço e devolve um parâmetro de controle para o SMAP. Se houver algum problema com a gravação, este parâmetro será um número negativo.

```
if(ipr(1).eq.-1)...        (- ao reassumir o controle o SMAP
                              testa a ocorrência de erro.
...
```

Os dados colhidos pelo SMAP são gravados no disco em arquivos de acesso direto (binário), em um único registro de 6400 bytes de comprimento. Embora a manipulação de arquivos diretos seja um pouco mais trabalhosa que a dos arquivos sequenciais, este procedimento é plenamente justificado pela grande economia de espaço obtida. Para as gravações em fita magnética, existe a opção de gravação sequencial em ASCII ou EBCDIC para facilitar o transporte de dados para outras instalações. A colocação dos dados em um único registro é obtida através dos "EQUIVALENCEs" entre os vetores IDAT, IVOLT e IABUF.

Os aspectos mais gerais relativos à estruturação desse programa estão contidos no manual do programa MASPE pois, como já mencionamos, ambos foram escritos com a mesma filosofia. O detalhamento de várias instruções pode ser visto na listagem dos programas fonte onde, em várias situações, os comentários foram feitos linha por linha.

Nas páginas seguintes, apresentamos uma relação detalhada dos comandos do SMAP.

COMANDOS DO SMAP

AD - Executar conversão em um dado canal
AQ - Iniciar o processo de aquisição
AT - Mudar para o modo automático
CA - Programar o canal do ADC
CR - Calibrar a rampa
CS - Cancelar exibição automática do status
DF - Colocar valores de default
DM - Controlar multímetro digital
DP - Exibir gráfico no vídeo
EN - Encerrar o programa
EP - Executar programa subordinado
ES - Habilitar exibição de status
FC - Programar a frequência de aquisição
FE - Exibir fundo de escala do ADC
GD - Gravar dados no disco
GN - Comandar goniômetro motorizado
GR - Gravar dados em fita magnética
HE - Exibir resumo dos comandos
IG - Ignorar dados na memória
LC - Listar canais
LR - Levar a rampa a um dado ponto
LK - Prender ou liberar a fita magnética
MF - Comandar os movimentos da fita magnética
MG - Mostrar grupo de comandos na memória
OS - Medir offset
PA - Colocar o programa em pausa
PF - Fazer uma pesquisa na fita magnética
PL - Fazer o gráfico dos dados
PN - Criar um prefixo de nome de arquivo
PR - Exibir parâmetros da reta
PV - Determinar % de varredura
QF - Escolher frequência de aquisição
RG - Repetir um grupo de comandos
RP - Exibir a posição corrente da rampa
ST - Exibir status
SY - Interface com o RTE
T0 - Zerar relógio do sistema
TD - Mostrar o tempo transcorrido
TF - Definir um título fixo
TI - Exibir o tempo de aquisição
TV - Determinar o tempo total de varredura
VC - Ler a intensidade do campo com o gaussímetro
WH - Executar o programa WHZAT,AL
(- Iniciar um grupo de comandos
) - Finalizar um grupo de comandos

Apêndice 3

Programas e Comandos do SMOP

```
&SMP7  T=00004 IS ON CR00002 USING 00060 BLKS R=0000
      ftn7x,l
*     ------------------------------------------------------------------
      program smpy,3,5                                    ! (880713.1009)
*     ------------------------------------------------------------------
      integer*2 nome(40), passo(0:3), ibuf(10), idir(2)
      integer*2 is(2000)                                  ! Espectro "subida"
      integer*2 id(2000)                                  ! Espectro "descida"
*     ------------------------------------------------------------------
      integer*4 ElapsedTime, msegs
*     ------------------------------------------------------------------
      real*4    x(2000), y1(2000), y2(2000), y3(2000)
      real*4    temp(100)
*     ------------------------------------------------------------------
      logical   valido, first
*     ------------------------------------------------------------------
      character dir*5
*     ------------------------------------------------------------------
      equivalence (ibuf,nome(21)), (idir,nome(39))
*     ------------------------------------------------------------------
      data passo /06B, 12B, 11B, 05B/                     ! Config. do motor
      data LuADC /36/                                     ! LU  ADC
      data LuMPG /44/                                     ! LU  Multiprogramador
      data LuDMM /42/                                     ! LU  Multimetro 195A
*     ------------------------------------------------------------------
      call lurq(1,8,1)                                    ! LU8 Lock
      write(1,*) char(12),'Programa SMPx - Versao Preliminar'
*     ------------------------------------------------------------------
      write(1,*) 'Fita Nova? [si na]'
      read (1,'(a2)') iresp
      call clcuc(iresp,1)
      if(iresp.ne.2hSI) then
        write(1,*) 'SMP/ Aguarde posicionamento da fita.'
        call finda(nfile)
        write(1,*) 'SMP/ Esta fita ja contem',nfile,' arquivos.'
      end if
*     ------------------------------------------------------------------
   10 call exec(9,6hTELA1 )
      call Izero(is,2000)
      call Izero(id,2000)
      continue
      call locc1(16,1)
      write(1,*) 'Quantos passos?, Incremento?'
      read (1,*,err=10)  ndpa, incr
      call loc(6,65)
      write(1,*) ndpa
      call loc(5,65)
      write(1,*) incr
*     ------------------------------------------------------------------
      call locc1(16,1)
      write(1,*) 'Quantas varreduras?'
      read (1,*,err=10)  nvar
      if(mod(nvar,2).ne.0) then
        write(1,*) 'Nesta versao do programa so e valido'
        write(1,*) 'um numero par de varreduras!'
        goto 10
      end if
      call loc(9,21)
      write(1,*) nvar
*     ------------------------------------------------------------------
      call locc1(16,1)
      write(1,*) 'Canal do ADC?'
      read (1,*,err=10)  ncan
      call loc(3,65)
      write(1,*) ncan
*     ------------------------------------------------------------------
      call locc1(16,1)
      write(1,*) 'Pausa? [em milisegundos]'
      read (1,*,err=10)  ipaus
      call loc(8,65)
      write(1,*) ipaus
*     ------------------------------------------------------------------
      call Reset_ADC (ncan, LuADC)                        ! Configura o ADC
*     ------------------------------------------------------------------
      call Cntrl_MPG (60B, LuMPG)                         ! Configura Multiprog.
*     ------------------------------------------------------------------
      call Init_K195A (LuDMM)                             ! Configura Multimetro
*     ------------------------------------------------------------------
      call locc1(16,1)
      write(1,*) 'Certifique-se de que o potenciometro de 10 voltas'
      write(1,*) 'encontra-se no extremo  anti-horario de seu curso'
      pause
*     ------------------------------------------------------------------
      i = 1
      first = .true.
      ivar = 0
      call step(i)
      write(1,*) 'Motor ligado e preso na posicao inicial'
      call locc1(16,1)
   15 write(1,*) 'Valor do campo?'
      read (1,*,err=15) b1
      pause
*     ------------------------------------------------------------------
*     Inicio de varredura de subida
*     ------------------------------------------------------------------
  100 i = 1
      call loc( 7,3)
      write(1,'(58(" "),a4,"IND:")') char(27)//'[7m'
*     write(1,*) char(27)//'[K'
      valido = .false.
      ks = 1
      dir = 'sobe'
*
      call mede_temp(temp(ivar+1))                        ! Mede temperatura
      call loc(9,41)
      write(1,*)  temp(ivar+1)                  ! informa
*
      call ResetTimer
      call p_ponto(LuADC,i,ks,incr,valido,is,dir,ipaus)
      i = i+1
*     ------------------------------------------------------------------
      do j = i,ndpa
      call p_ponto(LuADC,j,ks,incr,valido,is,dir,ipaus)
      end do
*     ------------------------------------------------------------------
      msegs = ElapsedTime()
      if(first) then
        npe = ks -1
        first = .false.
```

```
0179        x(j) = b1 + delta_b*(j-1)
0180        y1(j) = is(j)/div
0181        y2(j) = id(j)/div
0182        end do
0183  *     ------------------------------------------------------------------
0184     80 write(1,*) 'Plot? [1-sobe, 2-desce, 3-soma, 4-cancela]'
0185        read (1,*,err=80) iplot
0186        if(iplot.eq.4) goto 97
0187  *     ------------------------------------------------------------------
0188        write(1,*) 'Nome do espectro?'
0189        read(1,'(20a2)') (nome(k),k=1,20)
0190     95 write(1,*) 'X-tic? '
0191        read(1,*,err=95) xt
0192  *     ------------------------------------------------------------------
0193        goto(82,84,86) iplot
0194  *     ------------------------------------------------------------------
0195     82 call code
0196        write(idir,'("SO")')
0197        call hplot(x,y1,npe,1,7,nome,xt)          ! Plota subida
0198        goto 80
0199  *     ------------------------------------------------------------------
0200     84 call code
0201        write(idir,'("DE")')
0202        call hplot(x,y2,npe,1,7,nome,xt)          ! Plota descida
0203        goto 80
0204  *     ------------------------------------------------------------------
0205     86 do j = 1,npe
0206        y3(j) = (y1(j)+y2(j))/2.
0207        end do
0208        call hplot(x,y3,npe,1,7,nome,xt)          ! Plota soma
0209        goto 80
0210  *     ------------------------------------------------------------------
0211     97 write(1,*) 'Grava em fita?'
0212        read(1,'(a2)') iresp
0213        call clcuc(iresp,1)
0214        if(iresp.eq.2hSI) then
0215          write(8,'(40a2)') (nome(k),k=1,40)
0216          write(8,*) npe,b1,b2,avg
0217          call reio(2,110b,is,npe)
0218          endfile 8
0219          write(8,'(40a2)') (nome(k),k=1,40)
0220          write(8,*) npe,b1,b2,avg
0221          call reio(2,110b,id,npe)
0222          endfile 8
0223          write(1,*) 'Gravado'
0224        end if
0225  *     ------------------------------------------------------------------
0226        write(1,*) 'Outro espectro? [si na]'
0227        read(1,'(a2)') iresp
0228        call clcuc(iresp,1)
0229        if(iresp.eq.2hSI) goto 10
0230        endfile 8
0231        call exec(3,510b)
0232        call lurq(0,8,1)
0233        write(1,*) '-------------------------------------'
0234        write(1,*) char(12),'Fim do Programa SMPx'
0235        write(1,*) 'Fita magnetica rebobinada e desligada'
0236        write(1,*) '-------------------------------------'
0237        stop
0238        end
```

```
0119          call loccl(16,1)
0120     20   write(1,*) 'Valor do campo?'
0121          read (1,*,err=20) b2
0122          binc = (b2-b1)/(ndpa)                    !m
0123          delta_b = binc*(incr)                    !m
0124        end if
0125        i = ndpa
0126        ivar = ivar + 1
0127        call loccl(16,1)
0128        write(1,*) ' '
0129        write(1,*) 'Fim da varredura: ',ivar, ' T: ', msegs
0130        if(ivar.eq.nvar) goto 90
0131  *
0132  *     Fim da varredura de subida
0133  *     ==========================================================
0134  *     Inicio da varredura de descida
0135  *
0136        call loc(7,3)
0137        write(1,'(58(" "),a4,"IND:")') char(27)//'[7m'
0138  *
0139        kd = ks -1
0140        valido = .false.
0141        i = ndpa
0142        dir = 'desce'
0143  *
0144        call mede_temp(temp(ivar+1))              ! Mede temperatura
0145        call loc( 9,41)
0146        write(1,*)          temp(ivar+1)          ! Informa
0147  *
0148        call ResetTimer
0149        call p_ponto(LuADC,i,kd,incr,valido,id,dir,ipaus)
0150        i = i-1
0151  *     ------------------------------------------------------------------
0152        do j = i,1,-1
0153        call p_ponto(LuADC,j,kd,incr,valido,id,dir,ipaus)
0154        end do
0155  *     ------------------------------------------------------------------
0156        msegs = ElapsedTime()
0157        ivar = ivar + 1
0158        call loccl(16,1)
0159        write(1,*) 'Fim da varredura: ',ivar, ' T: ',msegs
0160        if(ivar.eq.nvar)goto 90
0161        goto 100
0162  *     ------------------------------------------------------------------
0163     90 call Cntrl_MPG(20B, LuMPG)                ! Corta motor
0164        call loccl(16,1)
0165        write(1,*) ' '
0166        write(1,*) 'Fim da aquisicao'
0167        write(1,*) 'Motor cortado'
0168        write(1,*) b1,b2
0169        write(1,*) 'DMM: ', (temp(j), j=1,nvar)
0170        do j=1,40
0171        nome(j) = 2h
0172        end do
0173        call avgst(temp,nvar,avg,std)
0174        call code
0175        write(ibuf,'("T: ",e11.5,4x,"DP: ",e11.5 )') avg, std
0176  *     ------------------------------------------------------------------
0177        div = nvar/2.
0178        do j = 1,npe
```

```
0239  *
0240  *     ================================================================
0241        subroutine p_ponto(LuADC,i,k,incr,valido,ispec,dir,ipaus)
0242  *     ================================================================
0243        integer   ispec(1)
0244        logical   valido
0245        character dir*5
0246  *     ----------------------------------------------------------------
0247        if(mod(i-1,incr).eq.0) then
0248          call Mede(LuADC,ivolt)
0249          valido = .true.
0250        else
0251          valido = .false.
0252        end if
0253  *
0254        call step(i,ipaus)
0255        if(mod(i-1,40).eq.0) then
0256          pct = -50. + .05*(i-1)
0257          call pofr(pct,dir)
0258        end if
0259  *
0260        if(valido) then
0261          ispec(k) = ispec(k) + ivolt
0262          if(mod(k,10).eq.0) then
0263            call loc(7,66)
0264            write(1,'(i4,a4," _")')k, char(27)//'[0m'
0265          end if
0266
0267          if(dir.eq.'sobe') then
0268            k = k+1
0269          else
0270            k = k-1
0271          end if
0272          valido = .false.
0273        end if
0274        return
0275        end
0276  *
0277  *     ================================================================
0278        subroutine Mede (LuADC, ivolt)
0279  *     ================================================================
0280        write(LuADC,'("J_")')
0281        call Reio (1, 100B+LuADC, ivolt, 1)
0282        write(LuADC,'("I_")')
0283  *     write(1,*) ivolt
0284        return
0285        end
0286  *
0287  *     ================================================================
0288        subroutine step(i,ipaus)
0289  *     ================================================================
0290        integer*2 passo(0:3)
0291        data passo /6b,12b,11b,5b/
0292  *     ----------------------------------------------------------------
0293        indx = iand(i,3)
0294        write(44,'("L",@5,"T")') passo(indx)
0295        if(ipaus.gt.0) then
0296        idelay = ipaus/10
0297        call exec(12,0,1,0,-idelay)
0298        end if
0299        return
0300        end
0301  *
0302  *     ================================================================
0303        subroutine Reset_ADC (ncan, LuADC)
0304  *     ================================================================
0305        nc = 2**(ncan-1)
0306        write(LuADC, '("HL", I1, "A_")') nc
0307        return
0308        end
0309  *     ================================================================
0310        subroutine Cntrl_MPG (icnt, LuMPG)
0311  *     ----------------------------------------------------------------
0312        write(LuMPG,'("0",@2,"T")') icnt
0313        return
0314        end
0315  *     ================================================================
0316        subroutine Izero (i, np)
0317  *     ----------------------------------------------------------------
0318        dimension i(1)
0319        do k = 1,np
0320        i(k) = 0
0321        end do
0322        return
0323        end
0324  *     ================================================================
0325        subroutine init_K195A (lu)
0326  *     ----------------------------------------------------------------
0327        call rmote(lu)                                  ! DMM modo remoto
0328        call clear(lu,1)                                ! Selective Dev Clear
0329        write(lu,'("F0X")')                             ! DC volts
0330        write(lu,'("R0X")')                             ! Auto Range
0331        return
0332        end
0333  *     ================================================================
0334        subroutine mede_temp (temp)
0335  *     ----------------------------------------------------------------
0336        dimension temp(1),ibuf(2)
0337        write(42,'("B0X")')
0338        read (42,'(2a2,e11.5)') ibuf, temp
0339        return
0340        end
0341  *     ================================================================
0342        subroutine avgst(x,n,avg,std)
0343  *     ----------------------------------------------------------------
0344        dimension x(1)
0345        call vsum(s1,x,1,n)
0346        call vdot(s2,x,1,x,1,n)
0347        avg = s1/n
0348        var = (s2 -s1*s1/n)/(n-1)
0349  *     std = sqrt(var)
0350        std = 0.
0351        return
0352        end
0353  *     ================================================================
0354        subroutine tela
0355        write(1,*) char(12)
0356        il = 10
0357        ic = 10
0358        call cursor_ansi(il,ic)
```

```
      write(1,'(53("-"))')
      il = 11
      ic = 30
      call cursor_ansi(il,ic)
      write(1,'("Percent of Range")')
      il = 12
      ic = 10
      call cursor_ansi(il,ic)
      write(1,'("-50  -40  -30  -20  -20   0   +10  +20  +30  +40  +50")
     .')
      il = 13
      ic = 10
      call cursor_ansi(il,ic)
      write(1,'(" |....|....|....|....|....|....|....|....|....|....| ")
     .')
      il = 15
      ic = 10
      call cursor_ansi(il,ic)
      write(1,'(53("-"))')
      return
      end
*     ------------------------------------------------------------------
      subroutine loc(il,ic)
*     ------------------------------------------------------------------
      write(1,'(A1,"[",i2,";",i2,"H_")') char(27), il, ic
      return
      end
*     ------------------------------------------------------------------
      subroutine loccl(il,ic)
*     ------------------------------------------------------------------
      character esc*1
      esc = char(27)
      write(1,'(a1,"[",i2,";",i2,"H",a1,"[J_")')esc,il,ic,esc
      return
*     ------------------------------------------------------------------
      end
*     ==================================================================
      subroutine pofr(pct,dir)
*     ------------------------------------------------------------------
      character dir*5, simbo*1
      data iof /5/ , il/07/
*     ------------------------------------------------------------------
      if(dir.eq.'sobe') then
        simbo = '>'
      else
        simbo = '<'
      end if
*     ------------------------------------------------------------------
      ipct = int(pct)
      ic = ipct/2 + 26 + iof
      call loc(il,ic)
      write(1,'(a1)') simbo
*     ------------------------------------------------------------------
      return
      end
      end$
```

```
&SMOF1 T=00004 IS ON CR00026 USING 00108 BLKS R=0000

0001   ftn7x,l
0002   *
0003   *     ================================================================
0004   *     XXXXXXX  XX   XX    XXXXX   XXXXXX     ! Fisica - ICEx/UFMG
0005   *     XX       XXX XXX   XX   XX  XX   XX    ! Programa:.. SMOP
0006   *      XXXXX   XX X XX   XX   XX  XXXXXX     ! Versao:.... 01
0007   *          XX  XX   XX   XX   XX  XX         ! Data:..... 21/03/89
0008   *     XXXXXXX  XX   XX    XXXXX   XX         ! Edit:..... (890816.1056)
0009   *                                            ! Autor:..... R A Nogueira
0010   *     ----------------------------------------------------------------
0011   *     Programa para aquisicao de dados de EPR atraves do Sistema do
0012   *     Motor de Passo ( SMOP )
0013   *     ================================================================
0014   *
0015   *     ----------------------------------------------------------------
0016   $EMA  /xy/                                   ! Memoria extendida
0017         program smop1
0018   *     ----------------------------------------------------------------
0019   *
0020   *     Principais rotinas do sistema chamadas neste programa:
0021   *
0022   *       SEGLD  - Carrega overlay na memoria
0023   *       REIO   - Entrada e Saida reentrante
0024   *       ABREG  - Devolve valores dos reg. A & B
0025   *       CLCUC  - Converte minuscula p/ maiuscula
0026   *       PARSE  - Separa parametros em um string
0027   *       EXEC   - Funcoes do RTE
0028   *
0029   *     ................................................................
0030   *     Segmentos deste programa ( overlays ):
0031   *
0032   *       SMOIN  - Inicializacao
0033   *       SMOMF  - Movimentos da Fita Magnetica
0034   *       SMOVR  - Comandos:
0035   *       SMOCI  - Comandos CI:
0036   *
0037   *     ----------------------------------------------------------------
0038   *     COMMONS
0039   *     =======
0040   *     1) Variaveis localizadas na Extended Memory Area ( EMA ):
0041   *
0042         common /xy/    x(2001), y(2001)
0043   *     ................................................................
0044   *     2) Variaveis localizadas na area normal da memoria:
0045
0046         common /cmdpr/   ip(8), it(7)              ! Parametros do comando
0047         common /comdo/   icmd                      ! Mnemo do comando
0048         common /ulogs/   lu(10)                    ! Unidades Logicas LUs
0049         common /vlogs/   logic (10)                ! Variaveis logicas
0050         common /data1/   kbuf(4200)                ! Dados gerais
0051         common /parm1/   iclas, ifcm, npa, idel    ! Parametros (1)
0052         common /parm2/   nm_gss                    ! Parametros (2)
0053   *
0054   *     ----------------------------------------------------------------
0055   *     Outros vetores:
0056   *
0057         dimension mcmd(50),                        ! Matriz de comandos
0058        .          ibuf(4200),                      ! Dados (Veja Equiv.)
0059        .          idat( 200),                      ! Dados (Veja Equiv.)
0060        .          rbuf(2000),                      ! Dados (Veja Equiv.)
0061        .          iresp(20),                       ! Resposta ao "CMD?"
0062        .          ipars(33),                       ! Saida de PARSE
0063        .          ifd(6),                          ! ADC - frequencias
0064        .          if1(6),                          ! ADC - char. cntrl
0065        .          ident(5),                        ! Identif. do prog.
0066        .          icv(3)                           ! Cntrl video
0067   *
0068   *     ----------------------------------------------------------------
0069   *     Variaveis e constantes do tipo 'character'
0070   *
0071         character*1   esc, ff                      ! escape, formfeed
0072         character*4   prmpt                        ! string de Pronto
0073         character*5   vm, vd, az, am, bc           ! atributos de cores
0074         character*5   mg, cy, vr                   ! atributos de cores
0075         character*5   cldsp                        ! escape apaga tela HP
0076         character*16  iclend                       ! string calendario
0077         character*25  pinv                         ! string mensagem erro
0078   *
0079   *     ----------------------------------------------------------------
0080   *     Variaveis LOGICAS
0081
0082         logical   usa_fita, usa_dmm, usa_fmo, usa_gss,
0083        .          usa_mpg,  usa_gon, logic,   mot_lib
0084
0085   *     ----------------------------------------------------------------
0086   *     Definicoes dos EQUIVALENCES
0087   *
0088   *     1) Equipamentos HP-IB
0089   *        ------------------
0090         equivalence (lu_plt,lu(1)), (lu_adc,lu(2)), (lu_dmm,lu(3)),
0091        .            (lu_fmo,lu(4)), (lu_gsm,lu(5)), (lu_mpg,lu(6)),
0092        .            (lu_gon,lu(7)), (lu_gsv,lu(8)), (lu_gss,lu(9))
0093
0094   *     2) Variaveis Logicas
0095   *        -----------------
0096         equivalence (usa_fita,logic(1)), (usa_dmm,logic(2)),
0097        .            (usa_fmo, logic(3)), (usa_gss,logic(4)),
0098        .            (usa_mpg, logic(5)), (usa_gon,logic(6))
0099
0100   *     3) Matriz dos dados
0101   *        ----------------
0102   *     ----------------------------------------------------------------
0103   *     Kbuf     1                          4001              4200
0104   *     ----------------------------------------------------------------
0105   *     Jbuf     1                        4000
0106   *     ----------------------------------------------------------------
0107   *     Rbuf     1                        2000
0108   *     ----------------------------------------------------------------
0109   *     Idat                                  1                  200
0110   *     ----------------------------------------------------------------
0111
0112         equivalence (idat(1), kbuf(4001)),
0113        .            (jbuf(1), kbuf(1)),
0114        .            (rbuf(1), jbuf(1))
0115
0116         equivalence (idat(1),  ifc), (idat(2),   nop), (idat(3), ipv1),
0117        .            (idat(4), ipv2), (idat(5), nchan), (idat(6), iri1),
0118        .            (idat(7), iri2), (idat(8), nspec), (idat(9), ipnd),
```

```
0119        .           (idat(10),  np), (idat(11), nmrn), (idat(12),nmri),
0120        .           (idat(13), nop), (idat(14),  nop), (idat(15),  b1),
0121        .           (idat(17),  b2), (idat(19),  nop), (idat(20), nop)
0122
0123  *     ------------------------------------------------------------------
0124  *     Data:
0125  *
0126  *mcmd Dados da matriz de comandos:
0127  *
0128        data mcmd /2hEN,2hWH,2hHE,2hRW,2hMF,2hPA,2hPP,2hAF,2hCA,2hFC,
0129        .          2hSY,2hTT,2hPV,2hRI,2hNM,2hOS,2hAD,2hST,2hIV,2hFV,
0130        .          2hCZ,2hCT,2hLM,2hAQ,2hGD,2hGR,2hMO,2hPL,2hFE,2hGN,
0131        .          2hCD,2hWD,2hOW,2hIM,2hPC,2hCM,2hAT,2h  ,2h  ,2h  ,
0132        .          2h  ,2h  ,2h  ,2h  ,2h  ,2h  ,2h  ,2h  ,2h  ,2h  ,2h  /
0133  *     ------------------------------------------------------------------
0134        data ipv1 /-50/                          ! Limites da
0135        data ipv2 /+50/                          ! varredura
0136        data ifd  /  5,  10,  20,  50, 100, 200/ ! ADC - frequencias
0137        data ifl  /107b,106b,105b,104b,103b,102b/ ! ADC - cntrl. freq
0138        data nmrn /1/                            ! No. med. reg normal
0139        data nmri /1/                            ! N. med reg interesse
0140  *     ------------------------------------------------------------------
0141  *                       INICIALIZACAO DO PROGRAMA
0142  *                       =========================
0143  *
0144  *                                          Parametros de identificacao
0145  *                                          deste programa
0146  *                                          ---------------------------
0147        ident(1) = 1          ! Versao
0148        ident(2) = 0          ! Revisao
0149        ident(3) = 0          ! Codigo do SMOP
0150        ident(4) = 0          ! Formato da fita
0151        ident(5) = 0          ! Uso futuro
0152  *
0153  *     Escapes:
0154  *
0155        esc   = char(27)                         ! escape code
0156        ff    = char(12)                         ! form feed
0157  *     ..........................................................
0158  *                                              ! Atributos de cor dos
0159  *                                              ! terminais HP
0160  *
0161        bc = esc//'&v0S'                         ! branco
0162        vm = esc//'&v1S'                         ! vermelho
0163        vd = esc//'&v2S'                         ! verde
0164        am = esc//'&v3S'                         ! amarelo
0165        az = esc//'&v4S'                         ! azul
0166        cy = esc//'&v5S'                         ! cyano
0167        mg = esc//'&v6S'                         ! magenta
0168        vr = esc//'&v7S'                         ! video invertido
0169  *     ..........................................................
0170        cldsp = esc//'h'//esc//'J'               ! Home Up, Clr Display
0171        pinv  = vm//'Parametro(s) Invalido(s)'   ! Mensagem de erro
0172        prmpt = 'CMD)'                           ! Prompt inicial
0173        npa   = 0                                ! Pos. inic. do motor
0174  *     ------------------------------------------------------------------
0175                                                 ! Chama o segmento de
0176                                                 ! inicializacao e ve-
0177      1 call segld(6hSMOIN , ier)                ! rifica se foi execu-
0178        call ERCHK(ier, 1)                       ! tado com exito
0179  *
0180        call Cntrl_MPG(Lu_MPG, 20b)
0181        call InMot(Lu_MPG, 6b)
0182        mot_lib = .false.
0183  *     ------------------------------------------------------------------
0184  *                                 ========================
0185  *                                 PROCESSADOR DE COMANDOS
0186  *                                 ========================
0187  *
0188  *     Indice/Comando       Label      Comentario
0189  *
0190  *          1-EN             100       Fim do programa
0191  *          2-WH             110       Executa WHZAT,AL
0192  *          3-HE             120       Help
0193  *          4-RW             130       Rebobina a fita magnetica
0194  *          5-MF             140       Movimentos da fita magnetica
0195  *          6-PA             150       Pausa no programa
0196  *          7-PP             160       Modifica sinal de pronto
0197  *          8-AF             170       Parametros de Area Fill
0198  *          9-CA             180       Canal ativo do ADC
0199  *         10-FC             190       Frequencia de conversao
0200  *         11-SY             200       Interface com o RTE
0201  *         12-TT             210       Topo da tela ( clear dsp )
0202  *         13-PV             220       Def. precentuais de varredura
0203  *         14-RI             230       Def. regiao de interesse
0204  *         15-NM             240       Def. num. de medidas
0205  *         16-OS             250       Mede offset
0206  *         17-AD             260       Executa uma conversao
0207  *         18-ST             270       Exibe status do programa
0208  *         19-IV             280       Vai p/ inicio de IRI
0209  *         20-FV             290       Vai p/ o fim de IRI
0210  *         21-CZ             300       Vai p/ o inicio do curso
0211  *         22-CT             310       Vai p/ o fim do curso
0212  *         23-LM             320       Libera o motor de passo
0213  *         24-AQ             330       Inicia aquisicao
0214  *         25-GD             340       Grava no disco
0215  *         26-GR             350       Grava na fita magnetica
0216  *         27-MO             360       Exibe espectro na tela
0217  *         28-PL             370       Exibe espectro no plotter
0218  *         29-FE             380       Exibe fundo de escala do ADC
0219  *         30-GN             390       Movimentos do goniometro
0220  *         31-CD             400       Cria diretorio (CI)
0221  *         32-WD             410       Exibe diretorio corrente (CI)
0222  *         33-OW             420       Exibe dono do diretorio (CI)
0223  *         34-IM             430       Inicializa motor de passo
0224  *         35-PC             440       Posicao corrente do motor
0225  *         36-CM             450       Campo Magnetico ( gaussimetro )
0226  *         37-AT             460       Atraso entre passos
0227  *
0228  *
0229  *     ------------------------------------------------------------------
0230      5 write(1,*) mg, prmpt, am, ' _'           ! Exibe sinal de pronto
0231        call REIO (1, 401b, iresp, 20)           ! Le o string comando
0232        call ABREG (ia, ib)                      ! Le reg. A & B da CPU
0233        if (ib.eq.0) then                        ! Se reg B = 0 houve
0234          write(1,*) vm, 'TO/EOF/CR'             ! time-out, EOF ou CR
0235          goto 5                                 ! Retorna ao Processa
0236        end if                                   ! dor de comandos
0237  *     ..........................................................
0238        call CLCUC (iresp, ib)                   ! Conv. p/ maiuscula
```

```
0239        call PARSE (iresp, ib*2, ipars)          ! Separa os parametros
0240        icmd = ipars(2)                          ! Separa mnemo do coman
0241                                                 ! do em (icmd)
0242  *     ..................................................................
0243        kf = 0                                   ! Inicializa pointer
0244        do j = 1, 50                             ! Verifica se o coman
0245           if(icmd.eq.mcmd(j))  kf = j           ! do existe e qual sua
0246        end do                                   ! posicao em (MCMD)
0247        if(kf.eq.0) then                         ! Se o pointer continua
0248          write(1,*) vm, 'Comando Inexistente'   ! zero entao o comando
0249          goto 5                                 ! nao existe
0250        else                                     ! Caso contrario, recu-
0251          do k = 2,8                             ! pera os parms nos vet.
0252             ip(k-1) = ipars(4*k-2)              ! ip (vlr. do parm.),
0253             it(k-1) = ipars(4*k-3)              ! it (tipo do parm.)
0254          end do                                 ! e coloca em ip(8) o
0255          ip(8) = ipars(33)                      ! no. de parametros.
0256          goto (100,110,120,130,140,150,
0257       .        160,170,180,190,200,210,
0258       .        220,230,240,250,260,270,
0259       .        280,290,300,310,320,330,
0260       .        340,350,360,370,380,390,
0261       .        400,410,420,430,440,450,
0262       .        460 ) kf
0263        end if
0264  *     ------------------------------------------------------------------
0265  *
0266  *                          BLOCOS DOS COMANDOS
0267  *                          ===================
0268  *
0269  *     ------------------------------------------------------------------
0270  *     Encerra a execucao do programa                          ! CMD = EN
0271  *     ------------------------------------------------------------------
0272  *
0273    100 write(1,*) vm, '/SMOP: Fim'
0274        STOP 77
0275  *
0276  *     ------------------------------------------------------------------
0277  *     Executa WHZAT                                           ! CMD = WH
0278  *     ------------------------------------------------------------------
0279    110 if(it(1).eq.1.and.ip(1).eq.1) then       ! Verifica parametros
0280          write(1,*) cldsp                       ! Apaga a tela
0281          iwh = 9                                ! Executa c/ espera
0282        else
0283          iwh = 10                               ! Executa s/ espera
0284        end if
0285        call EXEC (iwh, 5hWHZAT, ip(1), ip(2))   ! Despacha WHZAT
0286        write(1,*) ' '                           ! Mensagem
0287        write(1,*) am, '/SMOP: Retornando'       ! Mensagem
0288        goto 5                                   ! Retorna
0289  *
0290  *     ------------------------------------------------------------------
0291  *     Exibe tela de ajuda                                     ! CMD = HE
0292  *     ------------------------------------------------------------------
0293    120 write(1,*) cldsp                         ! Apaga  tela e executa
0294        call EXEC (9+100000B, 6hSMOHP , *999)    ! SMOHP. Se houver erro
0295        goto 5                                   ! processa em 999
0296  *
0297  *     ------------------------------------------------------------------
0298  *     Rebobina a fita magnetica                               ! CMD = RW
0299  *     ------------------------------------------------------------------
0300    130 call EXEC (3, 410B)
0301        goto 5
0302  *
0303  *     ------------------------------------------------------------------
0304  *     Movimentos da fita                                      ! CMD = MF
0305  *     ------------------------------------------------------------------
0306    140 if(usa_fita) then
0307          call segld (6hSMOMF , ier)             ! Carrega o segmento
0308          call ERCHK (ier, 140)                  ! Verifica erro
0309        else
0310          write(1,*) vm,'Impossibilidade logica'
0311        end if
0312        goto 5                                   ! Retorna
0313  *
0314  *     ------------------------------------------------------------------
0315  *     Pausa no programa                                       ! CMD = PA
0316  *     ------------------------------------------------------------------
0317    150 pause                                    ! Pausa
0318        write(1,*) am,'/SMOP: Retornando'        ! Mens. de retorno
0319        goto 5                                   ! Retorna
0320  *
0321  *     ------------------------------------------------------------------
0322  *     Modifica o sinal de pronto (CMD?)                       ! CMD = PP
0323  *     ------------------------------------------------------------------
0324    160 write(1,*) 'Entre c/ o novo sinal'       ! Avisa ao
0325        write(1,*) 'com ate  4 caracteres'       ! usuario e le
0326        read(1,'(a4)', end=5,iostat=ios,err=900) prmpt  ! o novo sinal.
0327        goto 5                                   ! Retorna
0328  *
0329  *     ------------------------------------------------------------------
0330  *     Parametros adicionais para "area fill" (video colorido) ! CMD = AF
0331  *     ------------------------------------------------------------------
0332    170 if(it(1).ne.1.and.it(2).ne.1) then       ! Se os parms
0333          write(1,*) 'Atuais: ',icv(2), icv(3)   ! nao estao no
0334          write(1,*) 'Dither?, Boundary?'        ! comando entao
0335          read(1,*,end=5,iostat=ios,err=900) icv(2),icv(3) ! le (icv).
0336        else                                     ! Se estao no
0337          icv(2) = ip(1)                         ! cmd, ajusta
0338          icv(3) = ip(2)                         ! (icv) com
0339        end if                                   ! parm1 e parm2
0340        goto 5                                   ! Retorna
0341  *
0342  *     ------------------------------------------------------------------
0343  *     Canal ativo do ADC                                      ! CMD = CA
0344  *     ------------------------------------------------------------------
0345    180 if(it(1).ne.1) then                      ! Se o dado nao esta no
0346          write(1,*) 'Canal do ADC? (1,2,3,4)'   ! string de cmd entao
0347          read(1,*,end=5,iostat=ios,err=900) ncha ! le do console
0348        else                                     ! Caso contrario o dado
0349          ncha = ip(1)                           ! e' o parm 1
0350        end if
0351  *     ..................................................................
0352        if(ncha.lt.1.or.ncha.gt.4) then          ! Se o canal nao e'
0353          write(1,*) vm,'Canal Inexistente'      ! 1,2,3,4 exibe mens.
0354          goto 5                                 ! de erro e retorna
0355        end if
0356  *     ..................................................................
0357        nchan = 2**(ncha - 1)                    ! Ajusta p/ formato
0358        goto 5                                   ! do ADC e retorna
```

```
*
*     ------------------------------------------------------------------
*     Frequencia de conversao                                  ! CMD = FC
*     ------------------------------------------------------------------
  190 ifc_save = ifc                                  ! Salva valor corrente
      if(it(1).ne.1) then                             ! Se o dado nao esta'
        write(1,*) 'Frequencia?'                      ! no cmd entao le
        read(1,*,end=5,iostat=ios,err=900) ifc        ! no console
      else                                            ! Caso contrario, o
        ifc = ip(1)                                   ! dado e' o parm 1
      end if                                          ! do string de cmd
*     ..................................................................
      kf = 0                                          ! Reset pointer
      do k = 1,6                                      ! Procura (ifc) na ma
        if(ifc.eq.ifd(k)) kf = k                      ! triz de frequencias
      end do
*     ..................................................................
      if(kf.eq.0) then                                ! Se nao existe na ma
        write(1,*) vm,'Frequencia Inexistente'        ! triz de freqs. entao
        write(1,*) bc,'(5,10,20,50,100 e 200)'        ! o parametro nao e'
        write(1,*) am,'Valor atual: ', ifc_save       ! valido e e' mantido
        ifc = ifc_save                                ! o valor corrente
        goto 5                                        ! Retorna
      else                                            ! Caso contrario,
        ifcm = ifl(kf)                                ! coloca em ifcm o ca
        goto 5                                        ! ractere de controle
      end if                                          ! adequado e retorna
*
*     ------------------------------------------------------------------
*     Interface com o RTE                                      ! CMD = SY
*     ------------------------------------------------------------------
  200 j = messs(iresp(2), 2*ib-2)                     ! Passa a mens. p/ RTE
      if (j.ne.0) then                                ! Se ha resposta (j#0)
         write(1,*) mg,')) _'                         ! entao exibe a respos
         write(1,'(20a2)') (iresp(k), k=2,-j)         ! ta no terminal
      end if
      goto 5                                          ! Retorna
*
*     ------------------------------------------------------------------
*     Move cursor alfa para o topo da tela                     ! CMD = TT
*     ------------------------------------------------------------------
  210 write(1,*) cldsp                                ! Apaga a tela, cursor
      goto 5                                          ! em 0,0 e retorna
*
*     ------------------------------------------------------------------
*     Define limites da varredura                              ! CMD = PV
*     ------------------------------------------------------------------
  220 if(it(1).ne.1.or.it(2).ne.1) then               ! Se os parms nao
        write(1,*) 'Limites da varredura?'            ! sao numericos,
        read(1,*,end=5,iostat=ios,err=900) ipv1, ipv2 ! le no console
      else                                            ! Caso contrario
        ipv1 = ip(1)                                  ! retira os da-
        ipv2 = ip(2)                                  ! dos do string
      end if                                          ! de comando.
*     ..................................................................
      if(ipv1.lt.ipv2) then                           ! Se dados OK
*       call cpr                                      ! atualiza
        goto 5                                        ! e retorna
      else                                            ! Caso contrario
        write(1,*) vm, 'Erro: IPV1 > IPV2'            ! exibe mensagem
        goto 5                                        ! e retorna
      end if
*
*     ------------------------------------------------------------------
*     Define limites da regiao de interesse                    ! CMD = RI
*     ------------------------------------------------------------------
  230 if(it(1).ne.1.or.it(2).ne.1) then               ! Se parms nao
        write(1,*) 'Limites de RI?'                   ! numericos, le
        read(1,*,end=5,iostat=ios,err=900) iri1, iri2 ! no console
      else                                            ! Caso contrario
        iri1 = ip(1)                                  ! retira os dados
        iri2 = ip(2)                                  ! do string de
      end if                                          ! comando,
*     call cpr                                        ! atualiza
      goto 5                                          ! e retorna
*
*     ------------------------------------------------------------------
*     Define numero de medidas                                 ! CMD = NM
*     ------------------------------------------------------------------
  240 if(it(1).ne.1.or.it(2).ne.1) then               ! Se os parametros nao
        write(1,*) 'No. medidas RN ?'                 ! estao no string de
        read (1,*)  nmrn                              ! comando entao le no
        write(1,*) 'No. medidas RI ?'                 ! terminal do
        read (1,*)  nmri                              ! operador.
      else                                            ! Caso contrario, os
        nmrn = ip(1)                                  ! parametros estao em
        nmri = ip(2)                                  ! ip(1) e ip(2)
      end if
*     call cpr                                        ! Atualiza
      goto 5                                          ! e retorna
*
*     ------------------------------------------------------------------
*     Mede off-set                                             ! CMD = OS
*     ------------------------------------------------------------------
  250 call segld (6hSMOVR , ier)
      call erchk (ier, 250)
      goto 5
*
*     ------------------------------------------------------------------
*     Executa uma conversao                                    ! CMD = AD
*     ------------------------------------------------------------------
  260 call segld( 6hSMOVR , ier)
      call erchk (ier, 260)
      goto 5
*
  270 continue
*     ------------------------------------------------------------------
*     Vai para o inicio da varredura (ipv1)                    ! CMD = VI
*     ------------------------------------------------------------------
  280 np1 = iqnp(ipv1)
      np = np1 - npa
      idir = 1
      if(np.lt.0) then
        np = -np
        idir = -1
      end if
      call steps(npa, np, idir, nnp, lu_mpg, ier)
      npa = nnp
      goto 5
*
```

```
*     ------------------------------------------------------------------
*     Vai para o fim da varredura (ipv2)                    ! CMD = VF
*     ------------------------------------------------------------------
  290 np2 = iqnp(ipv2)                                  ! Total de passos p/ VF
      np  = np2 - npa                                   ! Passos necessarios
      idir= 1                                           ! Sentido horario
      if(np.lt.0) then                                  ! Se np e' negativo o
        np   = - np                                     ! mov. e' anti-horario
        idir = -1                                       ! Ajusta np e idir
      end if
      call steps(npa,np,idir,nnp,lu_mpg,ier)            ! Movimenta motor
      npa = nnp                                         ! Ajusta (npa)
      goto 5
*
*     ------------------------------------------------------------------
*     Vai para o inicio do curso                            ! CMD = CZ
*     ------------------------------------------------------------------
  300 np  =  npa
      idir= -1
      call steps(npa,np,idir,nnp,lu_mpg,ier)
      npa = nnp
      goto 5
*
*     ------------------------------------------------------------------
*     Vai para o fim do curso                               ! CMD = CT
*     ------------------------------------------------------------------
  310 np  = 2000 - npa
      idir = 1
      call steps(npa,np,idir,nnp,lu_mpg,ier)
      npa = nnp
      goto 5
*
*     ------------------------------------------------------------------
*     Libera o motor de passo                               ! CMD = LM
*     ------------------------------------------------------------------
                                                        ! Manda controle para
  320 call Cntrl_MPG(Lu_MPG, 20B)                       ! o multiprogramador
      mot_lib = .true.                                  ! Inibe logica motor
      write(1,*) '---------------'                      ! Avisa no console
      write(1,*) 'Motor  liberado'
      write(1,*) 'no passo: ', npa
      write(1,*) '---------------'
      goto 5
*
*     ------------------------------------------------------------------
  330 continue
  340 continue
  350 continue
  360 continue
*
*     ------------------------------------------------------------------
*     Exibe dados plotter HP                                ! CMD = PL
*     ------------------------------------------------------------------
  370 call segld(6hSMOVR ,ier)
      call erchk (ier, 370)
      goto 5
*
*     ------------------------------------------------------------------
*     Exibe fundo de escala do ADC                          ! CMD = FE
*     ------------------------------------------------------------------
  380 call segld (6hSMOVR , ier)
      call erchk (ier, 380)
      goto 5
*
*     ------------------------------------------------------------------
*     Movimentos do goniometro motorizado                   ! CMD = GN
*     ------------------------------------------------------------------
  390 call segld (6hSMOVR , ier)
      call erchk (ier, 390)
      goto 5
*
*     ------------------------------------------------------------------
*     Cria diretorio  (CI)                                  ! CMD = CD
*     ------------------------------------------------------------------
  400 call segld(6hSMOCI ,ier)
      call erchk (ier, 400)
      goto 5
*
*     ------------------------------------------------------------------
*     Exibe diretorio corrente                              ! CMD = WD
*     ------------------------------------------------------------------
  410 call segld(6hSMOCI ,ier)
      call erchk (ier, 410)
      goto 5
*
*     ------------------------------------------------------------------
*     Exibe dono do diretorio                               ! CMD = OW
*     ------------------------------------------------------------------
  420 call segld(6hSMOCI ,ier)
      call erchk (ier, 420)
      goto 5
*
*     ------------------------------------------------------------------
*     Inicializa motor de passo com a configuracao 0110 (6b)! CMD = IM
*     ------------------------------------------------------------------
  430 in = 6b                                           ! Config: 0110 (6b)
      call InMot(Lu_MPG, in)                            ! Inicializa
      mot_lib = .false.                                 ! Ajusta flag
      goto 5                                            ! retorna
*
*     ------------------------------------------------------------------
*     Posicao corrente do motor                             ! CMD = PC
*     ------------------------------------------------------------------
  440 write(1,*) 'Posicao atual:'                       ! Exibe:
      write(1,*) 'Passo: ', npa                         ! No. do passo atual
      write(1,*) '% Range: ', ipor(npa)                 ! % of range atual
      if(mot_lib) then                                  ! Se o motor esta' li-
        write(1,*) vm,'Motor Liberado'                  ! berado exibe um avi-
      end if                                            ! so de advertencia e
      goto 5                                            ! retorna
*
*     ------------------------------------------------------------------
*     Faz leitura do campo via gaussimetro                  ! CMD = CM
*     ------------------------------------------------------------------
  450 if(lu_gss.eq.lu_gsv) then

        write(1,*) 'Gaussimetro Varian'
        read(lu_gss,*) campo
        write(1,'("Campo: ",f8.2)') campo
```

```
      else if(lu_gss.eq.lu_gsm) then

        write(1,*) 'Gaussimetro Metrolab'
        write(1,*) 'Nao implementado'

      end if
      goto 5
*     ------------------------------------------------------------------
*     Valor do atraso entre passos                              ! CMD = AT
*     ------------------------------------------------------------------
  460 write(1,*) 'Valor do atraso? [em milisegundos]'
      read(1,*) idel
      goto 5
*     ------------------------------------------------------------------
*     Erros de I/O
*     ------------------------------------------------------------------
  900 write(1,*) 'Erro de I/O - iostat: ',ios     ! Os erros (iostat) sao
      goto 5                                      ! descritos no HP RTE
*     ------------------------------------------------------------------
*     Erros de "schedule"
*     ------------------------------------------------------------------
  999 call abreg(ia,ib)                           ! Esses erros aparecem
      write(1,'("Erro: ",a2,a2)') ia, ib          ! qdo. ha algum proble
      goto 5                                      ! ma no despacho
*     ------------------------------------------------------------------

      end
*     ------------------------------------------------------------------
      subroutine erchk(ier,iblc)
*     ------------------------------------------------------------------
*
*     Esta rotina verifica e informa se houve erro na execucao de
*     um segmento (overlay) do programa principal
*          ier  = Codigo devolvido pela rotina segld
*          iblc = Label do bloco que originou a chamada
*
*     ------------------------------------------------------------------
      if(ier.eq.5) then
      write(1,*) 'Erro segld: ',ier
      write(1,*) 'CMD> Label: ',iblc
      end if
      return
      end
*     ------------------------------------------------------------------
      function iqnp (ipor)
*     ------------------------------------------------------------------
*
*     Esta funcao devolve o numero de passos necessarios para levar o
*     sistema a  um dado valor de  "percent of range".  Admite-se que
*     o helipot tenha sido levado a posicao CCW no inicio do programa.
*
      iqnp = 20 * ipor + 1000
      return
      end
*     ------------------------------------------------------------------
```

```
      function ipor (iqnp)
*     ------------------------------------------------------------------
*
*     Esta funcao devolve o valor do "percent of range" correspondente
*     a um dado numero de passos executados pelo motor. Admite-se  que
*     o helipot tenha sido levado a posicao CCW no inicio do programa.
*
      ipor = (iqnp - 1000)/20
      return
      end
*     ------------------------------------------------------------------
      subroutine InMot(Lu_MPG, in)
*     ------------------------------------------------------------------
*
*     Esta rotina inicializa o motor de passo na configuracao (in)
*     onde (in) e' um octal igual a 6b, 5b,11b ou 12b. Outras con-
*     figuracoes nao sao validas para motor de 4 fases e podem le-
*     var a resultador imprevisiveis. A rotina NAO testa a valida-
*     de da configuracao. LuMPG e' a unidade logica do Multiprog.
*
      write(Lu_MPG,'("O60T")')                    ! Inic. Multiprog.
                                                  ! Transmite (in)
      write(Lu_MPG,'("L",@2,"T")') in             ! p/ porta paralela
      return                                      ! e retorna
      end
*     ------------------------------------------------------------------
      subroutine Cntrl_MPG(LuMPG, icnwd)
*     ------------------------------------------------------------------
*
*     Esta rotina manda um controle para o multiprogramador para inibir
*     ou habilitar as saidas de todos os cartoes instalados.
*
*     A palavra de controle ICNWD ajusta os bits  HMT e HSS  onde:
*
*     HMT = 1 > Habilita o modo temporizado, onde o MPG  so aceita
*               um novo comando apos executar a tarefa em curso.
*
*     HSS = 1 > Habilita simultaneamente a saida de todoss cartoes
*               de saida instalados no MPG
*
*         ICNWD = 0    >>  HMT=0 e HSS=0
*         ICNWD = 20B  >>  HMT=1 e HSS=0
*         ICNWD = 40B  >>  HMT=0 e HSS=1
*         ICNWD = 60B  >>  HMT=1 e HSS=1
*     ------------------------------------------------------------------
      write(LuMPG,'("O",@2,"T")') icnwd
      return
      end
*     ------------------------------------------------------------------


      end$
```

```
&SMOIN T=00004 IS ON CR00002 USING 00004 BLKS R=0000

0001  ftn7x,l
0002        program SMOIN, 5
0003  *     ---------------------------------------------------------------
0004  *     COMMONS:
0005  *
0006        common  /ulogs/   lu(10)                    ! Unidades Logicas LUs
0007        common  /vlogs/   logic (10)                ! Variaveis logicas
0008        common  /parm1/   iclas, ifcm, npa, idel    ! Parametros (1)
0009        common  /parm2/   nm_gss                    ! Parametros (2)
0010  *     ---------------------------------------------------------------
0011  *     CHARACTERs:
0012  *
0013        character*1   esc                           ! escape
0014        character*5   cldsp                         ! clear display (HP)
0015        character*5   vm, vd, am, az                ! atributos de cor (HP)
0016        character*16  klend                         ! string calendario
0017  *     ---------------------------------------------------------------
0018  *     LOGICALs:
0019  *
0020        logical       sim, logic                    ! teste de resposta
0021        logical       usa_fita, usa_dmm, usa_fmo
0022        logical       usa_gss,  usa_mpg, usa_gon
0023  *     ---------------------------------------------------------------
0024  *     EQUIVALENCES:
0025  *
0026  *     1) Equipamentos HP-IB
0027  *        ------------------
0028        equivalence (lu_plt,lu(1)), (lu_adc,lu(2)), (lu_dmm,lu(3)),
0029       .            (lu_fmo,lu(4)), (lu_gsm,lu(5)), (lu_mpg,lu(6)),
0030       .            (lu_gon,lu(7)), (lu_gsv,lu(8)), (lu_gss,lu(9))
0031  *
0032  *     2) Variaveis Logicas
0033  *        -----------------
0034        equivalence (usa_fita,logic(1)), (usa_dmm,logic(2)),
0035       .            (usa_fmo, logic(3)), (usa_gss,logic(4)),
0036       .            (usa_mpg, logic(4)), (usa_gon,logic(5))
0037  *     ---------------------------------------------------------------
0038        esc   = char(27)                            ! Escape code
0039        cldsp = esc//'h'//esc//'J_'                 ! Clear display (HP)
0040        vd    = esc//'&v2S'                         ! atr. de cor -Verde
0041        am    = esc//'&v3S'                         ! atr. de cor -Amarelo
0042  *     ---------------------------------------------------------------
0043        write(1,*) esc//'&v1a1z4I'                  ! Reprog. cor 4 (az/vm)
0044        call clend (klend)                          ! hora corrente
0045        write(1,*) cldsp
0046        write(1,*)vd,'-------------------------------------------------------'
0047        write(1,*)am,'SMOP - Sistema do Motor de Passo'
0048        write(1,*)am,'Laboratorio de Ressonancia Magnetica'
0049        write(1,*)am, klend
0050        write(1,*)vd,'-------------------------------------------------------'
0051  *     ---------------------------------------------------------------
0052        lu_plt = 35                     ! Lu do plotter HP          HPIB 05
0053        lu_adc = 36                     ! LU do ADC                 HPIB 06
0054        lu_dmm = 42                     ! LU do DMM Keythley        HPIB 12
0055        lu_fmo = 43                     ! LU do Freq. M. Onda       HPIB 13
0056        lu_gsm = 41                     ! LU do Gauss. Metrolab     HPIB 11
0057        lu_mpg = 44                     ! LU do Multiprogramador    HPIB 14
0058        lu_gon = 39                     ! LU do Goniometro          HPIB 09
0059        lu_gsv = 37                     ! LU do Gauss. Varian       HPIB 07
0060  *     ---------------------------------------------------------------
0061  *     OBTENCAO DO NUMERO DE CLASSE:
0062  *
0063        ifunc  = 1 + 100000B                        ! Controle Class Numb
0064        iclas  = 0                                  ! reset class numb
0065        call CLRQ(ifunc, iclas)                     ! pede um num de classe
0066        call ABREG(ia,ib)                           ! Le registros A e B
0067        if(ia.eq.-1) then                           ! Se nao ha numero dis-
0068          write(1,*) vm, 'Nao ha  numero de'        ! ponivel, entao encer
0069          write(1,*) vm, 'classe disponivel'        ! ra o programa. Caso
0070          stop 77                                   ! contrario, isola no
0071        else                                        ! de classe,
0072          jclas = iand(iclas,377B)                  ! avisa no console
0073          write(1,*) am,'/SMOP'                     ! e prossegue
0074          write(1,*) am,' No. de classe: ',jclas
0075        end if
0076  *     ---------------------------------------------------------------
0077        write(1,*) 'Fita Magnetica? ... [si/na]'
0078        call input(iresp,1)
0079        if(sim(iresp)) usa_fita = .true.
0080
0081        write(1,*) 'Multimetro Digital? [si/na]'
0082        call input(iresp,1)
0083        if(sim(iresp)) usa_dmm = .true.
0084
0085        write(1,*) 'Freq. Micro Onda?   [si/na]'
0086        call input(iresp,1)
0087        if(sim(iresp)) usa_fmo = .true.
0088
0089        write(1,*) 'Goniometro? ..... [si/na]'
0090        call input(iresp,1)
0091        if(sim(iresp)) usa_gon = .true.
0092
0093        write(1,*) 'Gaussimetro? ... [Va/Me/na]'
0094        read(1,'(a2)') iresp
0095        call clcuc(iresp,1)
0096        if(iresp.eq.2hVA) then
0097          lu_gss  = lu_gsv
0098          usa_gss = .true.
0099          nm_gss  = iresp
0100        else if(iresp.eq.2hME) then
0101          lu_gss  = lu_gsm
0102          usa_gss = .true.
0103          nm_gss  = iresp
0104        end if
0105  *     ---------------------------------------------------------------
0106                                        ! Chama a atencao do operador
0107                                        ! para a inicializacao do motor
0108
0109        write(1,*) am,'Verifique se o motor de passo'
0110        write(1,*) am,'esta no  inicio  de seu curso'
0111        pause
0112        iresp = 2hNA
0113        do while(iresp.ne.2hSI)
0114        write(1,*) vm,'Voce tem certeza? [si/na]'
0115        read(1,'(a2)') iresp
0116        call clcuc(iresp,1)
0117        end do
0118
```

```
0119        in = 6b                                    ! Inicializa o motor
0120        call InMot(LuMPG, in)                      ! na config. 0110
0121  *     ------------------------------------------------------------------
0122        call segrt
0123        end
0124  *     ------------------------------------------------------------------
0125        subroutine input (iresp,nw)
0126        iresp = 0
0127        do while(iresp.ne.2hSI.and.iresp.ne.2hNA)
0128           call reio(1,401b,iresp,nw)
0129           call clcuc(iresp,nw)
0130        end do
0131        return
0132        end
0133  *     ------------------------------------------------------------------
0134        end$
```

```
$SMOVR T=00004 IS ON CR00026 USING 00012 BLKS R=0000

0001  ftn7x,l
0002  *
0003  $ema  /xy/
0004  *     ------------------------------------------------------------------
0005        program smovr,5                            ! (890601.0915)
0006                                                   ! )890523.1627(
0007  *     ------------------------------------------------------------------
0008        common /xy/       x(2001), y(2001)         ! Dados
0009        common /cmdpr/    ip(8), it(7)             ! Parametros de comando
0010        common /comdo/    icmd                     ! Comando
0011        common /ulogs/    lu(10)                   ! Unidades Logicas
0012        common /vlogs/    logic (10)               ! Variaveis Logicas
0013
0014  *     ------------------------------------------------------------------
0015        real*4     ofset (100)                     ! Dados de off set
0016        real*4     scadc (4)                       ! Escala ADC (mV/count)
0017
0018        integer*2 mcmd1 (10)                       ! matriz de comandos
0019        integer*2 nomsp (40)                       ! nome de espectro(PL)
0020        integer*2 ivost (100)                      ! medidas p/ off set
0021        integer*2 ifcos (1)                        ! Freq. ADC (100 Hz)
0022  *     ------------------------------------------------------------------
0023        logical   motmov
0024  *     ------------------------------------------------------------------
0025  *
0026        equivalence (lu(1),lu_plt), (lu(2),lu_adc), (lu(3),lu_dmm),
0027       .            (lu(4),lu_fmo), (lu(5),lu_gsm), (lu(6),lu_mpg),
0028       .            (lu(7),lu_gon)
0029
0030  *     ------------------------------------------------------------------
0031        data mcmd1  /2hFE,2h0S,2hPL,2hGN,2hAD,2h  ,2h  ,2h  ,2h  ,2h  /
0032        data scadc  /1.0, 2.5, 5.0, 10.0/
0033        data ifcos  /103B/
0034  *     ------------------------------------------------------------------
0035        kf1  = 0                                   ! Reset kf1
0036        do j = 1,10                                ! Procura comando na
0037           if(icmd.eq.mcmd1(j)) kf1 = j            ! matriz MCMD1 e colo
0038        end do                                     ! ca o numero em kf1
0039        if(kf1.eq.0) then                          ! Se kf1 continua nula
0040          write(1,*) 'SMAVR: Comando inexistente'  ! entao e' porque o co
0041          goto 5                                   ! mando nao existe.
0042        else                                       ! Caso contrario, vai
0043          goto (10, 20, 30, 40, 50) kf1            ! executa-lo
0044        end if
0045  *
0046  *     ------------------------------------------------------------------
0047  *     Informe sobre fundo de escala do ADC                    ! CMD = FE
0048  *     ------------------------------------------------------------------
0049
0050     10 write(1,*) '================================================='
0051        write(1,*) 'O ADC  foi inicialmente ajustado da seguinte maneira:'
0052        write(1,*) ' '
0053        write(1,*) 'Canal 1 :  2,000 volts pico a pico -  1.0 mV/contagem'
0054        write(1,*) 'Canal 2 :  5,000 volts pico a pico -  2.5 mV/contagem'
0055        write(1,*) 'Canal 3 :  10,00 volts pico a pico -  5.0 mV/contagem'
0056        write(1,*) 'Canal 4 :  20,00 volts pico a pico - 10.0 mV/contagem'
0057        write(1,*) '-------------------------------------------------'
0058        write(1,*) 'Para  mudar estes valores, consulte antes o manual do'
```

```
0059        write(1,*) 'instrumento  para  obter  as informacoes  necessarias'
0060        write(1,*) '=========================================================='
0061        goto 5
0062  *
0063  *     ------------------------------------------------------------------
0064  *     Mede Off-Set                                              ! CMD = OS
0065  *     ------------------------------------------------------------------
0066     20 if(it(1).ne.1) then                                ! Verifica parametros
0067          write(1,*) ' Canal do ADC? (1,2,3,4)'
0068          read (1,*,end=5,iostat=ios,err=900) nadc
0069        else
0070          nadc = ip(1)
0071        end if
0072  *
0073        if(nadc.lt.1.or.nadc.gt.4) then
0074          write(1,*) 'Canal Inexistente'
0075          goto 5
0076        end if
0077  *
0078        nadcb= 2**(nadc -1)                              ! Programa e dispara o
0079        write(lu_adc,'("HL",r1,i1,"J_")') ifcos, nadcb ! relogio do ADC
0080        call REIO (1, 100B+36, ivost, 100)               ! Toma 100 medidas
0081        write(lu_adc,'("I_")')                           ! Para o relogio do ADC
0082        do j = 1,100                                     ! Calcula off-set em
0083        ofset(j) = ivost(j) * scadc(nadc)                ! milivolts
0084        end do
0085        write(1,'(10(1x,f6.3))') (ofset(j), j=1,100)
0086  *
0087        soma = 0.
0088  *     do j = 1,100
0089  *     soma = soma + ofset(j)
0090  *     end do
0091  *     avg = soma/100.
0092        call sig2(ofset,100,avg,dp,vr)
0093        write(1,*) '----------------------------'
0094        write(1,*) '          Canal: ', nadc
0095        write(1,*) '       Off-Set: ', avg
0096        write(1,*) '       Sigma^2: ', vr
0097        write(1,*) '----------------------------'
0098        goto 5
0099  *     ------------------------------------------------------------------
0100  *     Exibe espectro - Plotter HP - Interface HPIB                CMD = PL
0101  *     ------------------------------------------------------------------
0102     30 write(1,*) 'Nome do grafico?'
0103        read(1,'(40a2)',end=5) (nomsp(k),k=1,40)
0104        write(1,*) 'Xtic ? _'
0105        read(1,*,end=5,iostat=ios,err=900) xt
0106        call Hplot(x,y,np,1,7,nomsp,xt)
0107        goto 5
0108  *
0109  *     ------------------------------------------------------------------
0110  *     Controle do goniometro motorizado                           CMD = GN
0111  *     ------------------------------------------------------------------
0112     40 if(ip(1).eq.2hRS) then                        ! Reset Axis 1
0113          write(lu_gon,'("C10")')
0114          goto 5
0115        end if
0116  *     ------------------------------------------------------------------
0117        if(ip(1).eq.2hMO) then                           ! Movimenta motor
0118          if((it(2)*it(3)).ne.1) then
0119            write(1,*) 'Parametros invalidos'
0120            goto 5
0121          else
0122            iang = 100*iabs(ip(2) + ip(3))
0123            if(ip(2).lt.0) iang = -iang
0124            if(iabs(iang).gt.9000) then
0125              write(1,*) 'Angulo invalido'
0126              goto 5
0127            end if
0128            if(motmov(lu_gon)) then
0129              write(1,*) 'Motor em movimento'
0130              goto 5
0131            else
0132  *           write(lu_gon,'(SP,"I1=",i5.5,"!",SS)') iang
0133              call move_mot(lu_gon, iang)
0134              goto 5
0135            end if
0136          end if
0137        end if
0138  *     ------------------------------------------------------------------
0139        if(ip(1).eq.2hDP) then                          ! Define Passo
0140          if((it(2)*it(3)).ne.1) then
0141            write(1,*) 'Parametros invalidos'
0142            goto 5
0143          end if
0144          ignstp = 100*iabs(ip(2) + ip(3))
0145          if(ip(2).lt.0) ignstp = -ignstp
0146          goto 5
0147        end if
0148  *     ------------------------------------------------------------------
0149        if(ip(1).eq.2hPA) then                          ! Posicao Atual
0150          call ipos(lu_gon, ipa)
0151          pa = ipa/100.
0152          write(1,*) 'Pos. atual: ', pa
0153        goto 5
0154        end if
0155  *     ------------------------------------------------------------------
0156        if(ip(1).eq.2h)) then
0157          call ipos(lu_gon, ipa)
0158          iang = ipa + ignstp
0159          call move_mot(lu_gon, iang)
0160          goto 5
0161        end if
0162  *     ------------------------------------------------------------------
0163        if(ip(1).eq.2h(( then
0164          call ipos(lu_gon, ipa)
0165          iang = ipa - ignstp
0166          call move_mot(lu_gon, iang)
0167          goto 5
0168        end if
0169  *     ------------------------------------------------------------------
0170        write(1,*) 'Parametro invalido'
0171        goto 5
0172  *
0173  *     ------------------------------------------------------------------
0174  *     Executa uma conversao do ADC                                ! CMD = AD
0175  *     ------------------------------------------------------------------
0176     50 if(it(1).ne.1) then
0177          write(1,*) 'Canal do ADC? [1,2,3,4]'
0178          read(1,*,end=5,iostat=ios,err=900) nadc
```

```
0179        else
0180          nadc = ip(1)
0181        end if
0182        if(nadc.lt.1.or.nadc.gt.4) then
0183          write(1,*) 'Canal Inexistente'
0184          goto 5
0185        else
0186          nadcb = 2**(nadc-1)
0187          write(lu_adc,'("H",i1,"ALJ_")') nadcb
0188          call reio(1, 100B+lu_adc, ivad, 1)
0189          write(lu_adc,'("I_")')
0190          volts = ivad*scadc(nadc)/1000.
0191          write(1,*) '      ------------------'
0192          write(1,*) '         Canal: ', nadc
0193          write(1,*) '         Volts: ', volts
0194          write(1,*) '       Leitura: ', ivad
0195          write(1,*) '      ------------------'
0196        end if
0197        goto 5
0198  *
0199  *     ------------------------------------------------------------------
0200  *     Erros de entrada e saida
0201  *     ------------------------------------------------------------------
0202    900 write(1,*) 'SMAVR: Erro I/O: ',ios
0203  *     ------------------------------------------------------------------
0204  *     Retorno ao programa principal
0205  *     ------------------------------------------------------------------
0206      5 call SEGRT
0207        end
0208  *     ------------------------------------------------------------------
0209        logical function motmov(lu)
0210  *     ------------------------------------------------------------------
0211        character ar*4
0212        write(lu, '("I1?")')
0213        read (lu, '(4a)') ar
0214        if(ar.ne.'AR1 ') then
0215          motmov = .true.
0216        else
0217          motmov = .false.
0218        end if
0219        return
0220        end
0221  *     ------------------------------------------------------------------
0222        subroutine move_mot(lu, iang)
0223  *     ------------------------------------------------------------------
0224        write(lu,'(SP,"I1=",i5.5,"!",SS)') iang
0225        return
0226        end
0227  *     ------------------------------------------------------------------
0228        subroutine ipos(lu, ipa )
0229  *     ------------------------------------------------------------------
0230        write(lu,'("C1?")')
0231        read (lu,'(3x,i7,1x)') ipa
0232        return
0233        end
0234  *     ------------------------------------------------------------------
```

```
&SMOCI T=00004 IS ON CR00026 USING 00006 BLKS R=0000

0001  ftn7x,1
0002        program smoci,5                                  ! (890524.2053)
0003                                                        ! )890524.1550(
0004  *     ------------------------------------------------------------------
0005        character ndir*63
0006        character nome*30
0007  *     ------------------------------------------------------------------
0008        common /comdo/  icmd
0009  *     ------------------------------------------------------------------
0010        if(icmd.eq.2hCD) then
0011
0012          write(1,*) 'Nome do diretorio a criar?'
0013          read(1,'(a63)') ndir
0014          write(1,*) 'Em qual LU?  [0 para default]'
0015          read(1,*) lu
0016          ier = FmpCreateDir(ndir, lu)
0017          call ervrf(ier, ndir)
0018          goto 5
0019
0020        else if(icmd.eq.2hOW) then
0021
0022          write(1,*) 'Nome do diretorio?'
0023          read(1,'(a63)') ndir
0024          ier = FmpOwner(ndir, nome)
0025          write(1,*) 'Dono do diretorio: ', nome
0026          call ervrf(ier, ndir)
0027          goto 5
0028
0029        else if(icmd.eq.2hWD) then
0030
0031          ier = FmpWorkingDir(ndir)
0032          if(ier.eq.0) then
0033            write(1,*) 'Dir: ',ndir
0034            goto 5
0035          end if
0036          call ervrf(ier, ndir)
0037          goto 5
0038
0039        end if
0040  *     ------------------------------------------------------------------
0041      5 call segrt
0042        end
0043  *
0044  *     ------------------------------------------------------------------
0045        subroutine ervrf(ier,nome)
0046        character nome*(*)
0047        if(ier.lt.0) then
0048        call FmpReportError(ier,nome)
0049        end if
0050        return
0051        end
0052  *     ------------------------------------------------------------------
0053        end$
```

```
&SMOAQ T=00004 IS ON CR00026 USING 00002 BLKS R=0000

0001  ftn7x,l
0002  *
0003  $ema  /xy/
0004  *
0005        program smoaq,5                                  ! (890816.1119)
0006  *     ---------------------------------------------------------------
0007        common /xy/     x(2001), y(2001)
0008  *     ---------------------------------------------------------------
0009        common /cmdpr/  ip(8), it(7)
0010        common /comdo/  icmd
0011        common /ulogs/  lu(10)
0012        common /vlogs/  logic(10)
0013        common /data1/  kbuf(4200)
0014        common /parm1/  iclas, ifcm, npa, idel
0015        common /parm2/  nm_gss
0016  *     ---------------------------------------------------------------
0017        integer*2       iabuf(1024)                 ! Buffer de aquisicao
0018        integer*2       idat(200)                   ! Dados
0019  *     ---------------------------------------------------------------
0020        logical         logic, cond1, cond2
0021  *     ---------------------------------------------------------------
0022        equivalence (idat(1), kbuf(4001))
0023
0024        equivalence (idat(1),  ifc), (idat(2),   no1), (idat(3), ipv1),
0025       .            (idat(4), ipv2), (idat(5), nchan), (idat(6), iri1),
0026       .            (idat(7), iri2), (idat(8), nspec), (idat(9), ipnd),
0027       .            (idat(10),  np), (idat(11), nmrn), (idat(12),nmri),
0028       .            (idat(13), no2), (idat(14),  no3), (idat(15),  b1),
0029       .            (idat(17),  b2), (idat(19),  no4), (idat(20), no5)
0030  *     ...............................................................
0031        equivalence (lu(1), lu_plt), (lu(2), lu_adc), (lu(3), lu_dmm),
0032       .            (lu(4), lu_fmo), (lu(5), lu_gsm), (lu(6), lu_mpg),
0033       .            (lu(7), lu_gon), (lu(8), lu_gsv), (lu(9), lu_gss)
0034
0035  *     ...............................................................
0036        equivalence (usa_fita,logic(1)), (usa_dmm,logic(2)),
0037       .            (usa_fmo, logic(3)), (usa_gss,logic(4)),
0038       .            (usa_mpg, logic(5)), (usa_gon,logic(6))
0039  *     ---------------------------------------------------------------
0040
0041
0042        if(ipnd.ne.0) then                          ! Verifica a existencia
0043          write(1,*) 'Dados pendentes na'           ! de espectro  pendente
0044          write(1,*) 'memoria.  Salve-os'           ! na memoria. Se existe,
0045          write(1,*) 'ou ignore-os! (IG)'           ! informa ao usuario e
0046          goto 5                                    ! retorna ao programa
0047        end if                                      ! principal.
0048  *
0049        if(ipor(npa).ne.ipv1) then                  ! Se o motor nao esta'
0050        write(1,*) 'Posicionando Motor..._'         ! no inicio da varre-
0051          np   = iqnp(ipv1) - npa                   ! dura, calcula a di-
0052          idir = 1                                  ! ferenca. Sentido CW
0053          if(np.lt.0) then                          ! Se esta' acima, entao
0054            idir = -1                               ! escolhe sentido CCW
0055            np   = -np                              ! ajusta o numero de
0056          end if                                    ! passos p/ motor.
0057          call steps(npa,np,idir,nnp,lu_mpg,ier)    ! Movimenta motor.
0058          write(1,*) 'Ok'                           ! Sinaliza.
0059        end if
0060        idir = 1                                    ! Retorna sentido CW
0061  *     ---------------------------------------------------------------
0062                                                    ! Programa o ADC com
0063                                                    ! a freq. e o canal.
0064
0065        write(lu_adc,'("H",r1,i1,"L_")') ifcm, nchan
0066  *     ---------------------------------------------------------------
0067        call cnfg(lu_adc,1,37000b)                  ! Configura DMA
0068        call exec(22, 1)                            ! Liga "Memory Lock"
0069  *     ---------------------------------------------------------------
0070        lu_bin = lu_adc+100b
0071                                                    ! Obtem valor do campo
0072        call field (b1, usa_gss, lu_gss, mn_gss)    ! no primeiro ponto
0073
0074        npaq = 1
0075        do while (npaq.le.np)
0076        naq    = nmrn                               ! Ajusta  o  numero  de
0077        cond1  = ipor(npa).ge.iri1                  ! aquisicoes de acordo
0078        cond2  = ipor(npa).le.iri2                  ! com a regiao em que o
0079        if(cond1.and.cond2) naq = nmri              ! sistema  se  encontra
0080
0081        call exec(17,lu_bin,iabuf,naq,0,0,iclas)    ! Inicia leitura Class
0082  *     ...
0083  *     ...
0084        call exec(21,iclas+40000B,iabuf,0)          ! Aguarda fim de Class
0085        if(npaq.eq.np) then                         ! Se esta' no ultimo
0086          call field (b2, usa_gss, lu_gss, nm_gss)  ! ponto, entao le o va
0087        end if                                      ! lor do campo
0088        call steps(npa,1,idir,nnp,lu_mpg,ier)       ! Da' um passo no motor
0089        npa = nnp                                   ! Ajusta npa
0090  *                                                 ! Recupera dados do buf
0091        call exec(21,iclas+20000b,iabuf,naq)        ! mas nao libera classe
0092                                                    ! Calcula a
0093        call isig2(iabuf,naq,rm,dp,vr)              ! soma das
0094        y(npaq) = rm                                ! leituras
0095                                                    ! Calc. media aritmetica
0096        npaq  =  npaq + 1                           ! Atualiza ponto
0097        if(idel.ne.0) then                          ! Se preciso,
0098          call atraso (idel)                        ! chama rotina
0099        end if                                      ! de atraso
0100        end do
0101  *     ---------------------------------------------------------------
0102      5 call SegRt                                  ! Return to main
0103        end
0104  *     ---------------------------------------------------------------
0105        subroutine isig2 (ix, np, rmed, dp, vr)
0106        dimension  ix(1)
0107  *     ---------------------------------------------------------------
0108        soma = 0.                                   ! Media aritmetica
0109        do j = 1, np
0110        soma = soma + ix(j)
0111        end do
0112        rmed = soma/np
0113  *     ---------------------------------------------------------------
0114        soma = 0.                                   ! Desvio padrao
0115        do j = 1, np
0116        soma = soma + (ix(j)-rmed)*(ix(j)-rmed)
0117        end do
0118        dp    = sqrt(soma/np)
```

```
0119  *     ------------------------------------------------------------------
0120        vr = dp*dp                                    ! Varianca
0121        return
0122  *     ------------------------------------------------------------------
0123        end
0124  *     ------------------------------------------------------------------
0125        subroutine atraso (idel)
0126  *     ------------------------------------------------------------------
0127        iresl = 1                                     ! Dezenas de miliseg
0128        call exec(12,0,iresl,0,-idel)                 ! Time schedule
0129        return
0130        end
0131  *     ------------------------------------------------------------------
0132        subroutine field (b, usa_gss, lu_gss, nm_gss)
0133  *     ------------------------------------------------------------------
0134        logical    usa_gss, c1, c2, c3
0135  *
0136        c1 = .not.usa_gss                             ! Nao usa gaussimetro
0137        c2 = nm_gss.eq.2hVA                           ! Gaussimetro Varian
0138        c3 = nm_gss.eq.2hME                           ! Gaussimetro Metrolab
0139  *
0140        if(c1) then
0141          write(1,*) '0 uso do gaussimetro nao foi previsto!'
0142          write(1,*) 'Entre com o valor do campo: _'
0143          read(1,*) b
0144          return
0145        else if(c2) then                              ! Varian
0146          read(lu_gss) b
0147          return
0148        else if(c3) then                              ! Metrolab
0149          read(lu_gss) b
0150          return
0151        end if
0152        end
0153  *     ------------------------------------------------------------------
0154        end$
```

Apêndice 4

Características do computador HP 1000

O nome "Computador HP 1000" é a denominação genérica de uma família de mini-computadores de tempo real produzidos pela Hewlett Packard Company, cuja configuração final pode ser escolhida pelo usuário de acordo com as suas necessidades e/ou disponibilidades orçamentárias. Dadas as suas características de "tempo real", estas máquinas encontram suas principais aplicações nos sistemas de aquisição de dados e controle de processos em laboratórios de pesquisa e unidades industriais em geral. Neste trabalho, usamos um computador HP 1000 - Série F em uma configuração cujas características principais estão resumidas a seguir.

CPU do sistema:

Modelo:	HP 2117F
Memória:	1 Megabytes
Palavra:	16 bits
DMA:	2 canais
Velocidade:	1 Mips (instruções básicas)
Placas de E/S:	14, máximo

Coprocessadores instalados:

FFP:	Fast Fortran Processor
FPP:	Floating Point Processr
VIS:	Vector Instruction Set

Periféricos principais:

Disco:	132 Megabytes
Fita 1:	1600 bpi, compatível com IBM

Fita 2:	Streammer, 60 Mb por cartucho
MUX:	8 linhas RS 232C
HP-IB:	14 instrumentos, norma IEEE 488
Impressora:	300 lpm, gráfica
Terminal 1:	Gráfico, a cores - 512H X 390V
Terminal 2:	Gráfico, a cores - 600H X 400V
Terminal 3:	Alfanumérico - caracteres especiais
Terminal 4:	Alfanumérico
Plotter:	Tamanho A2 - 4096H X 4096V
Mesa Digitadora:	50cm X 50cm - 4000H X 4000V

Sistema Operacional:

Nome:	RTE 6 - VM
Tipo:	Tempo Real, Multi-tarefa, Multi-usuário
Software instalado:	FORTRAN 77, Macro, Assembler, Editores, Montadores, Bibliotecas e Utilitários de uso geral.
Capacidade de endereçamento:	Programas: 64 kb - Dados: 128 Mb No. de periféricos: 256 (máximo)

O multiprogramador conectado ao HP 1000 tem as seguintes características:

Modelo:	STD 85-MP
Capacidade:	15 cartões funcionais
Interface:	IEEE-488
Processador:	8085
Cartões:	E/S digitais DAC - 12 bits ADC - 12 bits Relés

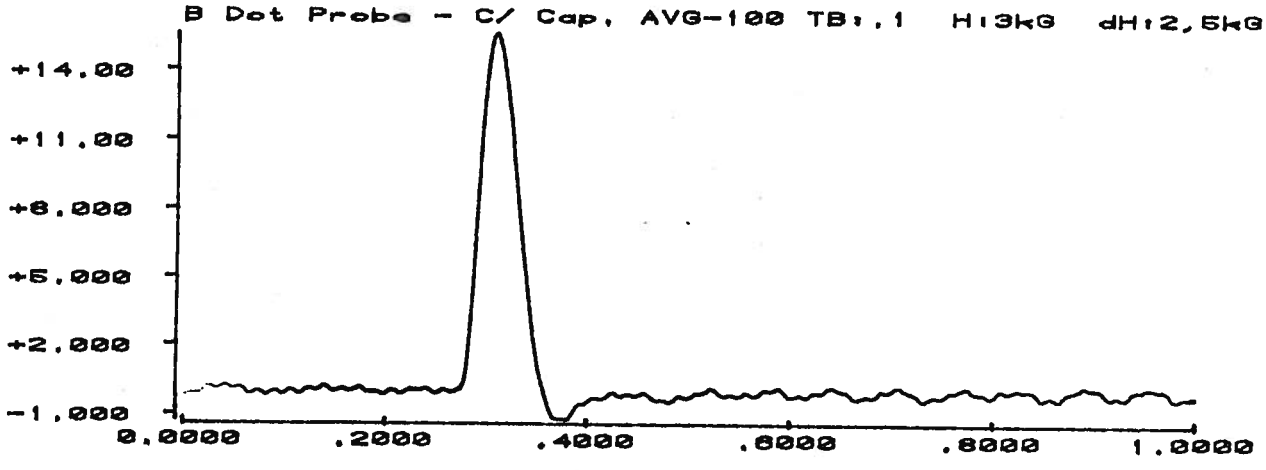
Apêndice 5

Outros Programas do Sistema

```
      ftn77x,l
            program rmu04
      *-----------------------------------------------------------------------
      *     Este programa foi escrito para a  medicao da variacao do campo
      *     magnetico ao longo de um  passo do motor. A ideia basica usada
      *     neste caso foi colocar uma  bobina no entreferro  do eletroima
      *     e amostrar a FEM induzida que aparece quando o campo varia.
      *
      *     A integracao desta FEM nos da' a variacao do fluxo e, portanto,
      *     a variacao do campo ao longo do passo. Como o  sinal da bobina
      *     e' de  poucos milivolts,  a medida e' repetida  varias vezes e
      *     cada uma e' somada no  processo de  average disponivel no osci-
      *     loscopio TEK 7854. A  sincronizacao foi  obtida usando-se o bit
      *     5 da saida paraleda do multiprogramador para disparar o trigger
      *     externo do 7854. Depois de armazenadas, as medidas sao transmi-
      *     tidas para o computador com o programa SENDT. A escolha  do mo-
      *     do "average" deve ser feita no painel do osciloscopio.
      *     Bob                                          ! (900427.1827)
      *-----------------------------------------------------------------------
      * +---------+
      * | M. Prog |
      * |         |
      * |   bit   |
      * |   1 o --|---\         +-------+      +--------------+          HP 1000
      * |   2 o   |    \ ____\  | Motor |      | Osciloscópio |            |
      * |   3 o   |    /      / |       |      |    Digital   |            ^
      * |   4 o --|---/         +-------+      |              |            ^
      * |   5 o --|------------------------->| Ext. Trigger |            |
      * |   6 o   |                          |              |         +--------+
      * |   7 o   |                          |              |         | HP-IB  |
      * |   8 o   |                          |        HP-IB |-))--+--  |        |
      * |         |                          |              |     |   |Extender|
      * |         |                          +--------------+     |   |        |
      * |  HP-IB -|-))-----------------------------------------+   +--------+
      * |         |
      * +---------+
      *-----------------------------------------------------------------------


      *     ------------------------------------------------------------------
            integer*2  passo(0:3)
      *     ------------------------------------------------------------------
            data passo /05b, 11b, 12b, 06b/              ! Config. das fases
      *     ------------------------------------------------------------------
            isconf = 60b                                 ! Configuracao do
            write(44,'("O",@2,"T")') isconf              ! multiprogramador
      *     ------------------------------------------------------------------
            write(1,*) 'Quantos passos?'                 ! Le o numero de passos
            read (1,*)  np                               ! e coloca
            ist = 0                                      ! o motor
            isa = passo(iand(ist,3))                     ! em uma
            write(44,'("L",@5,"T")') isa                 ! posicao inicial
            pause                                        ! Pausa
      *     ------------------------------------------------------------------
            do k = 1,np                                  ! Inicio do loop
              ist = ist+1                                ! Incr. cont. de passo
              call pulsa_bit(isa,5,44)                   ! Simula pulso no bit 5
              call exec(12,0,1,0,-25)                    ! Espera 250 ms
              isa = passo(iand(ist,3))                   ! Xmite configuracao
              write(44,'("L",@5,"T")') isa               ! para o motor
```

```
        call exec(12,0,1,0,-100)                    ! Espera 1000 ms
      end do                                        ! Fim do loop
      pause                                         ! Pausa
*     ------------------------------------------------------------------
      do k = np,1,-1                                ! Inicio do loop
        ist = ist -1                                ! Decr. cont. de passo
        isa = passo(iand(ist,3))                    ! Xmite configuracao
        write(44,'("L",@5,"T")') isa                ! para o motor
      end do                                        ! Fim do loop
*     ------------------------------------------------------------------
      isa = 0                                       ! Limpa a configuracao
      write(44,'("L",@5,"T")') isa                  ! Xmite para deixar o
      stop                                          ! motor livre e encerra
*     ------------------------------------------------------------------
      end
*     ==================================================================
      subroutine pulsa_bit(ivar,ibit,lu)
*     ------------------------------------------------------------------
*     Esta rotina cria um pulso no bit (ibit) da saida paralela do mul-
*     tiprogramador. A largura deste pulso nao e' controlada e depende
*     apenas do tempo de execucao da rotina. O bit pulsado por esta ro-
*     tina pode ser usado, por exemplo, como trigger para instrumentos
*     ligados a uma experiencia. O bit pulsado retorna a sua condicao
*     inicial, ou seja, se ele for 1 e' pulsado em zero e se for zero
*     e' pulsado em um.
*     ------------------------------------------------------------------
      if(btest(ivar,ibit)) then                     ! Se o bit vale 1 entao
        ivar = ibclr(ivar,ibit)                     ! faz o bit valer 0
        write(lu,'("L",@5,"T")') ivar               ! e xmite para a porta.
        ivar = ibset(ivar,ibit)                     ! Volta o bit para 1
        write(lu,'("L",@5,"T")') ivar               ! e xmite para a porta.
        return                                      ! Retorna
      else                                          ! Caso contrario,
        ivar = ibset(ivar,ibit)                     ! faz o bit valer 1
        write(lu,'("L",@5,"T")') ivar               ! e xmite para a porta.
        ivar = ibclr(ivar,ibit)                     ! Volta o bit para 0
        write(lu,'("L",@5,"T")') ivar               ! e xmite para a porta
        return                                      ! Retorna
      end if                                        ! Fim
*     ------------------------------------------------------------------
      end
```




```
*     Dados obtidos com este programa
```