UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E
DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES
DECOM/FEEC/UNICAMP

UNICAMP

Sistema Automatizado com Interface IEEE - 488 para a Calibração de Instrumentos

Por:
Eng. Vicente Idalberto Becerra Sablón

Orientdor:
Prof. Dr. José Geraldo Chiquito

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas, como parte dos pré-requisitos necessários para a obtenção do título de Mestre em Engenharia Elétrica.
Área de Concentração: Eletrônica e Comunicações

Campinas- SP - Brasil
1998

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

B386s

Becerra Sablón, Vicente Idalberto
Sistema automatizado com interface IEEE – 488 para a calibração de instrumentos. / Vicente Idalberto Becerra Sablón.--Campinas, SP: [s.n.], 1998.

Orientador: José Geraldo Chiquito
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Controle automático - Medição. 2. Controle automático – Instrumentação. 3. Calibração. 4. Laboratórios – Aparelhos e Instrumentos. 5. Instrumentos de medição. I. Chiquito, José Geraldo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

A

Zaida

Sandra

e *Daniel*

Agradecimentos

Ao Prof. Dr. José Geraldo Chiquito pela sua orientação, apoio e oportunidade de realização deste trabalho.

Ao Prof. Dr. Yuzo Iano, pela valiosa oportunidade de participar das atividades do seu grupo de pesquisa.

Ao Prof. Dr. Max Henrique Machado Costa pela oportunidade que me concedeu de estudar no Brasil.

Aos professores da Faculdade de Engenharia Elétrica e Computação da UNICAMP, especialmente do Departamento de Comunicações, por sua grande contribuição para a culminação deste trabalho.

Ao CEMEQ especialmente aos colegas do Laboratório de Calibração Elétrica (LACE), Alexandre e Antonio, pelo suporte técnico necessário à realização da parte experimental e por criarem um ambiente agradável de trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo apoio financeiro fornecido para a realização do Mestrado.

Ao Fundo de Amparo à Pesquisa do Estado de São Paulo (FAPESP), pelo apoio financeiro concedido para a realização do Doutorado.

À amiga Rosa e aos amigos Fábio e Evelio pelo apoio sem os quais esta tese não teria sido escrita.

Aos amigos do Departamento, especialmente Naylamp, Luis Rómulo, Flavio, Guillermo, Alexandra, Fabricio, Edimilson, e Carlos, pela companhia durante o Mestrado.

Aos amigos Camilo, Hector, e amiga Carmen pelo o incentivo nos momentos mais críticos desta e de outras jornadas.

Aos pesquisadores Alencar de Melo Jr e Antonio Agusto De Moraies pelas enriquecedoras discussões técnicas.

À minha família, pelo estimulo durante o mestrado e em outros momentos igualmente importantes e pelo carinho e atenção que dispensaram a mim durante toda minha vida.

Resumo

Nesta dissertação apresentamos um sistema automatizado para a calibração de instrumentos com interface IEEE-488 (*General Purpose Interface Bus*, **GPIB**), desenvolvido no Laboratório de Calibracão Elétrica (**LACE**) junto ao Departamento de Comunicações (**DECOM**) da **UNICAMP.** O sistema integra em uma única rede, instrumentos dos mais diversos fabricantes, tais como: *Hewlett Packard, Tektronix, Fluke, National Instruments, Valhalla Scientific* e outros. Para isso, é utilizado o cartão controlador **PC-IIA** da *National Instruments*, o qual, colocado dentro de um microcomputador **IBM-PC,** permite realizar o controle de toda a operação da rede, junto ao *software* desenvolvido chamado **GPIBN*et***, que por sua vez padroniza a comunicação entre todos os instrumentos. Este sistema permite automatizar rotinas de aferição, calibração e teste, compatibilizando as especificações dos diferentes equipamentos. Com essa utilização, busca-se um aumento significativo da produtividade e principalmente da qualidade da medição, devido à minimização e compensação dos erros relativos ao sistema de medição e à apreciação humana. Além disso, possibilita-se a operação do sistema na máxima eficiência e facilita-se a geração da documentação (relatórios, certificados e tabelas).

Abstract

In this work we present an automatized system for calibration of instruments with the IEEE-488 (*General Purpose Interface Bus*, **GPIB**) interface. This system was developed in the Electrical Calibration Laboratory (**LACE**) together with the collaboration of Department of the Communications (**DECOM**) of the **UNICAMP.** The system can integrates the instruments of different brands, such as: *Hewlett Packard, Tektronix, Fluke, National Instruments, Valhalla Scientific* and others, in a single network interconnected under the interface provided by the controller card of *National Instruments* **PC-IIA** put into an **IBM-PC** microcomputer. For the control of all operations of the network, it was developed a software called **GPIBN*et,*** which standardizes the communication of all connected instruments. This system permits the automation of the routines of measurement, calibration and test, making compatible the specifications of different equipment. By using this system, we can get a significant increase in the productivity and, mainly, in the measurement quality, due to the minimization and compensation of the relative errors of the measurement system and the human appreciation. Moreover it permits a maximum efficiency operation of the system and facilitates the documentation generation (reports, certificates and tables).

Glossário

AH	Acceptor Handshake
ANSI	American National Standard Institute
ATN	Attention
C	Controlador
CEMEQ	Centro de Manutenção de Equipamentos
CIC	Controller-in Charge.
DAV	Data Valid
DC	Device Clear
DCL	Device Clear
DT	Device Triger
EOI	End or Indentify
GET	Group Execute Trigger
GPIB	General Purpose Interface Bus
GPIBN*et*	Sistema Operacional da Rede
GTL	Go To Local
HP-IB	Hewlett-Packard Interface Bus
IEEE	Institute of Electrical and Electronics Engenieers
IFC	Interface Clear
INMETRO	Instituto Nacional de Metrologia, Normalização e Qualidade Industrial
L	Listener
LACE	Laboratório de Calibração Elétrica
LE	Listener Extendido
LLO	Local Lockout
NDAC	Not Data Accepted
NRFD	Not Ready For Data
PC-IIA	Cartão Controlador GPIB da National Instruments
PP	Parallel Poll
PPC	Parallel Poll Configure
PPD	Parallel Poll Disable

PPE	Parallel Poll Enable
PPU	Parallel Poll Unconfigure
RBC	Rede Brasileira de Calibração
REN	Remote Enable
RL	Remote/Local
SCPI	Standard Commands for Programmable Instruments
SDC	Selective Device Clear
SH	Sourse Handshake
SPD	Serial Poll Disable
SPE	Serial Poll Enable
SR	Service Request
SRQ	Service Request
T	Talker
TC	Take Control
TE	Talker Extendido
UNL	Unlisten
UNT	Untalk
bd	especifica o número do cartão interface ou dispositivo
cnt	especifica o número de bytes a ser enviados
name	nome simbólico do dispositivo
buf	*buffer* para o armazenamento dos dados que forem lidos do GPIB
mask	máscara de bits para os mesmos bits da palavra de status
v	habilita ou desabilitada a função pode ser, "0" ou diferente de "0"

Índice

Capítulo 1

Introdução

1.1 Introdução

O presente trabalho foi desenvolvido no Laboratório de Calibração Elétrica, **LACE** do Centro para Manutenção de Equipamentos, **CEMEQ** da UNICAMP. Este laboratório tem como função fundamental a calibração pós-manutenção de todo o equipamento elétrico da universidade e de outras instituições e empresas com certificação **ISO 9000**. Cada cliente tem a possibilidade de exigir a aplicação de sua própria carta ou metodologia de calibração conforme o procedimento adotado dentro da norma brasileira. Isto faz que ao **LACE** cheguem além de uma grande e variada quantidade de equipamentos, diferentes propostas de metodologias para que sejam aplicadas. Por outro lado o conjunto de padrões do laboratório também procede de diferentes fabricantes e em muitos casos, sobre tudo nos mais modernos, tem associados pacotes de *software* que permitem a interação apenas entre instrumentos da mesma família.

Até o momento, o trabalho de aferição e calibração no **LACE** se realizava através de sistemas manuais que se limitavam apenas a alguns casos de interconexão entre o computador e um instrumento, estes sistemas reportavam pouquíssimas vantagens em comparação com o dispêndio de tempo em programação que representava, devido aos programas serem desenvolvidos especificamente para o controle de um instrumento em cada caso.

Todas estas características expostas anteriormente sugeriram a necessidade de se criar um sistema que de forma automatizada:

- controle e permita a comunicação entre instrumentos de diferentes fabricantes
- execute procedimentos de calibração e teste
- gere relatórios de forma automática
- armazene os resultados.

A implantação de um sistema com as características anteriormente mencionadas permitirá conseguir um aumento significativo da produtividade e principalmente da qualidade da medição no processo de calibração no **LACE**.

1.2 Objetivos do trabalho

O objetivo do presente trabalho foi a criação de uma rede automatizada de instrumentos com interface IEEE-488. Para o funcionamento desta rede foi desenvolvido o sistema operacional **GPIBN*et***, o qual padroniza a comunicação de todos os instrumentos conectados em rede. Sobre este sistema é construído todo o conjunto de programas aplicativos que realizará a calibração e teste do equipamento dentro do laboratório, os quais são desenvolvidos somente fazendo uso do repertório de instruções de cada instrumento mais um conjunto reduzido de instruções do **GPIBN*et***

Fica conformada uma rede, constituídas por uma estação de trabalho que é um microcomputador IBM-PC, com cartão controlador **GPIB** da *National Instruments*, (**PC-IIA**) junto aos instrumentos padrões, que servem como referência, geradores de sinais, que entregam as sinais que serão aferidas, e os instrumentos que serão calibrados. Todo este conjunto de equipamentos opera baixo o controle do **GPIBN*et*** que foi o programa desenvolvido para realizar a função de sistema operacional desta rede.

1.3 Estrutura do trabalho

A apresentação deste trabalho está organizada em seis capítulos incluindo esta introdução. No Capítulo 2 é feito uma descrição das funções dos Laboratórios de Metrologia, assim como um estudo das principais características e dificuldades do processo de calibração de instrumentos no Laboratório de Calibração Elétrica, **LACE**. No apítulo 3 é apresenta-se um estudo da *General Purpose Inteface Bus*, **GPIB**. O Capítulo 4 apresenta um estudo do cartão controlador GPIB **PC-IIA** da *National Instruments* que foi usado no projeto. A partir das discussões dos três primeiros capítulos, no Capitulo 5 é apresentado um descrição das principais caraterísticas, estrutura e funções do sistema projetado e construído. Mostra-se o conjunto de funções do sistema operacional da rede, **GPIBN*et*** e é proposta a metodologia para a realização de programas aplicativos a partir deste *software*. No Capítulo 6 são repassados os objetivos propostos para este trabalho e os resultados nele alcançados. Feito isso, são sugeridas algumas linhas de ação para lhe dar continuidade. Finalmente, no apêndice A, apresenta-se a listagem do código fonte do **GPIBN*et*** e de alguns dos programas aplicativos feitos para avaliar o sistema.

Capítulo 2

Laboratórios de Metrologia

Neste capítulo realiza-se uma descrição das funções dos Laboratórios de Metrologia, das principais características e dificuldades do processo de calibração de instrumentos no Laboratório de Calibração Elétrica, **LACE**.

2.1 Laboratórios de Metrologia

Os objetivos dos laboratórios de metrologia são manter os padrões primários nacionais do setor e assegurar uma adequada cadeia de rastreabilidade das medições o padrões internacionais.

Assim sendo, os laboratórios proporcionam a calibração de equipamentos, com a finalidade de constatar o cumprimento das especificações do cliente de modo que se podam satisfazer os requerimentos da **ISO 9000** ou outra norma que assegure a qualidade.

Os serviços metrológicos ofertados por estos laboratórios são divididos em duas grandes áreas básicas **[De Medeiros, 1990] e [Langsdorff & Lafin, 1995]**:

- **Metrologia Legal**: Permite a realização do controle governamental. Dá uma exatidão conveniente das medidas usadas no comercio e uma garantia pública do ponto de vista da segurança.
- **Metrologia Cientifica e Industrial:** Controla todas as ações seqüenciais que vão desde a metrologia básica, buscando definições das unidades de medida do Sistema Internacional, reproduzindo e conservando essas unidades até o controle de especificações de produtos industrializados e das medições realizadas nas industrias.

Os serviços metrológicos não incluídos no âmbito da **Metrologia Legal** estão descentralizados ao longo do território nacional através da Rede Brasileira de Calibração, **RBC**, composta de laboratórios credenciados pelo Instituto Nacional de Metrologia, Normalização e Qualidade Industrial, **INMETRO** para executar serviços de aferição/calibração de instrumentos de medição não usados em transações comerciais **[De Medeiros, 1990].**

Dentro do conjunto de laboratórios credenciados pela **INMETRO,** para a **RBC** se encontra o Laboratório de Calibração Elétrica, **LACE** do Centro de Manutenção de Equipamentos, **CEMEQ** da **UNICAMP;** através do qual se asseguram a qualidade e o rigor das medições de todos seus clientes.

2.2 Descrição Geral do trabalho no LACE

No **LACE** se realiza a calibração de pós-manutenção de parte dos equipamentos elétricos e eletrónicos da UNICAMP, bem como os de outras instituições e empresas com certificação **ISO 9000**. Este laboratório recebe uma média anual de 250 equipamentos para realizar manutenção, segundo dados dos últimos três anos.

Cada cliente do laboratório tem a possibilidade de optar pela aplicação de seu próprio procedimento ou metodologia de calibração dentro da norma brasileira. Isto faz com que para um mesmo tipo de equipamento sejam possíveis até cinco tipos diferentes de procedimentos, os quais estão concentrados em três grandes grupos, a saber:

Grupo I: Chama-se também de Grupo de Pesquisas. Nele se incluem os equipamentos dos clientes que precisam exatamente do procedimento de calibração que propõe o fabricante; E o caso dos equipamentos que exigem o emprego da metodologia que propõe o fabricante e algum procedimento mais especifico; bem como os que solicitam o cálculo do erro de medição a partir da curva de calibração que fornece o fabricante.

Grupo II : Neste grupo se incluem os equipamentos daquelas empresas que, dentro da norma **ISO 9000**, têm seu própria metodologia de calibração.

Grupo III : A este grupo pertencem os equipamentos das empresas que exigem a calibração de seu instrumento sob os procedimentos do **INMETRO [De Medeiros, 1993].**

Através dos procedimentos que podem ser realizados por meio desses grupos, são emitidos os **Certificados** de Calibração, que são os documentos onde são colocados os procedimentos, resultados obtidos, instrumentos e configuração utilizada para a realização da aferição, registro das medições, etc. São editados a partir das exigências de cada um destes grupos **[De Medeiros, 1990].**.

O trabalho de aferição e calibração no laboratório era realizado através de:

- Sistemas manuais;
- Interconexão entre o computador e um instrumento;
- Sistemas para a calibração de equipamentos *FLUKE.*

Aproximadamente 90 % dos trabalhos realizados no laboratório eram feitos utilizando **Sistemas Manuais**. A configuração geralmente usada neste caso é a que se mostra na Figura 2.1. Nestes sistemas, o operador realiza o controle de todo o trabalho, a partir de um roteiro padronizado da seqüência de operações. O roteiro seguido pelo operador deve conter as seguintes tarefas:

- Aferição dos parâmetros e armazenamento dos dados;
- Realização dos cálculos, correções e avaliação estatística;
- Geração dos relatórios e gráficos;
- Armazenamento dos dados de acordo com o padrão usado;
- Desenvolvimento e edição dos Certificados na forma padrão, colocando todas as repetições das medições realizadas;
- Controle do banco de dados de calibração e de cada cliente.

Os roteiros têm a vantagem de ser núcleo para desenvolvimentos dos programas de muitos sistemas automatizados.

Figura 2.1:Sistemas Manuais

Neste tipo de sistema os erros devido à apreciação humana inevitavelmente estão presentes, além do fato de que um aumento da velocidade de medição e o incremento do número de leituras depende diretamente da experiência e da habilidade prática do operador. Por outro lado, o ajuste dos parâmetros, cálculo dos resultados, emissão da documentação, etc; consomem um tempo relativamente grande, ocasionando baixa produtividade e eficiência do sistema em comparação com os sistemas automatizados.

A Tabela 2.1 mostra o tempo que se investe na aferição e calibração de alguns instrumentos com o uso deste sistema. Observa-se que existem instrumentos, como analisadores de espectro, para os quais não era possível realizar a calibração.

Tabela 2.1:Tempo de aferição e calibração de alguns instrumentos.

Tipo de instrumento	*Tempo mínimo para sua calibração*	*Tempo máximo para sua calibração*
Multímetro	4 *h*	4 *dias*
Osciloscópio	2½ *h*	6*h*
Analisador de espectros*	8dias	-
Gerador*	3 *h*	10 *h*
Freqüencímetro*	3 *h*	4 *dias*

***Estes equipamentos não se calibravam no LACE**

Em se tratando da **Interconexão entre o computador e um instrumento,** este sistema só está projetado para um equipamento (multímetro) do **LACE,** como se mostra na Figura 2.2. Embora o sistema requereria a presença periódica do operador, sua produtividade e eficiência é superior a dos sistemas manuais, considerando que uma parte dos parâmetros podem ser ajustadas e medidos automaticamente. O resto é ajustado e

medido pelo operador, mas sob controle do programa. O sistema projetado no laboratório foi realizado a partir de um computador IBM-PC do tipo XT, e tem as seguintes caraterísticas:

- A programação foi realizada na linguagem **BASIC**. Os programas desenvolvidos para o controle do instrumento são simples e pobremente estruturados, com pouca flexibilidade e confiabilidade, e sua modificação se converte em uma coisa muito dificultosa em relação ao dispêndio de tempo em programação que necessita;
- O uso da interface **GPIB** se faz de forma muito primitiva;
- A pesar de o controle do instrumento ser interativo, o diálogo se realiza de forma forçada, uma vez que a seqüência de perguntas e respostas não ficou bem estruturada.

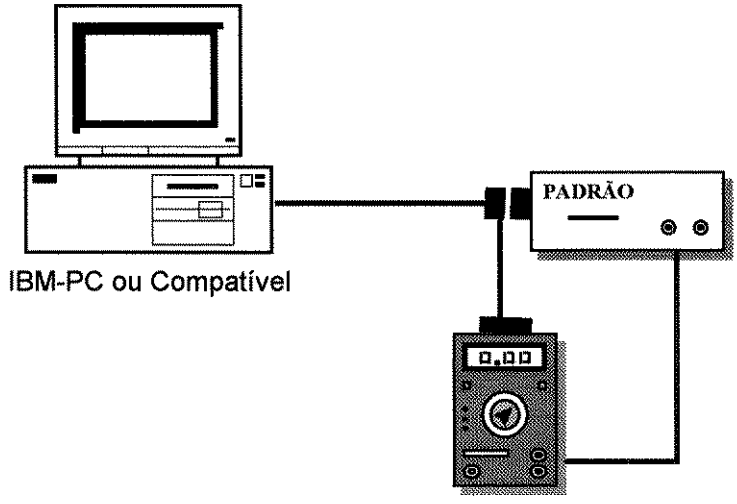


Figura 2.2:Interconexão entre o computador e um instrumento

Por sua vez, no **Sistema para a calibração de equipamentos *FLUKE,*** o controlador e o pacote de *software* associado a este sistema, estão especialmente fabricados para permitir a interação apenas entre os mais modernos instrumentos da mesma família. Esta é precisamente a dificuldade fundamental deste sistema para o trabalho no **LACE,** onde se realiza a calibração dos instrumentos dos mais diversos fabricantes.

O número de vantagens que representa o trabalho com estes sistemas no **LACE** é insignificante em relação ao dispêndio de tempo e esforço que têm que dispender os especialistas para obter um trabalho de qualidade como o centro o exige. A partir das condições próprias deste laboratório, equipamentos, informações e literatura fornecidos pelo próprios fabricantes dos múltiplos equipamentos e pelas outras vias de informação já conhecidas, a direções do **CEMEQ** e do **LACE** propõem o estudo da problemática e o começo da realização de um projeto de solução para esta.

Como primeiro resultado do estudo chegou-se à conclusão da necessidade de automatizar o processo de aferição e calibração no laboratório, a partir das justificativas expostas na Seção 2.3.

2.3 Justificativa da automatização no LACE

Os motivos pêlos quais o sistema projetado e desenvolvido é automatizado são as seguintes:

1. **Produtividade**
   - O sistema trabalha sempre visando à máxima eficiência;
   - Para ajustar os parâmetros, registrar as leituras, etc, se consome um tempo menor;
   - Os resultados dos cálculos (formulas complexas, correções, avaliações estatísticas, etc.) são realizados a uma alta velocidade.
2. **Incremento da precisão e confiabilidade**
   - A maior parte dos erros humanos podem ser eliminados;
   - Os resultados da repetição das medições são mais consistentes;
   - Os erros relativos ao sistema de medição podem ser calculados e compensados automaticamente;

- A velocidade do sistema de medição facilita um incremento do numero de leituras.

3. **Automatização da documentação**
   - Visa incrementar a qualidade e diminuir o tempo de edição e confecção de relatórios, tabelas, diagramas, certificados, etc.
4. **Aumento da capacidade para o armazenamento de dados**
   - Para manter os dados armazenados em memória e/ou discos para que possam ser processados em outro momento;
   - Para armazenar dados por um tempo longo;
   - Para manter bancos de dados de: medições, clientes, administração dos serviços, etc.
5. **Incremento da segurança**
   - Para minimizar a probabilidade de erros humanos;
   - Para verificar os parâmetros programados, controlar os limites críticos e prevenir uma *overloading* no sistema;
   - Para prevenir e/ou desligar o sistema ante a detecção de erros críticos não previsíveis e situações perigosas no *hardware*.

Outro, aspecto a avaliar em nosso projeto foi a forma em que se realizaria a comunicação dos instrumentos, ante duas alternativas, interface serial padrão **RS-232** e a interface paralelo padrão **IEEE-488** também conhecida como *General Purpose Interface Bus* (**GPIB**).

2.4 Avaliação da interface serial RS-232

Esta avaliação foi realizada considerando fundamentalmente as principais caraterísticas desta interface e sua aplicabilidade em processos de instrumentação.

2.4.1 Vantagens

- Simplicidade;
- Não precisa de um hardware extra pois, todos os PC têm de forma padrão uma ou duas portas séries;
- Pouco sensíveis ao ruído eletromagnético;

2.4.2 Problemas

- Nem todos os instrumentos e equipamentos tem compatibilidade com essa interface.
- Não é apropriada para uso em redes. Foi desenvolvida para a comunicação ponto a ponto.

2.5 Avaliação da interface paralelo IEEE-488.

2.5.1 Vantagens

- Desde um inicio foi concebida para ser aplicada em processos de instrumentação;
- Geralmente os equipamentos e instrumentos tem compatibilidade com essa interface;
- Facilidade de endereçamento;
- Facilita a conexão em rede dos equipamentos;
- Maior velocidade que a interface RS-232.

2.5.2 Problema

- Precisa de um hardware extra, isto é, de um cartão controlador dentro do computador;
- Utiliza um conector especializado, o qual tem um preço relativamente elevado.

A problemática da **RS-232** unida às grandes vantagens que proporciona a interface **IEEE-488** na área de instrumentação, fez com que fosse selecionada esta última. No Capitulo 3 faz-se um estudo detalhado das características da GPIB.

Como solução definitiva para o projeto decidiu-se projetar e construir uma **rede automatizada com interface IEEE-488 para a calibração de instrumentos** a qual permita: a comunicação e controle de instrumentos de diferentes fabricantes, execução de procedimentos de aferição, calibração e teste, gerar relatórios e certificados de forma automática, e armazenar os resultados obtidos.

Esta rede foi construída sob o cartão controlador **PC-IIA** fornecido pela *National Instruments* que é uma interface GPIB para microcomputadores compatíveis com o IBM-PC ,e utilizando o sistema operacional MS-DOS (PC-DOS). Esta interface é descrita no Capitulo 4.

Para controlar toda a operação desta rede, isto é, enviar processar, transformar, comunicar, etc foi desenvolvido o *software* que foi chamado de **GPIBN*et***. Este *software* constitui o sistema operacional do sistema **[Tanenbaum, 1992] e [Tanenbaum & Woodhull, 1997]**. As características do **GPIB*net*** são descritas no Capitulo 5

Capítulo 3

Interface GPIB

3.1 Histórico

Combinar instrumentos de medição num único sistema traz consigo, em geral, sérios problemas de interface devidos à: mistura de instrumentos de diferentes fabricantes, diferentes tipos de conectores, diferentes sinais de controle de dados, etc. Estes problemas se agravam quando se conectam equipamentos de medição a um computador **[Grimberg, 1990]**. É por isso que em 1965 a *Hewlett-Packard Company* projetou sua *Hewlett-Packard Interface Bus* (**HP-IB**) para conectar sua linha de instrumentos programáveis com seus computadores. Desta forma, se iniciam os estudos para a padronização do protocolo de comunicação não previstos até o momento **[National, 1997]**.

Devido à alta razão de transferência da **HP-IB** (nominalmente 1 Mbyte/s), esta interface foi ganhando popularidade rapidamente. Atualmente o nome de *General Purpose Interface Bus* (**GPIB**) é mais amplamente usado que **HP-IB**. Mais tarde em 1975 passou a ser chamado de IEEE *Standard* 488 e, em 1987 converteu-se em ANSI/IEEE *Standard* 488.1. Nesse ano surgiu também o ANSI/IEEE 488.2 que padronizou a definição original estabelecendo como controlar e comunicar instrumentos. Na IEEE 488.2 foi definida a estrutura da *Standard Commands for Programmable Instruments* (**SCPI**), que é um conjunto simples de comandos que podem ser usados com qualquer instrumento **SCPI**.

O SCPI e o padrão IEEE 488.2 eliminavam algumas das limitações da IEEE 488 original tornando possível o desenvolvimento de sistemas mais compatíveis e produtivos. O **SCPI** simplificava a tarefa de programação e a IEEE 488.2

aumentava o nível de padronização da IEEE 488.1 padronizando os formatos de dados, as palavras de *status,* erro de manipulação, funcionamento dos controladores, e comandos comuns para os quais todos os instrumentos devem responder de uma forma predefinida. O padrão IEEE 488.2 se concentra fundamentalmente no protocolo de *software* mantendo a compatibilidade com o *hardware* estabelecido pelo padrão IEEE 488.1 **[Tóth, 1996]** e **[National, 1997].**

Em 1992 é realizada uma revisão do padrão IEEE 488.2 e em 1993 a *National Instruments* desenvolveu o protocolo de manipulação **GPIB** de alta velocidade (chamado de **HS488**), como uma extensão do padrão IEEE 488.1, para incrementar a razão de transferência de dados de um sistema **GPIB**. Quando num sistema nem todos os instrumentos permitem a utilização do protocolo **HS488**, o dispositivo que o possui automaticamente usa o padrão de manipulação IEEE 488.1 para assegurar a compatibilidade . **HS488** é um padrão que engloba o IEEE 488 **[National, 1997]** e **[Patel, 1997].**

3.2 Mensagens GPIB

Os dispositivos **GPIB** se comunicam com outros dispositivos também **GPIB** enviando, através da interface, mensagem de dois tipos **[National, 1997]:**

- **Mensagens dependentes do dispositivo**, freqüentemente chamados de dados ou mensagens de dados, os quais contém informações específicas de cada dispositivo, tais como: instruções de programação, resultados das medições e arquivos de dados.
- **Mensagens da interface**, que controlam o barramento, usualmente chamados comandos ou mensagens de comandos. Eles realizam funções tais como inicialização do barramento, endereçamento de dispositivos, e colocação dos dispositivos no modo remoto ou de programação local.

O termo "comando" anteriormente usado não deve ser confundido com as instruções dos dispositivos que também são chamadas de comandos.

3.3 Funções básicas da interface

Cada instrumento tem sua própria função especial (*funtion device*) e as funções da interface, ambas podem ser desenvolvidas independentemente **[Grimberg, 1990]**.

As funções básicas da interface nos dispositivos **GPIB** podem ser de acordo ao padrão IEEE 488: ***Talker***, ***Listener***, e/ou **Controlador**. Um ***Talker*** é um dispositivo capaz de enviar uma mensagem de dados sobre a interface quando está endereçado para um ou mais ***Listener***, que são aqueles dispositivos capazes de receber os dados. O dispositivo **Controlador** administra o fluxo de informação sobre o **GPIB** pelo envio de comandos a todos os equipamentos **[Hewlett, 1988]** e **[ANSI & IEEE, 1992]**.

O **Controlador** determina qual dispositivo tem que enviar dados e qual tem que receber, também envia comandos especiais e sinais de controle. Usualmente endereça ou habilita um ***Talker*** e um ***Listener***, antes do ***Talker*** enviar uma mensagem ao ***Listener***. Após a transmissão da mensagem o **Controlador** pode endereçar outros ***Talkers*** e ***Listeners*** **[Grimberg, 1990]** e **[ANSI & IEEE, 1992]**.

Algumas configurações **GPIB** não requerem **Controlador**, por exemplo: um dispositivo que é sempre ***Talker*** , é designado ***Talk-only***, e conectado a um ou mais dispositivos ***Listen-only*** **[National, 1997]** e **[ANSI & IEEE, 1992]**.

Um **Controlador** é necessário quando o ***Talker*** e o ***Listener*** devem ser trocados. A função de controlador geralmente é manipulada por um computador

Durante a comunicação devem ser respeitadas as seguintes regras ou normas:

- Só um dispositivo pode ser ao mesmo tempo **Talker** e vários outros podem ser **Listeners**,

- A taxa à qual a informação é transmitida é automaticamente ajustada à velocidade do instrumento que processe mais lentamente a informação,
- É possível que um dispositivo seja ao mesmo tempo **Listener** e **Talker**,
- Em um sistema com vários **Controladores**, pelo menos um **Controlador** deve conter a função de controle do sistema, este é chamado **Controller-in *Charge*** (**CIC**). A função de controle pode ser passada do atual **CIC** a outro **Controlador**. Só o **Controlador** do sistema pode autoconverter-se em **CIC** **[Grimberg, 1990]** e **[ANSI & IEEE, 1992]**. Neste sistema o cartão GPIB da *National Instruments* (**PC-IIA**), é usualmente o **Controlador** do sistema.

3.4 Outras funções da interface

Alem das funções básicas mencionadas que mostram como em um sistema um dispositivo pode enviar, receber, e processar as mensagens, a interface tem outras funções que atuam de acordo com o protocolo específico definido nas recomendações do padrão IEEE - 488. Estas funções são: **[Grimberg, 1990]**, **[Hewlett, 1988]** e **[ANSI & IEEE, 1992]**.

Talker (**T**) ou ***Talker Estendido*** (**TE**), estas funções habilitam o dispositivo com a capacidade de enviar mensagens de dados sobre o barramento para outros dispositivos. Esta capacidade existe só quando o dispositivo está endereçado como **T**. A diferença entre estas duas alternativas da função é dada pela extensão do endereço. A função normal **T** usa só um byte de endereço e a função estendida, **TE**, utiliza dois bytes para o endereço.

Listener (**L**) ou ***Listener Estendido*** (**LE**), estas funções habilitam o dispositivo com a capacidade de receber mensagens de dados sobre o barramento proveniente de outros dispositivos. Esta capacidade existe só quando o dispositivo está endereçado como **L**. A diferença entre estas duas alternativas da função, está dada

também pela extensão do endereço. A função normal **L**, usa só um *byte* de endereço e a função estendida, **LE**, utiliza dois *bytes* para o endereço.

Sourse Handshake **(SH)**, esta função habilita o dispositivo com a capacidade para transferência de mensagens multi-linhas.

Acceptor Handshake **(AH)**, esta função habilita o dispositivo para a recepção de mensagens multi-linhas.

Remote/Local **(RL)**, esta função permite ao dispositivo selecionar entre duas fontes de entrada de informação. **Local,** a informação é procedente do painel frontal, e **Remoto,** a informação está chegando do barramento.

Service Request **(SR)**, esta função habilita o dispositivo para a requisição de serviço assincronamente a partir do controlador ativo do sistema.

Parallel Poll **(PP)**, esta função habilita o dispositivo poder ser unicamente identificado se este está solicitando serviço quando o controlador ativo está requisitando uma resposta. Esta capacidade difere do **SR** em que este requer o consentimento do controlador para realizar o **PP**.

Device Clear **(DC)**, esta função habilita ao dispositivo para ser inicializado num estado limpo. O efeito deste comando está em dependência do dispositivo

Device Triger **(DT)**, esta função habilita ao dispositivo para inicializar a operação sobre o barramento.

Controlador (C), esta função capacita a dispositivo para o envio de mensagens multi-linhas, tais como: endereços e comandos para todos os dispositivos que estão sobre o barramento. Também permite criar a capacidade de **PP**, para determinar quais dispositivos estão requisitando serviço.

3.5 Características e requerimentos da configuração

O padrão IEEE 488 fornece um sistema elétrico e mecânico para a interconexão de instrumentos, através de um cabo padrão **[ANSI & IEEE, 1992]**. Em cada

extremo o cabo tem um conector com uma distribuição de sinais, como mostra a Figura 3.1.

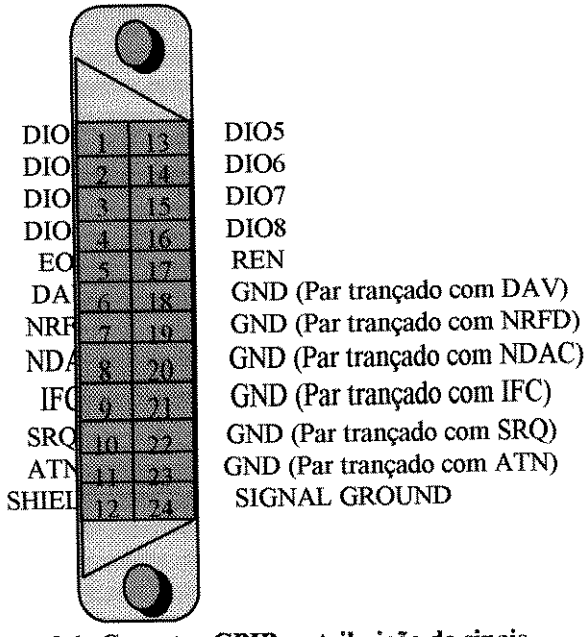


Figura 3.1: Conector GPIB e atribuição de sinais

Os dispositivos podem ser conectados através deste cabo segundo a configuração linear (Figura 3.2), estrela (Figura 3.3) ou uma combinação de ambas **[National, 1997]** e **[ANSI & IEEE, 1992].**

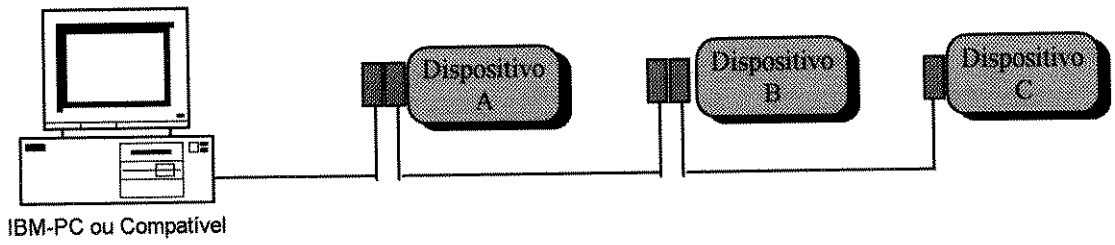


Figura 3.2: Configuração Linear

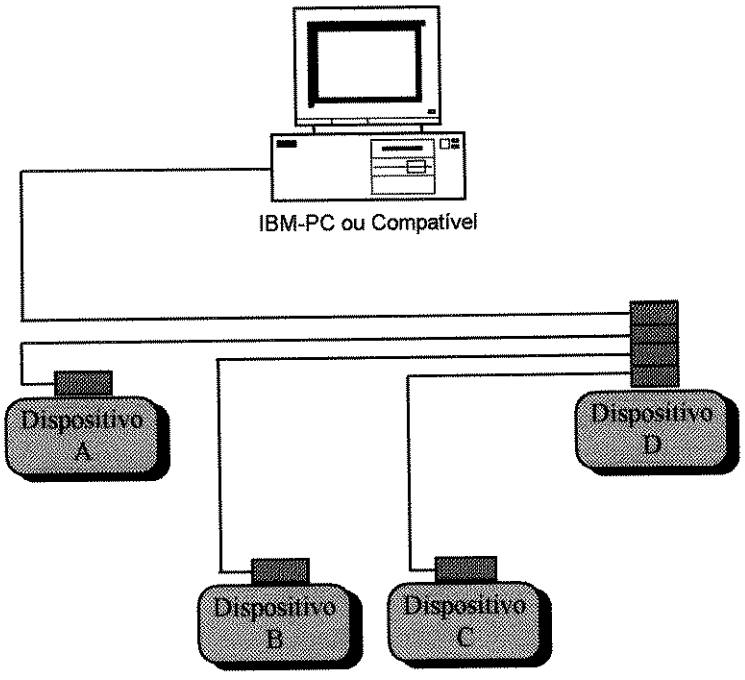


Figura 3.3: Configuração Estrela

Devido à alta taxa de transferência (**1 Mbyte/s**) para a qual a **GPIB** foi projetada, há uma limitação do número de dispositivos conectados ao barramento e da distância entre eles. Para uma operação normal devem ser considerados as seguintes restrições **[ANSI & IEEE, 1992]** e **[ANSI & IEEE, 1992]** :

- uma máxima separação de 4 metros entre dois dispositivos quaisquer e uma separação média de 2 metros entre os dispositivos e o barramento;
- o comprimento máximo total do cabo é de 20 metros.
- não mais de 15 equipamentos podem ser colocados em cada barramento.

Quando uma aplicação requer mais dispositivos e distâncias mais longas que as que estabelece o padrão, um expansor de barramento deve ser colocado, para garantir imunidade ao incremento do ruído e uma boa comunicação entre os dispositivos **[ANSI & IEEE, 1992]**.

3.6 Sinais e linhas

A interface do sistema **GPIB** está constituída por 16 linhas de sinais e 8 de retorno aterradas. Estas 16 linhas, são agrupadas da seguinte forma: 8 linhas de dados, 3 linhas de protocolo (*handshake*), e 5 linhas de gerenciamento da interface **[Grimberg, 1990]** e **[Hewlett, 1988]**, como mostra a Figura 3.4.

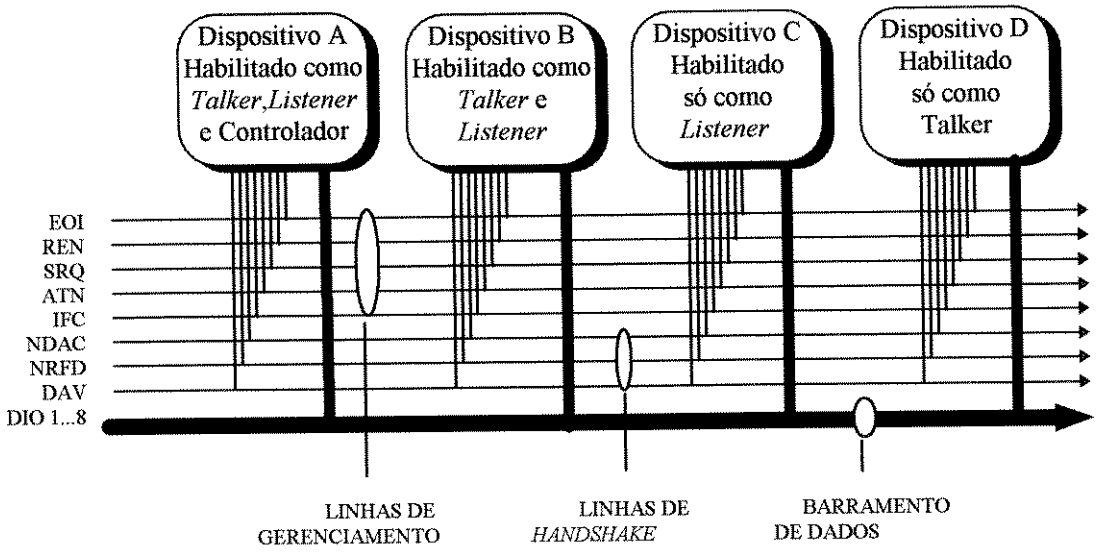


Figura 3.4:Linhas e Sinais da GPIB

Os sinais da interface utilizam lógica negativa para poder implementar um *wire-or* convencional com as linhas **NRFD** e **NDAC**, para fornecer o sinal de habilitação, e reduzir a sensibilidade ao ruído durante a habilitação.

A Tabela 3.1 mostra a relação entre os níveis lógicos.

Tabela 3.1 :Relação entre os níveis logicos e de tensão.

Nível Lógico	*Tensão Nominal*
"0"(**H**)	$\geq 2{,}0v$(***Alto***)
"1"(**L**)	$\leq 0{,}8v$(***Baixo***)

3.6.1 Linhas de dados (DIO 1-8)

É um barramento bidirecional usado para transferir informações de um dispositivo a outro sobre a interface. Os dados são transferidos utilizando qualquer código BCD, alfanumérico ou binário. Normalmente este é ASCII de 7 bits. Entretanto, outras técnicas podem ser utilizadas para codificar a informação sobre essas 8 linhas. A informação transferida inclui comandos de interfaces e endereçamento de dados aos dispositivos **[Hewlett, 1988]**.

Para assegurar a transferência integral, a técnica tem que ter as seguintes características **[ANSI & IEEE, 1992]**:

- a transferência de dados sobre o barramento é byte-serial e bit-paralelo e se realiza de forma assíncrona;
- a taxa máxima de transferência de dados é 1Mbyte/s e se ajusta automaticamente à velocidade do dispositivo mais lento;
- mais de um dispositivo pode receber dados ao mesmo tempo;
- todos os bytes são transferidos sob o protocolo de *handshake;*

- quando os comandos universais são enviados pelo barramento de dados, o dispositivo mais lento determina a taxa de transferência do comando. Neste caso todos os dispositivos utilizam o protocolo de *handshake;*
- a taxa de transferência de dados pode também ser ajustada pelo tempo que requer o instrumento para realizar sua leitura e o requerido pelo controlador para adquirir a informação.

O sistema tem dois modos de transferência: modo de dado e modo comando que podem ser selecionados através da linha de gerenciamento (**ATN**).

3.6.2 Linhas de *handshake*

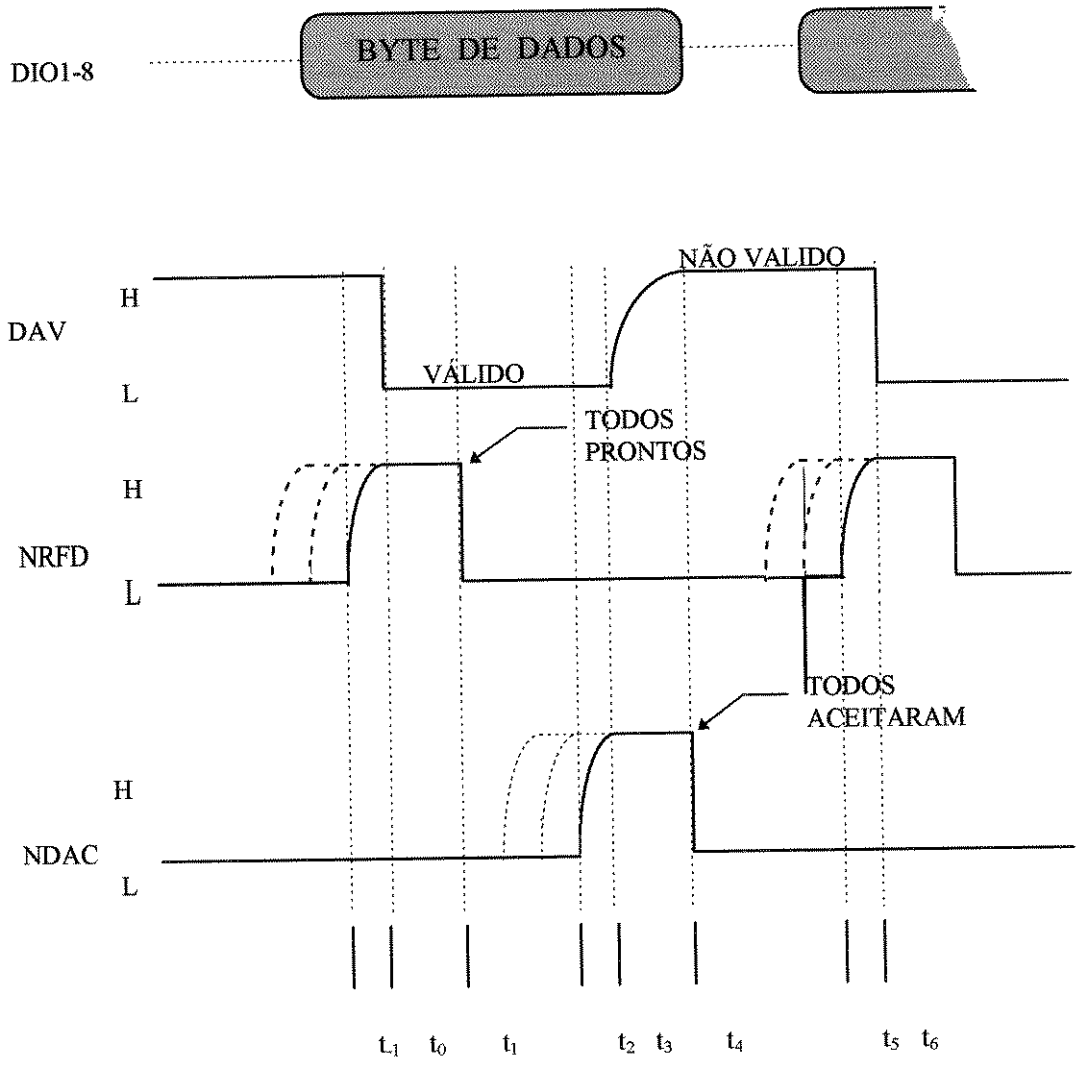
Estas 3 linhas controlam de forma assíncrona a transferência de bytes entre dispositivos **[Grimberg, 1990]** e **[ANSI & IEEE, 1992]**. Elas garantem que as mensagens sejam transmitidas e recebidas sem erro. São elas:

Data Valid (**DAV**): usada para indicar a condição de que as mensagens sobre a linha de dados estão estáveis (são validos) e podem ser aceitos

Not Ready For Data (**NRFD**): usada para indicar a condição de que todos os dispositivos estão prontos para aceitar dados. Um receptor de dados coloca essa linha no nível baixo para indicar que não está pronto para aceitar dados e em nível alto quando está pronto para receber. Contudo, a linha **NRFD** não será ativada até que todos dispositivos receptores estejam prontos para receber os dados.

Not Data Accepted (**NDAC**): é usada para indicar a condição de aceitação dos dados pelo dispositivo. O receptor coloca **NDAC** em baixo para indicar que este não recebeu os dados. Quando este aceita os dados, converte a linha **NDAC** a um nível alto. Entretanto, esta linha não irá a nível alto até que o receptor mais lento tenha recebido os dados.

As Figuras 3.5 e 3.6 ilustram a seqüência temporal do *handshake*.



t_{-1} : Todos os receptores estão prontos para receber. NRFD vai ao nível alto com o receptor mais lento.
t_0 : O transmissor coloca válida a linha DAV
t_1 : O primeiro receptor coloca a linha NRFD no nível baixo para indicar que ja não esta listo para receber um novo byte de dado.
t_2 : NDAC vai ao nível alto com o receptor mais lento indicando que todos os dados têm sido aceitos.
t_3 : O transmissor coloca DAV no nível alto indicando que o byte de dados já não é mais válido.
t_4 : O primeiro receptor coloca NDAC no nível baixo em preparação para o próximo ciclo.
t_5 : Volta a t_{-1} .

Figura 3.5: Temporização dos sinais do HANDSHAKE

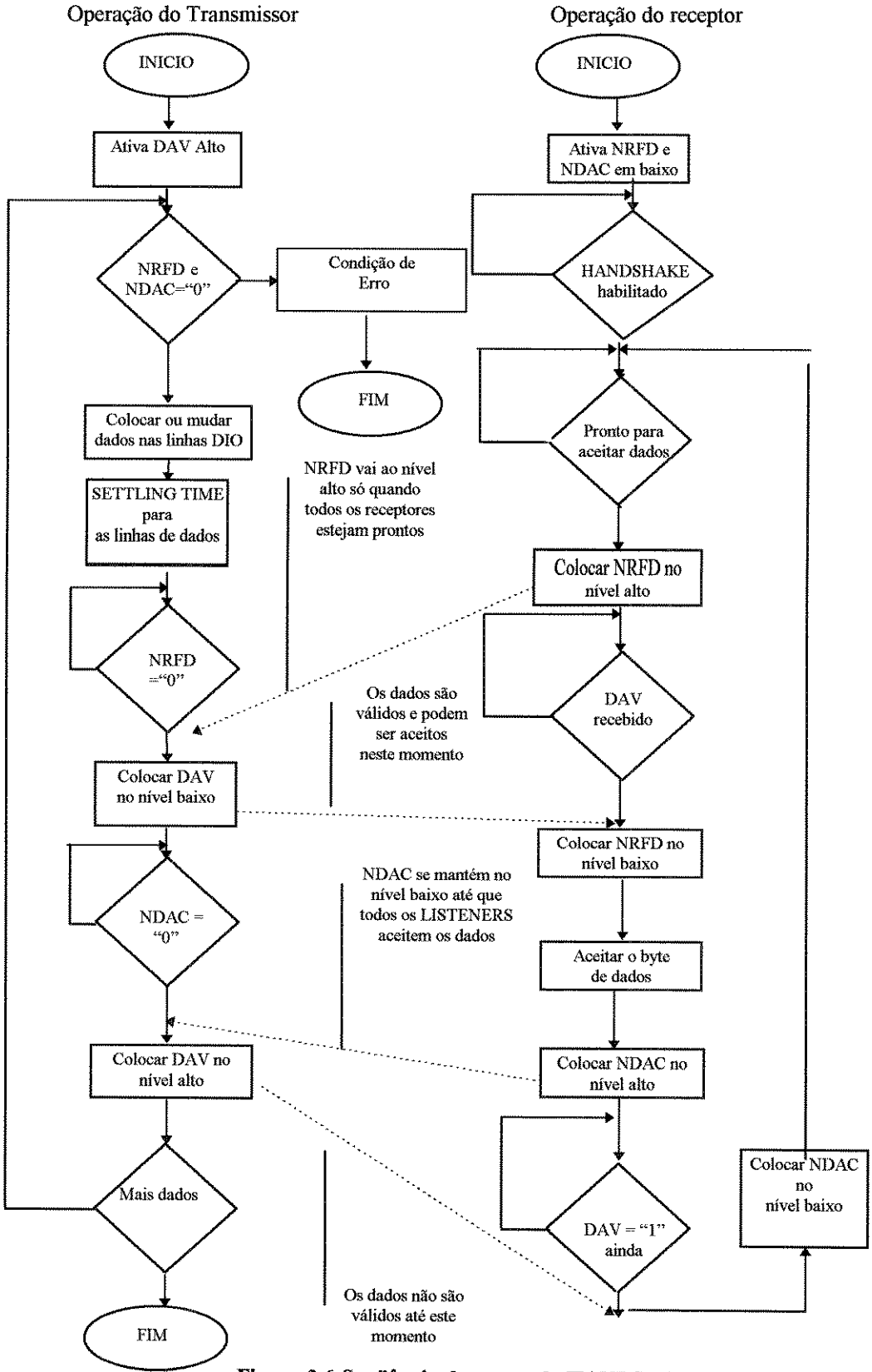


Figura 3.6:Seqüência de tempo do HANDSHAKE

3.6.3 Linhas de Gerenciamento da Interface

São 5 linhas usadas para dirigir e ordenar o fluxo de informação através da interface **[Grimberg, 1990]** e **[ANSI & IEEE, 1992]**.

Attention **(ATN)**: Todos os dispositivos devem monitorar esta linha ao mesmo tempo e responder dentro de 200 ns. Quando **ATN** esta no nível alto, a interface é colocada no modo de dado, onde o *talker* ativo envia dados para todos os *listener*. E quando **ATN** está no nível baixo, a interface fica no modo de comando, onde todos os dispositivos aceitam os dados que vêm através da linha de dados os quais são interpretados como comandos ou endereços

Interface Clear **(IFC)**: É utilizada só pelo controlador do sistema para inibir as operações atuais de comunicação no barramento. Todos os dispositivos devem sempre monitorar **IFC** e responder dentro de 100 μs.

Remote Enable **(REN)**: É utilizada só pelo controlador do sistema para habilitar os dispositivos no modo de programação remoto. Quando **REN** está no nível baixo todos os *listener* capazes de operar remotamente se colocam no modo remoto quando são endereçados. Quando **REN** vai para o nível alto todos os dispositivos voltam a operação local. Todos os dispositivos capazes de trabalhar em ambos modos (remoto e local) devem monitorar a linha **REN** o tempo todo e dar resposta dentro de 100 μs.

Service Request **(SRQ)**: É usada por um ou mais dispositivos para indicar que precisam ser atendidos, e pode ser utilizada também para interromper uma seqüência atual de eventos. Tipicamente essa linha indica que o dado está pronto para ser transmitido e/ou um error condicional existe (ex.: erro de sintaxe).

End or Indentify **(EOI)**: Quando **ATN** está no nível baixo a linha **EOI** é utilizada pelo controlador para executar um **PP** (descrito adiante). Quando **ATN** está no nível alto, a linha **EOI** é utilizada pelo *talker* ativo para indicar o último byte da mensagem de dados.

3.7 Endereçamento

É usado para selecionar qual dispositivo funcionará como *talker* e qual como *listener* em cada momento. Os dispositivos endereçados pelo controlador são ativados no modo comando. O endereço do dispositivo é usualmente colocado na fábrica e pode ser modificado durante a configuração do sistema, por uma chave de endereços, um *jumper*, ou por um acesso no painel frontal. Essa chave está tipicamente localizada na parte externa do painel traseiro do equipamento. O valor decimal equivalente dos cinco bits da chave determina o endereço do dispositivo na interface e pode variar de 0 a 30. Os bits 6 e 7 (DI06-DI07) são utilizados para distinguir entre um dispositivo endereçado como *talker* e um como *listener* **[Grimberg, 1990]**, **[Hewlett, 1988]** e **[Intel, 1990]**. Como se mostra na Tabela 3.2.

Tabela 3.2: Bit 6 e bit 7 do endereço

BIT	*BIT 7*	*BIT 6*
Talker	1	0
Listener	0	1

Os códigos correspondentes ao endereço 31, são usados para colocar todos os dispositivos em ***untalk*** ("-") ou ***unlisten*** ("?"). Portanto, os dispositivos com o endereço 31 são ilegais.

Uma chave adicional é usada em alguns dispositivos o que permite colocá-los nos modos *talk-only* ou *listen-only* ou implementar funções do tipo auto-teste e auto-calibração **[Hewlett, 1988]**.

O dispositivo com o GPIB dá a possibilidade de formar um endereçamento estendido pela adição de um segundo caracter no endereço principal. Isso pode ser usado para selecionar partes de um instrumento ou um grupo de dispositivos idênticos **[Grimberg, 1990]**.

Também é possível realizar um múltiplo endereçamento, em dispositivos que têm múltiplas capacidades que podem ser tratadas individualmente. **[Grimberg, 1990]**.

3.8 Comandos do Barramento

O padrão IEEE - 488 define comandos especiais para a programação de dispositivos desde o controlador ativo. Estes comandos controlam a operação do barramento e não a operação dos dispositivos. Eles podem dividir-se em três grupos **[ANSI & IEEE, 1992]** e **[ANSI & IEEE, 1992]**:

•**Comandos Universais**; que determinam que todos os dispositivos que estejam colocados sobre o barramento, endereçados ou não, realizem operações específicas da interface. Estes podem ser **Multi-linhas** e **Uni-linhas**.

•**Comandos de Endereçamento**, só os dispositivos que estejam endereçados podem responder a eles. Alguns destes comandos são para os dispositivos endereçados como *listeners* e outros para os que estão endereçados como *talkers.*

•**Comandos Secundários**, são sempre usados para estender um endereço, um Comando Universal ou Comando de Endereçamento.

3.8.1 Comandos Universais Multi-linha

Device Clear **(DCL)**: Determina que todos os dispositivos, com capacidade de responder a esta função, retornem a um estado predefinido.

Local Lockout **(LLO)**: Este comando junto com a ativação da linha **REN**, desabilita a chave de **RL** do panei frontal e coloca todos os dispositivos no modo local.

Serial Poll Enable **(SPE)**: Habilita o modo **SP** sobre o barramento, de forma que uma seqüência de **SP** possa começar.

Serial Poll Disable **(SPD)**: Desativa o modo **SP** sobre o barramento e pode ser usado para finalizar uma seqüência de **SP.**

Parallel Poll Unconfigure **(PPU)**: Coloca todos os dispositivos com a capacidade de **PP** no estado predeterminado onde não respondem ao **PP** .

Unlisten **(UNL)**: Desativa todos os *listener* que estão sobre o barramento

Untalk **(UNT)**: Desativa o *talker* ativo. Um *talker* pode também ser desativado automaticamente pela transmissão de outro endereço de *talker* sobre o barramento.

3.8.2 Comandos Universais Uni-linha

Os quatros comandos uni-linhas são : **IFC**, **REN**, **ATN** e **IDY** (*identify*). Os três primeiros já foram descritos.

$IDY = EOI \wedge ATN$

Esta linha é usada pelo *talker* no modo dado (**ATN** = ***Alto***) para indicar o fim de uma transferência múltipla de *byte*, e no modo comando (**ATN** = ***Baixo***) é usada pelo o controlador para a execução de um **PP.**

3.8.3 Comandos de Endereçamento

Group Execute Trigger **(GET)**: O controlador pode enviar este comando para inicializar simultaneamente uma ação preprogramada para um ou mais dispositivos os quais devem ter sido endereçados como *listener* previamente.

Selective Device Clear **(SDC:** Os dispositivos endereçados como *listener* podem retornar a um estado predefinido com este comando.

Go To Local **(GTL)**: Este comando faz com que os dispositivos atualmente endereçados retornem ao controle local (saída do estado remoto). O dispositivo sairá do estado remoto quando for endereçado como *listener* novamente e **REN** no nível baixo

Parallel Poll Configure **(PPC)**: Este comando permite que algumas linhas **DIO** possam ser utilizadas pelos dispositivos no barramentos com o propósito de dar resposta a um **PP;** só pode ser enviado para dispositivos que tenham sido endereçados como *listener.*

Take Control **(TCT)**: Este comando faz com que o controlador ativo transfira o controle a outro dispositivo. O novo controlador deve ser endereçado como *talker* previamente.

3.8.4 Comandos Secundários

Parallel Poll Enable **(PPE)**: Este comando secundário configura os dispositivos que receberam o comando **PPC,** a responder ao **PP** em uma linha **DIO** particular do barramento. Alguns dispositivos têm a possibilidade de implementar de forma local esta configuração através do uso de *jumpers*, chaves, etc.

Parallel Poll Disable **(PPD)**: Este comando desabilita os dispositivos que receberam o comando **PPC** em resposta ao **PP**.

3.9 Polling

3.9.1 Serial Polling

É uma seqüência que habilita um controlador a ler o *status* do dispositivo. Usando **SP** o controlador pode determinar se o dispositivo ou um grupo deles solicita atendimento.

Os equipamentos que sejam atendidos retornarão o *byte* de *status* para o controlador para indicar o seu *status*. O controlador monitora seqüencialmente cada dispositivo individualmente na interface (envia um **SPE** e endereça cada um seqüencialmente como *talker*) e analisa cada *byte* de *status* por vez. Portanto, este

processo pode ser demorado em grandes sistemas. Entretanto, o *byte* de *status* proporciona a natureza da solicitação assim como identifica quem fez a solicitação.

Deve-se (embora não seja imprescindível) checar todos os dispositivos para achar todos os que fizeram solicitação. Deve-se lembrar de enviar os comandos **SPD** e **UNT** quando termina a tarefa. A maioria dos controladores atualmente já fazem automaticamente estes comandos **[Hewlett, 1988]**.

Qualquer dos equipamentos integrantes do barramento pode também solicitar os serviços do **SP** ativando a linha **SRQ** (Baixo) **[Hewlett, 1988]**.

3.9.2 Parallel Poll

É uma operação iniciada pelo controlador para obter informação de vários dispositivos simultâneamente. Quando o controlador inicia um **PP**, cada dispositivo retorna o *bit* de *status* via uma das linhas de **DIO**. A designação das linhas **DIO** aos dispositivo são feitas por chaves, *jumpers*, ou pelo controlador usando a seqüência **PPC**. Os dispositivos respondem individualmente, cada um em uma linha **DIO** separada, ou coletivamente em uma única linha **DIO** ou uma combinação de ambas. Quando respondem coletivamente, o resultado é um *AND* lógico (*true high*) ou *OR* (*true low*) dos grupos de *bits* de *status*. Os dispositivos configurados devem responder ao **PP** (ocorre verificação simultânea de **ATN** e **EOI**) dentro de 200ns. O Controlador pode então ler os resultados do *Poll* 2 μs depois. O **PP** é sempre usado pelos computadores para checar o *status* de uma ação, ou seja, quais periféricos estão prontos para dados, enviando ou recebendo dados. Através desta informação os tempos de espera são reduzidos e a banda de freqüência do sistema é usada mais eficientemente **[Hewlett, 1988]**.

O controlador pode verificar simultaneamente 8 equipamentos cada um dos quais tem associado um bit do barramento de dado. Este bit é ativado pelo equipamento que demanda o serviço quando o controlador envia o comando **PPE**.

Neste caso, o controlador deve procurar periodicamente os dispositivos que solicitaram serviço para conhecer se ocorreu alguma solicitação **[Grimberg, 1990]**.

Capítulo 4

Cartão Controlador PC-IIA da National Instruments.

Neste capítulo forneceremos as informações gerais sobre o cartão controlador GPIB **PC-IIA** da *National Instruments* que foi usado no projeto. O cartão é mostrado na Figura 4.1 **[Grant, 1997]**.

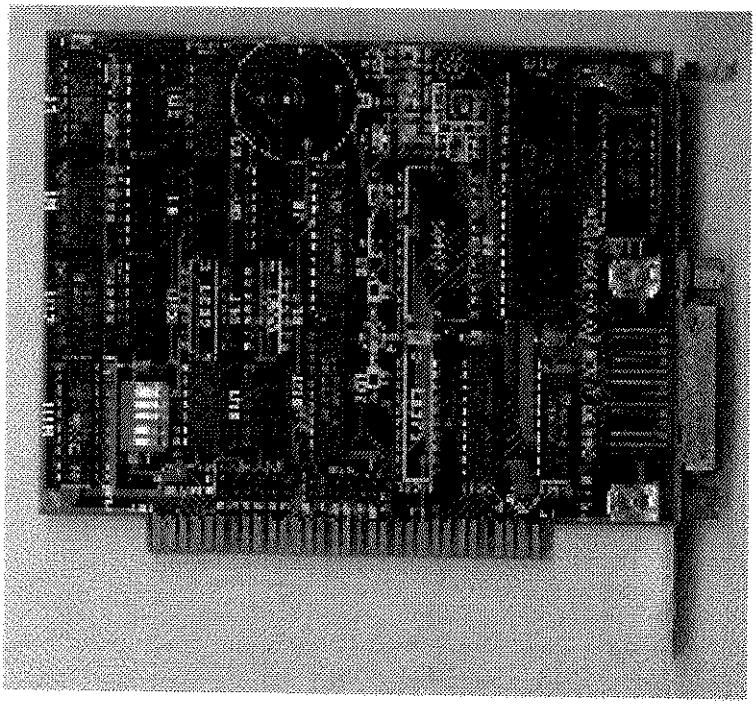


Figura 4.1:Cartão PC-IIA da *Natinal Instruments*

4.1 Descrição Geral

O cartão **PC-IIA** é uma interface GPIB para microcomputadores compatíveis com o IBM-PC, que utilizam o sistema operacional MS-DOS (PC-DOS).

Com a utilização desta interface conseguimos implementar de forma prática um sistema GPIB de teste e medição usando o IBM-PC como controlador de instrumentos e periféricos **[ANSI & IEEE, 1987]**.

4.1.1 Características do *software*

O *software* do **PC-IIA** é um *handler* que fornece amplo suporte para todas as funções e capacidades do IEEE - 488, conforme segue **[Holley, 1997]** e **[Grant, 1997]**:

- **Sistema Controlador** para inicializar o GPIB;
- **Controlador Ativo** para enviar comandos;
- ***Talker*** para escrever mensagens;
- ***Listener*** para ler mensagens;
- **Remoto/Local** para configurar dispositivos programáveis remotamente;
- ***Serial Poll*** e ***Parallel Poll*** para atender dispositivos no barramento;
- **Transferir** a função de controlador para outro dispositivo no barramento;
- **Detecção** de erro de *handshake* do GPIB.

O *handler* possui outras facilidades que melhoram seu desempenho, incluindo:

- **Espera** por programa de qualquer um dos 12 eventos selecionados;
- Operação via **DMA** ou sem **DMA;**
- **Time out** ajustável para evitar paradas infinitas.

4.1.2 Características do *Hardware*

As características básicas do *hardware* do cartão **PC-IIA** são as seguintes **[Holley, 1997]** e **[Grant, 1997]**:

- **Pequenas dimensões**, permitindo sua instalação em qualquer *slot* do computador;
- **Oito portas** para endereçamento de programas;
- **Capacidade** de transferência via **DMA** usando o controlador de DMA do computador;
- Habilitação/desabilitação de **DMA** dinâmica e automática;
- Habilitação /desabilitação de **interrupção** dinâmica e automática;
- Seleção dinâmica do **Controlador do Sistema**;
- Canal de DMA selecionável pelo usuário;
- Linha de interrupção selecionável pelo usuário.

O cartão **PC-IIA** possui dois conectores, sendo um para interligação com o barramento interno do computador e outro, padrão IEEE-488, para conexão com o GPIB. A alimentação do **PC-IIA** (+5Vdc) é extraída do próprio computador e o consumo típico é de aproximadamente 500 mA. Todas as linhas de entrada representam apenas uma carga TTL-LS para o computador.

4.2 Endereçamento E/S

O **PC-IIA** é endereçado como dispositivo de **E/S** e decodifica as dez linhas menos significativas (A0-A9) do barramento do computador. Ele ocupa oito endereços consecutivos de E/S, como mostramos na Tabela 4.1.

O endereço base é colocado pelo fabricante em **2B8H** através da configuração das chaves tipo *dip* .

Tabela 4.1 :Endereço dos registros do PC-IIA

Endereço	*Registros de leitura*	*Registros de escritura*
Base	*Data in*	*Command/Data out*
Base +1	*Interrupt Status 1*	*Interrupt Mask 1*
Base +2	*Interrupt Status 2*	*Interrupt Mask 2*
Base +3	*Serial Poll Status*	*Serial Poll Mode*
Base +4	*Address Status*	*Address Mode*
Base +5	*Command Pass Through*	*Auxiliary Mode*
Base +6	*Address 0*	*Address 0/1*
Base +7	*Address 1*	*End of String*

4.3 Interrupções

Permite ao usuário selecionar, através de *jumpers*, uma das seis linhas de interrupção vetorada (IRQ2-IRQ7) do computador. A arbitração e vetoração da interrupção é realizada através do circuito integrado 8259A (Controlador de Interrupções) localizado no computador **[Holley, 1997]**.

Os eventos do GPIB que a critério do programador podem causar interrupções são os seguintes:

- *Address Pass Through*
- *Address Status Change*
- *Command Out*
- *Command Pass Through*
- *Data In*
- *Data Out*
- *Device Clear*
- *Device Trigger*

- *DMA Terminal Count*
- *End of Data Block*
- *Error*
- *Remote Change*
- *Lockout Change*
- *Service Request Input*

4.4 Acesso Direto à Memória (DMA)

O **PC-IIA** permite ao usuário selecionar, através de *jumpers*, um dos três pares de linhas de **DMA** (DRQ1-DRQ3,DACK1-DACK3) disponíveis no computador. A arbitração de transferências por DMA é realizada através de circuito integrado 8237A (Controlador de DMA) localizado no computador **[Holley, 1997]**.

Os eventos que podem, a critério do programador, causar transferências de dados por DMA são:

- *Data In*
- *Data Out*

4.5 Transferência de Dados e Comandos

O **PC-IIA** pode receber/transmitir dados do/ao GPIB através de DMA, interrupções, ou *polling*.

Utilizando DMA, o **PC-IIA** é capaz de transferir dados até **300** Kbytes por segundo. A taxa real de transferência em um sistema é dependente tanto da velocidade de transferência do dispositivo mais lento acoplado ao barramento como do número de *bytes* transferidos por operação. Em vista disto, a taxa de

transferência entre o **PC-IIA** e o GPIB pode variar sensivelmente de um sistema para outro.

Utilizando *polling* e/ou interrupções, a máxima velocidade de transferência de dados depende sobremaneira da velocidade de execução do programa de controle **[Holley, 1997]**, **[ANSI & IEEE, 1987]** e **[Grant, 1997]**.

4.6 Descrição Funcional

O cartão pode ser visto, de maneira simplificada, como um conversor de barramento que converte mensagens e sinais presentes no barramento do computador em mensagens e sinais apropriados ao barramento do GPIB. A Figura 4.2, mostra o diagrama em blocos do **PC-IIA**, constituído pelos seguintes blocos funcionais **[Holley, 1997]** e **[Grant, 1997]**:

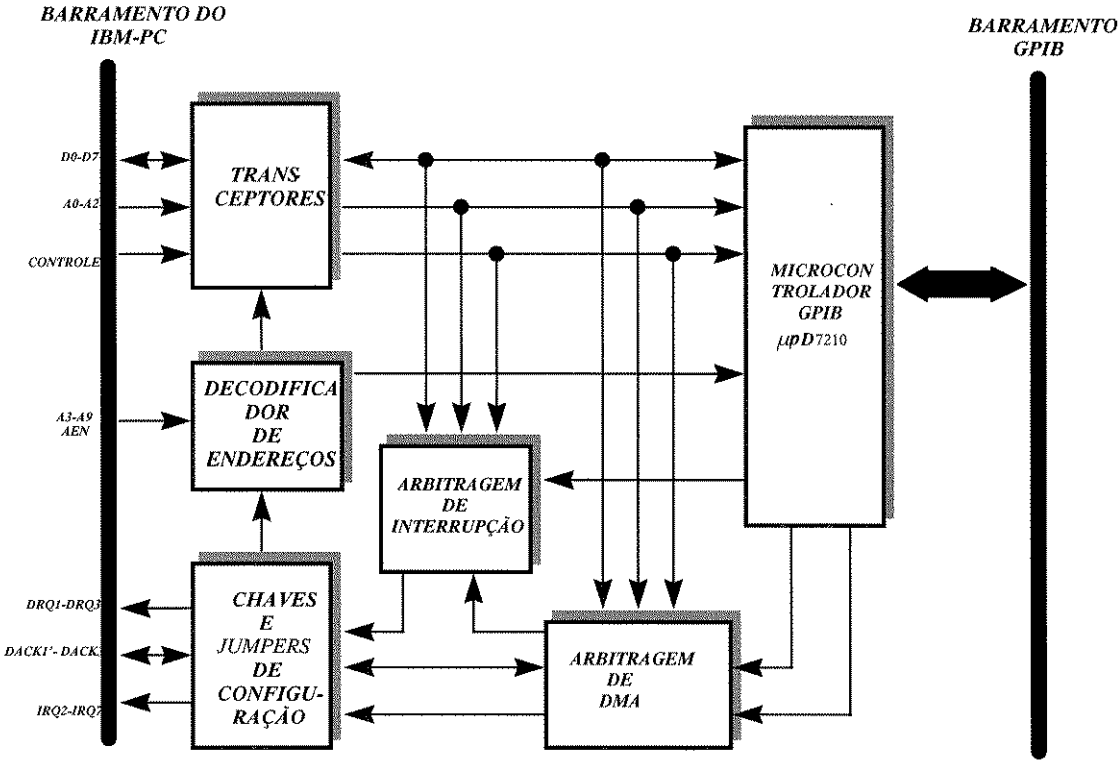


Figura 4.2: Diagrama em blocos do PC-IIA

Decodificador de endereços

Monitora as linhas de endereço do barramento do computador, de modo a reconhecer endereços válidos e permitir o acesso do computador aos registros internos do microcontrolador GPIB.

Transceptores

Realizam a função de acoplamento das linhas de endereço, dados e controle do barramento do computador ao barramento interno do **PC-IIA**. Garantem, também a minimização do carregamento do computador.

Arbitragem de DMA

Reconhece a habilitação ou desabilitação de operações por DMA e permite ou não a circulação dos sinais correspondentes entre o **PC-IIA** e o barramento do computador. O término da última transferência em uma operação de DMA pode causar, a critério do programador, uma interrupção.

Arbitragem de interrupções

Reconhece a habilitação ou desabilitação de interrupções e permite ou não a circulação dos sinais correspondentes entre o **PC-IIA** e o barramento do computador .

Chaves e Jumpers de configuração

Determinam, de acordo à configuração escolhida, o endereço base de acesso ao cartão, além do canal de DMA e/ou a linha de interrupção utilizada (quando for o caso).

Microcontrolador GPIB (μpD 7210)

Circuito integrado, que implementa virtualmente todas as funções do IEEE-488 para interagir com os dispositivos do GPIB. É composto por 21 registradores internos que são utilizados para configurar, controlar e monitorar as funções de interface e para a transferência do e para a GPIB, de comandos e dados.

O computador, equipado com o cartão **PC-IIA** , cabo e *handler software* pode ser um *Talker*, *Listener* ou Controlador. Oito linhas de *handshake* provêm um protocolo para coordenar a transferência de dados. Cinco linhas de gerenciamento desempenham uma variedade de funções de controle do barramento.

O **PC-IIA** tem capacidade completa de ***Source*** e ***Acceptor Handshake***. Pode operar como ***Talker*** básico ou ***Talker*** estendido e pode responder a um **Serial *Poll***. Ele fica desendereçado para ***Talker***, quando recebe seu endereço de ***Listen***. A interface pode operar como ***Listener*** básico ou ***Listener*** estendido. Ela fica sem endereçamento para ***Listen*** quando recebe seu endereço de ***Talk***. A **PC-IIA** tem capacidade plena para requerer serviço de outro controlador. A habilidade de colocar o **PC-IIA** em modo local está incluída, mas a interpretação de modo local versos remoto depende do software. A capacidade total de ***Parallel Poll*** está incluída na interface. A capacidade de ***Device Clear*** e ***Trigger*** está incluída na interface, mas a interpretação depende do *software*. Todas as funções do **Controlador**, conforme especificado pela norma **IEEE-488**, estão incluídas no **PC-IIA**. Estas incluem a capacidade de:

- Ser programado para ser o **Controlador do Sistema** com os objetivos de inicializar as interfaces GPIB de outros dispositivos e colocar esses dispositivos em modo remoto ou local (pode ser programado para não ser o **Controlador do Sistema**, de modo que o computador possa ser usado num GPIB com outro controlador que é o **Controlador do Sistema**).
- Enviar comandos ou mensagens **multi-linha** de interface para outros dispositivos
- Detectar quando outros dispositivos estão requerendo serviço e conduzir ***poll*** seriais ou paralelos

- Passar o controle para outros controladores e receber o controle dos mesmos
- Ir para ***standby*** para permitir que o ***Talker*** endereçado envie dados para os ***Listeners*** endereçados, e que tome o controle outra vez, síncrona o assincronamente **[Grant, 1997]** e **[ANSI & IEEE, 1992]**.

4.7 Configuração e instalação

4.7.1 Seleção do endereço de E/S

O endereço base de E/S para o **PC-IIA** é determinado pelo *dip-switch* U14. As chaves vêm posicionadas, de fábricas para o endereço **2B8H** de E/S. Este é o endereço *default* utilizado pelos pacotes de software para o **PC-IIA** . Com esta configuração, o **PC-IIA** ocupará a faixa de endereçamento de E/S de **2B8H** a **2BFH [Holley, 1997]** e **[Tektronix, 1987]**.

Se uma outra interface instalada no IBM-PC ocupar esta faixa de endereçamento, será necessário mudar o endereço base do **PC-IIA** ou do outro dispositivo. Caso se decida mudar o endereço do **PC-IIA**, é necessário não só mudá-lo no *hardware*, como também alterá-lo no *software*.

Cada chave do *dip-switch* corresponde a uma das linhas de endereçamento de **A9** a **A3**. Deve-se colocar a chave na posição *open* para selecionar o valor binário 1 para o *bit* correspondente de endereço.

Tabela 4.2:Distribução das chaves do *dip-switch*

1	*2*	*3*	*4*	*5*	*6*	*7*	*Chave*
A9	A8	A7	A6	A5	A4	A3	Linha de Endereço correspondente

Os 3bits menos significativos de endereço **(A2-A0)** são usados internamente pelo **PC-IIA** para selecionar os próprios registros.

Capítulo 5

Sistema Automatizado com Interface IEEE -488 para a Calibração de Instrumentos

Neste capítulo realizaremos uma descrição das principais caraterísticas, estrutura e funções do sistema projetado. Mostra-se o conjunto de funções do sistema operacional da rede **GPIBN*et*** assim como proposta a metodologia para a realização de programas aplicativos a partir deste *software*.

5.1 Estrutura Geral do Sistema

Os elementos que formam o sistema estão interconectados em rede sob o cartão controlador **PC-IIA,** o qual é colocado dentro de um microcomputador **IBM-PC**. Estes elementos podem ser **fixos** e **variáveis**. Os **fixos** são aqueles elementos que não podem faltar no sistema independentemente dos procedimentos de calibração a serem aplicados. A presença dos elementos **variáveis** depende das funções, instrumentos e tipos de tarefas de calibração que serão realizadas.

Para conseguir uma maior organização do trabalho de aferição, calibração e teste tanto do ponto de vista do *hardware* como do *software*, o sistema dá a opção de poder ser subdividido em **sub-redes**, permitindo desta forma concentrar em pequenos grupos equipamentos do mesmo tipo ou mesmo fabricante, ou ainda podendo-se agrupar a partir de qualquer outra caraterística, que seja de interesse do pessoal técnico do laboratório.

A construção destas **sub-redes** se faz a partir da incorporação ao sistema de outros cartões PC-IIA formando, deste modo, um sistema **GPIB multicartão** (*multiboard)* **[Holley, 1997].** A Figura 5.1 mostra uma configuração geral do sistema projetado, onde o cartão **GPIB0** interconecta-se à **sub-rede 1** de calibração de osciloscópios e o cartão **GPIB1** interconecta à **sub-rede 2**, de calibração de voltímetros digitais no sistema. Cada um destes cartões é chamado de **cartão de acesso**, porque são usados

para acessar automaticamente através das funções de alto nível aos dispositivos associados a eles.

Os cartões, computador *plotter* e impressora constituem os elementos **fixos** da rede. Os demais elementos tais, como calibradores, geradores, voltímetros, osciloscópios, etc, são os **variáveis**. Cada cartão admite como máximo quinze dispositivos.

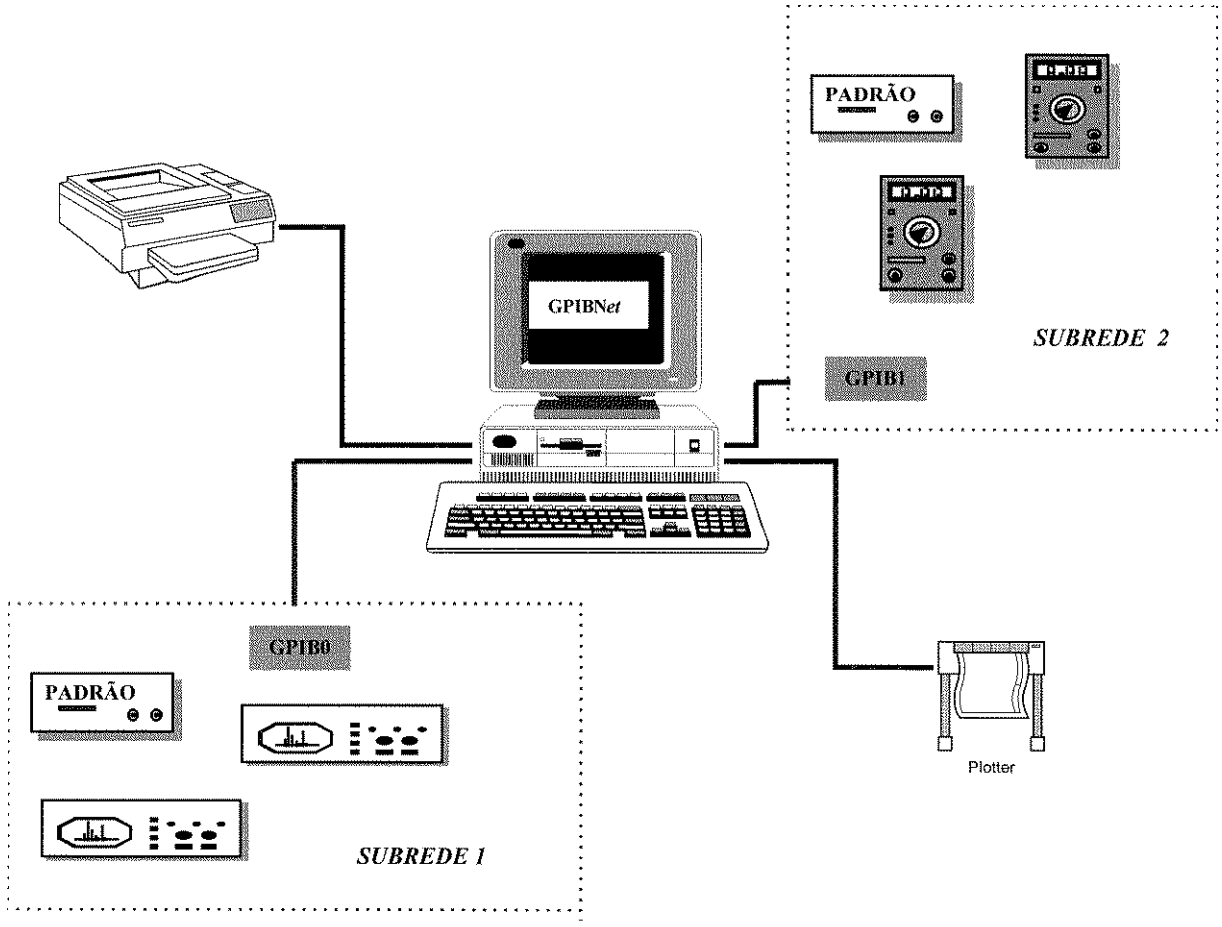


Figura 5.1:Sistema GPIB Multicartão projetado

Na Figura 5.2, mostra-se o sistema trabalhando com um **gerador de funções** e um **frequencímetro** da *National Instruments*, um ***plotter*** e um **voltímetro** da *Hewlett Packard* e um **analisador**, um **calibrador** e dois **osciloscópios** da *Tektronix*. A Figura 5.3 mostra o sistema *Fluke* que pode ser acoplado como uma sub-rede ao sistema . A Figura 5.4 mostra o conjunto *Valha Scientific* incorporado como uma sub-rede para trabalhar junto ao grupo de instrumentos mostrado na Figura 5.2.

O **GPIBN*et*** permite também a comunicação entre instrumentos que se encontrem em diferentes **sub-redes** sempre que a operação que se esta realizando o precise.

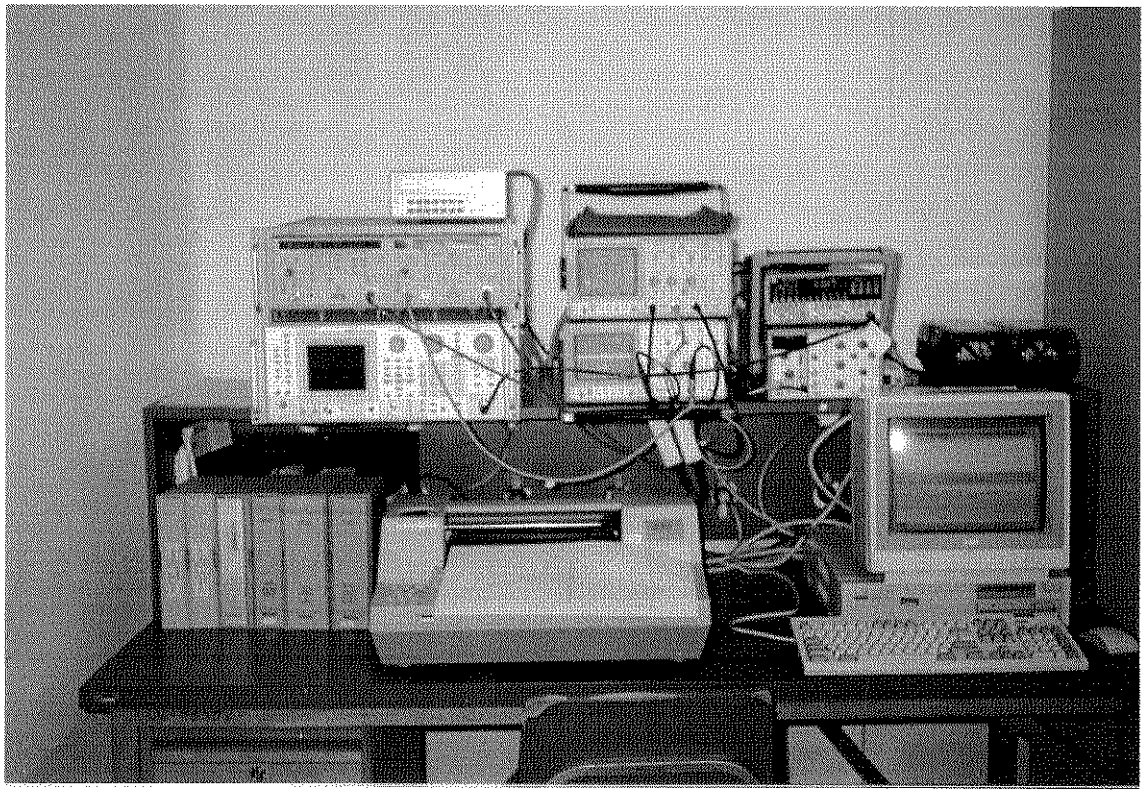


Figura 5.2:Sistema trabalhando com instrumento de diferentes fabricantes.

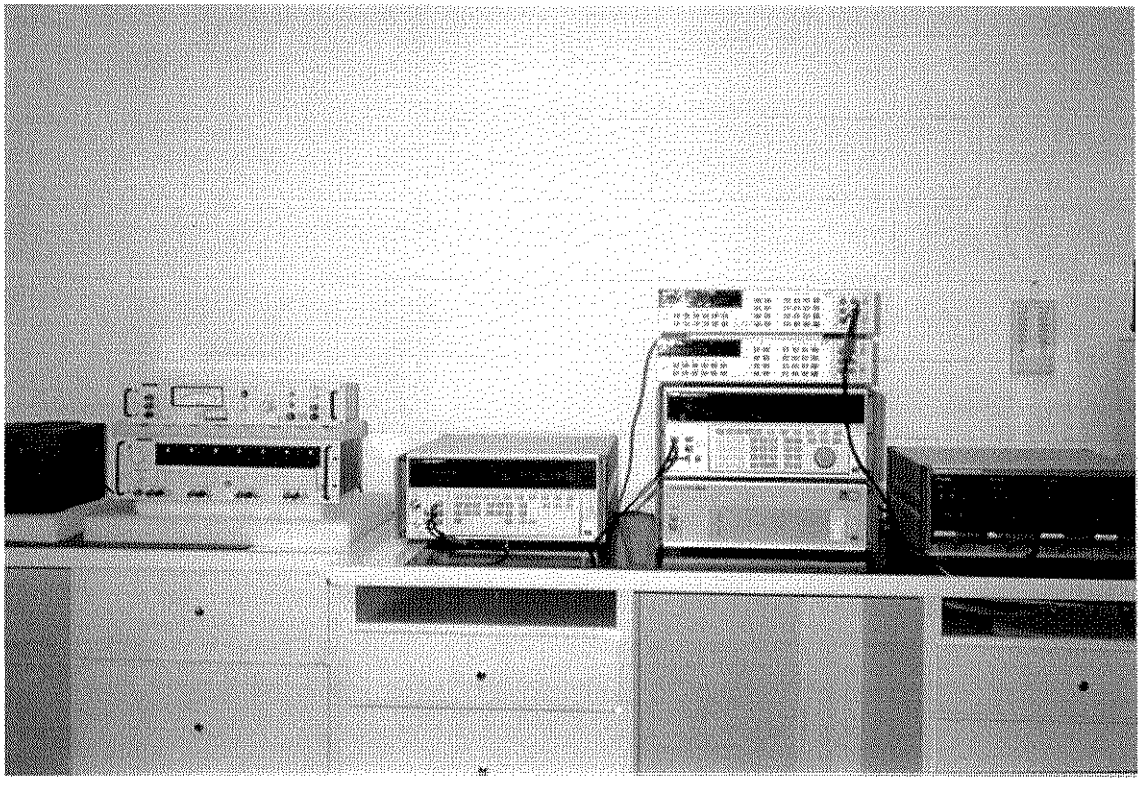


Figura 5.3:Sistema *Fluke* que pode ser acoplado como uma subrede ao sistema automatizado.

Figura 5.4:Sistema trabalhando com o conjunto *Valha Scientific* como uma subrede.

5.2 GPIBNet, O Sistema Operacional da Rede

GPIBN*et* é o *software* que controla toda a operação do sistema e de modo geral **[Campell, 1996] e [Shay, 1996]**:

- Isola os programas dos detalhes específicos de *hardware;*
- Permite que programas independentes cooperem periodicamente e compartilhem informações;
- Responde aos erros ou a solicitações do usuário;
- Impõe um escalonamento entre programas que solicitam recursos;
- Controla o fluxo de dados entre os componentes da rede;
- Permite que os programas armazenem e obtenham as informações.

O **GPIBN*et*** foi desenvolvido em linguagem '**C**' permitindo ao programador concentrar-se no código relativo aos instrumentos, sendo transparente o controle e a comunicação da rede, mostrando uma grande flexibilidade na programação e adição de novos instrumentos, sejam totalmente programáveis ou não.

Os comandos podem ser tomados a partir de um arquivo texto previamente editado ou diretamente da linha de comando do sistema; o que permite a operação em **modo programado** (arquivo) ou **modo interativo** (teclado).

Outra caraterística do sistema é a redução do tempo de implementação e manutenção de um programa a partir das instruções de um novo instrumento introduzido na rede, o que se deve ao fato de o sistema oferecer uma interface simples para a familiarização do programador com o novo conjunto de instruções. O novo sistema contrasta com os métodos anteriormente aplicados, onde para realizar a incorporação de um novo equipamento, ou *upgrade* de um equipamento já existente na rede, era necessário um conhecimento profundo das funções *handler* do **PC-IIA**.

5.2.1 Funções do GPIBNet

Estas funções são as que permitem o controle e comunicação entre todos os elementos que formam o sistema com o GPIB. Com o **GPIBN*et*** e o conjunto de comandos de cada instrumento, o envio, transformação e processamento dos dados para todos os elementos da rede podem ser realizados no **modo interativo,** diretamente do teclado, ou no **modo programado,** através de um arquivo previamente editado.

Para o trabalho no **modo interativo** as funções foram feitas de forma que o programador tenha uma ajuda imediata que indique como usá-las corretamente na programação dos dispositivos.

No **modo programado,** podem ser usados arquivos previamente editados com as seqüências e a ordem corretas de todos os passos das operações que se deseja sejam executadas pelo sistema, que posteriormente serão incorporadas aos **programas aplicativos** feitos sob a sintaxe das funções do **GPIBN*et.*** Estes arquivos podem ser tratados como módulos de *software* reutilizáveis o que simplifica a estruturação de aplicações complexas, alem de reduzir o tempo de implementação e manutenção dos programas. Os módulos podem ser usados em qualquer aplicação.

5.2.1.1 Funções

!# [String]

Assume que **[String]**,é um comentário e ignora a linha.

Exemplo:

!# Isto é um comentário.....

!? [Comando]

Envia pergunta para dispositivo e aguarda resposta. É necessária para dispositivos que não contenham '?'no **[Comando].**

Exemplo:

!# Pergunta para conhecer a identificação do *plotter* **Hp-gl**.

!? od

! ` ` [DosCmd]

Interpreta o *string* que segue ao espaço como um comando para o DOS. Envia o comando **[DosCmd]** para execução via DOS.

Exemplo:

! Edit

! dir a:

!ENd

Sinaliza fim de arquivo de comandos.

Obs: As duas primeiras letras da função, estão em maiúscula, para representar que apenas estas são necessárias para a executação da instrução. Esta regra se mantém para outras funções .

![End. Disp.]

Direciona o dispositivo endereçado por **[End. Disp.]**. Quando o dispositivo é endereçado com este comando, o numero de seu endereço é adicionado ao *prompt*

Exemplo:

```
!# Direciona o analisador tek2753, que está conectado ao endereço 5
!# e roda a MACRO 1
!5
run 1
!# Direciona o osciloscópio Tek2440, que está conectado ao endereço 3
!# e o apronta para receber sinal pelo CH1
!3
ID?
run acquire
acquire mode:normal
ch1 volts:2e-3
ch1 position:-3
!End
```

!Dev [Device Name]

Determina o endereço ocupado pelo dispositivo **[Device Name]** e o direciona. Quando o dispositivo é endereçado com este comando, o nome simbólico dele é adicionado ao *prompt*

Exemplo:

```
!# Endereça e envia comandos para a tela do plotter tek2753p.
!Device Tek2753p
  redout on
  grat on
!end
```

!Comm #TalkDev #ListDev :[CommCmd]

Estabelece comunicação entre dois elementos da rede, sendo que nenhum destes dois elementos é o controlador. Envia dados do dispositivo identificado pelo endereço **#TalkDev** para o dispositivo identificado por **#ListDev** utilizando o comando **[CommCmd]** como fonte geradora de tais dados.

Exemplo:

```
!# Permite a comunicação plotter-analisador,
!# o plotter esta no endereço 10 e o analisador no endereço 5
!dev tek2753p
    bwmode on
    grat on
    redout on
!comm 5 10 : plot?
!End
```

!ECho{X | 0}

Ativa **{X}** ou desativa **{0}** a apresentação dos comando lidos de um arquivo.

Exemplo:

```
!# Programa que realiza o envio a informação da tela do osciloscópio para um
!# plotter HP-GL, os comandos não são mostrados na tela do computador.

!#Osciloscópio por o endereço 1
!echo 1
!1
   message 10:"    enviando tela para plotter"
   device type:hpgl
   device grat:on
   device text:on
   device wavfrm:on
   device pagesize:A4
!c 1 10 : print
!echo 0
!end
```

!Print : [String]

Imprime mensagem na saída padrão (tela ou arquivo).

Exemplo

```
!# Teste do comando print
!Print : Este é o resultado de um comando print.
!End
```

!LEt VarName [VarValue1, VarValue2,..., VarValueN]

Declara variável **VarName** como um vetor composto pela sintaxe acima. A sintaxe de cada valor **VarValue** segue o formato da linguagem **'C'**. Estes valores são acessados com a entrada no modo de repetição.

!Help

Mostra a ajuda na tela do computador.

!REpeat #N_vezes

Repete a execução do comando que o segue **N_Vezes** vezes. Manipula ainda os índices das variáveis declaradas.

Exemplo:

```
!# Teste das funções básicas de repetição :let, rep
!# O arquivo pega os valores das colunas e executa

!let esco[1,2,3,4,5,6,7,8,9,10,11,12,13]
!let escg[1,2,3,4,5,6,7,8,9,10,11,12,13]
!print :Escala do Osc => %esco\tEscala do Gen => %escg\n
!rep 15
!run cal00.bat
!end
```

```
!# Este é o arquivo cal00.bat chamado desde o programa anterior
!# para testar repetição de arquivos dentro de arquivo
!print :Escala do Osc => %esco\tEscala do Gen => %escg\n
!end
```

!LOad [FileName]

Envia arquivo para o dispositivo atualmente endereçado.

Obs.: Pode ser substituído pelo comando **!Run**

Exemplo:

```
!# Carrega MACRO 2 no analisador
!echo1
!load mac2.bat
!end
```

!Net

Mostra na saída padrão (tela ou arquivo) a configuração de rede usada pelo sistema.

!Out { [FileName] | "off" }

Direciona a saída padrão para o arquivo **FileName** para geração de relatório. Caso se utilize **"off"** a saída padrão será retornada para a tela do computador.

Exemplo:

```
!# Declaração da Variáveis das escala do gerador e osciloscópio
!# O arquivo pega os valores por coluna como
!# Valor para executar a calibracão com no arquivo calosc.bat
!# Direciona a saída para o arquivo reporter

!let EscalaOsc [1e-3, 2e-3, 5e-3, 10e-3, 20e-3 ]
!let EscalaGen [1e-3, 2e-3, 5e-3, 10e-3, 20e-3 ]
!out reporter.rpt
!rep 5
!run calosc.bat
!out off
!end
```

!Quit

Sai do sistema.

!RUn [File Name]

Executa arquivo com seqüência de comandos. Muito útil para o trabalho com arquivos previamente editados.

Exemplo:

```
!# O arquivo pode ter qualquer extensão
!run arqivo.bat
```

!Free

Libera área de memória reservada para tabelas declaradas.

Exemplo:

```
!# Fica liberada a memória para que possa ser usada pelos
!# vetores esco e escg
```

```
!free
!let esco [20e-3, 50e-3, 100e-3, 200e-3]
!let escg [20e-3, 50e-3, 100e-3, 200e-3]
```

!TAble

Mostra na saída padrão as tabelas declaradas e seus valores.

!Wait

Faz a placa controladora aguardar por **TimeOut** ou por resposta de serviço.

!TIme #Tempo

Faz com que os sistema aguarde **Tempo** segundos.

Exemplo:

```
!# Direciona osc. Tek2440
!dev Tek2432a
run acquire
acquire mode:normal
ch1 volts:%EscalaOsc
ch1 position:-3
message 10:" Teste de transitório"
!time 0.1
!# Direciona gen. CG5050
!dev CG5010
mode v
u/d %EscalaGen
mult 6
out on
tring on
opc on
!end
```

No Anexo A se mostra-se a listagem do código fonte do **GPIBN*et*** e de alguns dos programas aplicativos feitos para avaliar o sistema.

5.3 Metodologia para a realização de programas aplicativos com o GPIBN*et*.

Associado com a projeção e construção deste sistema, foi proposta aos integrantes do **LACE** a metodologia para a realização de programas aplicativos a partir do **GPIBN*et***.

Após a realização das comprovações da operação do *hardware* com ajuda dos *software* para o diagnóstico fornecidos pela *National Instruments* no pacote associado ao cartão **PC-IIA,** pode ser aplicada a metodologia que propomos a seguir, para obter-se de uma forma algorítmica e gradual um bom funcionamento dos programas realizados.

5.3.1 Metodologia Proposta:

1. Editar a configuração do sistema com o arquivo de configuração **gpibnet.cnf.**
2. Utilização do **modo interativo**
3. Realização dos **programas aplicativos**

Uma vez que o sistema fica configurado, o programador ou operador pode consultar o endereço e/ou nome simbólico de cada dispositivo sempre que necessite. Posteriormente, propomos a utilização do **modo interativo** do **GPIBN*et*** que permite desde a linha de comando experimentar com funções individuais e observar os resultados antes de escrever o código definitivo. Neste ponto é onde testamos os algoritmos, comandos e instruções dos instrumentos, para realizar a incorporação de um novo equipamento, ou *upgrade* de um equipamento já existente na rede.

A última etapa proposta nesta metodologia é a obtenção do **programa aplicativo**, para a qual se faz uso do **modo programado** do **GPIBN*et***. De forma estruturada, se incorporam em um só programa um conjunto de arquivos com as seqüências e ordem corretas de todas operações que devem ser executados pelo sistema para cada

aplicação. Estes arquivos podem ser previamente editados e testados na etapa anterior.

5.4 O GPIBN*et* no Processo de Calibração Elétrica

Para realizar uma comparação em relação ao trabalho manual, que justifique o que a automação do processo de calibração representa para o trabalho no **LACE**, mostramos a Tabela 5.1. Nesta tabela, de forma resumida, apresenta-se parte do registro das medições do certificado de calibração de um osciloscópio digital. O preenchimento da tabela é somente parte do roteiro básico que de forma manual deve realizar o operador para calibrar tal instrumento.

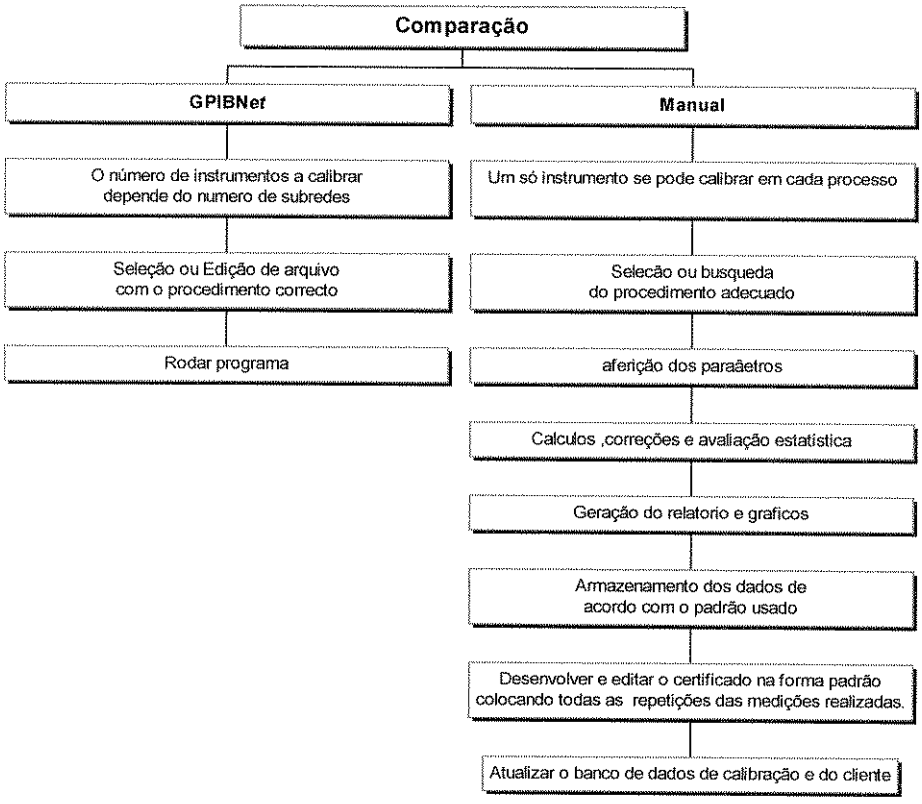
Tabela 5.1: Resumo do Registro das Medições de Calibração de um Osciloscópio.

Instrumento: Osciloscópio Digital			**Modelo: 2240**		**Fabricante:** TEKTRONIX		
Instrumentos Utilzados TEK SG502A, Série B077184 TEK SG503A, Série B010496 TEK SG504A, Série B014091 TEK SG506A, Série B011218 TEK TG501A, Série B010496			**Acessórios**		**Procedimentos de Calibração Utilizados** Procedimento manual fabricante		
Padrão	**Instrumento Sob Teste**				**Instrumento**		
Vertical TEK.PG 506A	Configurado	Canal Vertical A Esacala	Acoplamento	Valor lido sistema automático	Erro % Valor lido VPP (automático)	Valor lido (visual) tela	Erro % Valor lido (visual) tela
10mV	Bandwidth	2mV					
20mV	10MHz	5mV					
50mV	Smooth on	10mV					
.1V	Measure type	20mV					
.2V	VPP	50mV					
5V		1V					
20V		5V					
Vertical TEK.PG 506A	Configurado	Canal Vertical B Esacala	Acoplamento	Valor lido sistema automático	Erro % Valor lido VPP (automático)	Valor lido (visual) tela	Erro % Valor lido (visual) tela
10mV	Bandwidth	2mV					
20mV	10MHz	5mV					
50mV	Smooth on	10mV					
.1V	Measure type	20mV					
.2V	VPP	50mV					
1V		200mV					
5V		1V					
20V		5V					
	Time Mark TEK.TG 501[A] Segundo	Base de tempo A escala segundo	Acoplamento	Erro % Valor lido freqüência sistema automático	Erro % Valor lido (visual) relação padrão	Time Rise	Fall
	20 η	20 η					
	50 η	50 η					
	0.1 μ	0.1 μ					
	0.2 μ	0.2 μ					
	0.5 μ	0.5 μ					

Um operador com experiência trabalhando de forma ininterrupta demora em média **6 horas** para realizar todo o processo de calibração para este osciloscópio somente. Para cada ponto, devem ser tomadas **n leituras**, dependendo do método estatístico a utilizar. Observe-se que cada canal tem ao menos 23 parâmetros a medir, onde podemos incluir: escalas verticais, largura de faixa (*bandwidth*), base de tempo, acoplamento, *time rise/fall*, etc, além do grande número de instrumentos que participam no processo e que também devem ser controlados.

Com o sistema projetado, sempre é possível calibrar ao mesmo tempo mais de um instrumento. Por exemplo, o número de osciloscópios que podem ser calibrados num mesmo processo depende da habilidade do especialista para montar a rede, sabendo-se que cada sub-rede pode ter no máximo 15 equipamentos.

Na Figura 5.5 mostra-se uma comparação no processo de calibração realizado com os sistemas já existentes no laboratório e o novo sistema projetado.



Nos Anexos A.3, A.4 e A.5 mostra-se um programa aplicativo feito na linguagem do **GPIBN*et***, onde se realiza a calibração de todas as escalas verticais do amplificador vertical dos osciloscópios digitais Tek2440 e Tek2432, demorando em media 7 min.

Capítulo 6

Conclusões

6.1 Revisão de objetivos e resultados.

Estabeleceu-se como objetivo geral para este trabalho o projeto e construção de um sistema que de forma automatizada permitisse um aumento significativo da produtividade e da qualidade do processo de aferição e calibração no Laboratório de Calibração Elétrica, **LACE** da UNICAMP. Procurou-se atingir esse objetivo abordando as características gerais do trabalho no laboratório, a partir de uma detalhada avaliação dos recursos técnicos disponíveis no mesmo. Como solução definitiva foi criada uma **Rede automatizada com interface IEEE - 488 para a calibração de instrumentos** e desenvolvido o *software* **GPIBN*et*** que constitui seu sistema operacional, implementando-se um sistema que permite a comunicação e controle de instrumentos de diferentes fabricantes, execução de procedimentos de aferição, calibração e teste, gerar relatórios e certificados de forma automática e armazenar os resultados obtidos.

Algumas das principais idéias relacionadas ao sistema projetado foram originalmente apresentadas em **[Fukle, 1989], [Gao, Spiriti & Tortora, 1994]** e **[Fluke, 1996].**

O sistema proposto tem como preocupação central possibilitar ao pessoal técnico do laboratório desenvolver programas aplicados à calibração de qualquer tipo de instrumento e fabricante, a partir de arquivos previamente editados e testados individualmente com os equipamentos correspondentes. Estes arquivos podem ser vistos como módulos de *software* reutilizáveis, que permitem a redução do tempo de implementação e manutenção de um programa a partir das instruções de um novo instrumento introduzido na rede, bem como do tempo de realização da incorporação de um novo

equipamento, ou ainda o *upgrade* de um equipamento já existente na rede. Nesta abordagem o projeto de uma aplicação tem como preocupação central definir a Metodologia para a implementação de aplicações, a qual está caracterizada por duas etapas principais: programação de arquivos e configuração de aplicações a partir dos arquivos disponíveis.

A construção de aplicações a partir de arquivos (módulos reutilizáveis) dá um caráter funcional a este processo devido a que, os módulos podem ser usados em qualquer aplicação. Além disso o módulo pode ser completamente testado e verificado antes de ser incorporado ao programa aplicativo. Esta estratégia de desenvolvimento modular combinado com o uso das sub-rede permite uma ampla integração do *hardware/software*

6.2 Discussão dos resultados

Até recentemente, os sistemas com que contava o **LACE** para o desenvolvimento do trabalho de aferição/calibração, conforme foi discutido no Capitulo 2, não proporcionavam um modo eficiente e produtivo para a realização das principais tarefas que o laboratório realiza. Estabeleceu-se como objetivo principal deste trabalho contribuir para o desenvolvimento de um sistema para a calibração de instrumentos que estivesse de acordo com o nível de qualidade que a atividade exige, e que permitiria dar um respaldo metrológico de alto nível ao conjunto de clientes que o laboratório tem.

Procurou-se oferecer flexibilidade e facilidade através do modelo de programação, suportado pelo **GPIBN*et,*** que oferece uma interface simples para a familiarização do programador com o novo conjunto de instruções, em contraste com os métodos anteriormente aplicados. Por outro lado, buscou-se assegurar um sistema ao qual se poderia incorporar com facilidade novos equipamentos de modo que se pudessem atender com facilidade as exigências dos clientes do laboratório.

É importante ressaltar que o sistema foi desenvolvido não somente para que possa ser aplicado em laboratórios de calibração elétrica, mas também para que possa ser aplicado em qualquer outra rede com interface IEEE- 488.

6.3 Continuidade deste trabalho

A partir dos resultados obtidos neste trabalho, sugerem-se, a seguir, três linhas de ação que poderiam direcionar trabalhos futuros.

A primeira linha de ação sugerida teria como objetivo desenvolver e executar aplicações fazendo uso efetivo das possibilidades do conjunto de funções do **GPIBN*et.*** Uma segunda linha de ação teria por finalidade o aprofundamento em tópicos específicos considerados neste trabalho. A terceira linha de trabalho sugerida teria como objetivo aumentar a flexibilidade e generalidade do sistema projetado e construído através de um processo evolutivo.

- Desenvolver e executar aplicações com o ***GPIBNet***

A implementação de aplicações dentro do área de instrumentação, não limitadas só ao processo de calibração permitirá avaliar a efetividade do sistema. A sua utilização experimental possibilitará a obtenção de um entendimento maior das características associadas ao desenvolvimento e suporte deste sistema, a partir de que novas funções, mecanismos e conceitos poderiam ser propostos, implementados e avaliados.

A criação incremental de funções deve neste contexto ser entendida como um processo em várias etapas, ou estágios, onde a transição de uma etapa para a seguinte ocorre à medida que se obtém um maior entendimento da estrutura do sistema e dos seus componentes.

- Evolução do sistema

O sistema pode evoluir em diversos graus e direções. Uma evolução interessante do sistema seria a incorporação de outros pacotes de *software* dedicados à instrumentação como o ***LabWindows*** desenvolvido pela *National Instruments* o qual entre outras características possui um avançado editor gráfico e um completo suporte de instrumentação com bibliotecas para IEEE- 488, IEEE-488.2, etc **[Cautero, 1994]**.

Outra linha de evolução seria a inclusão do ***MATLAB GPIB code*** desenvolvido pela *Apple*, *Caltech* e *Stanford* fundamentalmente para o controle e aquisição de

dados usando os *drivers* da *National Instruments* **[Apple, Caltech & Satanford 1994]**.

Anexos.

A.1 Sistema Operacional GPIBN*et*

```
/*****************************************************************************\
* GPIBNet, Sistema Operacional  da rede de instrumentos com interface  IEEE-488 (GPIB)
*.
*****************************************************************************
 *
 * Comandos de controle disponíveis:
 *
 *******************************
 * !# [String]
 *  => Comentario.
 *******************************
 * !? [Comando]
 *  => Pergunta para dispositivo que não contenha '?' no [Comando].
 *******************************
 * !' '[DosCmd]
 *  => Envia o comando [DosCmd] para execução via sistema operacional.
 *******************************
 * ![End. Disp.]
 *  => Direciona dispositivo endereçado por [End. Disp.].
 *******************************
 * !Comm #TalkDev #ListDev : [CommCmd]
 *  => Estabelece comunicação entre dois elementos da rede, sendo que
 *  nenhum deste dois elementos , o controlador.
 *  Envia dados do dispositivo identificado pelo endereco #TalkDev para o
 *  dispositivo identificado por #ListDev utilizando o comando [CommCmd]
 *  como fonte geradora de tais dados.
 *******************************
 * !Dev [Device Name]
 *  => Determina o endereco ocupado pelo dispositivo [Device Name] e o
 *  direciona.
 *******************************
 * !ECho {X | 0}
 *  => Ativa {X} ou desativa {0} a apresentação dos comando lidos de um
 *  arquivo.
 *******************************
 * !ENd
 *  => Sinaliza fim de arquivo de comandos.
 *******************************
 * !LEt VarName [VarValue1, VarValue2, ..., VarValueN]
 *  => Declara variavel VarName como um vetor composto pela sintaxe acima,
 *  a sintaxe de cada valor VarValue segue o formato da linguagem 'C'.
 *  Estes valores são acessados com a entrada no mode de repetição.
 *******************************
 * !LOad [FileName]
 *  => Envia arquivo para o dispositivo atualmente endereçado.
 *  Obs.: Pode ser substituido pelo comando !RUn.
 *******************************
 * !Net
 *  => Mostra na saida padrão (tela ou arquivo) a configuração de rede
 *  usada pelo sistema.
 *******************************
```

```
* !Out {[FileName] | "off"}
*  => Direciona a saida padrão para o arquivo FileName para geração de
*  relatório. Caso se utilise "off" a saida padrão ser  retornada para a
*  tela do computador.
*********************************
* !Print :[String]
*  => Imprime mensagem na saida padrão (tela ou arquivo)
*
*********************************
* !Quit
*  => Sai do sistema.
*********************************
* !REpeat #N_Vezes
*  => Repete a execução do comando que o segue N_Vezes vezes.
*  Manipula ainda os índices  das variaveis declaradas.
*********************************
* !RUn [File Name]
*  => Executa arquivo com sequencia de comandos.
*********************************
* !Show
*  => Mostra Macro Redout Buffer do TEK2753P.
*  (Função Obsoleta)
*********************************
* !TAble
*  => Mostra na saÍda padrão as tabelas declaradas e seus valores.
*********************************
* !TIme #Tempo
*  => Faz com que os sistema aguarde Tempo segundos.
*********************************
* !Wait
*  => Faz a placa controladora aguardar por TimeOut ou por resposta de
*  serviço.
\*****************************************************************************/


/* Headers Padrão */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
/* Headere GPIB */
#include "decl.h"    /* Contem declaração de variaveis e funções GPIB */

/* Macros Utilizadas no Programa */
#define  isdigit(x)     (((x)>=48)?(((x)<=57)?(1):(0)):(0))
#define  istab(x)       ((((x)==' ')||((x)=='\t'))?(1):(0))
#define  islet(x)       ((((x)>='A')&&((x)<='Z')||((x)>='a')&&((x)<='z'))?(1):(0))
#define  isspechar(x)   ((((x)=='\\')||((x)=='%')||((x)=='\0'))?(1):(0))
#define  isvarchar(x)   ((((x)>='A')&&((x)<='Z')||((x)>='a')&&((x)<='z')||((x)>='0')&&((x)<='9'))?(1):(0))

#define  toupper(x)     ((((x)>='a')&&((x)<='z'))?((x)-32):(x))

/* Constantes Utilizadas no programa */
#define  RespBuffLen       2048
#define  LineLength        128
#define  MaxDevNumber      15
#define  MaxInputFiles     90
#define  MainHelpTableLength 8
```

```
#define  ListAddBase       ' '
#define  TalkAddBase       '@'
#define  SecAddBase        '\"

#define  TimeOut           10.0L

#define  GPIBNetConfLineLength  35
#define  MaxVarNameLength       33
#define  MaxNumberOfVar                    30

#define NumberOfCmd                    21

#define GRAY                           LIGHTGRAY
/* TextAttr          Letra    Fundo       */
#define WhiteBlack                     (WHITE  +(BLACK<<4))
#define CyanBlue                       (CYAN   +(BLUE<<4))
#define YellowBlue                     (YELLOW +(BLUE<<4))
#define RedBlue                        (RED    +(BLUE<<4))
#define WhiteBlue                      (WHITE  +(BLUE<<4))
#define GreeBlue              (GREEN  +(BLUE<<4))
#define CyanGray                       (CYAN   +(GRAY<<4))
#define YellowGray                     (YELLOW +(GRAY<<4))
#define RedGray                        (RED    +(GRAY<<4))
#define WhiteGray                      (WHITE  +(GRAY<<4))
#define GreenGray                      (GREEN  +(GRAY<<4))


/* Vari veis Globais */
char resp[RespBuffLen];
char *ptr;
int  bd;  /*numero del cartão */
int  DeviceNumber;
char **ListaDev;
int  ControlerNumber = 0;
char prompt[70] = "none> ";
char espaco[21] = "                    ";
char ** VarName;
float ** VarValue;
int NextFreeVar = 0;
int RepeatCnt   = 0;
int RepeatMode  = 0;
int RepeatIndexValue = 0;

char GPIBNetConf[15] = "gpibnet.cnf";
char NoUsedDev[5] = "none";
char Controler[17] = "Controler Board\0";

/**/
#define NumeroLines   2
#define NumeroIndex   0
char AjudaNumero[] =
" ![n&mero]                                                    "
"     Direciona o dispositivo endereçado por [número].                    ";
/**/
#define ComentLines   2
#define ComentIndex   1
char AjudaComent[] =
" !# [String]                                                  "
```

```
"     Assume que [String] , um comentario e ignora a linha.                    ";
/**/
#define PergunLines  3
#define PergunIndex  2
char AjudaPergun[] =
" !? [Comando]                                                                 "
"     Envia pergunta para dispositivo e aguarda resposta, desnecess rio caso   "
" a pergunta tenha o sinal '?'.                                                ";
/**/
#define EspacoLines  2
#define EspacoIndex  3
char AjudaEspaco[] =
" !"[DosCmd]                                                                   "
"     Envia [DosCmd] para execução via sistema operacional.                    ";
/**/
#define ComuniLines  5
#define ComuniIndex  4
char AjudaComuni[] =
" !Comm #Talk #List : [CommCmd]                                                "
"     Estabelece comunicação entre dois elementos da rede, sendo que nenhum    "
" deste elementos , o controlador. Envia dados do dispositivo identificado     "
" pelo endereço #TalkDev para o dispositivo identificado por #ListDev          "
" utilizando o comando [CommCmd] como fonte geradora de tais dados.            ";
/**/
#define DevLines  2
#define DevIndex  5
char AjudaDev[]  =
" !Dev [Device Name]                                                           "
"     Direciona o dispositivo nomeado por [Device Name].                       ";
/**/
#define EchoLines  3
#define EchoIndex  6
char AjudaEcho[]  =
" !ECho {X | 0}                                                                "
"     Ativa {X} ou desativa {0} a apresentação dos comando lidos de um         "
" arquivo.                                                                     ";
/**/
#define EndLines  2
#define EndIndex  7
char AjudaEnd[]  =
" !ENd                                                                         "
"     Sinaliza fim de arquivo de comandos.                                     ";
/**/
#define FreeLines  2
#define FreeIndex  8
char AjudaFree[]  =
" !Free                                                                        "
"     Libera aréa de memoria reservada para tabelas declaradas.                ";
/**/
#define HelpLines  2
#define HelpIndex  9
char AjudaHelp[] =
" !Help                                                                        "
"     Mostra a ajuda na tela do computador.                                    ";
/**/
#define LetLines  4
#define LetIndex  10
char AjudaLet[]  =
" !LEt VarName [VarValue1, VarValue2, ..., VarValueN]                          "
```

```
"     Declara vari vel VarName como um vetor composto pela sintaxe acima, a     "
" sintaxe de cada valor VarValue segue o formato da linguagem 'C'. Estes valores "
" sÆo acessados com a entrada no mode de repetição.                              ";
/**/
#define LoadLines   3
#define LoadIndex   11
char AjudaLoad[]  =
" !LOad [FileName]                                                              "
"     Envia arquivo para o dispositivo atualmente endereçado.                   "
" Obs.: Pode ser substituido pelo comando !RUn.                                 ";
/**/
#define NetLines   3
#define NetIndex   12
char AjudaNet[]  =
" !Net                                                                          "
"     Mostra na saida padrÆo (tela ou arquivo) a configuração de rede usada     "
" pelo sistema.                                                                 ";
/**/
#define OutLines   4
#define OutIndex   13
char AjudaOut[]  =
" !Out {[FileName] | off}                                                       "
"     Direciona a saida padrão para o arquivo FileName para geração de          "
" relatório. Caso se utilize off a saida padrão ser  retornada para a tela do   "
" computador.                                                                   ";
/**/
#define PauseLines   2
#define PauseIndex   14
char AjudaPause[]  =
" !PAuse :[String]                                                              "
"     Imprime mensagem na tela e aguarda pressionamento de uma tecla.           ";
/**/
#define PrintLines   2
#define PrintIndex   15
char AjudaPrint[]  =
" !PRint :[String]                                                              "
"     Imprime mensagem na saida padrão (tela ou arquivo).                       ";
/**/
#define QuitLines   2
#define QuitIndex   16
char AjudaQuit[]  =
" !Quit                                                                         "
"     Sai do sistema.                                                           ";
/**/
#define RepeatLines   3
#define RepeatIndex   17
char AjudaRepeat[] =
" !REpeat #N_Vezes                                                              "
"     Repete a execuao do comando que o segue N_Vezes vezes. Manipula ainda os  "
" índices de das variaveis declaradas.                                          ";
/**/
#define RunLines   2
#define RunIndex   18
char AjudaRun[]  =
" !RUn [File Name]                                                              "
"     Executa arquivo com sequencia de comandos.                                ";
/**/
#define TableLines   3
#define TableIndex   19
```

```
char AjudaTable[] =
" !TAble                                                    "
"     Mostra na sa¡da padrÆo as tabelas declaradas e seus valores.          ";
/**/
#define TimeLines  2
#define TimeIndex  20
char AjudaTime[] =
" !TIme #Tempo                                              "
"     Faz com que o sistema aguarde Tempo segundos.                    ";
/**/
#define WaitLines  2
#define WaitIndex  21
char AjudaWait[] =
" !Wait                                                     "
"     Faz a placa controladora aguardar por TimeOut ou por resposta de servi‡o  ";


int OldActFile;
int StartActFile;
int RptFileMode;
int DeltaActFile;
int MaxRptInd;
int RptIndFile;

char *HelpText[NumberOfCmd];
int  HelpLength[NumberOfCmd];

FILE *out;

/* Protótipos de funções */
void gpiberr(char *msg);
void show(void);
void start(void);
void load(char *);
void command(char *, int);
void trocafim(char *);
void trocafimdos(char *);
void trocafimambos(char *);
char **LeConf(char *);
void PreProc(char *, char *);
void HelpShow(int);
void InitHelp(void);
void FILEcpy(FILE *, FILE*);

int  RepeatLoop;

/*Programa Principal*/
void main(int argc, char **argv)
{
 char ComandoIn[LineLength], *ptr, *ptraux, resp[LineLength];
 char *ComandoPre, comando[LineLength], aux[31];
 FILE *in[MaxInputFiles], AuxFile, *LoopBackPoint;
 int fim = 0, EchoOn = 0, len, i , j;
 int TalkerDevAdd,ListerDevAdd, ActFile = 0, FimDeCmd = 0, Yinic;
 int VarLength, VarNumber;
 char gpibcmd[6];
 int Help = 0;
 double t3;
 float Time;
```

```
time_t t1,t2;

LoopBackPoint = &AuxFile;

textmode(C80);
textbackground(BLUE);
clrscr();

if (argc != 1){
        ActFile ++;
        if (!(in[ActFile] = fopen(argv[1], "rb"))){
         ActFile --;
         fprintf(stderr, "Não achei arquivo a ser executado : %s\n", argv[1]);
        }
}

textattr(WhiteGray);
cprintf("ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»");
cprintf("º GPIBNet    - Sistema Operacional para redes IEEE-488 (GPIB)          º");
cprintf("º LACE/CEMEQ - Laboratório de Calibração Elétrica                      º");
cprintf("º DECOM/FEE  - Departamento de Comunica‡äes                            º");
cprintf("º Unicamp    - Universidade Estadual de Campinas                       º");
cprintf("º Copyrigth (C) - 1997 por Eng. Vicente I. Becerra Sablón .            º");
cprintf("º Versão Beta.0                                                        º");
cprintf("ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼");

textattr(YellowBlue);
InitHelp();
ListaDev = LeConf(GPIBNetConf);
ListaDev[0] = &Controler[0];

start();
/* in , inicializado como stdin, ie, direciona a entrada para o teclado */
/* out , inicializado como stdout, ie, direciona a sa¡da para o video   */
in[0] = stdin;
out   = stdout;

if (!(VarName = (char **)calloc(sizeof(char *), MaxNumberOfVar))){
        fprintf(stderr, "Lamento não h  mem¢ria dispon¡vel para a operação.\n");
        exit(1);
}
if (!(VarValue = (float **)calloc(sizeof(float *), MaxNumberOfVar))){
        fprintf(stderr, "Lamento não h  mem¢ria disponível para a operação.\n");
        exit(1);
}

OldActFile = ActFile;
StartActFile = ActFile;
RptFileMode = 0;
fim = 0;

fprintf(stdout,"\n");
while (!fim){
        if (in[ActFile] == stdin)
         fprintf(stdout,"%s", prompt);
        /* Le de in a linha a ser interpretada*/
        fgets(ComandoIn, LineLength, in[ActFile]);

        ComandoPre = &ComandoIn[0];
```

```
if (istab(ComandoPre[0]))
  while(istab(*ComandoPre)) ComandoPre ++;

DeltaActFile = ActFile - OldActFile;
if (DeltaActFile > 1) { /* Foi dado um run apos um repeat */
  MaxRptInd    = DeltaActFile;
  RptIndFile   = DeltaActFile;
  RptFileMode  = DeltaActFile;
}
OldActFile  = ActFile;

do {
RepeatMode = (RepeatCnt != 0)?(1):(0);
PreProc(ComandoPre, &comando[0]);
if ((in[ActFile] != stdin) && (EchoOn))
  fprintf(out,"%s\n", comando);


/* Caso a linha comece com ! o prog entende como um comtrole */
/* interno, caso contr rio interpreta como comando para o    */
/* dispositivo atualmente direcionado                        */
if (comando[0] == '!')
        /* Caso o controle seja um n£mero, ser  direcionado o    */
        /* dispositivo com o endere‡o correspondente             */
        if (isdigit(comando[1])){
          if (Help) {
                HelpShow(0);
                Help = 0;
          }
          else {
                if (isdigit(comando[2]))
                  DeviceNumber = 10*(comando[1] - '0') + comando[2] - '0';
                else
                  DeviceNumber = comando[1] - '0';
                if (DeviceNumber > MaxDevNumber)
                  DeviceNumber = MaxDevNumber;
                if (DeviceNumber < 0)
                  DeviceNumber = 0;
                sprintf(prompt, "%s> ", ListaDev[DeviceNumber]);
          }
        }
        else /* Comandos para controle interno */
          switch (comando[1]){
                case '#': /* !# Considera string que segue como coment rio */
                        if(Help){
                          HelpShow(ComentIndex);
                          Help = 0;
                        }
                        break;
                case '?': /* !? Utilaza para fazer perguntas a dispositivo,*/
                                /* quando o c¢digo normal nÆo cont,m '?'       */
                        if(Help){
                          HelpShow(PergunIndex);
                          Help = 0;
                        }
                        else
                          command(&comando[2], 1);
                        break;
                case ' ': /* !<espa‡o> Interpreta a string que segue o */
```

```
                    /* espa‡o como um comando para o DOS        */
                if(Help){
                  HelpShow(EspacoIndex);
                  Help = 0;
                }
                else {
                  ptr = &comando[2];
                  while (istab(*ptr)) ptr ++;
                  j = wherey();
                  if (j == 25){
                        fprintf (stderr,"\n");
                        gotoxy(1,24);
                  }
                  system(ptr);
                  textmode(C80);
                  textbackground(BLUE);
                  textattr(YellowBlue);
                }
                break;
case 'b': /* !Begin                                */
case 'B': /* Declara in¡cio de bloco para repeticao */
                RepeatMode = 0;
                RepeatLoop = RepeatCnt;
                RepeatCnt = 0;
                FILEcpy(in[ActFile], LoopBackPoint);
                break;
case 'C': /* !Comm #t #l :[Cmd]                    */
case 'c': /* Estabelece comunica‡Æo entre dois elememtos da rede */
                    /* utiliza como parametro o numero do dispositivo      */
                    /* talker e listern e o comando que o ir  gerar os     */
                    /* dados a serem comunicados                           */
                if(Help){
                  HelpShow(ComuniIndex);
                  Help = 0;
                }
                else {
                  TalkerDevAdd = 0;
                  ListerDevAdd = 0;
                  ptr = &comando[1];
                  while (islet(*ptr)) ptr ++;
                  while (istab(*ptr)) ptr ++;
                  /*algoritmo de convers€o string para inteiro*/
                  while (isdigit(*ptr)){
                        TalkerDevAdd = 10*TalkerDevAdd + (*ptr - '0');
                        ptr++;
                  }
                  while (istab(*ptr)) ptr ++;
                  /*algoritmo de convers€o string para inteiro*/
                  while (isdigit(*ptr)){
                        ListerDevAdd = 10*ListerDevAdd + (*ptr - '0');
                        ptr++;
                  }
                  while (*(ptr++)!=':');
                  while (istab(*ptr))ptr++;

                  ptraux = ptr;
                  len = 0;
                  while (*(ptraux) != '\0') {
                        len ++;
```

```
                    ptraux++;
                }
                start();
                sprintf(gpibcmd, "%c%c%c%c",
LLO,ListAddBase+TalkerDevAdd,DCL,TalkAddBase+ControlerNumber);
                ibcmd (bd,gpibcmd,4);
                if (ibsta & ERR)
                    gpiberr("ibcmd Error");

                ibwrt(bd,ptr,len);
                if (ibsta & ERR)
                    gpiberr("Erro de comunica‡Æo");

                for(i = 0; i++; i<1000);


                sprintf(gpibcmd, "%c%c%c%c",
UNL,UNT,TalkAddBase+TalkerDevAdd,ListAddBase+ListerDevAdd);


                ibcmd (bd,gpibcmd,4);
                if (ibsta & ERR)
                    gpiberr("Erro de comando");
                ibgts (bd,1);
                if (ibsta & ERR)
                    gpiberr("Erro usando ibgts");
            }
            break;
        case 'd': /* !Dev [Device Name]        */
        case 'D': /* Ativa dispositivo pelo nome */
            if(Help){
              HelpShow(5);
              Help = 0;
            }
            else {
              ptr = &comando[2];
              while (islet(*ptr)) ptr ++;
              while (istab(*ptr)) ptr ++;
              ptraux = ptr;
              while (isvarchar(*ptraux)) ptraux ++;
              *ptraux = '\0';
              for (i = 0; i < MaxDevNumber+1; i++)
                  if (!stricmp(ptr, ListaDev[i])){
                    DeviceNumber = i;
                    sprintf(prompt, "%s> ", ListaDev[i]);
                  }
              }
            break;
        case 'e':
        case 'E':
            switch (comando[2]){
              case 'c': /* !ECho {X | 0}                */
              case 'C': /* Mostra os comando lidos de arquivo */
                                  /* se echo != 0                */
                  if(Help){
                    HelpShow(6);
                    Help = 0;
                  }
                  else {
```

```
                    ptr = &comando[2];
                    while(islet(*ptr)) ptr ++;
                    while(istab(*ptr)) ptr ++;
                    if ((*ptr)=='0')
                            EchoOn = 0;
                    else
                            EchoOn = 1;
                  }
                  break;
          case 'n': /* !ENd                    */
          case 'N': /* Retorna o nivel de arquivo ativo */
                                  /* para o arquivo anterior      */
                  if(Help){
                   HelpShow(7);
                   Help = 0;
                  }
                  else {
                   if (ActFile != 0){
                            fclose(in[ActFile]);
                            ActFile --;
                   }
                   if (RptFileMode >= 0)
                            RptFileMode --;
                  }
                  break;
          case 'r':
          case 'R':
                  if (RepeatLoop-- != 1)
                   FILEcpy(LoopBackPoint, in[ActFile]);
                  break;
          default:
                  fprintf(stderr, "Comando desconhecido.\n");
                  break;
        }
        break;
case 'f': /* !Free */
case 'F': /* Retira a declara‡Æo das tabelas */
        if(Help){
         HelpShow(8);
         Help = 0;
        }
        else {
         for (i = 0; i < NextFreeVar; i++){
                  free(VarName[i]);
                  free(VarValue[i]);
                  NextFreeVar = 0;
         }
        }
        break;
case 'h':
case 'H':
        if (Help) {
         HelpShow(9);
         Help = 0;
        }
        else {
         j = wherey();
         for (i = 0; i < MainHelpTableLength; i++)
                  fprintf(stderr, "\n");
```

```
                    if (j >= 25 - MainHelpTableLength)
                        gotoxy(1,25 - MainHelpTableLength);
                    else
                        gotoxy(1,j);
                    textattr(RedGray);
                    cprintf ("É͸»\n\r");
                    cprintf ("º !#    !?    !' '   !n£mero  º\n\r");
                    cprintf ("º !Comm  !Dev   !ENd   !ECho   º\n\r");
                    cprintf ("º !Free  !Help  !LEt   !LOad   º\n\r");
                    cprintf ("º !Net   !Out   !PRint !PAuse  º\n\r");
                    cprintf ("º !Quit  !REpeat !RUn   !TAble  º\n\r");
                    cprintf ("º !TIme  !Wait                  º\n\r");
                    cprintf ("È͸¼\n\r");
                    sprintf (prompt, "Digite comando a ser consultado: ");
                    textattr(YellowBlue);
                    Help = 1;
                  }
                  break;
        case 'l':
        case 'L':
                  switch (comando[2]) {
                    case 'e': /* !LEt VarName [VarValue, VarValue, ...] */
                    case 'E': /* Declara tabela                    */
                        if(Help){
                          HelpShow(10);
                          Help = 0;
                        }
                        else {
                          ptr = &comando[2];
                          while (islet(*ptr)) ptr ++;
                          while (istab(*ptr)) ptr ++;
                          /* Aloca espa‡o para o nome da tabela declarada */
                          if (!(VarName[NextFreeVar] = (char *) calloc(sizeof(char),
MaxVarNameLength))){
                              fprintf(stderr, "Lamento nÆo h  mem¢ria para a
opera‡Æo.\n");
                              exit(1);
                          }
                          /* Armazena Nome da tabela declarada */
                          i = 0;
                          while (isvarchar(*ptr))
                              VarName[NextFreeVar][i++] = *ptr++;
                          VarName[NextFreeVar][i] = '\0';

                          VarNumber = -1;
                          for (i = 0; i < NextFreeVar; i++)
                              if (!strcmp(VarName[NextFreeVar], VarName[i]))
                                VarNumber = i;
                          if (VarNumber != -1){
                              fprintf(stderr, "Tabela redeclarada : %s\n",
VarName[NextFreeVar]);
                              break;
                          }
                          /* Determina comprimento da tabela declarado */
                          ptraux = ptr;
                          i = 0;
                          while (*ptraux != '\0'){
                              if (*ptraux == ',')
                                i ++;
```

```
                                        ptraux ++;
                                    }
                                    VarLength = i + 1;
                                    /* Aloca espa‡o para os valores da tabela declarada */
                                    if (!(VarValue[NextFreeVar] = (float *) calloc(sizeof(float),
VarLength+1))){
                                        fprintf(stderr, "Lamento nÆo h  mem¢ria para a
opera‡Æo.\n");
                                        exit(1);
                                    }
                                    VarValue[NextFreeVar][0] = VarLength;
                                    /* Armazena valores das c,lulas */
                                    while (*ptr != '[') ptr ++;
                                    ptr ++;
                                    for (i = 1; i < VarLength+1; i++){
                                        ptraux = ptr;
                                        while (*ptraux != ',' && *ptraux != ']') ptraux ++;
                                     *ptraux = '\0';
                                     VarValue[NextFreeVar][i] = atof(ptr);
                                     ptr = ptraux+1;
                                    }
                                    NextFreeVar ++;
                                }
                                break;
                            case 'o': /* !LOad [File Name]              */
                            case 'O': /* Carrega arquivo para dispositivo */
                                                /* atualmente direcionado          */
                                if(Help){
                                  HelpShow(11);
                                  Help = 0;
                                }
                                else {
                                  ptr = &comando[2];
                                  while (islet(*ptr)) ptr++;
                                  while (istab(*ptr)) ptr++;
                                  load(ptr);
                                }
                                break;
                             default:
                                fprintf(stderr, "Comando desconhecido.\n");
                                break;
                        }
                        break;
                    case 'n': /* !Net                          */
                    case 'N': /* Mostra a configura‡Æo da rede gpib */
                        if (Help) {
                          HelpShow(12);
                          Help = 0;
                        }
                        else {
                          fprintf(out,"\n");
                          for (i = 0; i < MaxDevNumber + 1; i++)
                                fprintf(out, "\tDevice %02d => %s\n", i, ListaDev[i]);
                          fprintf(out,"\n");
                        }
                        break;
                    case 'o': /* !Out  {[File Name] | "off"}  */
                    case 'O': /* Direciona saida para arquivo */
                      if (Help) {
```

```
                HelpShow(13);
                Help = 0;
        }
        else {
                ptr = &comando[2];
                while (islet(*ptr)) ptr++;
                while (istab(*ptr)) ptr++;
                if (!stricmp(ptr, "off")){
                        if (out != stdout)
                          fclose(out);
                        out = stdout;
                }
                else
                        if (!(out = fopen(ptr, "wb")))
                          fprintf(stderr, "NÆo pude abrir arquivo : %s", ptr);
        }
        break;
case 'p':
case 'P':
        switch(comando[2]){
                case 'a': /* !PAuse :[Mensagem]            */
                case 'A': /* Mostra na tela uma mensagem e   */
                                /* aguarda a pressÆo de uma tecla. */
                  if (Help) {
                        HelpShow(14);
                        Help = 0;
                  }
                  else {
                        ptr = &comando[2];
                        while (islet(*ptr)) ptr ++;
                        while (istab(*ptr)) ptr ++;
                        ptr ++;
                        fprintf(stderr,"%s",ptr);
                        getch();
                  }
                  break;
                case 'r': /* !PRint : [String de Controle]      */
                case 'R': /* Imprime mensagem na tela do usu rio */
                  if (Help) {
                        HelpShow(15);
                        Help = 0;
                  }
                  else {
                        ptr = &comando[2];
                        while (islet(*ptr)) ptr ++;
                        while (istab(*ptr)) ptr ++;
                        ptr ++;
                        fprintf(out,"%s",ptr);
                  }
                  break;
                default:
                        fprintf(stderr, "Comando desconhecido.\n");
                        break;
        }
        break;
case 'q': /* !Quit          */
case 'Q': /* Sai do programa */
        if (Help) {
                HelpShow(16);
```

```
            Help = 0;
        }
        else
            fim = 1;
        break;
    case 'r':
    case 'R':
            switch(comando[2]){
             case 'e': /* !REpeat #N_Times  */
             case 'E': /* Modo de Repeti‡Æo */
              if (Help) {
                    HelpShow(17);
                    Help = 0;
              }
              else {
                    ptr = &comando[3];
                    while (islet(*ptr)) ptr ++;
                    while (istab(*ptr)) ptr ++;
                    if (isdigit(ptr[0]))
                     if (isdigit(ptr[1]))
                            RepeatCnt = 10*(ptr[0] - '0') + ptr[1] - '0';
                     else
                            RepeatCnt = ptr[0] - '0';
                    RepeatIndexValue = 0;
              }
              MaxRptInd = RepeatCnt;
              break;
             case 'u': /* !RUn [File Name]    */
             case 'U': /* Execucao de arquivo */
              if (Help) {
                    HelpShow(18);
                    Help = 0;
              }
              else {
                    ptr = &comando[3];
                    while (islet(*ptr)) ptr ++;
                    while (istab(*ptr)) ptr ++;
                    ptraux = ptr;
                    j = 0;
                    while (*ptraux != '\0'){
                     if (*ptraux == '.')
                            j ++;
                     ptraux ++;
                    }
                    if (!j){
                     ptraux = ptr;
                     while (isvarchar(*ptraux)) ptraux ++;
                     strcpy (ptraux, ".gnt");
                     fprintf(stderr, "%s\n", ptr);
                    }
                    ActFile ++;
                    /* Direciona a entrada para o arquivo requirido */
                    if (!(in[ActFile] = fopen(ptr, "rb"))){
                     ActFile --;
                     fprintf(stderr, "NÆo pude abrir o arquivo : %s\n", ptr);
                    }
              }
              break;
             default:
```

```
                fprintf(stderr, "Comando desconhecido.\n");
                break;
            }
            break;
case 't':
case 'T':
            switch(comando[2]){
              case 'a': /* !Table                          */
              case 'A': /* Mostra as tebelas decl. e seus conteudos */
                        if (Help) {
                          HelpShow(19);
                          Help = 0;
                        }
                        else {
                          for (i = 0; i < NextFreeVar; i ++){
                           fprintf(out, "%s(%d) =", VarName[i], (int)VarValue[i][0]);
                           for (j = 0; j < (int) VarValue[i][0]; j++){
                                    if(j == 7*(int)(j/7.0))
                                      fprintf(out, "\n");
                                    fprintf(out, "\t%5.1e", VarValue[i][j+1]);
                           }
                           fprintf(out, "\n");
                          }
                        }
                        break;
              case 'i': /* !Time #Sec                      */
              case 'I': /* Faz o proc. parar por #Sec segundos */
                        if (Help) {
                          HelpShow(20);
                          Help = 0;
                        }
                        else {
                          ptr = &comando[2];
                          while (islet(*ptr)) ptr++;
                          while (istab(*ptr)) ptr++;
                          Time = atof(ptr);
                          time(&t1);
                          do{
                                    time(&t2);
                                    t3 = difftime(t2,t1);
                          } while(t3 < Time);
                        }
                        break;
              default:
                        fprintf(stderr, "Comando Desconhecido\n");
                        break;
            }
            break;
case 'w': /* !Wait                                */
case 'W': /* Faz placa controladora esperar pelo Timeout */
                        /* ou por resposta de servico          */
            if (Help) {
              HelpShow(21);
              Help = 0;
            }
            else {
              ibwait(bd,TIMO|SRQI);
              if (ibsta & ERR)
                        gpiberr("Timeout Error");
```

```
                                }
                                break;
                        default:
                                fprintf(stderr, "Comando desconhecido.\n");
                                break;
                }
        else{
         if (DeviceNumber)
                command(&comando[0], 0);
         else{
                trocafimambos(comando);
                ptr    = &comando[0];
                ptraux = ptr;
                len = 0;
                FimDeCmd = 0;
                for (i = 0; i < 6; i++)
                 gpibcmd[i] = '\0';
                if (*ptraux){
                 do{
                        while ((*ptraux != ' ')&&(*ptraux != '\0')) ptraux ++;
                        if (*ptraux == '\0') FimDeCmd = 1;
                        *ptraux = '\0';
                        gpibcmd[len] = (char)atoi(ptr);
                        len ++;
                        ptr = ptraux+1;
                        ptraux = ptr;
                 }while (!FimDeCmd);
                 ibsic (bd);
                 if (ibsta & ERR)
                        gpiberr("ibsic Error");
                 ibsre (bd,1);
                 if (ibsta & ERR)
                        gpiberr("ibsre Error");
                 ibcmd(bd,gpibcmd,len);
                }
         }
        }


        if ((RepeatCnt != 0) && (RepeatMode != 0)){
                RepeatCnt --;
                RepeatIndexValue ++;
        }
        } while(RepeatCnt && RepeatMode);
}

 textbackground(BLACK);
 textattr(WhiteBlack);
 clrscr();
 ibonl(bd, 0);
}
/*****************************************************************************/
void start(void){
 bd = ibfind ("GPIB0");
 if (ibsta & ERR)
        gpiberr("ibfind Error");

 ibsic (bd);
 if (ibsta & ERR)
```

```
        gpiberr("ibsic Error");

 ibsre (bd,1);
 if (ibsta & ERR)
        gpiberr("ibsre Error");
}
/*************************************************************************/
void load(char *file)
{
 FILE *fptr;
 char gpibcmd[6], comando[LineLength], *ptr;
 int len, i;

 start();
 trocafim(file);
 if(!(fptr = fopen(file,"rb"))){
        fprintf(stderr,"Arquivo nÆo encontrado : %s\n", file);
        return;
 }


 /* sprintf(gpibcmd, "%c%c%c%c",
LLO,ListAddBase+DeviceNumber,DCL,TalkAddBase+ControlerNumber);*/
 ibcmd (bd,gpibcmd,4);
 if (ibsta & ERR)
        gpiberr("ibcmd Error");


 while(fgets(&comando[0],LineLength,fptr)){
        len = 0; ptr = &comando[0];
        while(*(ptr++) != '\0')
                len++;
        comando[len-2] = '\0';
        fprintf(stderr,"%s\n",comando);
        sprintf(gpibcmd, "%c%c", ListAddBase+DeviceNumber,TalkAddBase+ControlerNumber);
        ibcmd (bd,gpibcmd,2);
        ibwrt (bd, comando, len-2);
        if (ibsta & ERR)
                gpiberr("Erro na comunica‡Æo");
 }

 fclose(fptr);

 ibonl(bd, 0);

}
/*************************************************************************/
void command(char *cmd, int query)
{
 FILE *fptr;
 char comando[LineLength], gpibcmd[6], *ptr;
 int len, i, pergunta = 0;

 start();

 /* Ta faltando do DCL */
 sprintf(gpibcmd, "%c%c%c", LLO,ListAddBase+DeviceNumber,TalkAddBase+ControlerNumber);
 ibcmd (bd,gpibcmd,3);
 if (ibsta & ERR)
```

```
        gpiberr("ibcmd Error");

 ptr = cmd; len = 0;
 while(*(ptr++)!='\0'){
        pergunta = (*ptr=='?')?(1):(pergunta);
        len ++;
 }

 sprintf(gpibcmd, "%c%c", ListAddBase+DeviceNumber,TalkAddBase+ControlerNumber);
 ibcmd (bd,gpibcmd,2);
 ibwrt (bd, cmd, len);
 if (ibsta & ERR)
        gpiberr("Erro na comunica‡Æo");


 if (pergunta || query) {
        for(i = 0; i++; i<1000);

        sprintf(gpibcmd, "%c%c%c%c%c",
UNL,UNT,TalkAddBase+DeviceNumber,ListAddBase+ControlerNumber);
        ibcmd (bd,gpibcmd,4);

        for (i = 0; i < LineLength; i++)
                resp[i] = '\0';
        ibrd(bd,resp,LineLength);
        if (ibsta & ERR)
                gpiberr("Erro na comunica‡Æo");
        trocafim(resp);
        fprintf(out,"%s\n",resp);
 }
 ibonl(bd, 0);
}
/*****************************************************************************/
void  show(void)
{
 char comando[LineLength], gpibcmd[6], *ptr, aux[31];
 int len, i;

 start();
 sprintf(gpibcmd, "%c%c%c%c%c", LLO,ListAddBase+DeviceNumber,DCL,TalkAddBase+ControlerNumber);
 ibcmd (bd,gpibcmd,4);

 if (ibsta & ERR)
        gpiberr("ibcmd Error");

 for (i = 1; i < 17; i++){
                sprintf(aux, "MRDO?%d", i);
                sprintf(gpibcmd, "%c%c", ListAddBase+DeviceNumber,TalkAddBase+ControlerNumber);
                ibcmd (bd,gpibcmd,2);
                ibwrt(bd,aux,(i>9)?7:6);

                sprintf(gpibcmd, "%c%c%c%c%c",
UNL,UNT,TalkAddBase+DeviceNumber,ListAddBase+ControlerNumber);
                ibcmd (bd,gpibcmd,4);

                ibrd(bd,resp,75);
                ptr = &resp[0];
                while(*(ptr++) != '\"');
                fprintf(out,"%s%s\n",espaco,ptr);
```

```
  }
  ibonl(bd, 0);
}
/*****************************************************************************/
void gpiberr(char *msg)
{
        fprintf (stderr,"%s\n", msg);
        fprintf (stderr,"ibsta = &H%x  <", ibsta);
        if (ibsta & ERR )
                fprintf (stderr," ERR");
        if (ibsta & TIMO)
                fprintf (stderr," TIMO");
        if (ibsta & END )
                fprintf (stderr," END");
        if (ibsta & SRQI)
                fprintf (stderr," SRQI");
        if (ibsta & RQS )
                fprintf (stderr," RQS");
        if (ibsta & CMPL)
                fprintf (stderr," CMPL");
        if (ibsta & LOK )
                fprintf (stderr," LOK");
        if (ibsta & REM )
                fprintf (stderr," REM");
        if (ibsta & CIC )
                fprintf (stderr," CIC");
        if (ibsta & ATN )
                fprintf (stderr," ATN");
        if (ibsta & TACS)
                fprintf (stderr," TACS");
        if (ibsta & LACS)
                fprintf (stderr," LACS");
        if (ibsta & DTAS)
                fprintf (stderr," DTAS");
        if (ibsta & DCAS)
                fprintf (stderr," DCAS");
        fprintf (stderr," >\n");
        fprintf (stderr,"iberr = %d", iberr);
        if (iberr == EDVR)
                fprintf (stderr," EDVR <DOS Error>\n");
        if (iberr == ECIC)
                fprintf (stderr," ECIC <Not CIC>\n");
        if (iberr == ENOL)
                fprintf (stderr," ENOL <No Listener>\n");
        if (iberr == EADR)
                fprintf (stderr," EADR <Address error>\n");
        if (iberr == EARG)
                fprintf (stderr," EARG <Invalid argument>\n");
        if (iberr == ESAC)
                fprintf (stderr," ESAC <Not Sys Ctrlr>\n");
        if (iberr == EABO)
                fprintf (stderr," EABO <Op. aborted>\n");
        if (iberr == ENEB)
                fprintf (stderr," ENEB <No GPIB board>\n");
        if (iberr == EOIP)
                fprintf (stderr," EOIP <Async I/O in prg>\n");
        if (iberr == ECAP)
                fprintf (stderr," ECAP <No capability>\n");
        if (iberr == EFSO)
```

```
                fprintf (stderr," EFSO <File sys. error>\n");
        if (iberr == EBUS)
                fprintf (stderr," EBUS <Command error>\n");
        if (iberr == ESTB)
                fprintf (stderr," ESTB <Status byte lost>\n");
        if (iberr == ESRQ)
                fprintf (stderr," ESRQ <SRQ stuck on>\n");
        if (iberr == ETAB)
                fprintf (stderr," ETAB <Table Overflow>\n");
        fprintf (stderr,"ibcnt = %d\n", ibcnt);
        fprintf (stderr,"\n");

}
/*****************************************************************************/
void trocachar(char * string, char in, char out)
{
 while (*(string+1) != '\0')
        while(*(string++) != out);
          *(--string) = in;
}
/*****************************************************************************/
void trocafim(char * arquivo)
{
        while(*(arquivo++) != '\n');
        *(--arquivo) = '\0';
}
/*****************************************************************************/
void trocafimdos(char * arquivo)
{
        while(*(arquivo++) != '\r');
        *(--arquivo) = '\0';
}
/*****************************************************************************/
void trocafimambos(char * arquivo)
{
        while((*(arquivo) != '\r')&&(*(arquivo) != '\n')) arquivo++;
        *(arquivo) = '\0';
}
/*****************************************************************************/
char ** LeConf(char *InArq)
{
 FILE *in;
 char **Out;
 char *aux;
 int i, DevNumber;

 if (!(in = fopen(InArq, "rb"))){
        fprintf(stderr, "NÆo achei arquivo com configura‡Æo da Rede.\n");
        exit(1);
 }

 if (!(Out = calloc(sizeof(char *), MaxDevNumber+1))){
        fprintf(stderr,"Lamento, nÆo h  memória para a opera‡Æo\n");
        exit(1);
 }

 for (i = 0; i < MaxDevNumber; i++)
        Out[i] = &NoUsedDev[0];
```

```
 while(!feof(in)){
        if (!(aux = calloc(sizeof(char), GPIBNetConfLineLength))){
                fprintf(stderr,"Lamento, nÆo h  memçria para a opera‡Æo\n");
                exit(1);
        }
        fgets(aux,GPIBNetConfLineLength,in);
        DevNumber = (aux[0] - '0')*10 + (aux[1] - '0');
        trocafimambos(aux);
        Out[DevNumber] = &aux[3];
        while (isvarchar(*aux)) aux ++;
        *aux = '\0';
 }

 fclose(in);

 return Out;
}
/*****************************************************************************/
void PreProc(char * In, char * Out)
{
 int Ini, Outi, i, VarNumber;
 char * ptr, * ptraux;
 char Name[MaxVarNameLength];
 char AuxOut[256];
 int  AuxInd;

 trocafimambos(In);
 for (i = 0; i < LineLength; i++)
        Out[i] = '\0';

 ptr = In;
 Ini = 0;
 Outi = 0;
 while (*ptr != '\0'){
        while (!isspechar(*ptr)){ /* Caracteres especiais \, %, e \0 */
          Out[Outi++] = In[Ini++];
          ptr ++;
        }
        if (*ptr == '\0')
          Out[Outi] = *ptr;

        switch(*ptr){
          case '\\': /* Caracter de escape */
                Ini += 2;
                switch(*(ptr+1)){
                  case '%': /* Caracter identificador de vari vel*/
                        Out[Outi] = '%';
                        break;
                  case '\\': /* Caracter de escape */
                        Out[Outi] = '\\';
                        break;
                  case 'b': /* Beep */
                        Out[Outi] = '\b';
                        break;
                  case 'n': /* Nova Linha */
                        Out[Outi] = '\n';
                        break;
                  case 'r': /* Retorno de Carro */
                        Out[Outi] = '\r';
```

```
                        break;
                case 't': /* Tabula‡Æo */
                        Out[Outi] = '\t';
                        break;
                default:
                        break;
                }
                ptr +=2;
                Outi++;
                break;
        case '%': /* Identificador de Vari vel */
                ptr++; Ini++; i = 0;
                while(islet(*ptr)){
                 Name[i++] = *ptr++;
                 Ini ++;
                }
                Name[i] = '\0';
                VarNumber = -1;
                for (i = 0; i < NextFreeVar; i++)
                 if (!strcmp(Name, VarName[i]))
                        VarNumber = i;
                if (VarNumber == -1){
                 Out[0] = '\0';
                 fprintf(stderr, "Vari vel nÆo declarada : %s\n", Name);
                 *ptr = '\0';
                }
                else{
                 if (RepeatMode)
                        AuxInd = RepeatIndexValue;
                 else
                        if (RptFileMode)
                          AuxInd = MaxRptInd - RptFileMode;
                        else
                          if (RepeatLoop)
                                AuxInd = MaxRptInd - RepeatLoop;
                          else
                                AuxInd = 0;
                 strcpy(AuxOut, Out);
                 sprintf(Out, "%s%e", AuxOut, VarValue[VarNumber][AuxInd+1]);
                 ptraux = &Out[0];
                 Outi = 0;
                 while (*ptraux != '\0') {
                        ptraux ++;
                        Outi++;
                 }
                }
                break;
         }
        }
}

/*****************************************************************/
void HelpShow(int ComandoIndex)
{
 int i, j;

        j = wherey();
        for (i = 0; i < HelpLength[ComandoIndex] + 2 ; i++)
         fprintf(stderr, "\n");
```

```
        if (j >= 25 - HelpLength[ComandoIndex] - 2)
          gotoxy(1, 25 - HelpLength[ComandoIndex] - 2);
        else
          gotoxy(1,j);
        textattr(RedGray);
        cprintf("ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ");
        cprintf("%s", HelpText[ComandoIndex]);
        sprintf(prompt, "%s> ", ListaDev[DeviceNumber]);
        cprintf("ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ");
        textattr(YellowBlue);
}
/*****************************************************************/
void InitHelp(void)
{
 int i;
 i = 0;
 HelpLength[i] = NumeroLines; HelpText[i++] = AjudaNumero;
 HelpLength[i] = ComentLines; HelpText[i++] = AjudaComent;
 HelpLength[i] = PergunLines; HelpText[i++] = AjudaPergun;
 HelpLength[i] = EspacoLines; HelpText[i++] = AjudaEspaco;
 HelpLength[i] = ComuniLines; HelpText[i++] = AjudaComuni;
 HelpLength[i] = DevLines;    HelpText[i++] = AjudaDev;
 HelpLength[i] = EchoLines;   HelpText[i++] = AjudaEcho;
 HelpLength[i] = EndLines;    HelpText[i++] = AjudaEnd;
 HelpLength[i] = FreeLines;   HelpText[i++] = AjudaFree;
 HelpLength[i] = HelpLines;   HelpText[i++] = AjudaHelp;
 HelpLength[i] = LetLines;    HelpText[i++] = AjudaLet;
 HelpLength[i] = LoadLines;   HelpText[i++] = AjudaLoad;
 HelpLength[i] = NetLines;    HelpText[i++] = AjudaNet;
 HelpLength[i] = OutLines;    HelpText[i++] = AjudaOut;
 HelpLength[i] = PauseLines;  HelpText[i++] = AjudaPause;
 HelpLength[i] = PrintLines;  HelpText[i++] = AjudaPrint;
 HelpLength[i] = QuitLines;   HelpText[i++] = AjudaQuit;
 HelpLength[i] = RepeatLines; HelpText[i++] = AjudaRepeat;
 HelpLength[i] = RunLines;    HelpText[i++] = AjudaRun;
 HelpLength[i] = TableLines;  HelpText[i++] = AjudaTable;
 HelpLength[i] = TimeLines;   HelpText[i++] = AjudaTime;
 HelpLength[i] = WaitLines;   HelpText[i++] = AjudaWait;
}

void FILEcpy(FILE *In, FILE *Out)
{
  Out->level  = In->level;
  Out->flags  = In->flags;
  Out->fd     = In->fd;
  Out->hold   = In->hold;
  Out->bsize  = In->bsize;
  Out->buffer = In->buffer;
  Out->curp   = In->curp;
  Out->istemp = In->istemp;
  Out->token  = In->token;
}
```

A.2 Ca2432

```
!# Ca2432
!# tst para a avaliar a calibracão do canal vertical de um osc. Tek2432
!# conectado a uma rede gpib pelo endereco 3.
!# Usa-se um gerador de calibração CG5010 conectado ao endereço 15

!# Direciona osc. Tek2432
!dev tek2432a
message 9:"        calibrando o 2432 ..."
message 11:"       \l\a\b\o\r\a\t\o\r\i\o \d\e \c\a\l\i\b\r\a\c\a\o"
message 10:"           UNICAMP 1998"
run acquire
acquire mode:normal
ch1 volts:2e-3
ch1 position:-3
hor ase:5e-4
!# Direciona gen. CG5050
!dev cg5050
mode v
u/d 2e-3
mult 6
out on
tring on
opc on
!dev tek2432a
data source:ch1
value? pk2
message 9:"        Obrigado e volte sempre"
!end
```

A.3 Cal02

```
!# Cal02,Teste para avaliar o processo de repetição das medições a partir da
!# declaração da Variaveis de escala. Realiza a calibração parcial de dois osciloscopios.
!# O arquivo pega os valores da primera columna como
!# valor para executar a calibração com no arquivo cal1 e cal2

!free
!let esco [1e-3, 2e-3, 5e-3, 10e-3]
!let escg [1e-3, 2e-3, 5e-3, 10e-3]

!out reporter.rpt

!print :\n\tResultado da aferição dos osciloscópios TEK2432a\n
!print :\te TEK2440.\n\n\n\tCanal 1, Vertical

!dev tek2440
  message 9:"       Aferindo o 2440 Canal 1..."
  message 11:"      \\f\\a\\b\\o\\r\\a\\t\\o\\r\\i\\o \\d\\e \\c\\a\\l\\i\\b\\r\\a\\c\\a\\o"
  message 10:"       LACE.DECOM.UNICAMP 1997"
  vmode ch1:on
  vmode add:off
  vmode inv:off
  vmode ch2:off
  run acquire
  ch1 cou:dc,position:-3
  acquire mode:normal
  hor ase:5e-4


!dev tek2432a
  message 9:"       Aferindo o 2432a Canal 1..."
  message 11:"      \\f\\a\\b\\o\\r\\a\\t\\o\\r\\i\\o \\d\\e \\c\\a\\l\\i\\b\\r\\a\\c\\a\\o"
  message 10:"       LACE.DECOM.UNICAMP 1997"
  vmode ch1:on
  vmode add:off
  vmode inv:off

  vmode ch2:off
  run acquire
  ch1 cou:dc,position:-3
  acquire mode:normal
  hor ase:5e-4

!dev cg5010B
  mode v
  mult 6
  out on
  tring on
  opc on


!rep 4

!run cal1.bat

!free
!let esco [20e-3, 50e-3, 100e-3, 200e-3]
```

```
!let escg [20e-3, 50e-3, 100e-3, 200e-3]

!rep 4
!run cal1.bat


!free
!let esco [500e-3, 1, 2, 5]
!let escg [500e-3, 1, 2, 5]

!rep 4
!run cal1.bat

!print :\n\n\tMedidas Encerradas para o Canal 1.
!dev Tek2440
 message 9:"   medidas encerradas para o canal 1"

!dev Tek2432a
 message 9:"   medidas encerradas para o canal 1  "

!dev cg5010B
 out off

!free
!let esco [1e-3, 2e-3, 5e-3, 10e-3]
!let escg [1e-3, 2e-3, 5e-3, 10e-3]

!print :\n\n\tCanal 2, Vertical

!dev tek2440
 message 9:"       Aferindo o 2440 Canal 2..."
 message 11:"      \\l\\a\\b\\o\\r\\a\\t\\o\\r\\i\\o \\d\\e \\c\\a\\l\\i\\b\\r\\a\\c\\a\\o"
 message 10:"       LACE.DECOM.UNICAMP 1997"
 vmode ch1:off
 vmode add:off
 vmode inv:off

 vmode ch2:on
 run acquire
 ch2 cou:dc,position:-3
 acquire mode:normal
 hor ase:5e-4


!dev tek2432a
 message 9:"       Aferindo o 2432a Canal 2..."
 message 11:"      \\l\\a\\b\\o\\r\\a\\t\\o\\r\\i\\o \\d\\e \\c\\a\\l\\i\\b\\r\\a\\c\\a\\o"
 message 10:"       LACE.DECOM.UNICAMP 1997"
 vmode ch1:off
 vmode add:off
 vmode inv:off

 vmode ch2:on
 run acquire
 ch2 cou:dc,position:-3
 acquire mode:normal
 hor ase:5e-4

!dev cg5010A
```

```
 mode v
 mult 6
 out on
 tring on
 opc on

!rep 4
!run cal2.bat

!free
!let esco [20e-3, 50e-3, 100e-3, 200e-3]
!let escg [20e-3, 50e-3, 100e-3, 200e-3]

!rep 4
!run cal2.bat


!free
!let esco [500e-3, 1, 2, 5]
!let escg [500e-3, 1, 2, 5]

!rep 4
!run cal2.bat

!print :\n\n\tMedidas Encerradas para o Canal 2.
!dev Tek2440
 message 9:"   medidas encerradas para o canal 2"

!dev Tek2432a
 message 9:"   medidas encerradas para o canal 2  "

!dev cg5010A
 out off


!out off

! edit reporter.rpt

!end
```

A.4 Cal1

```
!# Arquivo chamado desde o Cal02

!print :\n\n\tEscala Osc. => %esco\tEscala Gen. => %escg\n\n
!# Direciona osc. Tek2432a
!dev Tek2440
  ch1 volts:%esco
!dev tek2432a
  ch1 volts:%esco

!# Direciona gen. CG5010B
!dev cg5010B
  u/d %escg

!dev tek2440
  data source:ch1
  !print:\n\t\t
  id?
  !print:\t\t\t
  value? pk2


  !print:\t\t\t
  value? pk2


  !print:\t\t\t
  value? pk2

!dev tek2432a
  data source:ch1
  !print:\n\t\t
  id?
  !print:\t\t\t
  value? pk2

  !print:\t\t\t
  value? pk2

  !print:\t\t\t
  value? pk2

!end
```

A.5 Cal2

```
!# Arquivo chamado desde o Cal02

!print :\n\n\tEscala Osc. => %esco\tEscala Gen. => %escg\n\n
!# Direciona osc. Tek2432a
!dev Tek2440
  ch2 volts:%esco
!dev tek2432a
  ch2 volts:%esco

!# Direciona gen. CG5010A
!dev cg5010A
  u/d %escg

!dev tek2440
  data source:ch2
  !print:\n\t\t
  id?
  !print:\t\t\t
  value? pk2
  !print:\t\t\t
  value? pk2
  !print:\t\t\t
  value? pk2

!dev tek2432a
  data source:ch2
  !print:\n\t\t
  id?
  !print:\t\t\t
  value? pk2
  !print:\t\t\t
  value? pk2
  !print:\t\t\t
  value? pk2

!end
```

A.6 Reporter, teste para avaliar o direcionado da saída padrão para um arquivo

Resultado da aferição dos osciloscópios TEK2432a
e TEK2440.

Canal 1, Vertical

Escala Osc. => 0.001000 Escala Gen. => 0.001000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.5600E-3
VALUE PK2PK:6.7200E-3
VALUE PK2PK:6.6400E-3

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:7.5200E-3
VALUE PK2PK:7.6000E-3
VALUE PK2PK:7.3600E-3

Escala Osc. => 0.020000 Escala Gen. => 0.020000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2320E-1
VALUE PK2PK:1.2240E-1
VALUE PK2PK:1.2320E-1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.3920E-1
VALUE PK2PK:1.3920E-1
VALUE PK2PK:1.3920E-1

Escala Osc. => 0.050000 Escala Gen. => 0.050000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:3.0800E-1
VALUE PK2PK:3.0600E-1
VALUE PK2PK:3.0600E-1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:3.4400E-1
VALUE PK2PK:3.4800E-1
VALUE PK2PK:3.4600E-1

Escala Osc. => 0.100000 Escala Gen. => 0.100000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.1600E-1
VALUE PK2PK:6.1600E-1
VALUE PK2PK:6.2000E-1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:6.9200E-1
VALUE PK2PK:6.8800E-1
VALUE PK2PK:6.8800E-1

Escala Osc. => 0.200000 Escala Gen. => 0.200000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2240
VALUE PK2PK:1.2240
VALUE PK2PK:1.2320

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.3760
VALUE PK2PK:1.3760
VALUE PK2PK:1.3760

Escala Osc. => 0.500000 Escala Gen. => 0.500000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:3.0800
VALUE PK2PK:3.0400
VALUE PK2PK:3.0400

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:3.4400
VALUE PK2PK:3.4400
VALUE PK2PK:3.4400

Escala Osc. => 1.000000 Escala Gen. => 1.000000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.2400
VALUE PK2PK:6.1600
VALUE PK2PK:6.1600

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:6.8800
VALUE PK2PK:6.9200
VALUE PK2PK:6.8800

Escala Osc. => 2.000000 Escala Gen. => 2.000000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2240E+1
VALUE PK2PK:1.2320E+1
VALUE PK2PK:1.2240E+1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.3760E+1

VALUE PK2PK:1.3760E+1
VALUE PK2PK:1.3760E+1

Escala Osc. => 5.000000 Escala Gen. => 5.000000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:3.0600E+1
VALUE PK2PK:3.0600E+1
VALUE PK2PK:3.0600E+1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:3.4400E+1
VALUE PK2PK:3.4400E+1
VALUE PK2PK:3.4600E+1

Medidas Encerradas para o Canal 1.

Canal 2, Vertical

Escala Osc. => 0.001000 Escala Gen. => 0.001000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.4800E-3
VALUE PK2PK:6.5600E-3
VALUE PK2PK:6.5600E-3

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:7.6800E-3
VALUE PK2PK:7.5200E-3
VALUE PK2PK:7.7600E-3

Escala Osc. => 0.002000 Escala Gen. => 0.002000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2640E-2
VALUE PK2PK:1.2560E-2
VALUE PK2PK:1.2480E-2

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.4480E-2
VALUE PK2PK:1.4480E-2
VALUE PK2PK:1.4400E-2

Escala Osc. => 0.005000 Escala Gen. => 0.005000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:3.0400E-2
VALUE PK2PK:3.0600E-2
VALUE PK2PK:3.0600E-2

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"

VALUE PK2PK:3.5200E-2
VALUE PK2PK:3.5200E-2
VALUE PK2PK:3.5400E-2

Escala Osc. => 0.010000 Escala Gen. => 0.010000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.0800E-2
VALUE PK2PK:6.0800E-2
VALUE PK2PK:6.0800E-2

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:7.0000E-2
VALUE PK2PK:7.0000E-2
VALUE PK2PK:7.0000E-2

Escala Osc. => 0.020000 Escala Gen. => 0.020000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2160E-1
VALUE PK2PK:1.2240E-1
VALUE PK2PK:1.2240E-1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.4000E-1
VALUE PK2PK:1.4080E-1
VALUE PK2PK:1.4080E-1

Escala Osc. => 0.050000 Escala Gen. => 0.050000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:3.0200E-1
VALUE PK2PK:3.0400E-1
VALUE PK2PK:3.0400E-1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:3.5200E-1
VALUE PK2PK:3.5000E-1
VALUE PK2PK:3.5000E-1

Escala Osc. => 0.100000 Escala Gen. => 0.100000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.0800E-1
VALUE PK2PK:6.1200E-1
VALUE PK2PK:6.1200E-1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:7.0000E-1
VALUE PK2PK:7.0000E-1
VALUE PK2PK:7.0400E-1

Escala Osc. => 0.200000 Escala Gen. => 0.200000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2160
VALUE PK2PK:1.2160
VALUE PK2PK:1.2080

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.3920
VALUE PK2PK:1.3920
VALUE PK2PK:1.3920

Escala Osc. => 0.500000 Escala Gen. => 0.500000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:3.0400
VALUE PK2PK:3.0400
VALUE PK2PK:3.0400

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:3.4800
VALUE PK2PK:3.5000
VALUE PK2PK:3.5000

Escala Osc. => 1.000000 Escala Gen. => 1.000000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:6.0800
VALUE PK2PK:6.1200
VALUE PK2PK:6.0800

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:7.0400
VALUE PK2PK:7.0000
VALUE PK2PK:7.0400

Escala Osc. => 2.000000 Escala Gen. => 2.000000

ID TEK/2440,V81.1,"11-DEC-89 V2.30 /2.5",TVTRIG
VALUE PK2PK:1.2160E+1
VALUE PK2PK:1.2240E+1
VALUE PK2PK:1.2160E+1

ID TEK/2432A,V81.1,"24-DEC-89 V2.30 /2.5"
VALUE PK2PK:1.4080E+1
VALUE PK2PK:1.4080E+1
VALUE PK2PK:1.4000E+1

Escala Osc. => 5.000000 Escala Gen. => 5.000000

```
ID TEK/2440,V81.1,"11-DEC-89  V2.30 /2.5",TVTRIG
        VALUE PK2PK:3.0600E+1
        VALUE PK2PK:3.0400E+1
        VALUE PK2PK:3.0400E+1

ID TEK/2432A,V81.1,"24-DEC-89  V2.30 /2.5"
        VALUE PK2PK:3.5400E+1
        VALUE PK2PK:3.5400E+1
        VALUE PK2PK:3.5400E+1
```

Medidas Encerradas para o Canal 2.

Referências Bibliográficas.

[Anderson, 1989]. Anderson Peter H. ***"GPIB controlled EE lab - Three year of experience".*** In: 1989 Frontiers in Education Conference. Binghamton, Oct. 1989. CD-ROM.

[ANSI & IEEE, 1987]. American National Standard Institute [ANSI], Institute of Electrical and Electronics Engenieers [IEEE]. ANSI / IEEE 488.1-1987. ***"IEEE Standard Digital Interface for Programmable Instrumentations"*** New York, 1987.

[ANSI & IEEE, 1992]. American National Standard Institute [ANSI], Institute of Electrical and Electronics Engenieers [IEEE].ANSI / IEEE 488.2-1992. ***"IEEE Codes, Formats, Protocolos, and Common Comands, and Standard Commands for Programmable Instruments"*** New York, 1992.

[Apple, Caltech & Satanford 1994]. Apple, Caltech, Stanford. ***"Matlab GPIB [and other] Code".*** Disponível na Internet. http://www.pcmp.caltech.edu/matlab. 2 Outubro 1996.

[Campell, 1996]. Campell T. P. ***"Instalando Redes em Pequenas e Médias Empresas. Resolvendo os Problemas de Redes em pequenos é Médios Ambientes"*** São Paulo: Makron Book,1996. 343 p.

[Cautero, 1994]. Cautero G. ***"LabView - based control system for a surface science experimental station".*** Measurement Science & Technology, New York, v.5 n.8, p. 1002-1011, Aug. 1994.

[Collier, 1988]. Collier R. ***"Accuracy and Precision".*** Industrial Reasarch/Development, Springfield, v.44, n8, p. 81-83, Abril 1988.

[Crouch & Golla, 1997]. Crouch G. , Golla L. ***"Configuring Your Driver to Match Your System".*** Disponível na Internet. http://femto.ssp.ameslab.gov/os2/mdos. 14 Março 1997.

[Crouch, 1997]. Crouch G. ***"Programmatic File transfers over the GPIB".*** Disponível na Internet. http://femto.ssp.ameslab.gov/os2/mdos. 14 Maio 1997.

[De Medeiros, 1990]. ***"Aspectos Metrológicos da Qualidade".*** Curso de especialização em Engenharia da Qualidade. Pontifica Universidade Católica, Rio Grande do Sul, 1990. p. 54-115.

[De Medeiros, 1993]. De Medeiros A. ***"Metrologia. Fundamentos e Princípios".*** Curso de especialização em Engenharia da Qualidade. Pontifica Universidade Católica, Rio Grande do Sul. 1993. p 89-96.

[Dehne, 1992]. Dehne T. ***"IEEE - 488.2 - Increasing test system compatibility and productivity".*** In: NEPCON West '92 Conference. Anaheim, Feb. 1992. CD-ROM.

[Dehne, 1997]. Dehne T. ***"Programming with NI - 488 ® Software".*** Disponível na Internet. http://www.natinst.com/gpib.html. 11 Março 1997.

[Dietel & Dietel, 1996]. Deitel M. H. , Deitel J. P. ***"C How to Program".*** Second Edition. New Jersey: Pretice Hall, 1996. 926 p.

[Fluke, 1989]. Dehne T. ***"Application Software Choices for PC- based instrument control".*** In: CONFERENCE, Wescon '89. San Francisco, Novembro 1989. CD-ROM.

[Fluke, 1996]. Fluke: MET/CAL. ***"Calibration Software".*** Manual do Usuário. Novembro, 1996. p. 1.1-9.9.

[Fluker, 1993]. Fluker. ***"Calibration: Philosophy in Practique".*** New York: Markon Book, 1993. Cap. 17: Calibrators and Calibration. Second Edition. p. 3-18.

[Gao, Spiriti & Tortora, 1994]. Gao Z. , Spiriti E. , Tortora L. ***"ADC test system based on the Macintosh Computer".*** In: Proceeding of the 1993 IEEE 8th Conference on Real-Time Computer Applications in Nuclear, Particle, and Plasma Physics[RT '93]. Feb.1994. CD-ROM.

[Golla & Thonson, 1993]. Golla L. , Thomson A. ***"High - speed GPIB with a wink".*** Evaluation Engineering, New York v.32 n.8 p. 62-63, Aug. 1993.

[Golla, 1997]. Golla L. ***"Dynamic System Configuration Using NI - 488_Routines".*** Disponível na Internet. http://femto.ssp.ameslab.gov/os2/mdos. 14 Março 1997.

[Golla-I, 1997]. Golla L. ***"Serial Polling and SRQ Servising with NI - 488.2_Software".*** Disponível na Internet. http://femto.ssp.ameslab.gov/os2/mdos. 16 Março 1997.

[Grant, 1997]. Grant McNinch W. ***"National Instruments GPIB and PARC Equipment FAQ".*** Disponível na Internet. http://www.egginc.com/bin/webmate/inst139/page/egg/gpibfaq. 8 Abril 1997.

[Grimberg, 1990]. Grimberg J. P. ***"Digital Instrument Course: IEC Bus Interface"*** Twente University of Technology, Departament of Electrical Engeneering Enschede The Netherlands. 1990. p. 5-109.

[Haque & Smith, 1995]. Haque Syed E. , Smith Dean L. ***"Test program for the Hewlett - Packard 5006A signature anlyzer".*** In: Proceeding of the IEEE Southeastcon '95 Conference. Raleigh, 1995. CD-ROM.

[Hewlett, 1988]. Hewlett Packard. ***"Description of the Hewlett-Packard Interface Bus"*** Tutorial. São Paulo: Designed HP-IB Systems, 1988.106 p.

[Holley, 1997]. Holley D. ***"Develop OPC Device Server Using New Toolkit".*** AutomationView, Austin,Texas, v.2, n.3, p.4-5, Outono 1997.

[Inmetro, 1994]. Instituto Nacional de Metrologia, Normalização e Qualidade Industrial, INMETRO. ***"Sumário dos Requisitos de Aferições e Medições Imposto pela ISO 9000 ".*** Rio de Janeiro, 1994. p. 54-58.

[Intel, 1990]. Intel. ***"8291 GPIB Talker/Listener".*** Manual. São Paulo, 1990. p.202-215.

[Jones, 1995]. Jones Eric D. , Preppernau Bryan L. ***"Catching the right bus V: a beginners` guide to programming the IEEE - 488 BUS".*** .Computer in Phisics, New York, v.9 n.2, p. 140-147, Mar-Apr. 1995.

[Langsdorff & Lafin, 1995]. Langsdorff Sanches E. , Lafin M. ***"Manual da Qualidade para Laboratórios".*** Curso de especialização. Instituto Brasileiro de Petróleo. São Paulo, Agosto, 1995. 43 p.

[Loew, 1997]. Loew Ch. ***"Software Components Bring Flexible, Powerful Solutions".*** Instrumentation Newsletter. Especial Academic Edition Austin,Texas, v.3, n.3, p. 5, Outono 1997.

[Machado, 1992]. Machado Jorge H. ***"Metrology and Quality Assurance in Europe. The Role of International and European Standards".*** In: 1st. Intl. Symposium on Electrical Metrology and the ISO 9000. São José dos Campos, Novembro, 1992 . 296 p.

[Marinov, 1992]. Marinov Stojan M. ***"Use of GPIB in microcomputer - controlled data acquisition systems for lidar measuremnts".*** SPIE - The International Society for Optical Engineering, Bellingham,v.1714 p. 326 - 333. Nov.1992.

[Mullins, 1992]. Mullins Mark F. ***"Building Systems with GPIB [IEEE - 488]".*** In: NEPCON West '92 Conference. Anaheim, Feb. 1992. CD-ROM.

[Narasimhan, Rangarajan & Rahman, 1990]. Narasimhan S. V. , Rangarajan S. , Rahman M. N. K. ***"Basic program for the transfers of electrochemical data from a digital oscilloscope to a personal computer through GPIB interface".*** Analitical Instrumentation, New York v.19 n.2-3 p. 141-159, Jun. - Sep. 1990.

[National, 1997]. National Instruments. ***"GPIB. Tutorial".*** Disponível na Internet. http://www.natinst.com/gpib.htm. 10 Outubro, 1997.

[National-I, 1997]. National Instruments. ***"LabVIEW Básico".*** Curso. Austin, Texas, Outubro 1997.

[National-II, 1997]. National Instruments. Productos: software. ***"GPIB Driver software".*** Disponível na Internet. http://www.natinst.com/ieee488gpib.html. 13 Outubro 1997.

[National-III, 1997]. National Instruments. Productos: GPIB. ***"Low - Cost Interface for PC/XT. GPIB-PCII/IIA"*** Disponível na Internet. http://www.natinst.com/gpib.html. 13 Outubro 1997.

[Ozkul, 1994]. Ozkul T. ***"New Experimental enhancement to GPIB standard: Geral purpose serial interface network [GPSI]".*** Computer Standards & Interfaces, New York v.16 n.2 p. 133-138, Jun. 1994.

[Patel, 1997]. Patel A. ***"HS488 Instruments Accelerate Next-Generation Text Systems"*** Instrumentation Newsletter. Especial Academic Edition Austin,Texas, v.3, n.3, p.6-7, Outono 1997.

[Potter & Mill, 1988]. Potter Caren D. , Mills Robert B. ***"Soft Instrumentattion for test labs".*** CAE, Computer - Aided Enginering, New York v.7 n.11. Nov.1988. 4 p.

[Ramdas, 1997]. Ramdas M. ***"Using Your NI - 488.2 Software for Dos/Windows with OS/2 2.2X".*** Disponível na Internet. http://femto.ssp.ameslab.gov/os2/mdos/winnos2/gpib.ini. 11 Março 1997.

[SaGomes, 1997]. SaGomes C. ***"Porting NI- 488.2 and NI- 488.2 Aplications Between Platforms and Operating Systems". Tecnhical Note 028.*** Disponível na Internet. http://www.natinst.com/ieee488gpib.html. 13 Outubro 1997.

[Santori, 1990]. Santori M. ***"I/O Options from Desktop to Field".*** AutomationView, Austin,Texas, v.2, n.3, p.2-3, Outono 1997.

[Seeler, 1992]. Seeler Aaron P. ***"Desingning bus - based software for calibration systems".*** In: NEPCON West '92 Conference. Anaheim, Feb. 1992. CD-ROM.

[Shavarini, 1992]. Shavarini Hadi K. ***"PC based instrumentation, another alternative for test and measurement".*** In: NEPCON West `92 Conference. Anaheim, Feb. 1992. CD-ROM.

[Shay, 1996]. Shay A. W. ***"Sistemas Operacionais".*** São Paulo: Makron Book,1996. 758 p.

[Sistemas, 1986]. Sistemas Técnicos Digitais S.A. ***"STD - 8410. Cartão Controlador GPIB"*** Manual do Usuário. Brasília, Junho 1986. 80p.

[Tanenbaum & Woodhull, 1997]. Tanenbaum S. A. , Woodhull S. A. ***"Operating Systems. Desing and Implementation".*** Second Edition. New Jersey: Pretice Hall, 1997. 952 p.

[Tanenbaum, 1992]. Tanenbaum S. A. ***"Modern Operating Systems".*** New Jersey: Pretice Hall, 1992. 728 p.

[Tek11, 1997]. Tektronix MBD: Products ***"CG 5011 Calibrations Generators".*** Disponível na Internet. http://www.tek.com/Measurement/Products/catalog/ca. 17 Abril 1997.

[Tek12, 1997]. Tektronix MBD: Products ***"2212 Digital + Analog Oscilloscope".*** Disponível na Internet. http://www.tek.com/Measurement/Products/catalog/ca. 17 Abril 1997.

[Tek20, 1997]. Tektronix MBD: Products ***"TDS 820 Digitizing Oscilloscope".*** Disponível na Internet. http://www.tek.com/Measurement/Products/catalog/ca. 17 Abril 1997.

[Tek92, 1997]. Tektronix MBD: Products ***"2792 Spectrum Analyzer".*** Disponível na Internet. http://www.tek.com/Measurement/Products/catalog/ca. 17 Abril 1997.

[Tektronix, 1987]. Tektronix. ***"GURU II. GPIB user's resource utility for the IBM personal computer".*** Manual do Usuário. Agosto 1987.100p.

[Timmerman, 1995]. Timmerman A. ***"Implementation of a C- based mechatronics lecture and lab class".*** In: Proceeding of the 1995 ASME International Mechanical Engineering Congress & Exposition. San Francisco, 1995. CD-ROM.

[Tóth, 1996]. Tóth E. ***"Automação de Laboratórios "***.Minicurso do II. Seminário Internacional de Metrologia Elétrica. INMETRO, Brasil, 1996.52 p.

[Venkatesttwarlu, 1992]. Venkatestwarlu R. ***"Real - time evaluation techniques for real - time image processing hardware"***. In: Acquisition, Tracking, and Pointing VI Conference. Orlando, Oct. 1992. CD-ROM.

[Zollner, 1997]. Zollner S. ***"Accessing GPIB device in OS/2 WARP"***. Disponível na Internet. http://femto.ssp.ameslab.gov/os2/gpib/gpib.htm. 11 Março 1997

[Zrudsky & Pichler, 1992]. Zrudsky Donald R. , Pichler James M. ***"Virtual instruments for instantaneus power measurement"***. IEEE Transctions on Instrumentations and Measurement, New York v.41 n.4 p. 528-534, Aug. 1992.