Instituto de Física de São Carlos
Departamento de Física e Ciência dos Materiais
Universidade de São Paulo

"SISTEMA AUTOMATIZADO DE MEDIDAS TSC EM PLATAFORMA GUI"

Odemir Martinez Bruno

[redacted]

Dissertação apresentada ao Instituto de Física de São Carlos, para a obtenção do Título de Mestre em Física Aplicada.

Orientador
Prof. Dr. Roberto Mendonça Faria

SÃO CARLOS
1995

IFSC — SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

UNIVERSIDADE
DE SÃO PAULO
Instituto de Física de São Carlos

Av. Dr. Carlos Botelho, 1465
CEP 13560-250 - São Carlos - SP
Brasil

Fone (0162) 72-6222
Fax (0162) 72-2218

MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE ODEMIR MARTINEZ BRUNO APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 24 DE AGOSTO DE 1995.

COMISSÃO JULGADORA:

Prof. Dr. Roberto Mendonça Faria

Prof. Dr. Luiz Antonio de Oliveira Nunes

Prof. Dr. Adilson Gonzaga

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Roberto Mendonça Faria, pela orientação científica, motivação, incentivo, credibilidade e amizade, sem os quais não teria realizado este trabalho.

Aos meus pais Orlando Bruno e Maria Candida Martinez Bruno, pelo incentivo e colaboração.

Ao amigo Ademir Soares, pelo auxílio técnico e excelente suporte em eletrônica.

Aos professores do Grupo pelas sugestões e contribuições, no decorrer deste trabalho.

Aos amigos do Grupo pela colaboração, incentivo e sugestões na execução deste trabalho.

Aos técnicos do Grupo, Níbio e Berto, pelo apoio técnico.

Ao meu grande amigo José de Souza Nogueira, pelas sugestões e incentivo.

A todos meus professores que com muita dedicação contribuíram em minha formação pessoal e profissional, desenvolvendo a arte de ensinar diante do preconceito da nossa sociedade néscia.

A todos que, direta ou indiretamente, prestaram seu apoio e colaboração, na realização deste trabalho.

Este trabalho foi realizado com o apoio financeiro do CNPq.

SUMÁRIO

LISTA DE FIGURAS i

RESUMO iii

ABSTRACT iv

CAPÍTULO I - INTRODUÇÃO 1

CAPÍTULO II - TÉCNICAS DE CORRENTES TERMOESTIMULADAS - TSC 6

2.1 - INTRODUÇÃO 6

2.2 - TSC EM CIRCUITO FECHADO 7

CAPÍTULO III - DESCRIÇÃO DO HARDWARE 12

3.1 - INTRODUÇÃO 12

3.2 - DESCRIÇÃO GERAL 12

3.3 - MICROCOMPUTADOR 16

3.4 - CONTROLE DE TEMPERATURA 18

3.4.1 - O Controlador 23

3.4.2 - A Comunicação Entre as Portas Seriais 24

3.4.3 - A Estufa 25

3.4.4 - O Sensor de Temperatura 29

3.4.5 - O Acionador de Carga 30

3.5 - AQUISIÇÃO DE DADOS 32

3.5.1 - A Placa Processadora de Sinais 35

3.5.2 - O Eletrômetro 40

CAPÍTULO IV - INTERFACE GRÁFICA COM O USUÁRIO (GUI) 41

4.1 - INTRODUÇÃO 41

4.2 - CONTEXTO HISTÓRICO 41

4.2.1 - História da Portabilidade 42

4.2.2 - História da GUI 46

4.3 - DEFINIÇÃO DE GUI 50

4.4 - ELEMENTOS DE UMA GUI 52

4.5 - MODELO DE CAMADAS 55

4.6 - FUNDAMENTOS DE PROGRAMAÇÃO EM GUI 58

4.6.1 - Comparação de um Programa Mínimo entre CLI e GUI 59

4.6.2 - Ferramenta de Auxílio à Programação 60

4.6.3 - Programação Orientada a Eventos 61

4.6.4 - Programação Orientada para Objetos 63

4.6.5 - Programação Visual *65*
4.6.6 - Ligações Dinâmicas *66*
4.6.7 - Multitarefa *68*

CAPÍTULO V - DESCRIÇÃO DO SOFTWARE **70**

5.1 - INTRODUÇÃO 70
5.2 - FERRAMENTAS DE PROGRAMAÇÃO 70
5.3 - DESCRIÇÃO E OPERAÇÃO DO SISTEMA 71
5.3.1 - Janela Principal *71*
5.3.2 - Janela de configuração do sistema *74*
5.3.3 - Janela de Gravação de Arquivos *77*
5.3.4 - Alteração de Fundo de Escala *78*
5.3.5 - Janelas Diversas *80*
5.4 - PROCESSAMENTO ESPECÍFICO 82

CAPÍTULO VI - RESULTADOS EXPERIMENTAIS **92**

6.1 - INTRODUÇÃO 92
6.2 - CONTROLE DE TEMPERATURA 92
6.2.1 - Comparações com o Sistema Anterior *95*
6.3 - MEDIDAS TSC 97

CAPÍTULO VII - CONCLUSÃO **100**

REFERÊNCIAS **102**

APÊNDICE A **104**

Padrão RS-232C *104*

APÊNDICE B **106**

APÊNDICE C **109**

LISTA DE FIGURAS

FIG. - 2.1 - ESQUEMA GENÉRICO DO SISTEMA TSC 7
FIG. - 2.2 - REPRESENTAÇÃO DOS DIPOLOS E DAS CARGAS ELÉTRICAS NA AMOSTRA 8
FIG. - 2.3 - REPRESENTAÇÃO DO MOVIMENTO DE CARGAS 8
FIG. - 2.4 - CURVAS REPRESENTATIVAS DO PROCESSO DE MEDIDAS TSC 9
FIG. - 2.5 - CURVAS TSC COM UMA SIMPLES RELAXAÇÃO DE DEBYE 11
FIG. - 3.1 - DIAGRAMA DAS CONEXÕES E MÓDULOS DO SISTEMA 13
FIG. - 3.2 - FLUXOGRAMA DO PROGRAMA ESPECIALIZADO EM CONTROLE 21
FIG. - 3.3 - SISTEMA DE CONTROLE AUTOMÁTICO 22
FIG. - 3.4 - FORMATO DO DADO NA COMUNICAÇÃO SERIAL 25
FIG. - 3.5 - ARRANJO ESQUEMÁTICO DA BASE 26
FIG - 3.6 - ARRANJO ESQUEMÁTICO DA CÚPULA 27
FIG - 3.7 - ARRANJO ESQUEMÁTICO DA ESTUFA 28
FIG. - 3.8 - ARRANJO ESQUEMÁTICO DA BOMBA DE VÁCUO 28
FIG. - 3.9 - CORTE LATERAL DE UM SENSOR TERMORESISTIVO DE PLATINA 29
FIG. - 3.10 - CURVA RESISTÊNCIA X TEMPERATURA DO PADRÃO PT100 30
FIG. - 3.11 - ESQUEMA ELÉTRICO DO CIRCUITO ACIONADOR DE CARGA 31
FIG. - 3.12 - ESQUEMA DO CIRCUITO CONVERSOR DA REFERÊNCIA DE TEMPERATURA 34
FIG. - 3.13 - DIAGRAMA ESQUEMÁTICO DA ENTRADA ANALÓGICA DA PC LAB PLUS 37
FIG. - 3.14 - TENSÕES DO AMPLIFICADOR DE INSTRUMENTAÇÃO 38
FIG. - 3.15 - INTERLIGAÇÃO ENTRE O ELETRÔMETRO E UM CIRCUITO TSC 40
FIG. - 4.1 - DIAGRAMA DE CAMADAS DO GUI EM UNIX 45
FIG. - 4.2 - DIAGRAMA DA EVOLUÇÃO DE SISTEMAS OPERACIONAIS E GUI 49
FIG. - 4.3 - O ESPAÇO DE TRABALHO 52
FIG. - 4.4 - CAIXA DE DIÁLOGO 54
FIG. - 4.5 - ASPECTO VISUAL DE UMA GUI 55
FIG. - 4.6 - MODELO EM CAMADAS DE UMA GUI 56
FIG. - 4.7 - DIAGRAMA COMPARATIVO DE CAMADAS DE GUI 58
FIG. - 4.8 - EXEMPLO DO PROGRAMA "HELLO WORLD" EM CLI 59
FIG. - 4.9 - JANELA DO PROGRAMA "HELLO WORLD" EM GUI 60
FIG. - 4.10 - ESQUEMA DO FUNCIONAMENTO DA FERRAMENTA CASE 61
FIG. - 4.11 - RESULTADO E DIAGRAMA DE BLOCOS DO PROGRAMA BASKARA EM CLI 62
FIG. - 4.12 - JANELA E DIAGRAMA DE BLOCOS DO PROGRAMA BASKARA EM GUI 63
FIG. - 4.13 - CONCEPÇÃO DE OBJETO 64
FIG. - 4.14 - ESCALONAMENTO DE TAREFAS 68
FIG. - 5.1 - JANELA PRINCIPAL 73
FIG. - 5.2 - MENUS DA JANELA PRINCIPAL 73
FIG. - 5.3 - JANELA DE CONFIGURAÇÃO 74

FIG. - 5.4 - PARTE DA JANELA DE CONFIGURAÇÃO COM ESTILO CONSTANTE 76
FIG. - 5.5 - JANELA DE GRAVAÇÃO 77
FIG. - 5.6 - JANELA DE ALTERAÇÃO DE PARÂMETROS DO ELETRÔMETRO 79
FIG. - 5.7 - JANELA DE AVISO 80
FIG. - 5.8 - JANELA DE CONTROLE DE TEMPERATURA 80
FIG. - 5.9 JANELA DE INFORMAÇÃO SOBRE O SISTEMA 81
FIG. - 5.10 - ARQUIVO DE AJUDA 82
FIG. - 5.11 - DIAGRAMA DE EVENTOS DO SISTEMA 83
FIG. - 5.12 - FLUXOGRAMA INICIAL 84
FIG. - 5.13 - FLUXOGRAMA DA SUB-ROTINA CALCPARM 85
FIG. - 5.14 - FLUXOGRAMA DA FUNÇÃO CALTIME 87
FIG. - 5.15 - FLUXOGRAMA DA SUB-ROTINA PRINCIPAL 90
FIG. - 6.1 - CURVAS TEMPERATURA X TEMPO 93
FIG. - 6.2 - CURVAS TEMPERATURA X TEMPO - INÉRCIA TÉRMICA 94
FIG. - 6.3 - VISUALIZAÇÃO DA REGRESSÃO LINEAR 95
FIG. - 6.4 - COMPARAÇÃO DE RAMPAS ENTRE SISTEMAS 96
FIG. - 6.5 - COMPARAÇÃO DAS REGRESSÕES LINEARES ENTRE SISTEMAS 97
FIG. - 6.6 - RESULTADO OBTIDO PELA MEDIDA TSC 98

RESUMO

Neste trabalho desenvolvemos um método de controle e aquisição de dados, particularmente aplicado a medidas de correntes termoestimuladas (TSC), mas que podem ser estendidas a sistemas semelhantes. O sistema automatizado de medidas TSC em plataforma GUI, foi implementado de modo a realizar o controle da temperatura de uma estufa, de forma que seja possível gerar rampas lineares com taxas programáveis, e também a leitura de sinais analógicos referentes ao sistema, como corrente e temperatura, e processar os sinais obtendo como resultado curvas gráficas (sinais processados em função do tempo). Os resultados são armazenados e podem ser exibidos, ou impressos e transferidos para o padrão de arquivo de entrada de outros aplicativos. Os dados são obtidos em tempo real e o sistema utiliza conceitos da tecnologia GUI, onde provemos um estudo desta, gerando janelas com interação orientada a eventos e uma interface gráfica com o usuário bastante interativa. O sistema faz uso de microcomputador padrão IBM PC AT e utiliza o MS-Windows 3.1 como plataforma GUI ("Graphics User Interface").

ABSTRACT

A method for control and data acquisition using GUI environment, applied to thermal and electrical measurements was developed. The method was specifically applied to thermal stimulated current (TSC), but it can be extended to similar systems. The GUI - based automated TSC system maked possible the accurate control of the oven temperature, at differents heating rates. This system performs the data acquisition of analogic signals (current and temperature, for instance), and those signals are processed generating functions of time and/or temperature (TSC curves, for example). Stored results can be displayed, printed, as well as transferred to files which are compatible to standard aplicatives for data treatment. Data are obtained in real time and the system uses GUI concepts, which are extensively studied in this work, generating events oriented windows and a user-friendly graphical interface. The system configuration consists of a IBM PC AT microcomputer and the MS-Windows 3.1 is used as the GUI environment.

CAPÍTULO I - INTRODUÇÃO

O surgimento do microcomputador na década de 70 iniciou uma verdadeira revolução na computação. O microcomputador obteve sucesso absoluto, ampliou e continua a cada dia ampliando seu horizonte de utilização.

Graças a esse sucesso, a indústria e a pesquisa foram largamente motivadas, e o aprimoramento tecnológico e o poder computacional apresentaram, nesse período um crescimento em escala exponencial. Ainda devido ao crescente aumento do poder computacional, a filosofia da programação dos computadores e da interação homem - máquina, evoluiu.

No princípio, com a limitação tecnológica e computacional, a filosofia operacional era que os programas e os métodos de operação dos computadores aproveitassem ao máximo os escassos recursos computacionais, não se preocupando com a forma com que a manipulação da informação era feita. Assim, o modo de operação dos computadores gerava um ambiente hostil, onde o usuário deveria conhecer uma grande quantidade de comandos e parâmetros, para introduzir e processar os dados.

Nestes ambientes hostis, não existia nenhuma padronização, o modo de operação era diferente para cada tipo de máquina. Caso o usuário quisesse operar um computador de uma linha diferente, deveria receber um novo e prolongado treinamento, pois sua operação seria diferente e também complexa. Não obstante, se desejasse operar um novo programa, na mesma máquina, sua metodologia operacional também seria diferente, de modo que seria necessário um treinamento adicional.

Tais fatos geravam desconforto, pois a operação dos computadores era uma tarefa bastante complexa. Os usuários eram forçados a fazer prolongados treinamentos, e a cada novo programa que desejassem manipular, pela falta de padronização e complexidade, novos treinamentos eram necessários.

Outrossim, o desconforto tornava ainda maior, caso houvesse a necessidade de migrar para um novo sistema ou programa. Neste caso o usuário descartava o treinamento anterior na migração, e um novo aprendizado era necessário.

A evolução tecnológica fez com que a máquina obtivesse recursos para se tornar cada vez mais adequada à utilização humana. Essa adequação ocorreu, devido à evolução do potencial computacional dos computadores, dos dispositivos de apresentação de informação e dos dispositivos de entrada de dados. Com isto, foi possível a evolução da manipulação da informação em função da correlação homem-máquina.

No início da década de 70, a Xerox criou o seu centro de pesquisa na área de computação: o Palo Alto Research Center (PARC). O seu objetivo era criar uma nova arquitetura para a forma de manipulação de informação.[1]

O trabalho pioneiro do PARC, gerou um novo conceito na manipulação de informação, o qual é denominado de GUI (Graphic User Interface) ou Interface Gráfica com o Usuário.

A partir deste novo conceito, o monitor (dispositivo de apresentação de informação) além de exibir letras, passou a exibir também formas gráficas. As formas gráficas possibilitaram uma melhor utilização da tela, transmitindo informações de uma maneira visual mais rica.

Além de exibir informações mais ricas, o monitor se transformou em uma fonte de entrada de dados do usuário. A tela do vídeo agora mostra vários objetos gráficos na forma de ícones e dispositivos de entrada de dados, como botões e barras de rolagem. Usando um dispositivo apontador, o usuário pode diretamente manipular esses objetos na tela. Os objetos gráficos podem ser arrastados, os botões podem ser pressionados e as barras de rolagem podem ser roladas.

A interação entre o homem e a máquina torna-se também então mais íntima com a adequação da máquina à utilização humana.

Desta forma os usuários não precisam gastar um longo período de treinamento para usar o computador ou para dominar um programa. Além do que, com a padronização imposta pela GUI, todos os programas têm a

mesma aparência e modo de operação fundamental. Uma vez que o usuário saiba como operar um programa, poderá facilmente operar outro. E caso seja necessário migrar para um novo programa, o treinamento anterior não é totalmente descartado.[2]

Uma importante área em que o microcomputador amplia sua utilização é a instrumentação eletrônica. Nesta, o computador auxilia o desenvolvimento de pesquisas científicas, agilizando, aperfeiçoando e melhorando o desempenho da área experimental. É o que vem acontecendo no Instituto de Física de São Carlos, que no decorrer do tempo atualiza-se, utilizando o avanço científico-tecnológico, para o desenvolvimento da pesquisa de base.

O Grupo de Polímeros "Prof. Bernhard Gross" do Instituto de Física de São Carlos tem suas atividades dirigidas às pesquisas de base e desenvolvimento de técnicas empregadas no estudo de propriedades elétricas de materiais poliméricos. Atualmente tem-se interesse em técnicas que permitem o estudo das propriedades de armazenamento e transporte de cargas elétricas em filmes de polímeros.

Uma importante ferramenta para efetuar tais estudos, foi desenvolvida há algum tempo atrás no Grupo e desde então, amplamente utilizada pelos pesquisadores do Grupo. Trata-se de um sistema automatizado para realizar medidas em filmes de polímeros.

O sistema funciona sob a plataforma do microcomputador Apple II de 8 bits e a ele são acoplados dispositivos para que possa controlar a temperatura de uma estufa e fazer a aquisição da corrente e da temperatura em função do tempo.[3]

Esse sistema tem sido largamente utilizado e auxiliado com muito êxito, importantes trabalhos realizados no Grupo. Entretanto devido ao avanço tecnológico, o sistema vem se tornando limitado e antiquado. Entre outras limitações podemos citar:

- a baixa qualidade de controle de temperatura, que gera uma rampa pouco precisa;
- a baixa qualidade da aquisição, gerando baixa resolução dos valores

obtidos;

- sensores com tecnologia superada, de pouca precisão;
- o sistema é baseado numa plataforma de 8 bits, há muito desvinculada do parque de microcomputadores, sendo portanto, necessários alguns processos para transferir os resultados para os microcomputadores de outra plataforma, onde são processados;
- programa com recursos limitados;
- pouca interação com o usuário;

Sendo o sistema vital para a pesquisa e desenvolvimento no Grupo, foi sugerido o desenvolvimento de um novo sistema para uma plataforma atualizada de microcomputadores, que venha a utilizar os melhoramentos gerados pelo avanço tecnológico no decorrer dos anos.

Iniciou-se então o desenvolvimento de um novo sistema com as melhorias disponíveis pelo avanço tecnológico, porém mantendo ainda a idéia original de baixo custo, incrementado com a utilização de arquitetura GUI e multitarefa. Sendo a instrumentação eletrônica ainda uma das barreiras de utilização da GUI, a proposta promove um estudo de certa forma pioneiro em relação à utilização de GUI e conceitos de multitarefa num sistema de instrumentação eletrônica em tempo real.

Para a realização desta tarefa, fora utilizado um microcomputador padrão IBM PC AT, com microprocessador 80486dlc, com relógio de 40 MHz, e 4 Mbytes de memória RAM, com disco rígido de 340 Mbytes. Como plataforma GUI o sistema comercial MS-Windows 3.1 , além de um microcontrolador modelo Gefran e módulo CLB de comunicação com microcomputador via porta serial padrão RS-232C, e ainda dispositivos para interfaceamento e instrumentação, descritos posteriormente.

O sistema foi implementado de modo a realizar o controle da temperatura de uma estufa, capaz de gerar rampas lineares com taxas programáveis, e também a leitura de sinais analógicos referentes ao sistema, como corrente e temperatura, e ainda processar os sinais, obtendo como resultado curvas gráficas (sinais processados em função do tempo).

Este sistema é semelhante ao anterior no que diz respeito às suas funções, porém com eficiência muito superior e a possibilidade de processar parte do resultado matemático no próprio sistema.

Os resultados são armazenados e podem ser exibidos, ou impressos e transferidos para o padrão de arquivo de entrada, de outros aplicativos comerciais ou não, desenvolvidos para a análise e processamento matemático, onde podem ser mais bem estudados, ou ainda transferidos para outras plataformas de computação tais como Workstations UNIX ou para o VMS do computador VAX (ambos disponíveis no Instituto de Física de São Carlos), via programas de comunicação disponíveis na rede do IFSC, ou ainda através de programas de conversão.

Os resultados são obtidos em tempo real e utiliza conceitos GUI, gerando janelas com interação orientada a eventos, além também da utilização da técnica de multitarefa, disponível no conceito GUI.

O sistema mostrou ser bastante versátil e possuir capacidade de expansão. Seu desenvolvimento também é justificado pelo baixo custo e boa qualidade, além, é claro, da simplicidade de utilização e da padronização imposta pela GUI, diminuindo assim o treinamento do usuário que deseja operar o sistema.

CAPÍTULO II - TÉCNICAS DE CORRENTES TERMOESTIMULADAS - TSC

2.1 - INTRODUÇÃO

Nosso trabalho está direcionado para medidas elétricas em materiais sólidos, mas a técnica que está sendo usada é a de Correntes Termoestimuladas, conhecida por TSC (Thermal Stimulated Current). Na verdade a técnica de TSC apresenta inúmeras variações, que levam diferentes designações, conforme alterações dos parâmetros experimentais, ou o fenômeno a ser observado nos materiais. Algumas delas são: Correntes de Polarização Termoestimuladas (TSP), Correntes de Despolarização Termoestimuladas (TSDC), Correntes Iônicas Termoestimuladas (ITC), etc. Não é objetivo deste trabalho tratar em profundidade essa família de técnicas de medidas. Daremos, entretanto, uma visão geral sobre sua operação e os fenômenos físicos envolvidos.[4]

A técnica de TSC consiste na medida de uma corrente elétrica numa amostra dielétrica, colocada entre dois eletrodos metálicos, a qual é aquecida a uma taxa constante de variação de temperatura. A corrente medida tem origem no movimento de cargas no interior do material, ou na reorientação de dipolos elétricos devido a movimentos térmicos. Ambos os processos são estimulados pelo aumento de temperatura, e conforme a energia associada a um determinado mecanismo (energia de ativação), picos de corrente podem surgir. Dependendo da amostra, e do intervalo de temperatura da medida, podem surgir até vários picos de corrente elétrica. Esses picos são objeto de investigação que, em geral, correlacionam propriedades elétricas com a estrutura do material.

Para detalhar esse método, apresentamos um tratamento um pouco mais formal do estudo de TSC em circuito fechado.

2.2 - TSC EM CIRCUITO FECHADO[3,4]

Temos na figura 2.1 representação de um esquema genérico do sistema TSC em circuito fechado.

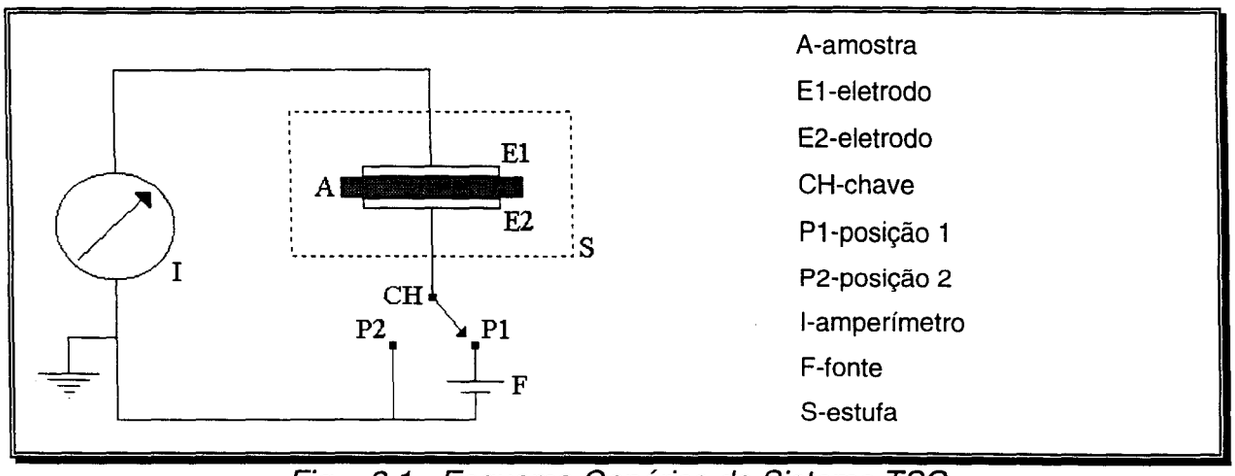


Fig. - 2.1 - Esquema Genérico do Sistema TSC

A amostra **A** possui dois eletrodos **E1** e **E2** metalizados diretamente sobre suas superfícies, a fim de proporcionar bom contato ôhmico. Acoplamos o conjunto amostra/eletrodos no interior da estufa **S**, cuja temperatura possa ser controlada, de modo que seja constante ou aquecida com taxa constante.

Conectamos o eletrodo **E1** ao dispositivo **I** que permite a leitura da corrente proveniente da amostra. O Eletrodo **E2** é conectado à chave **CH** que contém as opções **P1** e **P2**. Quando a chave está na posição **P2** a amostra fica curto-circuitada, e na posição **P1**, polarizamo-la com a tensão da fonte **F** (a qual deve propiciar um campo relativamente alto).

Vamos supor que possuímos uma amostra de um material sólido, cuja estrutura contenha uma determinada concentração de dipolos elétricos. Inicialmente, estes dipolos estão ordenados caoticamente (figura 2.2a). Colocando a chave **CH** na posição **P2**, polarizamos a amostra, por um campo elétrico relativamente alto. Aquecemos então, a temperatura da estufa, até que esta proporcione mobilidade às moléculas da amostra, de modo que seja possível a orientação dos dipolos, e mantemos essa temperatura constante por tempo determinado até que os dipolos se orientem com o campo (figura 2.2b e fase 1 da figura 2.4).

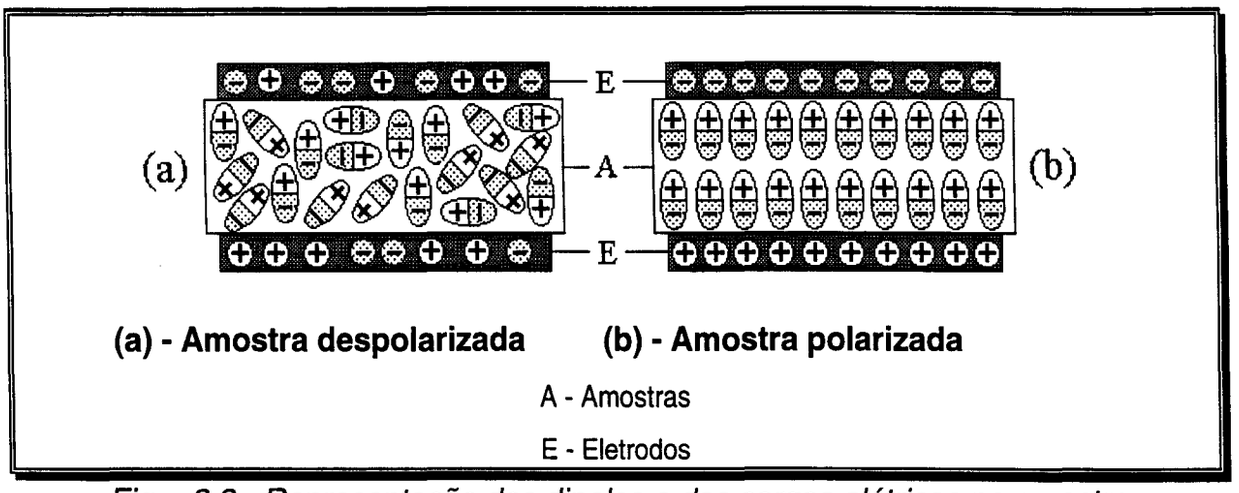


Fig. - 2.2 - Representação dos dipolos e das cargas elétricas na amostra

Mediante o campo elétrico, resfriamos a temperatura da amostra (fase 2 da figura 2.4), até que esta se torne novamente vítrea, impossibilitando movimento molecular e conseqüentemente desorientação dos dipolos. A seguir, colocamos a amostra em curto-circuito, que mantém uma polarização oriunda da orientação dipolar "congelada" em sua estrutura (figura 2.2b).

Com a orientação dos dipolos temos a situação representada pela figura 2.2b, onde os dipolos "prendem" cargas inversas à polaridade de suas extremidades.

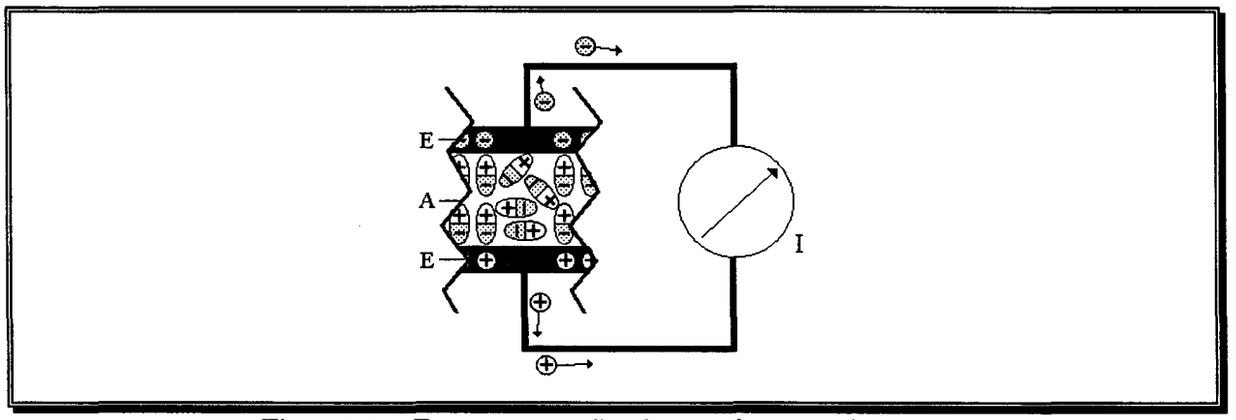


Fig. - 2.3 - Representação do movimento de cargas

Aquecemos a estufa com taxa constante, e deste modo, vamos "descongelando" a estrutura molecular da amostra, aumentando sua mobilidade, e deste modo possibilitando a desorientação dos dipolos (fase 3 da figura 2.4). À medida que os dipolos se desorientam, é "libertada" a carga que era prendida pela orientação dos dipolos, e desta forma é induzida a corrente (figura 2.3).

A figura 2.4 apresenta os gráficos da temperatura versus tempo, ddp versus tempo e corrente versus tempo, para o processo TSC. Na fase 1, temos o

aquecimento sobre campo elétrico da amostra e a polarização de seus dipolos, na fase 2 temos o resfriamento da amostra sobre o campo elétrico, e conseqüentemente o "congelamento" molecular. E finalmente na fase 3 temos a medida da corrente oriunda da desorientação dos dipolos, mediante o aquecimento com taxa constante.

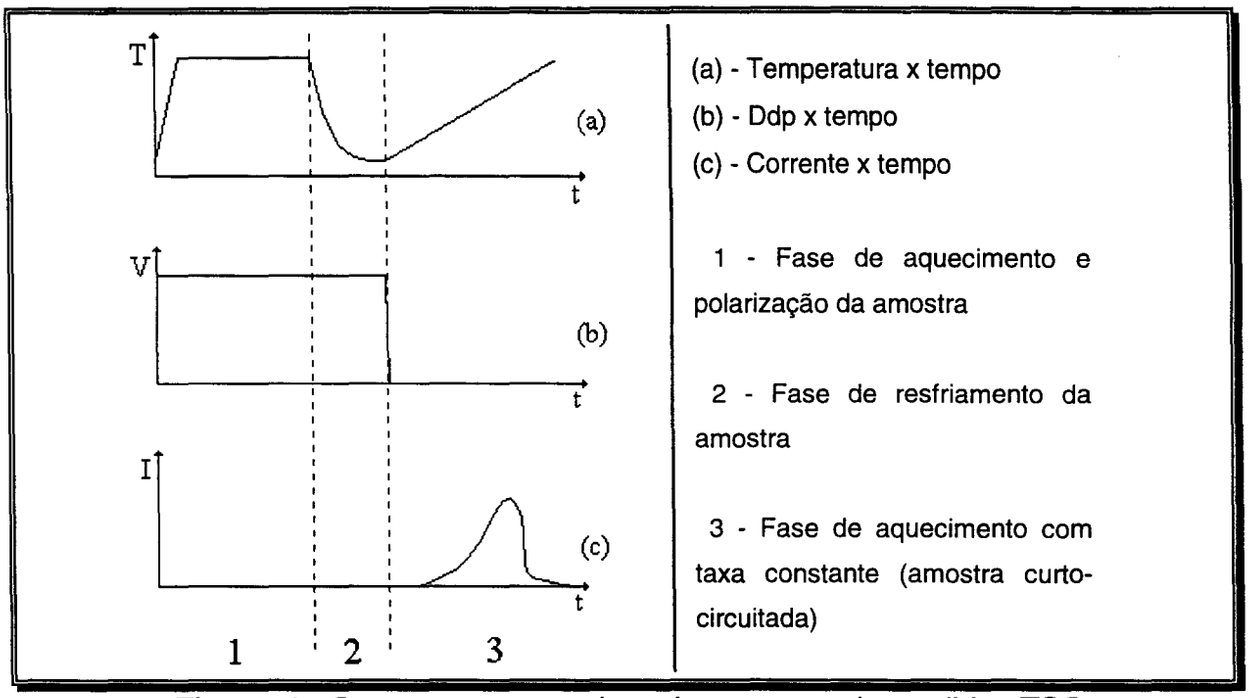


Fig. - 2.4 - Curvas representativas do processo de medidas TSC

Confome podemos observar na fase 3 da figura 2.4, ao se aquecer a amostra, com taxa constante, o que se observa é que uma corrente (de curto-circuito) começa a ser registrada, aumenta com a temperatura, atinge um máximo e decai a zero. O processo pode ser explicado pela desorientação dos dipolos à medida que a temperatura cresce, gerando a corrente de despolarização,

$$i(t) = \frac{dP}{dt} \qquad (2.1)$$

onde P é a polarização da amostra.
Se o decaimento de $P(t)$ é caracterizado por um único tempo de relaxação $\tau(T)$ (T é a temperatura da amostra), seu valor é governado pela equação de Debye,

$$\frac{dP(t)}{dt}+\frac{1}{\tau(T)}P(t)=0 \qquad (2.2)$$

Integrando a equação acima, temos a função polarização,

$$P(t)=P_0\exp\left[-\int_0^t\frac{t}{\tau(T)}dt\right] \qquad (2.3)$$

Sendo o aquecimento linear, isto é, considerando que a taxa de aquecimento seja β = d*t*/dT, derivamos, usando as equações acima, à seguinte expressão,

$$i(T)=-\frac{P_0}{\tau(T)}\exp\left[-\beta\int_{T_0}^{T}\frac{dT}{\tau(T)}\right] \qquad (2.4)$$

Para relaxações que envolvem rotações de pequenos grupos moleculares, o tempo de relaxação obedece à relação de Arrhenius,

$$\tau(T)=\tau_0\exp\left(\frac{A}{kT}\right) \qquad (2.5)$$

onde τ_0^{-1} é a freqüência de escape e *A* a energia de ativação do processo.

Substituindo a equação 2.5 na equação 2.4 e aproximando a integral[4],

$$\beta\int_0^T\frac{dt}{\tau(T)}=\frac{\beta kT^2}{(A+1.85kT)\tau(t)} \qquad (2.6)$$

nós obtemos i(T) da equação 2.4.

A figura 2.5 mostra resultados da expressão

$$\frac{i(T)\tau(T)}{P_0} = \exp\left[\frac{1}{\tau}\left(\frac{\beta\frac{T^2}{T_1}}{\frac{A}{kT_1}+1{,}85\frac{T}{T_1}}\right)\right], \qquad (2.7)$$

derivada da equação 2.6, para diferentes temperaturas (expressas pelo quociente A/kT_1) e diferentes valores de β, em função da temperatura adimensional T/T_1. Para esse caso T_1 é a temperatura do máximo de uma das curvas do gráfico, e é tomada como parâmetro de cálculo. As curvas de correntes termoestimuladas apresentadas na figura 2.5 são calculadas usando como parâmetros A/kT_1 = 10, 15 e 20 e $\beta_1/\beta_2 = \tau_1/\tau_2 = 10$ e $\beta_1/\beta_3 = \tau_1/\tau_3 = 100$.

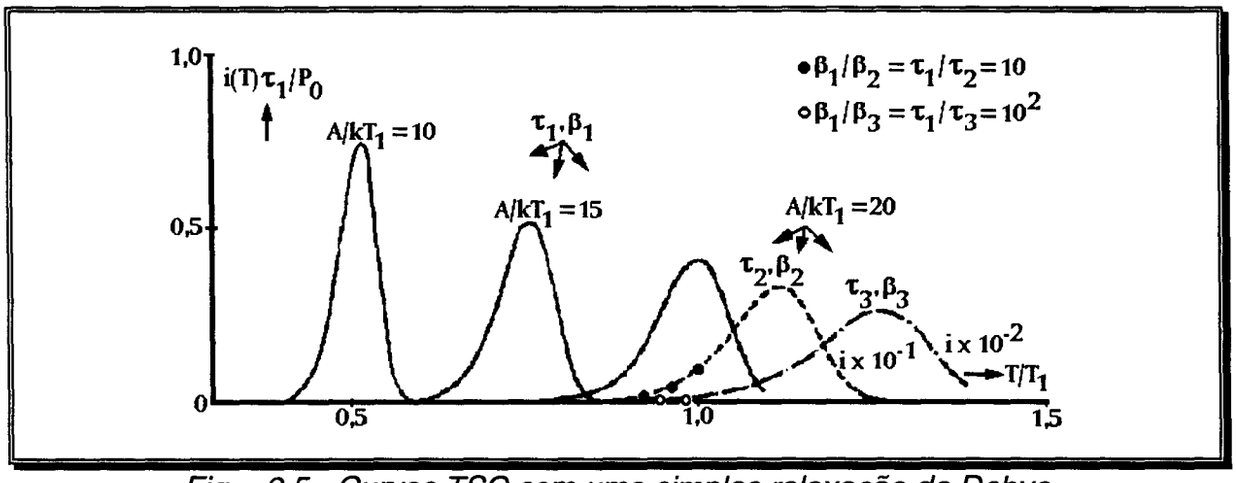


Fig. - 2.5 - Curvas TSC com uma simples relaxação de Debye

CAPÍTULO III - DESCRIÇÃO DO HARDWARE

3.1 - INTRODUÇÃO

A instrumentação de laboratório tem como objetivo auxiliar o trabalho do pesquisador proporcionando a realização adequada de seus trabalhos de experimentação. Com o devido suporte, o trabalho científico pode ser aprimorado, melhorando seu desempenho, qualidade, além de simplificar o manuseio experimental e poupar o tempo gasto em técnicas dos experimentos.

O sistema desenvolvido, abaixo descrito, visou simplificar e aprimorar o trabalho de pesquisadores. Através de um levantamento de idéias, sugestões e observações do cotidiano das medidas realizadas no Grupo de Polímeros "Prof. Bernhard Gross", foi estudado um novo sistema para as medidas TSC, visando melhorar a qualidade das medidas que vinham sendo realizadas e simplificar o manuseio do sistema e o acesso aos resultados [3].

Neste capítulo, descreveremos o "hardware" do sistema, a forma como foi implementado, as conexões entre os instrumentos, o controle de funções, a aquisição de dados e a função de cada um dos módulos.

3.2 - DESCRIÇÃO GERAL

O sistema automatizado de controle de medidas TSC foi implementado para melhorar a qualidade das medidas em corrente termo estimulada, assim como simplificar a operação do usuário e facilitar o acesso ao resultado processado.

O funcionamento básico do sistema é o controle da temperatura de uma estufa, que aquece uma amostra inserida num circuito, e a aquisição de dados de sinais elétricos provenientes da amostra e de sensores situados em determinados pontos da estufa. O resultado é processado e disposto ao usuário.

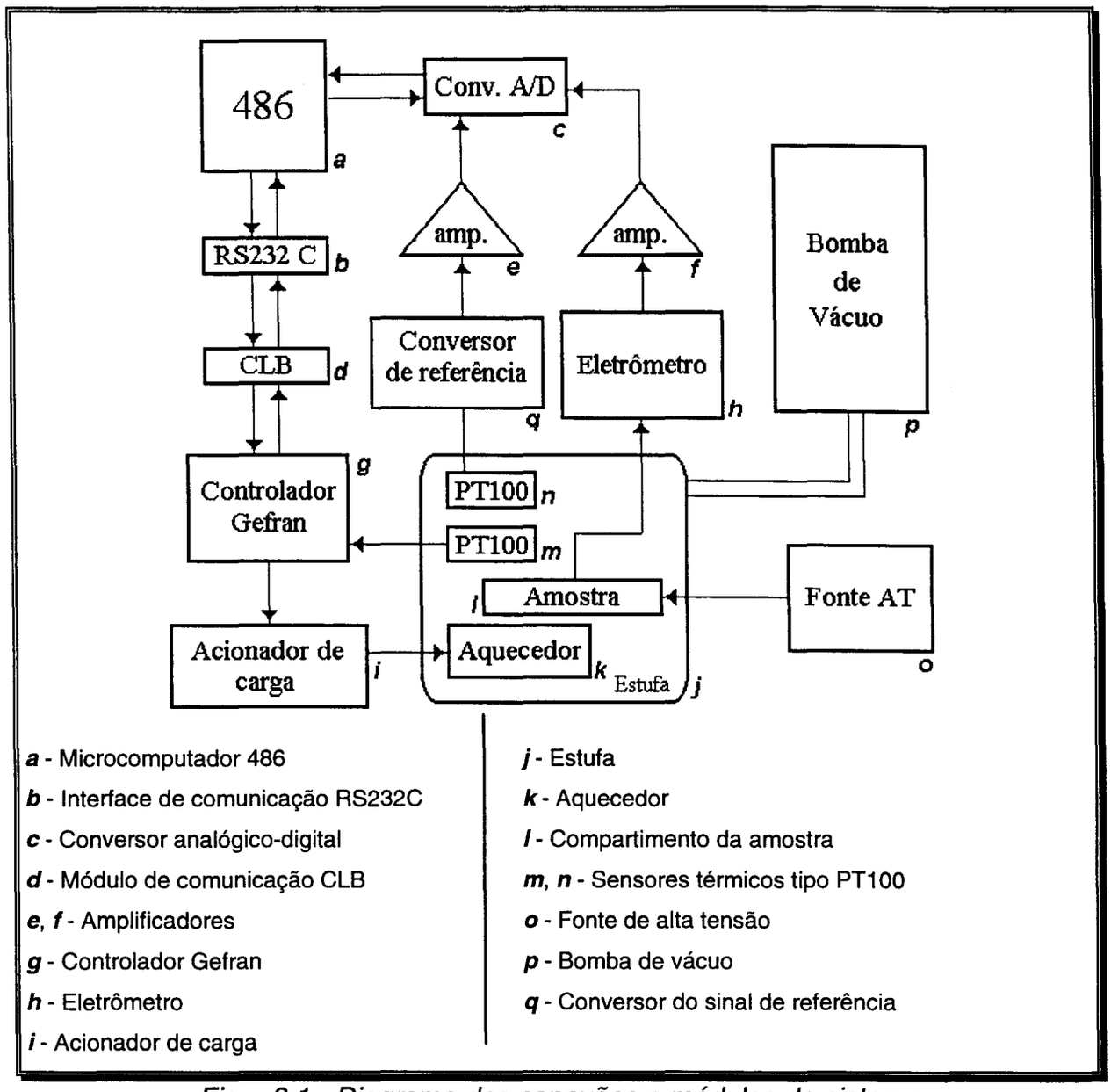


Fig. - 3.1 - Diagrama das conexões e módulos do sistema

Para melhor ilustrar, temos na figura (3.1) um diagrama do sistema, mostrando todas suas conexões e módulos.

O módulo principal do sistema é um microcomputador (*a*), que controla o sistema, faz a aquisição e processamento de dados e também a interface entre o usuário e o sistema. Neste, o usuário deve introduzir a função da temperatura pelo tempo, segundo o interesse do tipo de medida. Isto é feito através da entrada de parâmetros referente à função desejada. O microcomputador processa então a função e obtém as coordenadas temperatura e tempo que são armazenadas na

memória.

O microcomputador, através de programa especializado descrito posteriormente, obtém através de seu relógio interno os instantes de tempo de variação determinada. Esse tempo é então comparado ao instante armazenado na memória *. Quando o tempo atual do relógio corresponde ao tempo definido na memória, temos então o valor da temperatura associada à esse instante de tempo.

O dado resultante, que contém o valor da temperatura encontrada é então transmitido para o módulo responsável pelo controle da temperatura da estufa, ou seja, o controlador (***g***), de modelo comercial fabricado pela Gefran Elettronica.

A transferência de dados é feita pelo microcomputador através da porta serial padrão RS232-C, indicada pelo módulo (***b***), que através de seu protocolo de comunicação transfere o dado para o módulo (***d***), um conversor denominado de CLB. Este, por sua vez, faz a interface entre o padrão serial RS485 do controlador e o padrão serial RS232-C. O dado é então, finalmente, transferido para o controlador.

O controlador possui lógica de controle do tipo pid (proporcional integrativa e diferencial) [6] , e um programa de controle bastante eficiente e "inteligente". A partir dos valores de temperatura que recebe, é capaz de em tempo real, calcular a razão da função da temperatura pelo tempo, e fazer com que a temperatura da estufa, entre dois valores recebidos, se aproxime de uma reta.

Para que o controlador possa, com a maior precisão possível, controlar a temperatura da estufa, utilizamos um sensor de temperatura Pt100, descrito pelo módulo (***m***), que possibilita ao controlador, obter a temperatura da estufa (***j***).

O controlador possui também um sinal de disparo, que é recebido por um circuito acionador de carga, descrito pelo módulo (***i***). Este por sua vez é capaz de fornecer a corrente necessária para o aquecedor (***k***) responsável pelo aquecimento da estufa. Descrevemos os processos de controle da temperatura da estufa mais detalhadamente no item 3.4.

Dependendo da estufa utilizada no sistema podemos, a fim de obter melhor qualidade, interligá-la a uma bomba de vácuo (descrita pelo módulo (***p***)).

Enquanto a temperatura da estufa obedece a função estabelecida pelo usuário num determinado período de tempo, o sistema também faz aquisição de

---

* Oriundo da função temperatura-tempo

dados, que são armazenados no microcomputador, e resultarão na resposta medida pelo sistema.

A amostra possui eletrodos, que são interligados seguindo a técnica de corrente termo estimulada (TSC). Para isto, precisamos de uma fonte de alta tensão, representada pelo módulo (***o***).

A polarização da amostra é então medida utilizando-se um eletrômetro de alta precisão (***h***), devidamente conectado aos eletrodos da amostra, conforme a técnica TSC.

O eletrômetro, possui uma saída analógica que contém um sinal de referência, proporcional ao valor medido. Este sinal é então amplificado e ajustado, pelo amplificador (***f***)(interior da placa PC Lab Plus), e então é transmitido para o conversor de sinais analógicos em digitais, descrito pelo módulo (***c***).

Os amplificadores (***f***) e (***e***) e o conversor (***c***), são partes da placa comercial, PC Lab Plus, fabricada pela National Instruments Co., que é conectada em um dos "slots" do microcomputador. Essa placa possibilita ao microcomputador ler e amplificar sinais elétricos e também enviá-los na forma digital e analógica.

Após ser feita a conversão do sinal pela conversor, o microcomputador pode então ler o valor do sinal de referência. Deste modo é feita a aquisição da corrente proveniente da amostra.

Outro valor que precisa também ser medido e armazenado é a temperatura da amostra, concomitante à corrente medida e armazenada.

Para fazer a aquisição do valor da temperatura, utilizamos um segundo sensor PT100 localizado dentro da estufa representado pelo módulo (***n***), e conectado ao conversor de referência (módulo (***q***)), desta forma gerando um sinal de referência à temperatura em função da diferença de potencial. Este sinal é então conectado ao amplificador representado pelo módulo (***e***). A tensão de referência proveniente do amplificador (***e***) é convertida pelo conversor e então é realizada a aquisição da temperatura, que é armazenada pelo microcomputador.

O microcomputador faz a aquisição de três medidas, e as armazena na forma de vetor. Assim temos três vetores na memória, que formam a base de um sistema de coordenadas tridimensionais. Temos como vetores, a temperatura, a corrente e o tempo.

Como resultado o sistema gera um arquivo contendo a listagem dos três

vetores, devidamente correlacionados. O sistema tem a possibilidade de apresentar também gráficos contendo as curvas: temperatura por tempo, corrente por tempo e corrente por temperatura.

3.3 - MICROCOMPUTADOR

O controle principal do sistema, as decisões a serem tomadas (com exceção de parte do controle da estufa), o processamento e armazenamento de informações, assim como a interação com o usuário, são realizadas por um microcomputador, representado pelo módulo (***a***) da figura 3.1.

Utilizamos neste projeto um microcomputador padrão IBM PC AT, possuindo o circuito integrado 80486dlc da Cyrix, processando com "clock" de 40 Mhz, como seu microprocessador. Em sua configuração constam 4 Mbytes de memória RAM (Random Access Memory), disco rígido de 340 Mbytes, acionadores de disquetes e monitor de vídeo colorido de alta resolução com controlador gráfico padrão SVGA (Super Video Graphics Array), porto serial padrão RS232-C, e ao menos um "slot" de 8 bits (ou mais) disponível. Existem infinitas possibilidades de configuração em microcomputadores padrão IBM PC AT, devido ao fato de o sistema ser baseado em arquitetura aberta [7]. Utilizamos a configuração descrita pela sua disponibilidade no laboratório.

O parque atual de microcomputadores é formado, em quase sua totalidade por micros padrão IBM PC, tornando este padrão possuidor de uma infinita diversidade de softwares, inclusive sistemas operacionais. Por sua ampla utilização e por possuir arquitetura aberta, o padrão tornou-se bastante versátil. Assim, o acesso a diversos tipos de periféricos, que possibilitam uma boa interconexão a sinais externos é bastante facilitado.[5]

Outra vantagem na utilização deste modelo de microcomputador, é sua boa relação custo/desempenho. Baseado nestes fatos, e também na disposição do equipamento no laboratório, fizemos essa escolha.

Como plataforma de operação e interface com o usuário, optamos em utilizar a tecnologia GUI do inglês "Graphic User Interface" ou ainda em português "Interface Gráfica com o Usuário". Tal tecnologia é bastante revolucionária no

cenário atual da computação, provendo uma incontestável interação homem-máquina.[1]

Em termos de "hardware" a principal vantagem da utilização de GUI, é a facilidade de transportar um sistema desenvolvido em um microcomputador para outro, independendo de sua configuração ou do microprocessador utilizado. Isto é possível através do conceito de padronização e independência de "hardware" utilizado pela tecnologia GUI. Deste modo, é possível migrar um sistema desenvolvido sob plataforma GUI de uma máquina para outra, não importando a nova configuração. O único requisito é que a nova máquina esteja dentro das condições de funcionamento da plataforma GUI utilizada.[1]

Como plataforma de operação e interface gráfica com o usuário (GUI), optamos pelo software comercializado pela Microsoft Inc., o MS Windows 3.1, por ser, dentre os sistemas comerciais, o mais popular e o líder absoluto de utilização no padrão IBM PC. Tendo formado um forte padrão de utilização, e por ser conhecido pela grande maioria dos usuários do parque de microcomputadores atual. Isso diminui a possibilidade de um novo usuário necessitar de treinamento para a utilização da plataforma.

Ao microcomputador é conectado, em seu barramento externo para conexão de periféricos, denominado "slot", uma placa comercial de aquisição de dados e controle, modelo Pc Lab fabricada pela National Instruments, que será descrita posteriormente. Através dela, o microcomputador pode "ler" a tensão elétrica de dispositivos conectados à placa, e deste modo fazer a aquisição de dados. O microcomputador PC AT pode apresentar três tipos de "slot" diferenciados pelo barramento. Eles podem conter 8, 16 e 32 bits de barramento e a placa de aquisição e controle utilizada possui barramento de 8 bits. Aos "slots" com barramentos maiores (16 e 32) podem ser conectados periféricos com barramento menores. Podemos então conectar uma placa com barramento de 8 bits num "slot" de 16 bits ou ainda de 32 bits.

Para transferência de dados entre o microcomputador e o controlador Gefran, é utilizada a porta serial programável padrão RS232-C.

3.4 - CONTROLE DE TEMPERATURA

Uma das tarefas do sistema é o controle da temperatura de uma estufa. Este controle deve ser feito de forma que a temperatura da estufa corresponda a uma função do tempo pré-definida pelo usuário.

As funções de temperatura utilizadas pelo sistema são lineares,

$$T(t) = T_0 + kt \qquad (3.1)$$

onde T é a temperatura (°C), t o tempo (s), T_0 a temperatura inicial (°C) e k a razão de aquecimento (°C/s).

Para as medidas desejadas, optou-se utilizar a razão k sempre com valor positivo (maior que 0). Com esta posição obtemos medidas somente no aquecimento da estufa, desprezando assim, para fins de medidas, seu resfriamento. Além de obter medidas de aquecimento (k>0), o sistema opera também com temperaturas constantes (k=0).

Outra característica, é a composição de funções, ou seja, o sistema tem a flexibilidade de operar com diversas funções. A mudança de uma função para outra ocorre num instante de tempo pré-determinado. Podemos então ter a composição de funções,

$$\begin{cases} T_1(t) = T_0 + tk_1 & p/0 \le t < t_1 \\ T_2(t) = T_1(t_1) + tk_2 & p/t_1 \le t < t_2 \\ T_3(t) = T_2(t_2) + tk_3 & p/t_2 \le t < t_3 \\ \vdots & \\ T_{n-1}(t) = T_{n-2}(t_{n-2}) + tk_{n-1} & p/t_{n-2} \le t < t_{n-1} \\ T_n(t) = T_{n-1}(t_{n-1}) + tk_n & p/t_{n-1} \le t < t_n \end{cases} \qquad (3.2)$$

onde cada subfunção, seguindo o modelo definido em (3.1) corresponde ao período estabelecido em t, onde ti (i=1,..,n), corresponde ao tempo final do período.

Para o usuário cabe apenas definir as razões k e seus períodos correspondentes; o sistema controla automaticamente a temperatura da estufa seguindo as funções definidas. Com a composição de funções, para fins de medidas, pode-se explorar algumas características da curva resultante do experimento. Assim, o usuário pode explorar diferentes taxas de aquecimento para distintos intervalos de medida acentuando características desejadas e melhorando a qualidade da medida.

Para melhor descrever o mecanismo de controle de temperatura, vamos dividi-lo em duas partes: O controle feito pelo microcomputador e o controle realizado pelo controlador. Tomemos primeiramente o controle realizado pelo microcomputador.

No microcomputador, o usuário além de definir a razão k, também define o intervalo de temperatura ao qual a estufa será submetida, através da temperatura inicial T_0 e da temperatura final T_f. Caso a razão k definida seja nula, é definido também o tempo em que a temperatura constante será aplicada. A partir destes parâmetros, o microcomputador processa dois vetores, formando pares de coordenadas. Um vetor temperatura, contendo os valores da temperatura, e um vetor tempo contendo os instantes de tempo correspondentes às respectivas temperaturas.

O vetor de temperatura é calculado utilizando a função

$$T(x) = ix + T_0 \qquad / x \in N \qquad (3.3)$$

onde T é o valor da temperatura (°C) a ser armazenado no vetor, x o índice da posição do vetor, sendo 0 a primeira posição, T_0 a temperatura inicial e i o incremento de temperatura. O valor de i, é definido a fim de obter melhores resultados com o valor da resolução de temperatura permitido pelo equipamento (controlador e sensor). Neste projeto, utilizamos 0,1 °C como valor mínimo para i. São então calculados e armazenados no vetor os valores da temperatura, até que T(x) seja equivalente ao valor da temperatura máxima, definida pelo usuário. Desta forma armazenamos no vetor o intervalo compreendido entre a temperatura inicial e

a temperatura final.

Depois de obtermos o vetor de temperatura, é então processado o vetor de tempo. Para o cálculo dos instantes de tempo, tomamos a função 3.1 na forma,

$$t(T) = \frac{T - T_0}{k} \qquad (3.4)$$

substituímos T pela função 3.3 e obtemos então;

$$t(x) = \frac{ix}{k} \, . \qquad (3.5)$$

Assim calculamos e armazenamos os valores, obtendo então o vetor de tempo.

Depois de processados os vetores de temperatura e tempo, é realizado o controle de temperatura pelo microcomputador através de um programa especializado, cujo fluxograma é descrito pela figura 3.2. Através deste fluxograma podemos observar o funcionamento do programa de controle realizado pelo microcomputador, onde são comparados os instantes de tempo do vetor tempo, ao instante de tempo do relógio do microcomputador. Quando os instantes são iguais, é transmitido para o controlador o valor da temperatura armazenada no vetor temperatura e o indexador sofre um incremento. Esse ciclo repete-se com a comparação do próximo instante de tempo.

A segunda parte do controle é feita pelo controlador. Após receber o valor da temperatura a ser aplicada, o controlador tem a tarefa de exercer o controle automático da temperatura da estufa, deixando-a equivalente ao referente transmitido. Podemos então isolar os módulos (***g***),(***i***),(***k***),(***m***) e (***j***) (figura 3.1) e defini-los como um subsistema de controle automático.

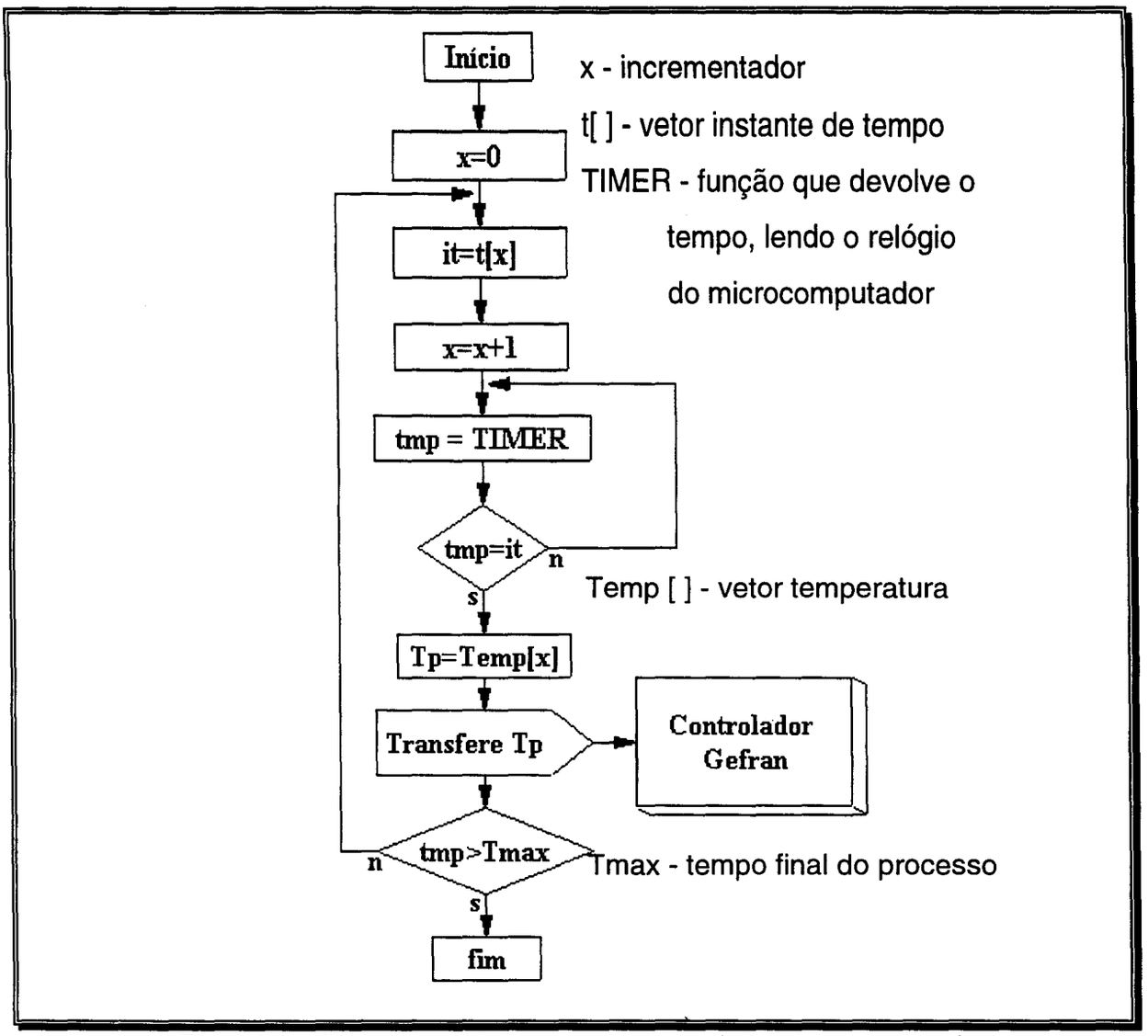
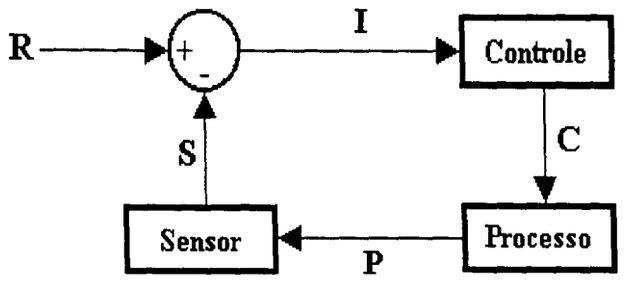


Fig. - 3.2 - fluxograma do programa especializado em controle

Num sistema de controle automático a saída do processo interage diretamente na ação do controle, através da realimentação deste sinal o qual é comparado com a referência desejada. Podemos descrever um sistema de controle automático genérico, através da figura 3.3, onde o controle do processo é realizado pelo sinal **C**, e responde com a saída **P**, medida através do sensor originando o sinal **S**. O sinal **S** é então comparado com o sinal de referência **R**, e a resposta desta comparação é dada ao controle pelo sinal **I**. Caso o sinal **I** não seja nulo, ou seja, caso **S** e **R** sejam diferentes, o controle então utiliza o sinal **C**, de forma a compensar esta diferença e equilibrar o processo com a referência.



R	-	sinal de referência
S	-	sinal do sensor
I	-	sinal de resultado da comparação
C	-	sinal de controle
P	-	saída do processo

Fig. - 3.3 - Sistema de Controle Automático

Aplicando o caso genérico ao nosso sistema, temos o sinal de referência **R**, como o valor de temperatura transmitido para a memória do controlador, que é comparado com o sinal **S** proveniente do sensor, contendo o valor da temperatura da estufa. Caso estes valores sejam diferentes, então esta diferença é compensada pelo sinal **C**, que através do circuito de carga alimentará o aquecedor, incrementando a temperatura da estufa (**P**).

O controle lógico, comparar e analisar **R** e **S**, assim como utilizar a melhor resposta para **C**, é realizado por um programa específico do controlador, que utiliza a técnica de controle PID (proporcional integrativa e derivativa) [6].

O programa, também apresenta a característica de armazenar os últimos valores recebidos, e tenta então interpolá-los linearmente.

A comunicação entre o microcomputador e o controlador, infelizmente é bastante lenta, da ordem de segundos. Portanto, analisando a função 3.5, podemos constatar que devido à demora do processo de comunicação, quando a taxa de aquecimento k é alta o suficiente para que o tempo de espera entre dois parâmetros de temperatura seja menor que o tempo de comunicação, torna-se impraticável a execução da função no processo.

Para contornar esse problema, o microcomputador calcula o tempo de comunicação, e analisa a taxa de aquecimento k. Caso este ultrapasse o seu limite, é acrescido 0,1 ao valor de i (funções 3.5 e 3.3), até que o tempo de espera seja

maior que o tempo de comunicação. Embora o valor de i defina a resolução dos degraus e conseqüentemente uma melhor interpolação da função (na estufa), para as taxas de aquecimento disponíveis não houve diminuição observável à qualidade do controle.

Devido à lentidão do processo de comunicação entre o microcomputador e o controlador, utilizamos os módulos (**e**), (**q**) e (**n**), adicionalmente a fim de evitar ao máximo a comunicação entre os equipamentos. Desta forma fazemos a aquisição da temperatura (que será descrita detalhadamente no item 3.5), por outro método, diminuindo pela metade o tempo de espera da comunicação (não necessitando ler a temperatura através do controlador).

Podemos também, ler a temperatura através do controlador, porém isto acarreta um incremento maior em i (funções 3.5 e 3.3). Caso queiramos desprezar este acréscimo e perder um pouco da qualidade do controle, podemos "ler" a temperatura através do controlador, e deste modo não utilizarmos os módulos (**e**), (**q**) e (**n**).

3.4.1 - O CONTROLADOR

Utilizamos no sistema, o controlador configurável Gefran 3300 fabricado pela Gefran Elettronica. O controlador é definido pelo fabricante como 3300C-D2-2R-0-2-1, retratando desta maneira as opções do equipamento.

Para a comunicação com microcomputadores, o equipamento (de acordo com a opção) é provido de um porto serial RS-485.

O controlador possui um programa especializado em controle, que utiliza lógica PID (proporcional, integrativa e derivativa), o que proporciona "suaves" interpolações à resposta do processo.[6]

Quando interligado à estufa, através de programa especializado, o equipamento é capaz de analisar todas suas características (a nível de controle), se auto-calibrando e configurando.

O controlador é configurado para utilização de sensor térmico tipo PT100, e para operar com degraus de 0,1 ^{0}C. Quando configurado desta maneira, sua faixa de operação é de -99,9 ^{0}C a +299,9 ^{0}C.

3.4.2 - A COMUNICAÇÃO ENTRE AS PORTAS SERIAIS

O microcomputador transfere o valor da temperatura para o controlador, como descrevemos anteriormente. Esta transferência de dados é feita através de uma interface serial, onde ambos aparelhos se interconectam.

A transferência de dados é feita através da conversão de um valor digital em uma onda quadrada, onde os bits são transpostos na forma de sinal elétrico. Desta forma podemos fazer com que dados de um dispositivo sejam transferidos para outro e vice-versa. Nesta onda os valores dos bits são expressos por tensões, sendo que quando a tensão é maior que o limite imposto, significa que o bit tem valor 1, caso contrário 0.

Para que a comunicação seja possível e universal, são impostos padrões e protocolos de comunicação de dados. No caso do sistema utilizado, temos padrões diferentes entre os equipamentos. Tal ocorrência impossibilita a comunicação, e por isso, é necessária a utilização de um conversor de padrões seriais. Esse converte e interfaceia os protocolos diferentes, tornando viável a comunicação entre os equipamentos.

O padrão utilizado pelo controlador é o RS-485, que é um dos padrões seriais mais utilizados no setor de equipamentos industriais, também popularmente conhecido por "loop de corrente". Já o padrão mais usual no universo dos microcomputadores é o RS-232C.

O conversor que utilizamos, é fabricado pela Gefran Elettronica, e denominado de CLB. Sua função é converter, e desta forma interfacear os padrões RS-485 / RS-232C.

Num padrão de comunicação existem vários protocolos para a comunicação. No caso das interfaces seriais, é através desses protocolos que configuramos a forma e a velocidade da onda quadrada, responsável pela comunicação.

O controlador possui seu protocolo de comunicação, configurado de fábrica, sendo impossível alterá-lo. Portanto devemos adequar o protocolo do microcomputador, a fim de viabilizar a comunicação. Por ser o padrão RS-485 transparente ao microcomputador, devido ao módulo CLB, discutiremos apenas o padrão RS-232C.

Configuramos o padrão RS-232C, de acordo com a velocidade com que os dados são transmitidos e recebidos. Esta pode variar de 75 a 115200 bps (bits por

segundo). A velocidade utilizada pelo controlador é de 1200 bps.

A configuração da forma da onda quadrada, descrimina seu formato em relação aos dados que estão sendo transferidos. A figura 3.4 mostra o formato geral de uma onda quadrada de comunicação serial, em função de seu conteúdo. Os bits representados por D0 a D7 compõem o dado transferido, que pode variar quanto ao tamanho, de 5 a 8 bits.

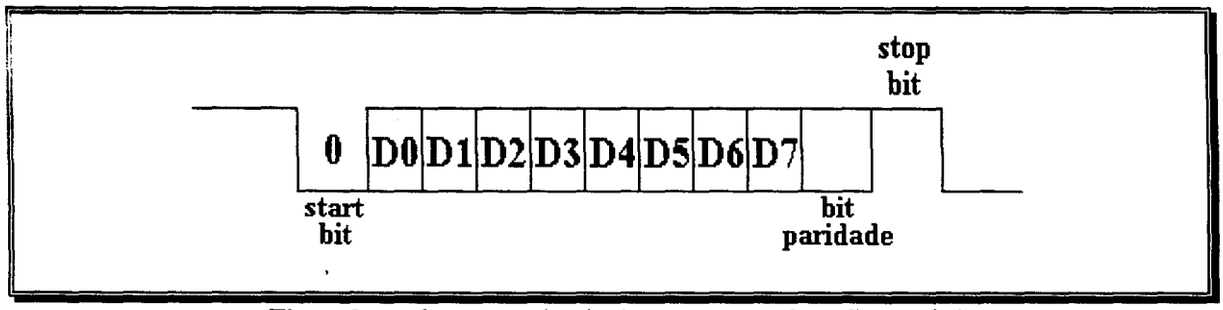


Fig. - 3.4 - formato do dado na comunicação serial

O primeiro bit da onda é denominado "start bit", que indica o começo da onda. Após os bits de dados, encontramos o bit de paridade. Este é de uso facultativo, e sua utilização é recomendada para diminuir a possibilidade de erros durante a comunicação, informando se os bits de dados com nível lógico um, quando somados, formam um número par ou ímpar. Por fim, temos o "stop bit" que marca o final do sinal.

Em nossa configuração utilizamos dados de 8 bits, "stop bit" e paridade par.

Essas informações e outras adicionais, podem ser encontradas na referência 8.

No apêndice A, é descrita a estrutura de pinos do conector DB25, assim como as funções dos sinais que compõem o padrão RS-232C e as tensões dos sinais.

3.4.3 - A ESTUFA

O sistema é flexível, quanto à utilização de estufas. No Grupo, dispomos de duas estufas desenvolvidas especificamente para a realização desse tipo de medidas.

No interior da estufa são alojados os sensores de temperatura, o sistema de aquecimento (resistências), e o compartimento da amostra. O compartimento da amostra é responsável pelo acoplamento mecânico e elétrico da amostra.

Uma das estufas é bem simples, e a outra tem acoplada a si um sistema de vácuo. Dependendo do tipo de medida realizado, não existe a necessidade de utilização de vácuo. Descreveremos a provida de sistema de vácuo, por ser a mais complexa.

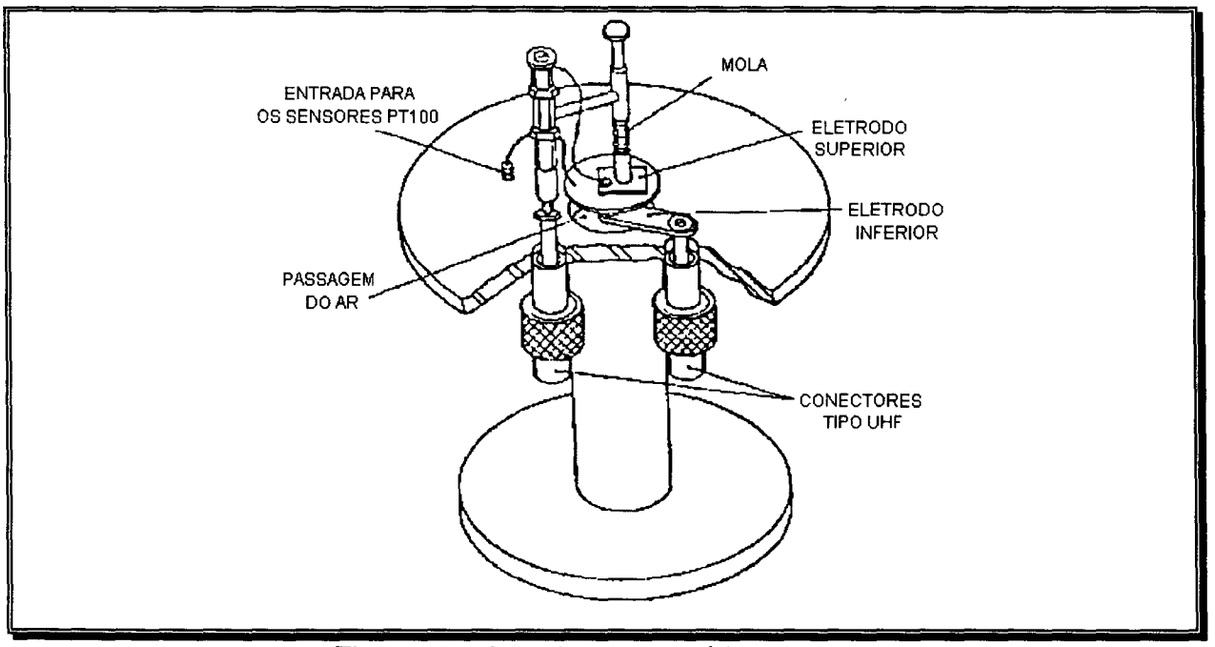


Fig. - 3.5 - Arranjo esquemático da base

A estufa que descreveremos é composta por duas peças principais: a base e a cúpula. A base é a parte inferior da estufa, dentro da qual é montado o compartimento da amostra. Este compartimento é composto por contatos elétricos; um sendo uma placa fixa que suporta a amostra, e o outro um pino que sob a ação de uma mola, pressiona suavemente a amostra, conforme mostrado na figura 3.5. Essa base possui também quatro saídas, duas delas são UHF, provenientes dos contatos elétricos, sendo uma conectada ao eletrômetro e a outra à fonte de tensão; as outras duas são conexões dos sensores térmicos. A base contém ainda, um orifício que permite a realização do vácuo.

A cúpula é a parte superior da estufa, figura 3.6, e tem em seu interior uma

outra cúpula de dimensões menores, na-qual são fixadas quatro resistências com potência de 800 Watts em suas paredes, responsáveis pelo aquecimento. Estas são alimentadas através das saídas laterais tipo UHF com tensão de 220 V. A cúpula apresenta também em sua porção superior duas válvulas, utilizadas para troca de gases e controle de umidade, cuja utilização é mais específica como no caso do trabalho realizado por NOGUEIRA [9].

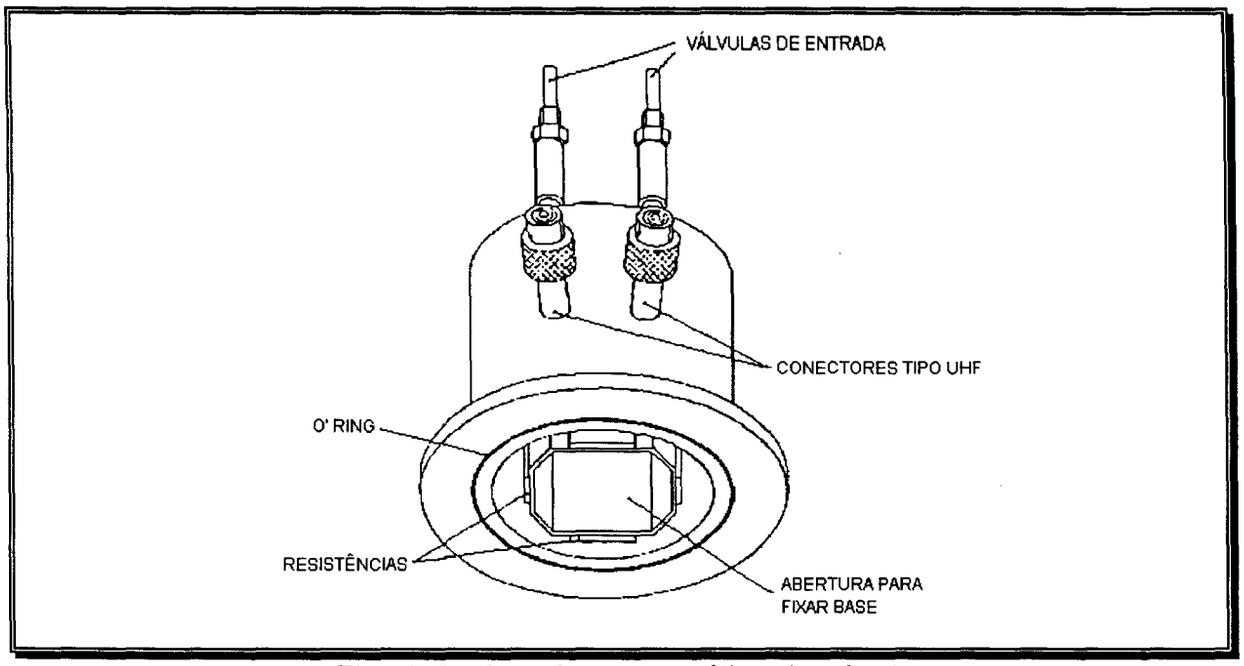


Fig - 3.6 - Arranjo esquemático da cúpula

A seguir na figura 3.7 observamos as duas peças, base e cúpula unidas formando a estufa. Entre a união das duas peças, utilizou-se um "o'ring" de viton, para uma boa vedação sob vácuo.

A estufa é colocada sob um sistema de vácuo, que é constituído basicamente por uma bomba mecânica de baixo vácuo, e uma bomba difusora de alto vácuo, como podemos observar em seu esquema, descrito pela figura 3.8.

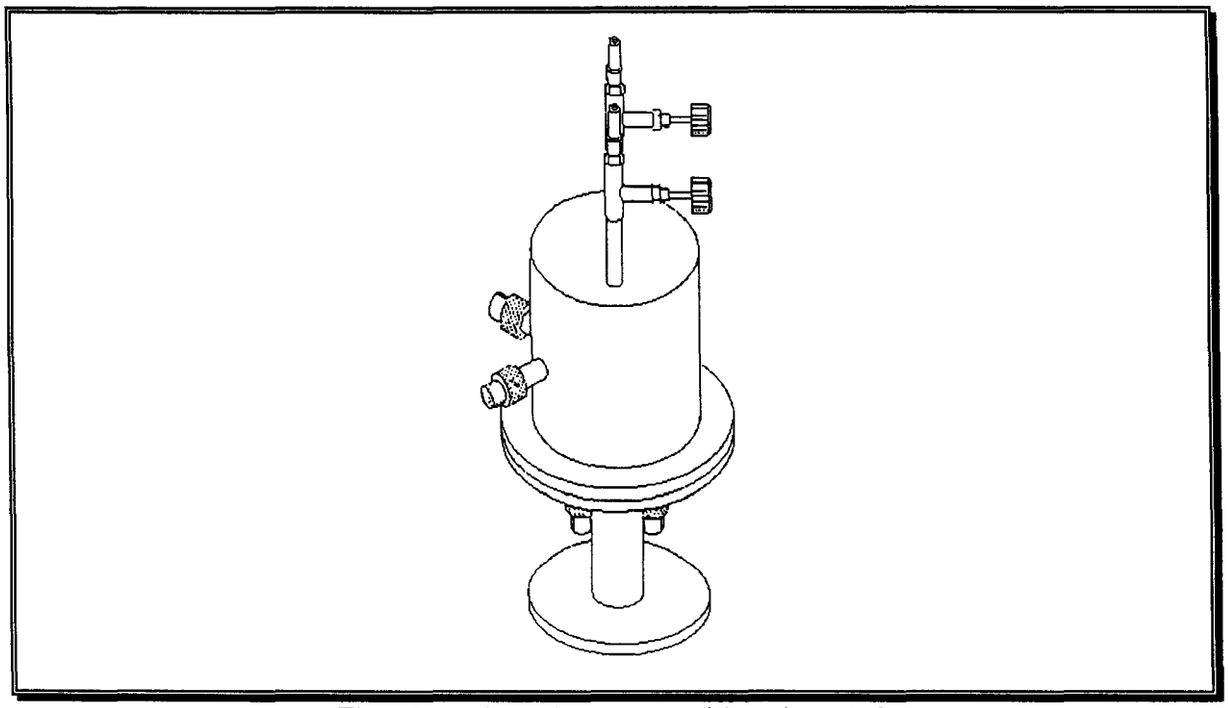


Fig - 3.7 - Arranjo esquemático da estufa

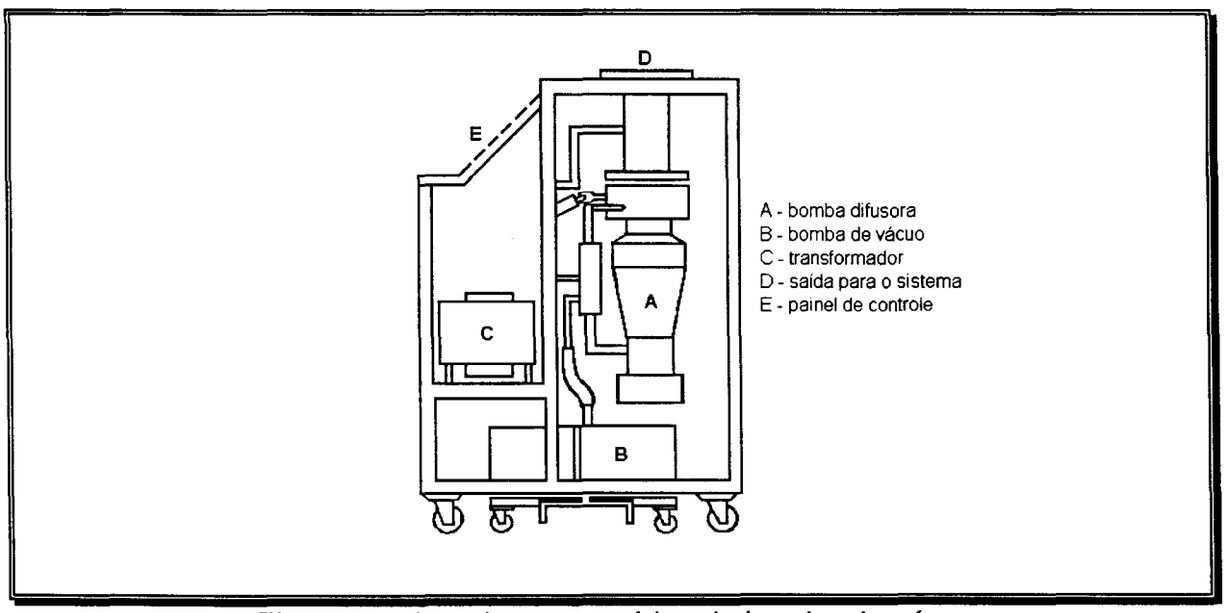


Fig. - 3.8 - Arranjo esquemático da bomba de vácuo

3.4.4 - O SENSOR DE TEMPERATURA

Dentre os materiais utilizados para a fabricação de sensores termoresistivos, a platina apresenta as melhores características dentro de uma larga faixa de temperatura. Baseado nas características termoresistivas da platina, são produzidos industrialmente sensores de temperatura, com excelente qualidade, linearidade e resolução.[10] A figura 3.9, apresenta um corte lateral de um sensor termoresistivo de platina genérico, produzido industrialmente.

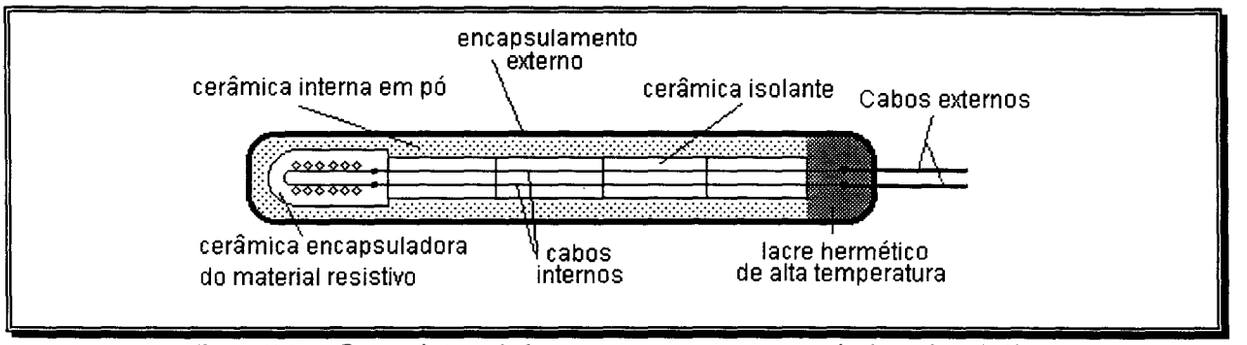


Fig. - 3.9 - Corte lateral de um sensor termoresistivo de platina

Utilizamos em nossos sistema dois sensores (idênticos) termoresistivos de platina, produzidos industrialmente, padrão PT100. Sua denominação vem do símbolo químico da platina Pt, e a numeração 100, porque a 0^0C a resistência lida é de 100Ω.

O sensor PT100 obedece aos padrões estabelecidos pela norma DIN 43760, e a sua faixa de operação é de -50^0C a 450^0C (PT100 utilizado pelo sistema). Na figura 3.10 é apresentada a curva resistência em função da temperatura, para o padrão PT100. E através dela podemos constatar o alto grau de linearização do padrão PT100, principalmente na faixa de operação do sensor utilizado.

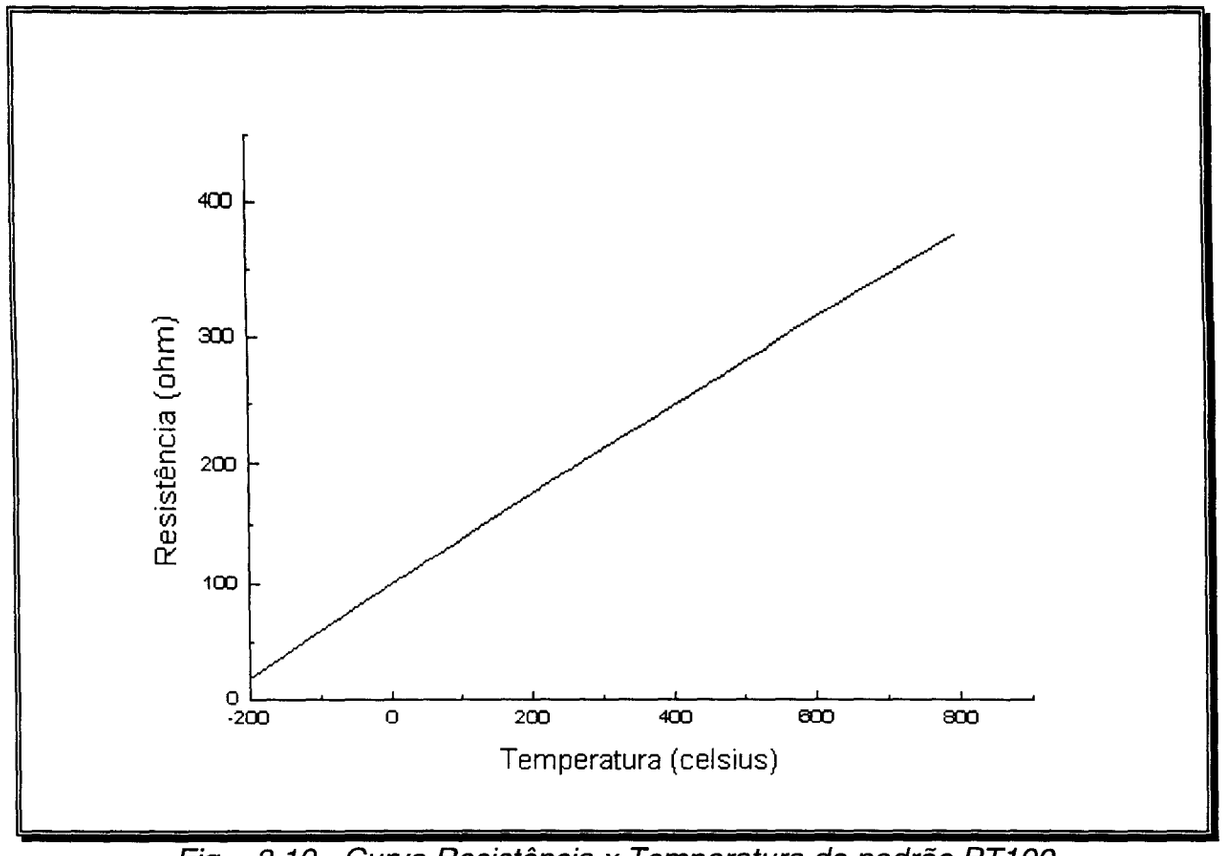


Fig. - 3.10 - Curva Resistência x Temperatura do padrão PT100

3.4.5 - O ACIONADOR DE CARGA

A carga que alimenta o aquecedor da estufa, não é drenada do controlador, pois este possui apenas uma saída de nível, que serve para acionar diversos tipos de "drivers" (devido à generalidade de uso do controlador). Para aquecer a estufa, utilizamos resistências de potência, alojadas em seu interior, que dissipam energia térmica. As resistências que utilizamos possuem potência superior a 500 Watts, são alimentadas diretamente pela rede elétrica.

O controle do acionamento e o cálculo da potência que será liberada para alimentar o aquecedor é feito pelo controlador, mas a liberação desta potência é realizada por um circuito auxiliar denominado Acionador de Carga (módulo (*i*) da figura 3.1), cuja função é ligar e desligar o aquecedor da rede elétrica. Desta forma o Acionador de Carga funciona como uma chave, controlada pela saída de nível do controlador.

O Acionador de Carga, cujo esquema constitui a figura 3.11, foi implementado a partir de um circuito integrado dedicado "Zero Volt Switch" -CA3079, um fotoacoplador 4N33, um triac TIC226D e alguns resistores e capacitores.

O controlador quando necessário altera o estado de sua saída de nível lógico e este estado é enviado pelo fotoacoplador, para o circuito "Zero Volt Switch", acionando desta forma a chave liga e desliga do Acionador de Carga. A função do fotoacoplador é o isolamento elétrico entre os equipamentos

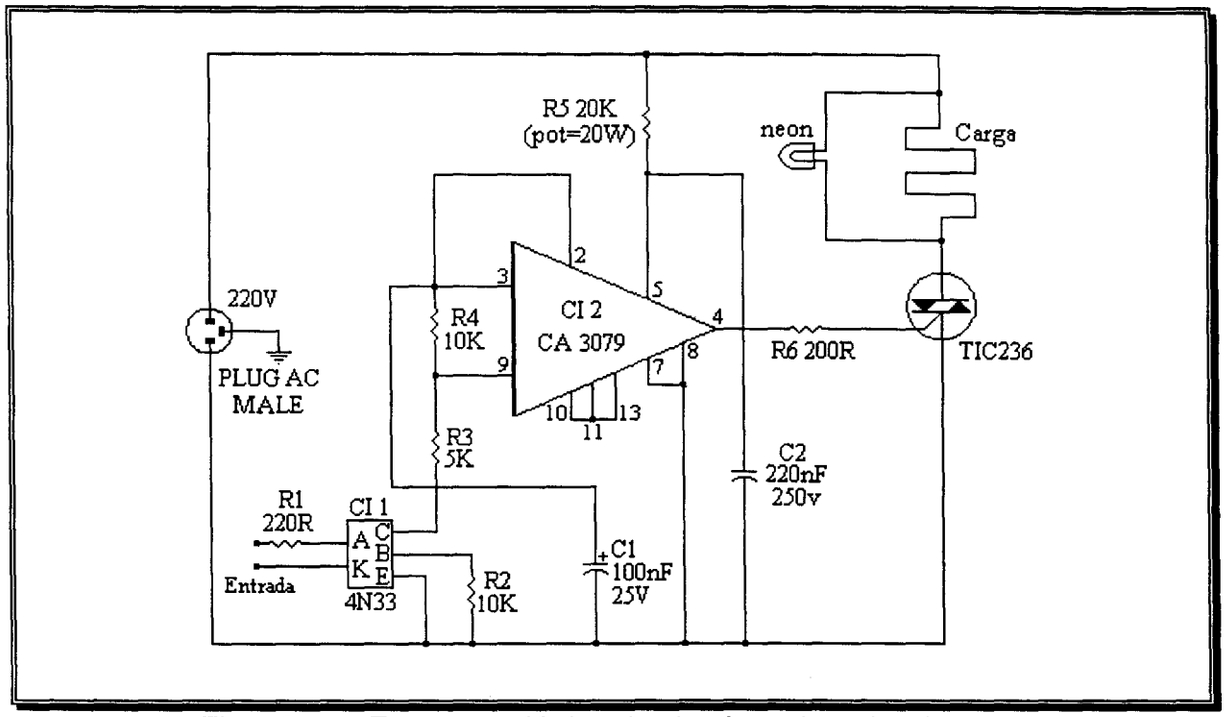


Fig. - 3.11 - Esquema elétrico do circuito acionador de carga

O CA3079 é alimentado diretamente pela rede elétrica (220V - 60Hz) e, utilizando seus circuitos internos, gera pulsos de 120Hz com aproximadamente 100μs de largura, toda vez que a senóide da rede se anula. Estes pulsos são liberados, toda vez que a saída de nível lógico do controlador assume nível alto (5V), para ligar o aquecedor à rede elétrica, através do triac. Assim, quando chaveado, o aquecedor é ligado sincronicamente com a senóide da rede, ou seja, quando esta se anula.

Chaveando a alimentação do aquecedor somente quando a senóide da rede elétrica passa pelo zero volt, diminuímos o ruído sobre a tensão da rede. Caso o chaveamento fosse aleatório, em qualquer posição da senóide, poderia gerar

transientes muito altos, devido às variações bruscas de tensão. Quanto maior o valor absoluto da tensão da senóide no instante do chaveamento, maiores poderão ser os picos de transientes. Deste modo diminuímos as possibilidades de interferência nas medidas, que poderiam ser provocadas por esses transientes.

O circuito utiliza uma lâmpada de neon, ligada em paralelo com a carga, como indicador do chaveamento, que acende quando o aquecedor é ligado.

3.5 - AQUISIÇÃO DE DADOS

Enquanto é executado o controle de temperatura da estufa, é realizada a segunda tarefa do sistema: a aquisição e o armazenamento de dados.

No procedimento de inicialização do sistema, além de entrar com os parâmetros de controle, o usuário também define o intervalo de tempo entre as aquisições. Deste modo pode-se prover o número de pontos das curvas resultantes, além de definir suas resoluções (em função do tempo). Um outro parâmetro necessário é a entrada do fundo de escala do eletrômetro, a partir do qual é calculado o valor da corrente através do sinal de referência proveniente do eletrômetro.

Após a inicialização do sistema, procedimento onde são inseridos os parâmetros iniciais, começa então o processo de medidas. Por um lado é realizado o controle da temperatura da estufa, mediante a definição do usuário, e por outro lado, simultaneamente com o controle, são realizadas as medidas através da aquisição e armazenamento de dados.

No nosso sistema, são realizadas três medidas: o instante de tempo, a temperatura da estufa e a corrente proveniente do sistema TSC. Estas medidas formam um sistema vetorial de coordenadas tridimensional inter-relacionadas.

O instante de tempo, em que são realizadas as medidas, é coletado a partir do temporizador do ambiente Windows, que por sua vez, é uma extensão relativamente simples da lógica do temporizador existente no "hardware" do IBM-PC e no BIOS (Sistema Básico de Entrada e Saída de Dados) em ROM (Memória somente para leitura). O BIOS configura o circuito integrado Intel 8259 para gerar a interrupção 08H do "hardware". Desta forma são gerados intervalos de 54,925 ms, a

partir dos quais o temporizador do ambiente Windows calcula e regulamenta seu relógio.[2]

O instante de tempo é coletado e armazenado aproximadamente no mesmo instante em que são coletadas a corrente e a temperatura. O usuário pode definir a unidade à qual deseja utilizar o instante de tempo, que pode ser minuto ou segundo.

A medida da temperatura é feita a partir de um sensor termoresistivo de platina PT100, alojado estrategicamente no interior da estufa. O PT100, varia sua resistência em função da temperatura, gerando um parâmetro de referência através do qual pode ser calculada a temperatura da estufa. Fazemos a aquisição dessa referência através de um conversor de sinais analógicos em digital, ao qual é convertido o sinal analógico proveniente do sensor, em digital, de forma que esse possa ser lido e interpretado pelo microcomputador. A unidade utilizada para as medidas de temperatura é o celsius.

O conversor de sinais A/D é capaz de interpretar a diferença de potencial de um sinal elétrico. Como o sensor PT100 possui a característica de variar a sua resistência em função da temperatura (item 3.4.4), utilizamos então um circuito capaz de transformar a referência de temperatura dada pela variação da resistência do PT100, numa referência fundamentada na variação do diferencial de potencial (módulo p da figura 3.1).

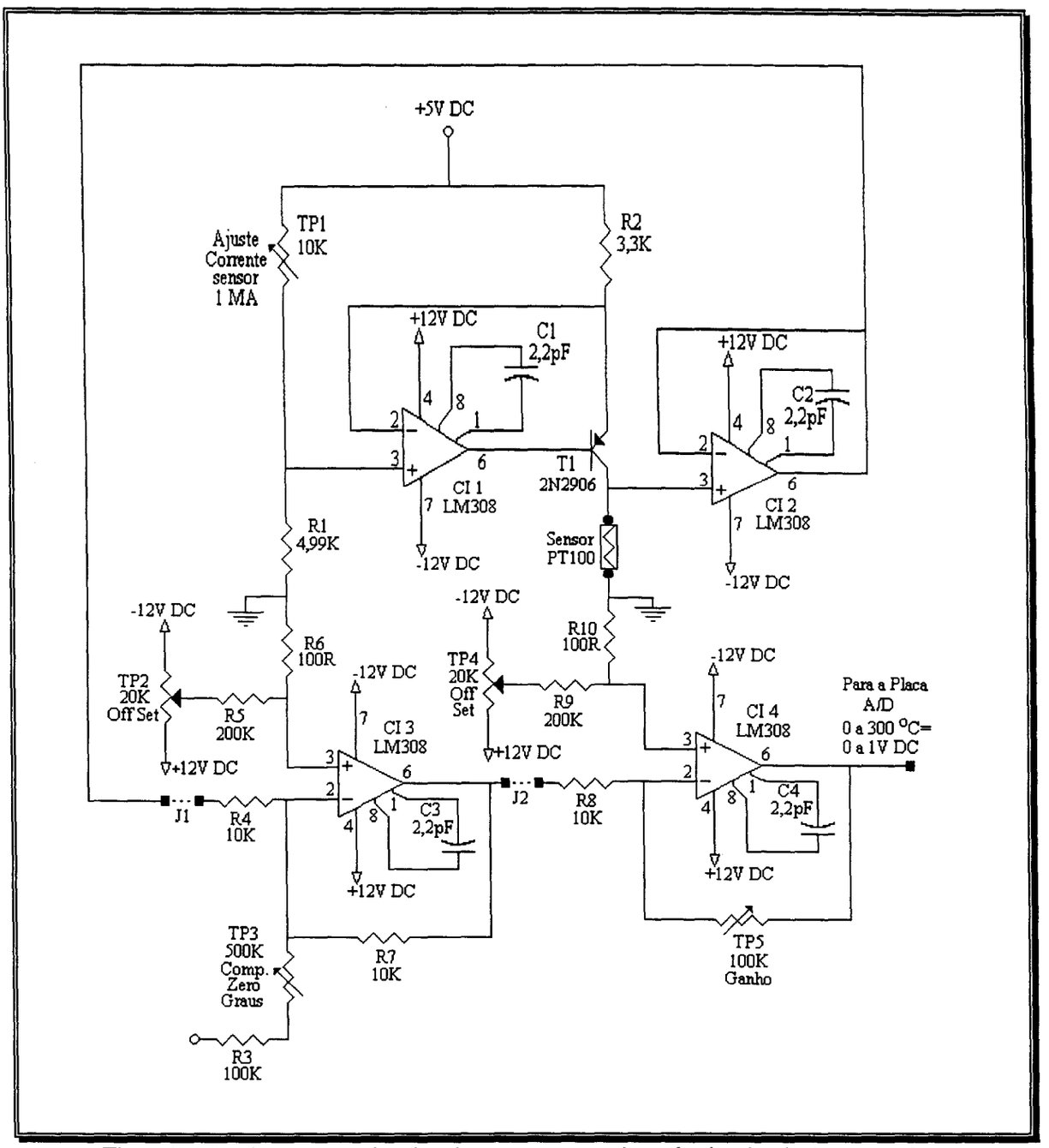


Fig. - 3.12 - Esquema do circuito conversor da referência de temperatura

A figura 3.12 apresenta o esquema elétrico desse circuito. Ajustamos o circuito de forma que a variação de temperatura de 0 a 300 ^{0}C no sensor, provoque na saída do circuito um sinal de referência variando de 0 a 1 V proporcionalmente à variação da temperatura.

Antes de executar a conversão do sinal, por este ser de amplitude inferior à amplitude do conversor A/D, fazemos uso de um amplificador de sinais, de modo que atinja a amplitude necessária para a conversão.

Utilizamos a placa PC Lab Plus da National Instruments para a aquisição de

dados. Trata-se de uma placa multifuncional para processamento de sinais, contendo conversores de sinais analógicos em digitais, digitais em analógicos e amplificadores de sinais entre outras funções. A PC Lab Plus será descrita mais detalhadamente posteriormente. A amplificação do sinal proveniente do PT100 é feita pelo amplificador interno da PC Lab, descrito pelo módulo (**e**) do diagrama da figura 3.1.

O sistema também realiza a medida da corrente da amostra sob a técnica TSC. A figura 3.15 mostra a conexão da amostra na qual é realizada a medida. Como o sinal proveniente da amostra é excessivamente baixo, da ordem de picoamperes, utilizamos um eletrômetro para efetuar a medida. O eletrômetro utilizado possui uma saída analógica, que possui um sinal de referência contendo o valor medido. O microcomputador executa a aquisição da referência a partir do conversor analógico-digital da placa PC Lab Plus. Como a amplitude do sinal de referência é inferior à amplitude de entrada do conversor, utilizamos o amplificador interno da placa (módulo **f** da figura 3.1), para aumentar a amplitude e conseqüentemente a resolução lida.

Após coletar, o microcomputador relaciona o valor lido com sua amplitude máxima, e então multiplica-o pelo fundo de escala do eletrômetro dado anteriormente pelo usuário, e é então obtido o valor da corrente.

Para polarizar a amostra, utilizamos uma fonte de alta tensão em regime DC, cuja conexão é também observável na figura 3.15. A fonte utilizada pelo sistema é uma fonte de tensão DC, modelo 610C, fabricada pela Trek Inc., capaz de fornecer tensões positivas e negativas variando de 0 a 10.000 Volts.

3.5.1 - A PLACA PROCESSADORA DE SINAIS

Os sinais elétricos utilizados no sistema exprimem, quando medidos, quantidades de um fenômeno elétrico. Denominamos os sinais que caracterizam fenômenos elétricos de sinais analógicos. Os microcomputadores em geral, são fundamentados em eletrônica digital, e deste modo operam com sinais digitais.[11]

Diferentes dos sinais analógicos, os sinais digitais não expressam diretamente fenômenos elétricos, mas sim valores discretizados através dos sinais

analógicos. Na eletrônica digital, o sinal digital mais típico é o binário, que é definido como uma tensão ou corrente que compõe informações na forma de mudança de estado entre dois tipos de estados. Estes estados são denominados de lógico 0 e de lógico 1. Para sinais de tensão, o estado lógico 0 é tipicamente o terra, ou 0 Volts, e o estado lógico 1, quando a tensão é maior que um limite situado um pouco abaixo da máxima tensão. O mesmo ocorre para a corrente, onde o estado lógico 0 é a ausência de corrente, e o estado lógico 1 quando a corrente ultrapassa um limite pré-determinado.[12]

Os microcomputadores (digitais) são incapazes de operarem com sinais analógicos, portanto é necessária a conversão de sinais. Como conversor de sinais fazemos uso de um conversor A/D (analógico-digital), que realiza a conversão de sinal analógico em sinal digital, de forma que tornamos possível o processamento dos sinais provenientes do sistema.

Utilizamos a placa multifuncional para processamento de sinais PC Lab Plus, fabricada pela National Instruments, que é acoplada a um dos "slots" (barramento 8 bits) do microcomputador, para o tratamento dos sinais provenientes do sistema. A PC Lab Plus contém um ADC (conversor A/D) de aproximação sucessiva de 12 bits, que pode ser configurado com oito canais simples, ou quatro canais diferenciais. Contém também dois DAC (conversor D/A) de 12 bits com saída de tensão e 24 linhas de entrada e saída digitais com lógica TTL, além de um amplificador de instrumentação (entrada analógica) e um temporizador (timming) de aquisição de dados.

No sistema utilizamos os módulos de entrada analógica (ADC) e amplificação de sinal da PC Lab Plus. Deste modo amplificamos os sinais analógicos, provenientes do sistema, e os convertemos em digitais, possibilitando ao microcomputador interpretar as medidas dos fenômenos elétricos, provenientes do sistema (sinais analógicos), e processá-las e armazená-las. Observando o diagrama do sistema dado pela figura 3.1, os módulos (**e**), (**f**) e (**c**) expressam na placa PC Lab Plus, respectivamente, os circuitos amplificadores (**e**) e (**f**) e o conversor (**c**).

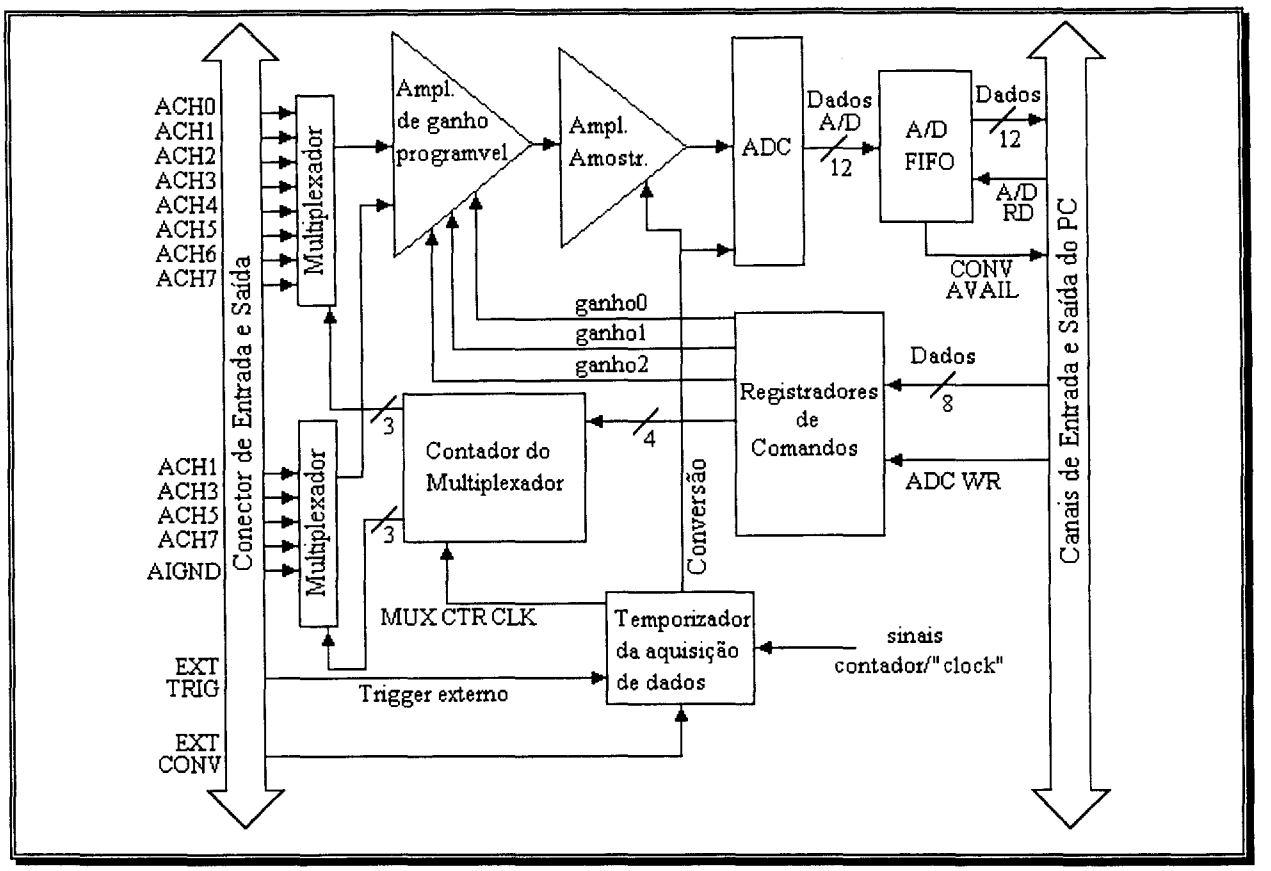


Fig. - 3.13 - Diagrama esquemático da entrada analógica da PC Lab Plus

A entrada analógica da Pc Lab Plus, é descrita pela figura 3.13, e consiste de duas entradas analógicas multiplexadas CMOS, um amplificador com ganho programável por "software", um ADC de 12 bits, e uma memória de 12 bits do tipo FIFO (first in first out - primeiro dado a entrar é o primeiro a sair).

O primeiro multiplexador contém os oito canais de entradas analógicas (canal 0 a 7 - ACH0 - ACH7). O outro multiplexador é conectado aos canais 1,3,5 e 7 para o modo diferencial. A entrada dos multiplexadores contém proteção para tensões superiores a ±45 V, com o circuito ligado ou desligado.

O amplificador de ganho programável, gera um sinal analógico amplificado, que é então amostrado por amplificador de amostragem e então convertido, a fim de aumentar a precisão e a resolução da medida. O ganho do amplificador de instrumentação é selecionado por "software". Os ganhos disponíveis pelo amplificador são de: 1, 2, 5, 10, 20, 50 e 100. A tensão medida pelo amplificador de instrumentação é dada por

$$V_m = [V_{in+} - V_{in-}]G \qquad (3.6)$$

onde G é o ganho, V_m a tensão medida, V_{in+} a tensão positiva de entrada e V_{in-} a tensão negativa de entrada. A figura 3.14 expressa o diagrama do amplificador de instrumentação e suas tensões.

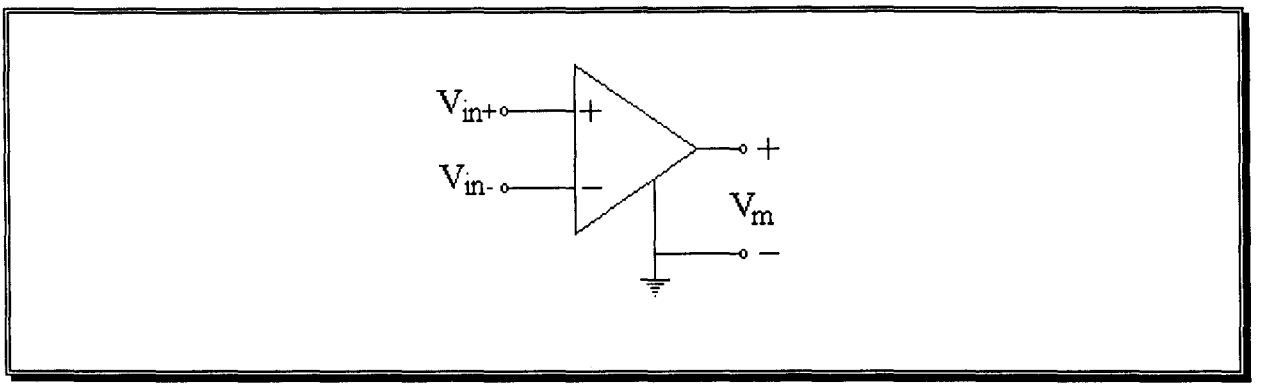


Fig. - 3.14 - Tensões do Amplificador de Instrumentação

O ADC, opera por aproximação sucessiva, e possui uma resolução de 12 bits, de modo que utiliza uma escala digital de 4096 (2^{12}) degraus diferentes [9] para conversão dos sinais analógicos. Esta resolução provém uma palavra de 12 bits, que representa o valor medido (sinal analógico) da tensão de entrada do ADC. O conversor sozinho opera com tensão de entrada entre 0 a +5V. A circuitaria adicional faz com que esta entrada seja de ±5V ou 0 a 10V.

Quando uma conversão é realizada pelo ADC, então é transferido o resultado desta conversão para a memória A/D FIFO. A memória A/D FIFO possui 16 bits de comprimento de palavra e é capaz de armazenar 512 palavras. Deste modo a memória A/D FIFO funciona como um "buffer" do ADC e proporciona então dois benefícios. Primeiro, a qualquer tempo que uma conversão A/D é realizada, o valor é gravado na memória A/D FIFO, para ser lido posteriormente, e então o ADC está disponível para uma nova conversão. Segundo, a memória A/D FIFO pode armazenar até 512 conversões antes que alguma conversão seja perdida, liberando assim algum tempo extra (512 vezes o tempo de uma aquisição) para que o software possa transferir os dados para a memória do microcomputador. A memória A/D FIFO envia um sinal quando possui alguma conversão ainda não lida pelo

microcomputador.

A razão do tempo da aquisição (número de amostras por segundo) realizada pela PC Lab Plus é determinada pelo período de conversão do ADC, o tempo de amostragem e o tempo de chaveamento e configuração dos circuitos, além também do tempo de programação do amplificador de instrumentação.

Dependendo do ganho utilizado no amplificador de instrumentação, temos uma razão de tempo de aquisição diferente. No modo de entrada simples, utilizado pelo sistema, a razão é de 83.300 amostras por segundo quando o ganho é 1, e de 71.400 amostras por segundo para os demais ganhos.

A tabela 3.1 expressa os limites das tensões de entrada da PC Lab Plus para diferentes programações de ganho do amplificador de instrumentação, quando utilizamos o modo de tensão unipolar, e a tabela 3.2 para o modo de tensão bipolar (segundo manual da National Instr.).

Ganho	Limite da tensão de entrada
1	-5V a 4,99756V
2	-2,5V a 2,49878V
5	-1,0V a 0,99951V
10	-500mV a 499.756V
20	-250mV a 249,878V
50	-100mV a 99,951V
100	-50mV a 49,975mV

Tabela - 3.1 - Limites da tensão de entrada pelo ganho, no modo bipolar

Ganho	Limite da tensão de entrada
1	0V a 9,99756V
2	0V a 4,99878V
5	0V a 1,99951V
10	0mV a 999.756mV
20	0mV a 499,877mV
50	0mV a 199,951mV
100	0mV a 99.975mV

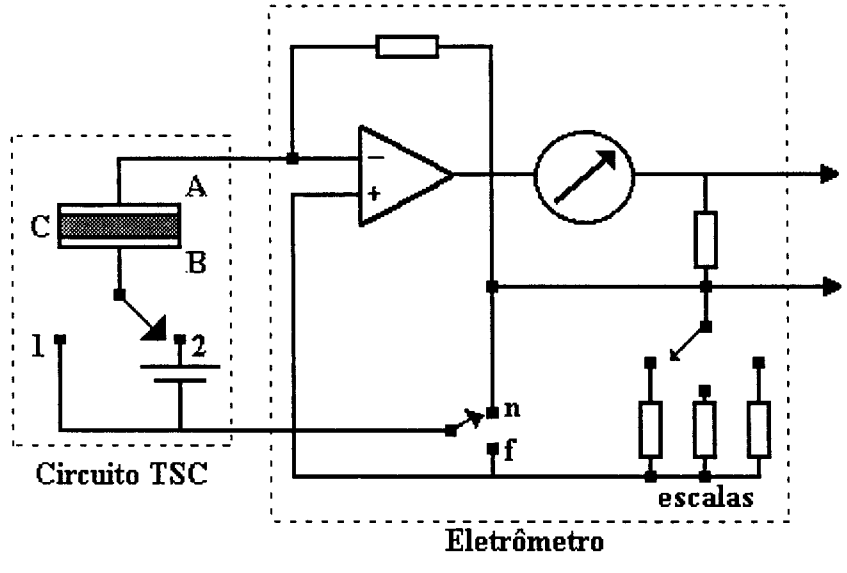
Tabela - 3.2 - Limites da tensão de entrada pelo ganho, no modo unipolar

3.5.2 - O ELETRÔMETRO

As amostras utilizadas no sistema induzem correntes demasiadamente baixas, da ordem de dezenas de picoamperes. Para medir tais grandezas, deve-se utilizar amperímetros de altíssima sensibilidade. Encontramos o eletrômetro como o instrumento que melhor se adapta ao nosso propósito, e optamos pelo modelo 610C fabricado pela Keithley Instruments, que é capaz de medir corrente de até 10 fA com precisão de 0,1%. Na figura 3.15 é representado o esquema de ligação do eletrômetro com o sistema TSC.

No painel traseiro deste equipamento, encontramos uma saída auxiliar em tensão, que contém uma referência ao valor medido pelo instrumento, cujo nível varia de 0 a 3V, correspondendo do mínimo ao máximo da escala utilizada pelo eletrômetro.

Este sinal é enviado à entrada da placa conversora conectada ao micro, de forma que o micro possa efetuar a aquisição de dados e conseqüentemente o armazenamento e processamento da medida.



A,B - Eletrodos
C - Amostra

Fig. - 3.15 - Interligação entre o eletrômetro e um circuito TSC

CAPÍTULO IV - INTERFACE GRÁFICA COM O USUÁRIO (GUI)

4.1 - INTRODUÇÃO

Este capítulo tem como objetivo apresentar alguns conceitos da tecnologia GUI de interfaceamento homem - máquina, assim como um pouco de sua história, que está diretamente ligada à história da própria computação. A importância deste interfaceamento é vital, pois é através dele que o homem se comunica com o computador e o utiliza como ferramenta de trabalho.

A interface gráfica com o usuário (GUI) tem demonstrado ser uma tecnologia bastante eficiente no relacionamento entre o homem e a máquina, satisfazendo todos os níveis de usuários, mostrando encontrar algumas respostas para as questões sobre a forma de utilização do computador. Com a utilização de GUI, a curva de aprendizado do usuário iniciante é menor, e o aproveitamento dos recursos do computador pelo usuário experiente ou "power user" é maior.[1]

4.2 - CONTEXTO HISTÓRICO

A história da interface entre homem e máquina está diretamente ligada à história da própria computação. A importância desta interface está simplesmente no fato de que é ela que ditará o modo com que iremos utilizar o computador. É através dela que o homem se comunica com o computador e utiliza-o como ferramenta de trabalho.

O fundamental para o desenvolvimento da interface homem - máquina é a padronização [1]; e um dos motivos que geraram os padrões foi a dificuldade de transportar programas desenvolvidos em uma máquina para outra. Devido a este fato, vamos começar nossa história com um enfoque na portabilidade.

4.2.1 - HISTÓRIA DA PORTABILIDADE[1]

No início dos dias da computação até o final da década de 50, não existia nenhuma discussão sobre a funcionalidade do computador, "hardware" ou "software". Um computador era um produto que se "ligava" e "desligava" quando se desejava executar um trabalho. As tarefas destinadas ao computador eram geralmente muito específicas, como folhas de pagamento ou controle de inventário.

Com o desenrolar do tempo, usuários começavam a ficar um pouco mais sofisticados e a demanda por computadores começou a se desenvolver.

No começo da década de 60, novas companhias entravam no mercado oferecendo apenas componentes de um sistema, como terminais ou unidades de disco. Para obterem sucesso, estes novos vendedores deveriam descobrir como conectar no computador, e assim eles tinham que aprender sobre interfaces. Os fabricantes do computador sabiam que os novos vendedores precisavam desta informação, e eles também sabiam que estes novos vendedores tinham potencial de competição. Por isso esta informação era "escondida". Não obstante, as informações sobre como as portas de entrada e saída (I/O) ou o sistema operacional funcionavam, tornavam-se conhecidas. Nesta época, também, os usuários estavam começando a desenvolver suas próprias aplicações, que eram específicas para seus negócios. Para isso eles tinham que ter informações sobre o funcionamento interno do sistema operacional e seus dispositivos de entrada e saída de dados.

Deste modo, os usuários começaram a ter mais conhecimento sobre a máquina, e eles então começaram a exigir mais da funcionalidade do computador. Assim aumenta a demanda sobre computadores e começa a surgir concorrentes na fabricação de computadores, antes praticamente exclusividade da IBM.

A esta altura, não existia nenhum padrão entre os equipamentos. Cada fabricante tinha um padrão, e até mesmo entre os modelos de máquinas de um mesmo fabricante, havia padrões completamente diferentes. O usuário ficava numa situação bastante incômoda, suas aplicações eram perdidas quando necessitava mudar ou melhorar seu computador.

Os primeiros passos para a padronização, apareceram quando as linguagens de programação padronizadas começaram a surgir. FORTRAN, COBOL e BASIC foram introduzidas, e a maioria dos fabricantes de computador tinham uma versão

delas para suas máquinas. Na teoria, um programa escrito em FORTRAM para um Honeywell ou Univac, poderia ser recompilado e montado para ser executado numa máquina IBM ou Burroughs (somente na teoria). Isto não ocorria na prática. Os programas escritos em uma linguagem para uma máquina, não compilavam em outra máquina.

Isto se devia à falta de um padrão geral, e da falta de desenvolvimento dos sistemas operacionais, e do poder computacional das máquinas. Estes fatos se desenrolaram por décadas e ainda não estão absolutamente resolvidos (em alguns casos).

No final dos anos 70, começaram a surgir os microcomputadores. No começo estes eram apenas "kits", mas brevemente foram surgindo modelos completamente montados de fábrica, como os modelos Commodore e Tandy. Rapidamente o microcomputador começou a proliferar, e mais e mais modelos apareciam.

Estas máquinas, em geral, ofereciam um simples sistema operacional e um interpretador BASIC. Como sempre o BASIC de uma máquina fatalmente não era completamente compatível com o BASIC de outra. Devido à falta de memória e poder computacional, e a vagarozidade destas máquinas, a maioria dos programas eram escritos em linguagem de máquina, o que acarretava numa total incompatibilidade.

No começo do anos 80, a IBM lança o PC. Ele era o maior microcomputador disponível e tinha a mais poderosa UCP (Unidade Central de Processamento). Por ser da IBM, muitos usuários interessados em investir em microcomputadores, agora se sentiam seguros. O resto da história todos nós conhecemos; existem PCs em qualquer parte do "universo".

O PC abriu um novo capítulo na computação em muitos sentidos. O mais importante de todos, foi que ele tornou-se um padrão comum. A tecnologia utilizada não era difícil, e dúzias de companhias copiaram seu projeto (atualmente este número é da ordem de centenas). Como o PC se infiltrou em escritórios, laboratórios, indústrias, lojas e casas pelo mundo todo, isto conduziu o primeiro sistema operacional estável. O que concedeu aos desenvolvedores de "software" algo que não existia antes - uma interface binária estável.

A facilidade de transportar uma aplicação gerada numa máquina IBM para uma máquina Compaq sem recompilação foi rapidamente adotada pelos usuários.

Isto acarretou em milhões de novos usuários de computador no mercado, e com eles a intolerância para a dificuldade. Eles não queriam saber sobre sistemas operacionais, compiladores, "megabytes", "bits" ou qualquer outra coisa que não iriam utilizar, para apenas operarem suas aplicações.

Neste ponto nasceu a verdadeira portabilidade, embora sendo apenas para uma categoria de máquina. Não obstante, esse fato mostrou o caminho, e armou os usuários contra a política não padronizadora dos fabricantes.

Em meados da década de 80, a Apple lançou o computador Macintosh, com a frase "O computador para o resto de nós". Os pesquisadores da Apple, ponderaram muito bem as necessidades dos usuários, e partindo da idéia pioneira que vinha sendo pesquisada, mais notavelmente pelo Xerox PARC (Palo Alto Research Center). Deste modo a Apple popularizou a noção de que o computador poderia ser utilizado por leigos.

O Macintosh, nos forneceu uma interface bastante simples e interativa para o usuário. Nesta época ainda não existia a denominação GUI, o termo surgiu posteriormente.

Também no meio dos anos 80, um professor universitário e alguns amigos começaram uma companhia que desenvolveria um computador especial para desenvolvedores de "software". Este computador tornou-se conhecido como "workstation", e usava o novo sistema operacional chamado de UNIX, que tinha sido desenvolvido pela AT&T. Essa companhia se auto intitulou Sun Microsystems.

Brevemente, as workstations começaram a se difundir e estavam presentes nas companhias desenvolvedoras de "software", nos CPDs de muitas companhias grandes e em quase toda universidade. Muitas coisas interessantes começavam a ser desenvolvidas nessas "workstations". Novas aplicações computacionais estavam sendo criadas, e uma nova visualização gráfica de programas começava a emergir.

Com as "workstations", muitas aplicações disponíveis ao mesmo tempo para muitos usuários, tornavam-se populares. Em pouco tempo um programador precisaria de dois ou mais computadores ao mesmo tempo, ou então que um proporcionasse a oportunidade de compilar um programa e escrever um texto ao mesmo tempo. Como as duas aplicações estariam em aberto, isso só seria possível com um sistema de janela.

Brevemente surgiram ambientes em que apareciam 2, 3 ou 4 janelas na tela

do computador, rodando várias aplicações ao mesmo tempo. Nascia então a era da barata e eficaz multitarefa que veio com a "workstation", e sua interface com o usuário com janelas na tela.

Porém, um ambiente com janelas não surgiria sem custo. Camadas extra de "software" deveriam ser postas entre o usuário e o sistema operacional, a fim de proporcionar adequação ao ambiente de janelas. Podemos observar essas camadas extra no diagrama da figura 4.1, que apresenta o UNIX com e sem GUI.

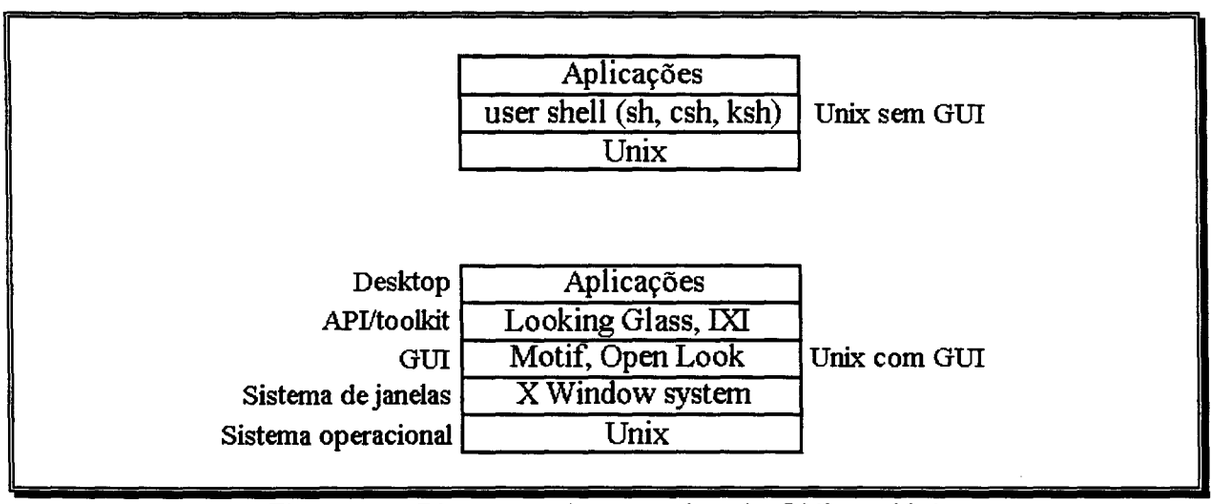


Fig. - 4.1 - Diagrama de camadas do GUI em Unix

O código utilizado para o gerenciamento dessas janelas começou a ficar demasiadamente grande. Com a observação de que cerca de 80 % do código utilizado numa aplicação era destinado ao gerenciamento das janelas, surgiu a necessidade de um sistema com código mais reutilizável. Assim os pesquisadores começaram a utilizar e logo foi adotada a programação orientada a objetos (OOP).

Seguindo a evolução, surgem as redes de comunicação de dados, que possibilitavam esta comunicação entre as máquinas (Macintoshes, Pcs, workstations e mainframes).

Dentro deste meio nasce então a GUI, interface gráfica com o usuário, e os conceitos de padronização do "software" e de independência de "hardware".

4.2.2 - *HISTÓRIA DA GUI*[1]

Sistemas com interfaces gráficas com o usuário não são uma idéia nova. Eles foram pela primeira vez antecipados por Vannevar Bush num artigo escrito em 1945. Ivan Sutherland fez um esboço no começo dos anos 60 em sua tese de graduação.

A Xerox estava pesquisando sobre interfaces gráficas com o usuário no "Palo Alto Research Center" (Xerox Parc) por toda a década de 70. As raízes históricas da GUI na Xerox são anteriores aos seus precoces trabalhos resultantes no Smalltalk e no Star. Quase todas as características agora desejadas de um sistema de janelas, estavam disponíveis no Xerox Star, que foi lançado em abril de 1981. Este tinha uma arquitetura fechada e somente executava pacotes desenvolvidos pela Xerox. Os usuários eram forçados a comprar o pacote todo, "software" e "hardware", da Xerox. O Smalltalk era uma aproximação do que hoje nós consideramos GUI. O Star foi a primeira implementação completamente satisfatória do que é hoje considerado GUI.

O Star foi sistema observado por Steve Jobs da Apple, durante sua visita (atualmente famosa) ao Xerox PARC, que inspirou o desenvolvimento do Apple Lisa, e depois do popular Apple Macintosh. Todavia, sempre consideramos que o Xerox Star é geralmente reconhecido como a principal introdução ao conceito GUI, porém, os próprios pesquisadores da Xerox citam 14 sistemas de computadores que influenciaram seus trabalhos.

Douglas Englebark do Xerox Parc, inventou o "mouse". Englebark tinha uma visão abrangente de seu novo dispositivo. Já considerava o "mouse" não apenas como um dispositivo de entrada de dados não convencional, mas também como uma maneira de deixar o monitor interativo (desta forma o monitor se transforma também numa entrada de dados). Devido à falta de precisão do primitivo "mouse" analógico, sua utilização não era tão interativa como é atualmente. Nos primórdios do "mouse" ele servia apenas para indicar o objeto, e a seguir eram introduzidos comandos via teclado.

Um dos primeiros sistemas que oferecia intimidade entre o usuário e o computador foi o sistema RAND desenvolvido pelos laboratórios RAND em 1967. Inicialmente criado como um sistema de interface para análises financeiras, no qual não eram digitados dados. O RAND possuia dimensionador automático, capacidade de vincular objetos e reconhecimento de caracteres. Nele o usuário poderia

desenhar caixas simbólicas, interconectá-las de acordo com suas relações e nomeá-las (com caracteres escritos à mão).

O sistema também tinha um corretor geométrico automático, onde o usuário poderia desenhar uma caixa aproximadamente quadrilátera, e o computador se incumbia de transformá-la em uma caixa com forma quadrilátera regular perfeita. O sistema era o primeiro a ter capacidade de reajuste de tamanho, que inspirou as janelas com tamanhos ajustáveis do Macintosh. O sistema RAND é literalmente a origem das capacidades de controle de janela do Macintosh, de acordo com Alan Kay, um dos inventores do GUI do Xerox PARC.

Este grau de intimidade contribuiu amplamente com o conceito de amigabilidade e interação da interface com o usuário. Os usuários se viam muitas vezes frustados com seus primeiros encontros com os computadores porque a apresentação destes se dava na forma de manipulação de palavras somente. Muitas vezes apenas eram utilizados processadores de texto, enquanto que a manipulação de idéias seria algo bem mais entusiástico, ao invés de manipular apenas palavras.

Por volta de 1983, todos os grandes vendedores de "workstations", tinham um sistema de janelas próprio. Por exemplo, Sun Microsystems tinha seu Sun View; Apollo seu DM, etc. Porém, as "workstations" não eram voltadas ao usuário médio. Seus consumidores eram grandes empresas e universidades. Em 1984, a Apple lança o Macintosh, um verdadeiro sistema de janelas, robusto e destinado ao consumidor médio. O GUI do Macintosh mudou completamente as expectativas do usuário para as interfaces com o computador, para sempre.

Em 1985 a Microsoft tinha o Windows para os PCs e compatíveis; Digital Research tinha o GEM, que era usado no Atari ST e alguns sistemas compatíveis ao PC; o Amiga da Commodore tinha a interface Intuition; e a Apple tinha o Lisa.

Em 1984, proveniente de um projeto do MIT (Massachussets Institute of Tecnology) denominado de Athena, surgiu o sistema X Window. O projeto Athena pesquisava a utilização de "workstations" gráficas e ligadas em rede, como ferramentas de aprendizado para estudantes em várias disciplinas. Os pesquisadores do MIT tentavam desenvolver um sistema de janelas que possibilitasse aos estudantes executar processadores de texto e planilhas, ao mesmo tempo na máquina local e simultaneamente consultassem ou importassem

documentos, figuras ou arquivos de outras máquinas ligadas à rede. O sistema X Window progrediu durante quatro anos, da versão 4 até a versão utilizada atualmente.

Na área do PC, interfaces com janelas baseadas em caracteres (modo texto) e "shell" para o sistema operacional começaram aparecer. Baseada nas janelas de caracteres, a IBM trouxe o Top View como um sistema de multitarefa. Porque era da IBM, muitas aplicações foram escritas para ele. Porém devido às limitações da UCP 808X, do gerenciamento do DOS (somente 640K), este sistema desapareceu.

Um outro sistema multitarefa com janelas baseadas em caracteres foi lançada pouco tempo depois do Top View, o DESQview, da Quarterdeck, embora tenha sido mais considerado como um gerenciador de memória. O DESQview continua popular até os dias de hoje.

O primeiro sistema de janelas gráficas de bits mapeados para PC foi o GEM da Digital Research. O GEM foi utilizado como interface gráfica do programa de editoração eletrônica Ventura da Xerox, e também como plataforma (GUI) dos microcomputadores ATARI.

Em 1985 a Microsoft lançou o Windows, em sua versão 1.01, que era bem limitado, por utilizar a UCP 8088. Mais tarde em 1987 foi lançado o Windows 2.0 que continha melhorias significativas, embora ainda muito limitado. Logo em seguida foi lançado o Windows/386, que era bastante melhorado, devido também à utilização de algumas características do novo microprocessador. Paralelamente foi lançado o Windows 2.1, que por utilizar a UCP 80286, foi denominado de Windows/286.[13]

Em 1990 foi lançado o Windows 3.0, que seria realmente o primeiro ambiente em janelas, viável para a linha PC. Seu sucesso foi imediato. Com apenas seis semanas após o lançamento, a Microsoft já tinha vendido 500.000 cópias, batendo todos os recordes de venda de "software" (num mesmo período).

Em 1992 a Microsoft lançou o Windows 3.1, consagrando definitivamente um GUI para a linha PC. Atualmente a Microsoft lançou o Windows NT (32 bits), um GUI voltado para "workstations".

Em 1988, a NeXT Inc. lançou o computador NeXT com a GUI NeXTstep. Este sistema foi muito interessante sob o ponto de vista da tecnologia de sua interface, porém era muito caro, não apresentava muitas aplicações e não mantinha

compatibilidade com os padrões industriais, o que fatalmente fez com que este modelo não prosperasse no mercado de massa.

Concomitantemente aos lançamentos das versões do Windows, a IBM lançou o OS/2, o seu sistema operacional para a sua linha de computadores PS/2. O GUI da IBM continua ainda hoje em plena concorrência com o da Microsoft, porém a Microsoft vem mantendo sua liderança no mercado.

Muitas outras GUI, especializadas ou não, foram lançadas também no decorrer destes anos. Atualmente a tendência industrial é adotar um sistema de janelas gráficas completamente controladas, que utilizam monitor formatado com mapas de bits, apontador ("mouse"ou "trackball") que pode movimentar pela tela, com precisão de pixel (menor elemento de tela - "picture element"), sistema WYSIWYG (do inglês "what you see is what you get", o que você vê é o que você tem), que possibilita ao usuário ver na tela exatamente o que será impresso na impressora ou transferido para outro dispositivo e multitarefa.

A figura 4.2 apresenta um diagrama mostrando a evolução dos GUI e dos sistemas operacionais. Por ser nosso trabalho realizado sobre plataforma Windows, nos preocupamos em detalhar melhor esta plataforma no diagrama:

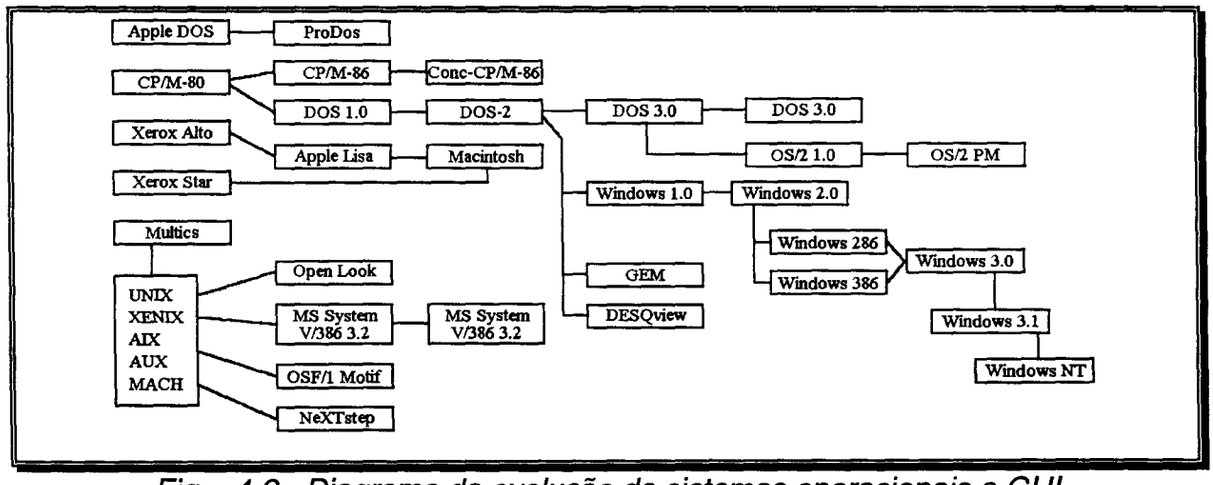


Fig. - 4.2 - Diagrama da evolução de sistemas operacionais e GUI

4.3 - DEFINIÇÃO DE GUI

O primeiro dispositivo que provia uma interface interativa homem - máquina, largamente utilizado nos primórdios da computação (décadas de 50 e 60), foram as "teletypewriters" (TTYs). Estes eram terminais baseados em caracteres, onde cada caracter digitado era impresso em papel. Com o aprimoramento tecnológico, tornaram-se disponíveis os terminais de vídeo (baseados em caracter). Pela incontestável superioridade de interação sobre os TTYs, os terminais de vídeo rapidamente tornaram-se uma norma na computação. [14]

Como interface interativa com o usuário, era então adotada a interface de linhas de comando, CLI do inglês "command-line interface". Este sistema vem sendo utilizado desde os TTYs até os dias de hoje.

Os sistemas operacionais com interface CLI, ou seja, operação através de linhas de comandos como o DOS, ou o UNIX, entre outros, têm sido por um bom tempo criticados, pela complexidade de suas interfaces com o usuário. Estas interfaces foram desenvolvidas entre o final dos anos 60 e o começo dos anos 70, devido à necessidade do novo estilo de computadores que estava surgindo. Desenvolvido para ser utilizado em minicomputadores, o UNIX foi criado como um conjunto de aplicações compartilháveis, que dispunham uma interface orientada a linha comum ao usuário. Esta interface foi revolucionária em seu tempo, por sua simplicidade e poder. O UNIX introduziu o conceito de "shell", um interpretador de comandos, que lê linhas de comandos provenientes do teclado, e então as executa como um processo separado para cada comando. O DOS, foi implementado mediante este conceito, no final dos anos 70.[1]

Os pesquisadores do Xerox PARC, contudo, estavam trabalhando com uma idéia diferente. Eles estavam pesquisando uma interface que substituía as linhas de comando e as telas de caracteres, por telas gráficas com grandes mapas de bits, através da qual os comandos eram exibidos graficamente (na forma icônica). Assim a interface com o computador se dava de forma "visual".[1, 2]

Para isso, os pesquisadores do PARC precisaram revolucionar o conceito de monitor de vídeo. Antigamente, o monitor de vídeo, era utilizado somente para reproduzir o texto que o usuário digitava no teclado. Com a nova concepção o monitor, através de um apontador (no caso o "mouse", que tinha sido criado na Xerox PARC), transformava-se numa verdadeira fonte de entrada de dados.[2]

A tela do monitor de vídeo, passa agora a exibir objetos gráficos na forma de ícones e janelas com dispositivos que são verdadeiros comandos "virtuais", como botões, barras de rolagem, menus, etc. Esses são manipulados virtualmente pelo usuário, através do apontador e também do teclado.

Com o apontador, pode se mover um cursor, geralmente com a forma de seta por toda a tela. O apontador ("mouse") possui um, dois ou até mesmo três botões. Pressionando os botões, ou combinando o pressionamento dos botões com o movimento do apontador, pode-se acionar os dispositivos "virtuais" posicionados na tela.[1]

Uma janela é um retângulo na tela através da qual é executado um programa. Em uma tela podem ser exibidas várias janelas. Estas podem ser movimentadas pela tela, e podem ser alteradas em tamanho ou forma pelo usuário através do apontador.

Os ícones são representações gráficas de objetos, onde cada objeto está associado a um arquivo, um programa, um grupo de arquivos, um grupo de programas, etc, que podem ser acionados mediante o apontador.

Os gráficos fazem uma utilização melhor da tela, transmitindo informações de maneira visual mais rica. E a combinação da exibição rica, com a manipulação virtual dos gráficos, fizeram com que as GUIs tornassem um novo degrau na evolução da interface homem-máquina. Deste modo com evolução de CLIs para GUIs, trabalhamos com um computador menos abstrato e mais "real".[1, 2]

Durante décadas, pesquisas tem sido realizadas com o intuito de desenvolver arquiteturas que aproveitem melhor o "software". Nestas pesquisas apareceram questões em relação a prioridades; as interfaces devem ser projetadas a fim de que os usuários experientes tirem, ao máximo, o proveito das máquinas, ou a fim de otimizar a curva de aprendizado dos iniciantes ?.[14]

As GUIs demostraram ser uma resposta efetiva para as questões de prioridade, no interfaceamento entre homem - máquina [14]. Com esta nova tecnologia, a utilização da máquina satisfaz todos os níveis de usuários: [1]

- O usuário iniciante, através da simplicidade de utilização, interação e conceitos intuitivos, imediatamente torna-se hábil para utilizar alguns recursos da máquina. O treinamento torna-se então rápido e eficaz, não necessitando de leitura de manuais extensos, e o aprendiz rapidamente está hábil a ser "mestre".

- O usuário casual, que executa várias diferentes aplicações diariamente, encontra a consistência do padrão entre as diferentes aplicações.
- E o usuário experiente, ou de "poder" ("power user"), também é beneficiado por executar suas tarefas com maior interação. Ele se beneficia do maior número de recursos providos com o novo sistema (ex: multitarefa).

4.4 - ELEMENTOS DE UMA GUI

Uma GUI deve conter certas características e capacidades para ser qualificada como consistente e utilizável.

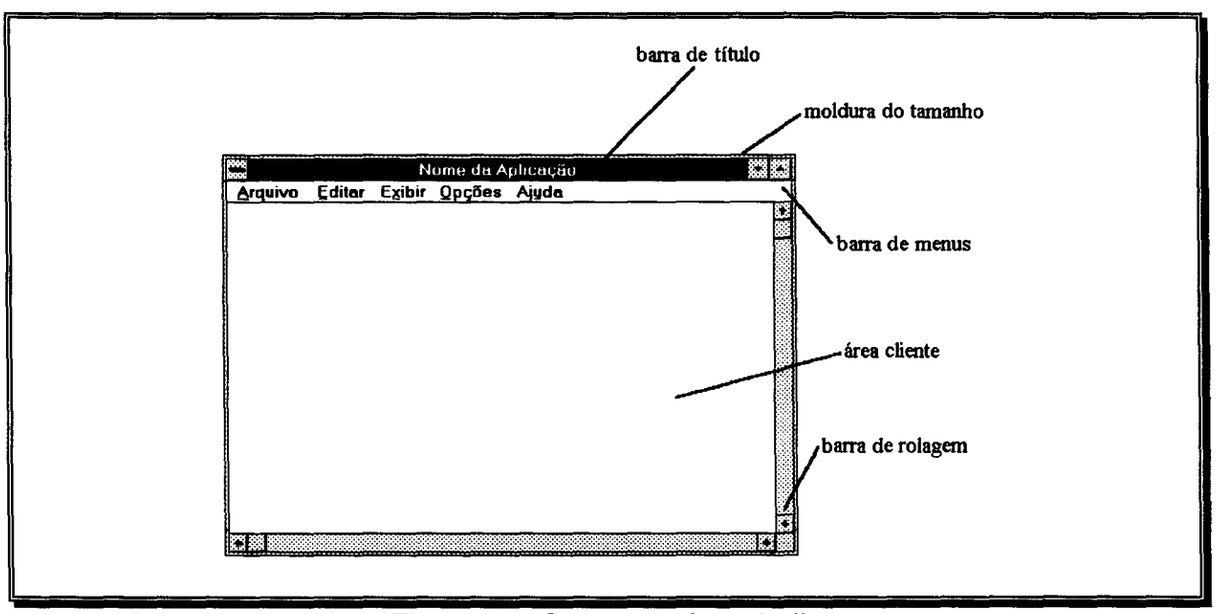


Fig. - 4.3 - O espaço de trabalho

A figura 4.3 apresenta uma organização primária de uma GUI, algumas vezes denominada como "o espaço de trabalho do usuário". O conceito de "desk", espaço de trabalho, organização e ícones foram estudados e compreendidos pela Xerox. Os pesquisadores da Star da Xerox elaboraram uma lista com os 5 mais importantes princípios de uma GUI.

1. A ilusão de manipular objetos. Um dos principais objetivos de uma GUI é criar a

IFSC — SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

ilusão de objetos que possam ser manipulados (movidos, alterados, descartados, etc.). Os objetos e ícones devem ser claramente exibidos para a seleção e manipulação.

2. Ordem visual e focalização do usuário. Isto envolve o balanço do contraste e intensidade entre itens e objetos. O contexto da janela deve ter maior intensidade em relação à tela. Isto inclui a apresentação clara dos estados de um ícone.
3. Estrutura revelada. O usuário necessita da capacidade de observar a diferença entre sua intenção e o efeito real (Impressão). Telas especiais podem ser usadas para mostrar ao usuário as estruturas subordinadas.
4. Vocabulário gráfico consistente e apropriado. Deve haver consistência entre ação e representação dos botões e objetos gráficos. O vocabulário gráfico é um do mais importantes elementos de uma interface amigável com o usuário.
5. Explorar as possibilidades do meio. Isto demanda atenção do projetista (de ícones e controles) a respeito dos recursos e limitações do sistema de exibição gráfica. Usuários e programadores devem estar sempre atentos a estes conceitos quando projetarem ou avaliarem uma GUI.

Atualmente existem uma grande variedade de GUI. Porém podemos observar que todas elas são constituídas de semelhanças básicas, que podemos citar, como:

- Um sistema apontador, tipicamente um "mouse".
- Uma tela que utilize gráfico mapeado a bit, com uma representação WYSIWYG ("What You See Is What Your Get" - O que você vê é o que você tem) da saída impressa.
- Janelas, que exibem graficamente o que o computador está fazendo.
- Menus, que podem aparecer ou desaparecer sobre controle do apontador.
- Ícones, que representam arquivos, diretórios, aplicativos e utilitários.
- Caixas de diálogos, botões, barras de rolagens, quadros de verificação, botões de opção e outros vários dispositivos gráficos, através dos quais o usuário instrui ao computador o que deve ser feito. A figura 4.4 apresenta elementos encontrados numa caixa de diálogos típica. Isto ilustra muito os conceitos necessários para uma GUI.

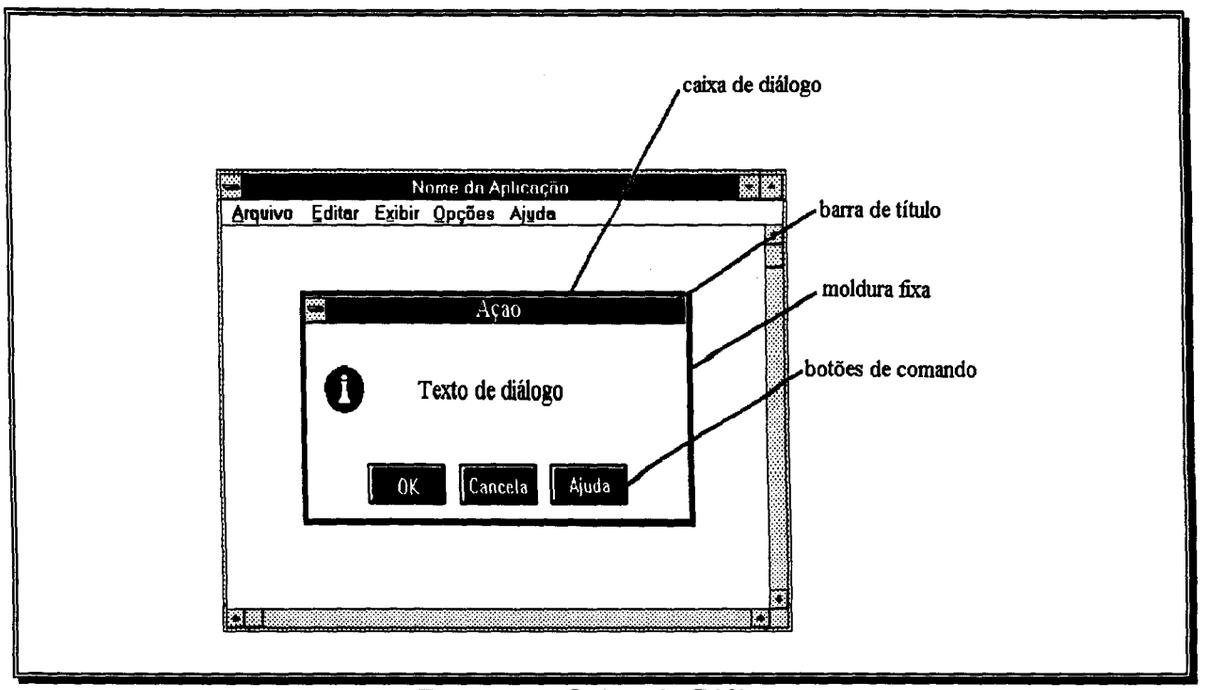


Fig. - 4.4 - Caixa de Diálogo

- Paradigma do Objeto-ação. O usuário primeiramente indica o objeto e depois executa o comando.

A figura 4.5 apresenta um exemplo típico da disposição das janelas e ícones numa tela de um sistema GUI. Podemos observar alguns elementos, como janelas, caixa de diálogos, dispositivos de controle e ícones.

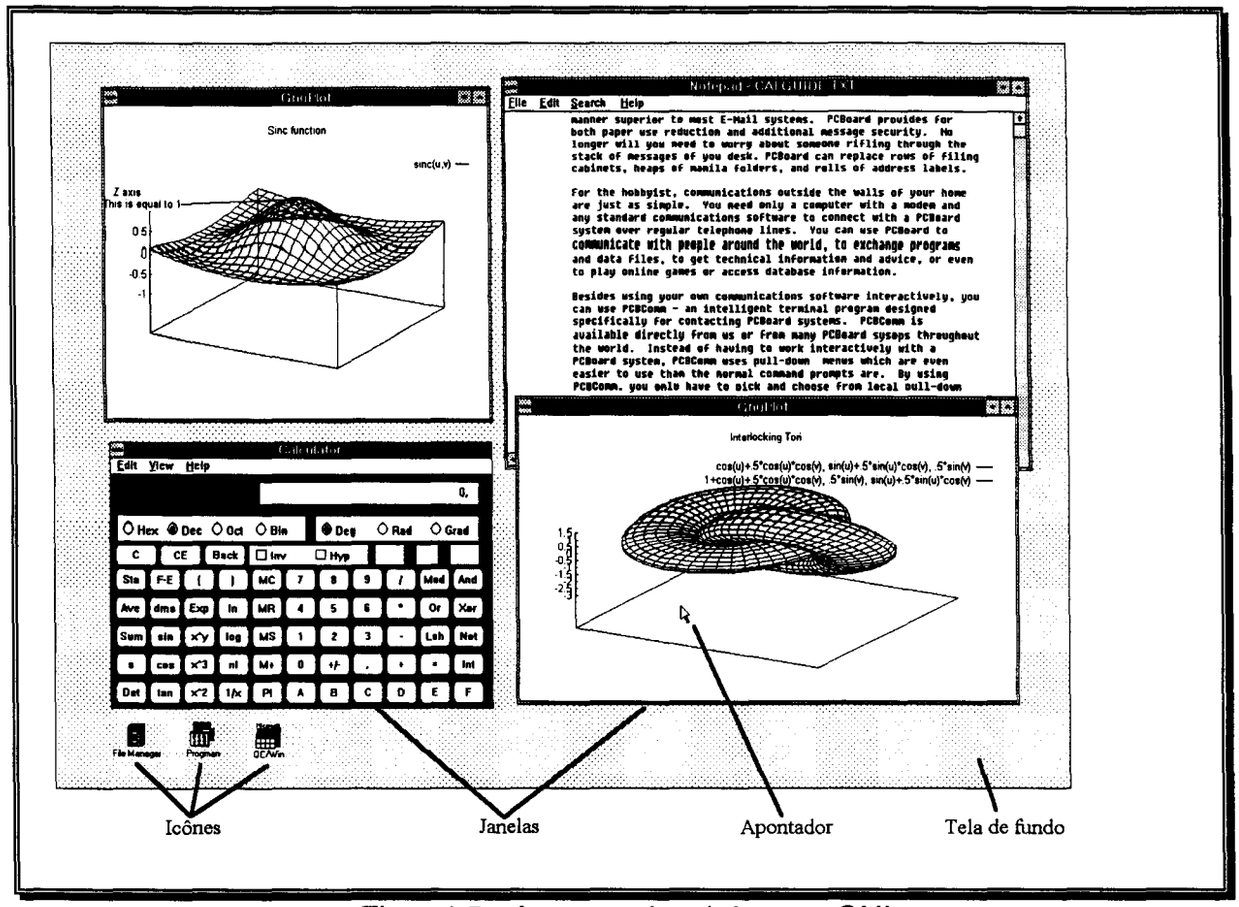


Fig. - 4.5 - Aspecto visual de uma GUI

4.5 - MODELO DE CAMADAS

Atualmente existe uma grande variedade de GUI. Muitas delas diferem em várias características. Existem GUI que não utilizam ícones, outras em que a utilização de ícones é opcional e ou disponível ocasionalmente. Algumas diferem sobre a utilização do apontador. Um outro elemento que diferencia algumas GUI, é o nível de integração entre a GUI e o sistema operacional. Algumas GUI são fortemente agregadas ao sistema operacional. Nestes casos, quando o sistema operacional é carregado, a GUI aparece automaticamente. Outro caso é quando a GUI é visivelmente colocada sobre um sistema operacional, como é o caso do Windows, e das GUIs para o UNIX.

A definição do que é ou não é parte de uma GUI variará de um desenvolvedor para outro. As partes de uma GUI que interagem entre si são várias. Apresentaremos um modelo genérico das camadas de uma GUI que tem como

objetivo mostrar seus componentes (figura 4.6). [1]

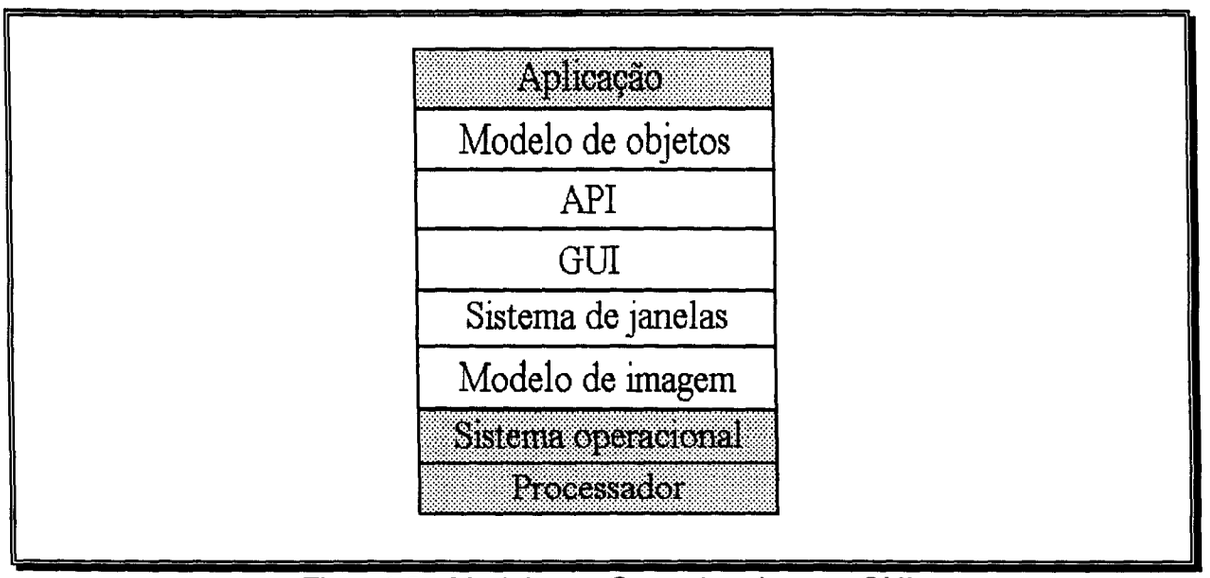


Fig. - 4.6 - Modelo em Camadas de uma GUI

O processador usado com uma determinada GUI, pode variar assim como o sistema operacional. Foram adicionados no modelo de camadas estes dois elementos com o intuito ilustrativo. O modelo inicia no topo, com a aplicação. Este é considerado o ponto onde o usuário contacta o sistema. Contudo, o usuário contacta o próprio GUI muitas vezes, através de ícones e menus. Este modelo não tem a intenção de propiciar um exato mapa da funcionalidade. Ele é utilizado somente para, genericamente, apresentar algumas camadas que compõem o sistemas GUI.

Existem cinco camadas básicas no sistema, além da aplicação e do sistema operacional e do processador. Em alguns casos o sistema operacional é parte integrante da GUI. Em outros a GUI é colocada sobre o sistema operacional.

Cada um dos parágrafos a seguir irá descrever cada uma das camadas do modelo.

Modelo de objetos: O caminho pelo qual as aplicações reagem entre si freqüentemente, envolve o uso de um modelo de objetos. O modelo "Object Link & Embedding (OLE)" é um exemplo desta camada. Existem algumas GUI que não possuem um modelo de objetos.

API: A Abreviação é oriunda do inglês "Application Program Interface". Esta camada é um conjunto de funções que são utilizadas pelos programas para se comunicarem com a GUI. O programador precisa especificar que função (ex:

janelas, menus, barras de rolagem, ícones e etc) são utilizadas e quando.

GUI: Como mencionado outrora, o que é e o que não é parte de uma GUI varia de desenvolvedor para desenvolvedor. A camada GUI é onde as ações e elementos da tela residem.

Sistema de janelas: Esta camada é tão difícil de definir quanto a camada GUI. Por exemplo, o X-Window é apenas um sistema de janelas, ao passo que o Windows é um sistema de janelas e uma GUI construídos juntamente.

Modelo de imagem: Enquanto o sistema X-Window não tem um modelo de imagem, algumas GUI suportam mais de um modelo de imagem. É o caso do NeWS da Sun Microsystem, que suporta um modelo similar ao modelo de imagem Display PostScript (DPS); porém, diferente do DPS, ele pode também trabalhar com um modelamento complexo como o PHIGS ou GKS para controle de programas de CAD ("Computer Aided Design").

É bastante difícil obter um modelo genérico de representação para todas as GUIs. Em modo geral as GUIs se diferem muito, algumas com características a mais ou a menos, mas este modelo tenta ser o mais genérico possível. Dentre uma destas diferenças podemos citar a existência de camadas que possuem suas próprias API, como é o caso do sistema de janelas X-Windows que possui uma API para o sistema de janelas, ou então o NeXT, onde existe uma API usada para construir a GUI.

A figura 4.6 apresenta todos as camadas que compõem uma GUI total, desde o processador até o usuário (aplicativo). O modelo de camadas representado por ela tem caráter comparativo. Na figura 4.7 são apresentados algumas GUI diferentes, possibilitando a comparação de uma para outra GUI. Não obstante, nem todas as GUIs possuem todas as camadas descritas no modelo genérico. Existem algumas que possuem camadas externas, como o caso do NeXTstep, que não é fundamentado numa plataforma orientada ao objeto. Neste caso, o modelo de objeto, quando usado nesse sistema, é externo.

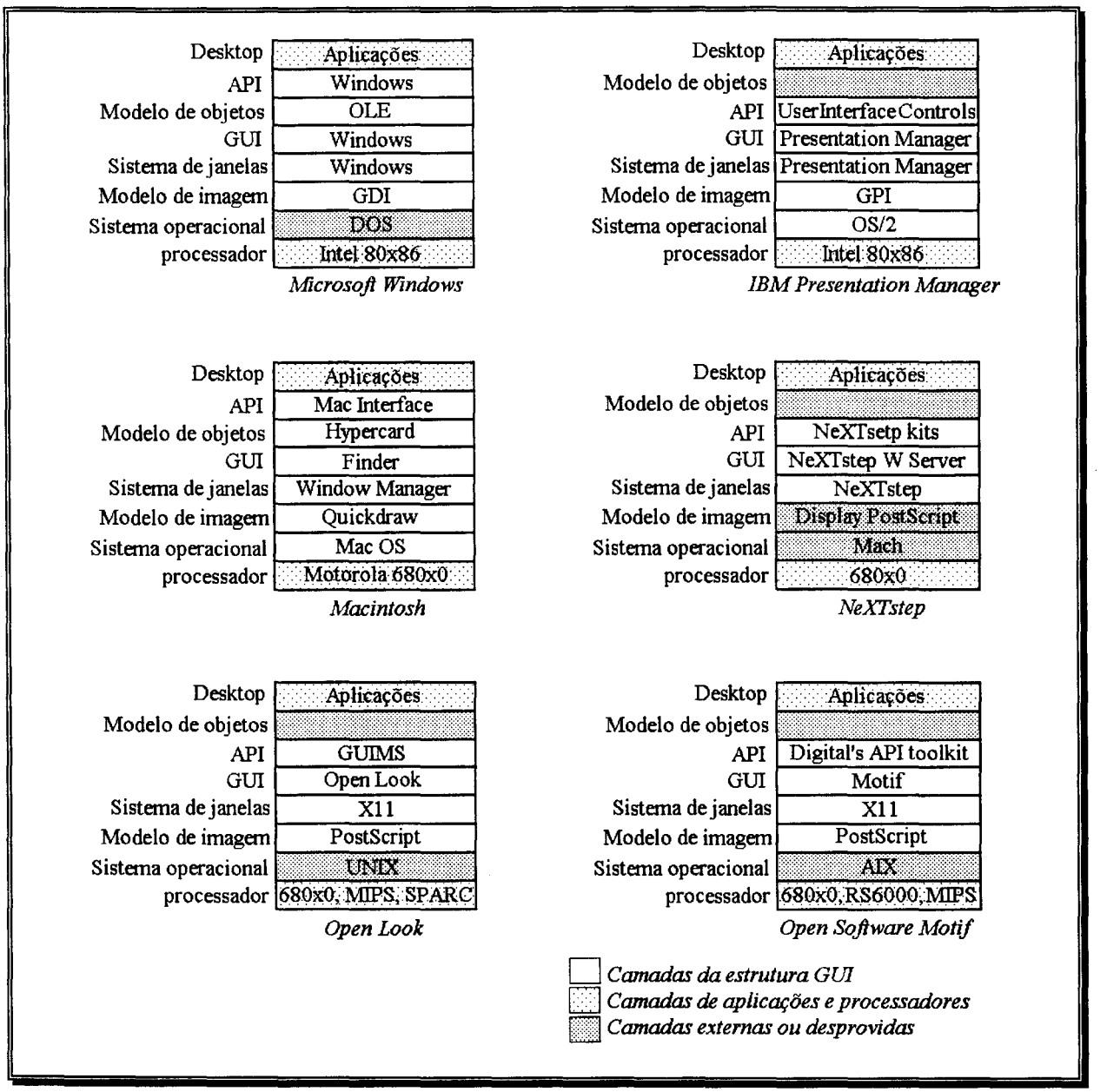


Fig. - 4.7 - Diagrama comparativo de camadas de GUI

4.6 - FUNDAMENTOS DE PROGRAMAÇÃO EM GUI

Com o conceito GUI, várias novas tecnologias de programação foram adotadas ou criadas para dar consistência a nova interface homem-máquina.

Diferentemente da filosofia CLI, a plataforma GUI é voltada para o usuário, e

não para o programador. Com isto, programar (utilizando um compilador tradicional) para um ambiente GUI, não é uma tarefa simples como programa para um ambiente CLI, cuja filosofia facilita o programador. Com isso, nos primórdios do ambiente GUI, a programação de aplicativos era uma tarefa extremamente difícil e trabalhosa, pois embora o sistema gerenciava as janelas e facilitava sua criação, uma boa parte do trabalho era destinado ao programador.

4.6.1 - Comparação de um Programa Mínimo entre CLI e GUI

Um bom exemplo para ilustrar esta complexidade de programação, é o caso do programa mínimo "Hello World". Este programa é um "bechmark" de linhas de programação. Deste modo observamos quantas linhas os compiladores necessitam para executar uma determinada tarefa. No nosso caso a tarefa é simplesmente exibir a mensagem "Hello World" no monitor de vídeo do computador.

Adotamos a linguagem C, e vamos comparar o programa "Hello World" destinado ao CLI com o voltado ao GUI.

Para o ambiente CLI simplesmente três linhas de código são necessárias para a criação do nosso programa, e podemos observar sua listagem (esquerda) e o resultado de sua execução (direita) na figura 4.8.

```
#include <stdlib.h>
main()
{printf(\n;'Hello World';\n);}
```

```
a:> hello

Hello World

a:>_
```

Fig. - 4.8 - Exemplo do Programa "Hello World" em CLI

No caso do mesmo programa em linguagem C, porém desenvolvido para um ambiente GUI, embora, a maior parte desta tarefa seja realizada pelo sistema, o programador deve gerenciar e criar uma janela, e então gerar a mensagem. O programa "Hello World" em C para Windows (plataforma GUI utilizada em nosso trabalho), possui cerca de 90 linhas de código, que são apresentadas no apêndice B. Com isto é demonstrada a árdua tarefa da programação em ambientes GUI.

A figura 4.9 apresenta o resultado da execução do programa "Hello World" em ambiente GUI. Comparando com o exemplo em CLI, observamos que em GUI, foi necessária a criação de uma janela gráfica, para exibir a mensagem, fator responsável pela "complexidade" do programa.

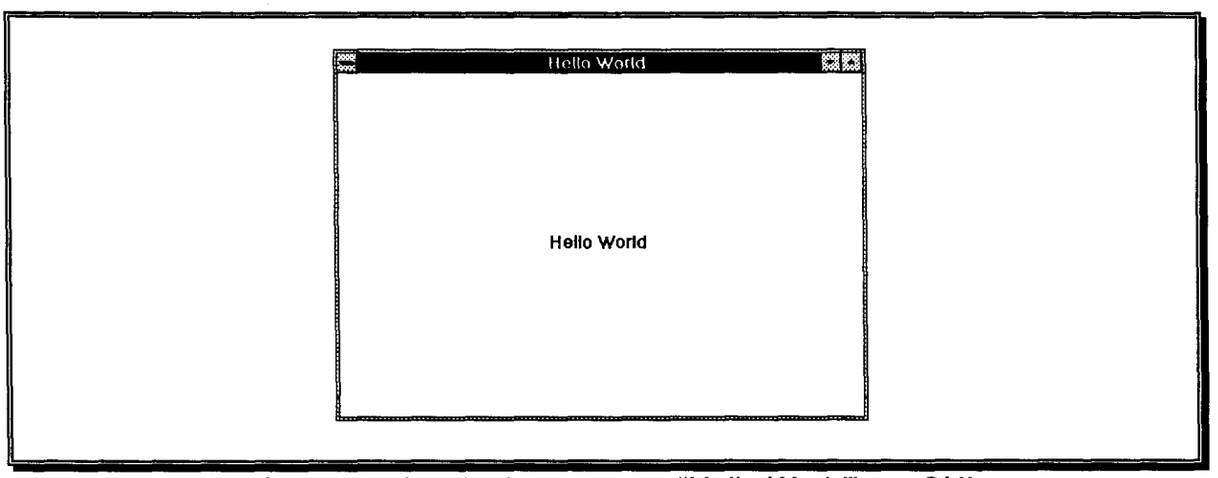


Fig. - 4.9 - Janela do programa "Hello World" em GUI

4.6.2 - FERRAMENTA DE AUXÍLIO À PROGRAMAÇÃO

Como o programador deve criar a janela, e como esta tarefa é bastante genérica, devido ao código de criação de janelas é bem parecido entre as aplicações. Foi então adotada a utilização de ferramenta CASE [15] (Computer Aided Software Engineering), para a elaboração das janelas. A ferramenta CASE, consiste em um compilador de parâmetros para uma determinada linguagem de programação.

A figura 4.10 apresenta o esquema de funcionamento de uma CASE. Neste

exemplo, o programador desenha uma janela, e coloca nesta, os comandos, botões, e menus. A ferramenta CASE compila então estes parâmetros para uma linguagem de programação qualquer (no exemplo linguagem C). A utilização de ferramentas CASE auxilia e diminui o trabalho do programador. Graças a sua utilização torna-se mais viável o desenvolvimento de aplicações com interfaces sofisticadas.

Praticamente todos os compiladores atuais direcionados ao ambiente GUI possuem um CASE.

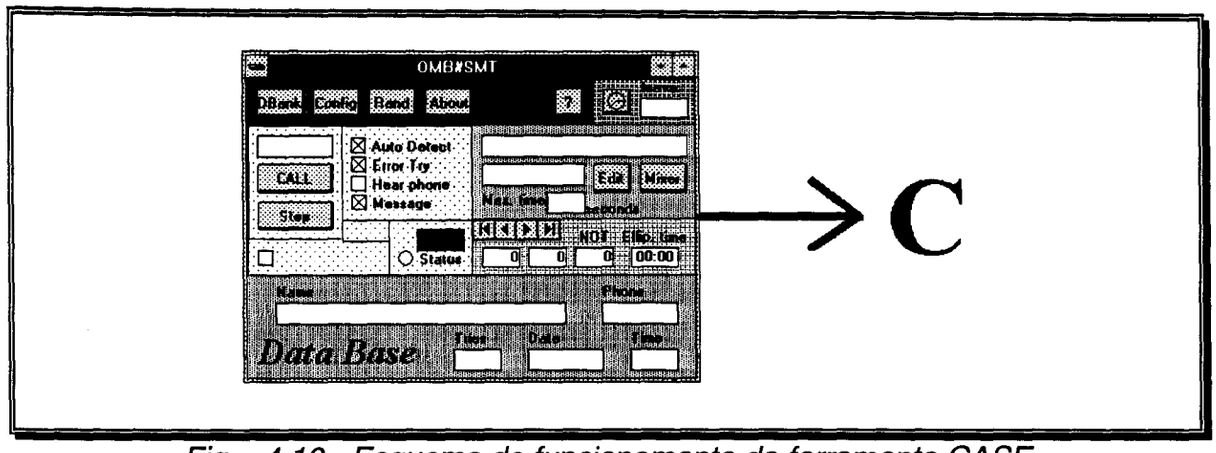


Fig. - 4.10 - Esquema do funcionamento da ferramenta CASE

4.6.3 - PROGRAMAÇÃO ORIENTADA A EVENTOS

Uma outra característica da programação em ambientes GUI, é a programação orientada a eventos [2, 13]. Num programa típico desenvolvido para as interfaces CLI, a programação é feita de maneira linear, ou seja, o programador define a ordem do que vai acontecer, assim como de quais dados deverão entrar. A seqüência do programa é bem definida: começo, meio e fim.

Com a programação orientada a eventos, não existe mais esta linearidade de programação. Diferentemente da programação linear ou convencional, o programa é controlado por mensagens que indicam ao programa que evento ocorreu. Estas mensagens são geradas pelo GUI, indicando ao programa qual foi a ação tomada pelo usuário. Neste, a ordem de execução dos eventos, é definida pelo usuário e

não pelo programador.

Para ilustrar melhor, tomaremos como exemplo um simples programa baseado na fórmula de Baskara, que soluciona equações polinomiais de segundo grau.

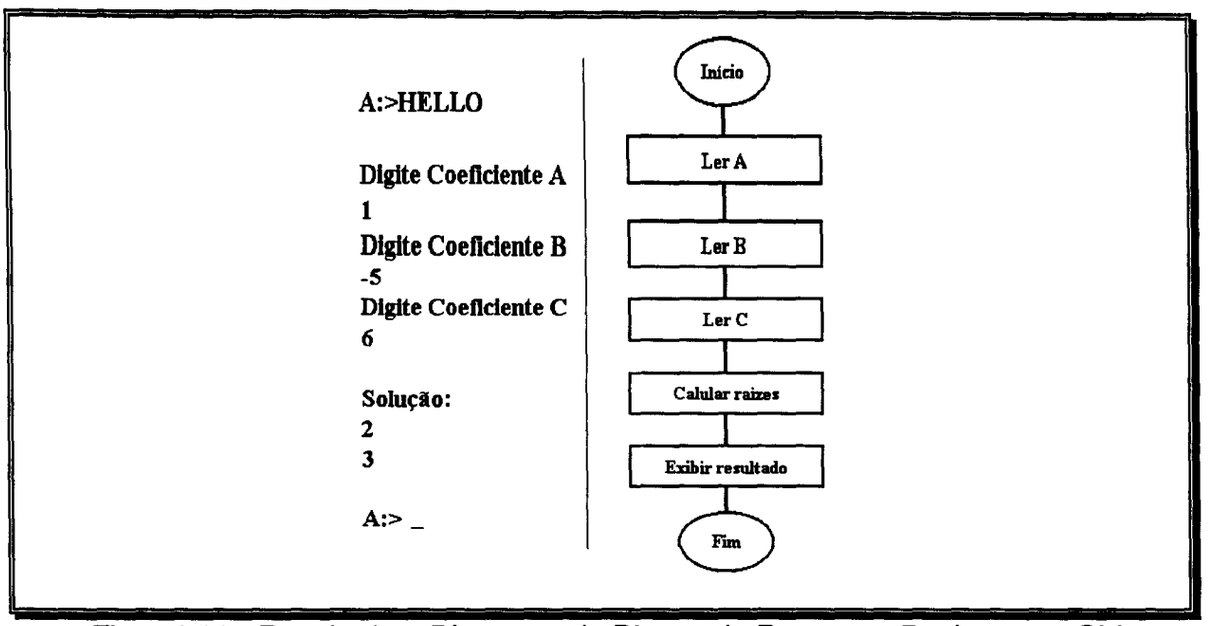


Fig. - 4.11 - Resultado e Diagrama de Blocos do Programa Baskara em CLI

Com a implementação do programa em ambiente CLI, teremos como podemos observar na figura 4.11, uma seqüência linear, onde o programa é inicializado, é definida uma ordem de entrada de dados, no caso os coeficientes da equação, e finalmente são calculadas as raízes e exibido o resultado, e então é finalizado o programa.

A implementação do programa em ambiente GUI, é bem diferente. Não existe seqüência linear no programa. O usuário decide qual é a ordem dos dados, e caso deseje, pode efetuar o cálculo sem entrarem os dados ou parte dos dados, ou ainda finalizar o programa quando desejar. Na porção inferior da figura 4.12 observamos um diagrama dos eventos do programa abaixo, mostrando sua não linearidade, e a interface com o usuário do programa na porção superior.

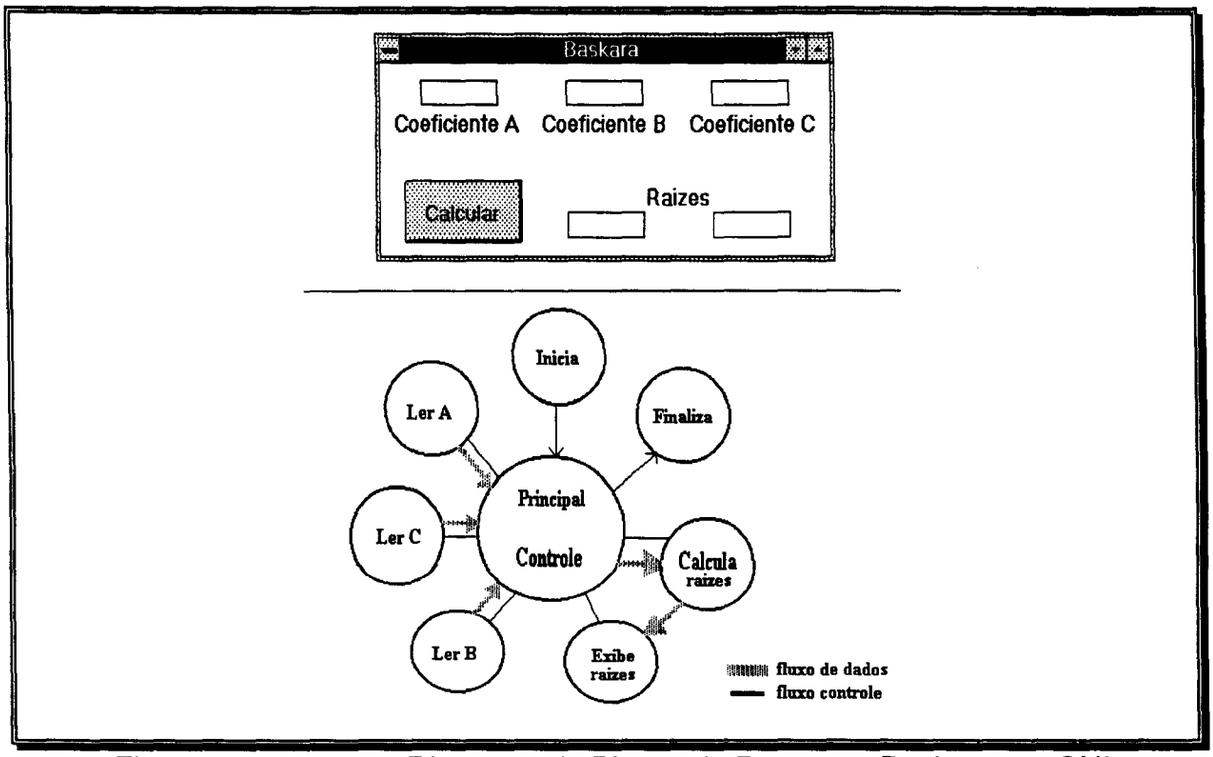


Fig. - 4.12 - Janela e Diagrama de Blocos do Programa Baskara em GUI

4.6.4 - PROGRAMAÇÃO ORIENTADA PARA OBJETOS

Com o surgimento da GUI, foi adotada a tecnologia de "software", da programação orientada para objetos. Diferentemente da programação convencional, a programação orientada a objeto, ou OOP do inglês "object-oriented programing", faz uma representação do mundo computacional através de objetos. Um objeto (sentido mais amplo) é qualquer coisa que tenha limites claramente definidos. Canetas, gatos, moléculas e galáxias, por exemplo, são objetos; porém, oceanos, neblina e céu não o são. [16]

A origem da programação orientada a objetos, surgiu com a linguagem Simula, criada no início dos anos 70. Porém seu real "nascimento" ocorreu somente no início da década de 80, com o lançamento do Smalltalk da Xerox PARC. Com o Smalltalk, o conceito de OOP, obteve êxito e foi adotado como a plataforma de programação em GUI. Com a proliferação do conceito de GUI, concomitantemente

proliferou o conceito de OOP.

A seguir, vamos descrever sucintamente a concepção da OOP segundo os termos do Smalltalk (figura 4.13). [17]

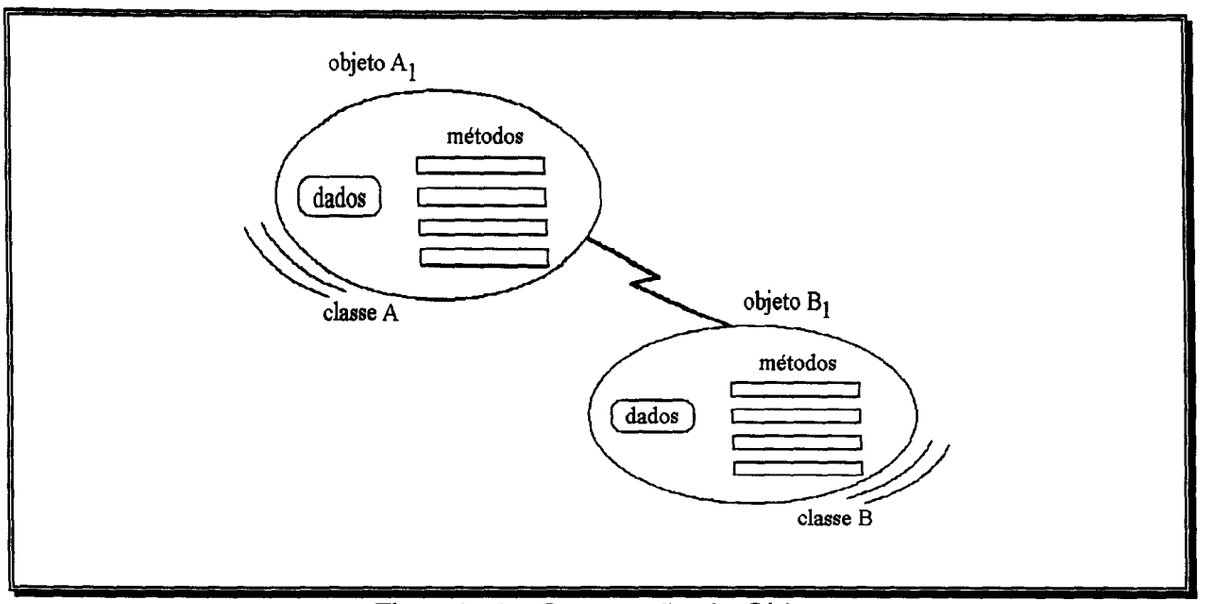


Fig. - 4.13 - Concepção de Objeto

Um objeto é um pacote de informações (dados) e as descrições de suas manipulações (procedimentos). Uma classe é uma descrição de um ou mais objetos similares. Uma mensagem é uma especificação para um manipulador do objeto. Um método é uma entidade (como um procedimento) que descreve uma seqüência de ações a serem realizadas com o recebimento de uma mensagem.

Objetos possuem a característica da encapsulação, isto é, os dados internos de um objeto são protegidos do mundo exterior. O único meio de obter acesso a esses dados é através do envio de uma mensagem para o objeto. Após receber uma mensagem, o objeto ativa o método correspondente e retorna o resultado para quem a enviou.

Outra característica dos objetos é a herança. Uma classe pode possuir diversas subclasses; uma subclasse herda todas ou parte das propriedades de sua classe superior. As propriedades herdadas não precisam ser redefinidas na subclasse; por outro lado, novas propriedades podem ser adicionadas à subclasse caso necessário.

O interesse da aplicação da OOP na engenharia de "software" é baseado na sua reabilitação e reciclagem.

Ainda uma outra característica da OOP é a sua estrutura hierárquica.

O conceito de OOP é essencial na programação de aplicativos voltados para plataforma GUI, já que esta é fundamentada sobre OOP.

4.6.5 - PROGRAMAÇÃO VISUAL

A idéia de programação visual, começou a surgir, com o famoso projeto Vivarium, no qual crianças utilizavam o Smalltalk para desenvolver um ambiente e então o povoavam com animais virtuais. Se crianças podem, gerar um "mundo" simplesmente movimentando objetos, não poderíamos utilizar a mesma abordagem para desenvolver aplicativos ?.

A idéia básica da programação visual, seria desenvolver uma ferramenta, na qual fosse possível gerar um aplicativo sem escrever nenhuma linha de código. Porém, o que ocorre é que na programação visual os elementos da interface com o usuário, suas características e controle são gerados visualmente (através do apontador) e a estes elementos é ligado o código que implementa a ação ("software") que provoca o elemento. Este código não é implementado visualmente, ficando como tarefa das linguagens convencionais (ou não) de programação.

Atualmente, parece predominar a tendência de ferramentas de programação visual, e muitos ambientes fundamentados em linguagens de programação convencionais estão emergindo.

O conceito de programação visual está baseado na inversão da ênfase da programação. Nas interfaces baseadas em caracteres (CLI), e nos compiladores tradicionais de linguagens procedimentais, a ênfase da programação residia no projeto das estruturas de dados e na tradução lógica de uma atividade para o código do programa. O projeto da interface com o usuário ficava normalmente em segundo plano. As ferramentas de programação visual invertem esta equação: o processo de programação começa com a interface com o usuário, e permanece centralizado nela. [18]

Com isso surge uma nova filosofia de codificação, onde o código do programa fica ligado diretamente aos objetos da interface com o usuário. Isto só é possível com um ambiente onde o compilador e o CASE estejam integrados.

Dentre as ferramentas de programação visual para ambiente Windows, a ferramenta que parece melhor satisfazer o conceito é o Visual Basic da Microsoft. Esta ferramenta possui atualmente um parque estabelecido de usuários, tornando um forte padrão para a programação em ambiente Windows. O Visual Basic oferece simplicidade na elaboração de protótipos e poderosas e abrangentes funções. Embora fundamentado na linguagem de programação Basic, o Visual Basic demonstra ser um novo enfoque para o velho Basic, passando de uma linguagem para iniciantes, para uma poderosa ferramenta de trabalho para os desenvolvedores de "software" em ambiente Windows. [18, 19]

4.6.6 - LIGAÇÕES DINÂMICAS [2]

O conceito de ligação dinâmica abordado neste item, está direcionado para a plataforma Windows.

Um programa executado sob o DOS (sistema operacional utilizado no padrão PC) realiza a comunicação com o sistema operacional mediante interrupções de "software" como por exemplo a 00x21 do DOS. Quando utilizamos a plataforma Windows, os programas se comunicam com o Windows através de um processo denominado de "ligação dinâmica".

Como os programas voltados para o DOS, os programas voltados para o ambiente Windows, têm a extensão .EXE. Entretanto, esse não é o mesmo formato usado no DOS. Os programas Windows usam um formato .EXE denominado de "Executável Novo", semelhante ao utilizado no OS/2 (GUI da IBM para o padrão PC). Sempre que um programa chama uma função do Windows, o compilador (da linguagem de programação utilizada) gera um código em Assembly para uma sub-rotina intersegmentada (ou seja, uma chamada do tipo "far"). Uma tabela no arquivo .EXE identifica a função sendo chamada, usando um nome de biblioteca de ligações dinâmicas e um nome ou número (denominado de número ordinal) da função nessa biblioteca. [2]

O próprio programa Windows consiste principalmente em três bibliotecas de

ligações dinâmicas, denominadas de KERNEL, USER e GDI. A biblioteca KERNEL tem como funções o gerenciamento da memória, carga e execução de programas e escalonamento das tarefas (multitarefa). A USER tem como funções a interface com o usuário e a criação das janelas, e a GDI o modelamento gráfico. Estas bibliotecas são encontradas no subdiretório SYSTEM (\windows\system) do diretório WINDOWS. Embora as bibliotecas de ligações dinâmicas possam possuir as extensões .EXE e .FON, o padrão de extensão no Windows 3.x é a extensão .DLL. [2]

O termo "ligação dinâmica" refere-se ao processo que o Windows usa para ligar a chamada a uma função em um módulo à função verdadeira no módulo da biblioteca. "Ligação estática" ocorre quando executamos o programa LINK para criar um arquivo executável (.EXE) a partir de vários módulos objetos (.OBJ) e arquivos de biblioteca (.LIB). Diferente da ligação estática, a ligação dinâmica ocorre durante a execução.

Um dos propósitos das bibliotecas de ligação dinâmica é oferecer funções e recursos que possam ser usados por muitos programas diferentes. Em um sistema operacional convencional, somente ele próprio contém rotinas que os outros programas podem chamar para realizar uma tarefa. No Windows, o processo de um módulo que chama uma função em outro módulo, é generalizado. Na verdade , ao escrever uma biblioteca de ligações dinâmicas, você está estendendo o Windows. Uma outra maneira de pensar, é que as bibliotecas de ligação dinâmicas, incluindo as que formam o Windows, são extensões do programa (sob a ótica do programador). O código, os dados e os recursos em um módulo de biblioteca de ligações dinâmicas são compartilhados entre todos os programas que usam o módulo.

Uma das vantagens das bibliotecas de ligações dinâmicas, é a padronização. Podemos desenvolver um aplicativo utilizando várias linguagens de programação. Deste modo podemos tirar proveito das características fundamentais das linguagens, como cálculo, programação visual, gerenciamento de banco de dados, controle de baixo nível, etc., modularizando o aplicativo e desenvolvendo cada módulo na linguagem mais apropriada.

IFSC — ... DE BIBLIOTECA E ... AÇÃO

4.6.7 - MULTITAREFA

A multitarefa é um mecanismo de "hardware" e "software", com um sofisticado gerenciamento de memória, que possibilita que dois ou mais programas compartilhem a dedicação de um microprocessador. Deste modo o usuário tem a ilusão de que seus programas estão sendo executados ao mesmo tempo. Nesse contexto o usuário pode abrir várias janelas e processá-las simultaneamente, e em sua ótica, ter vários programas sendo executados em paralelo ao mesmo tempo. [1, 13]

Embora a ilusão seja que os programas, de um sistema multitarefa, estejam sendo executados paralelamente, isto não ocorre, pois existe apenas um microprocessador. O que ocorre, é que a execução dos programas é alternada no tempo. Deste modo, o microprocessador executa um programa por um determinado tempo, e depois executa o outro, e assim por diante. Este chaveamento de tarefas é realizado em geral, rapidamente, gerando a impressão de que os programas estão sendo executados simultaneamente. A camada de "software" do sistema operacional responsável por este gerenciamento é em geral denominada de Kernel (UNIX e Windows), e esta é responsavél também pelo complexo gerenciamento de memória exigido pela multitarefa.

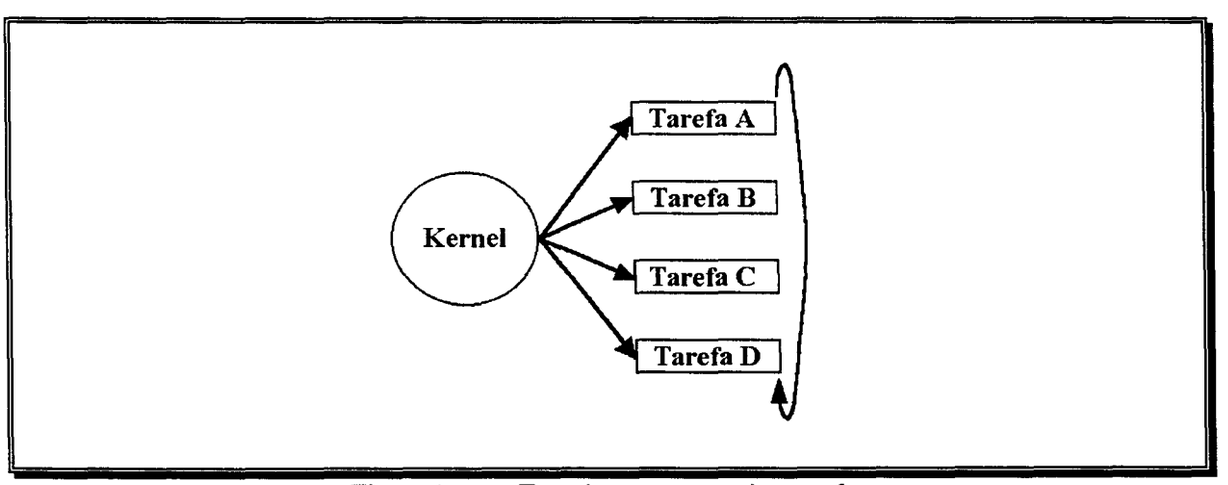


Fig. - 4.14 - Escalonamento de tarefas

A figura 4.14 ilustra o escalonamento de tarefas realizado pelo Kernel. Neste exemplo temos quatro programas sendo executados "simultaneamente" através da técnica da multitarefa. O Kernel, escalona então o tempo de execução da UCP

(Unidade central de processamento), e o distribui entre os programas. Desta forma, o programa A (tarefa A) é executado por um período de tempo determinado pelo Kernel. A seguir, sua execução é interrompida, e o próximo programa é executado. Este também é executado por um período determinado pelo Kernel, e então é, executado o próximo programa. Ao final da lista (no exemplo: tarefa D), a primeira tarefa da lista volta a ser executada no exato ponto onde fora interrompida. Esta, volta novamente a ser executada pelo mesmo período de tempo e então é interrompida novamente e o próximo elemento da lista é executado.

Observamos então, um laço de execução entre as tarefas. Este permanece até que todos os programas (tarefas) sejam finalizados.

Encontramos duas formas de controle distintas na técnica da multitarefa, são elas a "preemptive multitasking" e a "nonpreemptive multitasking".

Na "preemptive multitasking" (ex: UNIX), o escalonamento de tempo e o chaveamento das tarefas são realizados pelo sistema (Kernel), ao passo que na "nonpreemptive multitasking" (ex: Windows), a tarefa que "devolve" o controle para o sistema (Kernel), fica o controle do escalonamento a cargo do programador dos aplicativos. Deste modo, o programa define o período de sua execução, podendo também dominar o sistema, ou seja não devolver o controle para o Kernel, pelo tempo que "desejar".

CAPÍTULO V - DESCRIÇÃO DO SOFTWARE

5.1 - INTRODUÇÃO

O sistema desenvolvido neste trabalho, foi projetado a fim de proporcionar uma operação simples, dinâmica e interativa, possibilitando deste modo, que o usuário TSC realize medidas com boa qualidade, de maneira simples e com o mínimo de treinamento. Para que estas e outras condições fossem atendidas, desenvolvemos o sistema sob uma plataforma GUI (MS-Windows 3.1), e obtivemos, além dessas condições, também outras vantagens desta tecnologia de interfaceamento entre homem-máquina.

Este capítulo tem como objetivo apresentar a porção de "software" do sistema. Mostraremos a operação do sistema, enquanto comentaremos sua interface. Serão abordadas também, com maior atenção alguns procedimentos e funções mais específicos do sistema, como as comunicações com o "hardware" e o controle das medidas.

5.2 - FERRAMENTAS DE PROGRAMAÇÃO

Em nosso projeto foram utilizados três compiladores: MS Visual Basic 3.0, MS Visual C++ 2.0 e o MS Quick C for Windows 1.0.

Como base de implementação de nosso trabalho, utilizamos a ferramenta de programação visual (item 4.6.5) da Microsoft: Visual Basic 3.0. Embora quase a totalidade do sistema tenha sido implementado em Visual Basic, devido à falta de recursos para acessar o baixo nível ("hardware"), desenvolvemos uma biblioteca de ligação dinâmica (DLL- item 4.6.6), escrita em C, que possibilita o acesso ao baixo nível, exigido em determinadas partes do sistema.

O apêndice C contém a listagem do sistema. Esta listagem está dividida entre

o sistema principal, escrito para Visual Basic, e outras partes escritas em C, que são ligadas ao sistema.

5.3 - DESCRIÇÃO E OPERAÇÃO DO SISTEMA

Descreveremos a seguir a interface com o usuário desenvolvida em nosso projeto. Serão apresentadas as janelas e dispositivos GUI de controle, dos quais faremos um breve comentário sobre sua operação.

5.3.1 - JANELA PRINCIPAL

Quando executamos o sistema (clicando o ícone representativo do sistema), é então exibida sua janela principal, descrita na figura 5.1. Esta é a principal interface do sistema. Nela está contida a barra de menus (4), com os principais menus do sistema, e também são apresentados os resultados das medidas.

Vamos agora fazer uma descrição da operação do sistema, com o intuito de ilustrar sua simplicidade de utilização.

Caso deseje efetuar uma medida, o primeiro procedimento que o usuário deve efetuar é configurar o sistema de acordo com as necessidades de sua medida. O sistema é configurado, mediante a janela de configuração (que será descrita posteriormente). Para exibi-la, o usuário deve clicar o item *Config* (fig. 5.2) no interior do menu *Config.* (7) da barra de menus. Descreveremos posteriormente a janela de configuração, e sua operação.

Após definidos todos os parâmetros do sistema, o usuário seleciona o item *TSC* (fig. 5.2) do menu *Programa* (5), e então é realizado o controle da temperatura da estufa (conforme descrito pelo usuário) e também a aquisição dos dados provenientes do sistema.

Simultaneamente a realização do processo TSC (controle e aquisição), o sistema exibe interativamente, os dados que estão sendo lidos através da aquisição, e também o parâmetro que está sendo transposto para o controlador de

temperatura. Deste modo, o usuário pode monitorar sua medida e fazer análises prévias de sua qualidade, funcionalidade e interesse.

As caixas expressas pelo item 13 da figura 5.1, contém os dados referentes à temperatura da estufa. A cada aquisição e transposição, estes dados são atualizados. Na caixa *Tp T* é apresentado o parâmetro transferido para o controlador, ou seja, o valor da temperatura ideal, e na caixa *Tp P* é apresentado o valor da temperatura "lida" na estufa. Ambas as temperaturas são expressas em Celsius, e são traçadas ponto a ponto na sub-janela esquerda (11) (fig. 5.1). Deste modo, o usuário obtém visualmente a linearidade do controle que está sendo (ou foi) realizado.

A curva (1) da figura 5.2, contém o traçado da temperatura ideal (*Tp T*), e a curva (2), a temperatura proveniente da estufa. Podemos observar um ligeiro "atraso" a subida da temperatura (curva 2). Isto se deve à inércia térmica da estufa.

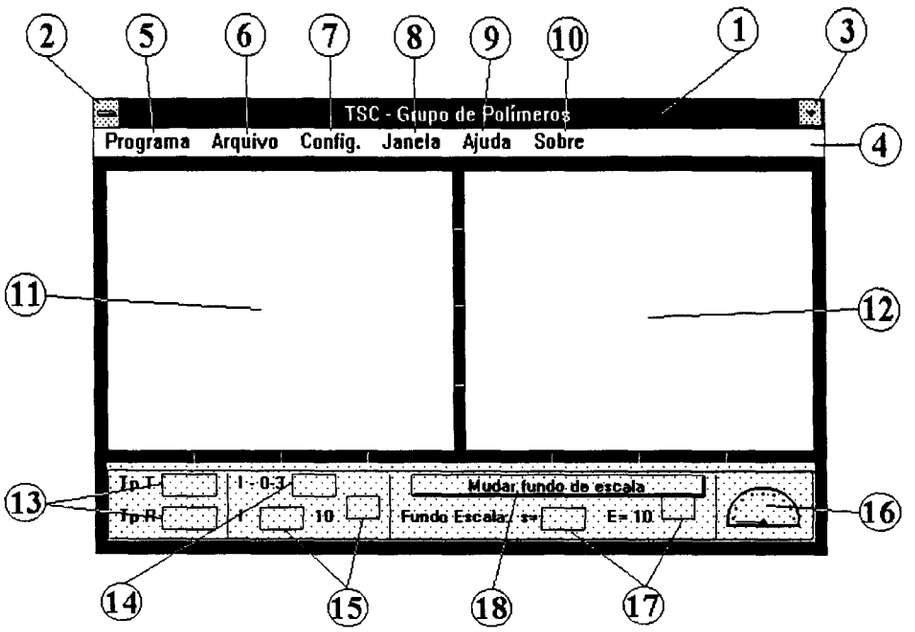
Enquanto é realizado o controle da temperatura da estufa, o sistema faz também a aquisição da corrente proveniente da amostra. Fazemos a aquisição da corrente através do sinal de referência proveniente do eletrômetro (item 3.5.2). O valor deste sinal é apresentado ao usuário logo após sua aquisição na caixa 14 da figura 5.1.

Antes de realizar a medida, ou seja na configuração desta, o usuário introduz parâmetros relativos ao fundo de escala do eletrômetro. Através destes valores, e do sinal de referência, é calculado o valor da corrente. Este é expresso nas caixas representadas pelo item 15 da figura 5.1.

Do mesmo modo que a temperatura, a corrente também é traçada ponto a ponto, porém esta é apresentada na sub-janela 12 (fig. 5.1). Porém ao invés de traçar o real valor da corrente (calculado apartir do fundo de escala), é traçado o sinal de referência. Com isso, esse gráfico não apresentará alterações de fundo de escala.

A fim de auxiliar o monitoramento da medida, o sistema conta ainda com um relógio percentual (item 16 figura 5.1), que indica a porcentagem da medida realizada, e a por se realizar.

Após finalizada a medida o usuário então seleciona o menu arquivo (item 6 figura 5.1), e através de outra janela (descrita posteriormente) entra com o nome dos arquivos nos quais serão armazenados os resultados das medidas.



1 - Barra de título
2 - Caixa de Controle
3 - Caixa Minimizar
4 - Barra dos menus
5 - Menu de programa
6 - Menu de arquivo
7 - Menu de configuração
8 - Menu de janelas
9 - Menu de ajuda
10 - Menu de informação do sistema
11 - Janela de apresentação (temperatura)
12 - Janela de apresentação (eletrômetro)
13 - Quadros de temperatura
14 - Quadro do eletrômetro (absoluto)
15 - Quadros do eletrômetro (leitura)
16 - Relógio percentual
17 - Quadros do fundo de escala
18 - Botão para alterar fundo de escala

Fig. - 5.1 - Janela Principal

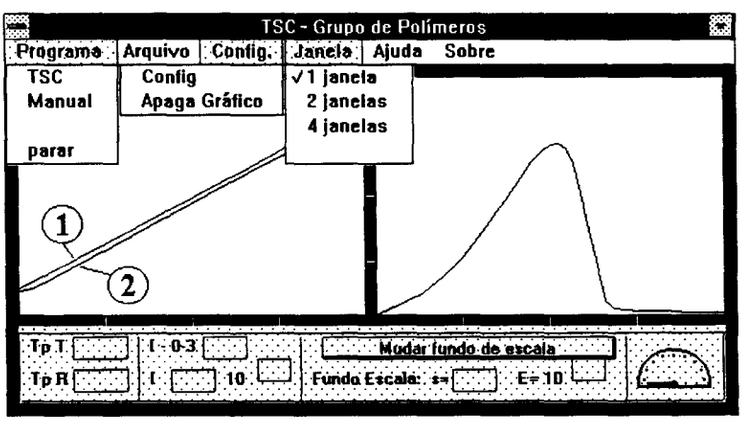


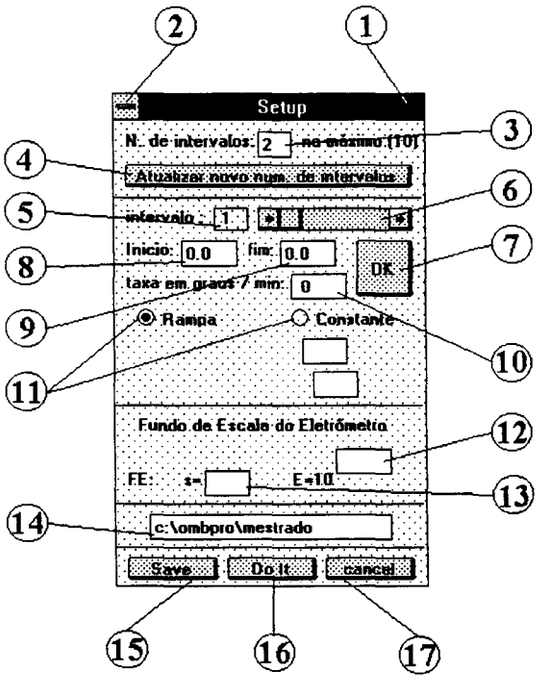
Fig. - 5.2 - Menus da janela principal

5.3.2 - *JANELA DE CONFIGURAÇÃO DO SISTEMA*

Na barra de menus da janela principal do sistema (figura 5.2), encontra-se a opção *Config.* que contém um sub-menu com as opções: *Config* e *Apaga Gráfico.*

Quando o usuário seleciona o item *Apaga gráfico*, são apagadas as sub-janelas (11 e 12 - figura 5.1), e também são zerados os vetores de armazenamento de dados e os contadores, desprezando deste, todos os dados de medidas anteriores.

Ao selecionar o item *Config*, é então apresentada a janela de configuração do sistema (figura 5.3). Através desta, são configurados os parâmetros para a realização de medidas. Esses parâmetros definem qual será o comportamento da estufa (temperatura inicial e final e taxa de aquecimento), e também o fundo de escala e a sensibilidade do eletrômetro durante a medida.



1- Barra de título
2 - Menu de controle
3 - Caixa de número de intervalos
4 - Botão de atualização (num. inter.)
5 - Caixa de intervalo (apres.)
6 - Barra de seleção de intervalo
7 - Botão de atualização
8 - Caixa de temeperatura inicial
9 - Caixa de temperatura final
10 - Caixa de taxa de aquecimento
11 - Botões de seleção do estilo do intervalo
12 - Caixa de fundo de escala
13 - Caixa de sensibilidade
14 - Caixa do diretório do arquivo .ini
15 - Botão de gravação e atualização
16 - Botão de atualização (sistema)
17 - Botão de cancelamento

Fig. - 5.3 - Janela de Configuração

Basicamente, a janela de configuração define ao sistema, a programação do comportamento da estufa, e os parâmetros de leitura do eletrômetro: sensibilidade e fundo de escala. Através destes são efetuados os cálculos a partir do valor de

referência proveniente do eletrômetro, obtendo os valores medidos.

Quanto à configuração da estufa, o usuário deve definir o número de intervalos que deverá ter sua medida e definir seu estilo (rampa ou constante), temperatura inicial e final e taxa de aquecimento. Nós denominamos intervalo, o período em que a estufa apresenta a mesma taxa de aquisição, ou a mesma temperatura, para o estilo constante.

Na configuração da estufa, primeiramente o usuário deve definir o número de intervalos que sua medida deverá apresentar. O sistema possibilita a utilização de 1 a 10 intervalos (em geral as medidas apresentam rampas simples com apenas um intervalo). Para isso, o usuário deve selecionar a caixa de número de intervalos (item 3 da figura 5.3) e então "pressionar" o botão de atualização (4).

Definido o número de intervalos, o usuário deve então configurar cada intervalo. A seleção e atualização de cada intervalo é feita respectivamente através da barra de seleção (6) e do botão de atualização (7). Através da barra de seleção o usuário seleciona o intervalo a ser configurado, e após configurá-lo então, este é atualizado através do botão de atualização.

Caso o número de intervalos seja definido como 1, a barra de seleção e o botão de atualização ficarão inativos, devido ao fato de ser desnecessária sua utilização.

O primeiro elemento a ser configurado é o estilo do intervalo, que pode ser rampa ou constante. A seleção do estilo do intervalo é feita através dos botões de opção (item 11 da figura 5.3). De acordo com a seleção do estilo, serão apresentados ou omitidos alguns ítens na janela de configuração.

A figura 5.3 apresenta a janela de configuração com os ítens do estilo rampa. Neste caso, o usuário deve configurar a temperatura inicial do intervalo, através da caixa de temperatura inicial (item 8 da fig. 5.3), a temperatura final (9), com valores em Celsius, e também a taxa de aquecimento, através da caixa da taxa de aquecimento (10), que deve ser dada em Celsius por minuto.

Caso o usuário selecione o estilo constante (11), então serão ativadas duas caixas, conforme podemos observar na figura 5.4: a caixa de duração do intervalo (1) e a caixa do intervalo de tempo entre os pontos da aquisição (2).

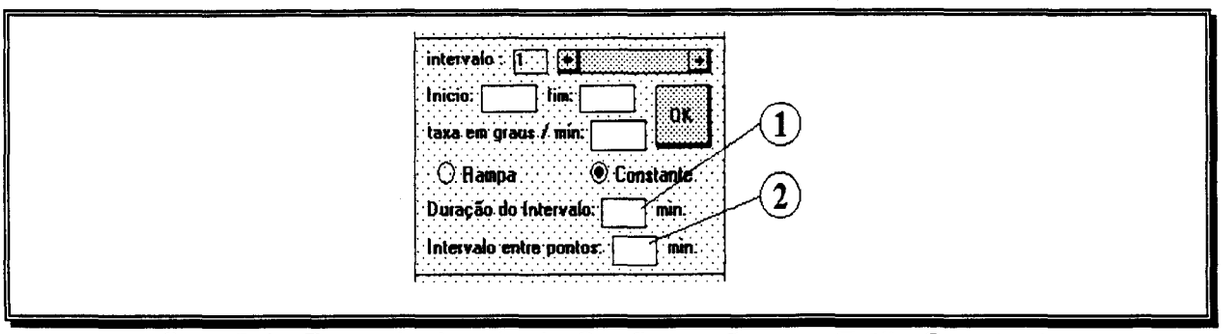


fig. - 5.4 - Parte da Janela de Configuração com Estilo Constante

Para configurar um intervalo com estilo constante, deve-se entrar com o valor da temperatura, o tempo do intervalo e a o intervalo de tempo entre as aquisições. O valor da temperatura é configurado mediante as caixas 8 e 9 da figura 5.3. Devido a temperatura ser constante, ambas devem apresentar o mesmo valor. Como não é utilizado, o valor contido na caixa de taxa de aquecimento (item 9 da figura 5.3), é desconsiderado.

O tempo de duração do intervalo é configurado mediante a caixa de duração do intervalo (item 1 da figura 5.4), cujo valor deve ser dado em minuto. O usuário deve configurar também o intervalo entre as aquisições, através da caixa do intervalo entre pontos (item 2 da figura 5.4). O intervalo entre as aquisições determina o tempo em que o sistema irá então esperar entre uma aquisição e outra.

Na janela de configuração o usuário deve determinar também os parâmetros do eletrômetro: fundo de escala e sensibilidade.

Essa configuração é feita através das caixas (12) e (13) da figura 5.3, onde a sensibilidade na qual estará o eletrômetro durante a medida é configurada pela caixa (12), e o fundo de escala, pela caixa (13).

Depois de entrarem os dados, o usuário então deve atualizá-los para o sistema, finalizando a configuração. Isto pode ser feito através dos botões (15) e (16) da figura 5.3.

O botão de gravação e atualização (15), só é permitido para medidas com apenas um intervalo (item 2 da figura 5.3). Além de atualizar o sistema com os parâmetros definidos pelo usuário, esta opção também armazena os parâmetros de temperatura no arquivo .ini do sistema. Deste modo, toda vez que iniciamos o sistema, ele está configurado com estes dados.

O botão de atualização (16), permitido para medidas de n intervalos, atualiza o sistema com os parâmetros definidos pelo usuário. Nesta opção, caso o usuário

saia do sistema, os parâmetros são perdidos.

Caso o usuário deseja sair da janela de configuração sem atualizar o sistema com os dados nela contidos, ele deverá pressionar o botão de cancelamento (item 17 da figura 5.3) ou ainda pressionar duas vezes a caixa do menu de controle (item 2 da figura 5.3). Deste modo, os dados da janela de configuração serão desprezados, prevalecendo os dados da configuração anterior.

5.3.3 - JANELA DE GRAVAÇÃO DE ARQUIVOS

Depois de finalizadas as medidas (ou caso estas sejam interrompidas pelo usuário), o usuário poderá então armazená-las em arquivo. Deve-se então selecionar o menu Arquivo, na barra de menus (item 4 da figura 5.1), e então será apresentada a janela de gravação de arquivos, conforme podemos observar na figura 5.4.

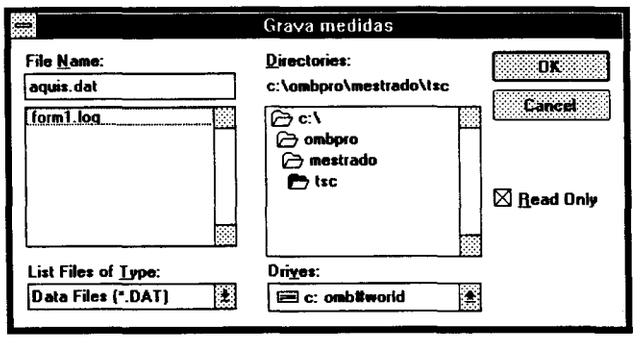


Fig. - 5.5 - Janela de Gravação

A janela de gravação de arquivos foi criada a partir de uma API (Aplication Program Interface), seguindo então o padrão das janelas de manipulação de arquivos do Windows. Deste modo, dispensando a descrição de sua operação.

Depois de definido o diretório, o usuário deverá então digitar o nome básico para os arquivos de suas medidas. A partir deste nome, serão gerados quatro

arquivos, com acréscimos de duas letras ao final do nome, a fim de sua diferenciação. Estes arquivos contém as curvas resultantes das medidas: temperatura X tempo (teórica e prática), corrente X tempo e corrente X temperatura, e suas respectivas abreviações: TI, TR, CC e CT.

A extensão dos arquivos adotada é o .DAT, e seus formatos obedecem aos padrões de alguns "softwares" populares, para tratamento de dados científicos. (duas colunas em código ASCII).

Para exemplificar, vamos supor que o usuário tenha denominado o arquivo resultante ,de MEDA01 (na caixa File Name da figura 5.5). Os arquivos resultantes para este exemplo estão retratados na tabela abaixo.

Como o sistema utiliza dois caracteres para a discriminação dos arquivos resultantes, e o DOS tem um limite de oito caracteres, então o usuário deverá entrar com um nome de no máximo seis caracteres.

Conteúdo do arquivo	Denominação
temperatura X tempo (ideal)	MEDA01TI.DAT
temperatura x tempo (real)	MEDA01TR.DAT
corrente x tempo	MEDA01CC.DAT
corrente x temperatura	MEDA01CT.DAT

Tabela - 5.1 - Denominação dos Arquivos Resultantes

5.3.4 - ALTERAÇÃO DE FUNDO DE ESCALA

O eletrômetro utilizado no sistema, é um modelo analógico, sem interfaces ou dispositivos de controle elétricos. Sendo a saída do sinal de referência o único canal de comunicação com outros aparatos (item 3.5.2).

Devido a esta falta de conexão entre o sistema e o controle do eletrômetro (sensibilidade e fundo de escala), este controle fica a cargo do usuário. Por outro lado, o sistema oferece mecanismos que auxiliam a alteração da configuração do eletrômetro, sem acarretar perda de medidas. O sistema é provido de um método

para a alteração manual, e um automático.

O método de alteração manual é executado quando o usuário seleciona o botão (18) da figura 5.1. É então apresentada a janela de alteração de fundo de escala e sensibilidade, descrita pela figura 5.6.

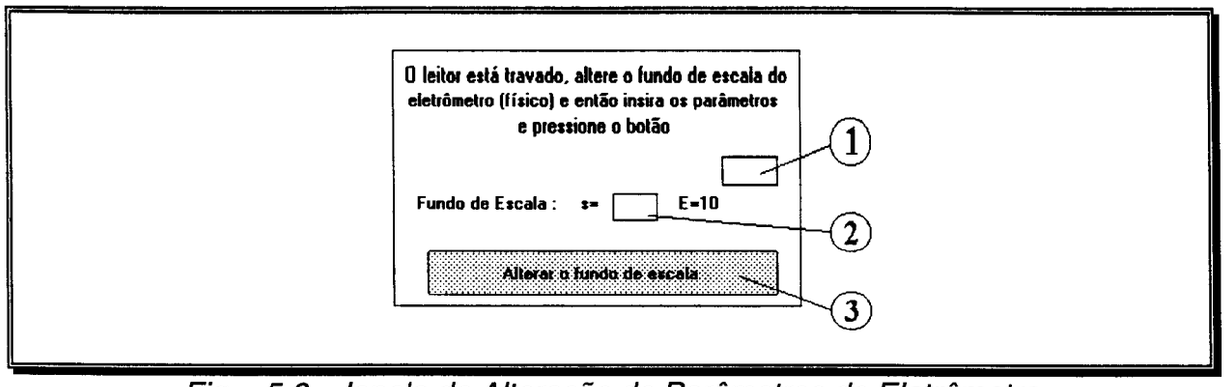


Fig. - 5.6 - Janela de Alteração de Parâmetros do Eletrômetro

Enquanto está sendo exibida a janela de alteração (fig. 5.6), o sistema continua executando a medida em multitarefa, e os valores lidos do eletrômetro são desconsiderados (já que o aparelho pode estar desligado ou com outra escala), sendo armazenado o último valor lido antes da execução do módulo de alteração.

O usuário deve então ajustar a nova sensibilidade e fundo de escala do eletrômetro, e atualizar o sistema com estes novos parâmetros. A nova sensibilidade deve ser atualizada a partir da caixa de sensibilidade (2) e o fundo de escala através da caixa de fundo de escala (1). Então o usuário deve "pressionar" o botão *Alterar o Fundo de Escala* (3). O sistema é finalmente atualizado com os novos parâmetros, voltando então ao seu funcionamento normal.

Quando a leitura do eletrômetro ultrapassa 90% do fundo de escala, o sistema executa o módulo automático de alteração de fundo de escala. É exibida então uma janela (caixa de diálogos) que avisa o operador que o aparelho está por saturar o fundo de escala. A janela de aviso, apresentada na figura 5.7, fica alternando suas cores entre o vermelho e o amarelo, enquanto um "beep" é emitido, a fim de chamar a atenção do usuário.

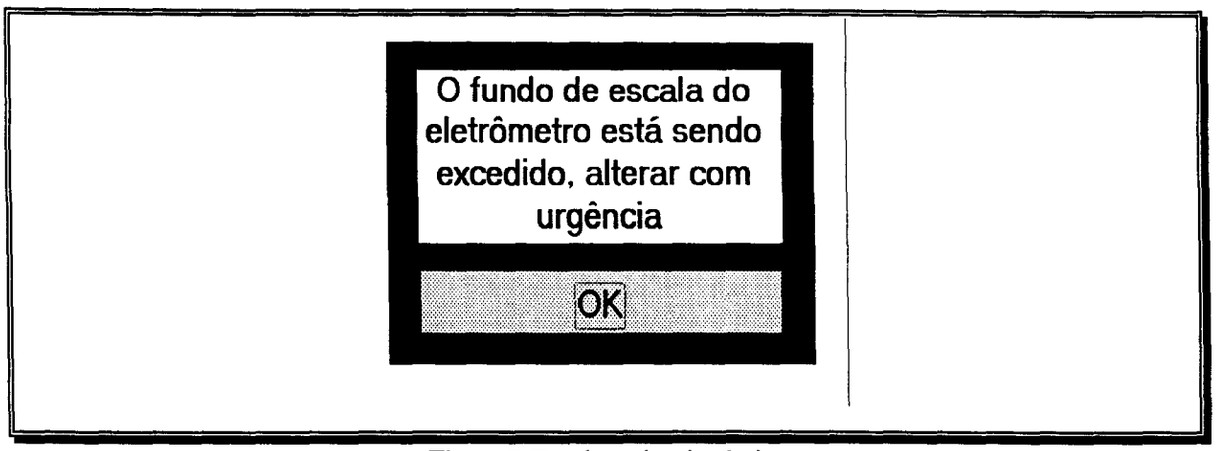


Fig. - 5.7 - Janela de Aviso

Na janela de aviso, o usuário deve "pressionar" o botão OK. A seguir, é apresentada a janela de alteração (fig. 5.6). Depois de alterada a configuração do eletrômetro e atualizados seus parâmetros no sistema, a medida volta então ao curso normal.

5.3.5 - JANELAS DIVERSAS

Descreveremos a seguir, as demais janelas de operação do sistema.

Além do controle automático da temperatura da estufa, o usuário pode também determinar uma temperatura específica, à qual a estufa deve obedecer, e este valor é então transferido ao controlador acarretando no estabelecimento dessa temperatura.

Este controle "manual" é provido de maneira amigável e interativa através da janela de controle manual, apresentada na figura 5.8. A apresentação desta janela ocorre quando o usuário seleciona o item Manual do menu Programa (figura 5.2).

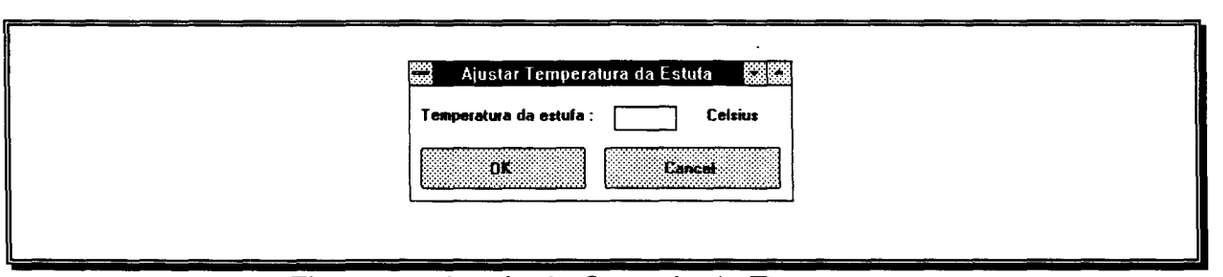


Fig. - 5.8 - Janela de Controle de Temperatura

Informações sobre o sistema podem ser obtidas através do item Sobre do

menu principal (figura 5.1). Será então exibida a janela de informação do sistema (figura 5.9)

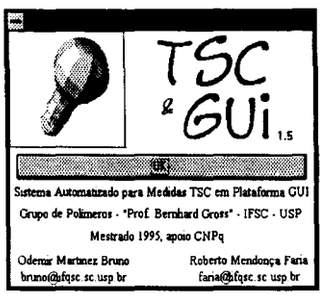


Fig. - 5.9 Janela de Informação Sobre o Sistema

O sistema utiliza um arquivo de ajuda, capaz de esclarecer algumas dúvidas que o usuário venha a ter, assim como efetuar o treinamento da operação básica do sistema. Para acessá-lo o usuário deve selecionar o item *Ajuda* (9) da barra de menus (4) da figura 5.1.

Para tornar o arquivo de ajuda mais dinâmico, implementamo-lo em forma de hipertexto, através da utilização de objetos ligados (OLE) no editor de texto padrão do Windows, o Write.

Com a implementação do arquivo de ajuda hipertexto no formato Write, este fica em aberto, ou seja, novas versões do arquivo de ajuda podem facilmente ser utilizadas (bastando terem o mesmo nome da anterior), assim como poderão ser complementados os itens do menu, ou novos itens acrescidos. Deste modo, o arquivo de ajuda fica bastante versátil, possibilitando sua evolução através do próprio usuário. Abaixo é apresentada na figura 5.10 a janela do aplicativo Write, com o arquivo de ajuda.

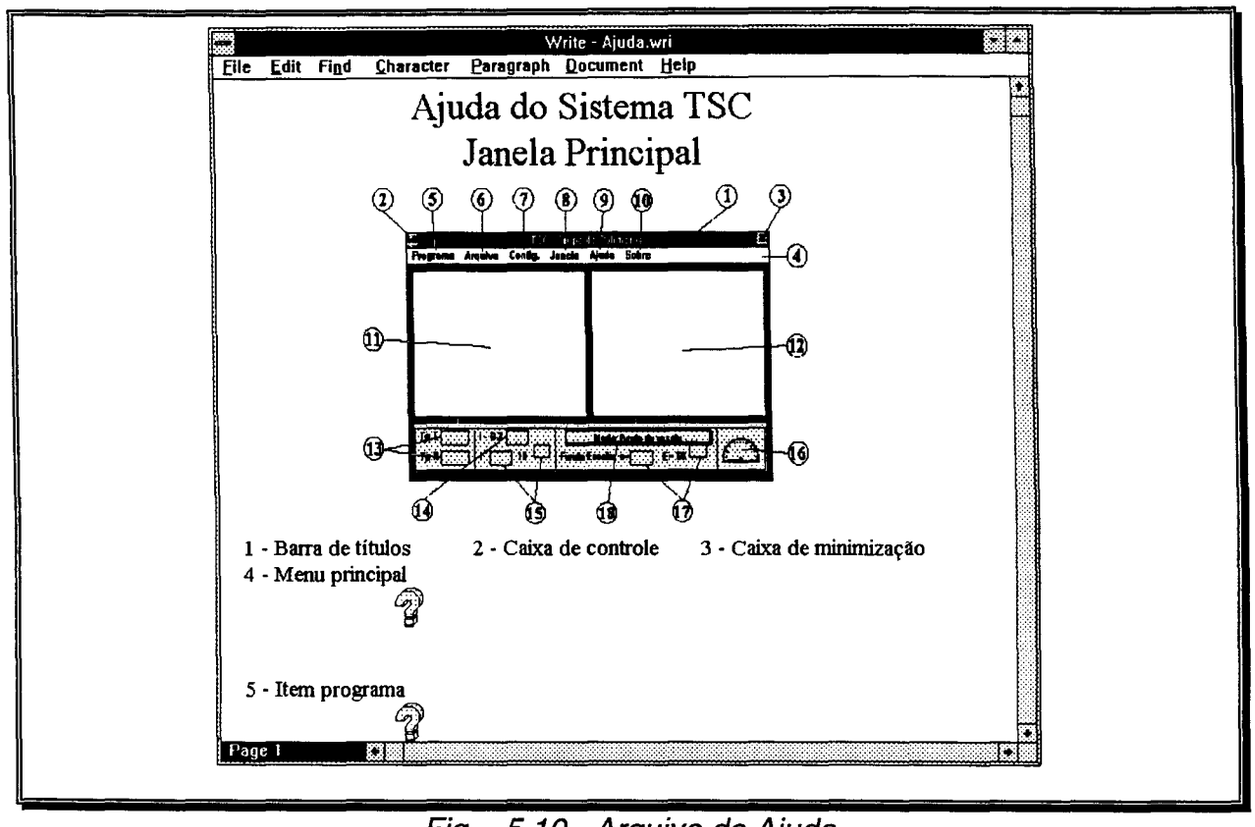


Fig. - 5.10 - Arquivo de Ajuda

5.4 - PROCESSAMENTO ESPECÍFICO

Como discutimos anteriormente (item 4.6.3), na programação orientada a eventos, a execução do programa não é linear, mas sim aleatoriamente definida pelo usuário através da interface gráfica.

Os eventos principais do sistema, suas conexões e o fluxo de dados entre os eventos são apresentados no diagrama da figura 5.11. Aqui observamos o processo principal, contido na janela principal (5.1), controlando e acessando os demais. Podemos observar também alguns detalhes no evento (ou processo) de medidas, formados por módulos em rotinas tradicionais (abaixo).

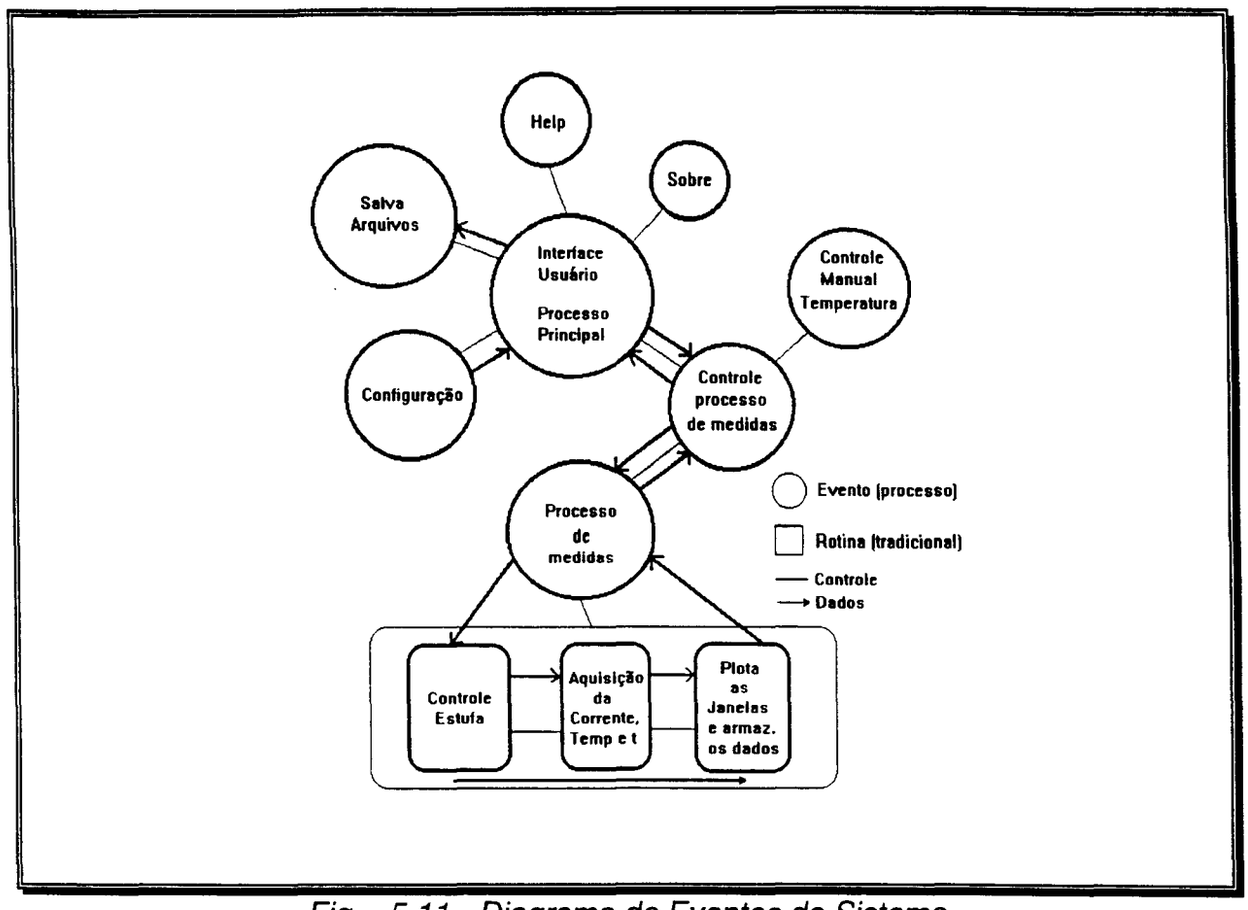


Fig. - 5.11 - Diagrama de Eventos do Sistema

Devemos observar que o módulo de rotinas tradicionais do processo de medidas, da figura 5.11, possui caráter ilustrativo apenas. Neste capítulo descreveremos mais detalhadamente as rotinas que formam o processo de medidas.

A seguir vamos descrever o funcionamento e apresentar alguns fluxogramas, do processo de medidas, responsável pelo módulo de aquisição e controle do sistema. Começamos nossa descrição pelo fluxograma contido na figura 5.12. Denominamos este módulo de módulo inicial, pois é através dele que serão "chamados" os demais módulos.

Quando executamos o processo de medidas, através da seleção do item TSC do menu *Programa*, é executado o módulo inicial (fig. 5.12). O código desta sub-rotina pode ser observado no apêndice C, que contém a listagem do código, na sub-rotina TSC_CLICK.

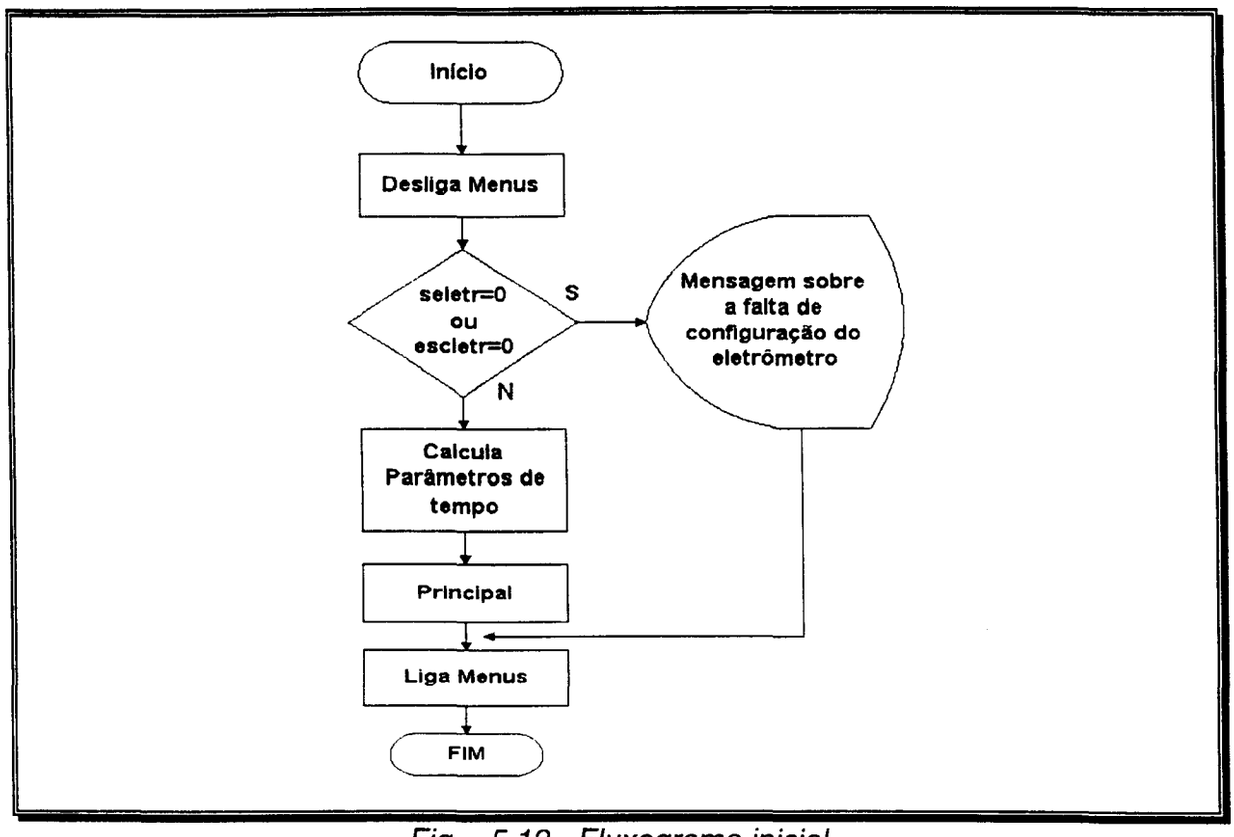


Fig. - 5.12 - Fluxograma inicial

Observando o fluxograma, temos primeiramente o bloco Desliga Menus, ele representa os procedimentos necessários para desativar o funcionamento de todos os menus, com exceção do Menu Programa item TSC (fig. 5.2)

Em seguida temos um desvio que verifica o fundo de escala ou a sensibilidade (variáveis seletr e esceletr). Caso não exista configuração para o eletrômetro (seletr=0 ou esceletr=0), é exibida uma mensagem através de uma caixa de diálogos e então é finalizada a rotina.

O próximo passo é executar o processo calcula parâmetros de tempo, através do qual serão calculados e determinados os parâmetros de tempo para o controle e a aquisição das medidas.

Então é executada a sub-rotina principal, através da qual é realizado o controle da temperatura da estufa e a aquisição de dados.

Após o término da sub-rotina principal, são novamente ativados os menus desativados anteriormente.

O processo calcula parâmetros de tempo é constituído basicamente por duas sub-rotinas: CALCPARM e CALTIME. Estas também se encontram na listagem do programa (apêndice C). A sub-rotina CALTIME é uma função, que é executada na sub-rotina CALCPARM.

A figura 5.13 ilustra o fluxograma da sub-rotina CALCPARM, que como dito anteriormente, é responsável pela determinação e cálculo dos parâmetros de tempo.

Descreveremos agora a sub-rotina CALCPARM, através de seu fluxograma (figura 5.13). Primeiramente, é atribuído 1 para os 10 elementos do vetor subida, através do laço Z, então é executado o procedimento SETPIC. Através deste são configuradas as janelas gráficas, de acordo com os parâmetros determinados pelo usuário (ver item 5.3.2).

O passo em seguida atribui à variável TMPGS o valor determinado pelo função CALTIME, que retorna o tempo de atraso dos processos de controle e aquisição, descreveremos posteriormente essa sub-rotina mais detalhadamente.

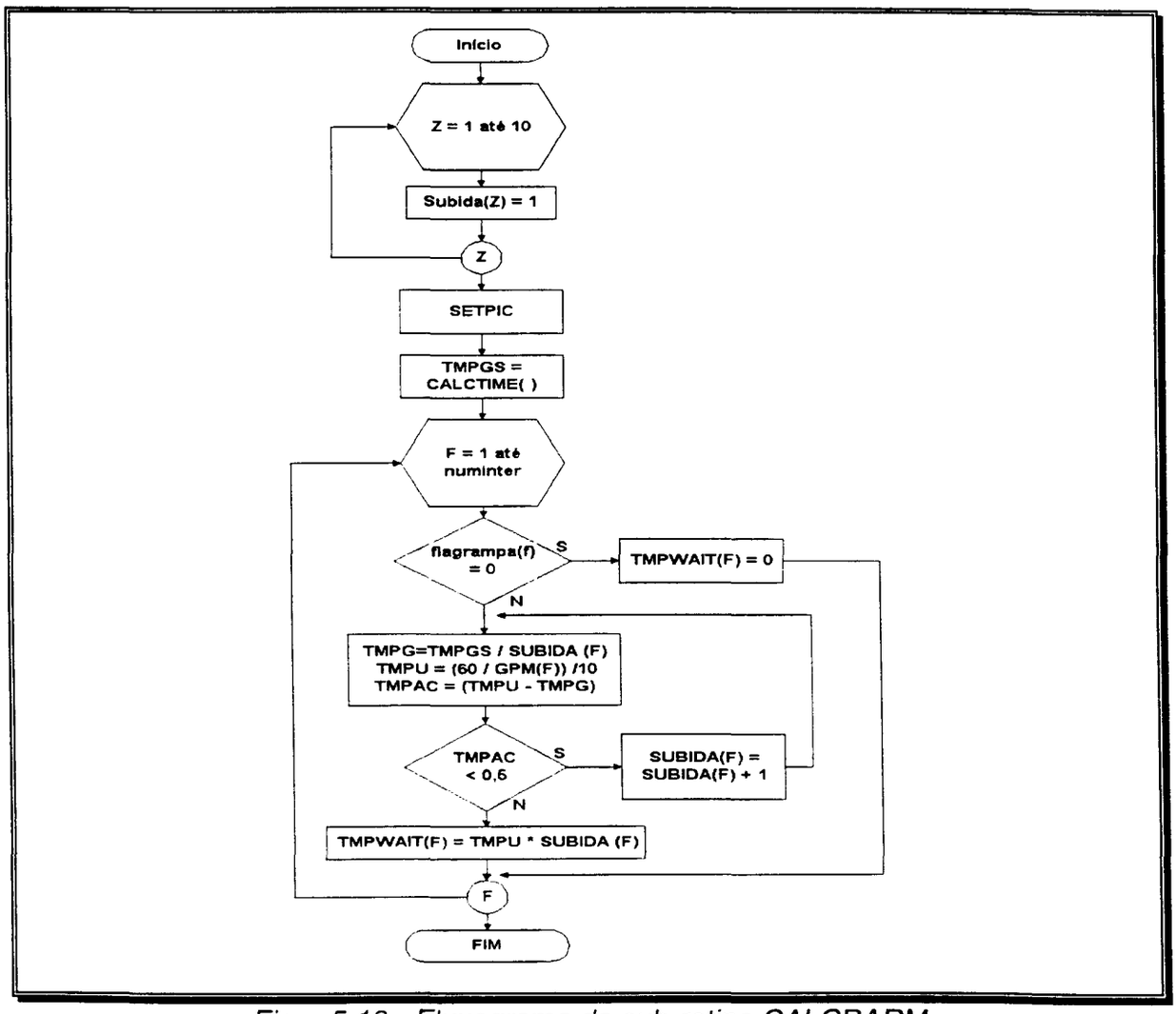


Fig. - 5.13 - Fluxograma da sub-rotina CALCPARM

A seguir temos o laço F, contando de 1 até a variável numinter, que contém o número de intervalos. Neste laço, partir do valor de TMPGS, serão calculados (próximos blocos) o degrau e o tempo de espera, para cada intervalo. O degrau, representado pelo vetor SUBIDA, contém o valor do incremento da subida de temperatura (padrão 1, equivalente a 0,1 ^{o}C), e o tempo de espera, representado pelo vetor TMPWAIT, contém o tempo de atraso, que o sistema deverá efetuar entre uma e outra medida (Descrito posteriormente).

A função CALTIME é responsável pela determinação do tempo de atraso da comunicação do microcomputador com o controlador, da aquisição de dados e da exibição dos resultados. A partir desse cálculo determinamos, como vimos no fluxograma da figura 5.13, os parâmetros necessários para que o sistema em tempo real, realize o controle da temperatura da estufa, obedecendo à taxa programada pelo usuário, e também a aquisição de dados (também em tempo real). A figura 5.14 apresenta o fluxograma da função CALTIME, o qual discutiremos a seguir.

Quando executada a sub-rotina CALTIME, são realizadas algumas medidas e comunicações, a fim de determinar seu tempo de atraso. Estas medidas estão dentro do laço F (primeiro passo do fluxograma).

O laço F conta de 401 até 409, simulando uma rampa de 40,1 ^{0}C a 40,9 ^{0}C. Estes valores foram determinados experimentalmente, pois como veremos posteriormente, não são transmitidos para o controlador.

O primeiro passo dentro do laço, atribui à variável T1 o valor de TIMER. Timer é uma função fornecida pelo Windows (Visual Basic), que retorna o tempo em segundos em que o Windows foi iniciado (elipsed time).

O próximo processo, faz a transferência de valores para o controlador de temperatura. Porém aqui ao invés de ser transferido o valor da temperatura da rampa (variável do laço - F), é transferido o valor da temperatura inicial do primeiro intervalo definido pelo usuário (tinicial(1)). Normalmente este valor é o da temperatura ambiente.

Depois de realizada a transferência, é feita a aquisição do sinal de referência proveniente do eletrômetro (item 3.5.2), pelo processo LECOR, além de exibir os valores relativos à corrente (itens 14 e 15 da figura 5.1), traçar o gráfico na janela da corrente (item 12 da figura 5.1) e armazenar os dados, este processo também efetua o cálculo do valor real da corrente, através dos valores da sensibilidade e

fundo de escala do eletrômetro, definido pelo usuário, e verifica se o fundo de escala do eletrômetro está se excedendo, e caso afirmativo, executa o evento descrito pela figura 5.7.

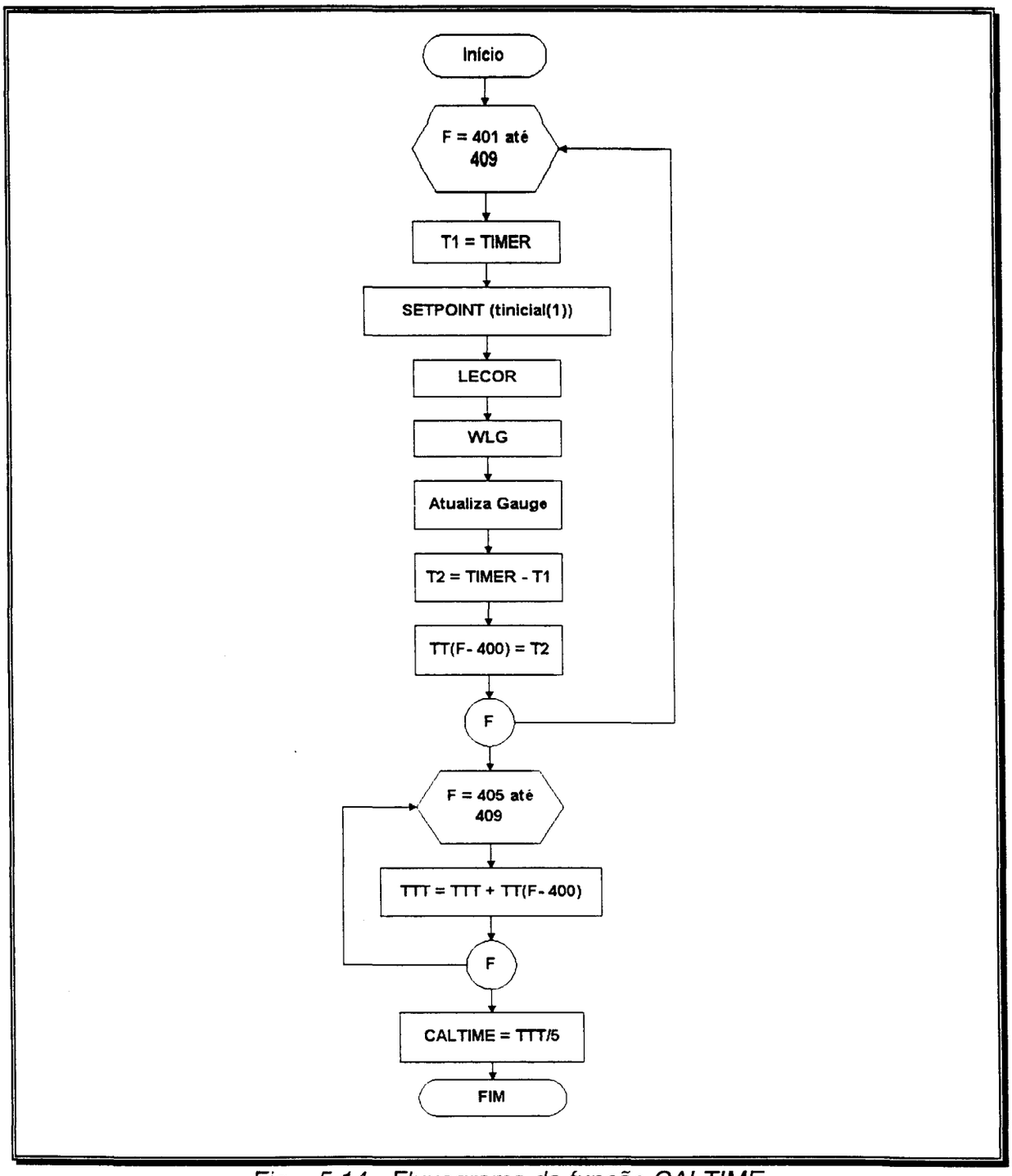


Fig. - 5.14 - Fluxograma da função CALTIME

O processo WLG, executado em seguida, é responsável pela aquisição da temperatura. É também responsável por exibir os valores da temperatura real e ideal

(item 13 da figura 5.1) e traçar seus respectivos gráficos na janela de temperatura (item 11 da figura 5.1) e armazenamento dos dados.

O processo seguinte, atualiza gauge, é responsável pela atualização do estado do relógio de porcentagem (item 16 da figura 5.1)

Finalizando o laço, é atribuído à variável T2 o valor da função TIMER e no passo seguinte, subtraindo T1 de T2, temos o tempo de duração dos processos entre a determinação das variáveis T1 e t2. Este valor é atribuído ao vetor TT na posição determinada pelo indexador F - 400.

Após o término do laço, nós temos novos elementos no vetor TT, cada um com o tempo de duração dos processos interiores do laço. Executamos então um novo laço com o intuito de obtermos a somatória dos cinco últimos elementos do vetor TT. Desconsideramos os quatro primeiros elementos deste vetor, devido a probabilidade de atraso na atualização da memória virtual do Windows (quando executamos o processo primeiramente) ou ainda na primeira leitura (do disco para a memória) de algumas dlls chamadas por esses processos.

Após obtermos a somatória do tempo de duração (variável TTT), nós a dividimos por 5, a fim de obtermos a média da duração, e então atribuímos este valor à variável de retorno CALTIME, fazendo com que este seja o valor devolvido pela função.

Após determinados os parâmetros de tempo, como vimos anteriormente, é então executada a sub-rotina principal. Nela a medida é realizada de fato. A figura 5.15 apresenta o fluxograma da sub-rotina principal.

A rotina principal, está basicamente fundamentada dentro do laço intervalos, como podemos observar no fluxograma. Este laço é definido pelo número de intervalos que a medida deverá conter (ver item 5.3.2). Dentro deste laço existem duas estruturas determinadas por um selecionador por condição (If). O vetor flagrampa contém um indicador do estilo do intervalo. Este pode ser rampa (flagrampa=0) ou constante (flagrampa<>0) (ver item 5.3.2).

Caso o intervalo seja rampa (flagrampa=0), então será selecionada a execução da estrutura de rampa (abaixo do selecionador) caso contrário o selecionador efetuará um desvio para a estrutura de constante (a direita do selecionador).

A estrutura de rampa, é baseada num laço, como podemos observar, que

incrementa o contador F com o valor do vetor subida, indexado pelo variável intervalo (calculado na sub-rotina CALCPARM), indo do temperatura inicial (definida pelo vetor tinicial) até a temperatura final do intervalo (vetor tfinal).

Dentro desse laço, é realizada a aquisição de dados, controle e armazenamento de dados. Podemos observar que os módulos interiores desse, são bem parecidos os módulos da função CALTIME. Isto ocorre, porque utilizamos a função CALTIME para determinar o tempo de atraso provocado por esses módulos de controle, aquisição e armazenamento.

O primeiro processo executado dentro do laço F, atribui à variável TINI1 o valor de TIMER (como vimos antes corresponde ao tempo do sistema - ellipsed). O passo, contém o processo SETPOINT, que transmite ao controlador o valor de F, definindo assim, a temperatura que será imposta a estufa.

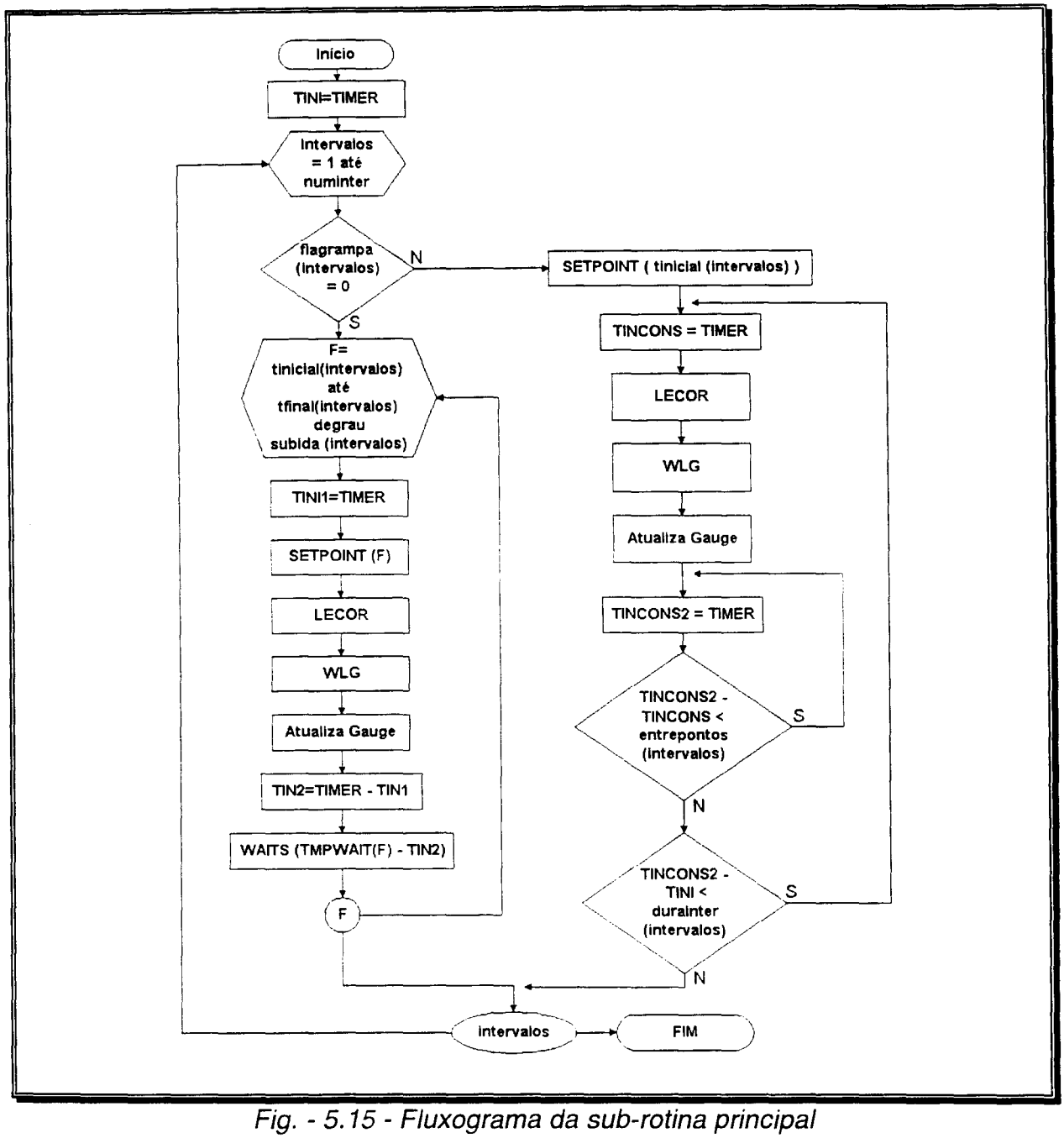


Fig. - 5.15 - Fluxograma da sub-rotina principal

O passo seguinte contém o processo LECOR, que como vimos anteriormente, faz a aquisição do valor da corrente da amostra, exibe-o e armazena-o. O próximo processo WLG, executa as mesmas funções, porém com relação a temperatura, e logo em seguida o relógio de porcentagem é atualizado (atualiza gauge).

Calculamos então o tempo de atraso dos processos, subtraindo TINI1 do valor atual de TIMER. Esse tempo de atraso é atribuído a variável TIN2.

O próximo processo gera um atraso, definido pela subtração de TIN2 (que

contém o tempo de atraso das medidas) do valor do vetor TMPWAIT, que contém o tempo de espera encontrado na sub-rotina CALCPARM (tempo que deveria demorar o laço), em segundos.

A estrutura de intervalo constante (a direita do primeiro selecionador), é um pouco mais simples. Primeiramente é transferido, ao controlador, o valor da temperatura do intervalo, através do processo SETPOINT. A seguir, é atribuído à variável TINCOS o valor de TIMER.

São chamados os processos LECOR, WLG, que como vimos anteriormente são responsáveis respectivamente pelas tarefas associadas a corrente e temperatura. Após realizada as aquisições, o processo atualiza gauge, atualiza o relógio percentual.

Temos então um laço, cuja função é gerar um atraso, para estabelecer a duração correta no intervalo entre uma e outra aquisição. Este atraso, é definido pelo vetor entrepontos, (item 2 da figura 5.4).

Após o atraso ser realizado, é executado um novo selecionador. Este definirá o tempo de duração do intervalo (item 1 da figura 5.4).

CAPÍTULO VI - RESULTADOS EXPERIMENTAIS

6.1 - INTRODUÇÃO

Neste capítulo apresentaremos alguns resultados de medidas realizadas através do Sistema Automatizado de Medidas TSC em plataforma GUI. Com essas medidas poderemos constatar a qualidade do controle e da aquisição do sistema. Primeiramente apresentaremos algumas rampas de temperatura, posteriormente algumas medidas de TSC em amostras de P(VDF/TrFE) copolímero de polifluoreto de vinilideno com trifluoretileno.

6.2 - CONTROLE DE TEMPERATURA

A seguir apresentaremos algumas rampas (curvas - temperatura x tempo) obtidas pelo sistema.

Mostramos na figura 6.1 um gráfico contendo algumas rampas simples (com um único período), com diferentes taxas de aquecimento, de modo a analisar a qualidade do sistema para diversas taxas de aquecimento.

Dentre as curvas, expressas na figura 6.1, quatro foram controladas pelo sistema e uma gerada com a máxima potência do aquecedor, indicada por ***máx***, expressando a curva de aquecimento máximo suportada pela estufa. As quatro rampas controladas pelo sistema, foram geradas com a configuração de taxas de 0,5 °C/min, 1,0 °C/min, 1,5 °C/min e 2,0 °C/min respectivamente indicadas pelos índices ***a***, ***b***, ***c***, ***d***.

Podemos observar a inércia térmica no início das rampas, fazendo com que inicialmente, as rampas, não se comportem de maneira retilínea. Na figura 6.2, são apresentadas as rampas ***a***, ***b***, ***c*** e ***d*** isoladamente, possibilitando a visualização individual de cada rampa, tornando mais nítida a inércia térmica inicial.

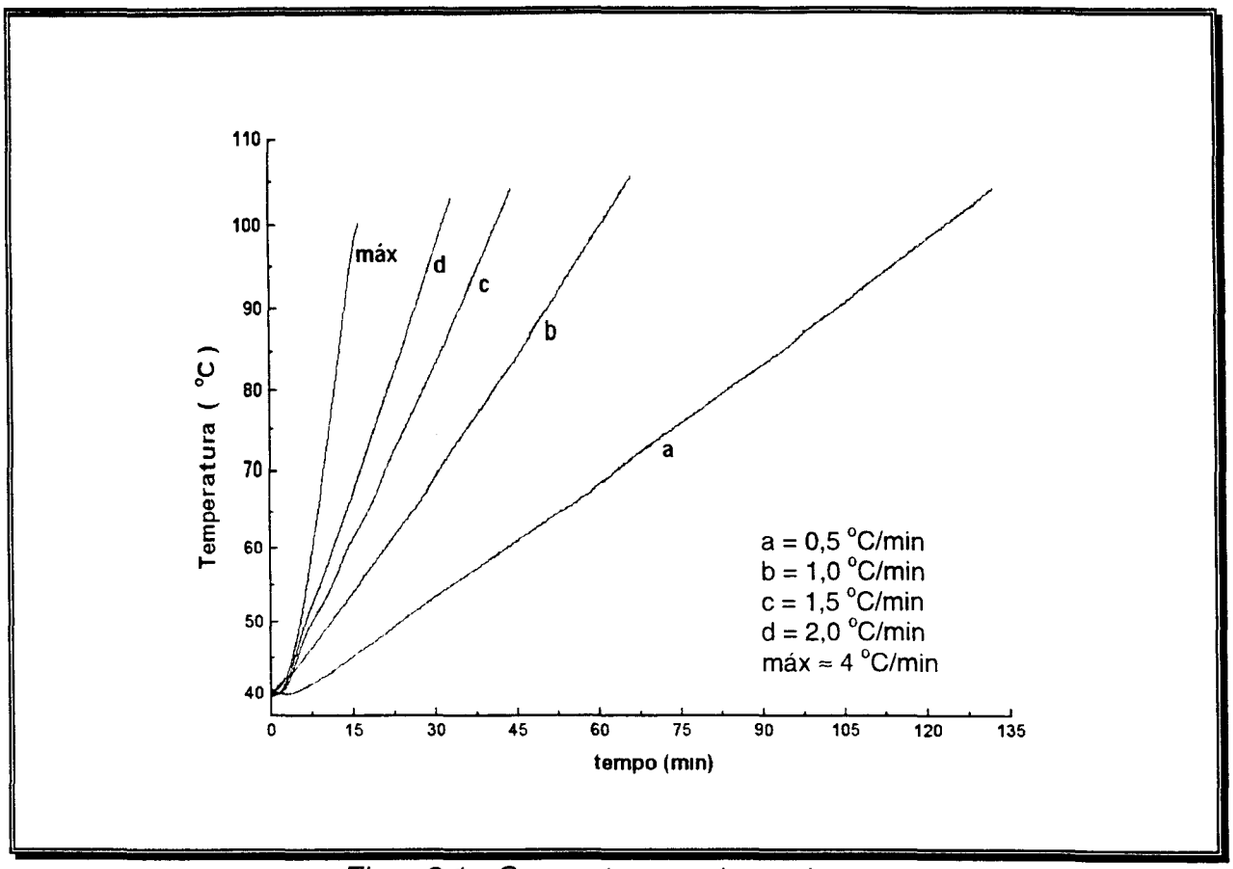


Fig. - 6.1 - Curvas temperatura x tempo

Existem algumas técnicas experimentais para aliviar "manualmente" a inércia térmica. Um exemplo, seria iniciar a medida durante um aquecimento, e aproveitando este, para aliviar o efeito da inércia térmica.

Devemos observar que as rampas aqui apresentadas foram realizadas sem a utilização de nenhuma dessas técnicas experimentais, possibilitando a observação do controle, em sua forma mais genérica.

Para determinar o erro das rampas de temperatura, efetuamos regressão linear (FIT), e deste modo encontramos o erro de ajuste. Estes dados estão contidos na tabela 6.1.

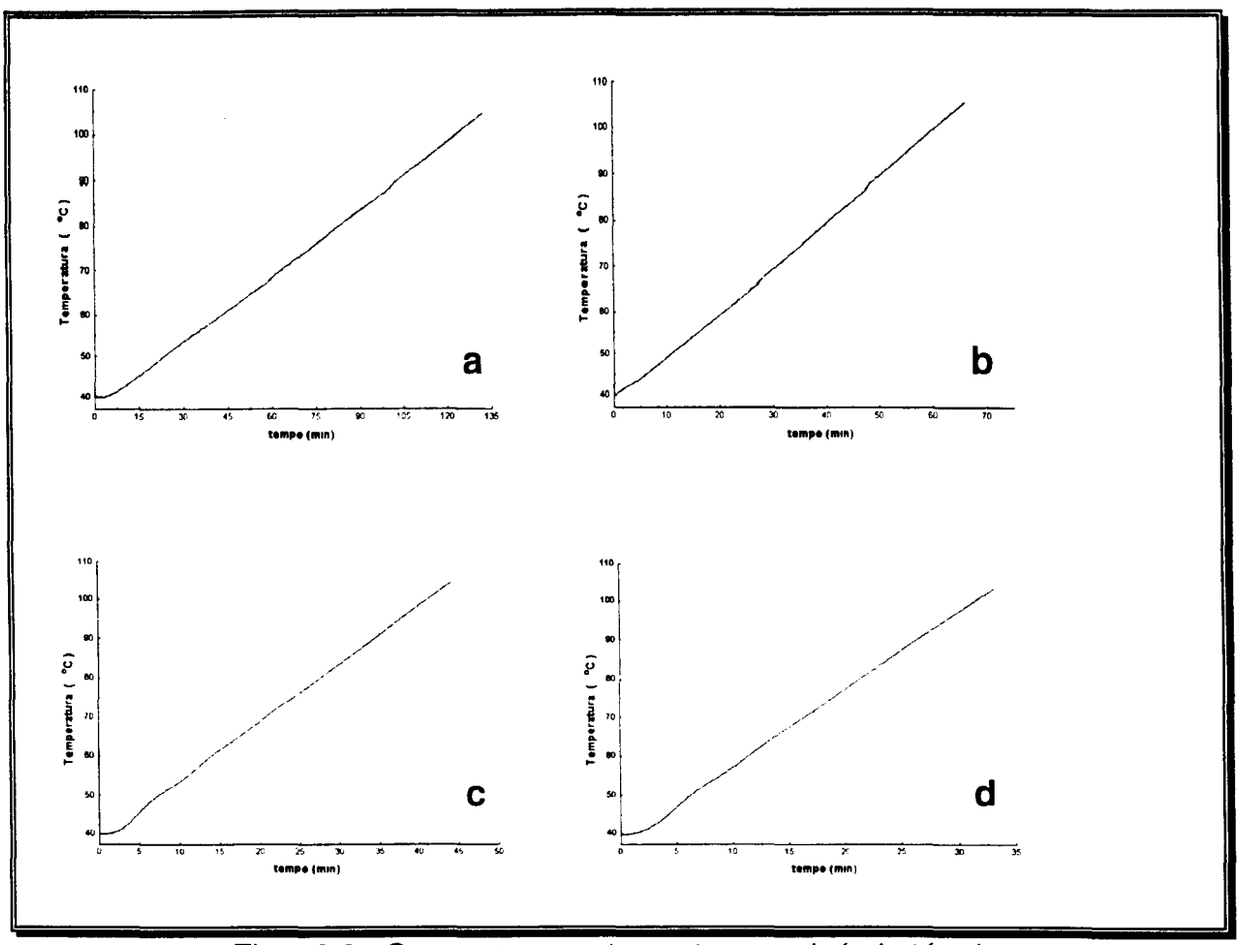


Fig. - 6.2 - Curvas temperatura x tempo - inércia térmica

$$Y = A + BX$$

rampas				
a	38,30	0,01	0,4994	0,0001
b	39,186	0,009	1,0032	0,0002
c	38,43	0,03	1,496	0,001
d	37,24	0,03	2,000	0,002

Tabela - 6.1 - Resultados da regressão linear

Para "visualizar" a regressão linear e deste modo analisar melhor a precisão das rampas, tomamos uma rampa ao acaso (rampa a), e juntamente com esta rampa, traçamos a reta correspondente a sua regressão linear. Amplificamos então uma determinada região deste gráfico. Através desta região amplificada, podemos observar a precisão da rampa, em comparação com a reta formada de sua regressão linear (FIT). Pelo gráfico da região ampliada, podemos constatar, que

nesse intervalo, a rampa diverge menos que 0,3 °C da ideal (reta da regressão linear).

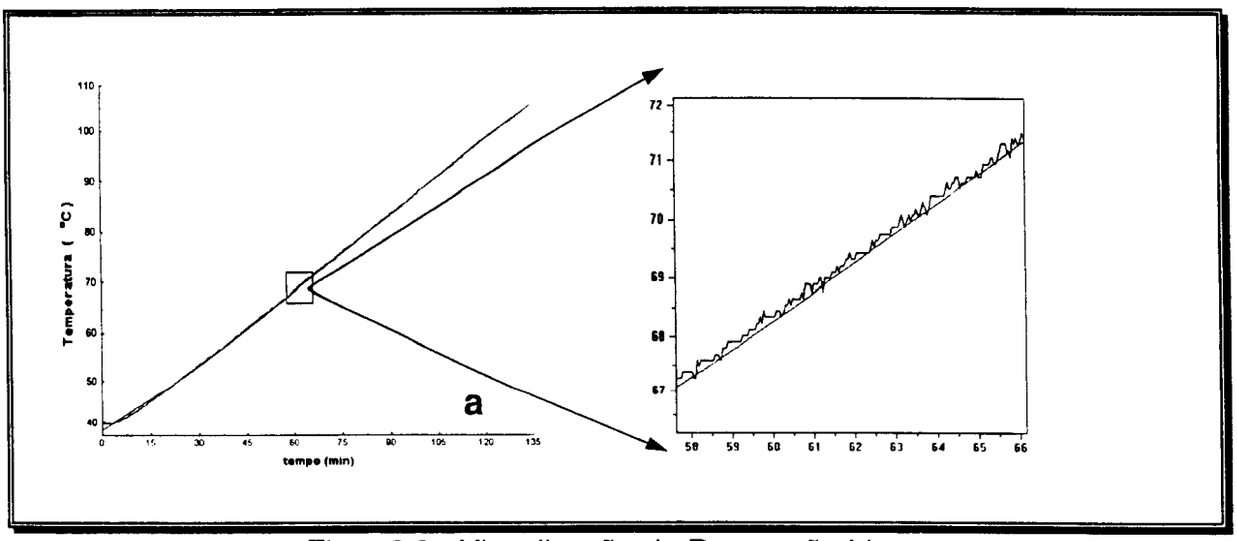


Fig. - 6.3 - Visualização da Regressão Linear

6.2.1 - COMPARAÇÕES COM O SISTEMA ANTERIOR

Para fins comparativos, tomamos algumas medidas de rampas realizadas pelo antigo sistema utilizado no grupo [3], e as comparamos com as realizadas pelo sistema atual.

Colocaremos "lado a lado" os resultados dos dois sistemas e apresentaremos os resultados a seguir.

A figura 6.4 contém três rampas, as quais foram controladas e aquisicionadas pelo sistema anterior (que utilizava plataforma Apple), representadas por **A**, e uma rampa cujo controle e aquisição foi realizado pelo sistema exposto neste trabalho, representada por **B**. Através da observação dessas rampas constatamos visivelmente a qualidade superior da rampa **B**.

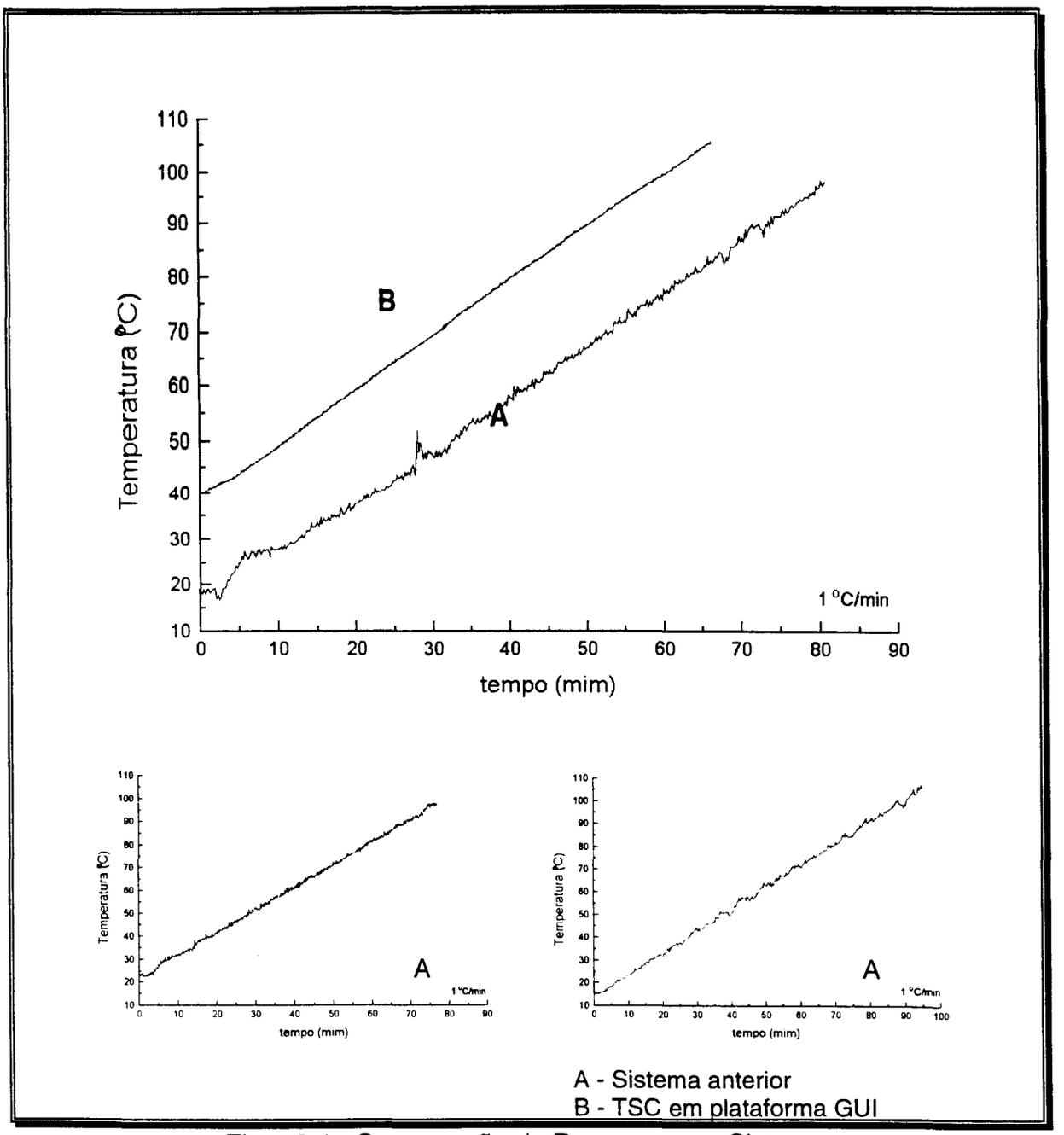


Fig. - 6.4 - Comparação de Rampas entre Sistemas

Para observar melhor a diferença entre as rampas, apresentamos na figura 6.5, uma região da rampa **B** da figura 6.4, e uma região da rampa **A** correspondente, ampliadas e com suas respectivas retas de regressão linear traçadas (do mesmo modo que a figura 6.3). Analisando os gráficos da figura 6.5, podemos claramente constatar a superior qualidade da rampa **B**.

FSC — SERVIÇO DE BIBLIOTECA E INFORMAÇÃO

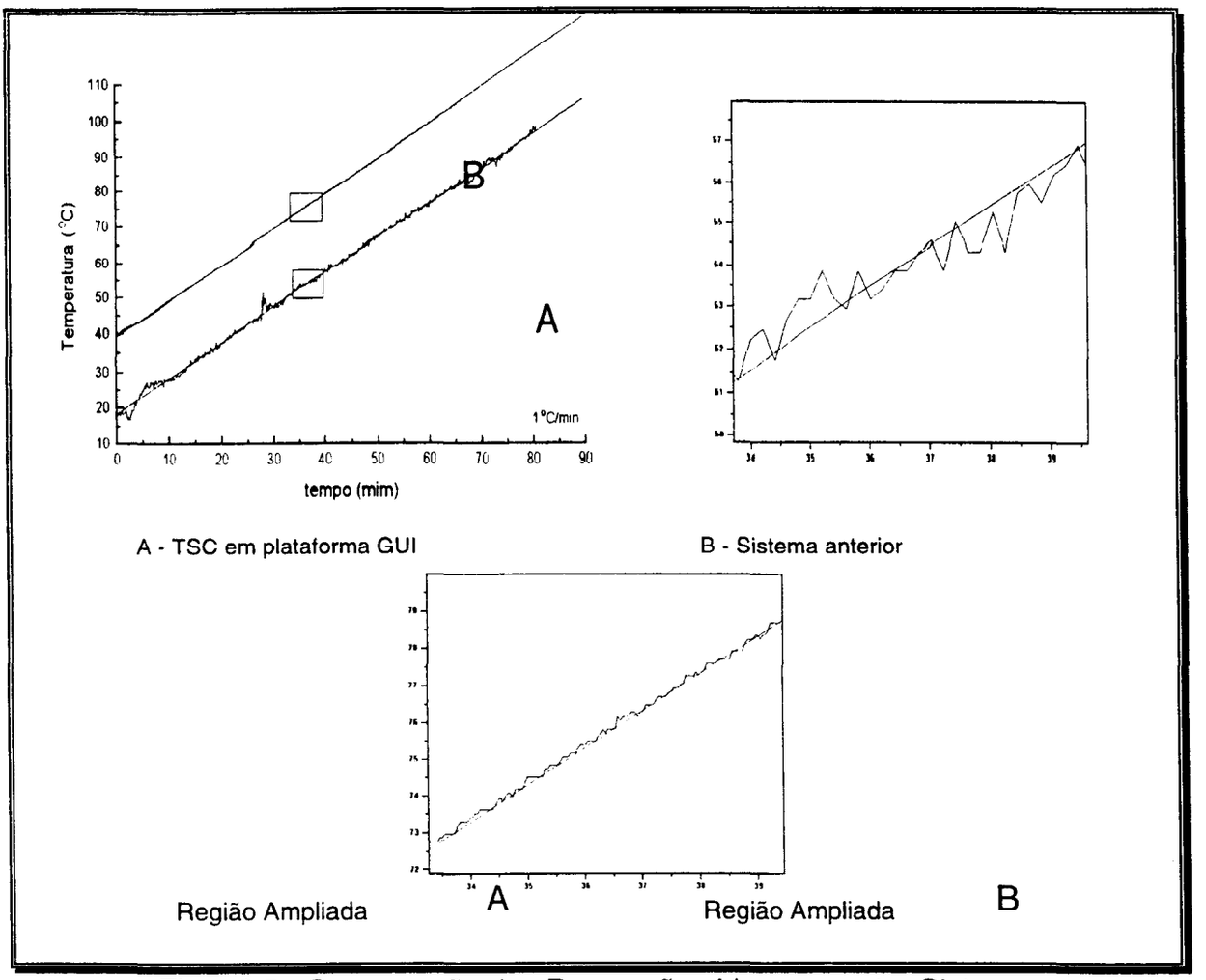


Fig. - 6.5 - Comparação das Regressões Lineares entre Sistemas

6.3 - MEDIDAS TSC

Utilizamos uma amostra de P(VDF/TrFE) copolímero de polifluoreto de vinilideno com trifluoretileno, e realizamos uma medida TSC (conforme a descrição desta no item 2.2), a fim de obter as curvas de despolarização. A amostra de 50 μm, foi polarizada com uma tensão de 900V à temperatura de 90°C, durante 20 minutos.

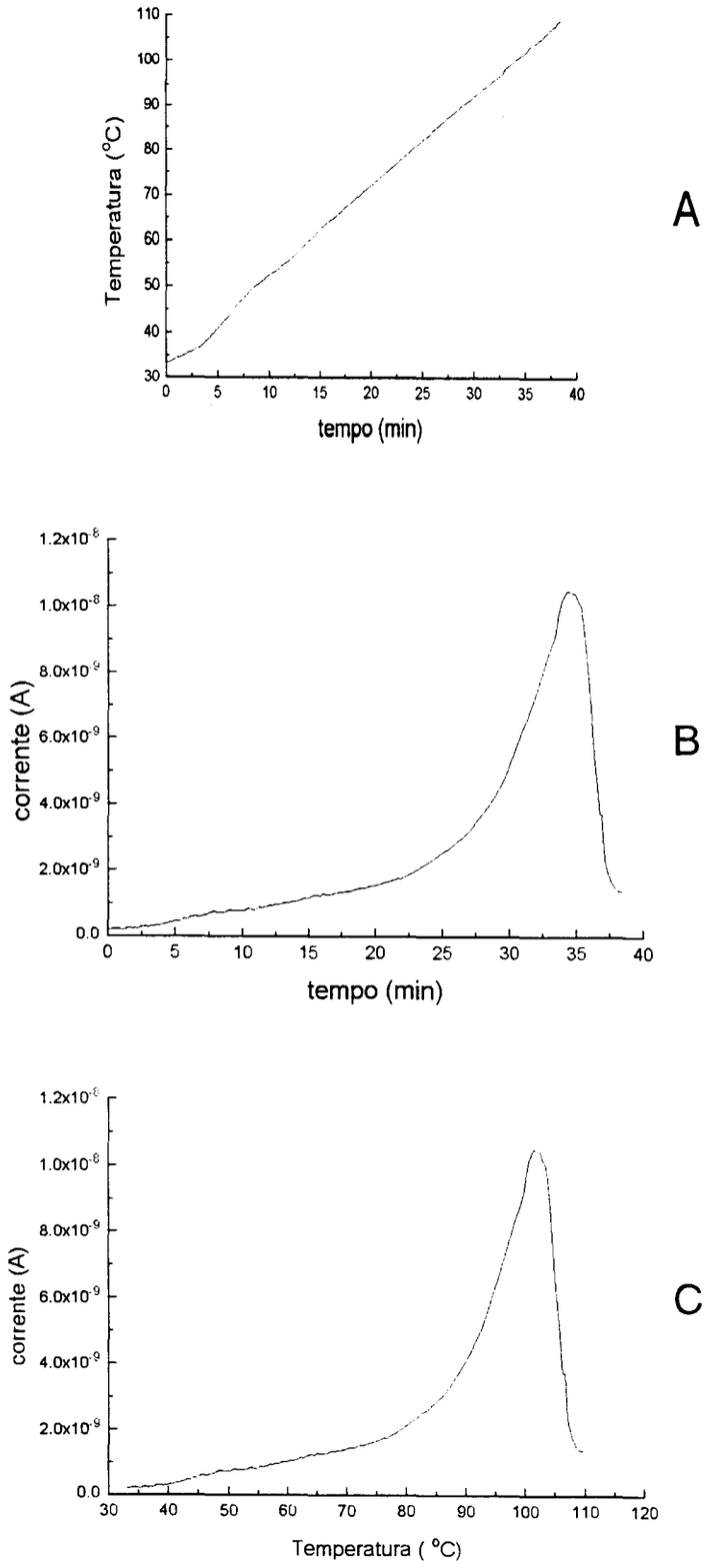


Fig. - 6.6 - Resultado obtido pela medida TSC

Como resultado desta medida obtivemos as curvas da figura 6.6, onde temos além do gráfico da rampa de temperatura (**A**), as curvas da corrente de relaxação da

amostra, em função do tempo (**B**) e da temperatura (**C**).

Como se pode observar a corrente cresce lentamente nos primeiros 25 minutos de medida, ou seja, até aproximadamente 80 ^{0}C, e a partir de então apresenta um pico de corrente bem agudo. Esse pico está relacionado com a bem conhecida transição de fase Ferro-Paraelétrica do material, e observa-se que assim que o pico ocorre, a corrente elétrica tende a zero.

CAPÍTULO VII - CONCLUSÃO

Neste trabalho realizamos um estudo da tecnologia de interface GUI, e através deste, desenvolvemos um sistema de instrumentação eletrônica, destinado ao estudo das propriedades termoelétricas de materiais poliméricos. Como resultado obtivemos um sistema com boa qualidade de medidas, e com uma operação bastante simples e intuitiva, levando à instrumentação eletrônica, os amigáveis recursos da tecnologia GUI. O usuário deste sistema, não necessitará de treinamento prolongado, nem decorar comandos, códigos ou seqüência de teclas; toda a interface é feita de forma clara e objetiva através de janelas e ícones padronizadas sob uma plataforma GUI.

Descrevemos toda a evolução do nosso trabalho, no capítulo II, abordamos sucintamente a técnica TSC, onde comentamos um pouco de seus fundamentos físicos. No capítulo III, descrevemos a implementação física (hardware) do sistema, o modo pelo qual coletamos os dados, e realizamos os controles. No capítulo IV, provemos um estudo da tecnologia GUI, seus fundamentos e técnicas. No capítulo V, foi descrito o software e a operação do sistema. E no capítulo VI, apresentamos os resultados experimentais, e realizamos uma análise em relação ao sistema anterior [3]. Apresentamos ainda, no apêndice C, a listagem dos programas que compõe o sistema.

Embora o desenvolvimento deste sistema tenha sido voltado às técnicas de correntes termoestimuladas, com pequenas alterações de software e hardware, o sistema pode ser modificado para automatizar diferentes técnicas de medidas. Ao invés de controlar a temperatura, é possível mediante alterações, controlar grandezas físicas tais como freqüência, pressão, corrente, força, ddp entre outras, e do mesmo modo realizar a aquisição de diferentes propriedades físicas, analisando o comportamento de um determinado material, sob a alteração das grandezas anteriormente citadas.

A utilização da tecnologia GUI tem mostrado ser uma tendência na computação, porém ainda existe um certo receio à adoção desta na instrumentação. Este receio pode ser percebido, quando se observa um número comparativamente

grande de sistemas em plataforma CLI, que povoam o parque de sistemas de instrumentação eletrônica. Além disso alguns trabalhos têm contextado a utilização de sistemas de intrumentação em microcomputadores baseados em GUI.[20, 21 e 22] Deste modo, este trabalho, tem também, o intuito de apresentar a viabilidade da utilização da tecnologia GUI em sistemas de instrumentação eletrônica de pequeno porte.

Como sugestão para trabalhos futuros, uma boa maneira de aperfeiçoar o sistema, seria incorporar à estufa um dispositivo que auxilie seu resfriamento, possibilitando assim a utilização de rampas negativas com boa qualidade de controle.

Pelas caracteríticas do sistema, é conveniente também um estudo mais aprofundado sobre multitarefa, devido a sua adequação ao sistema. Caso obtivesse êxito, através deste, seria possível desenvolver um sistema semelhante, totalmente multitarefa, que possibiltasse a utilização de vários sistemas sendo executados em tempo real.

REFERÊNCIAS

[1] PEDDIE, Jon Graphical User Interfaces and Graphic Standards New York, McGraw Hill, 1992

[2] PETZOLD, Charles Programando para Windows 3.1 São Paulo, Makron Books, 1993

[3] JAIME, Miriam N. Automatização de Sistemas para Medidas Elétricas em Polímeros (Dissertação de Mestrado) São Carlos, USP - IFQSC, 1989

[4] TURNHOUT, J. van Thermally Stimulated Discharge of Electrets In: Sessler, G. M., ed. Electrets Berlin, Springer, 1980 p. 81-215

[5] SESSLER, G. M. Charge Storage In: Mort, J. & Pfister, G., ed. Electronic Properties of Polymers New York, John Wiley and Sons, 1982 p. 70-100

[6] OGATA, Katsuhiko Modern Control Engineering New York, Prentice - Hall, 1990

[7] LEVENTHAL, Lance Lance Leventhal's 80386 Programming Guide... Toronto, Bantam Book, 1988

[8] NICHOLS, Elizabeth A. ; NICHOLS, Joseph C. ; MUSSON, Keith R. Data Communications for Microcomputers New York, McGraw Hill, 1982

[9] NOGUEIRA, José de Souza Cargas Espaciais e Efeitos de Água Dissociada em Filmes de Poli(Fluoreto de Vinilideno) (Dissertação de Mestrado) São Carlos, USP - IFQSC, 1991

[10] WHITE, Richard B. Thermoresistive Systems In: CONSIDINE, Douglas M., ed. Process Instruments and Controls Handbook 2 ed. New York, McGraw Hill, 1974(57) cap. 2, p. 2-36-40

[11] SHEINGOLD, Daniel H., ed. Analog - Digital Conversion Handbook 2 ed. Norwood, Analog Devices, 1976 pt. 2, cap. 1, p. II-1-9

[12] RONY, Peter R. Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing Indianapolis, Howard W. Sams, 1980 v. 1, p. 359-63

[13] NORTON, Peter ; YAO, Paul Programando em Borland C++ para Windows Rio de Janeiro, Berkeley, 1992 cap 1

[14] HAYES, Frank From TTY to VUI BYTE, v. 15, n. 4, p. 205-6, 208, 210-11, abr. 1990

[15] BAKER, S. J. Code generation tools for Windows/PM development IEE Colloquium on 'Software tools fo Windows-Based Application Development' London, mar. 1990

[16] COX, Brad J. Programação Orientada para Objetos Rio de Janeiro, Makron Books, 1991 cap. 3-4

[17] HURSON, A. R. ; PAKZAD, S. H. ; CHENG, J. Object-Oriented Database Managemment Systems: Evolution and Performance Issues Computer, v. 26, n. 2, p. 48-60

[18] BONNER, Paul Programação Visual: Promessa/Realidade Windows Computing, v. 2, n.1, p.64-6, 68, 70, 72, 74-5

[19] CRAIG, John Clark Microsoft Visual Basic: versão 3 São Paulo, Makron Books, 1994

[20] Baer, T. Can processs control do windows ? Managing Automation, v. 8, n. 2, p. 51-2

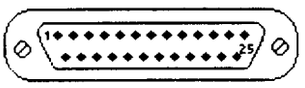
[21] Tomsick, C. Windows 3.1: is it ready for industrial action ? I&CS v. 65, n. 6, p. 61-5

[22] Potter, D. Data acquisition developers overcome Windows' limitations I&CS v. 65, n. 12, p. 41-3

APÊNDICE A

PADRÃO RS-232C

Abaixo é apresentado um conector DB25, utilizado para a conexão das portas seriais RS-232C.

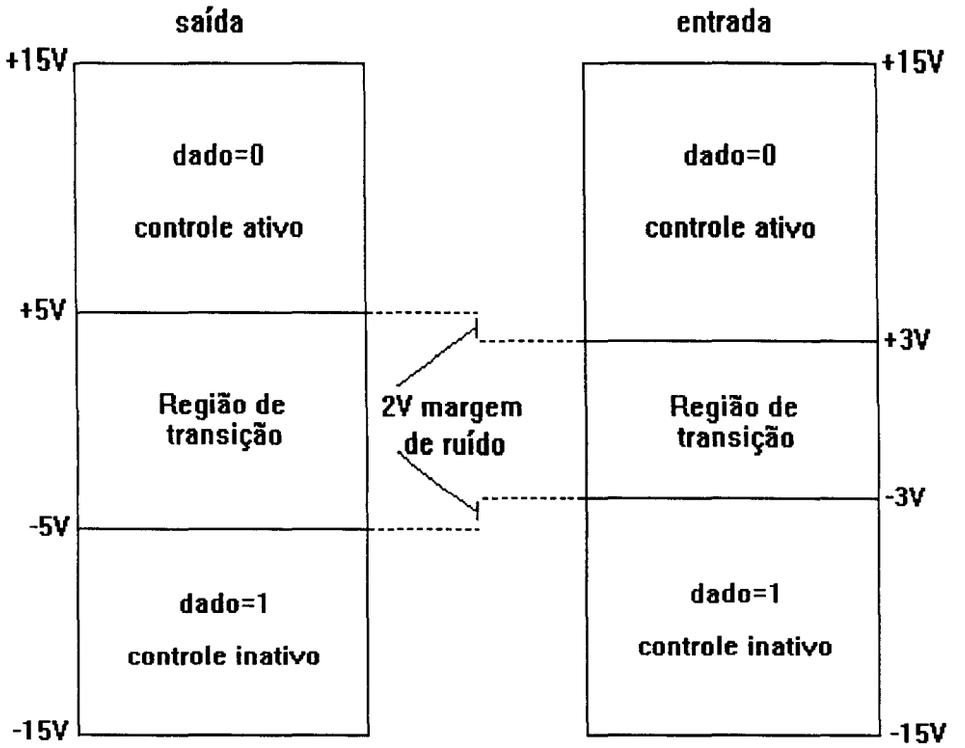


A tabela seguinte fornece as funções dos sinais nos pinos do conector padrão db-25 da interface RS-232C. Os sinais indicados com S são as saídas e os sinais indicados com E as entradas.

PINO	SINAL	E/S	Função
1	TRRCH	-	Terra chassi
2	TxD	S	Dados transmitidos
3	RxD	E	Dados recebidos
4	RTS	S	Requisição para transmitir (Request to send)
5	CTS	E	Permissão para transmitir (Clear to send)
6	DSR	E	Modem pronto (Data set ready)
7	TRR	-	Terra de sinal
8	DCD	E	Detecção da portadora (Data carrier detect)
20	DTR	S	Equipamento terminal pronto (Data terminal ready)
22	RI	E	Chamada telefônica no modem (ring indicator)

Os níveis de tensão correspondentes aos controles e dados no padrão RS-

232C são os seguintes:



APÊNDICE B

```
#-----------------------------------
# HELLO.MAK Arquivo Para a Criacao
#-----------------------------------

hello.exe : hello.obj hello.def
   $(WINLINK) hello, hello, NUL, $(WINLIB), hello
   rc -t hello.exe

hello.obj : hello.c
   $(WINCC) hello.c


;-----------------------------------------
; HELLO.DEF Arquivo de definicoes do modulo
;-----------------------------------------

NAME        HELLO

DESCRIPTION   'Programa Hello World"
EXETYPE       WINDOWS
STUB          'WINSTUB.EXE'
CODE          PRELOAD MOVEABLE DISCARDABLE
DATA          PRELOAD MOVEABLE MULTIPLE
HEAPSIZE      1024
STACKSIZE     8192


/*-------------------------------------------------------
  HELLO.C -- Mostra "Hello World" na area do cliente
-------------------------------------------------------*/

#include <windows.h>

long FAR PASCAL _export ProcJan (HWND, UINT, UINT, LONG) ;

int PASCAL WinMain (HANDLE hCopia, HANDLE hCopiaAnterior,
            LPSTR  lpszParamCmd, int nCmdMostrar)
   {
   static char szNomeAplic[] = "Hello World" ;
   HWND       hjan ;
   MSG        msg ;
   WNDCLASS   classejan ;

   if (!hCopiaAnterior)
      {
```

```
    classejan.style        = CS_HREDRAW | CS_VREDRAW ;
    classejan.lpfnWndProc  = ProcJan ;
    classejan.cbClsExtra   = 0 ;
    classejan.cbWndExtra   = 0 ;
    classejan.hInstance    = hCopia ;
    classejan.hIcon        = LoadIcon (NULL, IDI_APPLICATION) ;
    classejan.hCursor      = LoadCursor (NULL, IDC_ARROW) ;
    classejan.hbrBackground = GetStockObject (WHITE_BRUSH) ;
    classejan.lpszMenuName  = NULL ;
    classejan.lpszClassName = szNomeAplic ;

    RegisterClass (&classejan) ;
        }

  hjan = CreateWindow (szNomeAplic,  // nome da classe da janela
                  "Hello World",     // titulo da janela
          WS_OVERLAPPEDWINDOW, //  estilo da janela
          CW_USEDEFAULT,      //  posicao x inicial
          CW_USEDEFAULT,      // posicao y inicial
          CW_USEDEFAULT,      // tamanho x inicial
          CW_USEDEFAULT,      // tamanho y inicial
          NULL,               // handle da janela-mae
          NULL,               // handle do menu da janela
          hCopia,             // handle da copia do programa
                  NULL) ;                    // parametros de criacao

  ShowWindow (hjan, nCmdMostrar) ;
  UpdateWindow (hjan) ;

  while (GetMessage (&msg, NULL, 0, 0))
     {
     TranslateMessage (&msg) ;
     DispatchMessage (&msg) ;
     }
  return msg.wParam ;
  }

long FAR PASCAL _export ProcJan (HWND hjan, UINT mensagem, UINT wParam,
                                 LONG lParam)
  {
  HDC       hdc ;
  PAINTSTRUCT ps ;
  RECT  retang ;

  switch (mensagem)
     {
     case WM_PAINT:
            hdc = BeginPaint (hjan, &ps) ;

        GetClientRect (hjan, &retang) ;
```

```
            DrawText (hdc, "Hello World!", -1, &retang,
                        DT_SINGLELINE I DT_CENTER I DT_VCENTER) ;

            EndPaint (hjan, &ps) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }

return DefWindowProc (hjan, mensagem, wParam, lParam) ;
}
```

APÊNDICE C

Arquivo TSC.MAK:

```
FORM1.FRM
C:\WINDOWS\SYSTEM\GAUGE.VBX
C:\WINDOWS\SYSTEM\THREED.VBX
C:\WINDOWS\SYSTEM\CMDIALOG.VBX
C:\WINDOWS\SYSTEM\MSCOMM.VBX
GLOBAL.BAS
VBINI.VBX
WDAQ_VB.BAS
FORM2.FRM
FORM3.FRM
FORM6.FRM
FORM4.FRM
FORM5.FRM
ProjWinSize=71,573,226,177
ProjWinShow=2
IconForm="Form1"
Title="TSC"
ExeName="TSC.EXE"
```

Arquivo GLOBAL.BAS

```
Declare Function OutG Lib "C:\ombpro\mestrado\tsc\genport.dll" (ByVal a As Integer, ByVal b As Integer) As Integer
Declare Function InG Lib "c:\ombpro\mestrado\tsc\genport.dll" (ByVal a As Integer) As Integer

Global subida(10) As Integer 'numero de graus a acrescentar por intervalo
Global espera As Integer
Global Igraph  As Integer
Global arquivo As String
Global inicio As Integer
Global fim As Integer
Global cini As Integer

Global numinter As Integer 'número de intervalos
Global flagrampa(10) As Integer 'flag se 0 rampa se 1 constante
Global gpm(10) As Single 'graus por minuto defido pelo usuário
Global tinicial(10)  As Integer 'temperatura inicial do intervalo
Global tfinal(10) As Integer 'temperatura final do intervalo
Global tmpwait(10) As Single 'tempo de espera calculado
Global durainter(10)  As Single 'tempo de  duracao do intervalo (caso tenha)
Global entrepontos(10) As Single 'tempo do intervalo entre os pontos (constante)

'espera comunicacao
Global waitcom As Single

'fundo de escala do eletrometro
```

```
Global Seletr As Single
Global Esceletr As Single
Global fconver As Single
Global oldvolt As Single 'último valor lido em volts


'flags
Global paraja As Integer
Global eletrotrava As Integer
Global excedeu As Integer

Function calconfe (s As Single) As Single
'funcao que calcula o referencial do eletrometro
 Dim vesc, mu, dd
 mu = 1
 vesc = s * 100
 If vesc Mod 3 <> 0 Then
   mu = 3.33333
   dd = s / 10
 Else
   mu = 1
   dd = s / 3
 End If
 calconfe = mu * dd
End Function

Function getbuf ()
'funcao que retorna conteudo do buffer
'da serial
 b$ = "null"
 If form1.Comm1.InBufferCount Then
  b$ = form1.Comm1.Input
 End If
 If b$ <> "null" Then
   bb = Asc(b$)
 Else
   bb = 0
 End If
 getbuf = Val(bb)
End Function

Function letemp (t As Variant) As Integer
'le a temperatura do controlador gefran
 uscita% = &H2FB
 form1.Comm1.Settings = "1200,e,8,1"
 form1.Comm1.PortOpen = True
 x55% = InG(uscita%)
 x55% = x55% Or 1
 nada% = OutG(uscita%, x55%)
 form1.Comm1.InBufferCount = 0
```

```
form1.Comm1.Output = Chr$(85)
form1.Comm1.PortOpen = False
'r
form1.Comm1.PortOpen = True
x55% = InG(uscita%)
x55% = x55% Or 1
x55% = x55% And &HEF
nada% = OutG(uscita%, x55%)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(170)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(170)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(0)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(0)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(2)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(0)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(111)
waits (t)
nada% = getbuf()
waits (t)
nada% = getbuf()
waits (t)
valor1 = getbuf()
waits (t)
valor2 = getbuf()
waits (t)
letemp = (valor1 * 256 + valor2)
form1.Comm1.PortOpen = False
End Function

Sub lini ()
'leitura dos paramentros do arquivo .ini
form1.VBini1.FileName = app.Path + "\rampa.ini"
form1.VBini1.Refresh
'subida
form1.VBini1.Section = "inclin"
form1.VBini1.Entry = "subida"
form1.VBini1.String = ""
```

```
form1.VBini1.Refresh
subida(1) = Val(form1.VBini1.String)
form1.VBini1.Refresh
'espera
form1.VBini1.Section = "inclin"
form1.VBini1.Entry = "espera"
form1.VBini1.String = ""
form1.VBini1.Refresh
espera = Val(form1.VBini1.String)
form1.VBini1.Refresh
'lgraph
form1.VBini1.Section = "inclin"
form1.VBini1.Entry = "lgraph"
form1.VBini1.String = ""
form1.VBini1.Refresh
lgraph = Val(form1.VBini1.String)
form1.VBini1.Refresh
'arquivo
form1.VBini1.Section = "arquivo"
form1.VBini1.Entry = "arquivo"
form1.VBini1.String = ""
form1.VBini1.Refresh
arquivo = form1.VBini1.String
form1.VBini1.Refresh
'inicio
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "inicio"
form1.VBini1.String = ""
form1.VBini1.Refresh
inicio = Val(form1.VBini1.String)
tinicial(1) = inicio
form1.VBini1.Refresh
'fim
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "fim"
form1.VBini1.String = ""
form1.VBini1.Refresh
fim = Val(form1.VBini1.String)
tfinal(1) = fim
form1.VBini1.Refresh
'gpm
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "taxa"
form1.VBini1.String = ""
form1.VBini1.Refresh
gpm(1) = Val(form1.VBini1.String)
form1.VBini1.Refresh
'waitcom
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "waitcom"
```

```
form1.VBini1.String = ""
form1.VBini1.Refresh
waitcom = Val(form1.VBini1.String)
form1.VBini1.Refresh

End Sub

Sub setpoint (v As Variant, t As Variant)
'transfere o valor de temperatura para o controlador
valor0 = v
valor1 = Int(valor0 / 256)
valor2 = valor0 - (valor1 * 256)
uscita% = &H2FB
form1.Comm1.Settings = "1200,e,8,1"
form1.Comm1.PortOpen = True
x55% = InG(uscita%)
x55% = x55% Or 1
nada% = OutG(uscita%, x55%)
form1.Comm1.InBufferCount = 0
form1.Comm1.Output = Chr$(85)
form1.Comm1.PortOpen = False
'r
form1.Comm1.PortOpen = True
form1.Comm1.InBufferCount = 0
x55% = InG(uscita%)
x55% = x55% Or 1
x55% = x55% And &HEF
nada% = OutG(uscita%, x55%)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(170)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(170)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(2)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(0)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(2)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(0)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(95)
waits (t)
```

```
nada% = getbuf()
form1.Comm1.Output = Chr$(valor1)
waits (t)
nada% = getbuf()
form1.Comm1.Output = Chr$(valor2)
waits (t)
nada% = getbuf()
form1.Comm1.PortOpen = False
End Sub

Sub sini ()
'grava parametros no arquivo .ini
form1.VBini1.FileName = app.Path + "\rampa.ini"
form1.VBini1.Refresh
'subida
form1.VBini1.Section = "inclin"
form1.VBini1.Entry = "subida"
form1.VBini1.String = subida(1)
form1.VBini1.Refresh
'espera
form1.VBini1.Section = "inclin"
form1.VBini1.Entry = "espera"
form1.VBini1.String = espera
form1.VBini1.Refresh
'lgraph
form1.VBini1.Section = "inclin"
form1.VBini1.Entry = "lgraph"
form1.VBini1.String = lgraph
form1.VBini1.Refresh
'arquivo
form1.VBini1.Section = "arquivo"
form1.VBini1.Entry = "arquivo"
form1.VBini1.String = arquivo
form1.VBini1.Refresh
'inicio
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "inicio"
form1.VBini1.String = inicio
form1.VBini1.Refresh
'fim
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "fim"
form1.VBini1.String = fim
form1.VBini1.Refresh
'taxa gpm
form1.VBini1.Section = "rampa"
form1.VBini1.Entry = "taxa"
form1.VBini1.String = tirav(gpm(1))
form1.VBini1.Refresh
End Sub
```

```
Function tirav (X As Variant) As String
'sbstitui a virgula (português) por ponto (inglês)
Dim s$, antesp$, depoisp$, ponto$
s$ = Format$(X, "0.000")
depois$ = Right$(s$, 3)
antes$ = Left$(s$, Len(s$) - 4)
ponto$ = antes$ + "." + depois$
tirav = ponto$
End Function

Sub waits (se As Single)
'rotina de atraso (espera tempo em segundos (se))
'e libera o windows (multitaking)
Dim inw, enw, difw
inw = Timer
llp:
enw = Timer
w% = DoEvents()
difw = enw - inw
w% = DoEvents()
If se > difw Then GoTo llp
End Sub
```

Arquivo FORM1.FRM

```
VERSION 2.00
Begin Form Form1
  BackColor      = &H00808080&
  BorderStyle    = 3  'Fixed Double
  Caption        = "TSC - Grupo de Polímeros"
  ClientHeight   = 3870
  ClientLeft     = 1515
  ClientTop      = 1980
  ClientWidth    = 8055
  Height         = 4560
  Icon           = FORM1.FRX:0000
  Left           = 1455
  LinkTopic      = "Form1"
  MaxButton      = 0  'False
  ScaleHeight    = 3870
  ScaleWidth     = 8055
  Top            = 1350
  Width          = 8175
  Begin MSComm Comm1
    CommPort     = 2
    InputLen     = 1
    Left         = 840
```

```
  ParityReplace  =  "0"
  Settings       =  "1200,e,8,1"
  Top            =  0
End
Begin PictureBox pict
  AutoRedraw     =  -1  'True
  Height         =  2655
  Index          =  1
  Left           =  4080
  ScaleHeight    =  2625
  ScaleWidth     =  3825
  TabIndex       =  8
  Top            =  120
  Width          =  3855
End
Begin CommonDialog CMDialog1
  Flags          =  1
  Left           =  0
  Top            =  0
End
Begin VBini VBini1
  Entry          =  ""
  FileName       =  ""
  Left           =  480
  Section        =  ""
  String         =  ""
  Top            =  0
End
Begin PictureBox pict
  AutoRedraw     =  -1  'True
  Height         =  2655
  Index          =  0
  Left           =  120
  ScaleHeight    =  2625
  ScaleWidth     =  3825
  TabIndex       =  1
  Top            =  120
  Width          =  3855
End
Begin SSFrame Frame3D1
  Font3D         =  0  'None
  ForeColor      =  &H00000000&
  Height         =  855
  Left           =  120
  TabIndex       =  0
  Top            =  2880
  Width          =  7815
  Begin CommandButton Command1
    Caption      =  "Mudar fundo de escala"
    Height       =  255
```

```
  Left        = 3360
  TabIndex    = 10
  Top         = 120
  Width       = 3255
End
Begin Gauge Gauge1
  Autosize    = -1 'True
  ForeColor   = &H00000000&
  Height      = 450
  InnerBottom = 5
  InnerLeft   = 5
  InnerRight  = 5
  InnerTop    = 5
  Left        = 6840
  Max         = 100
  NeedleWidth = 1
  Picture     = FORM1.FRX:0302
  Style       = 2 "Semi' Needle
  TabIndex    = 9
  Top         = 240
  Width       = 810
End
Begin Label Label12
  BackColor   = &H00C0C0C0&
  Caption     = "10"
  Height      = 255
  Left        = 2280
  TabIndex    = 19
  Top         = 480
  Width       = 255
End
Begin Label Label11
  BackColor   = &H00C0C0C0&
  BorderStyle = 1 'Fixed Single
  Height      = 255
  Left        = 2640
  TabIndex    = 18
  Top         = 360
  Width       = 375
End
Begin Label Label10
  BackColor   = &H00C0C0C0&
  BorderStyle = 1 'Fixed Single
  Height      = 255
  Left        = 1680
  TabIndex    = 17
  Top         = 480
  Width       = 495
End
Begin Label Label9
```

```
  BackColor     = &H00C0C0C0&
  Caption       = "I"
  Height        = 255
  Left          = 1440
  TabIndex      = 16
  Top           = 480
  Width         = 135
End
Begin Label Label8
  BackColor     = &H00C0C0C0&
  BorderStyle   = 1 'Fixed Single
  Height        = 255
  Left          = 6120
  TabIndex      = 11
  Top           = 360
  Width         = 375
End
Begin Label Label7
  BackColor     = &H00C0C0C0&
  Caption       = "E= 10"
  Height        = 255
  Left          = 5520
  TabIndex      = 15
  Top           = 480
  Width         = 615
End
Begin Label Label6
  BackColor     = &H00C0C0C0&
  BorderStyle   = 1 'Fixed Single
  Height        = 255
  Left          = 4800
  TabIndex      = 14
  Top           = 480
  Width         = 495
End
Begin Label Label5
  BackColor     = &H00C0C0C0&
  Caption       = "s="
  Height        = 255
  Left          = 4560
  TabIndex      = 13
  Top           = 480
  Width         = 255
End
Begin Label Label3
  BackColor     = &H00C0C0C0&
  Caption       = "Fundo Escala:"
  Height        = 255
  Left          = 3240
  TabIndex      = 12
```

```
  Top        = 480
  Width      = 1335
End
Begin Line Line3
  X1         = 6720
  X2         = 6720
  Y1         = 120
  Y2         = 840
End
Begin Label lblcor
  BackColor     = &H00C0C0C0&
  BorderStyle   = 1 'Fixed Single
  ForeColor     = &H00FF0000&
  Height        = 255
  Left          = 2040
  TabIndex      = 7
  Top           = 120
  Width         = 495
End
Begin Label lblTr
  BackColor     = &H00C0C0C0&
  BorderStyle   = 1 'Fixed Single
  Caption       = " "
  ForeColor     = &H000000FF&
  Height        = 255
  Left          = 600
  TabIndex      = 6
  Top           = 480
  Width         = 615
End
Begin Label lblTt
  BackColor     = &H00C0C0C0&
  BorderStyle   = 1 'Fixed Single
  Caption       = " "
  Height        = 255
  Left          = 600
  TabIndex      = 5
  Top           = 120
  Width         = 615
End
Begin Label Label4
  BackColor     = &H00C0C0C0&
  Caption       = "I - 0-3"
  ForeColor     = &H00FF0000&
  Height        = 255
  Left          = 1440
  TabIndex      = 4
  Top           = 120
  Width         = 615
End
```

```
  Begin Line Line2
    X1          = 3120
    X2          = 3120
    Y1          = 120
    Y2          = 840
  End
  Begin Line Line1
    X1          = 1320
    X2          = 1320
    Y1          = 120
    Y2          = 840
  End
  Begin Label Label2
    BackColor     = &H00C0C0C0&
    Caption       = "Tp T"
    Height        = 255
    Left          = 120
    TabIndex      = 3
    Top           = 120
    Width         = 495
  End
  Begin Label Label1
    BackColor     = &H00C0C0C0&
    Caption       = "Tp R"
    ForeColor     = &H000000FF&
    Height        = 255
    Left          = 120
    TabIndex      = 2
    Top           = 480
    Width         = 495
  End
End
Begin Line Line12
  BorderColor     = &H0000FFFF&
  X1          = 4080
  X2          = 3960
  Y1          = 720
  Y2          = 720
End
Begin Line Line11
  BorderColor     = &H0000FFFF&
  X1          = 4080
  X2          = 3960
  Y1          = 2160
  Y2          = 2160
End
Begin Line Line10
  BorderColor     = &H0000FFFF&
  X1          = 3960
  X2          = 4080
```

FSC — SERVIÇO DE BIBLIOTECA E

```
  Y1          = 1440
  Y2          = 1440
End
Begin Line Line9
  BorderColor   = &H0000FFFF&
  X1          = 5040
  X2          = 5040
  Y1          = 2880
  Y2          = 2760
End
Begin Line Line8
  BorderColor   = &H0000FFFF&
  X1          = 6000
  X2          = 6000
  Y1          = 2880
  Y2          = 2760
End
Begin Line Line7
  BorderColor   = &H0000FFFF&
  X1          = 6960
  X2          = 6960
  Y1          = 2760
  Y2          = 2880
End
Begin Line Line6
  BorderColor   = &H0000FFFF&
  X1          = 1080
  X2          = 1080
  Y1          = 2880
  Y2          = 2760
End
Begin Line Line5
  BorderColor   = &H0000FFFF&
  X1          = 2040
  X2          = 2040
  Y1          = 2880
  Y2          = 2760
End
Begin Line Line4
  BorderColor   = &H0000FFFF&
  X1          = 3000
  X2          = 3000
  Y1          = 2760
  Y2          = 2880
End
Begin Menu roda
  Caption       = "Programa"
  Begin Menu tsc
    Caption       = "TSC"
  End
```

```
  Begin Menu manual
    Caption      = "Manual"
  End
  Begin Menu nada2
    Caption      = "------------"
    Enabled      = 0  'False
  End
  Begin Menu parar
    Caption      = "parar"
  End
End
Begin Menu grava
  Caption      = "Arquivo"
End
Begin Menu Setrampa
  Caption      = "Config."
  Begin Menu configura
    Caption      = "Config"
  End
  Begin Menu Apaga
    Caption      = "Apaga Gráfico"
  End
End
Begin Menu wind
  Caption      = "Janela"
  Begin Menu win1
    Caption      = "1 janela"
  End
  Begin Menu win2
    Caption      = "2 janelas"
  End
  Begin Menu win4
    Caption      = "4 janelas"
    Checked      = -1  'True
  End
  Begin Menu nada1
    Caption      = "---------------"
    Enabled      = 0  'False
  End
End
Begin Menu ajuda
  Caption      = "Ajuda"
End
Begin Menu sobre
  Caption      = "Sobre"
End
End
Dim vsp(10000), v(10000), errtime(10000), intert(10000) As Single, corrente(10000) As Single, escor(10000) As Integer
Dim pontos, tini
```

```
Sub ajuda_Click ()
 Dim errcode
 Dim f As Integer
 Dim volts As Double
 pict(1).Scale (0, 5)-(500, -5)
lllp:
 For f = 0 To 500
  Call lecor(f, 1)
  waits (.2)
 Next f
 pict(1).Cls
 GoTo lllp
End Sub

Sub apaga_click ()
 For f = 0 To 10000
  v(f) = 0
  vsp(f) = 0
 Next f
 For f = 0 To 1
  pict(f).Cls
 Next f
 gauge1.Value = 0
 roda.Enabled = True
End Sub

Sub calcparm ()
 Dim tmpg, tmpgs, tmpu, tmpac, f As Integer
 For z = 1 To 10
  subida(z) = 1
 Next z
 setpic
 tmpgs = caltime()
 For f = 1 To numinter
 If flagrampa(f) = 0 Then
comcalcparm:
  tmpg = tmpgs / subida(f)
  tmpu = (60 / gpm(f)) / 10
  tmpac = (tmpu - tmpg)
  If tmpac < .5 Then
    subida(f) = subida(f) + 1
    GoTo comcalcparm
  End If
  tmpwait(f) = (tmpu * subida(f))
 Else
  tmpwait(f) = 0
 End If
 Next f
End Sub
```

```
Function caltime () As Variant
 Static tt(9)
 Dim ttt, t1, t2
 Dim xx As Integer
 Dim f As Integer
 setpic 'configura a picture e o gauge
 xx = 0
 'loop da rampa
 For f = 401 To 409
  t1 = Timer
   Call setptes(tinicial(1), waitcom) 'rotina para teste
   'Call setpoint(tinicial(1), waitcom)  'setpoint do controlador
   Call lecortes(xx, 1)'rotina para teste
   'Call lecor(xx, 1)
   xx = wlgtes(f, xx) 'rotina para teste
   'xx = wlg(f, xx)
  gauge1.Value = xx
  t2 = Timer - t1
  tt(f - 400) = t2
  If paraja = 1 Then GoTo fimcaltime
 Next f
 For f = 405 To 409
  ttt = ttt + tt(f - 400)
 Next f
 caltime = ttt / 5
 gauge1.Value = 0
fimcaltime:
End Function

Sub Command1_Click ()
 eletrotrava = 1
 form4.Show
End Sub

Sub configura_Click ()
 form2.Show
End Sub

Sub Form_Load ()
 Dim status
 Dim volts As Double
 numinter = 1
 status = ai_vread(1, 0, 2, volts)
 Call lini
End Sub

Sub grava_Click ()
  cmdialog1.DialogTitle = "Grava medidas"
  cmdialog1.Filter = "Data Files (*.DAT)|*.log|All Files (*.*)|*.*"
```

```
cmdialog1.Filename = "aquis.dat"
cmdialog1.Action = 1 '1

If cmdialog1.Flags = 1 Then Exit Sub
temp$ = cmdialog1.Filename

'--- abrindo e salvando arquivos rampa
Close #1
Open temp$ For Output As 1
For f = 0 To pontos
 Print #1, Str$(errtime(f) - errtime(0)), Str$(v(f))
Next f
Close #1
ext$ = Right$(temp$, 4)
temp$ = Left$(temp$, Len(temp$) - 4)
'salva temperatura real
temp1$ = temp$ + "t" + ext$
Open temp1$ For Output As 1
For f = 0 To pontos
 Print #1, Str$(errtime(f) - errtime(0)), Str$(vsp(f))
Next f
Close #1
'salva corrente x tempo
temp1$ = temp$ + "cs" + ext$
Open temp1$ For Output As 1
For f = 0 To pontos
 Print #1, Str$(errtime(f) - errtime(0)), Format$(corrente(f), "#####.#####"); "e"; Right$(Str$(escor(f)), (Len(Str$(escor(f))) - 1))
Next f
Close #1
'salva corrente x temperatura
temp1$ = temp$ + "ct" + ext$
Open temp1$ For Output As 1
For f = 0 To pontos
 Print #1, Str$(vsp(f)), Format$(corrente(f), "#####.#####"); "e"; Right$(Str$(escor(f)), (Len(Str$(escor(f))) - 1))
Next f
Close #1
'teste ----------------------------
Open "c:\time.dat" For Output As 1
For f = 0 To pontos
 Print #1, Str$(f), Str$(errtime(f))
Next f
Close #1
End Sub

Sub lecor (posicao As Integer, gain As Integer)
Dim errcode
Dim volts As Double
Dim corr As Single
errcode = ai_vread(1, 0, gain, volts)
lblcor.Caption = Format$(volts, "00.00")
```

```
volts = Abs(volts)
If eletrotrava = 0 Then
  oldvolt = volts
  If (volts > 2.7) And excedeu = 0 Then
    excedeu = 1
    form5.Show
  End If
Else
  volts = oldvolt
End If
corr = volts * fconver
corrente(posicao) = corr
escor(posicao) = esceletr
label10.Caption = Format$(corr, "####.####")
label11.Caption = Str$(esceletr)
pict(1).PSet (posicao, volts), RGB(0, 0, 255)
End Sub

Sub lecortes (posicao As Integer, gain As Integer)
 Dim errcode
 Dim volts As Double
 Dim corr As Single
 'errcode = ai_vread(1, 0, gain, volts)
 volts = ruido(1) / 10
 lblcor.Caption = Format$(volts, "00.00")
 volts = Abs(volts)
 If eletrotrava = 0 Then
   oldvolt = volts
   If (volts > 2.7) And excedeu = 0 Then
     excedeu = 1
     form5.Show
   End If
 Else
   volts = oldvolt
 End If
 corr = volts * fconver
 corrente(posicao) = corr
 escor(posicao) = esceletr
 label10.Caption = Format$(corr, "####.####")
 label11.Caption = Str$(esceletr)
 pict(1).PSet (posicao, volts), RGB(0, 0, 255)
End Sub

Function lopgraph (f As Integer, X As Integer) As Integer
  'loop leitura do termopar
  For I = 1 To Igraph
   y1 = f
   y2 = letemp(.2) 'leitura do termopar
   'y2 = ruido(y1)
   pict(0).PSet (X, y1), RGB(0, 0, 0) 'plota teorica
```

```
 pict(0).PSet (X, y2), RGB(255, 0, 0) 'plota pratica
 lbltt.Caption = Str$(y1 / 10) 'escreve teorica
 lbltr.Caption = Str$(y2 / 10) 'escreve pratica
 vsp(X) = y1 / 10 'calcula teorica
 v(X) = y2 / 10   'calcula pratica
 errtime(X) = Timer - tini  'calcula tempo da rotina
 X = X + 1
 Next I
 lopgraph = X
End Function

Sub manual_Click ()
 form3.Show
End Sub

Sub parar_Click ()
 paraja = 1
End Sub

Sub principal ()
 Dim xx As Integer, f As Integer, tin1, tin2, intervalos As Integer
 setpic 'configura as pictures e o gauge
 xx = 0
 tini = Timer
 For intervalos = 1 To numinter
 'loop da rampa
  If flagrampa(intervalos) = 0 Then
   'rampa
   For f = tinicial(intervalos) To tfinal(intervalos) Step subida(intervalos)
    tin1 = Timer
    Call setptes(f, waitcom)'rotina de teste
    'Call setpoint(f, waitcom)  'setpoint do controlador
    Call lecortes(xx, 1)'rotina de teste
    'Call lecor(xx, 1) 'le corrente
    xx = wlgtes(f, xx) 'rotina de teste
    'xx = wlg(f, xx) 'le temperatura
    gauge1.Value = xx
    tin2 = Timer - tin1
    waits (tmpwait(1) - tin2)
    If paraja = 1 Then GoTo fimprinc
   Next f
   xx = xx - 1 'decrementa contador
  Else
  'constante
 consta:
   tincons = Timer
   'Call lecortes(xx, 1) 'teste
   Call lecor(xx, 1)
   xx = wlgtes(tinicial(intervalos), xx) 'teste
   'xx = wlg(tinicial(intervalos), xx) 'leitura de temperatura
```

```
  gauge1.Value = xx
constaloop:
  ww = DoEvents()
  tincons2 = Timer
  If paraja = 1 Then GoTo fimprinc
  If ((tincons2 - tincons) < entrepontos(intervalos)) Then GoTo constaloop
  If ((tincons2 - tini) < durainter(intervalos)) Then GoTo consta
 End If
 'acaba o loop principal
 Next intervalos
 'Call setpoint(inicio, waitcom)
fimprinc:
 pontos = xx  'calcula numero de pontos
 End Sub

Function ruido (valor As Variant)
 Randomize
 erro = 30
 desvio = (erro * Rnd) + 1
 ruido = Int(valor - (erro / 2) + desvio)
End Function

Sub setpic ()
 'scala da picture
 ppp = 0
 For f = 1 To numinter
  If flagrampa(f) = 0 Then
   ppp = ppp + ((tfinal(f) - tinicial(f)) / subida(f))
  Else
   ppp = ppp + (durainter(f) / entrepontos(f))
  End If
 Next f
 pict(0).Scale (0, tfinal(numinter) + 50)-(ppp, tinicial(1) - 50)
 'fim escala da picture
 pict(1).Scale (0, 4)-(ppp, 0)
 '
 'define a barra de status
 gauge1.Max = ppp
 gauge1.Min = 0
 'fim barra de status
 End Sub

Sub setptes (f As Integer, waitcom As Single)
 'nao faz nada além de esperar por waitcom + 1 segundos
 waits (waitcom)
 waits (1)
End Sub

Sub sobre_Click ()
 form6.Show
```

```
End Sub

Function tiravirg (X As Variant) As String
'Esta rotina substitui a virgula (português) para
'o ponto(inglês)
 Dim s$, antesp$, depoisp$, ponto$
 s$ = Format$(X, "0.000")
 depois$ = Right$(s$, 3)
 antes$ = Left$(s$, Len(s$) - 4)
 ponto$ = antes$ + "." + depois$
 tiravirg = ponto$
End Function

Sub tsc_Click ()
 tsc.Enabled = False
 manual.Enabled = False
 'outros menus
 grava.Enabled = False
 setrampa.Enabled = False
 wind.Enabled = False
 ajuda.Enabled = False
 sobre.Enabled = False
 'fim outros
 paraja = 0
 'verifica se existe escala
 If seletr = 0 Or esceletr = 0 Then
   msg$ = "Antes de iniciar as medidas," + Chr$(13)
   msg$ = msg$ + "definir a escala do eletrômetro" + Chr$(13)
   msg$ = msg$ + "no menu de configuração"
   MsgBox msg$, 48, "escala eletrômetro"
   GoTo termprinc
 End If
 label6.Caption = Str$(seletr)
 label8.Caption = Str$(esceletr)
 Call calcparm
 apaga_click
 principal
 'termina a execução
termprinc:
 tsc.Enabled = True
 manual.Enabled = True
 'outros menus
 grava.Enabled = True
 setrampa.Enabled = True
 wind.Enabled = True
 ajuda.Enabled = True
 sobre.Enabled = True
 'fim outros

 label6.Caption = ""
```

```
label8.Caption = ""
End Sub

Sub win1_Click ()
 form5.Show
End Sub

Function wlg (f As Integer, X As Integer) As Integer
 'leitura do termopar e plota tela sem loopgraph
 y1 = f
 y2 = letemp(waitcom) 'leitura do termopar
 'y2 = ruido(y1)
 pict(0).PSet (X, y1), RGB(0, 0, 0) 'plota teorica
 pict(0).PSet (X, y2), RGB(255, 0, 0) 'plota pratica
 lbltt.Caption = Str$(y1 / 10) 'escreve teorica
 lbltr.Caption = Str$(y2 / 10) 'escreve pratica
 vsp(X) = y1 / 10 'calcula teorica
 v(X) = y2 / 10   'calcula pratica
 errtime(X) = Timer - tini  'calcula tempo da rotina
 X = X + 1
 wlg = X
End Function

Function wlgtes (f As Integer, X As Integer) As Integer
 'leitura do termopar e plota tela sem loopgraph
 y1 = f
 'y2 = letemp(waitcom) 'leitura do termopar
 y2 = ruido(y1) 'teste (gera ruido)
 pict(0).PSet (X, y1), RGB(0, 0, 0) 'plota teorica
 pict(0).PSet (X, y2), RGB(255, 0, 0) 'plota pratica
 lbltt.Caption = Str$(y1 / 10) 'escreve teorica
 lbltr.Caption = Str$(y2 / 10) 'escreve pratica
 vsp(X) = y1 / 10 'calcula teorica
 v(X) = y2 / 10   'calcula pratica
 errtime(X) = Timer - tini  'calcula tempo da rotina
 X = X + 1
 wlgtes = X
End Function
```

Arquivo FORM2.FRM:

```
VERSION 2.00
Begin Form Form2
  BackColor      =  &H00FFFF00&
  BorderStyle    =  1  'Fixed Single
  Caption        =  "Setup"
  ClientHeight   =  5115
  ClientLeft     =  3330
  ClientTop      =  1425
  ClientWidth    =  3390
  Height         =  5520
```

```
Left        = 3270
LinkTopic   = "Form2"
MaxButton   = 0 'False
MinButton   = 0 'False
ScaleHeight = 5115
ScaleWidth  = 3390
Top         = 1080
Width       = 3510
Begin TextBox Text9
  Height    = 285
  Left      = 2160
  TabIndex  = 30
  Top       = 2760
  Width     = 495
End
Begin CommandButton Command5
  Caption   = "OK"
  Height    = 615
  Left      = 2640
  TabIndex  = 28
  Top       = 1320
  Width     = 615
End
Begin CommandButton Command4
  Caption   = "Atualizar novo num. de intervalos"
  Height    = 255
  Left      = 120
  TabIndex  = 27
  Top       = 480
  Width     = 3135
End
Begin TextBox Text8
  Height    = 285
  Left      = 2040
  TabIndex  = 25
  Text      = " "
  Top       = 2400
  Width     = 495
End
Begin OptionButton Option2
  BackColor = &H00FFFF00&
  Caption   = "Constante"
  Height    = 255
  Left      = 1920
  TabIndex  = 23
  Top       = 2040
  Width     = 1215
End
Begin OptionButton Option1
  BackColor = &H00FFFF00&
```

```
  Caption      = "Rampa"
  Height       = 255
  Left         = 240
  TabIndex     = 22
  Top          = 2040
  Width        = 975
End
Begin HScrollBar HScroll1
  Height       = 255
  Left         = 1560
  TabIndex     = 21
  Top          = 960
  Width        = 1695
End
Begin TextBox Text3
  BackColor    = &H0000FFFF&
  Height       = 285
  Left         = 1560
  TabIndex     = 17
  Text         = "1"
  Top          = 120
  Width        = 375
End
Begin TextBox Text2
  Height       = 285
  Left         = 2400
  TabIndex     = 15
  Text         = ""
  Top          = 3600
  Width        = 615
End
Begin TextBox Text1
  Height       = 285
  Left         = 960
  TabIndex     = 13
  Text         = ""
  Top          = 3840
  Width        = 495
End
Begin TextBox Text7
  Height       = 285
  Left         = 1920
  TabIndex     = 1
  Text         = ""
  Top          = 1680
  Width        = 615
End
Begin TextBox Text6
  Height       = 285
  Left         = 1800
```

```
  TabIndex      = 2
  Text          = " "
  Top           = 1320
  Width         = 615
End
Begin TextBox Text5
  Height        = 285
  Left          = 720
  TabIndex      = 8
  Text          = " "
  Top           = 1320
  Width         = 615
End
Begin TextBox Text4
  Height        = 285
  Left          = 360
  TabIndex      = 6
  Text          = " "
  Top           = 4320
  Width         = 2655
End
Begin CommandButton Command3
  Caption       = "cancel"
  Height        = 255
  Left          = 2280
  TabIndex      = 0
  Top           = 4800
  Width         = 975
End
Begin CommandButton Command2
  Caption       = "Do It"
  Height        = 255
  Left          = 1200
  TabIndex      = 5
  Top           = 4800
  Width         = 975
End
Begin CommandButton Command1
  Caption       = "Save"
  Height        = 255
  Left          = 120
  TabIndex      = 4
  Top           = 4800
  Width         = 975
End
Begin Label Label15
  BackColor     = &H00FFFF00&
  Caption       = "min."
  Height        = 255
  Left          = 2760
```

```
  TabIndex      = 31
  Top           = 2760
  Width         = 495
End
Begin Label Label14
  BackColor     = &H00FFFF00&
  Caption       = "Intervalo entre pontos:"
  Height        = 255
  Left          = 120
  TabIndex      = 29
  Top           = 2760
  Width         = 2055
End
Begin Label Label13
  BackColor     = &H00FFFF00&
  Caption       = "min."
  Height        = 255
  Left          = 2640
  TabIndex      = 26
  Top           = 2400
  Width         = 615
End
Begin Label Label12
  BackColor     = &H00FFFF00&
  Caption       = "Duração do Intervalo:"
  Height        = 255
  Left          = 120
  TabIndex      = 24
  Top           = 2400
  Width         = 1935
End
Begin Label Label11
  BackColor     = &H00FFFF00&
  BorderStyle   = 1 'Fixed Single
  Height        = 255
  Left          = 1080
  TabIndex      = 20
  Top           = 960
  Width         = 375
End
Begin Label Label10
  BackColor     = &H00FFFF00&
  Caption       = "intervalo :"
  Height        = 255
  Left          = 120
  TabIndex      = 19
  Top           = 960
  Width         = 975
End
Begin Label Label9
```

```
  BackColor      = &H00FFFF00&
  Caption        = "no máximo (10)"
  Height         = 255
  Left           = 2040
  TabIndex       = 18
  Top            = 120
  Width          = 1335
End
Begin Line Line4
  X1             = 0
  X2             = 3360
  Y1             = 840
  Y2             = 840
End
Begin Label Label8
  BackColor      = &H00FFFF00&
  Caption        = "N. de intervalos:"
  Height         = 255
  Left           = 120
  TabIndex       = 16
  Top            = 120
  Width          = 1455
End
Begin Line Line3
  X1             = 0
  X2             = 3360
  Y1             = 4680
  Y2             = 4680
End
Begin Line Line2
  X1             = 0
  X2             = 3360
  Y1             = 4200
  Y2             = 4200
End
Begin Label Label7
  BackColor      = &H00FFFF00&
  Caption        = "E=10"
  Height         = 255
  Left           = 1920
  TabIndex       = 14
  Top            = 3840
  Width          = 495
End
Begin Label Label3
  BackColor      = &H00FFFF00&
  Caption        = "s="
  Height         = 255
  Left           = 720
  TabIndex       = 12
```

```
  Top        = 3840
  Width      = 255
End
Begin Label Label2
  BackColor    = &H00FFFF00&
  Caption      = "FE:"
  Height       = 255
  Left         = 120
  TabIndex     = 11
  Top          = 3840
  Width        = 375
End
Begin Label Label1
  BackColor    = &H00FFFF00&
  Caption      = "Fundo de Escala do Eletrômetro"
  Height       = 255
  Left         = 240
  TabIndex     = 10
  Top          = 3240
  Width        = 2895
End
Begin Line Line1
  X1           = 0
  X2           = 3360
  Y1           = 3120
  Y2           = 3120
End
Begin Label Label6
  BackColor    = &H00FFFF00&
  Caption      = "taxa em graus / min:"
  Height       = 255
  Left         = 120
  TabIndex     = 3
  Top          = 1680
  Width        = 1935
End
Begin Label Label5
  BackColor    = &H00FFFF00&
  Caption      = "fim:"
  Height       = 255
  Left         = 1440
  TabIndex     = 9
  Top          = 1320
  Width        = 375
End
Begin Label Label4
  BackColor    = &H00FFFF00&
  Caption      = "Inicio:"
  Height       = 255
  Left         = 120
```

```
    TabIndex      = 7
    Top           = 1320
    Width         = 615
  End
End

Sub atualiza (indice As Integer)
 text5.Text = tiravg(tinicial(indice) / 10)
 text6.Text = tiravg(tfinal(indice) / 10)
 text7.Text = Str$(gpm(indice))
 option1.Value = False
 option2.Value = False
  If flagrampa(indice) = 1 Then
    option2.Value = True
    text8.Text = Str$(durainter(indice) / 60)
    text8.Enabled = True
    text9.Text = Str$(entrepontos(indice) / 60)
    text9.Enabled = True
    label12.ForeColor = &H80000008
    label13.ForeColor = &H80000008
    label14.ForeColor = &H80000008
    label15.ForeColor = &H80000008

  Else
    option1.Value = True
    text8.Text = ""
    text8.Enabled = False
    text9.Text = ""
    text9.Enabled = False
    label12.ForeColor = &HFFFF00
    label13.ForeColor = &HFFFF00
    label14.ForeColor = &HFFFF00
    label15.ForeColor = &HFFFF00
  End If
 End Sub

 Sub Command1_Click ()
  arquivo = text4.Text
  inicio = Val(text5.Text) * 10
  fim = Val(text6.Text) * 10
  gpm(1) = Val(text7.Text)
  sini
  'calcula fundo de escala eletrometro
  seletr = Val(text1.Text)
  esceletr = Val(text2.Text)
  fconver = calconfe(seletr)
  Unload form2
 End Sub

 Sub Command2_Click ()
```

```
'define parâmetros dos intervalos
If numinter = 1 Then 'para 1 intervalo
 arquivo = text4.Text
 inicio = Val(text5.Text) * 10
 tinicial(1) = inicio
 fim = Val(text6.Text) * 10
 tfinal(1) = fim
 gpm(1) = Val(text7.Text)
 'define flagrampa
 If option2.Value = -1 Then
  flagrampa(1) = 1
  entrepontos(1) = Val(text9.Text) * 60
  durainter(1) = Val(text8.Text) * 60
 End If
 If option1.Value = -1 Then flagrampa(1) = 0
End If
'calcula fundo de escala eletrometro
seletr = Val(text1.Text)
esceletr = Val(text2.Text)
fconver = calconfe(seletr)
form1.Label6.Caption = Str$(fconver)
Unload form2
End Sub

Sub Command3_Click ()
 Unload form2
End Sub

Sub Command4_Click ()
 Dim f As Integer
 numinter = Val(text3.Text)
 If numinter > 10 Then numinter = 10
 If numinter < 1 Then numinter = 1
 For f = 1 To 10
  flagrampa(f) = 0
  tinicial(f) = 0
  tfinal(f) = 0
  gpm(f) = 0
  durainter(f) = 0
  entrepontos(f) = 0
 Next f
 If numinter = 1 Then
   hscroll1.Enabled = False
   command1.Enabled = True
   command3.Enabled = True
   command5.Enabled = False

 Else
   hscroll1.Enabled = True
   hscroll1.Min = 1
```

```
  hscroll1.Max = numinter
  command1.Enabled = False
  command3.Enabled = False
  command5.Enabled = True
 End If
End Sub

Sub Command5_Click ()
 indice = hscroll1.Value
 tinicial(indice) = Val(text5.Text) * 10
 tfinal(indice) = Val(text6.Text) * 10
 gpm(indice) = Val(text7.Text)
 'define flagrampa
 If option2.Value = True Then
   flagrampa(indice) = 1
   durainter(indice) = Val(text8.Text) * 60
   entrepontos(indice) = Val(text9.Text) * 60
 End If
 If option1.Value = True Then
   flagrampa(indice) = 0
   durainter(indice) = 0
   entrepontos(indice) = 0
 End If
End Sub

Sub Form_Load ()
 text4.Text = arquivo
 text5.Text = tiravg(inicio / 10)
 text6.Text = tiravg(fim / 10)
 text7.Text = Str$(gpm(1))
 option1.Value = False
 option2.Value = False
  If flagrampa(1) = 1 Then
    option2.Value = True
    text8.Text = durainter(1)
    text8.Enabled = True
    text9.Text = entrepontos(1)
    text9.Enabled = True
    label12.ForeColor = &H80000008
    label13.ForeColor = &H80000008
    label14.ForeColor = &H80000008
    label15.ForeColor = &H80000008

  Else
    option1.Value = True
    text8.Text = ""
    text8.Enabled = False
    text9.Text = ""
    text9.Enabled = False
    label12.ForeColor = &HFFFF00
```

```
  label13.ForeColor = &HFFFF00
  label14.ForeColor = &HFFFF00
  label15.ForeColor = &HFFFF00
 End If
command5.Enabled = False
hscroll1.Enabled = False
label11.Caption = "1"
If numinter = 1 Then
  hscroll1.Enabled = False
  command5.Value = False
 End If
End Sub

Sub HScroll1_Change ()
 label11.Caption = Str$(hscroll1.Value)
 atualiza (hscroll1.Value)
End Sub

Sub Option1_Click ()
 text8.Text = ""
 text8.Enabled = False
 text9.Text = ""
 text9.Enabled = False
 label12.ForeColor = &HFFFF00
 label13.ForeColor = &HFFFF00
 label14.ForeColor = &HFFFF00
 label15.ForeColor = &HFFFF00
 End Sub

 Sub Option2_Click ()
  text8.Enabled = True
  text9.Enabled = True
  label12.ForeColor = &H80000008
  label13.ForeColor = &H80000008
  label14.ForeColor = &H80000008
  label15.ForeColor = &H80000008
 End Sub

 Function tiravg (X As Variant) As String
 'Esta rotina substitui a virgula (português) para
 'o ponto(inglês)
  Dim s$, antesp$, depoisp$, ponto$
  s$ = Format$(X, "0.0")
  depois$ = Right$(s$, 1)
  antes$ = Left$(s$, Len(s$) - 2)
  ponto$ = antes$ + "." + depois$
  tiravg = ponto$
 End Function
```

Arquivo FORM3.FRM:

```
VERSION 2.00
Begin Form Form3
  BackColor      = &H00808000&
  BorderStyle    = 1 'Fixed Single
  Caption        = "Ajustar Temperatura da Estufa"
  ClientHeight   = 1350
  ClientLeft     = 3600
  ClientTop      = 7035
  ClientWidth    = 4470
  Height         = 1755
  Left           = 3540
  LinkTopic      = "Form3"
  ScaleHeight    = 1350
  ScaleWidth     = 4470
  Top            = 6690
  Width          = 4590
  Begin CommandButton Command2
    Caption      = "Cancel"
    Height       = 495
    Left         = 2280
    TabIndex     = 4
    Top          = 720
    Width        = 2055
  End
  Begin CommandButton Command1
    Caption      = "OK"
    Height       = 495
    Left         = 120
    TabIndex     = 3
    Top          = 720
    Width        = 1935
  End
  Begin TextBox Text1
    Height       = 285
    Left         = 2400
    TabIndex     = 1
    Text         = " "
    Top          = 240
    Width        = 735
  End
  Begin Label Label2
    BackColor    = &H00808000&
    Caption      = "Celsius"
    Height       = 255
    Left         = 3480
    TabIndex     = 2
    Top          = 240
    Width        = 735
  End
  Begin Label Label1
```

```
    BackColor      = &H00808000&
    Caption        = "Temperatura da estufa :"
    Height         = 255
    Left           = 120
    TabIndex       = 0
    Top            = 240
    Width          = 2175
  End
End
Sub Command1_Click ()
 Call setpoint((Val(text1.Text) * 10), waitcom)
 Unload form3
End Sub

Sub Command2_Click ()
 Unload form3
End Sub
```

Arquivo FORM4.FRM:

```
VERSION 2.00
Begin Form Form4
  BackColor      = &H00808000&
  BorderStyle    = 0 'None
  Caption        = "Muda fundo de escala"
  ClientHeight   = 2520
  ClientLeft     = 4455
  ClientTop      = 5955
  ClientWidth    = 4455
  Height         = 2925
  Left           = 4395
  LinkTopic      = "Form4"
  MaxButton      = 0 'False
  MinButton      = 0 'False
  ScaleHeight    = 2520
  ScaleWidth     = 4455
  Top            = 5610
  Width          = 4575
  Begin CommandButton Command1
    Caption      = "Alterar o fundo de escala"
    Height       = 495
    Left         = 360
    TabIndex     = 8
    Top          = 1920
    Width        = 3855
  End
  Begin TextBox Text1
    Height       = 285
    Left         = 2400
    TabIndex     = 4
    Text         = " "
```

```
  Top          = 1320
  Width        = 495
End
Begin TextBox Text2
  Height       = 285
  Left         = 3600
  TabIndex     = 3
  Text         = " "
  Top          = 960
  Width        = 615
End
Begin Label Label5
  BackColor    = &H00808000&
  Caption      = "Fundo de Escala :"
  Height       = 255
  Left         = 240
  TabIndex     = 7
  Top          = 1320
  Width        = 1695
End
Begin Label Label4
  BackColor    = &H00808000&
  Caption      = "s="
  Height       = 255
  Left         = 2040
  TabIndex     = 6
  Top          = 1320
  Width        = 255
End
Begin Label Label7
  BackColor    = &H00808000&
  Caption      = "E=10"
  Height       = 255
  Left         = 3120
  TabIndex     = 5
  Top          = 1320
  Width        = 495
End
Begin Label Label3
  Alignment    = 2 'Center
  BackColor    = &H00808000&
  Caption      = "e pressione o botão "
  Height       = 255
  Left         = 120
  TabIndex     = 2
  Top          = 600
  Width        = 4215
End
Begin Label Label2
  Alignment    = 2 'Center
```

```
    BackColor      = &H00808000&
    Caption        = "eletrômetro (físico) e então insira os parâmetros "
    Height         = 255
    Left           = 120
    TabIndex       = 1
    Top            = 360
    Width          = 4215
  End
  Begin Label Label1
    Alignment      = 2 'Center
    BackColor      = &H00808000&
    Caption        = "O leitor está travado, altere o fundo de escala do"
    Height         = 255
    Left           = 120
    TabIndex       = 0
    Top            = 120
    Width          = 4215
  End
End
Sub Command1_Click ()
 'calcula fundo de escala eletrometro
 seletr = Val(text1.Text)
 esceletr = Val(text2.Text)
 fconver = calconfe(seletr)
 form1.Label6.Caption = Str$(seletr)
 form1.Label8.Caption = Str$(esceletr)
 eletrotrava = 0
 excedeu = 0
 Unload form4
End Sub
```

Arquivo FORM5.FRM:

```
VERSION 2.00
Begin Form Form5
  BackColor      = &H000000FF&
  BorderStyle    = 0 'None
  Caption        = "Form5"
  ClientHeight   = 4230
  ClientLeft     = 1095
  ClientTop      = 1485
  ClientWidth    = 6630
  Height         = 4635
  Left           = 1035
  LinkTopic      = "Form5"
  ScaleHeight    = 4230
  ScaleWidth     = 6630
  Top            = 1140
  Width          = 6750
  Begin Timer Timer1
    Interval     = 1000
```

```
  Left        = 0
  Top         = 0
End
Begin CommandButton Command1
  Caption     = "OK"
  FontBold    = -1 'True
  FontItalic  = 0 'False
  FontName    = "MS Sans Serif"
  FontSize    = 24
  FontStrikethru = 0 'False
  FontUnderline = 0 'False
  Height      = 855
  Left        = 480
  TabIndex    = 1
  Top         = 3000
  Width       = 5655
End
Begin Label Label1
  Alignment   = 2 'Center
  BackColor   = &H0000FFFF&
  Caption     = "O fundo de escala do eletrômetro está sendo excedido, alterar com urgência"
  FontBold    = -1 'True
  FontItalic  = 0 'False
  FontName    = "MS Sans Serif"
  FontSize    = 24
  FontStrikethru = 0 'False
  FontUnderline = 0 'False
  ForeColor   = &H000000FF&
  Height      = 2295
  Left        = 480
  TabIndex    = 0
  Top         = 360
  Width       = 5655
End
End
Dim okpress, c

Sub Command1_Click ()
okpress = 1
End Sub

Sub Timer1_Timer ()
Beep
c = c + 1
If c = 1 Or c = 3 Then form5.BackColor = &HFF&
If c = 2 Or c = 4 Then form5.BackColor = &HFFFF&
If c = 4 Then c = 0
If okpress = 1 Then
  timer1.Enabled = False
  eletrotrava = 1
```

```
  form4.Show
  Unload form5
 End If
End Sub
```

Arquivo FORM6.FRM:

```
VERSION 2.00
Begin Form Form6
  BorderStyle    = 3 'Fixed Double
  Caption        = "About OMB#SMT"
  ClientHeight   = 4095
  ClientLeft     = 3000
  ClientTop      = 3960
  ClientWidth    = 5685
  Height         = 4500
  Left           = 2940
  LinkTopic      = "Form6"
  MaxButton      = 0 'False
  MinButton      = 0 'False
  Picture        = FORM6.FRX:0000
  ScaleHeight    = 4095
  ScaleWidth     = 5685
  Top            = 3615
  Width          = 5805
  Begin Timer Timer1
    Interval     = 100
    Left         = 5280
    Top          = 0
  End
  Begin PictureBox Picture1
    BackColor    = &H00FFFFFF&
    Height       = 1815
    Index        = 8
    Left         = 0
    Picture      = FORM6.FRX:D03A
    ScaleHeight  = 1785
    ScaleWidth   = 2145
    TabIndex     = 9
    Top          = 0
    Width        = 2175
  End
  Begin PictureBox Picture1
    BackColor    = &H00FFFFFF&
    Height       = 1815
    Index        = 7
    Left         = 0
    Picture      = FORM6.FRX:11D7C
    ScaleHeight  = 1785
    ScaleWidth   = 2145
```

```
  TabIndex      = 8
  Top           = 0
  Width         = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 6
  Left          = 0
  Picture       = FORM6.FRX:16ABE
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 7
  Top           = 0
  Width         = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 5
  Left          = 0
  Picture       = FORM6.FRX:1B800
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 6
  Top           = 0
  Width         = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 4
  Left          = 0
  Picture       = FORM6.FRX:20542
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 5
  Top           = 0
  Width         = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 3
  Left          = 0
  Picture       = FORM6.FRX:25284
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 4
  Top           = 0
```

```
  Width        = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 2
  Left          = 0
  Picture       = FORM6.FRX:29FC6
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 3
  Top           = 0
  Width         = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 0
  Left          = 0
  Picture       = FORM6.FRX:2ED08
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 2
  Top           = 0
  Width         = 2175
End
Begin PictureBox Picture1
  BackColor     = &H00FFFFFF&
  Height        = 1815
  Index         = 1
  Left          = 0
  Picture       = FORM6.FRX:33A4A
  ScaleHeight   = 1785
  ScaleWidth    = 2145
  TabIndex      = 1
  Top           = 0
  Width         = 2175
End
Begin CommandButton Command1
  Caption       = "OK"
  Height        = 375
  Left          = 120
  TabIndex      = 0
  Top           = 1920
  Width         = 5415
End
End
Dim f, about

Sub Command1_Click ()
```

```
 about = 2
End Sub

Sub Form_Activate ()
 about = 1
End Sub

Sub Form_Unload (Cancel As Integer)
 about = 2
 Unload form6
End Sub

Sub Timer1_Timer ()
 f = f + 1
 If f > 8 Then f = 1
 picture1(f).Visible = True
 For n = 1 To 8
  If (n <> f) Then picture1(n).Visible = False
 Next n
 x% = DoEvents()
 If about = 2 Then
   timer1.Enabled = False
   Unload form6
 End If
End Sub
```

Arquivo WDAQ_VB.BAS:

```
'*****************************************************************************
' *
' *   NI-DAQ Windows - Function Prototypes
' *   Copyright    (C) National Instruments 1994.
' *
'*****************************************************************************


Declare Function A2000_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e#)
Declare Function A2000_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function A2150_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function AI_Check% Lib "nidaq.dll" (ByVal a%, b%, c%)
Declare Function AI_Clear% Lib "nidaq.dll" (ByVal a%)
Declare Function AI_Configure% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%)
Declare Function AI_Mux_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function AI_Read% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%)
Declare Function AI_Read_Scan% Lib "nidaq.dll" (ByVal a%, b%)
Declare Function AI_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function AI_VRead% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d#)
Declare Function AI_VRead_Scan% Lib "nidaq.dll" (ByVal a%, b#)
Declare Function AI_VScale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d#, ByVal e#, ByVal f%, g#)
Declare Function Align_DMA_Buffer% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d&, ByVal e&, f&)
```

```
Declare Function AO_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function AO_Change_Parameter% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&, ByVal d&)
Declare Function AO_Configure% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e#, ByVal f%)
Declare Function AO_Update% Lib "nidaq.dll" (ByVal a%)
Declare Function AO_VScale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c#, d%)
Declare Function AO_VWrite% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c#)
Declare Function AO_Write% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function Calibrate_E_Series% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&, ByVal d#)
Declare Function Config_Alarm_Deadband% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c$, ByVal d#, ByVal e#, ByVal f%,
ByVal g%, ByVal h%, ByVal i&)
Declare Function Config_ATrig_Event_Message% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c$, ByVal d#, ByVal e#, ByVal
f%, ByVal g&, ByVal h&, ByVal i&, ByVal j%, ByVal k%, ByVal l&)
Declare Function Config_DAQ_Event_Message% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c$, ByVal d%, ByVal e&, ByVal
f&, ByVal g&, ByVal h&, ByVal i&, ByVal j%, ByVal k%, ByVal l&)
Declare Function Configure_HW_Analog_Trigger% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&, ByVal d&, ByVal e&, ByVal
f&)
Declare Function CTR_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%)
Declare Function CTR_EvCount% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function CTR_EvRead% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%)
Declare Function CTR_FOUT_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%)
Declare Function CTR_Period% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function CTR_Pulse% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%)
Declare Function CTR_Rate% Lib "nidaq.dll" (ByVal a#, ByVal b#, c%, d%, e%)
Declare Function CTR_Reset% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function CTR_Restart% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function CTR_Simul_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d%)
Declare Function CTR_Square% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%)
Declare Function CTR_State% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function CTR_Stop% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function DAQ_Check% Lib "nidaq.dll" (ByVal a%, b%, c&)
Declare Function DAQ_Clear% Lib "nidaq.dll" (ByVal a%)
Declare Function DAQ_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function DAQ_DB_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function DAQ_DB_HalfReady% Lib "nidaq.dll" (ByVal a%, b%, c%)
Declare Function DAQ_DB_StrTransfer% Lib "nidaq.dll" (ByVal a%, ByVal b$, c&, d%)
Declare Function DAQ_DB_Transfer% Lib "nidaq.dll" (ByVal a%, b%, c&, d%)
Declare Function DAQ_Monitor% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d&, e%, f&, g%)
Declare Function DAQ_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e&, ByVal f#)
Declare Function DAQ_Rate% Lib "nidaq.dll" (ByVal a#, ByVal b%, c%, d%)
Declare Function DAQ_Start% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e&, ByVal f%, ByVal g%)
Declare Function DAQ_StopTrigger_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&)
Declare Function DAQ_to_Disk% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d$, ByVal e&, ByVal f#, ByVal g%)
Declare Function DAQ_VScale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d#, ByVal e#, ByVal f&, g%, h#)
Declare Function DIG_Block_Check% Lib "nidaq.dll" (ByVal a%, ByVal b%, c&)
Declare Function DIG_Block_Clear% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function DIG_Block_In% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d&)
Declare Function DIG_Block_Out% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d&)
Declare Function DIG_Block_PG_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%,
ByVal g%)
Declare Function DIG_DB_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%)
```

```
Declare Function DIG_DB_HalfReady% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function DIG_DB_StrTransfer% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c$, ByVal d&)
Declare Function DIG_DB_Transfer% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d&)
Declare Function DIG_Grp_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%)
Declare Function DIG_Grp_Mode% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%, ByVal
g%)
Declare Function DIG_Grp_Status% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function DIG_In_Grp% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function DIG_In_Line% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%)
Declare Function DIG_In_Port% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function DIG_Line_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function DIG_Out_Grp% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function DIG_Out_Line% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function DIG_Out_Port% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function DIG_Prt_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function DIG_Prt_Status% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function DIG_SCAN_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e%)
Declare Function DSP2200_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function DSP2200_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function Get_DAQ_Device_Info% Lib "nidaq.dll" (ByVal a%, ByVal b&, c&)
Declare Function Get_DAQ_Event% Lib "nidaq.dll" (ByVal a&, b%, c%, d%, e&)
Declare Function Get_NI_DAQ_Version% Lib "nidaq.dll" (a&)
Declare Function GPCTR_Change_Parameter% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&, ByVal d&)
Declare Function GPCTR_Config_Buffer% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&, ByVal d&, e&)
Declare Function GPCTR_Control% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&)
Declare Function GPCTR_Set_Application% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&)
Declare Function GPCTR_Watch% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&, d&)
Declare Function ICTR_Read% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function ICTR_Reset% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function ICTR_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%)
Declare Function Init_DA_Brds% Lib "nidaq.dll" (ByVal a%, b%)
Declare Function Lab_ISCAN_Check% Lib "nidaq.dll" (ByVal a%, b%, c&, d%)
Declare Function Lab_ISCAN_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e&, ByVal f#, ByVal g#, h%)
Declare Function Lab_ISCAN_Start% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e&, ByVal f%, ByVal g%,
ByVal h%)
Declare Function Lab_ISCAN_to_Disk% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d$, ByVal e&, ByVal f#, ByVal
g#, ByVal h%)
Declare Function LPM16_Calibrate% Lib "nidaq.dll" (ByVal a%)
Declare Function MAI_Arm% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function MAI_Clear% Lib "nidaq.dll" (ByVal a%)
Declare Function MAI_Coupling% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function MAI_Read% Lib "nidaq.dll" (ByVal a%, b%)
Declare Function MAI_Scale% Lib "nidaq.dll" (ByVal a%, ByVal b&, c%, d#)
Declare Function MAI_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, ByVal e%, ByVal f%, ByVal g%)
Declare Function Master_Slave_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function MDAQ_Check% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d&, e&)
Declare Function MDAQ_Clear% Lib "nidaq.dll" (ByVal a%)
Declare Function MDAQ_Get% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d&, ByVal e&, ByVal f&, ByVal g&, h%,
i&, j&, k&, l%)
Declare Function MDAQ_ScanRate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
```

```
Declare Function MDAQ_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c%, ByVal d&, ByVal e&, f%)
Declare Function MDAQ_Start% Lib "nidaq.dll" (ByVal a%, ByVal b&)
Declare Function MDAQ_Stop% Lib "nidaq.dll" (ByVal a%)
Declare Function MDAQ_StrGet% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d&, ByVal e&, ByVal f&, ByVal g&,
ByVal h$, i&, j&, k&, l%)
Declare Function MDAQ_Trig_Delay% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function MDAQ_Trig_Select% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%, ByVal
g%)
Declare Function MIO_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%, ByVal g%,
ByVal h#, ByVal i%)
Declare Function MIO_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function NI_DAQ_Mem_Alloc% Lib "nidaq.dll" (a%, ByVal b%, ByVal c&, ByVal d%, ByVal e%)
Declare Function NI_DAQ_Mem_Attributes% Lib "nidaq.dll" (ByVal a%, b%, c&, d%, e%, f%)
Declare Function NI_DAQ_Mem_Copy% Lib "nidaq.dll" (ByVal a%, b%, ByVal c&, ByVal d&, ByVal e%)
Declare Function NI_DAQ_Mem_Free% Lib "nidaq.dll" (ByVal a%)
Declare Function NI_DAQ_Mem_Lock% Lib "nidaq.dll" (ByVal a%, b&)
Declare Function NI_DAQ_Mem_Unlock% Lib "nidaq.dll" (ByVal a%)
Declare Function Peek_DAQ_Event% Lib "nidaq.dll" (ByVal a&, b%, c%, d%, e&)
Declare Function REG_Level_Read% Lib "nidaq.dll" (ByVal a%, ByVal b%, c&)
Declare Function REG_Level_Write% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&, ByVal d&, e&)
Declare Function RTSI_Clear% Lib "nidaq.dll" (ByVal a%)
Declare Function RTSI_Clock% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function RTSI_Conn% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function RTSI_DisConn% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function SC_2040_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function SCAN_Demux% Lib "nidaq.dll" (a%, ByVal b&, ByVal c%, ByVal d%)
Declare Function SCAN_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, e%, ByVal f&, ByVal g#, ByVal h#)
Declare Function SCAN_Sequence_Demux% Lib "nidaq.dll" (ByVal a%, b%, ByVal c&, d%, ByVal e%, f%, g&)
Declare Function SCAN_Sequence_Retrieve% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%)
Declare Function SCAN_Sequence_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, e%, f%, g%)
Declare Function SCAN_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%)
Declare Function SCAN_Start% Lib "nidaq.dll" (ByVal a%, b%, ByVal c&, ByVal d%, ByVal e%, ByVal f%, ByVal g%)
Declare Function SCAN_to_Disk% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, ByVal e$, ByVal f&, ByVal g#, ByVal h#,
ByVal i%)
Declare Function SCXI_1200_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%,
ByVal g%, ByVal h%, ByVal i#, ByVal j#)
Declare Function SCXI_AO_Write% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f#, ByVal g%,
h%)
Declare Function SCXI_Cal_Constants% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%,
ByVal g#, ByVal h%, ByVal i%, ByVal j%, ByVal k#, ByVal l#, ByVal m#, ByVal n#, ByVal o#, p#, q#)
Declare Function SCXI_Calibrate_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function SCXI_Change_Chan% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function SCXI_Configure_Filter% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e#, ByVal f%,
ByVal g%, h#)
Declare Function SCXI_Get_Chassis_Info% Lib "nidaq.dll" (ByVal a%, b%, c%, d%, e%, f%)
Declare Function SCXI_Get_Module_Info% Lib "nidaq.dll" (ByVal a%, ByVal b%, c&, d%, e%)
Declare Function SCXI_Get_State% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, e&)
Declare Function SCXI_Get_Status% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d&)
Declare Function SCXI_Load_Config% Lib "nidaq.dll" (ByVal a%)
Declare Function SCXI_MuxCtr_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
```

```
Declare Function SCXI_Reset% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function SCXI_Scale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d#, ByVal e#, ByVal f%, ByVal g%,
ByVal h%, ByVal i&, j%, k#)
Declare Function SCXI_SCAN_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, e%, ByVal f%, ByVal g%)
Declare Function SCXI_Set_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%, g&, h%,
i%)
Declare Function SCXI_Set_Gain% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d#)
Declare Function SCXI_Set_Input_Mode% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function SCXI_Set_State% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e&)
Declare Function SCXI_Single_Chan_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function SCXI_Track_Hold_Control% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function SCXI_Track_Hold_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%,
ByVal g%)
Declare Function Select_Signal% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&, ByVal d&)
Declare Function Set_DAQ_Device_Info% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c&)
Declare Function Timeout_Config% Lib "nidaq.dll" (ByVal a%, ByVal b&)
Declare Function Trigger_Window_Config% Lib "nidaq.dll"(ByVal a%, ByVal b%, ByVal c%)
Declare Function WFM_Chan_Control% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function WFM_Check% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d&, e&)
Declare Function WFM_ClockRate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e&, ByVal f%)
Declare Function WFM_DB_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d%, ByVal e%, ByVal f%)
Declare Function WFM_DB_HalfReady% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%)
Declare Function WFM_DB_Transfer% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, ByVal e&)
Declare Function WFM_DB_StrTransfer% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d$, ByVal e&)
Declare Function WFM_from_Disk% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d$, ByVal e&, ByVal f&, ByVal g&, ByVal
h#)
Declare Function WFM_Group_Control% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function WFM_Group_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d%)
Declare Function WFM_Load% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, ByVal e&, ByVal f&, ByVal g%)
Declare Function WFM_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, ByVal e&, ByVal f&, ByVal g#)
Declare Function WFM_Rate% Lib "nidaq.dll" (ByVal a#, ByVal b%, c%, d&)
Declare Function WFM_Scale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&, ByVal d#, e#, f%)


'*****************************************************************************
' *
' *   NI-DAQ Windows - Obsolete Function Prototypes
' *
'*****************************************************************************


Declare Function AI_Scale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d#)
Declare Function AI_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function AO_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d#, ByVal e%)
Declare Function DAQ_Scale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&, d%, e#)
Declare Function DAQ_Trigger_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&)
Declare Function Get_DA_Brds_Info% Lib "nidaq.dll" (ByVal a%, b%, c%, d%, e%, f%, g%, h%, i%)
Declare Function Lab_SCAN_Check% Lib "nidaq.dll" (ByVal a%, b%, c&, d%)
Declare Function Lab_SCAN_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e&, ByVal f#, g%)
```

```
Declare Function Lab_SCAN_Start% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, d%, ByVal e&, ByVal f%, ByVal g%)
Declare Function Lab_SCAN_to_Disk% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d$, ByVal e&, ByVal f#, ByVal
g%)
Declare Function MDAQ_Ext_Setup% Lib "nidaq.dll" (ByVal a%, ByVal b&, ByVal c%, ByVal d&, ByVal e&, ByVal f&)
Declare Function MIO16_F5_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function MIO16_F5_Calibrate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e%, ByVal f%,
ByVal g%, ByVal h#, ByVal i%)
Declare Function SCXI_Set_Filter% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%, ByVal e#, f#)
Declare Function SCXI_Set_Module_Gain% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function Set_DAQ_Mode% Lib "nidaq.dll" (ByVal a%, ByVal b%)
Declare Function WF_Check% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d&, e&)
Declare Function WF_Clear% Lib "nidaq.dll" (ByVal a%)
Declare Function WF_DB_Config% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d%)
Declare Function WF_DB_StrTransfer% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c$, ByVal d&)
Declare Function WF_DB_Transfer% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d&)
Declare Function WF_from_Disk% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c$, ByVal d&, ByVal e#, ByVal f&, ByVal g&)
Declare Function WF_Load% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, ByVal d&, ByVal e&)
Declare Function WF_Op% Lib "nidaq.dll" (ByVal a%, ByVal b%, c%, d%, ByVal e&, ByVal f&, ByVal g&, ByVal h&, ByVal i#)
Declare Function WF_Pause% Lib "nidaq.dll" (ByVal a%)
Declare Function WF_Rate% Lib "nidaq.dll" (ByVal a#, ByVal b%, c%, d%)
Declare Function WF_Resume% Lib "nidaq.dll" (ByVal a%)
Declare Function WF_Scale% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c&, ByVal d#, e#, f%)
Declare Function WF_Start% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%)
Declare Function WFM_UpdateRate% Lib "nidaq.dll" (ByVal a%, ByVal b%, ByVal c%, ByVal d&, ByVal e%)


'*****************************************************************************
' *
' *   NI-DAQ Windows - Compatibility with NI-DAQ DOS
' *
'*****************************************************************************

Declare Function USE_E_Series% Lib "nidaq.dll" ()
Declare Function USE_E_Series_AI% Lib "nidaq.dll" ()
Declare Function USE_E_Series_AO% Lib "nidaq.dll" ()
Declare Function USE_E_Series_DAQ% Lib "nidaq.dll" ()
Declare Function USE_E_Series_DIO% Lib "nidaq.dll" ()
Declare Function USE_E_Series_GPCTR% Lib "nidaq.dll" ()
Declare Function USE_E_Series_GPCTR_Simple% Lib "nidaq.dll" ()
Declare Function USE_E_Series_Misc% Lib "nidaq.dll" ()
Declare Function USE_E_Series_WFM% Lib "nidaq.dll" ()
```

Linguagem C:

Arquivo GENPORT.MAK:

```
ORIGIN   = QCWIN
ORIGIN_VER       = 1.00
```

```
PROJ     =genport
DEBUG  =1
PROGTYPE        =2
CALLER =c:\vb\vb.exe
ARGS     =
DLLS     =
CVPACK =1
CC        =cl -qc
RC        =rc
CFLAGS_G_WEXE            =/AS /G2w /Zp /W3 /D_WINDOWS
CFLAGS_D_WEXE            =/Gi /Od /Zi
CFLAGS_R_WEXE            =/O /Os /DNDEBUG
CFLAGS_G_WDLL =/AS /G2w /Zp /Aw /W3 /D_WINDOWS /D_WINDLL
CFLAGS_D_WDLL =/Gi /Od /Zi
CFLAGS_R_WDLL =/O /Os /DNDEBUG
CFLAGS_G_WTTY=/AS /G2w /W3 /D_WINDOWS
CFLAGS_D_WTTY=/Gi /Od /Zi
CFLAGS_R_WTTY=/O /Os /DNDEBUG
CFLAGS_G_DEXE =/AS /W2
CFLAGS_D_DEXE =/Gi /Od /Zi
CFLAGS_R_DEXE =/O /Ot /DNDEBUG
CFLAGS =$(CFLAGS_G_WDLL) $(CFLAGS_D_WDLL)
LFLAGS_G_WEXE=/ST:5120 /A:16
LFLAGS_D_WEXE=/CO
LFLAGS_R_WEXE=
LFLAGS_G_WDLL =/A:16/ST:5120
LFLAGS_D_WDLL =/CO
LFLAGS_R_WDLL =
LFLAGS_G_WTTY =/ST:5120 /A:16
LFLAGS_D_WTTY =/CO
LFLAGS_R_WTTY =
LFLAGS_G_DEXE =/NOI /ST:2048
LFLAGS_D_DEXE =/CO
LFLAGS_R_DEXE =
LFLAGS =$(LFLAGS_G_WDLL) $(LFLAGS_D_WDLL)
RCFLAGS          =
RESFLAGS         =/t
RUNFLAGS         =
DEFFILE =        genport.DEF
OBJS_EXT =
LIBS_EXT =

.rc.res: ; $(RC) $(RCFLAGS) -r $*.rc

all:        $(PROJ).DLL

genport.OBJ:        genport.C $(H)

$(PROJ).DLL:        genport.OBJ $(OBJS_EXT) $(DEFFILE)
```

```
        echo >NUL @<<$(PROJ).CRF
genport.OBJ +
$(OBJS_EXT)
$(PROJ).DLL

C:\COMPILER\QCWIN\LIB\+
/NOD sdllcew oldnames libw
$(DEFFILE);
<<
        link $(LFLAGS) @$(PROJ).CRF
        rc $(RESFLAGS) $(PROJ).DLL

run: $(PROJ).DLL
        $(PROJ) $(RUNFLAGS)
```

Arquivo GENPORT.DEF:

```
LIBRARY   genport

EXETYPE   WINDOWS

PROTMODE

CODE     PRELOAD MOVEABLE DISCARDABLE
DATA     PRELOAD SINGLE
SEGMENTS 'WEP_TEXT' FIXED PRELOAD
SEGMENTS _IOSEG CLASS 'IOSEG_CODE' IOPL

HEAPSIZE  1024

EXPORTS
        WEP       @1 RESIDENTNAME
        OutG              @2
        InG                       @3
```

Arquivo GENPORT.C:

```
#include <windows.h>
#include <conio.h>

int FAR PASCAL LibMain(HANDLE hModule, WORD wDataSeg, WORD cbHeapSize, LPSTR lpszCmdLine)
{
        return TRUE;
}

/*
```

```
**
*/


int FAR PASCAL OutG(int adr, int dat)
{
 int a=0;
 a=outp(adr,dat);
 return (a);
}

int FAR PASCAL InG(int adr)
{
 int a;
 a=inp(adr);
 return(a);
}
```

ARQUIVO VBINI.VBX:

```
;-----------------------------------------------------------
; VBini.def - Arquivo de definicoes do modulo para o
; controle personalizado VBINI.VBX
;-----------------------------------------------------------
LIBRARY        VBINI
EXETYPE        WINDOWS
DESCRIPTION    'Controle Personalizado VBini do Visual Basic'

CODE           MOVEABLE
DATA           MOVEABLE SINGLE
HEAPSIZE       2048
EXPORTS
               WEP     @1      RESIDENTNAME
SEGMENTS
               WEP_TEXT FIXED
;-----------------------------------------------------------
```

Arquivo VBINI.RC:

```
//-----------------------------------------------------------
// VBini.rc
//-----------------------------------------------------------
#include "vbini.h"

//-----------------------------------------------------------
// Recursos de controle para o modelo de controle VBINI
//-----------------------------------------------------------
IDBMP_VBINI        BITMAP  DISCARDABLE "vbinicu.bmp"
IDBMP_VBINIDOWN    BITMAP  DISCARDABLE "vbinicd.bmp"
IDBMP_VBINIMONO    BITMAP  DISCARDABLE "vbinimu.bmp"
IDBMP_VBINIEGA     BITMAP  DISCARDABLE "vbinieu.bmp"
```

Arquivo VBINI.H:

```
//-----------------------------------------------------------
// VBini.h
//-----------------------------------------------------------

//-----------------------------------------------------------
// Informacoes do Recurso
//-----------------------------------------------------------
// Numeros de Identificacao dos mapas de bits da Caixa de
// Ferramentas
//-----------------------------------------------------------
#define IDBMP_VBINI          8000
#define IDBMP_VBINIDOWN      8001
#define IDBMP_VBINIMONO      8003
#define IDBMP_VBINIEGA       8006

#ifndef RC_INVOKED
//-----------------------------------------------------------
// Macro para referenciar membro da estrutura
//-----------------------------------------------------------
#define OFFSETIN(struc, field)   ((USHORT)&(((struc *)0)->field))


//-----------------------------------------------------------
// Dados de controle e estruturas de VBini
//-----------------------------------------------------------
typedef struct tagVBINI
  {
  HSZ  hszFileName;
  HSZ  hszSection;
  HSZ  hszEntry;
  HSZ  hszString;
  } VBINI;

typedef VBINI FAR * PVBINI;


//-----------------------------------------------------------
// Procedimento de Controle
//-----------------------------------------------------------
LONG FAR PASCAL _export VBiniCtlProc(HCTL, HWND, USHORT,  USHORT,  LONG);


//-----------------------------------------------------------
// Informacoes de Propriedade
//-----------------------------------------------------------
PROPINFO Property_FileName =
  {
```

```
  "FileName",
  DT_HSZ | PF_fGetData | PF_fSetData | PF_fSetMsg | PF_fSaveData,
  OFFSETIN(VBINI, hszFileName),
  NULL
  };

PROPINFO Property_Section =
  {
  "Section",
  DT_HSZ | PF_fGetData | PF_fSetData | PF_fSetMsg | PF_fSaveData,
  OFFSETIN(VBINI, hszSection),
  NULL
  };

PROPINFO Property_Entry =
  {
  "Entry",
  DT_HSZ | PF_fGetData | PF_fSetData | PF_fSetMsg | PF_fSaveData,
  OFFSETIN(VBINI, hszEntry),
  NULL
  };

PROPINFO Property_String =
  {
  "String",
  DT_HSZ | PF_fGetData | PF_fSetData | PF_fSetMsg | PF_fSaveData,
  OFFSETIN(VBINI, hszString),
  NULL
  };


//-----------------------------------------------------------
// Lista de propriedade
//-----------------------------------------------------------
// Define os indices consecutivos para as propriedades
//-----------------------------------------------------------
#define IPROP_VBINI_CTLNAME        0
#define IPROP_VBINI_INDEX          1
#define IPROP_VBINI_LEFT           2
#define IPROP_VBINI_TOP            3
#define IPROP_VBINI_TAG            4
#define IPROP_VBINI_FILENAME       5
#define IPROP_VBINI_SECTION        6
#define IPROP_VBINI_ENTRY          7
#define IPROP_VBINI_STRING         8

PPROPINFO VBini_Properties[] =
  {
  PPROPINFO_STD_CTLNAME,
  PPROPINFO_STD_INDEX,
```

```
  PPROPINFO_STD_LEFT,
  PPROPINFO_STD_TOP,
  PPROPINFO_STD_TAG,
  &Property_FileName,
  &Property_Section,
  &Property_Entry,
  &Property_String,
  NULL
  };


//-----------------------------------------------------------
// Lista de eventos
//-----------------------------------------------------------
// Define os indices consecutivos para os eventos
//-----------------------------------------------------------
#define IEVENT_VBINI_CLICK            0

PEVENTINFO VBini_Events[] =
  {
  PEVENTINFO_STD_CLICK,
  NULL
  };


//-----------------------------------------------------------
// Estrutura Modelo
//-----------------------------------------------------------
// Define o modelo controle (usando estruturas de evento e
// de propriedade)
//-----------------------------------------------------------
MODEL modelVBini =
  {
  VB_VERSION,                // Versao do Visual Basic
  MODEL_fInvisAtRun,         // Sinalizadores MODEL
  (PCTLPROC)VBiniCtlProc,    // Procedimentos de Controle
  0,                         // Estilo da Classe
  0,                         // Estilo padrao do Windows
  sizeof(VBINI),             // Tamanho da estrutura VBINI
  IDBMP_VBINI,               // ID do Mapa de bits da Palete
  "VBini",                   // Nome de controle padrao
  "VBini",                   // Nome da classe do Visual Basic
  NULL,                      // Nome da classe Mae
  VBini_Properties,          // Informacoes de Propriedade
  VBini_Events,              // Tabela de informacoes dos eventos
  IPROP_VBINI_FILENAME,      // Propriedade padrao
  IEVENT_VBINI_CLICK,        // Evento padrao
  -1                         // Propriedade que representa
                             // o valor do controle
  };
```

```
#endif // RC_INVOKED

//-----------------------------------------------------------
```

Arquivo VBINI.C:

```
//-----------------------------------------------------------------
// VBini.c
//-----------------------------------------------------------------

#include <windows.h>
#include <vbapi.h>
#include <string.h>
#include "vbini.h"


//-----------------------------------------------------------------
// Variaveis Globais
//-----------------------------------------------------------------
HANDLE hmodDLL;
char FAR szBuf[80];


//-----------------------------------------------------------------
// Prototipos Locais
//-----------------------------------------------------------------
VOID NEAR GetIniString(HCTL);
VOID NEAR PutIniString(HCTL);


//-----------------------------------------------------------------
// VBini - Procedimento de Controle
//-----------------------------------------------------------------
LONG FAR PASCAL _export VBiniCtlProc
(
   HCTL   hctl,
   HWND   hwnd,
   USHORT msg,
   USHORT wp,
   LONG   lp
)
{
   PVBINI pvbini = NULL;
   LPSTR lpstr;

   switch (msg)
   {
   case VBM_CREATED:
      if (VBGetMode() != MODE_DESIGN)
```

```
        return 0;
      break;

    case WM_NCDESTROY:
      pvbini = (PVBINI)VBDerefControl(hctl);
      if (pvbini->hszFileName)
        VBDestroyHsz(pvbini->hszFileName);

      pvbini = (PVBINI)VBDerefControl(hctl);
      if (pvbini->hszSection)
        VBDestroyHsz(pvbini->hszSection);

      pvbini = (PVBINI)VBDerefControl(hctl);
      if (pvbini->hszEntry)
        VBDestroyHsz(pvbini->hszEntry);

      pvbini = (PVBINI)VBDerefControl(hctl);
      if (pvbini->hszString)
        VBDestroyHsz(pvbini->hszString);

      break;

    case VBM_METHOD:
      switch (wp)
      {
      case (METH_REFRESH):
        {
        pvbini = (PVBINI)VBDerefControl(hctl);
        lpstr = VBLockHsz(pvbini->hszString);
        if (lstrlen(lpstr))
          {
          VBUnlockHsz(pvbini->hszString);
          PutIniString(hctl);
          }
        else
          {
          VBUnlockHsz(pvbini->hszString);
          GetIniString(hctl);
          }
        return 0;
        }
      }
    }

    return VBDefControlProc(hctl, hwnd, msg, wp, lp);
  }


  //-----------------------------------------------------------------------
  // Registra o Controle personalizado; o Visual Basic chama essa rotina
```

```
// quando o controle DLL e carregado para uso
//-----------------------------------------------------------------------
BOOL FAR PASCAL _export VBINITCC
(
   USHORT usVersion,
   BOOL   fRuntime
)
{
   // Evita advertencias sobre paramentros formais nao usados (mas exigidos)
   fRuntime  = fRuntime;
   usVersion = usVersion;

   // Registra o(s) controle(s)
   return VBRegisterModel(hmodDLL, &modelVBini);
}


//-----------------------------------------------------------------------
// Inicializa a biblioteca; esta rotina e chamada quando o primeiro
// cliente carrega a DLL
//-----------------------------------------------------------------------
int FAR PASCAL LibMain
(
   HANDLE hModule,
   WORD   wDataSeg,
   WORD   cbHeapSize,
   LPSTR  lpszCmdLine
)
{
   // Evita advertencias sobre paramentros formais nao usados (mas exigidos)
   wDataSeg        = wDataSeg;
   cbHeapSize      = cbHeapSize;
   lpszCmdLine = lpszCmdLine;

   hmodDLL = hModule;

   return 1;
}


//-----------------------------------------------------------------------
// Tenta gravar uma string de perfil privada usando as propriedades atuais
//-----------------------------------------------------------------------
VOID NEAR PutIniString( HCTL hctl )
{
   PVBINI pvbini = NULL;
   LPSTR  lpstrFileName;
   LPSTR  lpstrSection;
   LPSTR  lpstrEntry;
   LPSTR  lpstrString;
```

```
  BOOL  fSuccess;

  pvbini = (PVBINI)VBDerefControl(hctl);
  lpstrFileName = VBLockHsz(pvbini->hszFileName);

  pvbini = (PVBINI)VBDerefControl(hctl);
  lpstrSection = VBLockHsz(pvbini->hszSection);

  pvbini = (PVBINI)VBDerefControl(hctl);
  lpstrEntry = VBLockHsz(pvbini->hszEntry);

  pvbini = (PVBINI)VBDerefControl(hctl);
  lpstrString = VBLockHsz(pvbini->hszString);

  if (lstrlen(lpstrFileName) && lstrlen(lpstrSection) &&
          lstrlen(lpstrEntry) && lstrlen(lpstrString))
     fSuccess = WritePrivateProfileString( lpstrSection,
            lpstrEntry, lpstrString, lpstrFileName );

  VBUnlockHsz(pvbini->hszFileName);
  VBUnlockHsz(pvbini->hszSection);
  VBUnlockHsz(pvbini->hszEntry);
  VBUnlockHsz(pvbini->hszString);
}


//-----------------------------------------------------------------------
// Tenta ler uma string de perfil privada usando as propriedades atuais
//-----------------------------------------------------------------------
VOID NEAR GetIniString( HCTL hctl )
{
   PVBINI pvbini = NULL;
   LPSTR  lpstrFileName;
   LPSTR  lpstrSection;
   LPSTR  lpstrEntry;

   pvbini = (PVBINI)VBDerefControl(hctl);
   lpstrFileName = VBLockHsz(pvbini->hszFileName);

   pvbini = (PVBINI)VBDerefControl(hctl);
   lpstrSection = VBLockHsz(pvbini->hszSection);

   pvbini = (PVBINI)VBDerefControl(hctl);
   lpstrEntry = VBLockHsz(pvbini->hszEntry);

   GetPrivateProfileString( lpstrSection, lpstrEntry, "",
                    szBuf, sizeof(szBuf), lpstrFileName );

   VBSetControlProperty( hctl, IPROP_VBINI_STRING, (LONG)szBuf );
```

```
    VBUnlockHsz(pvbini->hszFileName);
    VBUnlockHsz(pvbini->hszSection);
    VBUnlockHsz(pvbini->hszEntry);
}

//-----------------------------------------------------------------
•
```