UNIVERSIDADE ESTADUAL DE CAMPINAS - UNICAMP
FACULDADE DE ENGENHARIA ELÉTRICA - FEC
DEPARTAMENTO DE SEMICONDUTORES E FOTÔNICA - DSIF

AIDS-TME : Ambiente Interativo para Desenvolvimento de Software para Testes e Medidas Elétricas, utilizando Instrumentação com Interface GPIB (IEEE-488)

Autor: JOSÉ OTÁVIO SIMÕES
Orientador: Prof. Dr. FURIO DAMIANI

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade de Campinas como parte dos requisitos para título de Mestre em Engenharia Elétrica.

Este exemplar corresponde à redação final da tese defendida por José Otávio Simões e aprovada pela Comissão Julgadora em 25/06/91.

Orientador

Julho de 1991

AGRADECIMENTO

À todas as pessoas que me acompanharam nesta jornada e que tiveram a compreensão necessária para que eu pudesse chegar ao fim e principalmente a DEUS.

R E S U M O

O escopo deste trabalho é a especificação, implementação e teste operacional de um "Ambiente Interativo de Desenvolvimento de Software para Testes e Medidas Elétricas" - AIDS_TME, com características modulares e de alta reusabilidade de rotinas básicas na produção de software. O aumento da produtividade e da qualidade dos programas gerados dentro deste ambiente é uma característica inerente a sistemas dessa natureza (CASE).

A filosofia que norteia este desenvolvimento está baseada em anos de experiências em laboratórios e áreas de manutenção eletroeletrônica, bem como na utilização de instrumentação com interface GPIB - "GENERAL PURPOUSE INTERFACE BUS", visando atender as necessidades de produção de uma enorme variedade de programas de testes para automatizar atividades rotineiras ou para a execução de ensaios extremamente longos e complicados.

A B S T R A C T

The purpose of this work is the specification, implementation and operational testing of an Interative Environment of Software Development for Testing and Electrical Measurements (AIDS_TME), with modular characteristics and high reusability of basic routines in the software production. The increase in the productivity and quality of programms generated into this environment is an inherent characteristic to the systems of this nature (CASE).

The philosophy that guides this development is based on years of experiency in laboratories and eletronic mantenance areas as well as the utilization of instrumententationwith GPIB "General Purpose Interface Bus" interface, aiming at the attendance of necessity of production of a large variety of testing programs to automatize routine activities or to perform complex and extremely long testing

M O T I V A Ç A O

"Produtividade", "Qualidade", "Eficiência" e muitos outros termos sempre foram mencionados e repetidos nas diversas áreas e setores por onde trabalhei. Todos os profissionais com quem tive contato ou trabalho conjunto, sempre, buscavam em sua rotina atingir essas "metas" tão cobiçadas.

O reflexo desse trabalho está no esforço comum de se buscar a resposta correta, o valor verdadeiro, o procedimento fiel para problemas e projetos, individuais ou coletivos, que permitam uma realização plena e segura do(s) profissional(is) envolvido(s).

Trabalhando durante vários anos em diversas áreas pude detectar e sentir muitas vezes a necessidade de ferramentas ou dispositivos especiais e por muitas vezes dedicados que poderiam resolver o meu problema de uma maneira mais rápida e eficaz, sem causar os traumas tradicionais de uma rotina "pesada" e "cansativa".

A idéia da realização desse trabalho está embasada na rotina de trabalho de um laboratório de Eletroeletrônica, que realiza ensaios e/ou testes de caracterização de componentes e dispositivos Eletroeletrônicos.

Antes de iniciar a apresentação deste trabalho cabe, neste início, a definição dos termos "ensaios" e "testes", intensamente utilizados nos capítulos que se segue.

> - "Ensaio: Experiência; exame; reconhecimento; treinamento; prova; tentativa; tirocínio, dissertação sobre determinado assunto, mais curta e menos metódica do que um tratado formal e acabado."
>
> - "Teste: Conjunto de provas que se aplicam a indivíduos para apreciar o seu desenvolvimento mental, aptidão, etc.; provas que se executam para aferir a eficiência de equipamentos ou os efeitos de determinadas substâncias (Acomodação da palavra inglesa 'Test' = prova, eficiência).

Do dicionário Aurélio <27>, edição 1986, paginas 659 e 1671, respectivamente. Resumidamente e referenciado a área de Eletroeletrônica, "ensaios" e "testes" são muitas vezes utilizados com o mesmo significado. Neste trabalho "ensaio" é aplicado em situaçÕes e açÕes rápidas e "testes" para açÕes mais complexas e com um procedimento bem definido. Ex: " .. o ensaio de alta tensão do Teste de Comissionamento da Unidade Geradora ..."; " ... o teste de extração de parâmetros de um semi-condutor .. ".

O Ambiente Interativo de Desenvolvimento de Software para Teste e Medidas Elétricas, ou simplesmente, "AIDS_TME" tem a intensão de ser aquela "ferramenta" que à muito se desejava na produção e desenvolvimento de programas de testes e/ou ensaios na área de Eletroeletrônica, tanto para campo como laboratório.

Motivação maior ainda, e sempre será, a de se superar em desafios e projetos dessa natureza; que DEUS nos ajude.

I N D I C E

1 - INTRODUÇAO I.1

2 - SISTEMAS DE DESENVOLVIMENTO DE SOFTWARE II.1

2.1 - A PRODUÇAO DE SOFTWARE II.1
2.2 - VISUALIZAÇAO DE IDéIAS II.2
2.3 - QUANTO CUSTA UM ERRO ? II.2
2.4 - A IMPORTÂNCIA DAS INTERFACES II.3
2.5 - PROTOTIPAGEM II.4
2.6 - DESENVOLVIMENTO DE PROGRAMAS II.5
2.7 - ESTAÇAO DE TRABALHO DO PROGRAMADOR II.6
2.8 - DEPURADOR SIMBÖLICO II.6
2.9 - LINGUAGEM DE 4@ GERAÇAO (L4g) II.7
2.10 - EDIÇAO / GERéNCIA DE TELAS II.7
2.11 - MODELAGEM DE DADOS II.7
2.12 - GERADOR DE PROGRAMAS II.7
2.13 - GERéNCIA DE FONTES E DOCUMENTOS II.8
2.14 - GERéNCIA DE MÖDULOS II.8

3 - AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE III.1

3.1 - ENGENHARIA DE SOFTWARE III.2
3.1.1 - CICLO DE VIDA DE ENGENHARIA DE SOFTWARE III.2
3.1.2 - CICLO DE VIDA E TéCNICAS DE DESENVOLVIMENTO III.3
3.1.3 - TéCNICAS GERéNCIAIS III.3
3.1.4 - CONSIDERAÇöES SOBRE GERENCIAMENTO III.4
3.2 - CASE - "Computer Aided Software Engineering" III.5
3.2.1 - INTRODUÇAO AO CASE III.5
3.2.2 - ARQUITETURA CASE III.6
3.3 - FERRAMENTAS DE DESENVOLVIMENTO III.7
3.3.1 - EDITORES GRÄFICOS E DE TEXTO III.7
3.3.2 - DICIONÄRIO DE DADOS III.7
3.3.3 - ANALISADORES E VERIFICADORES III.8
3.4 - AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE SOFTWARE III.8

4 - INSTRUMENTAÇAO E LINGUAGENS DE PROGRAMAÇAO IV.1

4.1 - SISTEMA DE COMUNICAÇAO GPIB (IEEE-488) IV.1
4.2 - LINGUAGEM DE PROGRAMAÇAO QuickBASIC IV.2
4.3 - INSTRUMENTAÇAO COM INTERFACE GPIB (IEEE-488) IV.3
4.3.1 - O SOFTWARE E OS INSTRUMENTOS IV.4
4.3.2 - O HARDWARE DE COMUNICAÇAO INTERFACE IEEE-488 IV.4
4.3.3 - "DRIVER" E LINGUAGENS IV.5
4.3.4 - VALORES "DEFAULT" DOS PARÂMETROS DE SOFTWARE IV.6

5 - ESPECIFICAÇAO FUNCIONAL V.1

5.1 - INTRODUÇAO V.1
5.2 - REQUISITOS BÄSICOS V.1
5.2.1 - OBJETIVOS DO AIDS_TME V.2
5.2.2 - CONDIÇÕES NECESSÄRIAS PARA O FUNCIONAMENTO V.2
5.2.3 - USUÄRIOS V.2
5.2.4 - RESULTADOS ESPERADOS V.2
5.2.5 - DADOS FORNECIDOS V.2
5.2.6 - RELACIONAMENTO COM OUTROS PROGRAMAS V.2
5.2.7 - VOLUMES ESPERADOS V.2
5.2.8 - PORTABILIDADE V.2
5.2.9 - OBJETIVO DE QUALIDADE, NORMAS E PADRÕES V.3
5.2.10 - EXPECTATIVAS DE NECESSIDADES DE RECURSOS V.3
5.2.11 - RESTRIÇÕES E REQUISITOS ADICIONAIS V.3
5.3 - REQUISITOS FUNCIONAIS V.3
5.4 - DIAGRAMA EM BLOCOS V.5
5.5 - FACILIDADES DO AMBIENTE V.6
5.5.1 - AMBIENTE V.7
5.5.2 - EDITORES V.7
5.5.3 - MONTADOR V.7
5.5.4 - "COMPILADOR" V.7
5.5.5 - ARQUIVOS V.7
5.5.6 - CONFIGURAÇAO V.8

6 - AIDS_TME VI.1

6.1 - SENHA DE ACESSO AO AIDS_TME VI.1
6.2 - MENU PRINCIPAL DO AIDS_TME VI.2
6.2.1 - AMBIENTE VI.2
6.2.2 - EDITORES VI.2
6.2.3 - MONTADOR VI.3
6.2.4 - COMPILADOR VI.3
6.2.5 - ARQUIVOS VI.3
6.2.6 - CONFIGURAÇAO VI.3
6.2.7 - XFIM VI.3
6.3 - AMBIENTE VI.4
6.3.1 - UNIDADE VI.4
6.3.2 - DIRETÓRIO VI.5
6.3.3 - OPERADOR VI.5
6.4 - EDITORES VI.5
6.4.1 - EDITOR DE "SETUP" VI.6
6.4.2 - EDITOR DE MEDIDA VI.9
6.4.3 - EDITOR DE GRÄFICO VI.10
6.4.4 - EDITOR DE PLANILHAS VI.13
6.5 - MONTADOR VI.14
6.6 - COMPILADOR VI.15
6.7 - ARQUIVOS VI.16
6.8 - CONFIGURAÇAO VI.17
6.9 - XFIM VI.19

7 - MANUAL DO PROGRAMADOR - PROCEDIMENTOS GERAIS VII.1

7.1 - AMBIENTE VII.1
7.2 - EDITORES DE "SETUP" E MEDIDAS VII.1
7.3 - EDITOR DE GRÄFICO VII.2
7.4 - EDITOR DE PLANILHAS VII.2
7.5 - MONTADOR VII.3
7.6 - COMPILADOR VII.3

7.7 - ARQUIVOS VII.4
7.7.1 - EDITAR VII.4
7.7.2 - VISUALIZAR VII.4
7.7.3 - IMPRIMIR VII.4
7.7.4 - CANCELAR VII.5
7.8 - CONFIGURAÇAO VII.5
7.8.1 - PROGRAMA DE CONFIGURAÇAO DOS INSTRUMENTOS VII.5
7.8.2 - PROGRAMA DE CONTROLE INTERATIVO - IBIC VII.6
7.8.3 - PROGRAMA PARA VERIFICAÇAO DE ARQUIVOS VII.7
7.9 - XFIM VII.7

8 - EXEMPLO DE APLICAÇAO - MEDIDA DE DESVIO DE BASE DE TEMPO VIII.1

8.1 - MEDIDA DE DESVIO DE BASE DE TEMPO VIII.1
8.2 - EDIÇAO DO MÖDULO DE "SETUP" VIII.2
8.3 - EDIÇAO DO MÖDULO DE MEDIDAS VIII.5
8.4 - EDIÇAO DO MÖDULO DE GRÁFICO VIII.7
8.5 - EDIÇAO DO MÖDULO DE PLANILHA VIII.9
8.6 - MONTAGEM DO ARQUIVO DE TESTE VIII.11
8.7 - VERIFICAÇAO DO INSTRUMENTOS ACESSADOS VIII.12
8.8 - "COMPILAÇAO" E "LINK-EDIÇAO" VIII.13
8.9 - O ARQUIVO EXECUTÁVEL VIII.14
8.10 - OS RESULTADOS DO ENSAIO VIII.15
8.11 - CONSIDERAÇÕES FINAIS VIII.18

9 - CONSIDERAÇÕES FINAIS IX.1

9.1 - O HARDWARE IX.1
9.2 - O SOFTWARE IX.2
9.3 - O AIDS_TME IX.2
9.4 - OS PROGRAMAS FINAIS (EXECUTÁVEIS) IX.3
9.5 - NOVAS INTERFACES IX.3
9.6 - A EVOLUÇAO DO AIDS_TME IX.3
9.7 - O OBJETIVO X RESULTADOS IX.4

10 - REFERÊNCIAS BIBLIOGRÁFICAS X.1

ANEXO I - PROGRAMA FONTE DO AMBIENTE

ANEXO II - PROGRAMA FONTE DO EXEMPLO

ANEXO III - BIBLIOTECA DE FUNÇÕES

ANEXO IV - SISTEMA DE COMUNICAÇAO GPIB (IEEE-488)

ANEXO V - LINGUAGEM DE PROGRAMAÇAO QUICKBASIC

ANEXO VI - INSTRUMENTAÇAO COM INTERFACE GPIB (IEEE-488)

ANEXO VII - MANUAL DE INSTALAÇAO - DISCOS DE INSTALAÇAO

1 - INTRODUÇAO

O objeto deste trabalho é a especificação, implementação e teste operacional de um "Ambiente Interativo de Desenvolvimento de Software para Testes e Medidas Elétricas" - AIDS_TME, com características modulares e de alta reusabilidade de rotinas básicas na produção de software. O aumento da produtividade e da qualidade dos programas gerados dentro deste ambiente é uma característica inerente a sistemas dessa natureza.

A filosofia que norteia este desenvolvimento está baseada em anos de experiências em laboratórios e áreas de manutenção eletroeletrônica. A utilização de instrumentação com interface GPIB - "GENERAL PURPOUSE INTERFACE BUS" «1», a necessidade de produção de uma enorme variedade de programas de testes para automatizar atividades rotineiras ou para a execução de ensaios extremamente longos e complicados deu início a uma idéia, até então inédita, da criação de uma "ferramenta" capaz de executar esta difícil tarefa de geração de programas dedicados para esta área.

A experiência mostrou que a maioria dos programas de testes, pelo menos 90%, tem sua estrutura básica bem definida, composta por: Um módulo de Inicialização, Um módulo de Execução das Medidas propriamente dito, Um módulo de Análise dos dados adquiridos ou gerados durante o ensaio e/ou teste. Neste módulo de análise está incorporada a criação e geração de relatórios, planilhas, gráficos, etc.

A partir de uma metodologia estruturada para a geração destes programas, uma nova e original idéia foi incorporada a esse ambiente, ou seja, a da utilização de programas modulares, compostos de procedimentos e rotinas básicas, em pequenos módulos, que podem ser utilizados por qualquer tipo de teste e/ou ensaio.

A modularidade é definida a nível de atividade «2», ou seja, a atividade de inicializar os instrumentos que irão participar do teste foi agrupada em um único módulo, definido como "SETUP", que contem um dado número de comandos para os instrumentos e para o ambiente. Este módulo será agrupado aos demais (Medidas, Gráficos, Planilhas) formando um programa único e original.

Baseado na reusabilidade de software «3» estes módulos poderão ser utilizados e/ou agrupados em diferentes programas. Uma vez definido o módulo, este poderá ser utilizado em qualquer outro programa de teste, que tenha a necessidade da mesma seqüência de inicialização.

O ambiente de Hardware necessário como plataforma de utilização é tão somente um Microcomputador da Linha IBM PC AT/XT, com uma interface de comunicação GPIB/PC, e com uma memória mínima de 640K bytes de memória RAM e uma unidade de disco rígido (mínimo 20M bytes) e um vídeo de média resolução. Como ambiente de software é necessário um sistema operacional na versão 3.0 do MS-DOS ou mais recente «20».

Estão incorporadas ao ambiente ferramentas de edição, compilação e "linker", sendo a sua utilização totalmente transparente ao usuário final. Arquivos específicos e programas de domínio publico também estão incorporados. A definição da abrangência deste ambiente está intimamente ligada as áreas da engenharia, pois todo e qualquer ensaio ou teste pode ser decomposto em módulos e produzidos a partir de módulos já existentes de programas anteriores.

Outra caracteristica desse ambiente é a interface Homem/Máquina utilizada. Após estudos de viabilidade e consulta a diversos usuários foi possível utilizar uma metodologia mista de interface Homem/Máquina com a utilização de "POP-UP MENU" (janelas) e Linhas de Comandos. Todas as telas que apresentam as opções das facilidades do ambiente são acessadas através de um "MOUSE" ou por uma tecla (primeira letra da opção) ou pelas teclas de direção do teclado (23).

2. SISTEMAS DE DESENVOLVIMENTO DE SOFTWARE

Aumento da produtividade (1?) é o que todo líder de projeto persegue. Mas, em se tratando de produzir software, como é possível conciliar a rapidez no atendimento á demanda dos usuários com a qualidade do produto destinado a estes mesmos usuários ?.

Na verdade, o que se quer é combinar eficiência e eficácia. Eficiência é aumentar a produtividade dos profissionais de informática através de metodologias e ferramentas de apoio ao desenvolvimento de sistemas. Eficácia é aumentar a produtividade e a satisfação dos usuários através de sistemas funcionalmente adequados, confiáveis e simples de usar.

Partindo da premissa de que um sistema é a concretização de idéias e conceitos abstratos, e de que quanto mais tarde um erro é detectado maior será o seu custo, buscam-se ferramentas que ajudem a "visualizar as soluçÕes buscadas" e testar tais idéias e conceitos, antes que se invista um grande esforço no desenvolvimento e implementação da solução final.

Hoje, o quadro da Informática é curioso. De um lado, persiste a visão conservadora de alguns de que basta uma linguagem de programação sem metodologia ou ferramentas de apoio. De outro, divisa-se no horizonte os sistemas especialistas e as interfaces em linguagem natural, transformando o usuário final em "programador" de suas soluçÕes de organização e recuperação de informaçÕes.

Entre o passado e o futuro, profissionais de informática buscam ferramentas que tornem o trabalho de projetar e construir sistemas mais proveitoso e previsível. EvoluçÕes tecnológicas entrelaçadas com evoluçÕes culturais vão avançando as fronteiras do que é "comercial", ou seja, digerível pelos gerentes de Informática.

Bancos de dados relacionais, uso de micros como estação de trabalho, processamento distribuído, sistemas interativos e interfaces gráficas já deixaram de ser "futurismos" e foram incorporados pelo mercado.

Inteligência Artificial, Case (Computer-Aided Software Engineering), estaçÕes de programação (como os da linguagem Ada e dos sistemas Unix), ambientes de prototipagem, integração de voz, vídeo, imagem e dados em automação de escritórios breve (quem duvida?) serão tão acessíveis como as velhas e ainda úteis linguagens Cobol, Fortran e Basic. Façam suas apostas e boa sorte!

Neste capítulo será abordado o contexto de desenvolvimento de programas e aplicativos de uma forma genérica, ou seja, o termo "sistemas" tem uma conotação abrangente e irrestrita no que diz respeito a Engenharia de Software e sistemas de desenvolvimento de software.

2.1 - A PRODUÇAO DE SOFTWARE (22)

Com a evolução da Informática nas diversas áreas da engenharia é comum atualmente referir-se a Engenharia de Software como um ramo igual as demais áreas.

Felizmente existem produtos de Engenharia de Software suficientemente palpáveis para convencer os mais céticos de que, se por um lado construir software não é uma atividade de uma linha de montagem: operacional e não criativa, por outro, não se limita a lampejos de genialidade. Hoje, o processo de produção de software conta com ferramentas que concretizam os conceitos e princípios da Engenharia de Software, visando apoiar a organização da criatividade e possibilitar o trabalho em equipe, permitindo a produção de software em "verdadeiras" linhas de produção (montagem).

2.2 - VISUALIZAÇAO DE IDéIAS

Grande parte do trabalho de projetar e implementar sistemas e programas relaciona-se à concretização de idéias e modelos abstratos.

Assim, por exemplo, um sistema transacional parte da idéia de que é possivel dividir a operação de um negócio em atividades atômicas (transaçÕes) que concorrem no tempo. Essas operaçÕes podem ou não ser independentes, mas o resultado final depende da concorrência e das transaçÕes existente entre elas, processo este, que ocorre de modo paralelo.

Vários modelos podem ser desenhados para cada sistema que se pretenda implementar, um outro tipo de sistema imaginado é aquele onde as açÕes ocorrem de maneira interativa e em linha, as açÕes são dependentes e o processo ocorre de modo serial.

Através da distribuição do processamento para terminais e impressoras remotas ou até mesmo com a ligação de micros ao sistema central, ou como mais recentemente, a formação de redes locais ou geográficas onde pode-se obter um sistema descentralizado e autônomo, possibili-tando a visualização de soluçÕes antes mesmo de imple-mentá-la o que amplia o número de variaçÕes consideradas, reduzindo a margem de erro.

A existência de ferramentas de software que facilitem a exploração de diferentes alternativas é fundamental para a implementação de sistemas interativos, transacionais ou não, adequados às necessidades dos usuários. Em termos de hardware, os terminais de vídeo já existem a anos, mas é possível ainda encontrar pessoas desenvolvendo sistemas como se tudo fosse da era dos cartÕes perfurados e gabaritos de espaçamento. Projetar e desenvolver sistemas em um ambiente interativo permite usufruir a capacidade de visualização e de "feedback" imediato, inerente a terminais de vídeo e microcomputadores.

2.3 - QUANTO CUSTA UM ERRO ?

Quanto mais tarde um erro é descoberto em um projeto, maior é seu impacto e custo para removê-lo ou absorvê-lo. A disponibilidade de ferramentas de desenvolvimento de software podem afetar o custo do erro e sua detecção de duas formas simultâneas:

I - Permitindo a descoberta de procedimentos, conceitos e implementaçÕes equivocadas o mais cedo possível;

II - Barateando o custo de alteração de algoritmos, "layout" de telas, restriçÕes de integridade e formas de diálogo do sistema em desenvolvimento.

O reconhecimento do erro, como elemento presente nas fases de projeto e desenvolvimento de sistemas, permite efetuar um refinamento das supostas necessidades do usuário e no modo de operação de um sistema, antes mesmo de se iniciar os testes de pré-implantação.

Através da utilização de ferramentas adequadas torna-se possível a realização de ciclos de prototipagem e progressiva mutação no processo de manutenção de sistemas em evolução.

2.4 - A IMPORTANCIA DAS INTERFACES

Como o proverbio que "Uma corrente é tão forte quanto o mais fraco dos seus elos", nessa linha de raciocínio, um sistema que tenha uma interface com o usuário inadequada pode fracassar, apesar de ser funcionalmente perfeito.

O diálogo entre o usuário e computador pode ser efetuado de várias formas: comandos, questionários, menus, telas, teclas especiais e dispositivos indicadores como: "mouse", "joystick", "light pen", etc.

O uso de comandos é o mais antigo meio de diálogo, e o único disponível em sistemas que trabalham estritamente em "batch". O usuário deve conhecer a sintaxe dos comandos, embora alguns sistemas interativos oferecem facilidades de ajuda ("help on-line"), aceitando comandos incompletos, assumindo ou requisitando os parâmetros essenciais esquecidos, e permitindo o uso de formas abreviadas desses comandos.

A aplicação de questionários é interessante para usuários inexperientes, mas aborrece e atrasa a quem tem uma certa prática e pode ser usada em conjunto com formas mais diretas. O uso de menus é simples, sendo adequado a usuários casuais ou inexperientes e seu uso intensivo torna-se cansativo.

Telas para consulta e entradas de dados são compostas de textos informativos e campos de dados. Campos podem ter atributos tais como: formato (comprimento, dimensão dos caracteres, tipo de edição); domínio (alfabético, numérico, data, hora, faixas ou conjunto de valores possiveis, dígito verificador); presença (opcional, obrigatório); conversão para maiúsculo; apresentação (piscante, sublinhado, intensificado, vídeo invertido); acesso (proteção contra escrita ou leitura).

Em termos práticos, o ideal é poder combinar diferentes tipos de interfaces levando-se em conta as áreas de aplicação, os perfis dos usuários, os terminais disponíveis, os tempos de resposta, etc.

Um sistema que disponha, ao mesmo tempo, de menus para usuários novatos ou casuais e comandos para os experientes oferece o melhor de dois mundos. Como mesmo os usuários habituais esquecem as funções pouco usadas os menus puderão ser úteis toda vez que surgir alguma dúvida sobre o comando a utilizar.

Oferecer ajuda ("help on-line") ao usuário também é interessante. Apresentar uma tarja ou janela de comandos ou opções disponiveis é um misto de menu e ajuda. Em qualquer tempo, é importante poder pressionar uma tecla de ajuda e receber algumas linhas ou telas de orientação sobre como proceder (sintaxe e efeito de comandos, função de teclas especiais, forma de preenchimento de campos, procedimentos possíveis).

Mais recentemente a utilização de "Icones", em forma gráfica de representação, facilita ainda mais a relação entre o usuário e o software, complementando assim o modelo menus/comandos.

2.5 - PROTOTIPAGEM (26)

Um protótipo é uma versão experimental, contendo as características essenciais de um objeto, embora de forma incompleta e/ou imperfeita. Prototipagem é a aplicação de uma abordagem de tentativa-e-erro ao desenvolvimento de sistemas.

Uma das premissas, que parece sustentar os enfoques tradicionais de desenvolvimento de sistemas, é a de que o sistema existe implicitamente, cabendo ao analista explicitá-lo.

A construção de protótipos é longamente usada na engenharia e na arquitetura, pois permite estudar certos aspectos em profundidade do objeto projetado e desprezar alguns reduzindo a complexidade do desenvolvimento.

A capacidade de produzir, a baixo custo e em pouco tempo, um ou mais protótipos de um sistema, oferece a possibilidade de testar diferentes soluções para um dado problema e adotar aquela que for mais adequada e de preferência do usuário.

Com ferramentas adequadas, o analista pode rapidamente oferecer ao usuário um protótipo do sistema em discussão. é sabido que, após utilizar um sistema por vários dias, o usuário altera sua percepção sobre o que realmente precisa. A possibilidade de manipulação e utilização de um protótipo aguça essa percepção no inicio do projeto, fase onde as correções de rota são fáceis e baratas, pois explicita mal-entendidos, ambiguidades e inconsistências; também expõe as interpretações conflitantes que diferentes usuários têm sobre os mesmos dados e procedimentos do sistema; possibilita e ajuda a encontrar as omissões; facilitando ao usuário antever novas características; e expondo partes difíceis de usar ou confusas até esta etapa do desenvolvimento.

A forma tradicional de desenvolver sistemas só chega a expor o produto ao usuário quando está supostamente

concluído, após um grande investimento de tempo e dinheiro. Nesse ponto, qualquer distanciamento entre o que o usuário esperava ver e o que na realidade o sistema apresenta provoca sérios traumas, pois o usuário se decepciona, e o pessoal de sistemas fica com a sensação de que o usuário escondeu as reais necessidades para agora impor as alterações trabalhosas, demoradas e as vezes impossíveis.

Para tornar viável a prototipagem é preciso que o custo da exploração de diferentes alternativas seja baixo, o que só pode ser conseguido com a utilização de ferramentas de software mais poderosas que as linguagens imperativas e os algorítimos comumente utilizados.

Ferramentas úteis para construção de protótipos são geradores e gerenciadores de telas, dicionário de dados, linguagem de consulta e entrada de dados.

O objeto desse trabalho carrega as características de um sistema interativo com interface mistas do tipo menu, comandos e telas permitindo a rápida prototipagem dos programas gerados e a correção dos erros detectados.

2.6 - DESENVOLVIMENTO DE PROGRAMAS (24-29)

As ferramentas voltadas para construção de protótipos são igualmente úteis no desenvolvimento do sistema final. As telas e estruturas de dados já preparadas e usadas como linguagem de quarta geração devem ser usadas em programas escritos em linguagens de terceira geração tais como, Cobol, Basic, C, Fortran ou Pascal. O sistema final pode ser assim ser composto de módulos de terceira ou quarta geração, conforme a conveniência.

Uma vez que as definições de telas e estruturas de dados estejam prontas e armazenadas em um dicionário de dados, resta ao programador referenciá-las e desenvolver os procedimentos de acesso a arquivos, cálculos e manipulação de dados, bem como estabelecer a seqüência do diálogo com o usuário (através das telas). Desta forma, além da independência de dados, busca-se a independência de interface com as descrições de dados e telas externas ao programa.

A produtividade e a qualidade na produção de software podem ser aumentadas se o programador dispuser de um ambiente integrado de desenvolvimento e de depuração de programas, pois tarefas como edição, compilação e testes consomem parcela significativa do cronograma de um projeto.

A gerência automática de documentação, de programas fonte e de módulos executáveis facilita a reutilização de rotinas padrões e evita a propagação de erros, quando se faz a manutenção em módulos referenciados de múltiplos pontos em sistemas modulares completos.

Ainda são desejáveis facilidades automáticas para controlar os ciclos de teste de programas, registrar sessões interativas para revisões, detectar pontos no programa que requerem otimização.

Com ferramentas como citadas acima, o desenvolvimento de sistemas ganha eficiência e pode dedicar mais tempo na obtenção de soluções mais voltadas aos usuários.

2.7 - ESTAÇAO DE TRABALHO DO PROGRAMADOR

Um bom editor de programas agiliza a criação e depuração de programas. Existem bons editores que trabalham com toda a tela, permitindo rolar o texto, procurar e substituir trechos do programa.

Recentemente tem crescido a consciência de que um editor de programas é muito pouco. O que se deseja é um ambiente de desenvolvimento no qual o programador possa criar o programa a partir de objetos pré-existentes, tais como, rotinas padrões, modelos para expandir ou completar, estruturas de dados e esquemas de manipulação de telas e campos.

A cada módulo completado deve ser possível compilar, corrigir os erros de codificação e monitorar a execução sem sair do contexto do problema. Tal ambiente tem sido chamado de estação de trabalho do programador e deveria contar com um editor sensível a sintaxe da linguagem, um depurador simbólico e interfaces com bibliotecas de fontes, de telas e com dicionário de dados.

Um editor sensível à sintaxe da linguagem, ou seja, capaz de detectar erros de sintaxe durante a geração do programa, pode agilizar a expansão de programas, a partir de estruturas sintáticas. Por exemplo, ao expandirmos um comando, o sistema pode mostrar a lista de comandos e nos permitir escolher e completar lacunas com nossas opções. Se não soubermos como preencher podemos pressionar uma tecla de ajuda e um trecho do manual ser exibido; se o assunto for manutenção inserimos uma linha com o nome do comando desejado e solicitamos sua expansão. A produtividade decerto melhorará se o programador concentra-se na lógica da solução e não nas alegorias da linguagem.

2.8 - DEPURADOR SIMBÓLICO

Agora, que tal se contarmos com um depurador simbólico, ou seja, se pudermos testar o programa compilado como se fosse interpretado?. A idéia é monitorar a execução do programa: acompanhamento no fonte o caminho percorrido, passo a passo ou através de pontos de parada(número da linha ou nome da rotina); consultando e alterando o valor de variáveis; determinando que o valor de dada variável seja exibido sempre que alterado seu conteúdo, e assim por diante.

O depurador deve permitir acompanhar sistemas modulares, entrando em subprogramas disponíveis no sistema. O objetivo é poder trabalhar a nível simbólico (fonte) sem ter que decifrar códigos hexadecimais, linguagem de máquina e hieroglifos semelhantes, e ainda, conseguir esse tipo de acesso quando for necessário localizar falhas do programa.

2.9 - LINGUAGEM DE 4@ GERAÇÃO (L4g)

O encadeamento de procedimentos bem como a consulta e a entrada de dados podem ser simplificados pelo uso de uma linguagem de especificação (L4g), na qual basta dizer "o que" se quer que seja feito, em contraste com as linguagens imperativas e algorítmicas (L3g), nas quais é imprescindível detalhar passo a passo a forma "como" o resultado deve ser atingido.

Funções para selecionar conjuntos de dados, operações relacionais sobre bancos de dados e arquivos convencionais, manipulação de telas, elaboração de relatórios e gráficos são algumas das facilidades comuns às L4g.

2.10 - EDIÇÃO / GERÊNCIA DE TELAS

O projeto de interfaces pode ser facilitado por um editor de telas que permite o desenho da tela, a especificação do formato e dos atributos dos campos e, se possível, das validações de domínio, conforme já abordamos no Tópico "A importância das interfaces".

O principal é poder projetar e alterar a composição de constantes e campos da tela sem precisar programar, facilitando o projeto em parceria com o usuário e barateando a experimentação de diversas alternativas, o que seria de outra forma impraticável.

2.11 - MODELAGEM DE DADOS

Um dicionário de dados permite armazenar definições de dados e eventualmente de telas. Desta forma existe uma única representação de cada dado, mesmo sendo referenciado em diversos programas, o que facilita a manutenção e aumenta a integridade.

Ferramentas que auxiliam na normalização e na validação dos modelos de dados são também de grande utilidade.

2.12 - GERADOR DE PROGRAMAS

Um gerador de programas permite que se especifique, através de menus ou de comandos, "o que" se deseja que o programa faça e com quais dados. A partir das especificações colhidas será produzido um programa em uma L3g. A geração de programas pode ser o meio caminho entre as linguagens de 3@ e 4@ gerações.

É interessante que os programas gerados estejam integrados ao mesmo ambiente da L3g e L4g, compartilhando o dicionário de dados, o editor de telas e o depurador simbólico. Assim, os sistemas resultantes poderão conter programas construídos de mais de uma forma e em tempos diversos, como exige a realidade dos centros de processamento de dados e a dinâmica das empresas.

A realização de testes requer planejamento e antecipação dos objetivos que o programa deve atingir para ser considerado correto e adequado. Uma ferramenta para auxiliar no controle dos ciclos de teste de programas deve

permitir antecipar conjuntos de resultados a serem atingidos, documentar a execução dos testes, comparar os resultados previstos com os obtidos e repetir este ciclo variando as condiçÕes sistematicamente.. Conferir os registros gravados ou registrar uma sessão interativa para posterior análise são algumas possibilidades para facilitar os testes de programas "batch" e interativos.

Otimizar programas é uma tarefa árdua e nem sempre realizada, por exigir tempo e quase sempre depender do "olho clínico" de um programador experiente. Descobrir quais trechos são mais executados com certa massa de dados ou, ainda, que partes do programa provocam mais paginação tem sido muitas vezes um exercício de tentativa-e-erro ou de adivinhação.

Um ambiente de desenvolvimento de programas que vise produtividade e qualidade deve facilitar a detecção de gargalos de execução, frente a diferentes situaçÕes de utilização.

2.13 GERÊNCIA DE FONTES E DOCUMENTOS

Manter controle sobre as alteraçÕes efetuadas em programas fonte por uma ou mais pessoas é fundamental no desenvolvimento de sistemas complexos.

Um ambiente de desenvolvimento deve incluir formas simples de documentar a evolução sofrida pelos programas durante sua construção e posterior manutenção, registrando quais foram as mudanças, em que ordem e executadas por quem. Isto requer a gerência das bibliotecas de fontes e a integração com os editores de programas usados.

É útil também que a gerência de fontes estenda-se aos documentos gerados, durante o projeto e a manutenção dos sistemas, e mantidos automaticamente, tais como especificaçÕes e listagem de testes.

2.14 - GERÊNCIA DE MÓDULOS

As vantagens de um sistema modular são conhecidas: viabilizar o trabalho em equipe, reduzir a redundância de programas, possibilitar a construção por camadas funcionais e criar condiçÕes para aumentar a produtividade e a qualidade do software produzido.

Contudo, a interdependência entre os muitos módulos torna necessário manter o controle das versÕes correntes de rotina, tabelas e arquivos de mensagens, pois a alteração de um componente pode exigir recompilação ou religação de outros. Uma ferramenta que controle a relação entre as diferentes versÕes de módulos fontes, objetos executáveis, e que recompile e religue os módulos alterados e aqueles que os referenciam é de extrema utilidade no desenvolvimento e na manutenção de sistemas modulares.

3 - AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE (22)

Desde a invenção do transistor até os "milagres" da microeletrônica de nossos dias muita coisa mudou na concepção, na geração e utilização de "sistemas" ou programas de computadores, comumente denominados de "software" ou puramente programas de computadores.

Esta concepção de evolução é extremamente pobre e desprovida de amparo tecnológico e com uma visão muita pequena da evolução industrial e científica que propiciou toda esta mudança comportamental da sociedade mundial. É bom relembrar alguns dados ou "pontos" dessa trajetória.

. Ponto 1: Em 1983 a IBM vendeu mais computadores pessoais do tipo IBM/PC que todos os tipos de seus computadores juntos desde a sua fundação;

. Ponto 2: Em menos de 5 anos a Apple Computer Company mudou-se de uma pequena garagem para uma enorme e bem equipada área. ("Fortune 500");

. Ponto 3: Em meados dos anos 70, a divisão de computadores Radio Shack Division da Tandy Company entrou para os negócios (ramo) de computadores pessoais. Os computadores da linha TRS-80 são hoje os responsáveis pela maior parte do faturamento da Tandy Co.;

. Ponto 4: Em 1983 pela revista "TIMES", mundialmente consagrada, O computador foi escolhido como "O Homem do Ano";

. Ponto n: Muitos outros fatos que poderíamos enumerar que foram gerados e/ou produzidos pela interferência direta ou indireta de computadores e/ou sistemas.

Poderíamos aqui relatar o proliferação dos computadores e principalmente dos microcomputadores pessoais nas mais variadas áreas da ciência e da tecnologia por vários capítulos, mas o que realmente tem a responsabilidade e provoca cada vez mais essa "geração" quase que incontrolada de novos sistemas e produtos para essas "máquinas maravilhosas", não é apresentado em evidência e/ou importância devida : "A Produção de Software" ou a "Engenharia de Software".

Muito menos ao "ambiente" e aos "elementos" necessários para a geração do produto final, ou seja, para o usuário final, receberá somente a informação sobre o sistema (produto acabado), sua operação e seus "sub-produtos". Neste trabalho a nossa atenção será totalmente direcionada para este "Ambiente de Produção" que ao nosso ponto de vista deverá ser tratado tão somente como um "Ambiente de Desenvolvimento de Software" (ADS).

Tópicos como Ciclo de vida de Software e/ou Sistemas, Técnicas de Gerenciamento, Tipos de Ambientes e sua Arquitetura serão a tônica deste capítulo.

3.1 - ENGENHARIA DE SOFTWARE (25)

A definição e aplicação correta para o termo "Engenharia de Software" ainda produz alguns embaraços para os profissionais de engenharia e de informática pura. Quando o termo "engenharia" é utilizado sempre se está relacionando atividades de geração, produção ou mesmo de manutenção de bens ou produtos bem definidos.

Com o intuito de apresentar e colocar um ponto de vista próprio da concepção de Engenharia de Software, os itens que se seguem discutem definições e termos correlatos a geração e produção de software e principalmente de sistemas de desenvolvimento utilizados como meio e ferramenta de desenvolvimento de software.

3.1.1 - CICLO DE VIDA DE ENGENHARIA DE SOFTWARE

O processo de Engenharia de Software é um projeto orientado. O desenvolvimento de um projeto é definido com um começo e um final, feliz ou não, com uma grande quantidade de atividades intermediárias entre esses dois pontos, que por sua vez resulta no produto final.

O produto final, no caso de software, não é somente o programa "executável" adquirido pelo usuário mas também do código fonte do programa; especificações básicas do sistema para manutenções futuras e novos desenvolvimentos; procedimentos básicos para o usuário explicando como funciona a interface Homem/Máquina; procedimentos de entrada de dados; "Backup" dos dados; inicialização e finalização do processo (sistema).

Todos esses "produtos" são gerados através de uma série de trabalhos intermediários, com vários estágios e etapas bem definidas. Estas séries de transformações de idéias, dados e informações em sub-produtos, até o produto final, é definido como sendo "Ciclos de Vida" da Engenharia de Software.

Se utilizarmos um alto grau de abstração os ciclos de vida de Engenharia de Software poderão ser definidos em cinco grandes fases:

1.- Definições de requisitos;
2.- Especificações lógicas;
3.- Especificações físicas;
4.- Implementação do sistema;
5.- Testes e instalação.

Cada fase definida acima é composta por uma série de pequenas e importantes atividades. A definição de requisitos consiste basicamente em uma especificação funcional do sistema ou software a ser desenvolvida. As especificações lógicas detalham os blocos funcionais e definem os fluxos de dados e interfaces com o usuário. As especificações físicas evoluem as definições ainda teóricas para o ambiente real onde o sistema será implantado ou executado. As duas etapas finais são correlatas e muitas vezes atividades de

desenvolvimento e implementação do sistema carecem de dados e informaçÕes obtidas durante os testes a instalação do sistema, constituindo assim uma realimentação positiva no processo de desenvolvimento do produto final.

3.1.2 - CICLO DE VIDA E TÉCNICAS DE DESENVOLVIMENTO

Toda e qualquer atividade dentro do ramo da engenharia obedece uma "metodologia", definida como "criação, atualização, validação, regras, etc", e principalmente dentro da Engenharia de Software esta metodologia é definida como "um sistema de métodos" onde métodos são definidos como "uma ordenação regular de atividades". Ainda dentro da Engenharia de Software existem dois tipos diferentes de metodologia: ciclos de vida e técnicas de desenvolvimento.

Técnicas de desenvolvimento provê regras e procedimentos para a criação e execução das transformaçÕes dentro das etapas de criação. As técnicas de desenvolvimento especificam também as formas gráficas e/ou lingüísticas a serem utilizadas no desenvolvimento das atividades intermediárias bem como das regras de validação para estas atividades.

A metodologia de ciclo de vida, por outro lado, provê regras bem claras sobre o Gerenciamento dos projetos de desenvolvimento, definindo as etapas de trabalho que deverão ser completadas, bem como sua seqüência. A definição de regras de avaliação e de seleção do tipo de trabalho a ser executado e dos elementos de execução também são geridos por essa metodologia. Em essência, o ciclo de vida provê regras para determinação do trabalho a ser executado e por quem deve ser executado.

3.1.3 - TÉCNICAS GERENCIAIS

Desde o início dos anos 70 que os especialistas na área de Engenharia de Software (Constantine, Orr, Yourdon, e DeMarco), começaram a entender que tanto a qualidade como a quantidade dos programas e sistemas que estavam sendo construídos poderiam ser aumentadas sem aumentar a complexidade de linguagens ou de ferramentas, mas sim, aprimorando as técnicas de Gerenciamento das atividades e dos elementos envolvidos na produção de software.

Essa eficiência culminou na utilização de uma programação estruturada, onde as atividades e programas foram agrupadas em blocos e possuíam o seu gerente próprio com um ganho real de eficiência. A análise estruturada e técnicas de desenvolvimento tiveram uma rápida evolução enfocando uma lógica e sistemática identificação dos requisitos para programação, projeto do sistema, teste e codificação do sistema.

Como mencionado anteriormente, essa técnica está embasada em duas categorias básicas : técnicas lingüísticas e técnicas gráficas. Algumas das técnicas lingüísticas são na verdade uma linguagem "estruturada", com palavras bem definidas e que expressam corretamente as atividades que deveram ser desenvolvidas. As técnicas gráficas enfocam um processo de fluxo onde figuras "falam mais" que palavras na definição e gerenciamento das atividades. Esta técnica teve a sua evolução acelerada com a utilização e popularização de ferramentas gráficas em computadores de pequeno e médio porte.

3.1.4 - CONSIDERAÇ©ES SOBRE GERENCIAMENTO

Pode-se agrupar as técnicas de gerenciamento em quatro grandes grupos (escolas), e que se individualizam de acordo com a sua aplicação dentro das técnicas de desenvolvimento da Engenharia de Software.

1.- Fluxo de Processo, também chamado de Fluxo de Dados concentra o fluxo de dados de um sistema e o processo de modificação e transformação dos dados dentro desse processo. Fluxo de Processo baseia-se em um método de entrada-processo-saída para especificação de sistemas.

2.- Projeto de Dados Estruturados, concentra basicamente na natureza dos dados disponíveis para o sistema e nas transformaç©es que devem ser executadas sobre esses dados. O Projeto de Dados Estruturados começa com a especificação dos dados de saída, definindo seus componentes, construindo os dados de entrada e especificando o processo, baseia-se em um método saída-processo-entrada para especificação de sistema.

3.- Estados e Transiç©es, é o método utilizado para especificação de sistemas em tempo real (processo de controle), que não opera e não requer um grande banco de dados de informaç©es. A metodologia de estados e transiç©es baseia-se nos estados do sistema e especifica como e qual estado deve produzir uma determinada saída para uma dada entrada.

4.- Projeto por Base de Conhecimento, este método é dedicado basicamente para projetos que envolvam inteligência artificial e prevê a especificação de requisitos para sistemas sofisticados de simulação utilizando inteligência artificial. O Projeto por Base de Conhecimento é um método extremamente experimental não implementado ainda em grande escala.

Todos esses métodos apresentam vantagens e desvantagens se forem comparados de uma maneira pura. Cada aplicação e cada gerente tem sua particularidade o que nos permite utilizar esses métodos de uma forma extremamente abrangente e podendo ser respeitado as

particularidades de cada projeto sem a descaracterização do método.

3.2 - CASE - "Computer Aided Software Engineering" (21-26)

Conforme apresentado anteriormente a Engenharia de Software busca as seguintes metas:

- Aumentar a produtividade dos elementos que produzem software ;

- Aumentar a qualidade do software produzido ;

- Incrementar o controle e gerenciamento sobre as etapas desse processo.

"Computer-Aided Software Engineering" (CASE) é a aplicação de uma tecnologia automatizada de procedimentos da Engenharia de Software. CASE é na essência a automação dos processos e procedimentos definidos dentro da Engenharia de Software.

As técnicas e metodologias definidas pelos pioneiros da análise estruturada são agora automatizadas e difundidas através de novos e poderosos ambientes automáticos e automatizados de produção e gerenciamento de software.

3.2.1 - INTRODUÇAO AO CASE

Os elementos básicos do CASE são:

- Procedimentos: uma disciplinada e bem estruturada metodologia para o desenvolvimento de software;

- Método: técnicas e procedimentos padrões para produção de projetos;

- Integração: ferramentas automáticas para: estimativa e planejamento de projetos; regras de acompanhamento do progresso do projeto; criação /modificação do projeto; gerenciamento das informações do projeto; protótipos e módulos de código; análise e verificação de projeto; considerações sobre a qualidade do produto e acompanhamento dos requisitos do sistema dentro e durante a implementação do sistema.

O que caracteriza o CASE é a extrema integração, flexibilidade e facilidade de utilização do ambiente de desenvolvimento pelo usuário.

Ambientes CASE são atualmente sistemas computadorizados capazes de construir automaticamente sistemas. As etapas de projeto, implementação e teste do sistema são contempladas dentro de um ambiente CASE.

O primeiro sistema CASE, que nunca foi chamado dessa maneira, foi provavelmente o Prde/ASDM ("Automated System Development Methodology"), desenvolvido por M. Bryce and Associates, Inc., nos anos 70 e teve a sua extensão com o PRIDE ("Profitable Information by DEsing"), ambos os métodos foram criados e direcionados para o gerenciamento e controle dos ciclos de vida e das técnicas dentro da Engenharia de Software.

3.2.2 - ARQUITETURA CASE

O sistema CASE verdadeiro é aquele onde é possível encontrar todos os ingredientes necessários para a geração, implementação e teste de sistemas de uma forma integrada e compacta.

Dentro de um sistema CASE pode-se identificar dois componentes básicos: software de gerenciamento dos ciclos de vida e ferramentas de desenvolvimento. O software de gerenciamento de ciclo de vida controla o planejamento, o gerenciamento e a execução do projeto de software.

As ferramentas de desenvolvimento dão suporte de gerenciamento técnico para cada etapa dentro do ciclo de desenvolvimento do software. É muito comum em sistema CASE a utilização de ferramentas especificas , uma para cada etapa do desenvolvimento, sendo utilizada pelo usuário de uma maneira extremamente transparente e automática.

A figura abaixo apresenta a arquitetura básica de um sistema CASE e cada bloco ali citado possui uma ferramenta básica e integrada de desenvolvimento permitindo assim uma automação estruturada (bloco); das atividades e etapas do desenvolvimento.

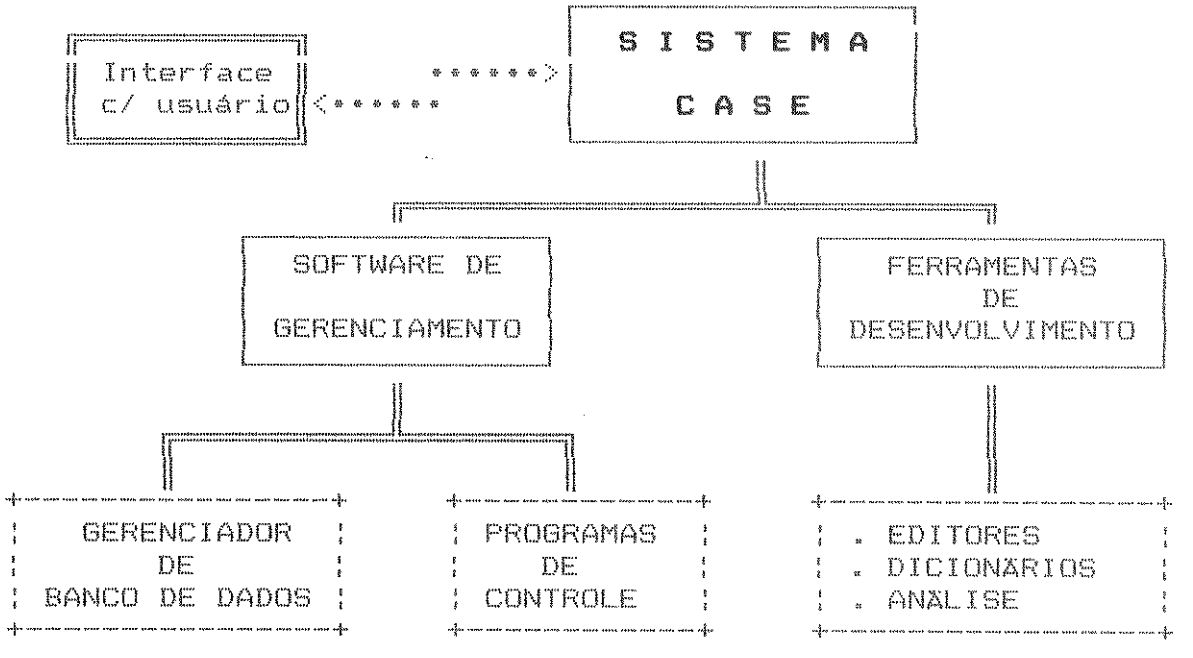


FIGURA 3.1 - ARQUITETURA DE SISTEMAS CASE

No contexto das ferramentas automatizadas cabe ressaltar a importância da atualização e gerenciamento do banco de dados do projeto. Esta atualização é que caracteriza os ciclos de vida do desenvolvimento.

3.3 - FERRAMENTAS DE DESENVOLVIMENTO (21)

O número e a variedade das ferramentas de desenvolvimento inseridas (integradas), em um CASE é praticamente ilimitada. Pode-se incluir desde um simples procedimento de inicialização até o gerenciamento e controle de uma etapa inteira dentro do ciclo de desenvolvimento do sistema.

3.3.1 - EDITORES GRAFICOS E DE TEXTO

É comum dizer que uma figura é melhor que cem palavras. Muitos sistemas avançados de desenvolvimento usam extensamente representações gráficas em seus projetos. Virtualmente todas essas representações são uma mistura de textos e gráficos concebidos de forma a expressar fluxos de dados, diagramas estruturais, etc de uma forma clara e precisa.

Essas ferramentas caracterizadas como editores de texto e gráficos constituem uma ferramenta largamente utilizada na geração e simulação do sistema a ser implementado. Alguns desses editores possuem uma característica de prototipagem permitindo ao projetista uma visualização e avaliação, a nível de protótipo, do sistema final. Esses protótipos não requerem uma compilação ou execução mas sim uma simples e rápida verificação permitindo a inserção direta do código do protótipo ao sistema final.

3.3.2 - DICIONARIO DE DADOS

Outra ferramenta bastante integrada ao CASE são os bancos de dados ou simplesmente dicionários de dados. Essa ferramenta aumenta substancialmente a funcionalidade para os elementos que desenvolvem sistemas. Os dicionários guardam informações sobre objetos ou simplesmente os elementos de dados.

Os dicionários podem também conter informações ou dados de onde outras informações contidas no mesmo banco poderão ser utilizadas dentro do programa principal.

O fluxo de dados dentro de um ambiente de desenvolvimento do tipo CASE também é contemplado dentro dessa estrutura de dados o que aumenta sensivelmente o poder de desenvolvimento; reduzindo o número de erros e duplicidade de módulos dentro do desenvolvimento.

3.3.3 - ANALISADORES E VERIFICADORES

Uma das ferramentas dentro do CASE que teve a sua maior e mais rápida evolução foram os ANALISADORES e verificadores.

Esse tipo de ferramenta tem como característica a avaliação e correção dos elementos que compõe o sistema a ser desenvolvido permitindo uma avaliação rápida e eficiente dos elementos do sistema através de um levantamento dos objetos e relações de interligação entre esses objetos, diminuindo a redundância e repetição dos módulos do sistema.

3.4 - AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE SOFTWARE

Dentro da definição clássica de sistemas do tipo CASE, sistemas ou ambientes que "geram" ou "criam" novos programas ou até mesmo outros ambientes podem ser classificados de acordo com a sua interação com o usuário e com o ambiente em que está inserido.

Neste trabalho buscou-se a "criação" de um ambiente de desenvolvimento de programas para teste e medidas elétricas voltado exclusivamente para a área de instrumentação microprocessada e "inteligente". Desta forma, o que caracteriza esse novo ambiente é uma forte integração e interação com o programador dos testes e medidas elétricas, sendo que as atividades do teste propriamente dito ficam por conta do usuário final.

O perfil do programador é definido como um engenheiro ou técnico especializado, que tem uma base mínima de conhecimento de linguagens de programação e de programação estruturada. Já o perfil do usuário final pode variar bastante, com uma característica comum, ele é especialista no teste para o qual foi gerado o programa.

A definição e opção por um "Ambiente Interativo de Desenvolvimento de Software" está baseado no perfil desse programador e do usuário final. Este tipo de ambiente permite uma grande interação com as ferramentas do ambiente, durante o desenvolvimento e implementação do programa.

Essa interatividade é conseguida através de uma interface Homem/Máquina de alto nível e da utilização conjugada das técnicas de "gerenciamento" por menus e linhas de comandos, que interferem no processo de criação do programador, orientando-o nos passos seguintes e corrigindo-o na etapa atual.

Não é objeto deste trabalho definir exaustivamente sistemas CASE ou outros sistemas de desenvolvimento de software. É evidente que novos desenvolvimentos e ferramentas de software, capazes de gerar novas e mais potentes ferramentas, estão em franco desenvolvimento nas mais diversas áreas da Engenharia de Software.

O AIDS_TME (Ambiente Interativo de Desenvolvimento de Software para Testes e Medidas Elétricas) é definido como um sistema tipo CASE, onde o fator de Interatividade é muito alto, com uma abrangência ainda não explorada na sua totalidade, ou seja, o número de áreas que este ambiente poderá ser ainda utilizado (Civil, Mecânica, Eletrônica, Eletrotécnica, Microeletrônica, Química, etc.).

4 - INSTRUMENTAÇAO E LINGUAGENS DE PROGRAMAÇAO

4.1. SISTEMA DE COMUNICAÇàO IEEE-488/78 (12-13-20)

O objetivo da NORMA IEEE-488/78 - "STANDART DIGITAL INTERFACE FOR PROGRAMMABLE INSTRUMENTATION" (11) é o de possibilitar a interligação de instrumentos programáveis de diferentes tipos e fabricantes, evitando a necessidade de adaptadores e numerosos tipos de cabos de ligação.

A norma provê um conjunto de regras para estabelecer um elo de COMUNICAÇAO entre instrumentos e controladores, com alto grau de confiabilidade e flexibilidade.

Nela são definidas:

- Características elétricas: parâmetros dos circuitos de entrada e saída, níveis lógicos, exigências de carga e terra;

- Características mecânicas: especificação do conector, alocação de contatos e montagem do cabo;

- Características funcionais: definição de cada linha do barramento, o protocolo de COMUNICAÇAO, o relacionamento de tempos utilizados para transferir todas as mensagens e a resposta esperada como resultado de recebimento destas mensagens.

Aplica-se a interfaces de sistemas de instrumentos ou a partes deles, nas quais:

- A troca de dados entre os equipamentos é digital;

- O número de dispositivos que estão interligados não excede a 15 unidades;

- O comprimento total do caminho de transmissão sobre os cabos de interligação não excede a 20 metros, ou 2 metros entre dispositivos;

- A taxa de dados através da interface em qualquer linha de sinal não excede a 1 Mbyte/segundo.

As especificações funcionais básicas da NORMA IEEE-488/78 podem ser utilizadas em aplicações de interfaces digitais que exigem maiores distâncias, maior número de instrumentos, ambientes que exigem uma maior imunidade ao ruído, ou diferentes especificações elétricas e mecânicas.

Nestes casos, são necessários dispositivos de adaptação para estender as especificações do Sistema 488.

A IEEE-488/78 especifica somente a operação do barramento e da interface. Ela garante que os dados e a informação de controle serão transferidas sem erros na COMUNICAÇAO dos instrumentos, mas não especifica, entretanto, a maneira como esta informação será interpretada pelo instrumento.

Para este fim, foi elaborada uma norma complementar, denominada IEEE-728, que trata dos códigos e formatos de transmissão. O Anexo IV deste trabalho apresenta um resumo da norma IEEE-488, sendo opcional a sua consulta.

4.2 - LINGUAGEM DE PROGRAMAÇÃO QuickBASIC (28)

A linguagem de programação conhecida como BASIC ("Beginners' All-purpose Symbolic Instruction Code") não possue uma boa reputação entre os mais experimentados programadores. Ela foi originalmente implementada para microcomputadores da linha IBM-PC como uma linguagem interpretada onde os programas desenvolvidos eram excessivamente lentos, ou seja cada linha do programa fonte requeria um número e era impossível aplicar o uso de "Labels", para identificar uma seção de códigos.

As facilidades de edição eram pré-históricas, existindo somente uma rudimentar estrutura e uma certa modularidade para PROGRAMAÇÃO. Muitos outros pontos puderam ser apontados como negativos ou insatisfatórios com relação a utilização do BASIC em programas longos e complicados, tornando a tarefa dos programadores árdua e frustantes.

Muitas coisas mudaram nos últimos anos, vários fabricantes de Software publicaram diversas versões do BASIC e a MICROSOFT desenvolveu um novo produto utilizando a sintaxe básica do BASIC e todas as ferramentas disponíveis das mais famosas e poderosas linguagens de programação produzindo o QuickBASIC, hoje já na versão 4.5.

As mais interessantes e avançadas facilidades e qualidades do QuickBASIC podem ser listadas como:

- Estruturas de variáveis definidas pelo programador;

- Suporta programação estruturada com funções e subprogramas;

- Um sofisticado editor que facilita em muitos aspectos a programação;

- Um sofisticado sistema de ajuda "On-Line Help", que permite informações de contexto de variáveis, sub-rotinas e do próprio QuickBASIC;

- Funções especiais para programação modular ;

- Suporta programas escritos em linguagens C, Assembler e utiliza bibliotecas definidas pelo usuário ou de outros aplicativos;

- Suporta um "Debugger", para análise e correção de programas.

A opção da utilização da linguagem de programação QuickBASIC no desenvolvimento e implementação desse trabalho, deve-se principalmente por razões históricas,

referentes aos programas desenvolvidos na área de instrumentação.

Na década de 80 mais de 90% dos instrumentos produzidos por fabricantes como: HP, TEKTRONIX, FLUKE, etc; possuíam interfaces de comunicação GPIB (IEEE-488) e todos os manuais de operação dos instrumentos possuíam programas explicativos e demonstrativos, nas mais diversas versões de BASIC, constituindo assim um elemento extremamente familiar aos usuários desses instrumentos.

Fica claro que a utilização do BASIC no desenvolvimento de programas para controle e comunicação com instrumentos tem uma excelente receptividade junto aos técnicos (usuários) desses instrumentos. Todos esses programas puderam ser implementados e adaptados a versão 4.5, do QuickBASIC.

Atualmente devido a disseminação da informática nos laboratórios e áreas de manutenção, tornou-se imperativo a utilização de uma ferramenta mais poderosa e versátil que o BASIC. A possibilidade de utilização de linguagens de programação como: C, Pascal, Fortran, etc no desenvolvimento de programas para instrumentação não é totalmente inviável devido ao fato da qualidade dos programas gerados, mas sim da necessidade de programadores extremamente competentes e com experiência na área de instrumentação.

Concluindo, a utilização da linguagem de programação QuickBASIC atende a esses dois requisitos sendo historicamente conhecida pelos usuários de instrumentação e com as mesmas qualidades e facilidades de programação das demais linguagens citadas. O Anexo V deste trabalho apresenta um resumo dos comandos e declarações do QuickBASIC (QB), sendo opcional a sua consulta.

4.3 - INSTRUMENTAÇAO COM INTERFACE GPIB (IEEE-488) (1-7-13)

A partir do final dos anos 60, quando a Hewellt Packard deu início à pesquisas na área de comunicação e automação de seus instrumentos, uma nova etapa dentro da Engenharia de Instrumentação estava por iniciar.

Com a introdução de Microprocessadores em seus equipamentos e/ou instrumentos por parte dos outros fabricantes, e de uma grande normalização por parte de entidades internacionais de usuários como o IEEE, ANSI, etc, o padrão de comunicação entre os instrumentos e um controlador (Microcomputador), teve a sua consolidação confirmada nos anos 70 e 80.

Todos os equipamentos produzidos por fabricantes da área de instrumentação, desde equipamentos cirúrgicos até grandes equipamentos de testes de componentes eletrônicos, tem instalados, diretamente pelo fabricante ("Built-in"), uma interface de comunicação no padrão IEEE-488 que permite a comunicação com computadores (controladores) que enviam comandos e mensagens a esses instrumentos transformando-os em máquinas "inteligentes".

A tônica dos anos 90 é a utilização de microcomputadores da linha IBM PC XT/AT como controladores desses instrumentos por motivos de custo e de ferramentas de software que permitem ao usuário a utilização de toda potencialidade de seus instrumentos.

4.3.1 - O SOFTWARE E OS INSTRUMENTOS (4)

Cada vez mais a busca por uma maior performance dos equipamentos que compõe uma estação de teste ou mesmo de uma simples bancada de um laboratório, conduz o usuário para uma especialização e utilização de ferramentas de software para desenvolver e implementar os programas de controle desses instrumentos.

Inicialmente, e por tradição foi utilizado a linguagem BASIC para o desenvolvimento desses programas e que possuia como controlador calculadoras eletrônicas com pouca capacidade de memória e pouquíssimas ferramentas de desenvolvimento.

Como já citado a utilização de Microcomputadores da linha IBM PC XT/AT abrem para o usuário um novo e espetacular horizonte de desenvolvimento com a real possibilidade de se utilizar de linguagens de programação de alto nível tais como: PASCAL, C, QUICKBASIC, FORTRAN, e até mesmo em ASSEMBLER (baixo nível) no desenvolvimento de seus programas de controle e testes.

O ambiente, ou seja o sistema operacional onde são gerados e manipulados os dados provenientes dos resultados dos testes e/ou ensaios realizados por esses instrumentos permite ao usuário a integração e utilização de outros aplicativos de software como: banco de dados, planilhas eletrônicas, editores de texto, etc na análise dos resultados e geração de relatórios finais.

4.3.2 - O HARDWARE DE COMUNICAÇAO - INTERFACE IEEE-488. (13)

O meio físico que permite a comunicação entre o controlador de instrumento, é baseado em uma interface de comunicação, conectada diretamente ao barramento do microcomputador e que tem como característica principal a adequação dos níveis de sinais elétricos que trafegam entre o microcomputador e o instrumento, e na geração e interpretação do protocolo de comunicação.

Essa interface foi desenvolvida para diversos tipos de microcomputadores (APPLE, IBM PC, COMODORE, etc), e por diversos fabricantes. A mais utilizada é a fabricada pela empresa National Instruments, Austin, Texas, USA, para microcomputadores da linha IBM PC XT/AT, e no Brasil esta interface é fabricada, até o presente momento (1991), somente pela empresa STD, de Brasília, DF.

4.3.3 - "DRIVER" E LINGUAGENS (20)

Nesse trabalho será utilizado como referência a interface de fabricação nacional denominada Cartão Controlador GPIB, modelo STD-8410 doravante referenciado somente pelo modelo.

O STD-8410 permite sua utilização em micros da linha IBM PC tanto XT como AT e é fornecido pelo fabricante vários módulos de comunicação para as linguagens BASIC, QUICKBASIC, C, PASCAL, FORTRAN e ASSEMBLER.

Esses módulos são denominados "Drivers" de comunicação e permitem a comunicação com os instrumentos, nos programas desenvolvidos pelos usuários, de uma forma transparente e altamente eficiente em termos de facilidade de programação e velocidade de comunicação (transferência de dados).

Normalmente os "Drivers" são utilizados ou carregados ao mesmo tempo com o programa desenvolvido pelo usuário e permite que chamadas de alto nível do tipo sub-rotina ou procedimentos sejam ativados diretamente pelo programa do usuário. Nas linguagens estruturadas tipo C, PASCAL e QUICKBASIC , essas rotinas são definidas como externas e na montagem do programa final "executável", são agregadas a este.

Para cada linguagem é fornecido um arquivo específico de declaração dessas rotinas e um arquivo tipo objeto, que possibilita ao usuário a geração de novos programas a partir desses módulos básicos permitindo a inclusão de novas rotinas definidas por ele próprio que irão compor o programa final.

O termo mais comum utilizado para identificar o "Device Driver" que contém as funçÕes e sub-rotinas do GPIB denomina-se "Handler", este termo será largamente utilizado nos itens que se seguem e tem como característica principal a sua individualização para cada linguagem utilizada, ou seja, cada modulo de declaração possui um arquivo tipo objeto (.OBJ) como "Handler", que deverá ser agregado ao programa do usuário ("Link").

4.3.4 - VALORES "DEFAULT" DOS PARAMETROS DE SOFTWARE

Toda a vez que o Sistema Operacional for inicializado no microcomputador ou a função IBONL for executada, alguns parâmetros de software são carregados com os seus valores "default". Esses valores e as respectivas funç0es usadas para modificá-los estão apresentados na tabela 4.1 abaixo.

PARAMETRO	DEFAULT
DMA ou E/S	DMA 1
Byte fim de Seqüência	0AH
Enviar EOI com o último byte	NAO
Bit de Status Individual	0
Mensagem de Poll Local	0
Endereço Primário	0
Habilitar Controlador do Sistema	SIM
Byte de Status de Poll Serial	0
Endereço Secundário	DESABI.
Sinal de Habilitação Remota	FALSO
"Timeout" em segundos	5
Endereço base p/ os cart0es STD-8410	2B8H
"Settling Time" p/ cart0es STD-8410	LONGO

TABELA 4.1 - VALORES "DEFAULT" DOS PARAMETROS DE SOFTWARE

5 - ESPECIFICAÇAO FUNCIONAL (22)

Respeitando os critérios de desenvolvimento dentro da metodologia da Engenharia de Software apresentada anteriormente, a etapa inicial de um desenvolvimento é obrigatoriamente a especificação funcional do objeto de desenvolvimento.

Neste trabalho a especificação funcional do desenvolvimento (produto) direciona-se para a especificação de um "Ambiente Interativo" com caracteristicas próprias e originais.

5.1 - INTRODUÇAO

O objeto deste trabalho é a especificação, implementação e teste operacional de um "Ambiente Interativo de Desenvolvimento de Software para Testes e Medidas Elétricas" - AIDS_TME, com características modulares e de alta reusabilidade de software (rotinas básicas) na produção de software. O aumento da produtividade e da qualidade dos programas gerados dentro deste ambiente é uma característica inerente a sistemas dessa natureza.

A modularidade é definida a nível de atividade, ou seja, a atividade de inicializar os instrumentos que irão participar do teste foi agrupada em um único módulo, definido como "SETUP", que contem um dado número de comandos para os instrumentos e para o ambiente. Este módulo será agrupado aos demais (Medidas, Gráficos, Planilhas) formando um programa único e original.

Baseado na reusabilidade de software estes módulos poderão ser utilizados e/ou agrupados em diferentes programas. Uma vez definido o módulo, este poderá ser utilizado em qualquer outro programa de teste, que tenha a necessidade da mesma seqüência de inicialização.

Estão incorporadas ao ambiente ferramentas de edição, compilação e "linker", sendo a sua utilização totalmente transparente ao usuário final. Arquivos específicos e programas de domínio publico também estão incorporados. A definição da abrangência deste ambiente está intimamente ligada as áreas da engenharia, pois todo e qualquer ensaio ou teste pode ser decomposto em módulos e produzidos a partir de módulos já existentes de programas anteriores.

Neste capítulo serão apresentados os requisitos básicos, requisitos funcionais e o diagrama em blocos do AIDS_TME, bem como as facilidades do ambiente.

5.2 - REQUISITOS BÁSICOS (23)

A primeira fase dentro de um desenvolvimento de software compreende a definição dos requisitos básicos que este ambiente deve atender.

5.2.1 - OBJETIVOS DO AIDS_TME:

O AIDS_TME é um ambiente interativo para desenvolvimento de software para teste e medidas elétricas utilizando instrumentação microprocessada com interface de comunicação IEEE-488 (GPIB).

5.2.2 - CONDIÇÕES NECESSÁRIAS PARA O FUNCIONAMENTO:

O AIDS_TME necessita basicamente, para funcionar, de um microcomputador compatível com a linha IBM PC XT/AT, um vídeo comum, com interface de comunicação IEEE-488 (GPIB).

5.2.3 - USUÁRIOS:

Os prováveis usuários do AIDS_TME serão técnicos especializados, engenheiros, estudantes de eletrônica que desejam gerar aplicativos de testes e/ou ensaios com comunicação e controle de instrumentos com interface de comunicação IEEE-488 (GPIB).

5.2.4 - RESULTADOS ESPERADOS:

Espera-se que a utilização do AIDS_TME proporcione uma melhora significativa no desenvolvimento de novos programas de testes e/ou ensaios.

5.2.5 - DADOS FORNECIDOS:

Os dados manipulados são aqueles transacionados com os instrumentos e o microcomputador, sempre de uma forma pura, onde o tratamento e/ou manipulação depende do programa do usuário (aplicativos).

5.2.6 - RELACIONAMENTO COM OUTROS PROGRAMAS:

Será o mesmo de programas e/ou aplicativos gerados por QuickBASIC e os dados gerados serão intercambiáveis por programas que manipulem dados no formato ASCII.

5.2.7 - VOLUMES ESPERADOS:

Os arquivos executáveis gerados pelo AIDS_TME não deveram ultrapassar a 130 K bytes e os dados gerados por este programa (executável) depende do tipo do teste e/ou ensaio, sendo o usuário responsável por esta alocação (dimensionamento de memória).

5.2.8 - PORTABILIDADE:

O ambiente de funcionamento está restrito a microcomputadores da linha IBM PC AT/XT e compatíveis, devido a placa de comunicação IEEE-488 (GPIB).

5.2.9 - OBJETIVO DE QUALIDADE, NORMAS E PADRÕES:

O AIDS_TME deverá ser concebido a fim de atingir algumas expectativas de qualidade, normas e padrões como:

- Assemelhar-se a outros ambientes de outras linguagens;

- Linguagem eletrônica;

- Rapidez na montagem de aplicativos;

- As operações no AIDS_TME deveram assemelhar-se a outros aplicativos.

5.2.10 - EXPECTATIVAS DE NECESSIDADES DE RECURSOS:

A configuração mínima para a utilização do AIDS_TME consiste em um microcomputador da linha IBM PC AT/XT, um vídeo comum, um disco rígido de no mínimo 20 M bytes e uma interface de comunicação IEEE-488 (GPIB).

5.2.11 - RESTRIÇÕES E REQUISITOS ADICIONAIS:

O AIDS_TME deverá ser desenvolvido em linguagem de programação QuickBASIC e utilizará bibliotecas especiais para manipulação de telas e "MOUSE".

5.3 - REQUISITOS FUNCIONAIS (25)

A especificação dos requisitos funcionais de um desenvolvimento de software incorpora parte da segunda fase do ciclo do desenvolvimento. Em conjunto com o diagrama em blocos, os requisitos funcionais fazem parte das especificações lógicas do desenvolvimento.

A figura 5.3.1 apresenta a seqüência de ações que compõem a execução de um teste e/ou ensaio tradicional. Neste processo o resultado do teste depende diretamente da ação do usuário, ficando este com a responsabilidade de gerenciar, executar e analisar os dados do ensaio para posteriormente gerar o(s) relatório(s) correspondente(s).

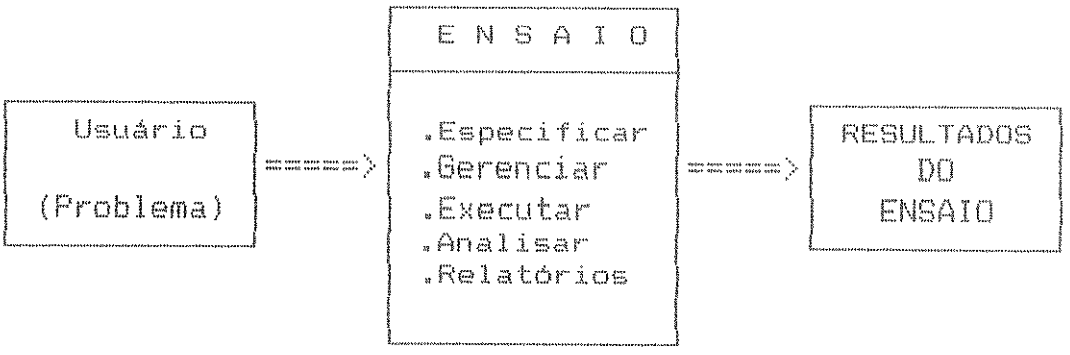


FIGURA 5.3.1 - TESTE E/OU ENSAIO TRADICIONAL

Como requisito funcional básico o AIDS_TME deve ser capaz de gerar programas que retirem do usuário as açÕes de gerenciamento e execução do teste propriamente dito, deixando para este somente as atividades de maior importância com a especificação do(s) teste(s) e análise dos resultados do(s) teste(s), utilizando-se dos relatórios finais, gerados automaticamente pelo programa.

As açÕes apresentadas na figura 5.3.2 mostram que o usuário passa a ter uma postura mais clara junto a execução do teste e/ou ensaio e de uma maneira mais objetiva. O usuário, com a utilização do AIDS_TME, deverá primeiramente gerar uma série de especificaçÕes sobre o teste e/ou ensaio a ser executado.

Está ação de especificação visa homogeneizar as atividades e obrigar o usuário a definir antecipadamente as açÕes a serem executadas automaticamente pelos instrumentos, ou seja, produzir procedimentos padrÕes de ensaios. A ação do programador é basicamente transpor as especificaçÕes do usuário em um linguagem de programação capaz de executar as atividades e açÕes especificadas pelo usuário.

Funcionalmente o AIDS_TME deve auxiliar e gerenciar o programador a gerar os módulos que deverão compor o arquivo executável do teste. Para o usuário final as etapas de transformação que estão inseridas no ambiente são totalmente transparentes.

Nesta filosofia o programador não necessita conhecer profundamente o ensaio que deverá ser executado e o usuário não precisa conhecer nada de programação. Cada um desses elementos tem agora a sua atividade bem definida e as áreas de "sombra" são totalmente "apagadas" proporcionando uma melhora significativa na produtividade e qualidade dos produtos (programas executáveis) e resultados (dados dos testes e/o ensaios).

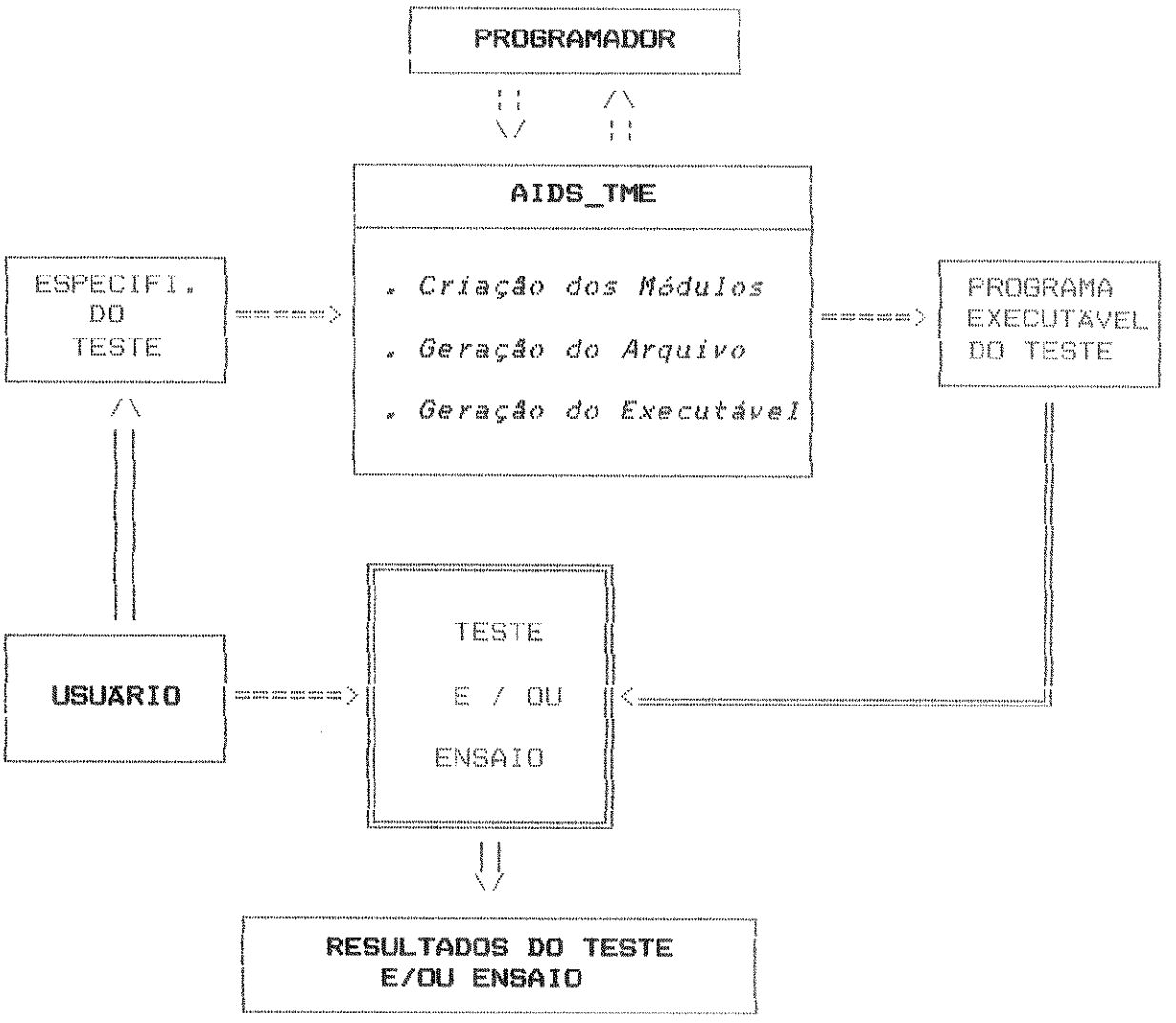


FIGURA 5.3.2 - AÇÕES COM O AIDS_TME

5.4 - DIAGRAMA EM BLOCOS

Funcionalmente o ambiente pode ser definido por quatro grandes blocos ou módulos. Esses módulos são totalmente independentes e possuem características próprias conforme a sua utilização.

A modularidade do software é a principal característica deste ambiente, permitindo um alto grau de reusabilidade e confiabilidade do software produzido (programas). Para atingir esse grau de eficiência o ambiente conta com um bloco capaz de gerar (editar), novos programas.

Este bloco, denominado EDITOR, é composto por quatro facilidades básicas ou seja editores de programas de inicialização, medidas, gráficos e planilhas. Esses editores possuem regras rígidas para geração dos programas, o que garante a qualidade do software produzido.

O segundo bloco tem como atividade principal a montagem do programa de teste e/ou ensaio a partir dos módulos gerados (editados) anteriormente. Essa montagem é realizada pelo programador de uma forma assistida e controlada pelo ambiente. Os módulos que irão compor o programa são escolhidos dentro de uma lista de arquivos já editados e testados.

Após a montagem (geração) do arquivo de testes, este deverá ser "compilado e lincado", produzindo um arquivo do tipo executável, arquivo este que será entregue ao usuário. Este é o terceiro bloco do ambiente.

O quarto e último bloco é composto basicamente das facilidades relacionadas com o ambiente propriamente dito e com a configuração e comunicação com os instrumentos através da interface GPIB.

A figura abaixo apresenta o diagrama em blocos do ambiente e procura salientar a característica básica deste desenvolvimento, que é a modularidade X reusabilidade.

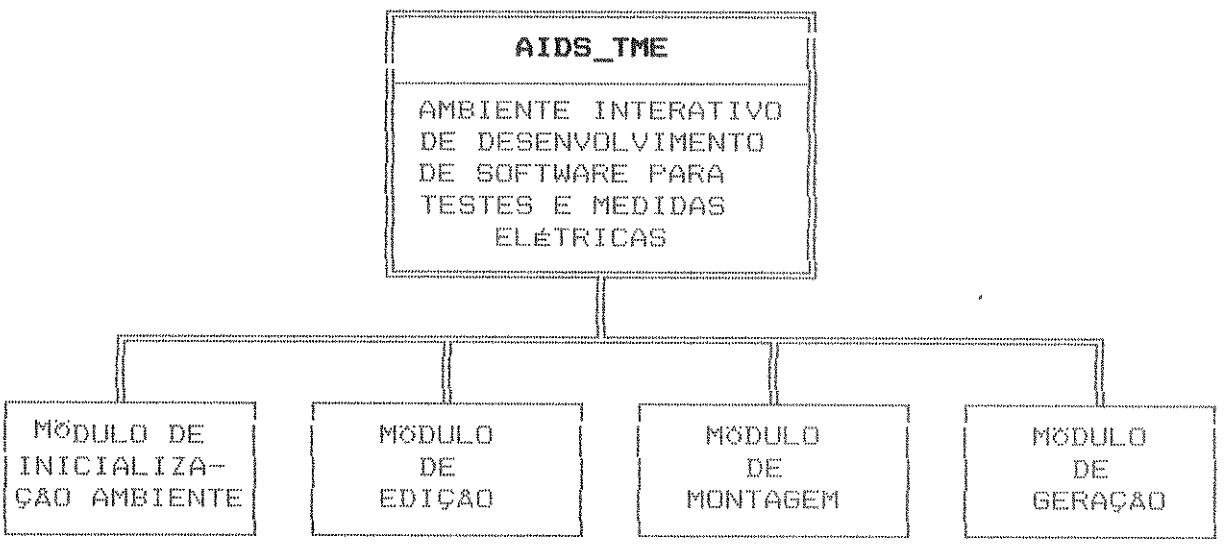


FIGURA 5.4.1 - DIAGRAMA EM BLOCOS DO AIDS_TME

5.5 - FACILIDADES DO AMBIENTE

Antes de iniciarmos a apresentação mais detalhada das funções do ambiente, cabe definir o termo "facilidade", que nesse trabalho será utilizado para identificar as opções e comandos do ambiente. Todas as facilidades do ambiente são descritas nesse capítulo de uma forma funcional e objetiva. Sua utilização e procedimentos serão detalhadas nos capítulos a seguir.

As facilidades serão apresentadas na seqüência abaixo e agrupados de acordo com os blocos definidos anteriormente.

5.5.1 - AMBIENTE

Nesta facilidade é definida a unidade ativa onde estão os programas módulos gerados pelo sistema e também a possibilidade de uma listagem do diretório ativo.

Devido as razões de segurança, cada operador do sistema possue uma senha de acesso, nesta facilidade poderão ser inseridos, alterados ou cancelados os operadores, todos os arquivos gerados levam o nome do operador que o criou.

5.5.2 - EDITORES

Todos os editores de inicialização, medida, gráfico e planilha estão agrupados nesta facilidade.

Funcionalmente este é o bloco mais importante de todo o sistema pois todas as regras e procedimentos para a geração e montagem dos módulos são gerados nessa facilidade.

Cada editor possue funcionalmente um conjunto de regras para a geração dos módulos. Esses módulos poderão ser recuperados, cancelados ou salvos na unidade ativa de acordo com a necessidade do usuário. Linhas de comandos auxiliam o programador na geração do módulo apresentando a sintaxe do comando a ser enviado ao instrumento.

5.5.3 - MONTADOR

Nesta facilidade os módulos gerados pelos editores são agrupados em um único arquivo com o objetivo de produzir o módulo ou programa final (executável).

A escolha dos módulos é feita através de uma lista com os nomes e conteúdo dos módulos de inicialização, medida, gráfico e planilha.

5.5.4 - "COMPILADOR" (GERAÇAO DO EXECUTAVEL)

Após a montagem do arquivo de teste pela facilidade anterior, o programador escolhe dentre os arquivos já editados e montados aquele que irá gerar o arquivo executável final.

Esse arquivo também é escolhido dentro de uma lista de arquivos já montados.

5.5.5 - ARQUIVOS

Nesta facilidade todos os arquivos montados ou os módulos que por ventura serão utilizados em outros arquivos de testes ou ensaios, serão editados, visualizados, impressos ou cancelados pelo sistema.

5.5.6 - CONFIGURAÇAO

Todos os procedimentos e programas para a comunicação e configuração dos equipamentos presentes no barramento GPIB estão contemplados nesta facilidade.

Uma ferramenta muito importante para a verificação e detecção de comandos ou erros de programação de acesso aos instrumentos GPIB, é incorporada a esta facilidade, permitindo uma rápida conferência e correção do programa gerado.

6 - AIDS_TME

O AIDS_TME (Ambiente Interativo para Desenvolvimento de Software para Teste e Medidas Elétricas) é classificado como um software capaz de simular um ambiente de desenvolvimento de software onde o programador pode interagir com as aç®es do sistema. Esta forma interativa de comunicação está baseada na utilização de menus do tipo "POP-DOWN" e linhas de comandos com mensagens de ajuda com a sintaxe correta de cada comando e/ou função.

Neste capítulo serão descritas todas as facilidades e opç®es do AIDS_TME, com as suas telas e menus especiais. O programador tem como ferramentas de desenvolvimento editores especiais para geração dos módulos de "SETUP", MEDIDAS, GRÁFICOS e PLANILHAS, compilador e "linker" para a linguagem QuickBASIC, versão 4.5, bibliotecas com as funç®es especiais de "Mouse", tela, menus, etc utilizadas no desenvolvimento dos programas para os usuários.

Todas as telas serão apresentadas e descritas na forma e/ou seqüência de utilização pelo programador sendo os procedimentos de operação descritos no próximo capítulo.

6.1 - SENHA DE ACESSO AO AIDS_TME

A primeira facilidade ao executar o AIDS_TME é a introdução de uma senha individual, com no máximo 10 caracteres, seguida do nome do operador (programador) que não deve exceder 15 caracteres.

A figura 6.1.1 apresenta a mensagem para a digitação da primeira senha. Caso seja a primeira interação com o AIDS_TME, o operador deve confirmar a senha para que está seja aceita pelo sistema. Somente serão aceitas 3 tentativas para o acesso ao sistema, após terceiro erro o sistema devolve o controle para o sistema operacional do microcomputador.

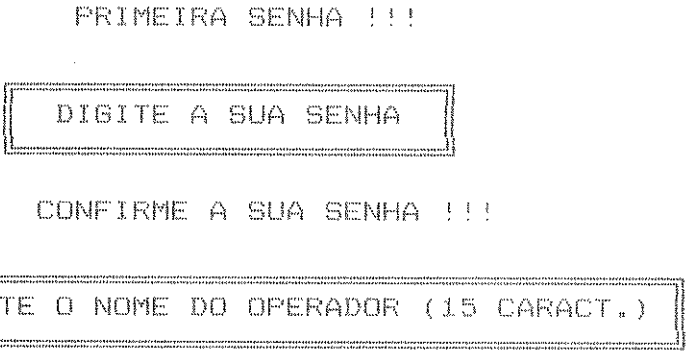


FIGURA 6.1.1 - PRIMEIRA SENHA

Após a primeira senha o sistema não solicita mais o nome do operador e caso esteja correta é dado acesso ao menu principal do AIDS_TME. Para a alteração da senha, pelo operador que à detem, basta digitar a palavra ALTERAR e executar a seqüência solicitada pelo sistema.

6.2 - MENU PRINCIPAL DO AIDS_TME

O primeiro contato com o ambiente será feito através da tela de menu principal, a figura 6.2.1 mostra esta tela e as suas principais facilidades (opções).

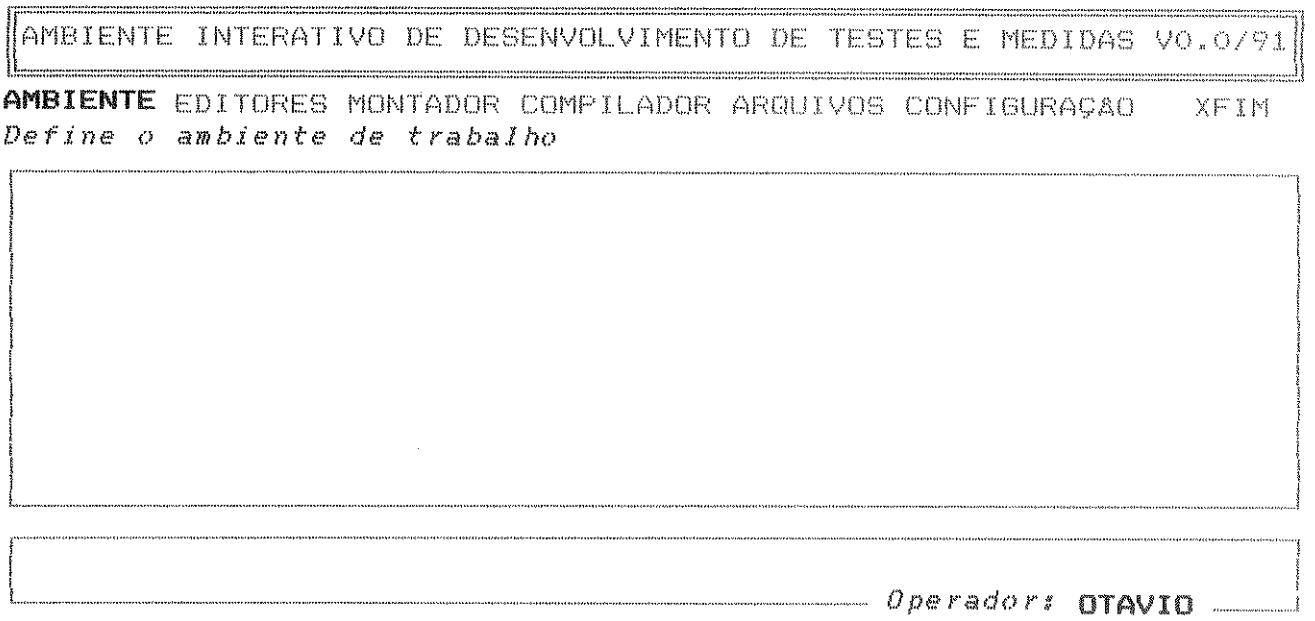


FIGURA 6.2.1 - MENU PRINCIPAL DO AIDS_TME

A linha imediatamente inferior á linha de opções contém a mensagem relativa a cada uma das opções. A área central da tela será utilizada para a edição e outras operações como a escolha e seleção de arquivos.

A última área da tela é reservada para as mensagens do ambiente para o operador (programador), como: erros, confirmações, final de operações, etc. Nesta área também aparece o nome do operador ativo (atual).

6.2.1 - AMBIENTE

Facilidade de configuração da unidade e do diretório de trabalho do ambiente onde serão arquivados os módulos de programa, listagens dos diretórios e insersão, cancelamento e alteração dos dados sobre o operador. *Mensagem: Define o ambiente de trabalho*

6.2.2 - EDITORES

Nesta facilidade estão agrupados os editores de "SETUP", MEDIDA, GRÁFICO e PLANILHAS. Este é o módulo mais importante do ambiente, pois estes editores foram especialmente desenvolvidos para a geração dos módulos, que são criados pelo programador, que iram compor o arquivo final de teste e/ou ensaio.

Todos os editores possuem uma série de menus e comandos específicos que atuam de forma interativa com o programador, orientando-o na geração do módulo e apresentando a sintaxe dos comandos á serem utilizados na comunicação GPIB/IEEE-488. *Mensagem: Editores de SETUP, MEDIDA, GRÁFICOS e PLANILHAS.*

6.2.3 - MONTADOR

Após a geração dos módulos que deveram compor o programa final de teste, estes devem ser agregados em um único arquivo. Nesta facilidade o arquivo final é montado através da utilização de um menu de escolha que permite escolher um entre vários módulos já gerados e editados pela facilidade anterior. *Mensagem: Montagem do arquivo de teste e/ou ensaio.*

6.2.4 - COMPILADOR

O usuário final irá receber somente um arquivo do tipo executável e não um arquivo tipo fonte na linguagem QuickBASIC. Nesta facilidade é possível gerar de forma automática este arquivo executável.

Através da utilização de menus de escolha, os mesmos utilizados nas facilidades anteriores, permitem a escolha de um entre vários arquivos já montados para ser compilado e "link" editado. *Mensagem: Geração do arquivo executável de teste e/ou medida.*

6.2.5 - ARQUIVOS

Muitas vezes é necessário executar algumas mudanças e/ou alterações nos módulos já editados ou até mesmo no próprio arquivo montado. Esta facilidade permite que os módulos e arquivos sejam visualizados ("MORE"), editados (EDITOR), impressos ou simples-mente cancelados. *Mensagem: Acesso aos arquivos e módulos editados.*

6.2.6 - CONFIGURAÇAO

Todas as operações e ações no barramento de comunicação GPIB (IEEE-488) são executadas nesta facilidade. Dois módulos específicos permitem o acesso aos instrumentos conectados ao barramento, bem como a sua configuração.

Outro módulo, especialmente desenvolvido, verifica no arquivo selecionado quais são os instrumentos acessados durante a execução do arquivo (programa) e apresenta, em tela, um quadro esquemático com a ligação entre os instrumentos e o controlador (microcomputador). *Mensagem: Configuração e comunicação com os instrumentos no barramento GPIB/IEEE-486.*

6.2.7 - XFIM

Ao término das operações o programador deve deixar o ambiente e executar o programa executável. Nesta facilidade todos os arquivos temporários são elimina-dos, os arquivos de dados são fechados e comando volta ao sistema operacional do microcomputador. *Mensagem: FINAL DAS OPERAÇÕES.*

6.3 - AMBIENTE

A facilidade AMBIENTE possuem um sub-menu de opçºes com : UNIDADE, DIRETORIO e OPERADOR conforme figura 6.3, essas opçºes podem ser acessadas através de um "mouse" ou pelas teclas de movimentação: esquerda, direita, acima e abaixo, ou aind pela primeira letra do nome da opção desejada.

A seguir são apresentadas as telas dos menus que compoem esta facilidade e os dados necessários para se executar a opção selecionada.

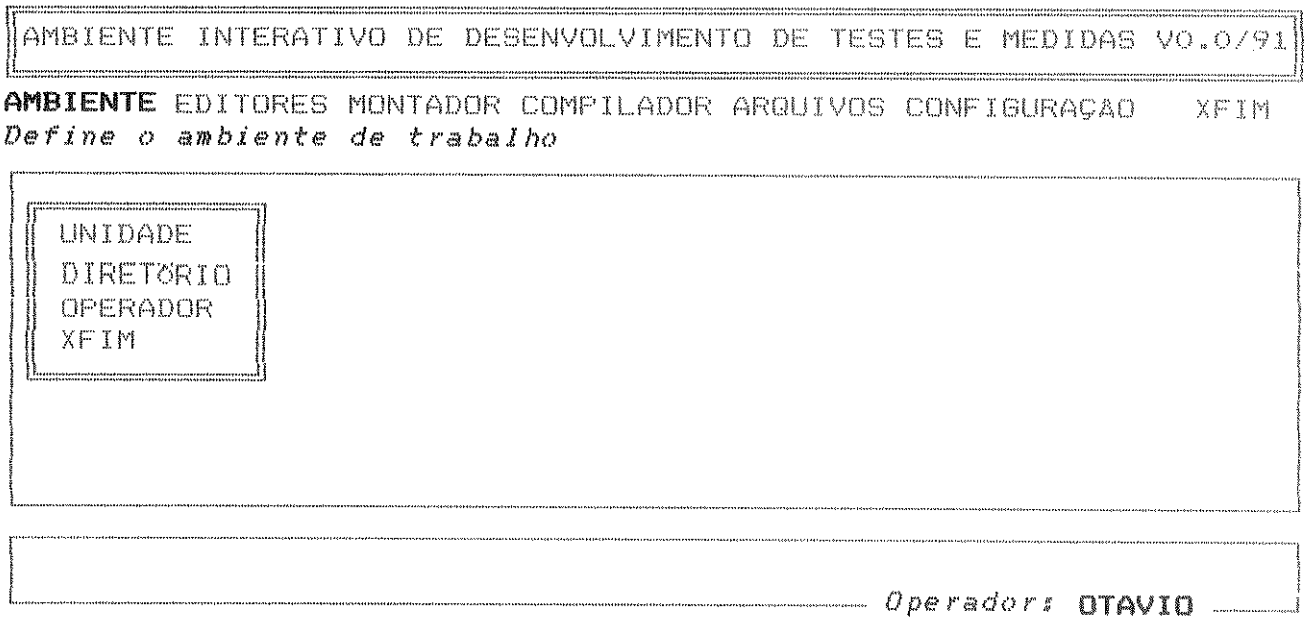


FIGURA 6.3 - MENU DO AMBIENTE

6.3.1 - UNIDADE

Nesta opção é selecionada a unidade ativa do ambiente, ou seja, qual o disco ou sub-diretório onde serão arquivados os módulos e os arquivos gerados pelo programador, figura 6.3.1.

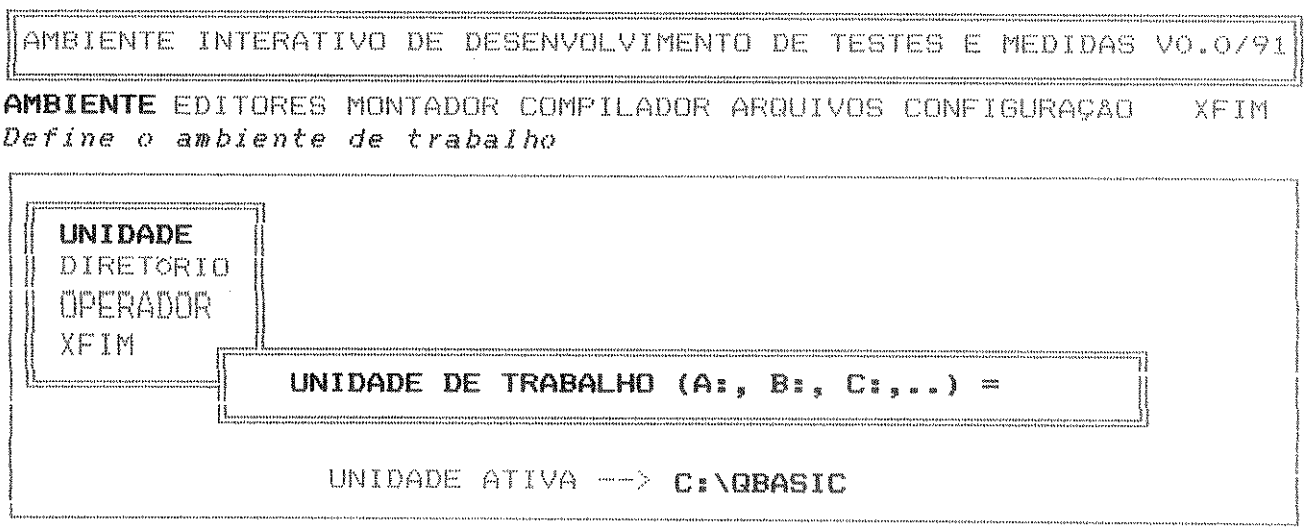


FIGURA 6.3.1 - MENU DO AMBIENTE - OPÇAO UNIDADE

6.3.2 - DIRETÓRIO

Após a seleção da unidade ativa basta ativar esta opção para se obter uma listagem do diretório da unidade ativa com os arquivos de extensão do tipo .BAS, ou seja, dos arquivos já gerados pelo programador.

6.3.3 - OPERADOR

Todos os módulos e arquivos gerados possuem o nome do programador, data, hora e descrição do módulo. Cada operador (programador) possuem um senha especifica e que pode ser altera e/ou cancelada.

Esta opção possue um sub-menu onde é possível inserir, alterar ou cancelar o acesso desse operador ao ambiente. O propósito desta facilidade é de garantir a segurança e confiabilidade dos módulos produzidos, protegendo-os de acessos indevidos ou de cancelamentos, figura 6.3.3.

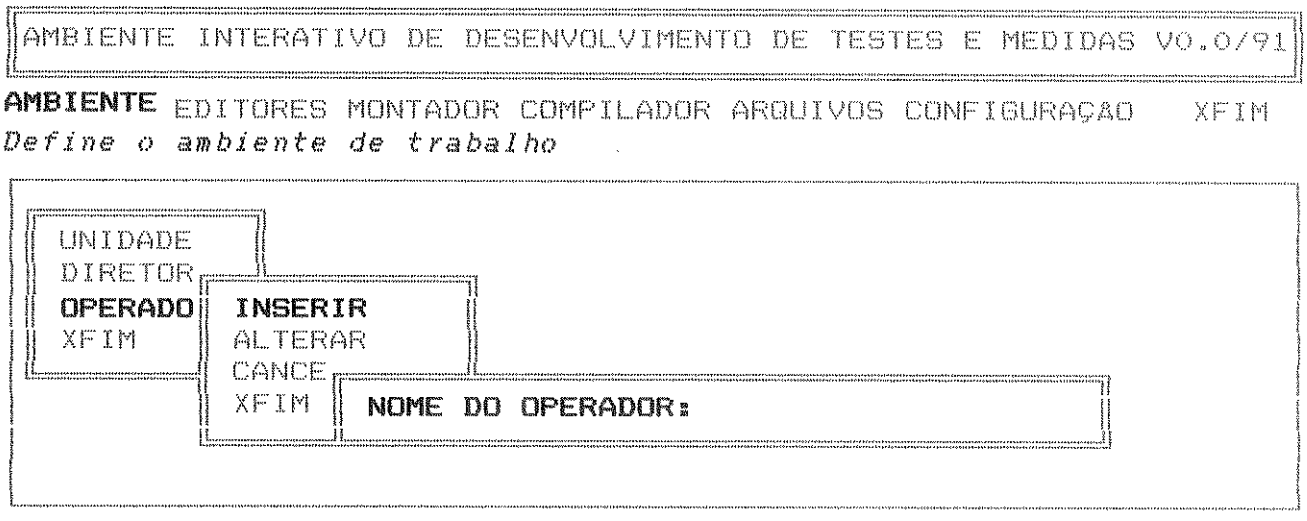


FIGURA 6.3.3 - MENU DO AMBIENTE - OPÇÃO OPERADOR

6.4 - EDITORES

Esta facilidade possue um sub-menu para a escolha do editor a ser utilizado, cada editor por sua vez, possue um sub-menu com as suas própria opç%es e linhas de comandos. A figura 6.4 apresenta o sub-menu dos editores.

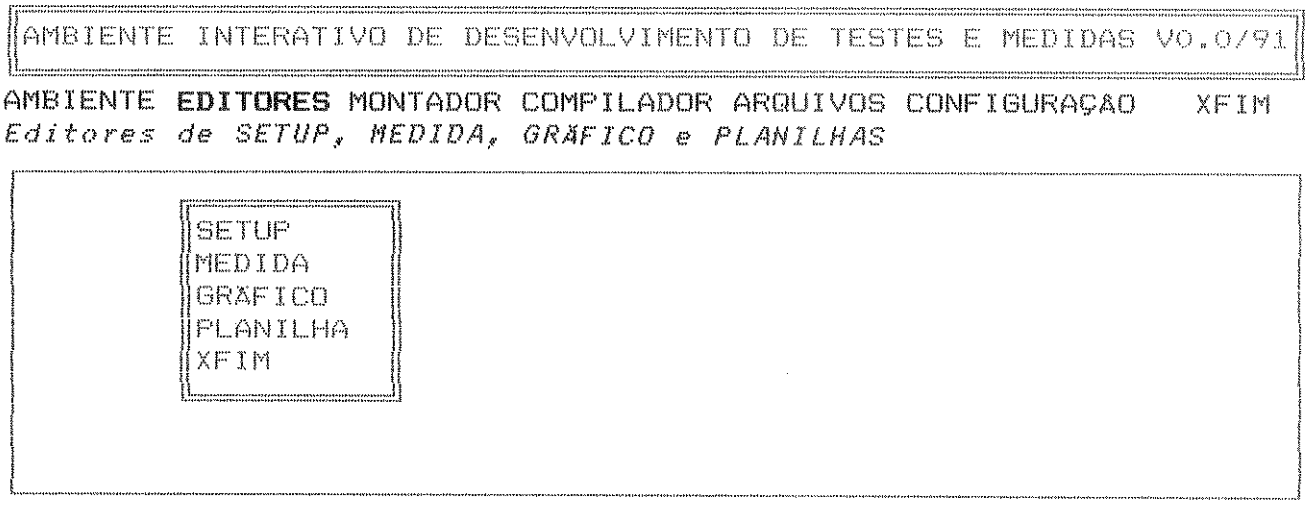


FIGURA 6.4 - MENU DOS EDITORES

6.4.1 - EDITOR DE "SETUP"

O editor de "SETUP" é subdividido em outros quatro sub-menus de ediçâo : 1. GRUPO, 2. GRUPO, 3. GRUPO e QBASIC. Cada sub-menu possue um sub-grupo especifico de comandos e estão ordenados de maneira que o programador pode acessa-los em qualquer sequência (seleçâo em anel).

A figura 6.4.1 apresenta o sub-menu do editor de "SETUP" (1. GRUPO). As opções deste sub-menu permitem ao programador salvar o módulo editado, cancelar os arquivos já editados ou recuperar (reinserir) módulos anteriormente editados. Ao acessar o editor de "SETUP'" o ambiente acessa automaticamente o primeiro grupo de comandos (1. GRUPO). São apresentados neste sub-menu os dados do operador, data, hora e o nome temporário do arquivo a ser editado.

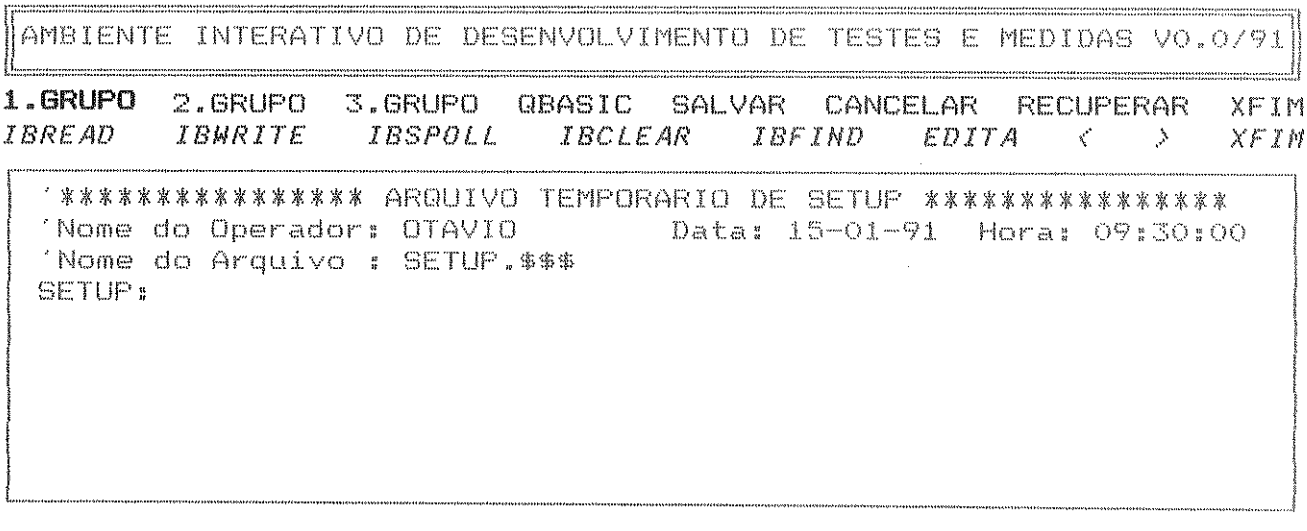


FIGURA 6.4.1 - MENU DOS EDITORES - ACESSO AO 1. GRUPO

6.4.1.1 - 1. GRUPO

No primeiro grupo estão os comandos relativos a escrita e leitura nos instrumentos conectados ao barramento. Quando da seleção do 1. GRUPO um novo sub-menu é apresentado, figura 6.4.1.1, onde estão listados os comandos e a sintaxe de cada um, conforme ANEXO VI.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

IBREAD IBWRITE IBSPOLL IBCLEAR IBFIND  EDITA  <   >   /\   \/  XFIM
Executa leitura em um instrumento

 '***************** ARQUIVO TEMPORARIO DE SETUP *****************
 'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
 'Nome do Arquivo : SETUP.$$$
 SETUP:
```

FIGURA 6.4.1.1 - 1. GRUPO

Um vez selecionado o comando o ambiente apresenta para o programador a sintaxe do comando escolhido. Após a digitação dos dados referentes ao comando o programador deverá confirmar a inclusão do comando editado no módulo ativo.

Os comandos serão inseridos seqüencialmente no módulo e ao final da programação o programador deverá inserir o comando do QuickBASIC "RETURN" para finalizar o módulo. A sintaxe dos comandos neste grupo é a seguinte:

```
IBREAD   --> CALL IBRD (DEVICE%, VAR$)
IBWRITE  --> CALL IBWRT (DEVICE%, VAR$)
IBSPOLL  --> CALL IBLSP (DEVICE%)
IBCLEAR  --> CALL IBCLR (DEVICE%)
IBFIND   --> CALL IBFIND (DEVICE%)
```

6.4.1.2 - 2. GRUPO

No segundo grupo estão os comandos relativos a comunicação com os instrumentos conectados ao barramento. Quando da seleção do 2. GRUPO um novo sub-menu é apresentado, figura 6.4.1.2, onde estão listados os comandos e a sintaxe de cada um, conforme ANEXO VI.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

IBIFC IBTRIG IBLOC IBSRE IBWAIT EDITA <  >  /\  \/ XFIM
Executa a inicialização da interface

'***************** ARQUIVO TEMPORARIO DE SETUP ****************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : SETUP.$$$
SETUP:
```

FIGURA 6.4.1.2 - 2. GRUPO

Igualmente aos comandos do 1. GRUPO estes serão inseridos seqüencialmente no módulo e ao final da programação o programador deverá inserir o comando do QuickBASIC "RETURN" para finalizar o módulo. A sintaxe dos comandos neste grupo é a seguinte:

```
IBIFC    --> CALL IBIFC (DEVICE%)
IBTRIG   --> CALL IBTRG (DEVICE%)
IBLOC    --> CALL IBLOC (DEVICE%)
IBSRE    --> CALL IBSRE (DEVICE%, VAR%)
IBWAIT   --> CALL IBWAIT (DEVICE%, MASK%)
```

6.4.1.3 - 3. GRUPO

No terceiro grupo estão os comandos relativos a execução de "serial poll" e "parallel poll" com os instrumentos conectados ao barramento. Quando da seleção do 3. GRUPO um novo sub-menu é apresentado, figura 6.4.1.3, onde estão listados os comandos e a sintaxe de cada um, conforme ANEXO VI.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

PPOLL CPOLL UPOLL LOCKOUT SENDCMD EDITA <  >  /\  \/ XFIM
Executa um "Parall Poll" nos instrumentos

'***************** ARQUIVO TEMPORARIO DE SETUP ****************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : SETUP.$$$
SETUP:
```

FIGURA 6.4.1.3 - 3. GRUPO

Igualmente aos comandos dos grupos 1. e 2., estes também serão inseridos seqüencialmente no módulo e ao final da programação o programador deverá inserir o comando do QuickBASIC "RETURN" para finalizar o módulo. A sintaxe dos comandos neste grupo é a seguinte:

```
PPOLL     --> CALL IBRPP (DEVICE%, RPP%)
CPOLL     --> CALL IBLPE (DEVICE%, LPE%)
UPOLL     --> CALL IBLPE (DEVICE%, LPE%)
LOCKOUT   --> CALL IBRTL (DEVICE%, VAR%)
SENDCMD   --> CALL IBCMD (DEVICE%, CMD$)
```

6.4.1.4 - QuickBASIC

Nesta opção o programador pode inserir qualquer comando ou função do QuickBASIC no corpo do módulo, conforme ANEXO V, a figura 6.4.1.4 apresenta o sub-menu para o "QBASIC" com a opção de ajuda "help on-line".

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

COMANDOS   AJUDA   EDITA                          <   >   /\   \/   XFIM
Comandos e funções do QuickBASIC

'***************** ARQUIVO TEMPORARIO DE SETUP *****************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : SETUP.$$$
SETUP:
```

FIGURA 6.4.1.4 - QBASIC

Igualmente aos comandos dos grupos 1.,2. e 3., estes serão inseridos seqüencialmente no módulo e ao final da programação o programador deverá inserir o comando do QuickBASIC "RETURN" para finalizar o módulo.

6.4.2 - EDITOR DE MEDIDA

O editor dos módulos de medidas possue as mesmas facilidades do editor de "SETUP" apresentado no item anterior. Somente os dados de identificação do módulo é que serão alterados, como o rótulo do rotina de medida. São apresentados neste sub-menu os dados do operador, data, hora e o nome temporário do arquivo a ser editado. A figura 6.4.2 apresenta o corpo interno do módulo de medida.

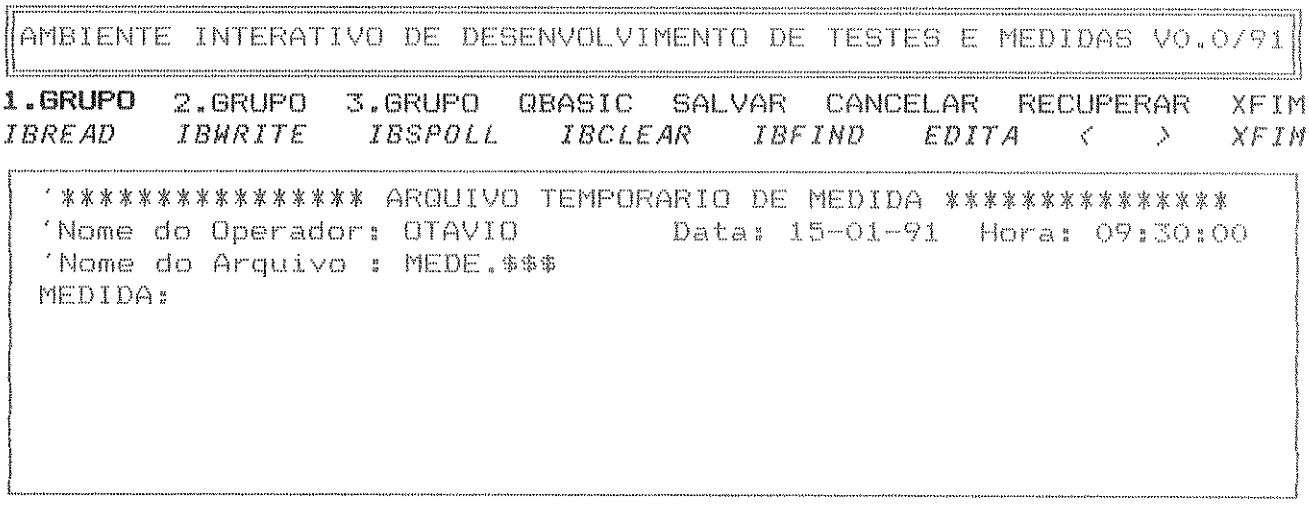


FIGURA 6.4.2 - MENU DOS EDITORES - MEDIDAS

6.4.3 - EDITOR DE GRÁFICOS

O editor de gráficos permite ao programador a geração de gráficos do tipo X-t e X-Y, semelhantes aqueles gerados por planilhas eletrônica, com definição de faixas, títulos e o arquivo que irá fornecer os dados para a geração do gráfico.

A figura 6.4.3.1 apresenta o sub-menu com as opções para a geração do módulo de gráfico que irá dar origem ao módulo final. Os módulos que contem as informações iniciais são do tipo temporários e posteriormente pode-se gerar o módulo que contém as rotinas para a geração do gráfico.

Todas essas rotinas são inseridas automaticamente pelo ambiente quando acionada a opção GERAR dentro do sub-menu do editor de gráfico. A seqüência básica de geração é inicialmente ativada com a criação de um módulo com extensão tipo .GRM (módulo temporário) e após a geração do módulo definitivo essa extensão é alterada para .GRA. A diferença básica entre os dois módulos é a inserção dos comandos e rotinas especiais pelo ambiente.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
TIPO ARQUIVO FAIXA TITULOS GERAR SALVAR CANCELAR RECUPERAR XFIM
Tipo do gráfico a ser gerado com o resultado das medidas

'**************** ARQUIVO TEMPORARIO DE GRAFICO ************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : GRAFO.$$$
GRAFICO:
 ARQUIVO:                                TIPO:
 PRIN.:
 SEC.:
 EIXO X:
 EIXO Y:

 FAIXAS ATIVAS:
 INICIO: 0%                              FINAL: 100%
```

FIGURA 6.4.3.1 - MENU DOS EDITORES - GRÁFICOS

O tipo do gráfico é escolhido através de um sub-menu (janela) com as opçÕes X-t e X-Y, o tipo escolhido é inserido automaticamente no arquivo.

O nome do arquivo de dados que contém as informaçÕes das medidas executadas deve respeitar o seguinte formato: os dados deveram estar codificados em ASCII e em 8 colunas seqüencialmente ordenados. O tipo dos dados inseridos neste arquivo devem ser do tipo precisão simples ("single precision"), definidos no módulo medida.

A escolha das faixas (colunas) que irão compor o gráfico é executa através de um sub-menu, figura 6.4.3.2, assim como o inicio e o final ("Range") da faixa de dados a serem utilizados na geração do gráfico.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
A    B    C    D    E    F    G    H    INICIO%    FINAL%    XFIM
Faixa de dados - A

'***************** ARQUIVO TEMPORARIO DE GRAFICO **************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : GRAFO.$$$
GRAFICO:
 ARQUIVO: SENO.DAD                        TIPO: X - t
 PRIN.:
 SEC.:
 EIXO X:
 EIXO Y:

 FAIXAS ATIVAS: A - B - C
 INICIO: 0%                               FINAL: 100%
```

FIGURA 6.4.3.2 - GRÁFICOS - ESCOLHA DAS FAIXAS

Para a formatação final do gráfico são escolhidos quatro títulos, que serão automaticamente inseridos pelo ambiente, correspondentes ao: título principal, secundário, do eixo X e do eixo Y, sendo este ultimo limitado em 20 caracteres. A figura 6.4.3.3 apresenta o sub-menu de preenchimento dos titulos.

Nesta opção também é possivel salvar, cancelar ou recuperar arquivos e/ou módulos já editados.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
PRINCIPAL      SECUNDARIO      EIXO X      EIXO Y      XFIM
Titulo principal do gráfico

'***************** ARQUIVO TEMPORARIO DE GRAFICO **************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : GRAFO.$$$
GRAFICO:
 ARQUIVO: SENO.DAD                        TIPO: X - t
 PRIN.: Ensaio de Base de Tempo
 SEC.:  Modelo 5334A
 EIXO X: Medidas
 EIXO Y: Frequência em Hz

 FAIXAS ATIVAS: A - B - C
 INICIO: 0%                               FINAL: 100%
```

FIGURA 6.4.3.3 - GRÁFICOS - ESCOLHA DOS TITULOS

6.4.4 - EDITOR DE PLANILHAS

O editor de planilhas permite ao programador a geração de planilhas semelhantes aquelas geradas por planilhas eletrônica, com definição de faixas, títulos e do arquivo que irá fornecer os dados para a geração da planilha.

A figura 6.4.4.1 apresenta o sub-menu com as opções para a geração do módulo da planilha que irá dar origem ao módulo final. Os módulos que contém as informações iniciais são do tipo temporários e posteriormente é gerado o módulo que conterá as rotinas para a geração da planilha.

Todas essas rotinas são inseridas automaticamente pelo ambiente quando acionada a opção GERAR dentro do sub-menu do editor de planilha. A seqüência básica de geração é inicialmente ativada com a criação de um módulo com extensão tipo .PLM e após a geração a extensão é alterada para .PLA. A diferença básica entre os dois módulos é a inserção dos comandos e rotinas especiais pelo ambiente.

O número de colunas da planilha é definido através de um sub-menu, na opção coluna, e pode ser escolhido entre 1 a 8 colunas, com no máximo 10 dígitos por coluna (8 colunas).

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

COLUNA ARQUIVO FAIXA TITULOS GERAR SALVAR CANCELAR RECUPERAR XFIM
Número de colunas que complem a Planilha

'***************** ARQUIVO TEMPORARIO DE PLANILHA ************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : PLANI.$$$
PLANILHA:
 ARQUIVO:                              COLUNAS:
 PRIN.:
 FAIXA A:                              FAIXA E:
 FAIXA B:                              FAIXA F:
 FAIXA C:                              FAIXA G:
 FAIXA D:                              FAIXA H:
 FAIXAS ATIVAS:
 INICIO: 0%                             FINAL: 100%
```

FIGURA 6.4.4.1 - MENU DOS EDITORES - PLANILHAS

O nome do arquivo de dados é inserido através da opção ARQUIVO. As faixas que irão compor a planilha são escolhidas na opção FAIXA, sendo respeitado o número de colunas escolhidos, ou seja, se o número de colunas é 2, somente duas faixas dentre as oito possíveis serão selecionadas.

O título principal e os títulos das faixas selecionadas são inseridos na opção TITULOS, semelhantes ao menu da figura 6.4.3.3 - seleção de títulos para o gráfico.

Todas as facilidades como: salvar, cancelar ou recuperar arquivos ou módulos também são implementados neste editor.

6.5 - MONTADOR

Após a geração dos módulos de "SETUP", MEDIDAS, GRÁFICOS e PLANILHAS o programador deve agrega-los em um único arquivo, extensão .BAS, para que este execute o teste especificado pelo usuário.

Esta filosofia está baseada na reusabilidade total de software onde módulos gerados para testes anteriores, mas que utilizam os mesmos instrumentos na sua execução, possam ser modificados ou simplesmente reutilizados em um novo programa.

Todos os módulos são arquivados em formato ASCII e possue um nome e descrição especifica para sua identificação. Os nomes e descriçºes dos módulos são arquivados e acessados durante a montagem do arquivo final.

A figura 6.5.1 apresenta o sub-menu da facilidade MONTADOR, onde o arquivo final será montado com os módulos gerados anteriormente. A escolha do módulo é feita através da apresentação de uma lista com o nome do módulo, nome do operador (programador), descrição do módulo, unidade e diretório onde o módulo está arquivado.

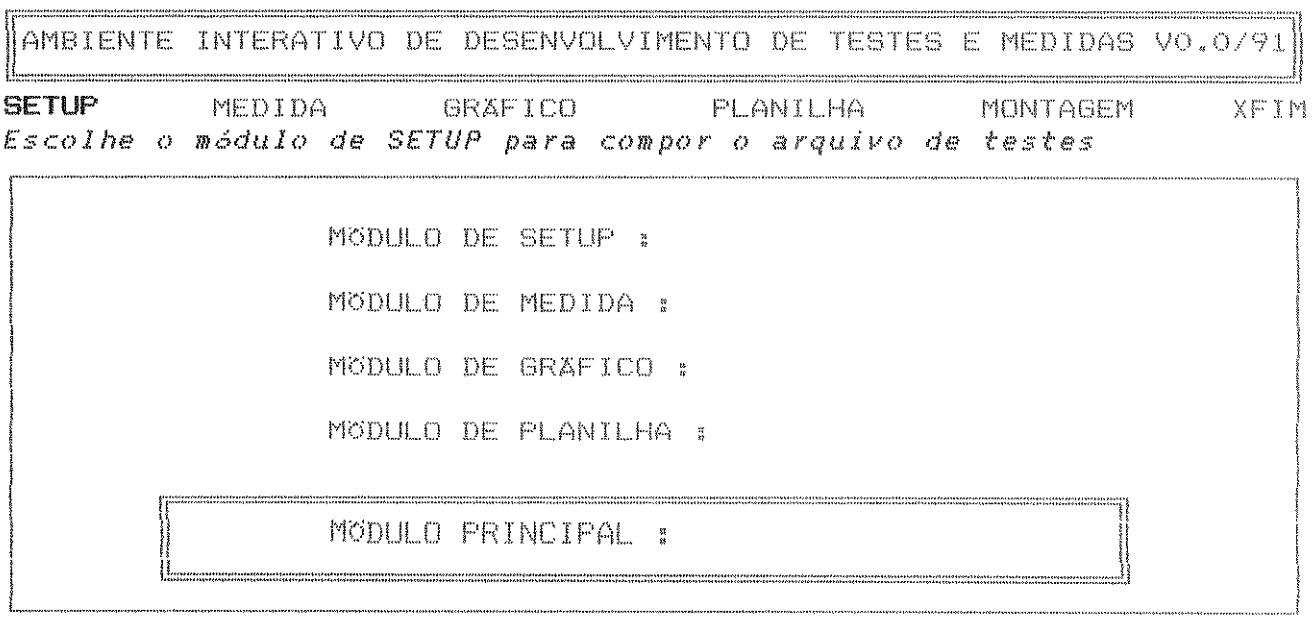


FIGURA 6.5.1 - MENU DO MONTADOR DE ARQUIVOS (.BAS)

Caso o programador não queira inserir todos os quatro módulos no arquivo final basta deixar em branco o nome do módulo, ou seja, não executar a busca em diretório daquele módulo. O ambiente colocará em seu lugar um arquivo com mensagem ao usuário que este módulo não foi implementado pelo programador.

Após a montagem do arquivo, com a escolha dos módulos, o programador deverá executar a opção "MONTAGEM", onde o ambiente se encarregará de "montar" o programa principal inserindo novas rotinas e procedimentos padrões para completar o novo arquivo, agora com extensão .BAS.

O arquivo gerado também contém os dados do operador, descrição e diretório, estes dados são inseridos em uma tabela de arquivos, que serão posteriormente utilizados quando da geração do arquivo executável, através da facilidade COMPILADOR.

6.6 - COMPILADOR

O usuário irá receber somente um arquivo do tipo executável para a realização do teste e/ou ensaio, desta forma o ambiente deve implementar automaticamente a geração desse programa.

A facilidade COMPILADOR permite ao programador que este selecione e gere o arquivo executável necessitando somente escolher o nome do arquivo montado (.BAS) apresentado em forma de uma lista de diretório ao programador.

A escolha do arquivo montado pode ser feita através de um "mouse" ou por tecla de movimentação (acima e abaixo). Este tipo de escolha é padrão dentro dos menus do ambiente e após o programador escolher o arquivo basta digitar a confirmação para que o arquivo seja compilado e "link"-editado.

O compilador utilizado é compatível com a versão 4.5 do QuickBASIC da MicroSoft, bem como o "link"- editor. Foi desenvolvida e implementada um biblioteca de funções e sub-rotinas especiais para o ambiente. Nesta biblioteca estão as rotinas para "mouse", montagem de telas, abertura e fechamento de arquivos, procedimentos de inicialização e busca de módulos ,etc.

A figura 6.6.1 apresenta o sub-menu para a geração do arquivo executável, o arquivo a ser compilado estará em destaque no vídeo e para a sua seleção basta digitar "ENTER" e confirmar a sua escolha.

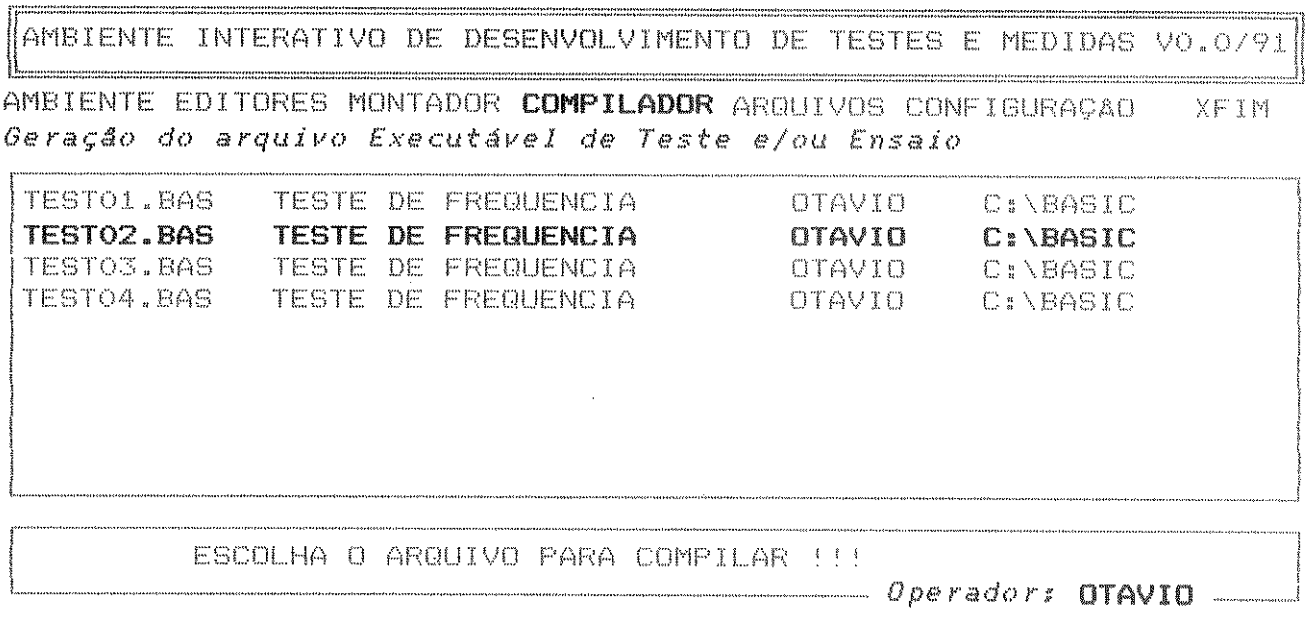


FIGURA 6.6.1 - COMPILADOR

Esta é a última etapa para a geração do arquivo executável de acordo com as especificaçÕes iniciais do usuário. Caso o arquivo gerado possua algum erro ou não atenda as especificaçÕes o programador pode voltar a qualquer uma das etapas anteriores e recomeçar a geração do arquivo final, otimizando assim a geração de protótipos e permitindo o teste de novas configuraçÕes e situaçÕes para o ensaio.

6.7 - ARQUIVOS

A facilidade ARQUIVOS permite ao programador editar, visualizar, imprimir ou cancelar módulos ou arquivos já montados. A figura 6.7.1 mostra o sub-menu para a escolha de um grupo de módulos ou arquivos montados.

FIGURA 6.7.1 - ARQUIVOS

Uma vez escolhido o bloco a ser manipulado um novo sub-menu é apresentado, figura 6.7.2, e o sistema de escolha do módulo ou arquivo é o mesmo das facilidades anteriores.

A edição utiliza um editor de tela completa e permite executar qualquer alteração que o programador desejar. A visualização utiliza a função "MORE" do MS-DOS e apresenta em vídeo blocos de informações do arquivo ou módulo selecionado.

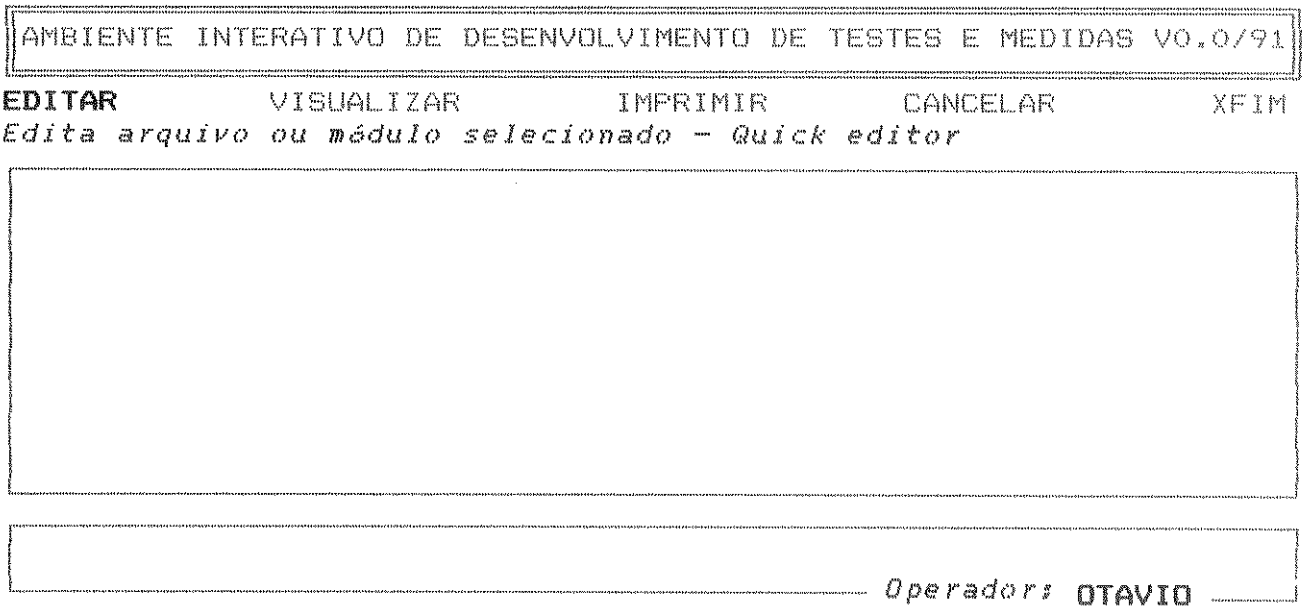


FIGURA 6.7.2 - ARQUIVOS SUB-MENU PARA OPÇÕES

O programador pode imprimir integralmente um módulo ou arquivo diretamente do ambiente, função "PRINTER" do MS_DOS. Caso o programador deseje cancelar, por qualquer motivo um módulo ou arquivo, basta executar a opção de CANCELAR e escolher o módulo ou arquivo dentro do sistema padrão de escolha, já exemplificado anteriormente.

6.8 - CONFIGURAÇAO

O programador muitas vezes tem necessidade comunicar-se com os instrumentos que estão conectados e configura-los no barramento GPIB (IEEE-488). A facilidade CONFIGURAÇAO permite que o programador acesse diretamente estes instrumentos através de um software especifico, desenvolvido pela NATIONAL INSTRUMENTS, Austin, Texas, USA (13), que acompanha a placa de interface PC/GPIB (STD 8410).

Outra opção é a re-configuração dos dados referentes aos instrumentos conectados ao barramento. Este software também acompanha a placa de interface PC/GPIB (STD 8410).

Esses dois softwares, IBIC e IBCONF, respectivamente, possuem uma documentação completa e menus de ajuda "help on-line" de suas funções e procedimentos, fornecidos pelo fabricante, neste trabalhos somente referenciados. A figura 6.8.1 apresenta o sub-menu de configuração.

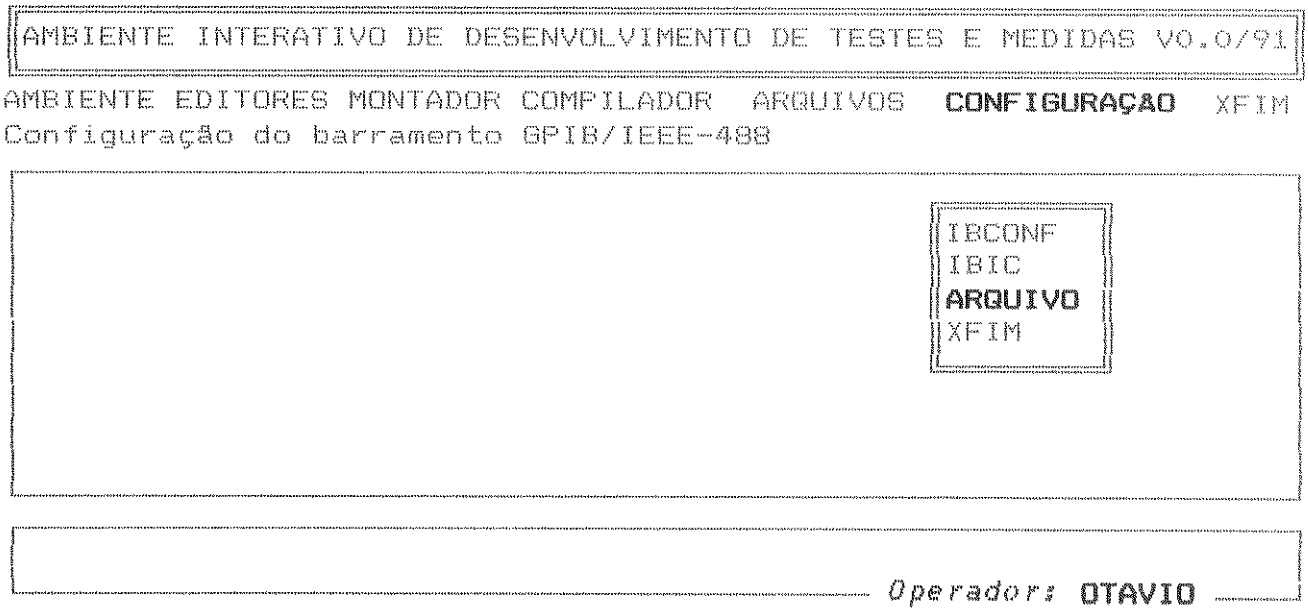


FIGURA 6.8.1 - CONFIGURAÇAO

A terceira opção deste sub-menu, ARQUIVO, foi especialmente desenvolvida para o ambiente, permitindo ao programador verificar em poucos segundos quais os instrumentos referenciados e/ou acessados durante a execução do teste e/ou ensaio.

A figura 6.8.2 exemplifica um arquivo montado que acessa somente um instrumento, o contador universal da HP, modelo 5334A. A escolha do arquivo a ser verificado é a mesma das facilidades anteriores, padrão do ambiente.

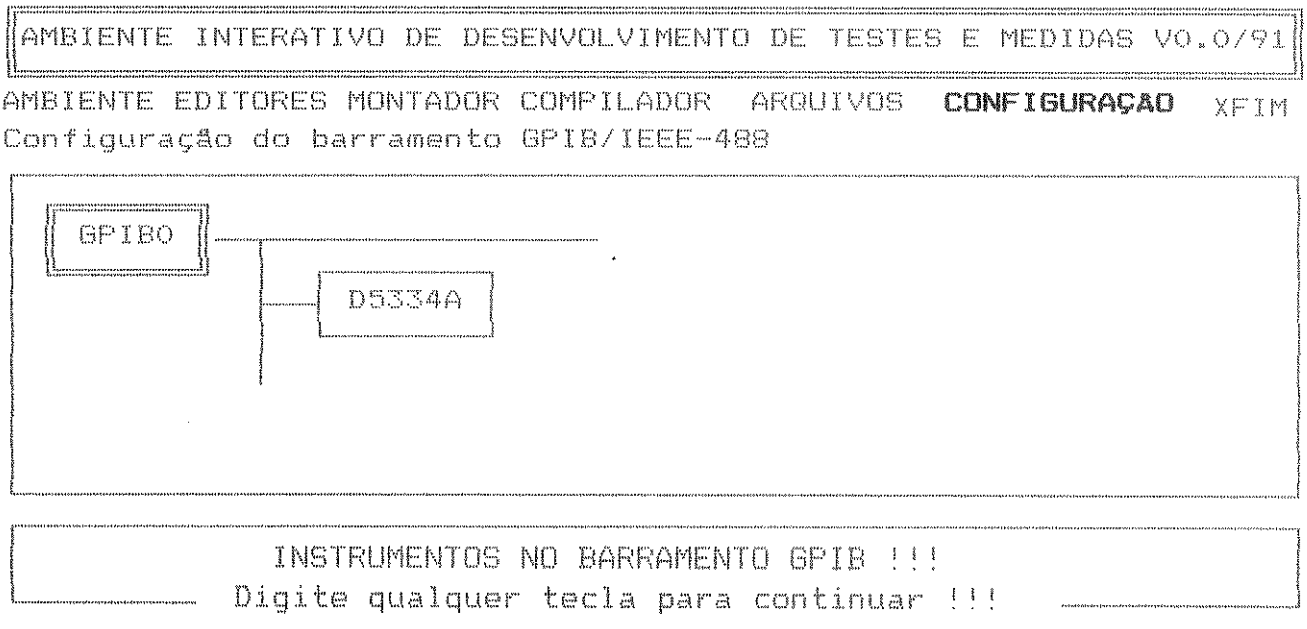


FIGURA 6.8.2 - VERIFICAÇAO DOS INSTRUMENTOS NO BARRAMENTO GPIB

6.9 - XFIM

Ao término das operaçÕes o programador deve deixar o ambiente e testar o programa criado. Devido a capacidade de memória do microcomputador não é possível testar o programa executável "dentro" do ambiente.

A facilidade XFIM fecha todos os arquivos abertos, elimina os arquivos temporários e volta o controle para o MS-DOS. Todos os menus e sub-menus do ambiente terminam suas atividades através da mesma facilidade (XFIM).

7 - MANUAL DO PROGRAMADOR - PROCEDIMENTOS GERAIS

Neste capítulo serão abordados tópicos específicos sobre o desenvolvimento e programação dentro do AIDS_TME visando otimizar a atividade do programador.

O programador tem toda a potencialidade do AIDS_TME a sua disposição o que permite aumentar o seu desempenho no desenvolvimento e testes de novos protótipos de software e/ou variações dentro de um mesmo teste e/ou ensaio.

Toda a filosofia do AIDS_TME esta baseada na reusabilidade de software e na geração rápida e eficiente de protótipos utilizando-se módulos já desenvolvidos e testados. Nos itens que se seguem serão abordados tópicos importantes para o auxilio do programador no desenvolvimento de programas.

7.1 - AMBIENTE

O AIDS_TME deve ser instalado em um sub-diretório de um disco rígido (Winchester), devido a quantidade e volume das bibliotecas utilizadas pelo ambiente. Os arquivos necessários para a sua instalação estão contidos em discos de 5 1/4", 360 Kbytes.

Neste sub-diretório serão automaticamente criados arquivos temporários e arquivos de dados, com extensão .DAT, onde será inserida a relação dos módulos e arquivos à serem desenvolvidos. A cada desenvolvimento um módulo ou arquivo um dado (nome) é inserido ou cancelado dentro desta lista.

Na opção de escolha de módulos e arquivos, padronizadas pelo AIDS_TME, essa lista é apresentada em forma cronológica de geração dos arquivos. O ambiente não reconhecerá arquivos ou módulos que não tenham sido gerados por um dos seus quatro editores.

Com relação ao cadastramento de novos programadores, ao se inserir um novo operador (programador) ele se tornará imediatamente ativo, para o retorno ao operador inicial basta reiniciar o AIDS_TME.

Os módulos e arquivos de teste desenvolvidos podem ou não estar no mesmo sub-diretório do AIDS_TME, sugere-se, por segurança, colocar esses arquivos em disco flexíveis (A:, B:) ou em outro sub-diretório do disco rígido.

7.2 - EDITORES DE "SETUP" E MEDIDAS

Os editores de "SETUP" e MEDIDAS tem a mesma filosofia de desenvolvimento. Todos os comandos necessários para a comunicação e controle dos instrumentos GPIB/IEEE-488 estão disponíveis nestes editores.

O programador deve inicialmente criar (editar) o módulo a ser desenvolvido e finaliza-lo com o comando "RETURN" do QuickBASIC. A filosofia de geração do arquivo final está baseada em sub-rotinas, em número de 4, e um programa como corpo principal. Quando o programador recupera um arquivo

já salvo os novos comandos serão inseridos imediatamente abaixo da ultima linha editada no arquivo.

Para maiores alteraçºes no conteúdo dos módulos sugere-se a utilização do editor de tela inteira, Quick Editor, disponível na facilidade arquivos, dentro do próprio AIDS_TME.

Caso o programador possua o programa QuickBASIC, versão 4.5 ou mais recente, e tenha um bom conhecimento de programação, pode-se utilizar o próprio editor do QuickBASIC, após a geração do arquivo pelos editores de "SETUP" e MEDIDAS, a operação inversa não é permitida.

7.3 - EDITOR DE GRÄFICO

O programador deve ter em mente que os dados gerados durante o procedimento de medidas e que serão utilizados na geração do gráfico devem respeitar o formato definido pelo AIDS_TME.

Os dados devem ser do tipo precisão simples, "single precision", formatados em 8 colunas seqüenciais. Caso não sejam utilizadas as 8 colunas, estas deverão ser preenchidas com o valor zero (0), completando o total de 8 colunas. O gerador de gráficos necessita das 8 colunas para inicializar as variáveis do programa de teste e/ou ensaio.

O programador deve utilizar os comandos de "OPEN" e "CLOSE" do QuickBASIC para gerar o arquivo de dados e deve utilizar a extensão .DAD para estes arquivos. Estão disponíveis na biblioteca de funçºes do ambiente sub-rotinas e funçºes especiais que o programador pode acessar a qualquer momento, dentro dos módulos. (TOOLBOX.LIB) para realizar estas açºes.

Os arquivos de dados são gerados durante o procedimento de medida, portanto é muito importante o programador manter um padrão na geração desses arquivos facilitando a reutilização desses módulos em novos programas.

O gráfico gerado poderá ser salvo em disco, impresso ou transferida para editores de texto ou outros aplicativos, pelo usuário final, quando da utilização do arquivo executável.

7.4 - EDITOR DE PLANILHAS

Na geração das planilhas o programador deve seguir a mesma metodologia do editor de gráficos para a geração do arquivo de dados. Estes arquivo pode ou não ser o mesmo utilizado no editor anterior.

O número de faixas utilizadas para a inicialização das variáveis do programa também é em número de 8. A escolha do número de faixas é feita através da opção COLUNAS, 1 a 8, e os dados existentes no arquivo serão simplesmente copiados para as colunas.

Caso o usuário deseje executar qualquer manipulação com os dados (médias, desvios, divisºes, multipicaçºes,

filtragem, etc) está deverá ser executada no módulo de medida. O programador deve montar o arquivo de dados já com os dados preparados e formatados para a geração da planilha.

Os nomes (identificação) de cada coluna são inseridos durante o procedimento de geração do módulo da planilha. Na opção TITULOS são inseridos esses nomes e o programador deve ter em mente que a largura (tamanho) de cada coluna depende diretamente do número de colunas escolhidas, ou seja, largura do nome da coluna = (80-(NC+1)/NC), onde NC = número de colunas.

Para o número máximo de 8 colunas, o tamanho dos títulos das colunas é de 7 caracteres. A planilha gerada poderá ser salvar em disco, impressa ou transferida para planilhas eletrônicas, pelo usuário final, quando da utilização do arquivo executável.

7.5 - MONTADOR

A principal recomendação para a montagem do programa de teste e/ou ensaio ao programador é a de que estes sempre utilize a "DESCRIÇAO" dos módulos de uma forma completa e clara. Quando da montagem é importante o programador reconhecer rapidamente o módulo e identificar o seu conteúdo.

A não utilização de todos os módulos na montagem do arquivo não trás nenhum problema na geração. O AIDS_TME insere automaticamente os módulos não utilizados, com mensagem ao usuário, mencionado que este módulo não foi implementado naquele programa executável.

O programador pode ou não utilizar o mesmo nome dos módulos para o programa final. Quando da "MONTAGEM" o AIDS_TME solicita ao programador um novo nome e descrição para constar na tabela dos arquivos já montados.

7.6 - COMPILADOR

O objetivo maior do AIDS_TME está na simplificação e automatização das atividades rotineiras e suceptível a erro do programador (operador). Na facilidade "COMPILADOR" o programador deve simplesmente escolher o arquivo já montado dentre a lista apresentada pelo ambiente e confirmar a execução da atividade.

Ao final da "compilação" e "link - ediçao" do arquivo o ambiente apresenta no video o resultado, em caso de erro as mensagens correspondentes, e espera pela digitação de qualquer tecla para voltar ao menu principal.

O programador deve estar atento as essas mensagens e anota-las e/ou imprimi-las para executar a correção no módulo correspondente.

Estima-se um redução nos erros de programação devido a reusabilidade dos módulos (software) e na manutenção dos mesmos. Uma vez corrigido o erro estes não irá se propagar para outros programas ou arquivos.

A seqüência de correção é bastante simples e o programador deve simplesmente editar o módulo, pelos editores do ambiente ou pelo Quick Editor, corrigi-lo e executar novamente o procedimento de geração (MONTADOR) e em seguida repetir o procedimento de "compilação" (COMPILADOR).

7.7 - ARQUIVOS

Nesta facilidade estão disponíveis ao programador opções para editar, visualizar, imprimir e cancelar módulos e arquivos já montados.

7.7.1 - EDITAR

A recomendação maior ao programador quando da edição dos módulos é que este não utilize processadores de textos que "formatem" o arquivo, ou seja, coloquem caracteres de controle ou acentuação em algumas palavras. O compilador não reconhece estes caracteres gerando mensagens de erros para cada alteração executada.

Outra regra básica é a não alteração do nomes dos módulos ou sub-rotinas existentes dentro dos módulos ou do próprio arquivo montado. Certamente isto fará com sejam geradas mensagens de erro durante a compilação e o programa final não irá funcionar.

O editor de texto que acompanha o AIDS_TME possuem um bom menu de ajuda ("help on-line") e pode ser acionado a qualquer momento através da tecla "F1".

7.7.2 - VISUALIZAR

A visualização do módulo ou arquivo é implementada através do programa "MORE" do MS-DOS e a única informação é que para módulos menores que 25 linhas estes não serão visualizados, pois "cabem" em uma única tela e são apresentados rapidamente, sendo esta uma limitação do programa "MORE".

Para módulos maiores que uma tela, 25 linhas, estes serão visualizados sem nenhum problema e para interromper a visualização em qualquer parte do módulo ou arquivo basta digitar <Ctrl> - C e o ambiente voltará para o sub-menu de ARQUIVOS.

7.7.3 - IMPRIMIR

O módulo ou arquivo será impresso sem nenhum caracter de controle ou mudança de página, respeitando a formatação (tabulação) interna do arquivo.

O programador deve certificar-se que a impressora está devidamente conecta e confirmar a impressão do arquivo quando solicitado. A largura padrão dos módulos e arquivos é de 80 colunas, mas pode ocorrer que linhas editadas por outros editores sejam maiores que este valor.

7.7.4 - CANCELAR

Esta opcão é semelhante as existentes nos editores do AIDS_TME acrescida da opção de cancelamento de arquivos já montados.

Uma vez cancelado o arquivo este será eliminado da lista do AIDS_TME e do diretório ativo. Por segurança o arquivo de rascunho ("backup") não é cancelado do diretório ativo.

Para a utilizacão de arquivos de rascunho, estes deverão ser renomeados para um outro nome qualquer e em seguida gerar um arquivo vazio, via AIDS_TME, e copiar o arquivo rascunho renomeado no novo arquivo vazio gerado.

Este procedimento se faz necessário pelo fato de que o AIDS_TME só manipula arquivos constantes na sua lista interna de módulos e arquivos. Quando do cancelamento o nome do arquivo é eliminado desta lista.

7.8 - CONFIGURAÇAO (20)

O AIDS_TME é um ambiente próprio para a geracão de programas que utilizam basicamente a comunicação com instrumentação via interface IEEE-488/GPIB. Todas as ferramentas necessárias para configurar o controlador (microcomputador) e os instrumentos e ainda executar um rápido teste de comunicação para verificação se todos os parâmetros da configuração estão corretos, estão disponíveis nesta facilidade.

Outra ferramenta que deverá ser muito utilizada pelo programador é a identificação de instrumentos acessados pelo programa de teste e/ou ensaio.

7.8.1 - PROGRAMA DE CONFIGURAÇAO DOS INSTRUMENTOS

O IBCONF, como é conhecido, é o programa fornecido pelo fabricante da placa de interface STD-8410 para a configuração dos instrumentos no barramento IEEE-488/GPIB.

Não é objeto deste trabalho descrever o funcionamento e operação do IBCONF, pois todos os dados e procedimentos são descritos no manual do fabricante que acompanha a placa. Contudo são necessários alguns esclarecimentos e "dicas" para o programador.

Inicialmente o programador deve utilizar o IBCONF para configurar a interface que está conectada ao barramento do microcomputador. O endereço GPIB desta placa sempre será ZERO (0), pois o microcomputador será o controlador principal do barramento. O nome lógico para a interface deverá ser "GPIB0", onde o caracter "0" indica que é a primeira placa controladora do sistema.

No caso de uma segunda placa controladora, esta obrigatoriamente, terá o nome lógico "GPIB1", mas com o mesmo endereço GPIB (0).

De acordo com a norma IEEE-488, será possível e permitido a conecxão de 16 instrumentos por placa controladora do tipo STD-8410. O endereço GPIB e o nome lógico de cada instrumento dependerá basicamente do usuário e da metodologia de automação adotada.

Para cada instrumento o usuário deverá dar um nome, endereço e os parâmetros do instrumento como: "time-out", "EOI", "EOS", etc. . O programador deve acionar o programa IBCONF diretamente do AIDS_TME, executar todos os passos do procedimento de configuração solicitados pelo programa, para cada instrumento, e ao final da configuração o microcomputador deverá ser reinicializado ("RESET").

O IBCONF, da NATIONAL INSTRUMENTS (13), utilizado neste desenvolvimento, possuem um menu de ajuda ("help on-line") muito eficiente e completo. O programador pode esclarecer qualquer dúvida sobre os parâmetros, a cada etapa do procedimento de configuração, acionando a tecla de função "F1".

7.8.2 - PROGRAMA DE CONTROLE INTERATIVO PARA INTERFACEAR COM INSTRUMENTOS CONECTADOS AO BARRAMENTO IEEE-488/GPIB

O IBIC ("Interface Bus Interactive Control Programm") é igualmente fornecido pelo fabricante da interface e permite ao programador acessar, de forma interativa, os instrumentos configurados no barramento.

Esta ferramenta é extremamente importante para o programador, pois é através de comandos de escrita e leitura nos instrumentos que se é possível a determinação do formato dos dados enviados pelos instrumentos ao controlador. Cada instrumento possuem um formato e pode variar dentro do mesmo fabricante para modelos diferentes.

Normalmente os instrumentos enviam os dados ao controlador em formato ASCII, com ou sem caracteres de controle como: Fim de Linha - LF ("line feed") e Retorno do Carro - CR ("carrier return"). Esses dados são recebidos como cadeias de caracteres ("strings") e para sua manipulação o programador deve converte-los em valores numéricos.

O IBIC permite que o programador verifique o formato dos dados recebidos, quantidade e caracteres de controle, definido então o tipo e complexidade da manipulação de cada dado ("string").

O programador deve acessar o IBIC diretamente do AIDS_TME e para qualquer dúvida de utilização de comandos e de sua sintaxe o IBIC possuem um menu de ajuda ("help on-line") que pode ser acessado a qualquer momento através da tecla "F1".

7.8.3 - PROGRAMA PARA VERIFICAÇAO DE ARQUIVOS

Outra ferramenta disponível na facilidade CONFIGURAÇAO é a opçâo ARQUIVO, que permite ao programador verificar através de um diagrama lógico de interligaçâo, apresentado no vídeo, dos instrumentos acessados por um dado programa de teste.

Todos os instrumentos estâo conectados logicamente à interface GPIBO definido assim um barramento paralelo único. Esta ferramenta verifica dentro do arquivo de teste já montado quais sâo os instrumentos mencionados nos comandos e funç%es do GPIB.

O nome lógico de cada instrumento deverá ter no máximo 8 caracteres e ser mnemônico, ou seja, utilizar o modelo ou funçâo do instrumento para caracteriza-lo. Por exemplo, um frequencimentro, marca HP, modelo 5334A, leva o nome lógico D5334A.

Em caso de erro de sintaxe em comandos e funç%es do GPIB, está ferramenta indicará ao programador qual o instrumento e o nome lógico desse instrumento. A escolha do arquivo para a verificaçâo é a padronizada pelo AIDS_TME.

7.9 - XFIM

Obrigatoriamente o programador deve finalizar as suas operaç%es através da facilidade XFIM, caso contrario vários arquivos temporários e permanentes permanecerâo abertos podendo ocasionar a perda dos dados neles contidos.

Em caso de perda de energia, quando da reinicializaçâo do AIDS_TME, todos os arquivos serâo fechados e as variáveis do ambiente serâo reinicializadas automaticamente.

8 - EXEMPLO DE APLICAÇAO - MEDIDA DE DESVIO DE BASE DE TEMPO

Dentre os diversos programas de testes gerados pelo AIDS_TME durante o período de avaliação em laboratório um dos mais completos e de fácil entendimento foi o desenvolvido para um ensaio de calibração de uma base de tempo, opcional de um contador de frequência de microondas.

A base de tempo a ser calibrada é parte opcional do contador HP 5342A "Microwave Frequency Counter", sua frequência central é de 10 MHz, + 0 dBm. O padrão de frequência utilizado para determinar o desvio da base do instrumento é de um gerador padrão de frequência de rubídio HP 5065A, com precisão de 1*10E-11 partes por mês (Padrão de Tempo Nacional, aferido pelo Observatório Nacional). (1-8)

A medida da frequência é obtida através de um Contador Universal de frequência HP 5334A, com a base de tempo padrão externa de 10 MHz.

8.1 - MEDIDA DE DESVIO DE BASE DE TEMPO

A medida de desvio de base de tempo não é normalizada pelo NIST ("National Institute of Standart Tecnology"), antigo NBS, dos EUA, mas a maioria dos fabricantes recomenda o seguinte procedimento:

I - Executar uma seqüência de medidas de curta duração no primeiro instante da alimentação da base de tempo;

II - Executar uma seqüência de medidas de média duração após a base de tempo atingir o seu valor nominal.

III - Executar uma seqüência de medidas de longa duração para determinar a variação e o tempo de estabilização da base (amortização).

Baseando-se nestas recomendações especificou-se a seguinte seqüência para o ensaio:

a. Executar 50 medidas com intervalo de 1 segundo entre cada medida - tempo total (T1) = 50 segundos;

b. Executar 50 medidas com intervalo de 10 segundos entre cada medida - tempo total (T2) = 500 segundos;

c. Executar 50 medidas com intervalo de 30 segundos entre cada medida - tempo total (T3) = 1.500 segundos;

d. Executar 50 medidas com intervalo de 60 segundos entre cada medida - tempo total (T4) = 3.000 segundos.

Tempo total do ensaio (T1+T2+T3+T4) = 5.050 segundos - (1 horas, 24 minutos e 10 segundos).

Os valores para os intervalos entre medidas dependem basicamente da qualidade e precisão da base de tempo, para bases com maior tempo de estabilização esses valores podem ser ajustados para: 1 segundo, 30 segundos, 60 segundos, 300 segundos. O tempo médio de estabilização, especificado pelos fabricantes, é de aproximadamente 30 minutos.

Os valores do desvio e erro ppm (partes por milhão) são obtidos pelas expressões:

(1) Desvio = Fcentral - Fmedida

(2) Erro ppm = (Desvio/Fcentral) * 1E+4

O ensaio está sub-dividido em 4 módulos, ou seja, "setup", medidas, gráficos e planilhas descritos nos próximos itens. A figura 8.1.1 apresenta o esquema básico das ligações entre os instrumentos e com o controlador (Microcomputador).

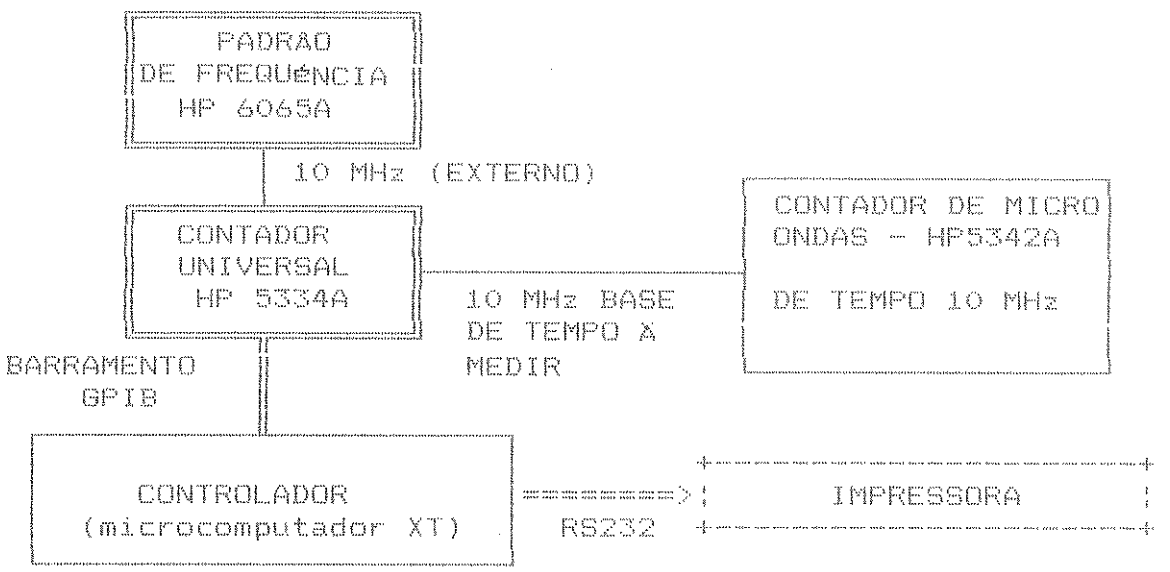


FIGURA 8.1.1 - ESQUEMA BÁSICO

8.2 - EDIÇÃO DO MÓDULO DE "SETUP"

O módulo de "setup" ou inicialização permite ao usuário (operador) colocar todos os instrumentos em uma condição conhecida e desejada. Neste ensaio somente o Contador Universal será inicializado.

O procedimento de inicialização deverá conter instruções e comandos, via GPIB, para o contador, com o objetivo de: coloca-lo em estado remoto de funcionamento, inicializa-lo ("reset" do instrumento), programa-lo para leitura de frequência e executar uma primeira leitura ("trigger" no instrumento).

Já com o AIDS_TME inicializado, o programador deve ativar o editor de "SETUP", através da facilidade EDITORES -> SETUP, figura 8.2.1.

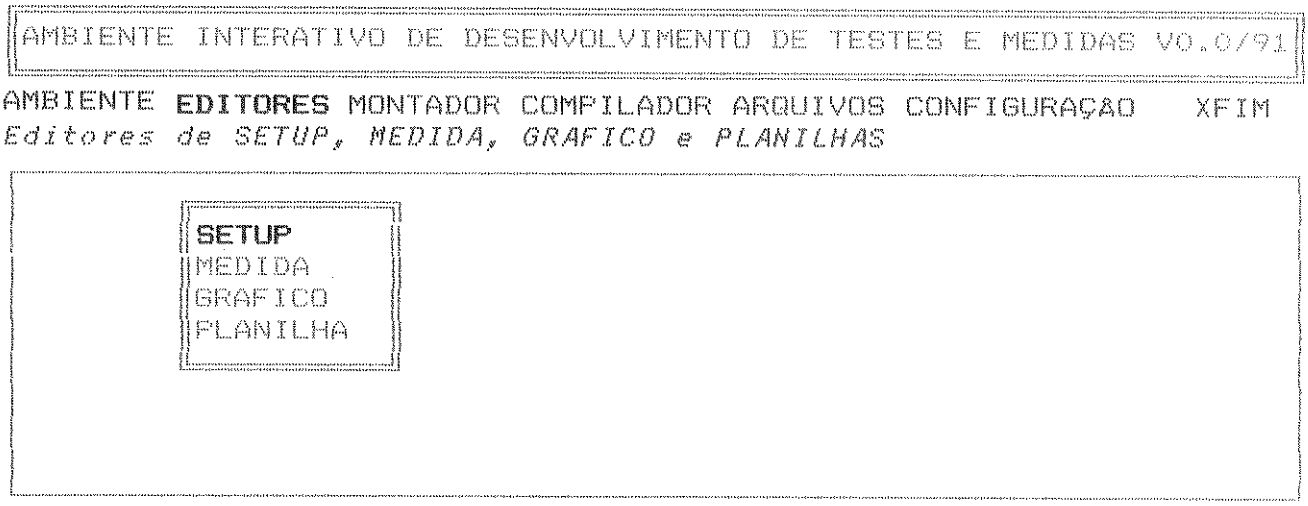


FIGURA 8.2.1 - MENU DOS EDITORES

O procedimento de edição é gerenciado pelo ambiente e é mostrado nas figuras 8.2.2 à 8.2.6, onde para cada linha de comando o programador deverá selecionar a opção, completar a linha de comando de acordo com a sintaxe e confirmar a sua edição (inclusão no arquivo).

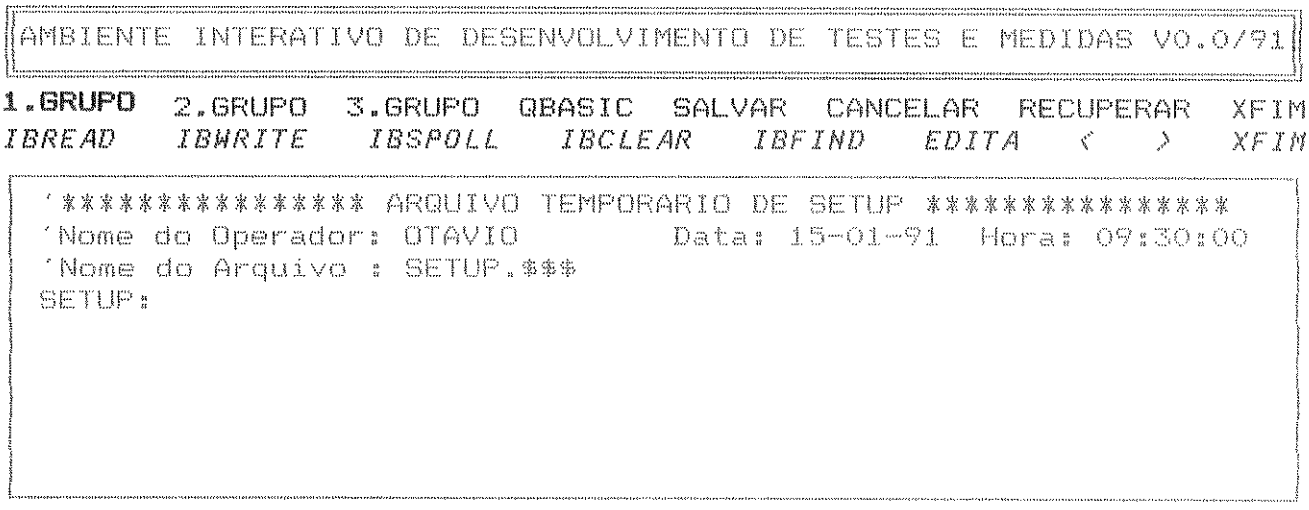


FIGURA 8.2.2 - MENU DOS EDITORES - ACESSO AO 1. GRUPO

Seleção e inclusão do comando "clear" no arquivo de "SETUP".

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
IBREAD   IBWRITE   IBSPOLL   IBCLEAR   IBFIND   EDITA   <   >   XFIM
Executa inicialização em um instrumento
 '***************** ARQUIVO TEMPORARIO DE SETUP ****************
 'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
 'Nome do Arquivo : SETUP.$$$
 SETUP:
      BDNAME$ = "D5334A"
      CALL IBFIND (BDNAME$,D5334A%)
```

FIGURA 8.2.3 - SINTAXE E EDIÇAO DE COMANDOS

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
Sintaxe: CALL IBCLR(DEVICE%)
CALL IBCLR(D5334A%)
 '***************** ARQUIVO TEMPORARIO DE SETUP ****************
 'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
 'Nome do Arquivo : SETUP.$$$
 SETUP:
      BDNAME$ = "D5334A"
      CALL IBFIND (BDNAME$,D5334A%)
```

FIGURA 8.2.4 - SINTAXE DO COMANDO "CLEAR"

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
Ultimo comando preparado:
    CALL IBCLR(D5334A%)
 '***************** ARQUIVO TEMPORARIO DE SETUP ****************
 'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
 'Nome do Arquivo : SETUP.$$$
 SETUP:
      BDNAME$ = "D5334A"
      CALL IBFIND (BDNAME$,D5334A%)
        Confirme a inclusão da linha no arquivo (S/N): S
```

FIGURA 8.2.5 - ACEITAÇAO DO COMANDO "CLEAR" - EDITA

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
IBREAD   IBWRITE   IBSPOLL   IBCLEAR   IBFIND   EDITA   <   >   XFIM
Executa leitura em um instrumento

 '***************** ARQUIVO TEMPORARIO DE SETUP ****************
 'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
 'Nome do Arquivo : SETUP.$$$
 SETUP:
      BDNAME$ = "D5334A"
      CALL IBFIND (BDNAME$,D5334A%)
      CALL IBCLR (D5334A%)
```

FIGURA 8.2.6 - FINAL DA EDIÇAO DO COMANDO DE "CLEAR"

Após a edição de todos os comandos no arquivo de "setup" o programador deverá salvar o módulo através do menu SALVAR, inserindo o nome e a descrição do mesmo, de acordo a solicitação do ambiente.

O arquivo final de "SETUP" tem a seguinte estrutura e comandos específicos para o contador. O arquivo deverá obrigatoriamente terminar com o comando de "RETURN", pois o módulo é tratado como sub-rotina pelo programa principal (executável).

Listagem final do módulo de "SETUP":

```
'************* ARQUIVO TEMPORARIO DE SETUP: ****************
'Nome do Operador= OTAVIO   Data: 04-04-1991 Hora: 14:37:58
'Nome do Arquivo = DESVIO.SET
SETUP:
     BDNAME$ = "D5334A"
     CALL IBFIND(BDNAME$,D5334A%)
     CALL IBCLR(D5334A%)
     CALL IBWRT(D5334A%,"INWA1SM1")
     CALL IBTRG(D5334A%)
     RETURN
```

8.3 - EDIÇAO DO MÓDULO DE MEDIDAS

De acordo com a especificação funcional do teste, definida no item 8.1, o módulo de MEDIDA deverá executar a seqüência de leituras e calcular o desvio e o erro em ppm.

Todos os dados serão arquivados em um arquivo, em formato ASCII, com 8 colunas e com números formatados em precisão simples. Serão utilizadas somente 4 colunas, sendo as demais preenchidas com o valor zero (0).

O programa interage com o operador solicitando o início do teste e o valor da frequência central, em Mega Hz, ao final das medidas, para processar os cálculos necessários.

A montagem das linhas de funções e comandos é idêntica a da edição do módulo anterior. Caso o programador possua um

maior conhecimento de programação em QuickBASIC, este poderá utilizar o editor de tela completa do ambiente, após a criação do módulo de medida pelo editor de MEDIDA. A seguir é apresentada a listagem final do módulo de MEDIDA.

Listagem final do módulo de "MEDIDA":

```
'************* ARQUIVO TEMPORARIO DE MEDIDA: ****************
'Nome do Operador= OTAVIO   Data: 04-04-1991   Hora: 14:38:24
'Nome do Arquivo = DESVIO.MED
MEDIDA:
       ' DIMENSIONAMENTO DA VARIAVEIS
       DIM ValorDaMedida$ (200)
       DIM TempoDaMedida$ (200)
       TIMER ON
          Mes$  = "INICIO DAS MEDIDAS - DIGITE QUALQUER TECLA PARA
CONTINUAR !!"
       CALL Mensagem(Mes$)
       '
       '      1. CICLO DE 50 MEDIDAS DE 1 SEGUNDO
       RD$ = SPACE$(19)
       FOR X=1 TO 50
           GOSUB LOOPMEDE
           SLEEP 1
        NEXT X
       '
       '      2. CICLO DE 50 MEDIDAS DE 10 SEGUNDOS
       FOR X=51 TO 100
           GOSUB LOOPMEDE
           SLEEP 10
       NEXT X
       '
       '      3. CICLO DE 50 MEDIDAS DE 30 SEGUNDOS
       FOR X=101 TO 150
           GOSUB LOOPMEDE
           SLEEP 30
       NEXT X
       '
       '      4. CICLO DE 50 MEDIDAS DE 60 SEGUNDOS
       FOR X=151 TO 200
           GOSUB LOOPMEDE
           SLEEP 60
       NEXT X
       Mes$ = "FINAL DO ENSAIO"
       CALL Mensagem(Mes$)
       '
       '  final de operaçÕes no instrumento
       CALL IBCLR(D5334A%)
       CALL IBONL(D5334A%, 0)
       CALL IBLOC(D5334A%)
       TIMER OFF
       ' SELECAO DA FREQUENCIA CENTRAL
       FreCentral! = 0
        LOCATE 16,10:INPUT  "QUAL A  FREQUENCIA CENTRAL  EM MHz  :
";FreCentral!
       ' ARQUIVO PARA COLOCAR OS DADOS DO ENSAIO
       CALL FIDir (Unit$,Dir$)
       OPEN Unit$+Dir$+"\DESVIO.DAD" FOR OUTPUT AS #3
       FOR x = 1 TO 200
           Valor! = VAL(RIGHT$(ValorDaMedida$(x),LEN(ValorDaMedid
a$(x))-1))
```

```
'    Calculo do desvio
     Desvio! = (FreCentral!*1E6) - Valor!
'    Calculo do erro em ppm
     Derro! = (Desvio!/(FreCentral!*1E6))*1E4
'    Escrita dos valores no arquivo de dados .dad
     WRITE #3,x,Valor!,Desvio!,Derro!,0,0,0,0
  NEXT x
  CLOSE # 3
  ' FINAL DAS MEDIDAS
  RETURN

LOOPMEDE:
     CALL IBRD(D5334A%,RD$)
     CALL IBTRG(D5334A%)
     ValorDaMedida$(x) = RD$
     TempoDaMedida$(x) = TIME$
     LOCATE 10,10:PRINT "NUMERO DA MEDIDA : ";X
     LOCATE 12,10:PRINT "VALOR MEDIDO : ";ValorDaMedida$(x)
     LOCATE 14,10:PRINT "TEMPO ATUAL : "; TempoDaMedida$(x)
     RETURN
```

8.4 - EDIÇÃO DO MÓDULO DE GRÁFICO

O gráfico a ser implementado deve apresentar os valores do erro de desvio, para cada medida executada no tempo, ou seja, um gráfico do tipo t - Y, onde t é o eixo x (medidas) e Y corresponde ao valor do erro.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91
TIPO ARQUIVO FAIXA TITULOS GERAR SALVAR CANCELAR RECUPERAR XFIM
Tipo do gráfico a ser gerado com o resultado das medidas

'***************** ARQUIVO TEMPORARIO DE GRAFICO **************
'Nome do Operador: OTAVIO          Data: 15-01-91  Hora: 09:30:00
'Nome do Arquivo : GRAFO.$$$
GRAFICO:
 ARQUIVO:                              TIPO:
 PRIN.:
 SEC.:
 EIXO X:
 EIXO Y:

 FAIXAS ATIVAS:
 INICIO: 0%                             FINAL: 100%
```

FIGURA 8.4.1 - MENU DOS EDITORES - GRÁFICOS

Através do editor de GRÁFICO, ativa-se a tela de inserção dos dados para a montagem do gráfico, figura 8.4.1, descreve a seqüência de inserção dos dados, de acordo com o menu do editor de gráfico.

TIPO:

Seleciona-se o tipo do gráfico, X - Y ou t - Y, para o exemplo seleciona-se a opção t - Y, tipo número 2.

ARQUIVO:

Nome do arquivo de dados definido no módulo de medida. No exemplo o nome escolhido foi DESVIO.DAD. A extensão .DAD é obrigatória.

FAIXA:

Através do menu FAIXA, deve-se escolher as faixas de dados que irão compor o grafico. No gráfico tipo t - Y a primeira faixa define o eixo X, no exemplo FAIXA A, e a segunda o eixo Y, FAIXA C, valor do desvio.

Neste menu também são escolhidos os limites de visualização dos dados, INICIO em 0%, desde a primeira medida, FINAL em 100%, até a ultima medida.

TITULOS:

Neste menu são inseridos os títulos do grafico e dos eixos X e Y.

```
PRINCIPAL = "MEDIDA DE DESVIO DE BASE - 10 MHz"
SECUNDARIO = "PADRAO HP 5334A"
EIXO X = "NUMERO DE MEDIDAS (200)"
EIXO Y = "DESVIO (Hz)"
```

SALVAR:

Salvar em disco o módulo de dados do gráfico. Definir o nome e descrição do módulo. A extensão do arquivo é do tipo .GRM e sua formatação é listada abaixo.

```
'********************* MODULO DE GRAFICO ********************
'Nome do Operador= OTAVIO  Data: 04-11-1991   Hora: 11:14:49
'Nome do Arquivo = DESVIO.GRM
GRAFICO:
NomeGrafo$ = "DESVIO"
TipoGrafo =  2
TiTlePri$ = "MEDIDA DE DESVIO DE BASE - 10 MHz"
TiTleSec$ = "PADRAO HP 5334A"
TiTleX$ = "NUMERO DE MEDIDAS (200)"
TiTleY$ = "DESVIO (Hz)"
StartFaixa =  0
StopFaixa =  100
Faixas$ = "AC"
```

GERAR:

Após a edição do módulo dos dados é necessário a geração do arquivo que irá realizar as operaçṏes para a apresentação do gráfico.

Através da escolha da opção GERAR, escolhe-se o módulo de dados para a geração do arquivo, dentre os módulos já existentes na lista dos módulos. A extensão do arquivo gerado é .GRA e este poderá ser editado pelo editor de página completa do ambiente.

Módulos de dados já editados poderão ser alterados utilizando-se da opção RECUPERAR do editor de gráfico, esta facilidade é estendivel a todos os outros editores.

8.5 - EDIÇAO DO MÓDULO DE PLANILHA

A planilha a ser implementada deve apresentar o número da medida, o valor lido, o valore do desvio e o erro em ppm, para cada medida executada no tempo, ou seja, uma planilha com 4 colunas com todas as medidas (200).

Através do editor de PLANILHA, ativa-se a tela de inserção dos dados para a montagem da planilha, figura 8.5.1, a seguir descreve-se a sequência de inserção dos dados, de acordo com o menu do editor da planilha.

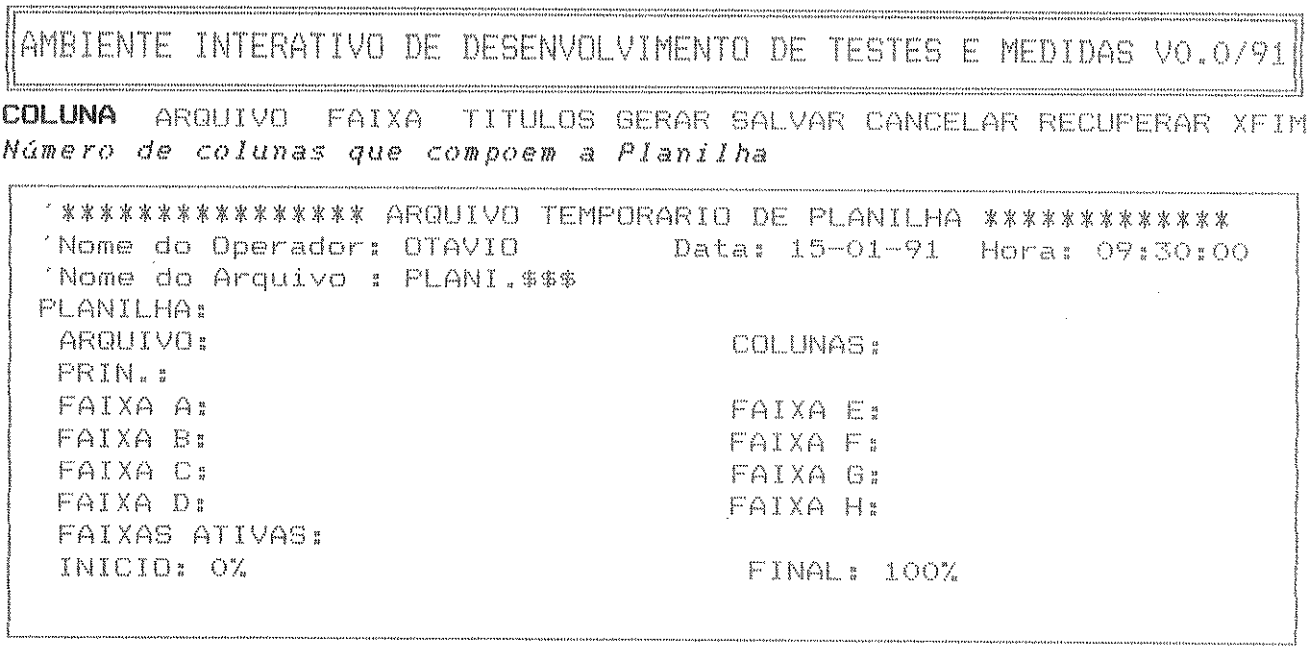


FIGURA 8.5.1 - MENU DOS EDITORES - PLANILHAS

COLUNAS:

Seleciona-se o número de colunas para geração da planilha, para o exemplo seleciona-se a opção de 4 colunas.

ARQUIVO:

Nome do arquivo de dados definido no módulo de medida. No exemplo o nome escolhido foi DESVIO.DAD. A extensão .DAD é obrigatória.

FAIXA:

Através do menu FAIXA, deve-se escolher as faixas de dados que irão compor a planilha. Para 4 colunas são escolhidas as faixa A, B, C e D, conforme arquivo de dados.

Neste menu também são escolhidos os limites de visualização dos dados, INICIO em 0%, desde a primeira medida, FINAL em 100%, até a ultima medida.

TITULOS:

Neste menu são inseridos os titulos da planilha e das faixas.

```
PRINCIPAL = "TABELA DE VALORES DA MEDIDA DE DESVIO"
FAIXA A = "LEITURA"
FAIXA B = "FREQUENCIA"
FAIXA C = "DESVIO"
FAIXA D = "ERRO PPM"
```

SALVAR:

Salvar em disco o módulo de dados da planilha. Definir o nome e descrição do módulo. A extensão do arquivo é do tipo .PLM e sua formatação é listada abaixo.

```
'********************* MÓDULO DE PLANILHA *********************
'Nome do Operador= OTAVIO   Data: 04-11-1991   Hora: 11:16:23
'Nome do Arquivo = DESVIO.PLM
PLANILHA:
NomePlani$ = "DESVIO"
Colunas =  4
TiTlePri$ = "TABELA DE VALORES DA MEDIDA DE DESVIO"
TitleFA$ = "LEITURA"
TitleFB$ = "FREQUENCIA"
TitleFC$ = "DESVIO"
TitleFD$ = "ERRO PPM"
TitleFE$ = ""
TitleFF$ = ""
TitleFG$ = ""
TitleFH$ = ""
StartFaixa =  0
StopFaixa =  100
Faixas$ = "ABCD"
```

GERAR:

Após a edição do módulo dos dados é necessário a geração do arquivo que irá realizar as operações para a apresentação da planilha.

Através da escolha da opção GERAR, escolhe-se o módulo de dados para a geração do arquivo, dentre os módulos já existentes na lista dos módulos. A extensão do arquivo gerado é .PLA e este poderá ser editado pelo editor de página completa do ambiente.

8.6 - MONTAGEM DO ARQUIVO DE TESTE

Com os módulos de "SETUP", MEDIDAS, GRAFICOS e PLANILHAS já editados e/ou gerados pelos editores, o programador deve "montar" o arquivo de teste por completo. A extensão desse arquivo será .BAS.

Selecionando-se a facilidade "MONTADOR" o ambiente apresentará um sub-menu, figura 8.6.1, onde existe a opção para a montagem do arquivo principal. A seqüência para a montagem não é rigida, podendo ser escolhido qualquer módulo em qualquer seqüência.

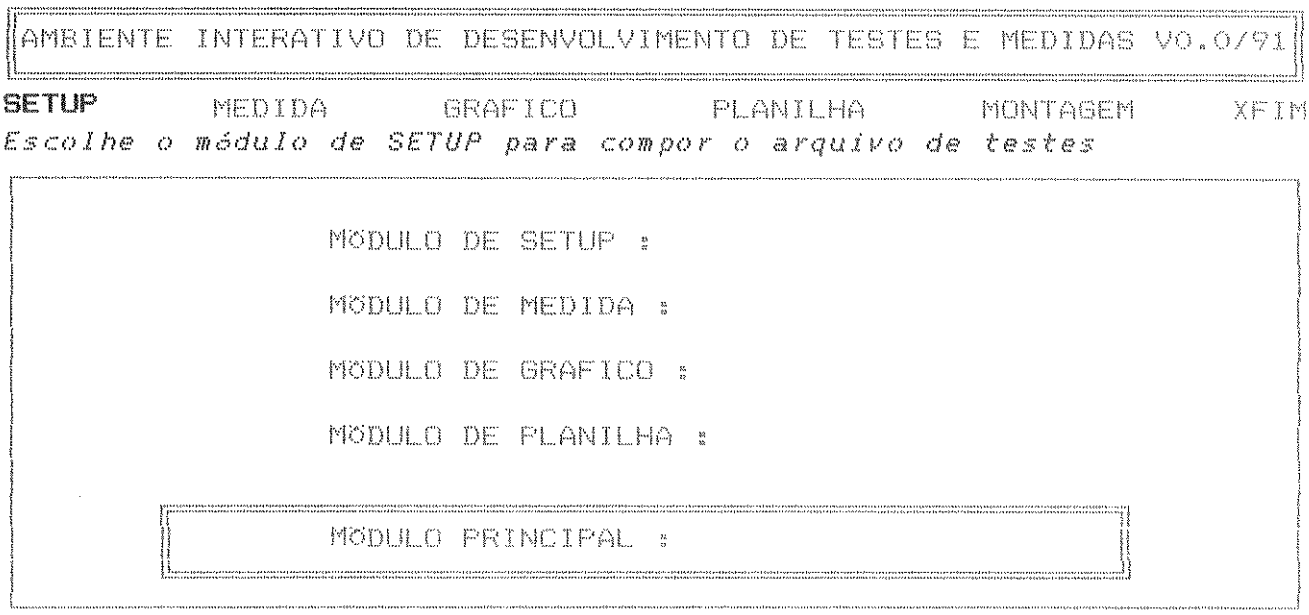


FIGURA 8.6.1 - MENU DO MONTADOR DE ARQUIVOS (.BAS)

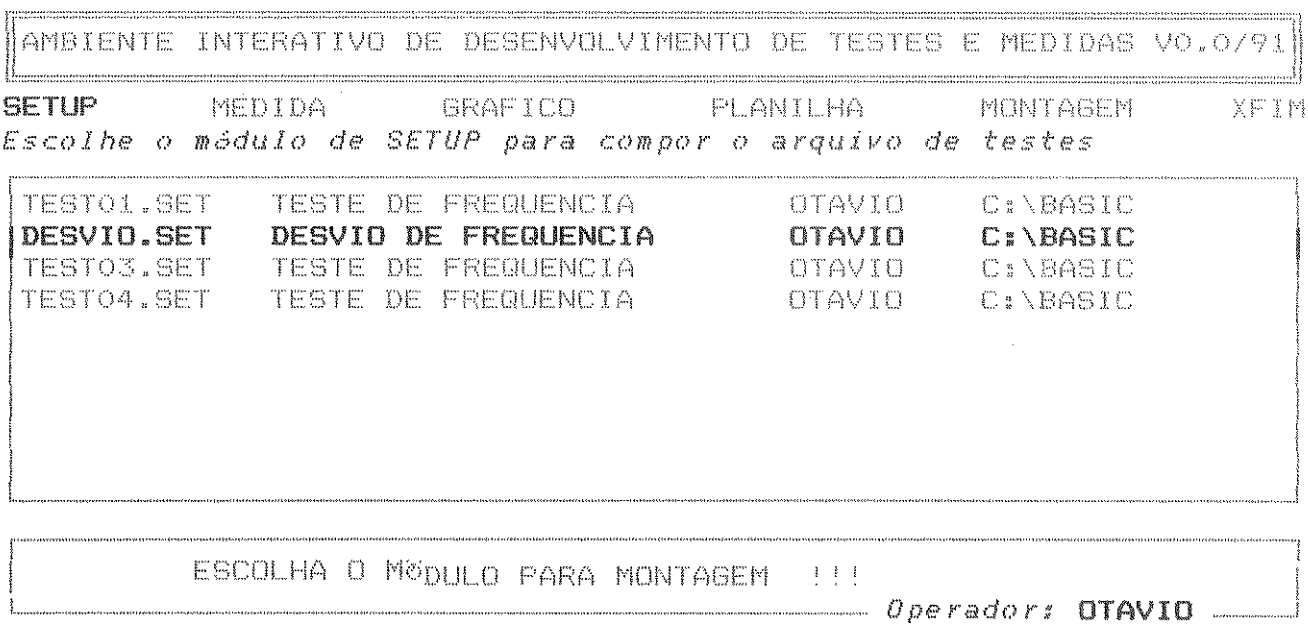


FIGURA 8.6.2 - ESCOLHA DO MÓDULO DE "SETUP"

A figura 8.6.2 apresenta um menu padrão de escolha de módulos, onde o programador escolhe 1 entre vários módulos já editados e/ou gerados. Com todos os módulos já selecionados o programador deve executar a opção "MONTAGEM", inserindo o nome do arquivo a ser montado e sua descrição (identificação), figura 8.6.3.

O arquivo principal pode conter os 4 módulos ou os que o programador desejar, caso não seja selecionado um dos módulos o ambiente inserirá automaticamente um módulo, em substituição, que tem por finalidade enviar uma mensagem ao usuário que aquele módulo em específico não foi implementado.

O arquivo montado pode ser editado pelo editor de página inteira do ambiente e/ou outro qualquer editor, fora do ambiente, que não insira caracteres de controle no arquivo final.

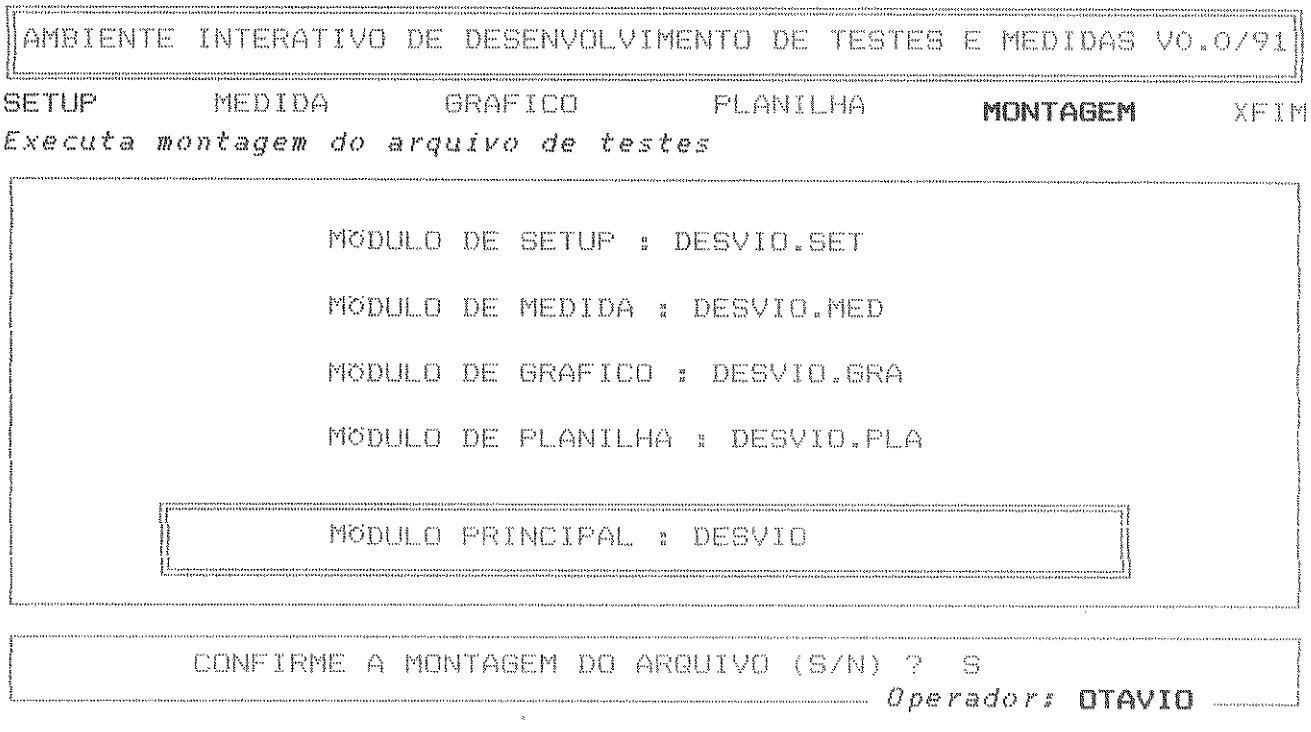


FIGURA 8.6.3 - MONTAGEM DO ARQUIVO PRINCIPAL

8.7 - VERIFICAÇÃO DOS INSTRUMENTOS ACESSADOS

Outra facilidade bastante útil é a verificação dos instrumentos que estão sendo acessados pelo programa principal, no barramento GPIB/IEEE-488. Figura 8.7.1.

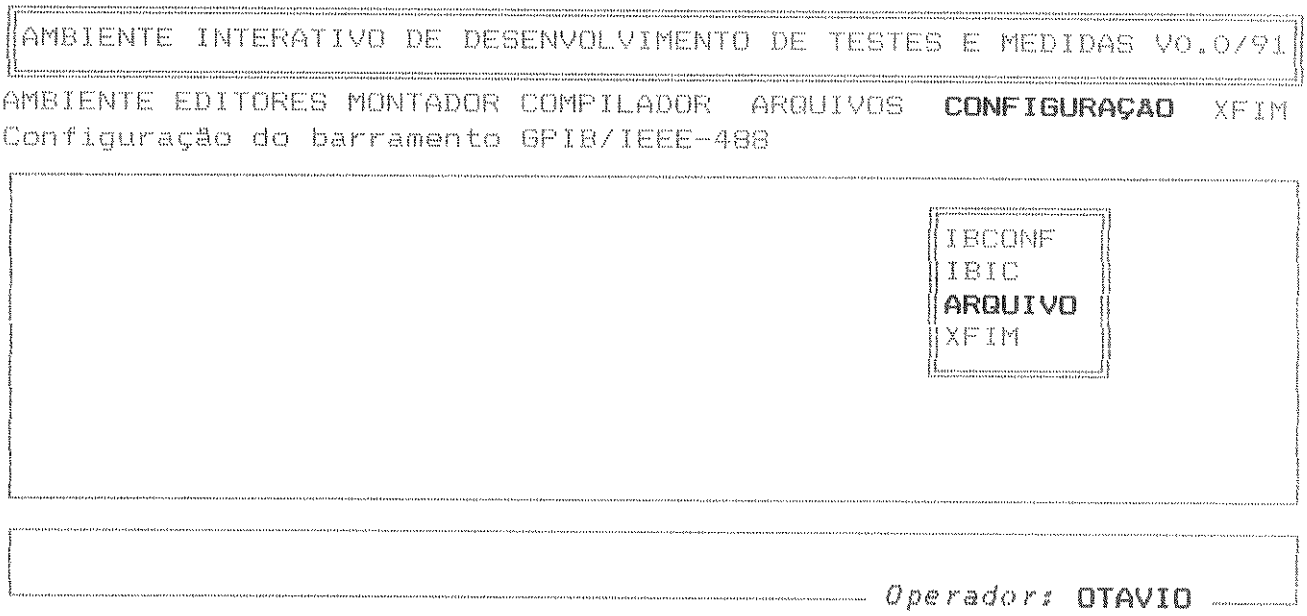


FIGURA 8.7.1 - CONFIGURAÇAO

Utilizando-se de um sub-menu e de uma lista de escolha padrão, seleciona-se o arquivo, de extensão .BAS, e o ambiente apresentará no vídeo, em forma gráfica, os instrumentos conectados ao barramento e acessados pelo programa. Figura 8.7.2.

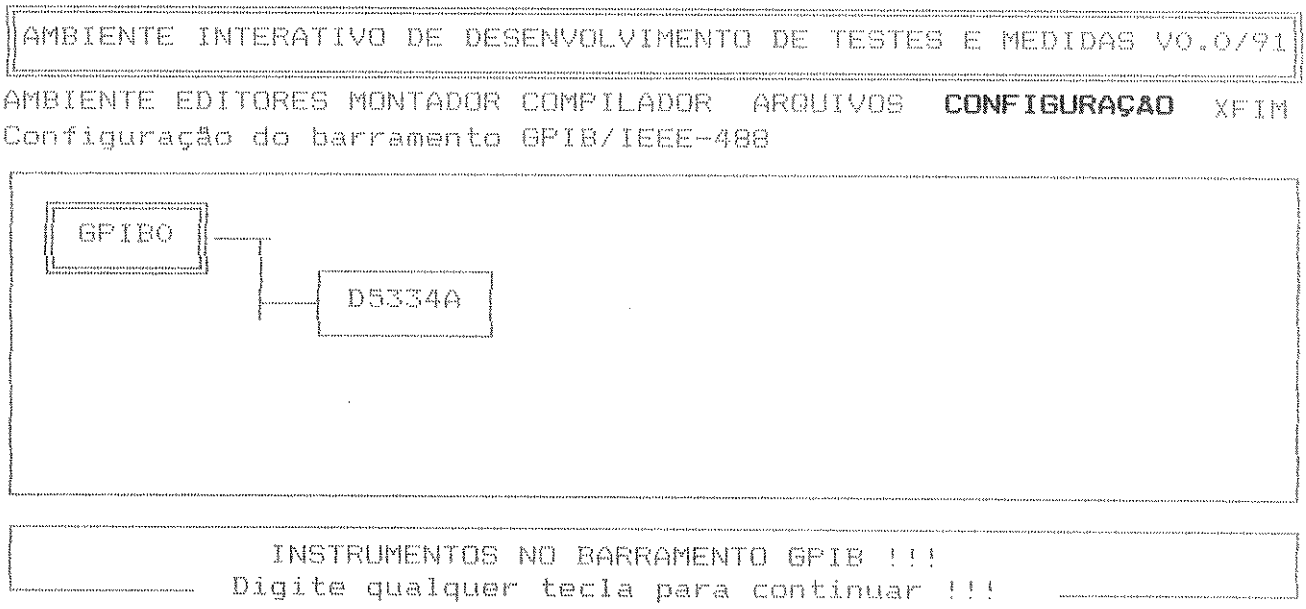


FIGURA 8.7.2 - VERIFICAÇAO DOS INSTRUMENTOS NO BARRAMENTO GPIB

8.8 - "COMPILAÇAO" E "LINK-EDIÇAO"

Após a verificação dos instrumentos acessados pelo programa principal, o programador deve ativar a facilidade "COMPILADOR" e através de um sub-menu e de uma lista padrão de escolha, escolher qual dos arquivos montados (.BAS) será "compilado" e "link-editado". Figura 8.8.1)

O ambiente gerará automaticamente todos os parâmetros para o compilador e o "link" do QuickBASIC. Ao final do processo tem-se o arquivo executável (.EXE) que será entregue ao usuário final.

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

AMBIENTE EDITORES MONTADOR COMPILADOR ARQUIVOS CONFIGURAÇÃO    XFIM
Geração do arquivo Executável de Teste e/ou Ensaio

TESTO1.BAS    TESTE DE FREQUENCIA          OTAVIO    C:\BASIC
DESVIO.BAS    DESVIO DE FREQUENCIA         OTAVIO    C:\BASIC
TESTO3.BAS    TESTE DE FREQUENCIA          OTAVIO    C:\BASIC
TESTO4.BAS    TESTE DE FREQUENCIA          OTAVIO    C:\BASIC

          ESCOLHA O ARQUIVO PARA COMPILAR !!!
                                    Operador: OTAVIO
```

FIGURA 8.8.1 - COMPILADOR

8.9 - O ARQUIVO EXECUTAVEL

O usuário deve ativar o programa executável digitando o nome do programa (DESVIO.EXE) e a unidade ou diretório e/ou sub-diretório deve conter os arquivos de comunicação com o barramento GPIB/IEEE-488 ("driver's").

```
AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTES E MEDIDAS V0.0/91

SETUP          MEDIDA          GRAFICO          PLANILHA          XFIM
Executa procedimento de inicialização nos instrumentos

                                    Operador: OTAVIO
```

FIGURA 8.9.1 - MENU PRINCIPAL DO ARQUIVO EXECUTAVEL

A figura 8.9.1 apresenta o menu principal do arquivo executável. As facilidades básicas do programa consiste em 4 módulos : "SETUP", MEDIDA, GRAFICO e PLANILHA.

Estas facilidades permitem ao usuário a execução do ensaio e/ou teste de forma automática, com um alto grau de confiabilidade e repitibilidade.

A opção planilha apresenta um sub-menu com opções de apresentação em tela ou impressora, com a alteração do "passo" de apresentação do arquivo.

8.10 - OS RESULTADOS DO ENSAIO

Como resultado do teste, medida de desvio de frequência de uma base de tempo, o programa executável produziu um gráfico, figura 8.10.1, e uma planilha com os dados referentes a cada ponto de medida, listados a seguir.

Para otimizar a apresentação dos dados em forma tabular, foi alterado o passo de apresentação, ou seja, 1:4, sendo listados apenas 50 pontos do 200 medidos, facilidade do programa de teste.

TABELA DE VALORES DA MEDIDA DE DESVIO

LEITURA	FREQUENCIA	DESVIO	ERRO PPM
1	1E+07	0	0
5	1E+07	0	0
9	1E+07	0	0
13	1E+07	0	0
17	1E+07	0	0
21	1E+07	-1	-.001
25	1E+07	-1	-.001
29	1E+07	-1	-.001
33	1E+07	-1	-.001
37	1E+07	-1	-.001
41	1E+07	-1	-.001
45	1E+07	-1	-.001
49	1E+07	-1	-.001
53	1E+07	-1	-.001
57	1E+07	-1	-.001
61	1E+07	-2	-.002
65	1E+07	-2	-.002
69	1E+07	-1	-.001
73	1E+07	-2	-.002
77	1E+07	-2	-.002
81	1E+07	-2	-.002
85	1E+07	-2	-.002
89	1E+07	-2	-.002
93	1E+07	-2	-.002
97	1E+07	-3	-.003
101	1E+07	-3	-.003
105	1E+07	-3	-.003
109	1E+07	-3	-.003
113	1E+07	-4	-.004
117	1E+07	-4	-.004
121	1E+07	-4	-.004
125	1E+07	-4	-.004

LEITURA	FREQUENCIA	DESVIO	ERRO PPM
129	1.000001E+07	-5	-.005
133	1.000001E+07	-5	-.005
137	1.000001E+07	-5	-.005
141	1.000001E+07	-5	-.005
145	1.000001E+07	-6	-.006
149	1.000001E+07	-6	-.006
153	1.000001E+07	-6	-.006
157	1.000001E+07	-6	-.006
161	1.000001E+07	-7	-.007
165	1.000001E+07	-8	-.008
169	1.000001E+07	-8	-.008
173	1.000001E+07	-8	-.008
177	1.000001E+07	-9	-.009
181	1.000001E+07	-9	-.009
185	1.000001E+07	-9	-.009
189	1.000001E+07	-10	-.01
193	1.000001E+07	-10	-.01
197	1.000001E+07	-10	-.01

Máximo A	Máximo B	Máximo C	Máximo D
200	1.000001E+07	0	0

Mínimo A	Mínimo B	Mínimo C	Mínimo D
1	1E+07	-10	-.01

- CONSIDERAÇÕES SOBRE OS NÚMEROS DA TABELA

O Contador Universal HP 5335A envia ao controlador um conjunto de caracteres ASCII em forma alfanumérica. O formato da mensagem recebida pelo programa , em QuickBASIC, é "F Y.YYYYYYYE+XX". Esta variável alfanumérica deverá ser convertida em um valor numérico, para tal é desprezado o primeiro caracter da cadeia (SUBSTR()), eliminado os espaços em branco (ALLTRIM()) e convertido em número.

A conversão alfanumérica para numérica é executa pela função VAL(), valor numérico de uma expressão alfanumérica, a variável numérica que receberá o valor foi definida como de Precisão Simples (SINGLE).

Uma variável de precisão simples possuem somente 7 caracteres para sua representação. Neste caso o valor convertido sofrerá um processo de "arredondamento" que consiste em:

- Se o último caracter for menor que 5, será mantido o penúltimo caracter;

- Se o último caracter for maior que 5, o penúltimo caracter será acrescido de 1;

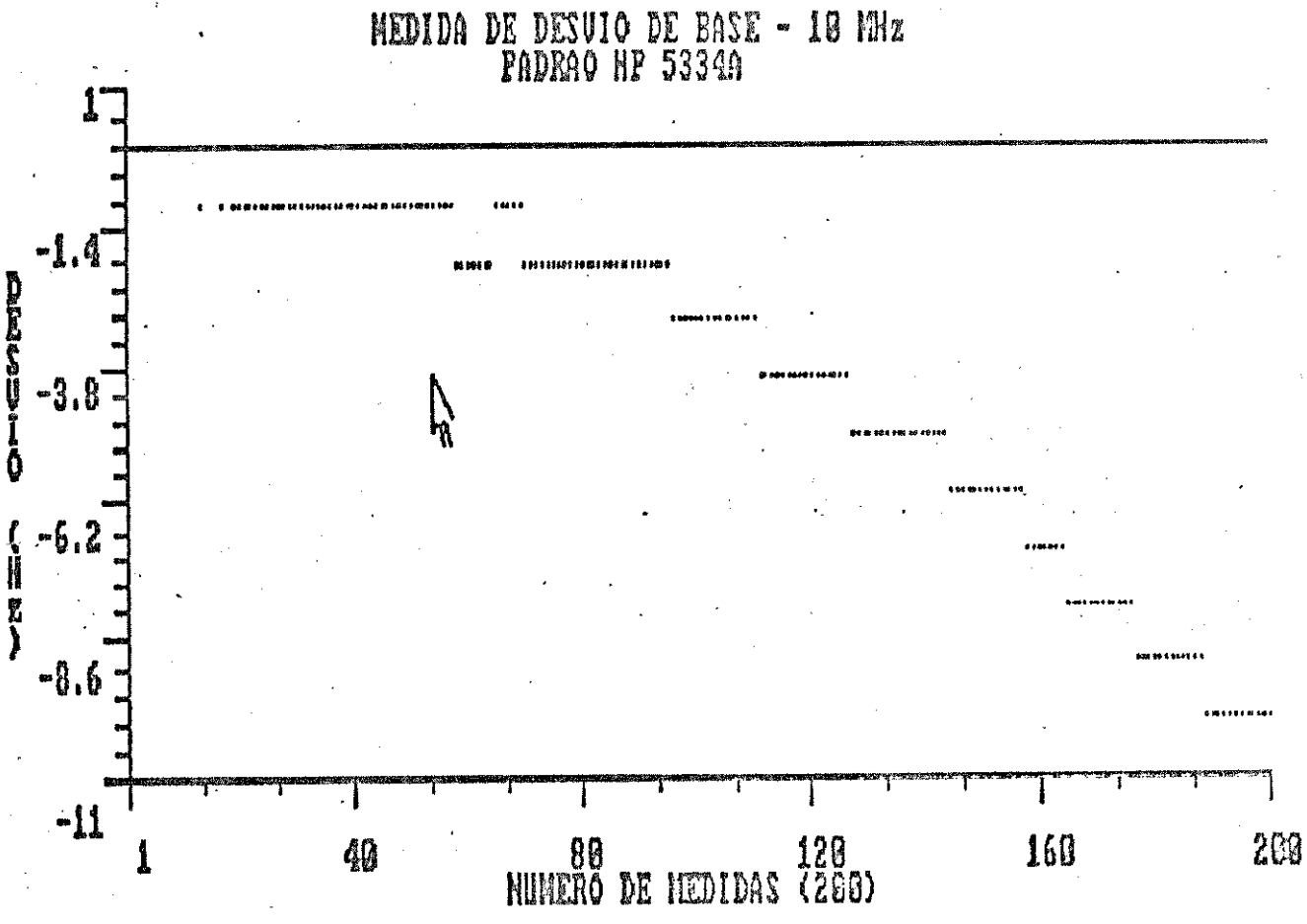
- Se o último caracter for igual a 5 e se o penúltimo caracter for par, este será acrescido de 1, caso contrário, será mantido.

Exemplo:

"1.0000023"	---> 1.000002	menor que 5
"1.0000027"	---> 1.000003	maior que 5
"1.0000025"	---> 1.000003	igual a 5, par
"1.0000035"	---> 1.000003	igual a 5, impar

Este procedimento de "arredondamento" é valido operações de Divisão, Multiplicação, Soma, Subtração e Potênciação. Somente a apresentação é feita com 7 caracteres, sendo que os valores totais é que são utilizados na operação. Veja na tabela a coluna de desvio e erro em ppm a propagação do erro de "arredondamento".

- GRÄFICO PRODUZIDO PELO PROGRAMA DE TESTE:



8.11 - CONSIDERAÇÕES FINAIS

O objetivo maior do AIDS_TME é a otimização e automação do processo de criação e produção de arquivos de teste e/ou ensaio, com uma alta produtividade e qualidade.

O exemplo de teste apresentado permitiu a validação dos procedimentos e facilidades do ambiente. O tempo necessário para a edição, montagem e teste do protótipo do arquivo executável foi reduzido em mais de 50%.

Para a montagem de módulos já testados em novos programas a redução do tempo foi ainda mais significativa, em máquinas do tipo PC XT (8 MHz), o tempo de geração ficou entre 5 a 6 minutos, dependendo da complexidade do módulo de medida (número de linhas). Já para máquinas do tipo PC AT 286 (16 MHz) este tempo ficou em torno de 3 a 3,5 minutos.

O princípio de modularidade e reusabilidade de software, neste ambiente, foi altamente proveitoso permitindo ao programador implementar rapidamente protótipos executáveis de forma e conteúdo variáveis.

9 - CONSIDERAÇÕES FINAIS

Após quase 1 (um) ano desenvolvimento (89-90) e 06 meses de teste de laboratório (90-91) o AIDS_TME está pronto para sua divulgação e dissiminação junto aos usuários e programadores da área de testes e medidas elétricas.

A idéia inicial de aplicação do AIDS_TME era basicamente na área de microeletrônica, ou seja, na geração de programas para extração de parâmetros CC e CA de estruturas e dispositivos semicondutores, utilizando-se de instrumentação com interface IEEE-488/GPIB, para realimentação de parâmetros de entrada em programas simuladores do tipo "SPICE".

Como a mudança de área de trabalho do autor, uma nova perspectiva sobre à aplicação do AIDS_TME foi vislumbrada, agora com maior abrangência e aplicação imediata. Utilizando-se de instrumentação específica para a extração de parâmetros de semicondutores, HP 4145A, foi validada a aplicação do AIDS_TME na área de microeletrônica e seu maior retorno, de imediato, ficou para as áreas de teste e/ou ensaios eletroeletrônicos.

Foram gerados vários programas de testes como: medidas de impedância, levantamento de curva CxV de capacitores, medidas de tempo, medidas de desvios de base de tempo, ensaios de varistores, etc , utilizando-se do AIDS_TME, com um resultado bastante satisfatório em termos de produtividade e qualidade do produto final (software). Estes programas e os módulos criados pelo AIDS_TME estão disponíveis no Laboratório Central - LAC, da COPEL, Curitiba, Paraná, fone (041) 366.2020 ramais 49 e 50.

A seguir são feitas algumas considerações sobre o hardware, o software, o AIDS_TME, os programas gerados, novas interfaces e a evolução exigida pelos usuários.

9.1 - O HARDWARE (20)

A principal consideração a ser feita sobre o hardware necessário para o AIDS_TME esta na utilização da interface de comunicação IEEE-488/GPIB, da STD, modelo STD-8410. Atualmente todos os microcomputadores comprados pela COPEL, para utilização em áreas de manutenção ou de testes, são especificados com a interface GPIB.

Esta limitação é natural, pois sem a interface é impossível executar qualquer tipo de comunicação com a instrumentação existente, que possuem a interface, não como opção, mais sim como parte do próprio equipamento ("built-in").

A máquina ideal para o AIDS_TME seria um PC/AT, devido a massa de dados gerada pelos teste e/ou ensaios, mas o desempenho em um microcomputador da linha ITAUTEC, PC/XT, modelo IS30, foi bastante satisfatório. Como ponto importante, ressalta-se a necessidade de um disco rígido, de pelo menos 20M bytes, para a suportar todas as bibliotecas, módulos e programas gerados pelo AIDS_TME.

9.2 - O SOFTWARE (1s)

O AIDS_TME nos 6 meses em que foi testado em laboratório teve 85% de suas rotinas testadas, sendo que o 15% restantes deveram ser atingidos durante os testes de campo e em teste especiais de laboratório.

Não forão testadas situaçºes especiais como: duplo controle de barramento, acesso a instrumentação com endereço secundário (quase inexistente), manipulação de dados em tempo real (taxa de transmisão baixa), etc.

Outra opção ainda não implementada é a de ajuda das funçºes do QuickBASIC, que somente o arquivo com as mensagens tem um tamanho de 330K bytes. Estão em andamento um série de atividades que visam complementar e/ou modificar telas, menus e linhas de comando do AIDS_TME, por estágiarios (02) do quinto (5) ano de Engenharia Elétrica, Departamento de Eletrônica da Universidade Federal do Paraná - UFPR, dentro do LAC.

Esta prevista uma nova versão do AIDS_TME para o final de 91, após os teste de campo e da distribuição oficial aos programadores das diversas áreas da COPEL, ainda no 2. semestre de 91 os programadores serão treinados sobre a operação e aplicação do ambiente.

Ficou evidente, quando dos teste em laboratório, da necessidade de aumentar a utilização de menus que operem através de "MOUSE" e de um menu de ajuda "help on-line" para outras funçºes do IEEE-488 que não estão implementadas nos grupos 1, 2 e 3 dos módulos de "SETUP" e MEDIDAS.

9.3 - O AIDS_TME

O AIDS_TME faz parte de uma filosofia de automação de testes e/ou ensaios que está sendo implantada na COPEL desde 1990. A primeira etapa foi a utilização de uma Planilha Eletrônica ("Lotus 1-2-3") com um "Driver" de comunicação com instrumentos, para a geração de programas, utilizando-se das "Macros" do próprio "Lotus 1-2-3".

A segunda etapa consiste na utilização de uma linguagem de programação ("QuickBASIC") e de ferramentas automáticas para geração dos programas de testes. O AIDS_TME será utilizado como meio para se executar esta etapa. A criação da figura do "programador" na área, visa a especialização de elementos das diversas áreas, permitindo uma maior penetração e dissiminação dessa filosofia dentro da própria área.

A terceira etapa é baseada na integração de ferramentas de desenvolvimento e bancos de dados permitindo a ligação do usuário final a facilidades gerenciamento, bases de dados mais consistentes e conecxão lógica a "Main Frame". O objetivo final é integrar o dados gerados durante um simples teste à ferramentas de alta complexidade e desempenho, com extrema rapidez e confiabilidade.

9.4 - OS PROGRAMAS FINAIS (EXECUTÁVEIS)

Os programas executáveis gerados pelo AIDS_TME demonstraram serem pequenos (80K a 100K bytes) e bem rápidos durante a sua execução, ficando limitados aos instrumentos conectados ao barramento IEEE-488/GPIB.

A interface Homem X Máquina do programa executável deverá ser alterada para as próximas versões, foi constatado a necessidade de outros tipos de menus, sendo assim, quando da montagem do programa final (.BAS), no módulo de "MONTAGEM", o usuário poderá escolher entre vários tipos de menus, de acordo com a sua necessidade e tipo de montagem.

Outras facilidades estão sendo desenvolvidas pelo grupo de estagiários com o objetivo de colocar a disposição do programador todas as sub-rotinas e funções desenvolvidas durante o desenvolvimento do AIDS_TME, biblioteca "TOOLBOX.LIB".

9.5 - NOVAS INTERFACES

A partir do primeiro semestre de 92, estarão disponíveis dentro dos editores de "SETUP" e "MEDIDAS" menus para a geração de comandos para controle e comunicação com a interface digitalizadora, Analógica/Digital, também fabrica pela empresa STD, modelo 5012, em linguagem QuickBASIC.

As rotinas de acesso a interface são fornecidas pelo fabricante e o seu acionamento é feito através de chamadas do tipo "CALL". Com este novo conjunto de menus o AIDS_TME terá seu campo de aplicação ampliado ainda mais.

Estam previstas ainda as interfaces lógicas com outros programas e/ou utilitários como banco de dados ("DBASE, CLIPPER"), editores de textos ("WORD, PCWRITE, WORDSTAR"), planilhas eletrônicas ("Lotus 1-2-3, SUPERCAL"), etc para o segundo semestre de 92.

9.6 - A EVOLUÇAO DO AIDS_TME

Quando do início do trabalho não imaginou-se a repercusão e a aceitação recebida pelo AIDS_TME dentro das áreas de aplicação. Hoje é cobrada uma evolução pelos programadores e futuros usuários, que imaginam ter seus problemas de testes e/ou ensaios parcialmente resolvidos por esta ferramenta.

Estão previstos cursos e palestras para a divulgação do AIDS_TME dentro de empresas da área de energia elétrica e entidades de pesquisas que utilizam equipamentos para caracterização de semicondutores.

O AIDS_TME é um produto de propriedade do autor e da COPEL, devendo portanto ser registrado junto aos orgãos competentes para sua comercialização.

O estágio atual de desenvolvimento do AIDS_TME, a nível acadêmico, permite validar sua tecnologia e resultados, sua evolução está garantida não só pela evolução do trabalho do autor, mas sim, pela necessidade das áreas e de sua intensa utilização pelos programadores especializados.

9.7 - O OBJETIVO X RESULTADOS

Ao final deste trabalho cabe ressaltar dois resultados bem diferentes. O primeiro referente ao objetivo principal que era a de um trabalho de dissertação a nível de mestrado, resultado este representado pelo conteúdo deste volume.

O segundo, bem mais gratificante, o da implantação e dissiminação de uma nova filosofia de automação de testes e/ou ensaios, a nível nacional, baseada em novas ferramentas de desenvolvimento de software, permitindo uma economia de milhares de dolares/ano em mão de obra especializada para as Companhias e Instituições de Pesquisa que venham a se utilizar dessa nova filosofia.

10 - REFERÊNCIAS BIBLIOGRAFICAS

1. HONDA, THE IMPEDANCE MEASUREMENT HANDBOOK, A Guide to Measurement Technology and Techniques, Yokogawa-Hewlett-Packard LTD., USA, 1989, 70 pag.

2. Application Note 315, DC PARAMETRIC ANALYSYS OF SEMICONDUTORES DEVICES, Hewlett-Packard LTD., USA, 1982, 20 pag.

3. Application Note 322, ANALYSIS OF SEMICONDUCTOR CAPACITANCE CHARACTERISTICS, Hewlett-Packard LTD., USA, 1983, 20 pag.

4. KEITHLEY INSTRUMENTS, CV-SEMICONDUTOR MEASUREMENTS, Keithley Instruments Division, USA, 1987, 15 pag.

5. Application Note 369-5, MULTI-FREQUENCY C-V MEASUREMENT OF SEMICONDUCTORS, Hewlett-Packard LTD., USA, 1987, 8 pag.

6. Application Note 339-5, MULTI-FREQUENCY C-V MEASUREMENTS AND DOPING PROFILE ANALYSIS OF SEMICONDUCTORS, Hewlett-Packard LTD., USA, 1986, 10 pag.

7. Sze-Hon Kwan, Kit M. Chan, H. A. Richard Wegener, AN AUTOMATED TEST SYSTEM FOR MNOS TRANSISTOR CHARACTERIZATION, IEEE Transactions on Instrumentation and Measurement, Vol. IM-32, n. 4, DEC 83.

8. Il-Song Han, A NEW MEASUREMENT TECHNIQUE FOR MOS CAPACITORS, IEEE Transactions on Instrumentation and Measurement, Vol. IM-34, n. 4, DEC 85.

9. Wasim Ahmad, A NEW SIMPLE TECHNIQUE FOR CAPACITANCE MEASUREMENT, IEEE Transactions on Instrumentation and Measurement, Vol. IM-35, n. 4, DEC 86.

10. Umesh Sharma, Richard V. H. Booth,A TIME-DEPENDENT PARAMETER ACQUISITION SYSTEM FOR THE CHARACTERIZATION OF MOS TRANSISTORS AND SONOS MEMORY DEVICES, IEEE Transactions on Instrumentation and Measurement, Vol. IM-38, n. 1, DEC 89.

11. IEEE-488 (1978). STANDARD DIGITAL INTERFACE FOR PROGRAMMABLE INSTRUMENTATION. 42 p.

12. HEWLET-PACKAD. TUTORIAL DESCRIPTION OF THE HEWLETT-PACKARD INTERFACE BUS, designed for HP-IB systems. Santa Clara, CA, 1980. 63 p.

13. NATIONAL INSTRUMENTS. IEEE-488 INSTRUMENTATION INTERFACE MANUAL. Austin, Texas, 1984, USA. 152 p.

14. FRAGOMENI, Ana H.. DICIONARIO ENCICLOPÉDIO DE INFORMATICA. Ed. Novel, Campos, SP, 1986. 332 p.

15. LEVENTHAL, Lance A.. Z80 ASSEMBLY LANGUAGE PROGRAMMING. Osborne/McGraw-Hill, Berkeley, California, 1979.

16. TEXAS INSTRUMENTS INC.. THE INTERFACE CIRCUITS DATA BOOK FOR DESIGN ENGINEERS. 1981

17. TEXAS INSTRUMENTS INC.. TMS9914A GENERAL PURPOSE INTERFACE BUS (GPIB) CONTROLLER - Data Manual.

18. TEXAS INSTRUMENTS INC.. THE TTL DATA BOOK FOR DESIGN ENGINEERS. 1981

19. DATA NEWS, volume 11, número 367, 15/Jun/87. página 26.

20. MANUAL DO USUARIO, CARTAO CONTROLADOR GPIB, STD, Brasília, Junho/86, 80 p.

21. Shlaer, Sally; Melllor, Sthephen J., ANÁLISE DE SISTEMAS ORIENTADA PARA OBJETO. McGraw Hill - São Paulo, 1990, 178 p.

22. Case, Albert f. INFORMATION SYSTEMS DEVELOPMENT: PRINCIPLES OF COMPUTER AIDED SOFTWARE ENGINEERING, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1986, 240 p.

23. Pressman, Roger S., SOFTWARE ENGINEERING, A PRACTITIONER'S APPROACH, McGraw Hill-USA, 1982, 354 p.

24. Barnes, D. & Brown, P., SOFTWARE ENGINEERING 86, Peter Peregrinus Ltd., London, United Kingdom, 1986, 434 p.

25. Freman, Peter. TUTORIAL ON SOFTWARE DESIGN TECHNIQUES, IEEE Computer Society, New York, USA, 1980, 456 p.

26. Allen, Belton E. TUTORIAL MICROCOMPUTER SYSTEM SOFTWARE AND LANGUAGE, IEEE Computer Society, New York, USA, 1980, 232 p.

27. Ferreira, Aurélio B. de Holanda. NOVO DICIONÁRIO DA LINGUA PORTUGUESA, editora Nova Fronteira, Rio de Janeiro, 1986, 1500p.

28. Belserene, Rita. MASTER QUICKBASIC, Sybex Inc., California, USA, 1990, 447

29. Rugg, Tom; Feldman, Phil, QUICKBASIC: PROGRAMMER'S TOOLKIT, QUE Corporation, USA, 1989, 570p.

30. Aitken, Peter G., QUICKBASIC ADVANCED TECHNQUES, QUE Corporation, USA, 1989, 470p.

AUTOR:

JOSÉ OTÁVIO SIMÕES, Engenheiro Eletrônico pelo Instituto Nacional de Telecomunicações - INATEL - MG - 1979.

Atualmente pertence ao quadro de funcionários da Companhia Paranaense de Energia - COPEL - lotado no Laboratório Central - LAC/DPEO/VELG - Divisão de Ensaios e Desenvolvimento.

ANEXO - 01

PROGRAMA PRINCIPAL

AIDS_TME - Versão 91.01

Neste anexo são apresentadas as sub-rotinas que compl em o programa principal. As funções e sub-rotinas não listadas aqui fazem parte da biblioteca de funções especiais, desenvolvidas exclusivamente para o programa principal, descritas no ANEXO 03 e contidas no arquivo 'TOOLBOX.LIB' do disco de instalação. (29-30)

PROGRAMA PRINCIPAL:

```
'  ---------------------------------------------------------------------------
'  |
'  |AMBIENTE INTERATIVO DE DESENVOLVIMENTO DE TESTE E MEDIDAS-AIDTM
'  |
'  |       TRABALHO DE MESTRADO - UNICAMP - FEE - 1990
'  |
'  |       ORIENTADOR: Prof. Dr. FURIO DAMIANI
'  |
'  |       ALUNO: JOSE OTAVIO SIMOES        RA.: 905090
'  |                                                 Versao: 91.01
'  ---------------------------------------------------------------------------
DEFINT A-Z
'*****************************************************************
' DECLARACAO DO MODULOS QUE COMPOEM O AIDTM
'
'
DECLARE SUB Diretorio ()
DECLARE SUB Operador ()
DECLARE SUB Setup ()
DECLARE SUB SetMed (Menux%, Opedi$())
DECLARE SUB Planilha (Opedi$())
DECLARE SUB Grafico (Opedi$())
DECLARE SUB Arquivo (File$, Menux%)
DECLARE SUB Montador ()
DECLARE SUB Compilador ()
DECLARE SUB ConfiGPIB ()
DECLARE SUB RecuArq (Menux%)
DECLARE SUB DeleArq (Menux%)
DECLARE SUB SaveArq (Menux%)
DECLARE SUB Editor (Menux%)
DECLARE SUB GPIBexe ()
DECLARE SUB Unidade (Unit$, Dir$)
```

```
'
'*****************************************************************
'
REM DEFINICAO DAS CONSTANTES DO PROGRAMA PRINCIPAL
'*****************************************************************
'
REM DEFINICAO DAS ESTRUTURAS DOS ARQUIVOS
TYPE SenhaType
     OperatorName AS STRING * 15
     OperatorSenha AS DOUBLE
END TYPE

'
TYPE FileDirType
     OperatorName AS STRING * 15
     FileName AS STRING * 25
     FileDes AS STRING * 35
END TYPE
'
TYPE Regtypex
     ax AS INTEGER
     bx AS INTEGER
     cx AS INTEGER
     dx AS INTEGER
     bp AS INTEGER
     si AS INTEGER
     di AS INTEGER
     flags AS INTEGER
     ds AS INTEGER
     es AS INTEGER
END TYPE
'
TYPE FileSetup
     LineSet AS STRING * 128
END TYPE
'
'*****************************************************************
'
REM DIMENSIONAMENTO DAS VARIAVEIS DO SISTEMA.
'
'========================  M O U S E  ==========================
'
CONST TRUE = 1, FALSE = 0, YES = 1, NO = 0, LEFT = 0, RIGHT = 1
CONST BOTH = 2, EITHER = 3, HARDCURSOR = 1, SOFTCURSOR = 0
'
' DEFINICAO DO TIPO DE DADOS

TYPE mouseinfo
     Exists AS INTEGER          ' > 0 se mouse existe
     CursorOn AS INTEGER        ' 1 se cursor ativo, 0 desligado
     BtnStatus AS INTEGER       ' status do botao ativo
     BtnClicks AS INTEGER       ' vezes que o botao ativo foi apertado
     Column AS INTEGER          ' coluna do mouse
     Row AS INTEGER             ' linha do mouse
     HMovement AS INTEGER       ' deslocamento horizontal do mouse
     VMovement AS INTEGER       ' deslocamento vertical do mouse
END TYPE
```

```
DIM SHARED button AS INTEGER
DIM SHARED a AS INTEGER, b AS INTEGER, c AS INTEGER, d AS INTEGER
DIM SHARED cursortype AS INTEGER, scan1 AS INTEGER, scan2 AS INTEGER
DIM SHARED leftcol AS INTEGER, rightcol AS INTEGER
DIM SHARED upperrow AS INTEGER, lowerrow AS INTEGER
DIM SHARED RRow  AS INTEGER, CCol AS INTEGER

DIM Mous AS mouseinfo
'=============================================================================
'================  MODULO DO EDITOR DE SENHA  ================
'
DIM SHARED SenhaRecord AS SenhaType
COMMON SHARED NumberOfSenha AS INTEGER
DIM SHARED Senha AS SenhaType
DIM SHARED RecordSenha AS SINGLE
COMMON SHARED OpName  AS STRING * 15
'
DIM SHARED InRegs AS Regtypex
DIM SHARED OutRegs AS Regtypex
'
'================  MODULO DO EDITOR DE SETUP  ============
'
DIM SHARED FileSet AS STRING
DIM SHARED FileTemp AS STRING
COMMON SHARED TabDir AS FileDirType
DIM SHARED NumberOfDir AS SINGLE
DIM SHARED Menux AS INTEGER, PointerDir AS INTEGER
COMMON SHARED Cmd AS STRING
DIM SHARED Syntax AS STRING
COMMON SHARED Pfile AS INTEGER
COMMON SHARED Pvideo AS INTEGER
COMMON SHARED Pline AS INTEGER
DIM SHARED FileAtive AS STRING
DIM SHARED ModuloAtive AS STRING
DIM SHARED Unit AS STRING
DIM SHARED Dir AS STRING
DIM SHARED UnitQB AS STRING
DIM SHARED DirQB AS STRING
'
'================  MODULO DE ABERTURA DE ARQUIVOS ========
'
DIM SHARED Linex AS FileSetup
DIM SHARED Linein(500) AS STRING
DIM SHARED Ferro AS INTEGER
'
'================  MODULO DE ARQUIVOS ==================
DIM SHARED File AS STRING
'================  MODULO DO EDITOR DE GRAFICO ==================
'
DIM SHARED NomeGrafo AS STRING
DIM SHARED TipoGrafo AS INTEGER
DIM SHARED TipoGra AS STRING
DIM SHARED TitlePri AS STRING
DIM SHARED TitleSec AS STRING
DIM SHARED TitleX AS STRING
DIM SHARED TitleY AS STRING
DIM SHARED Faixas AS STRING
DIM SHARED TempFile AS STRING
DIM SHARED StopFaixa AS INTEGER
DIM SHARED StarFaixa AS INTEGER
'
```

```
'================== MODULO DO EDITOR DE PLANILHA ==================
'
DIM SHARED NomePlani AS STRING
DIM SHARED TipoPla AS INTEGER
DIM SHARED TitleFA AS STRING
DIM SHARED TitleFB AS STRING
DIM SHARED TitleFC AS STRING
DIM SHARED TitleFD AS STRING
DIM SHARED TitleFE AS STRING
DIM SHARED TitleFF AS STRING
DIM SHARED TitleFG AS STRING
DIM SHARED TitleFH AS STRING
'
'*****************************************************************
'
REM DEFINICAO DAS FUNCOES E SUBROTINAS
REM $INCLUDE: 'toolbox.bi'
'
'#################################################################################################
#
REM INICIO DA PROGRAMA PRINCIPAL
'
'***************************************
TIMER OFF
Mous.Exists = 0

'**********************************
REM MENU PRINCIPAL - CHAMADA DOS MODULOS
CALL M.FindMouse
'**********************************
'
REM SENHA DO OPERADOR
CALL FirstSenha
'
CALL QBDir(UnitQB$, DirQB$)
CALL FIDir(Unit$, Dir$)
'
'**********************************
REM MENU PRINCIPAL - CHAMADA DOS MODULOS
hpos = 1
DIM SHARED Opcao$(7, 2)
Opcao$(1, 1) = "AMBIENTE"
Opcao$(1, 2) = "Define o ambiente de trabalho"
Opcao$(2, 1) = "EDITORES"
Opcao$(2, 2) = "Editores de SETUP, MEDIDA, GRAFICOS e TABELAS"
Opcao$(3, 1) = "MONTADOR"
Opcao$(3, 2) = "Montagem do arquivo de Teste e/ou Medida"
Opcao$(4, 1) = "COMPILADOR"
Opcao$(4, 2) = "Geracao do Arquivo Executavel de Teste e/ou Medida"
Opcao$(5, 1) = "ARQUIVOS"
Opcao$(5, 2) = "Diretorio com arquivos anteriores"
Opcao$(6, 1) = "CONFIGURACAO"
Opcao$(6, 2) = "Configuracao do barramento GPIB/488"
Opcao$(7, 1) = "XFIM"
Opcao$(7, 2) = "FINAL DE OPERACOES"
Escolha = 1
ON TIMER(180) GOSUB TimeOut
DO WHILE Escolha <> 0
    ON ERROR GOTO Roterro
    COLOR 14, 9, 1: CLS
      Mes$  = "   AMBIENTE  INTERATIVO DE DESENVOLVIMENTO DE TESTES E
MEDIDAS     V0.0/90"
```

```
CALL Saymes(2, 0, 2, Mes$)
CALL Borda(6, 1, 20, 80, 1)
CALL Borda(21, 1, 23, 80, 1)
Mes$ = " Operador: " + RTRIM$(LTRIM$(OpName$)) + " "
CALL Saymes(23, 78 - LEN(Mes$), 0, Mes$)
Escolha = Menu(4, 1, Opcao$())
SELECT CASE Escolha
CASE 1
     REDIM SHARED Op$(4)
     Op$(1) = "UNIDADE"
     Op$(2) = "DIRETORIO"
     Op$(3) = "OPERADOR"
     Op$(4) = "XFIM"
     Menux = Achoice%(7, 2, Op$())
     SELECT CASE Menux
     CASE 1
          CALL Unidade(Unit$, Dir$)
     CASE 2
          CALL Diretorio
     CASE 3
          CALL Operador
     CASE ELSE
          BEEP
     END SELECT
CASE 2
     REDIM SHARED Op$(5)
     Op$(1) = "SETUP"
     Op$(2) = "MEDIDAS"
     Op$(3) = "GRAFICO"
     Op$(4) = "PLANILHA"
     Op$(5) = "XFIM"
     Menux = Achoice%(7, 13, Op$())
     CALL Editor(Menux)
CASE 3
     CALL Montador
CASE 4
     CALL Compilador
CASE 5
     REDIM SHARED Op$(6)
     Op$(1) = "SETUP"
     Op$(2) = "MEDIDA"
     Op$(3) = "GRAFICO"
     Op$(4) = "PLANILHA"
     Op$(5) = "MONTADOS"
     Op$(6) = "XFIM"
     Menux = Achoice%(7, 46, Op$())
     SELECT CASE Menux
     CASE 1
          File$ = "DIRSET.DAT"
          CALL Arquivo(File$, Menux)
     CASE 2
          File$ = "DIRMED.DAT"
          CALL Arquivo(File$, Menux)
     CASE 3
          File$ = "DIRGRA.DAT"
          CALL Arquivo(File$, Menux)
     CASE 4
          File$ = "DIRPLA.DAT"
          CALL Arquivo(File$, Menux)
     CASE 5
          File$ = "DIRBAS.DAT"
          CALL Arquivo(File$, Menux)
```

```
            CASE ELSE
                BEEP
            END SELECT
        CASE 6
            REDIM SHARED Op$(4)
            Op$(1) = "IBCONF"
            Op$(2) = "IBIC"
            Op$(3) = "ARQUIVO"
            Op$(4) = "XFIM"
            Menux = Achoice%(7, 58, Op$())
            IF Menux <> 0 THEN
                CALL GPIBexe
            END IF
        CASE 7
            EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    LOOP
    CLOSE

    a = SetDefaultDri(InRegs, OutRegs, UnitQB$) = 0
    CHDIR DirQB$

    COLOR 14, 9, 1: CLS
    END
    '
    '*********************************************
    ' SUBROTINA DE ERRO
    Roterro:
          CALL Roterror(Ferro)
          RESUME NEXT
    '
    '*********************************************
    'SUBROTINA DE TIMEOUT DO MOUSE - 3 MINUTOS
    TimeOut:
    Mes$ = "Mouse desativado por TIMEOUT (3 min) - use o teclado !!!"
        CALL Mensagem(Mes$)
        Mous.Exists = 0
        TIMER OFF
        RETURN
    '
    '
    '*********************************************
    ' FINAL DO PROGRAMA PRINCIPAL
    END
```

NOME DA SUBROTINA: Arquivo
PARAMETROS: Files$, Menux%
DESCRICAO: Executa manipulação de arquivo com facilidades de edição, impressão, cancelamento e visualização do arquivo escolhido.

```
SUB Arquivo (File$, Menux%)
DirFile$ = File$
NumberFile = 2
SizeFile = LEN(TabDir)
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfDir = LOF(2) / LEN(TabDir)
Confirme$ = SPACE$(1)
IF NumberOfDir = 0 THEN
    CALL DirZero
    EXIT SUB
END IF
Mes$ = "ESCOLHA O ARQUIVO PARA MANIPULAR !!"
CALL Saymes(22, 0, 0, Mes$)
CALL MostraDir(PointerDir)
IF PointerDir <> 0 THEN
    Mes$ = "Confirme o arquivo (S/N) : " + TabDir.FileName + "  "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        File$ = RTRIM$(TabDir.FileName)
    ELSE
        CALL LimpaVideo
        CLOSE #2
        EXIT SUB
    END IF
ELSE
    CALL LimpaVideo
    CLOSE #2
    EXIT SUB
END IF

DIM OpArq$(5, 2)
OpArq$(1, 1) = "EDITAR"
OpArq$(1, 2) = "Edita arquivo selecionado - Quick EDITOR"
OpArq$(2, 1) = "VISUALIZAR"
OpArq$(2, 2) = "Visualiza arquivo selecionado  - MORE"
OpArq$(3, 1) = "IMPRIMIR"
OpArq$(3, 2) = "Imprime arquivo selecionado"
OpArq$(4, 1) = "CANCELAR"
OpArq$(4, 2) = "Cancela arquivo selecionado"
OpArq$(5, 1) = "XFIM"
OpArq$(5, 2) = "FINAL DE OPERACOES"

Escolha = 1
DO WHILE Escolha <> 0
    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
    CALL LimpaVideo
    Escolha = Menu(4, 1, OpArq$())
    SELECT CASE Escolha
```

```
        CASE 1
             SCREEN 0, 7, 1, 0: COLOR 14, 9, 1: CLS
             SHELL UnitQB$ + DirQB$ + "\EDITOR " + File$
             CLS : SCREEN 0, 7, 0, 0
             EXIT DO
        CASE 2
             SCREEN 0, 7, 1, 1: COLOR 14, 9, 1: CLS
             SHELL UnitQB$ + DirQB$ + "\MORE < " + File$
             CLS : SCREEN 0, 7, 0, 0
        CASE 3
             Mes$ = "Prepare a impressora e confirme a impressao (S/N):"
             CALL Getmes(22, 0, 0, Mes$, Confirme$)
             IF UCASE$(Confirme$) = "S" THEN
                  SCREEN 0, 7, 1, 1: COLOR 14, 9, 1: CLS
                  Llinha$ = SPACE$(80)
                  OPEN File$ FOR INPUT AS #3
                  DO WHILE NOT EOF(3)
                       LINE INPUT #3, Llinha$
                       LPRINT Llinha$
                  LOOP
                  CLOSE #3
                  CLS : SCREEN 0, 7, 0, 0
             END IF
        CASE 4
             CALL DeleArq(Menux)
             EXIT DO
        CASE 5
             EXIT DO
        CASE ELSE
             BEEP
        END SELECT
   LOOP
   VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
   ERASE OpArq$
   CLOSE #2
   END SUB
```

NOME DA SUBROTINA: Compilador
PARAMETROS: Nenhum
DESCRICAO: Escolhe o arquivo a se compilado, ou seja, para gerar o arquivo executável e prepara os parâmetros a serem passados ao programa de "Link" e "Compilação".

```
SUB Compilador
ModPri$ = ""
File$ = "DIRBAS.DAT"
NumberFile = 2
SizeFile = LEN(TabDir)
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfDir = LOF(2) / LEN(TabDir)
Confirme$ = SPACE$(1)
IF NumberOfDir = 0 THEN
     CALL DirZero
     EXIT SUB
END IF
Mes$ = "ESCOLHA O ARQUIVO PARA COMPILAR  !!"
CALL Saymes(22, 0, 0, Mes$)
```

```
CALL MostraDir(PointerDir)
IF PointerDir <> 0 THEN
    Mes$ = "Confirme o arquivo (S/N) : " + TabDir.FileName + "  "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        ModPri$ = TabDir.FileName
    END IF
END IF
CALL LimpaVideo
'-------------------------------------------------------------
' Gerador do arquivo de Compilacao
IF ModPri$ = "" THEN
    CLOSE #2
    EXIT SUB
END IF
SCREEN 0, 7, 1, 1: COLOR 14, 9, 1
File$ = RTRIM$(ModPri$)
File$ = MID$(File$, 1, LEN(File$) - 4)
FileComp$ = File$
Filexe$ = File$ + ".EXE"
CALL FindBar(File$)

FileObj$ = File$

CALL FIDir(Unit$, Dir$)

File$ = "FILA.TMP"
NumberFile = 2
SizeFile = 80

CALL OpenSeqFile(File$, NumberFile, SizeFile)
Linein(1)  = "  /EX /NOE  " +  FileObj$ +  " "  + UnitQB$  + DirQB$ +
"\QBIB4.LIB"
Linein(2) = Filexe$
Linein(3) = ""
Linein(4) = UnitQB$ + DirQB$ + "\TOOLBOX.LIB"
FOR x = 1 TO 4
    PRINT #2, Linein(x)
NEXT x
CLOSE #2
SHELL "cls"
SHELL    UnitQB$   +    DirQB$   +   "\BC   "   +   FileComp$   +   "
/E/X/O/V/W/AH/T/C:512;"
SHELL UnitQB$ + DirQB$ + "\LINK @" + Unit$ + Dir$ + "\FILA.TMP"
KILL Unit$ + Dir$ + "\FILA.TMP"
SLEEP 5
SCREEN 0, 7, 0, 0: COLOR 14, 9, 1

END SUB
```

NOME DA SUBROTINA: ConfiGPIB
PARAMETROS: Nenhun
DESCRICAO: Ativa os programas executável de configuração do GPIB, de dentro do ambiente, permite a verificação dos instrumentos conectados ao barramento GPIB.

```
SUB ConfiGPIB
File$ = "DIRBAS.DAT"
DirFile$ = File$
NumberFile = 2
SizeFile = LEN(TabDir)
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfDir = LOF(2) / LEN(TabDir)
IF NumberOfDir = 0 THEN
    CALL DirZero
    EXIT SUB
END IF

Mes$ = "ESCOLHA O ARQUIVO PARA VERIFICAR A CONFIGURACAO !!"
CALL Saymes(22, 0, 0, Mes$)
CALL MostraDir(PointerDir)
Confirme$ = SPACE$(1)
IF PointerDir <> 0 THEN
    Mes$ = "Confirme o arquivo (S/N) : " + TabDir.FileName + "   "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        File$ = RTRIM$(TabDir.FileName)
      END IF
ELSE
    CALL LimpaVideo
    CLOSE #2
    EXIT SUB
END IF
CALL LimpaVideo
CALL FindBar(File$)

NumberFile = 10
SizeFile = LEN(Linex)
CALL OpenSeqFile(File$, NumberFile, SizeFile)
dev = 1
Linhain$ = ""
DIM Device$(16)
DO WHILE NOT EOF(10)
    LINE INPUT #10, Linhain$
    IF INSTR(1, Linhain$, "CALL IB") <> 0 THEN
        Linhain$ = RTRIM$(Linhain$)

        x = INSTR(1, Linhain$, "(")
        Linhain$ = MID$(Linhain$, x + 1, LEN(Linhain$) - x)

        x = INSTR(1, Linhain$, "%")
        Linhain$ = MID$(Linhain$, 1, x - 1)
```

```
            IF INSTR(1, Linhain$, ",") <> 0 THEN
                  x = INSTR(1, Linhain$, ",")
                  Linhain$ = MID$(Linhain$, x + 1, LEN(Linhain$))

            END IF
            FOR x = 1 TO 16
                IF Linhain$ = Device$(x) THEN
                    EXIT FOR
                END IF
            NEXT x
            IF x = 17 THEN
                Device$(dev) = Linhain$
                dev = dev + 1
            END IF
        END IF
    LOOP
    CLOSE #10

    IF dev = 1 THEN
        Mes$ = "NAO EXISTE INSTRUMENTOS NO BARRAMENTO GPIB !!!"
    ELSE
        CALL Saymes(8, 5, 2, "GPIB0")
        LOCATE 8, 13: PRINT CHR$(196); CHR$(196)
        Y = 9
        Col = 20
        FOR x = 1 TO dev - 1
            CALL Saymes(Y, Col, 1, Device$(x))

            FOR z = Y - 2 TO Y
                LOCATE z, Col - 5: PRINT CHR$(179)
            NEXT z
            FOR z = Col - 5 TO Col - 3
                LOCATE Y, z: PRINT CHR$(196)
            NEXT z

            IF Col > 20 AND Y = 9 THEN
                LOCATE 7, Col - 5: PRINT CHR$(194)
            END IF
            IF Col = 20 AND Y = 9 THEN
                LOCATE 8, 15: PRINT CHR$(180)
            END IF
            LOCATE Y, Col - 5: PRINT CHR$(195)
            Y = Y + 3
            IF Y = 21 AND (x MOD 4) = 0 THEN
                Y = 9
                LOCATE 7, Col - 5: PRINT CHR$(194);
                FOR t = Col - 4 TO Col + 9
                    LOCATE 7, t: PRINT CHR$(196);
                NEXT t
                Col = Col + 15
            END IF

        NEXT x
        Mes$ = "INSTRUMENTOS NO BARRAMENTO GPIB !!!"
    END IF
    ERASE Device$
    CALL Mensagem(Mes$)
    END SUB
```

NOME DA SUBROTINA: DeleArq

PARAMETROS: Menux%

DESCRICAO: Elimina arquivo do disco de trabalho e do arquivo que contem a lista dos arquivos editados ou montados. Não executa 'backup' do arquivo eliminado.

```
SUB DeleArq (Menux)
Confirme$ = SPACE$(1)

SELECT CASE Menux
CASE 1
    File$ = "DIRSET.DAT"
CASE 2
    File$ = "DIRMED.DAT"
CASE 3
    File$ = "DIRGRM.DAT"
CASE 4
    File$ = "DIRPLM.DAT"
CASE 5
    File$ = "DIRBAS.DAT"
END SELECT
TempFile$ = "\" + File$
NumberFile = 2
SizeFile = LEN(TabDir)
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfDir = LOF(2) / LEN(TabDir)
IF NumberOfDir = 0 THEN
    CALL DirZero
    EXIT SUB
END IF
IF Menux <> 5 THEN
    Mes$ = "ESCOLHA O ARQUIVO PARA CANCELAR !!"
    CALL Saymes(22, 0, 0, Mes$)
    CALL MostraDir(PointerDir)
END IF
IF PointerDir <> 0 THEN
      Mes$  = "Confirme  o  cancelamento  do  arquivo  (S/N)  :  "  +
TabDir.FileName + "  "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        File$ = RTRIM$(TabDir.FileName)
        CALL FindBar(File$)
        IF LTRIM$(RTRIM$(FileAtive$)) = LTRIM$(RTRIM$(File$)) THEN
            Mes$ = "ARQUIVO ABERTO, IMPOSSIVEL DELETAR !!!"
            CALL Mensagem(Mes$)
            FOR x = 7 TO 19
                LOCATE x, 2: PRINT SPACE$(78)
            NEXT x
            CLOSE #2
            EXIT SUB
        END IF
        KILL TabDir.FileName
        CALL DelFileDir(NumberOfDir, PointerDir)
        CLOSE #2
        KILL Unit$ + Dir$ + TempFile$
        NAME Unit$ + Dir$ + "\TEMP.DAT" AS Unit$ + Dir$ + TempFile$
```

```
    ELSE
        BEEP
    END IF
ELSE
    LOCATE 22, 2: PRINT SPACE$(76)
END IF
FOR x = 7 TO 19
    LOCATE x, 2: PRINT SPACE$(78)
NEXT x
END SUB
```

NOME DA SUBROTINA: Diretório
PARAMETROS: Nenhum
DESCRICAO: Executa procedimento de listagem do diretório ativo. Função padrão do MS-DOS ativada de dentro do ambiente.

```
SUB Diretorio
ON ERROR GOTO Roterro
SCREEN 0, 7, 1, 1: COLOR 14, 9, 1: CLS
SHELL "dir *.bas/p"
SLEEP: CLS : SCREEN 0, 7, 0, 0
END SUB
```

NOME DA SUBROTINA: Editor
PARAMETROS: Menux%
DESCRICAO: Ativa os editores de 'SETUP', MEDIDAS, GRAFICO e PLANILHAS. Escolha através de pop-menu e linhas de comando. Cria arquivo temporário para edição do módulo escolhido.

```
SUB Editor (Menux%)
VIEW PRINT 4 TO 20
COLOR 14, 9, 1: CLS : VIEW PRINT
CALL Borda(6, 1, 20, 80, 1)
CALL FIDir(Unit$, Dir$)

SELECT CASE Menux
CASE 1
    FileAtive$ = "SETUP.$$$"
    ModuloAtive$ = "SETUP:"
CASE 2
    FileAtive$ = "MEDE.$$$"
    ModuloAtive$ = "MEDIDA:"
CASE 3
    FileAtive$ = "GRAFO.$$$"
    ModuloAtive$ = "GRAFICO:"
CASE 4
    FileAtive$ = "PLANI.$$$"
    ModuloAtive$ = "PLANILHA:"
END SELECT
```

```
FileTemp$ = Unit$ + Dir$ + "\" + FileAtive$
NumberFile = 1
SizeFile = LEN(Linex)
File$ = FileAtive$
CALL OpenFile(File$, NumberFile, SizeFile)

Linein(1)  = "'***************  ARQUIVO TEMPORARIO DE " + ModuloAtive$
+ " ********************"
Linein(2)   =  "'Nome   do  Operador=   "  +   RTRIM$(OpName$)  +   "
Data: " + DATE$ + "        Hora: " + TIME$
Linein(3) = "'Nome do Arquivo = '" + FileAtive$
Linein(4) = ModuloAtive$

FOR Pline = 1 TO 4
    PUT #1, Pline, Linein(Pline)
NEXT Pline

SELECT CASE Menux
CASE 1, 2
    DIM Opedi$(8, 2)
    Opedi$(1, 1) = "1. GRUPO"
      Opedi$(1,  2) =  "IBREAD   IBWRITE    IBSPOLL  IBCLEAR   IBFIND
EDITA   <    >    XFIM"
    Opedi$(2, 1) = "2. GRUPO"
      Opedi$(2,  2) =  "IBIFC    IBTRIG     IBLOC    IBREM     IBWAIT
EDITA   <    >    XFIM"
    Opedi$(3, 1) = "3. GRUPO"
      Opedi$(3,  2) =  "PPOLL    CPOOL     UPOOL    LOCKOUT   SENDCMD
EDITA   <    >    XFIM"
    Opedi$(4, 1) = "QBASIC"
    Opedi$(4, 2) = "Funcoes e Comandos do QuickBASIC"
    Opedi$(5, 1) = "SALVAR"
    Opedi$(5, 2) = "Salva o arquivo na unidade ativa"
    Opedi$(6, 1) = "CANCELAR"
    Opedi$(6, 2) = "Cancela arquivos da unidade ativa"
    Opedi$(7, 1) = "RECUPERAR"
    Opedi$(7, 2) = "Recupera arquivos ja salvos"
    Opedi$(8, 1) = "XFIM"
    Opedi$(8, 2) = "FINAL DE OPERACOES"
CASE 3, 4
    DIM Opedi$(9, 2)
    IF Menux = 3 THEN
        Opedi$(1, 1) = "TIPO"
         Opedi$(1, 2) = "Tipo do Grafico a ser gerado com o resultado
das medidas"
    ELSE
        Opedi$(1, 1) = "COLUNA"
        Opedi$(1, 2) = "Numero de colunas que compoem a Planilha"
    END IF
    Opedi$(2, 1) = "ARQUIVO"
      Opedi$(2,  2) =  "Nome do arquivo padrao que contem o resultado
das medidas"
    Opedi$(3, 1) = "FAIXA"
      Opedi$(3,  2) =  "Escolha das  faixas que  compoem o Grafico ou
Planilha (Maximo de 8)"
    Opedi$(4, 1) = "TITULOS"
    Opedi$(4, 2) = "Titulos do Grafico e/ou Planilha"
    Opedi$(5, 1) = "GERAR"
    Opedi$(5, 2) = "Geracao do arquivo que produzira o grafico"
    Opedi$(6, 1) = "SALVAR"
    Opedi$(6, 2) = "Salva o arquivo na unidade ativa"
    Opedi$(7, 1) = "CANCELAR"
```

```
    Opedi$(7, 2) = "Cancela arquivos da unidade ativa"
    Opedi$(8, 1) = "RECUPERAR"
    Opedi$(8, 2) = "Recupera arquivos ja salvos"
    Opedi$(9, 1) = "XFIM"
    Opedi$(9, 2) = "FINAL DE OPERACOES"
END SELECT
' incializacao das variaveis
'
' VARIAVEIS DO GRAFICO
'
Pfile = Pline
NomeGrafo$ = ""
TipoGra$ = ""
TipoGrafo = 0
TitlePri$ = ""
TitleSec$ = ""
TitleX$ = ""
TitleY$ = ""
StartFaixa = 0
StopFaixa = 100
Faixas$ = ""
'
' VARIAVEIS DA PLANILHA
'
NomePlani$ = ""
TipoPla = 0
TitleFA$ = ""
TitleFB$ = ""
TitleFC$ = ""
TitleFD$ = ""
TitleFE$ = ""
TitleFF$ = ""
TitleFG$ = ""
TitleFH$ = ""
'
Escolha = 1
Passo1 = Passo2 = Passo3 = FALSE
'
SELECT CASE Menux
CASE 1, 2
    CALL SetMed(Menux, Opedi$())
CASE 3
    CALL Grafico(Opedi$())
CASE 4
    CALL Planilha(Opedi$())
END SELECT

CLOSE #1
CLOSE #2
ERASE Opedi$
KILL Unit$ + Dir$ + "\*.$$$"
END SUB
```

NOME DA SUBROTINA: GPIBexe

PARAMETROS: Nenhum

DESCRICAO: Ativa de dentro do ambiente o programa de reconfiguração dos instrumentos do barramento. (NATIONAL).

```
SUB GPIBexe
Confirme$ = SPACE$(1)
SELECT CASE Menux
CASE 1
    Mes$ = "Deseja Reconfigurar os Instrumentos (S/N) :   "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        CALL FIDir(Unit$, Dir$)
           Mes$ = "CUIDADO !!!, APOS A CONFIGURACAO SERA' DADO O BOOT
NO SISTEMA"
        CALL Mensagem(Mes$)
        COLOR 14, 9, 1: CLS : SCREEN 0, 7, 1, 1: COLOR 14, 9, 1: CLS
        SHELL "cd " + UnitQB$ + DirQB$
        SHELL "c:\gpib-pc\ibconf"
        SHELL "cd " + Unit$ + Dir$
        CLS : SCREEN 0, 7, 0, 0
    ELSE
        BEEP
    END IF
    LOCATE 22, 2: PRINT SPACE$(76)
CASE 2
    Mes$ = "Deseja Operar os Instrumentos (S/N) :   "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        COLOR 14, 9, 1: CLS : SCREEN 0, 7, 1, 1: COLOR 14, 9, 1: CLS
        SHELL "cd " + UnitQB$ + DirQB$
        SHELL "c:\gpib-pc\ibic"
        SHELL "cd " + Unit$ + Dir$
        CLS : SCREEN 0, 7, 0, 0
    ELSE
        BEEP
    END IF
    LOCATE 22, 2: PRINT SPACE$(76)
CASE 3
    CALL ConfiGPIB
CASE ELSE
    BEEP
END SELECT

END SUB
```

NOME DA SUBROTINA: Grafico
PARAMETROS: Opedi$()
DESCRICAO: Edita o módulo GRAFICO para a subrotina editor. Salvar os valores inseridos no arquivo temporário criado pelo ambiente.

```
SUB Grafico (Opedi$())
Escolha = 1
Confirme$ = SPACE$(1)
DO WHILE Escolha <> 0
     FOR x = 7 TO 19
         LOCATE x, 2: PRINT SPACE$(78)
     NEXT x
     Pvideo = 7
     FOR Pfile = 1 TO 4
         GET #1, Pfile, Linein(Pfile)
         LOCATE Pvideo, 2: PRINT Linein(Pfile)
         Pvideo = Pvideo + 1
     NEXT Pfile
     Pvideo = Pvideo - 1
     Pfile = Pfile - 1
     CALL Saymes(11, 5, 0, "ARQUIVO: " + NomeGrafo$)
     CALL Saymes(11, 40, 0, "TIPO: " + TipoGra$)
     CALL Saymes(12, 5, 0, "PRIN.: " + TitlePri$)
     CALL Saymes(13, 5, 0, "SEC.: " + TitleSec$)
     CALL Saymes(14, 5, 0, "EIXO X: " + TitleX$)
     CALL Saymes(15, 5, 0, "EIXO Y: " + TitleY$)
     FF$ = ""
     FOR f = 1 TO LEN(Faixas$)
          FF$ = FF$ + " - " + MID$(Faixas$, f, 1)
     NEXT f
     CALL Saymes(17, 5, 0, "FAIXAS ATIVAS:" + FF$)
       Mes$  = "INICIO:  " +  STR$(StartFaixa) +  "%  " + SPACE$(23) +
"FINAL: " + STR$(StopFaixa) + "%  "
     CALL Saymes(18, 5, 0, Mes$)

     ON ERROR GOTO Roterro
     Escolha = Menu(4, 1, Opedi$())
     SELECT CASE Escolha
     CASE 1
         GOSUB Tipo:
     CASE 2
         GOSUB Arquivo:
     CASE 3
         GOSUB Faixa:
     CASE 4
         GOSUB Titulos:
     CASE 5
         GOSUB GeraGrafo:
     CASE 6
         CALL SaveArq(Menux)
     CASE 7
         CALL DeleArq(Menux)
     CASE 8
         CALL RecuArq(Menux)
     CASE 9
```

```
        EXIT DO
    CASE ELSE
        BEEP
    END SELECT
LOOP
EXIT SUB
END

'*****************************************
' INICIO DAS SUBROTINAS DO EDITOR DE SETUP
'
'-----------------------------------------------------------------------
'
Tipo:
    DIM Op1$(2)
    Op1$(1) = "GRAFICO X - t"
    Op1$(2) = "GRAFICO X - Y"
    OpGx = Achoice%(7, 2, Op1$())
    SELECT CASE OpGx
    CASE 1
        TipoGrafo = 1
        TipoGra$ = "X - t"
        Mes$ = "ESCOLHIDO GRAFICO DO TIPO X - t"
    CASE 2
        TipoGrafo = 2
        TipoGra$ = "X - Y"
        Mes$ = "ESCOLHIDO GRAFICO DO TIPO X - Y"
    CASE ELSE
        BEEP
        RETURN
    END SELECT
    CALL Saymes(14, 0, 2, Mes$)
    SLEEP 1
    Passo1 = TRUE
    RETURN
'
'-----------------------------------------------------------------------
Arquivo:
    NomeGrafo$ = SPACE$(8)
    CALL Getmes(22, 0, 0, "NOME DO ARQUIVO : ", NomeGrafo$)
    IF NomeGrafo$ = "" THEN
        RETURN
    END IF
    IF LEN(NomeGrafo$) > 8 THEN
        Mes$ = "NOME DO ARQUIVO MAIOR QUE 8 CARACTERES !!!"
        CALL Mensagem(Mes$)
        RETURN
    ELSE
        SLEEP 1
        LOCATE 22, 2: PRINT SPACE$(76)
    END IF
    NomeGrafo$ = UCASE$(NomeGrafo$)
    GrafoSet$ = Unit$ + Dir$ + "\" + NomeGrafo$ + ".DAD"
    GrafoAtive$ = NomeGrafo$ + ".DAD"
    Passo2 = TRUE
    RETURN
```

```
'
'_____________________________________________________________________
Faixa:

    DIM OpsetF$(11, 2)
    FOR x = 1 TO 8
        OpsetF$(x, 1) = " " + CHR$(64 + x) + " "
        OpsetF$(x, 2) = "Faixa de dados - " + CHR$(64 + x)
    NEXT x

    OpsetF$(9, 1) = "INICIO%"
    OpsetF$(9, 2) = "Inicio da faixa para o garfico - 0%"
    OpsetF$(10, 1) = "FINAL%"
    OpsetF$(10, 2) = "Final da faixa para o grafico - 100%"
    OpsetF$(11, 1) = "XFIM"
    OpsetF$(11, 2) = "FINAL DE OPERACOES"

    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT

    SELECT CASE TipoGrafo
    CASE 1
        Mes$ = "GRAFICO DO TIPO X - t, FAIXAS SEQUENCIAIS"
    CASE 2
         Mes$ = "GRAFICO DO TIPO X - Y, a PRIMEIRA FAIXA DEFINE EIXO
X !!!"
    CASE ELSE
        Mes$ = "Tipo de grafico nao definido !!!"
        CALL Mensagem(Mes$)
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        ERASE OpsetF$
        RETURN
    END SELECT
    CALL Saymes(22, 0, 0, Mes$)
    SLEEP 2
    LOCATE 22, 2: PRINT SPACE$(76)
    DO WHILE Escolha <> 0
        Escolha = Menu(4, 1, OpsetF$())
        SELECT CASE Escolha
        CASE 1 TO 8
            Letra$ = CHR$(64 + Escolha)
            GOSUB TestaLetra:
        CASE 9
            OldFaixa = StartFaixa
              LOCATE 22, 15: INPUT "Inicio da faixa do grafico (0%):
", StartFaixa
            LOCATE 22, 2: PRINT SPACE$(78)
            IF StartFaixa > StopFaixa THEN
                Mes$ = "DEFINICAO DE FAIXA INCORRETA !!!"
                CALL Mensagem(Mes$)
                StartFaixa = OldFaixa
                EXIT DO
            END IF
            IF StartFaixa < 0 THEN
                StartFaixa = 0
            END IF
                 Mes$  = "INICIO:  " +  STR$(StartFaixa) +  "%   " +
SPACE$(23) + "FINAL: " + STR$(StopFaixa) + "%  "
            CALL Saymes(18, 5, 0, Mes$)
        CASE 10
            OldFaixa = StopFaixa
              LOCATE 22, 15: INPUT "Final da faixa do grafico (100%):
", StopFaixa
```

```
                LOCATE 22, 2: PRINT SPACE$(78)
                IF StopFaixa < StartFaixa THEN
                    Mes$ = "DEFINICAO DE FAIXA INCORRETA !!!"
                    CALL Mensagem(Mes$)
                    StopFaixa = OldFaixa
                    EXIT DO
                END IF
                IF StopFaixa > 100 THEN
                    StopFaixa = 100
                END IF
                 Mes$ = "INICIO: " + STR$(StartFaixa) + "% " + SPACE$(23)
+ "FINAL: " + STR$(StopFaixa) + "%  "
                CALL Saymes(18, 5, 0, Mes$)
            CASE 11
                EXIT DO
            CASE ELSE
                BEEP
            END SELECT
        LOOP
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        ERASE OpsetF$

    RETURN
    '
    '**********************************
    ' sub rotina usada pela sub Faixa:
    TestaLetra:
            FOR f = 1 TO LEN(Faixas$)
                IF MID$(Faixas$, f, 1) = Letra$ THEN
                    Mes$ = "Faixa ja selecionada !!!"
                    CALL Mensagem(Mes$)
                    RETURN
                END IF
            NEXT f
            Faixas$ = Faixas$ + Letra$
            FF$ = ""
            FOR f = 1 TO LEN(Faixas$)
                 FF$ = FF$ + " - " + MID$(Faixas$, f, 1)
            NEXT f
            CALL Saymes(17, 5, 0, "FAIXAS ATIVAS:" + FF$)
            Passo3 = TRUE
            RETURN
    '-------------------------------------------------------------------------
    Titulos:
        DIM Opset4$(5, 2)
        Opset4$(1, 1) = "PRINCIPAL"
        Opset4$(1, 2) = "Titulo principal do grafico"
        Opset4$(2, 1) = "SECUNDARIO"
        Opset4$(2, 2) = "Titulo secundario do grafico"
        Opset4$(3, 1) = "EIXO X"
        Opset4$(3, 2) = "Titulo do eixo X"
        Opset4$(4, 1) = "EIXO Y"
        Opset4$(4, 2) = "Titulo do exio Y"
        Opset4$(5, 1) = "XFIM"
        Opset4$(5, 2) = "FINAL DE OPERACOES"
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT

        DO WHILE Escolha <> 0
            Escolha = Menu(4, 1, Opset4$())
            SELECT CASE Escolha
```

```
        CASE 1
            Sintax$ = "Titulo Principal do Grafico :"
            Cmd$ = ""
            CALL Command(Sintax$, Cmd$)
            TitlePri$ = Cmd$
            CALL Saymes(12, 5, 0, "PRIN.: " + Cmd$)
        CASE 2
            Sintax$ = "Titulo Secundario do Grafico :"
            Cmd$ = ""
            CALL Command(Sintax$, Cmd$)
            TitleSec$ = Cmd$
            CALL Saymes(13, 5, 0, "SEC.: " + Cmd$)
        CASE 3
            Sintax$ = "Titulo do eixo X do Grafico :"
            Cmd$ = ""
            CALL Command(Sintax$, Cmd$)
            TitleX$ = Cmd$
            CALL Saymes(14, 5, 0, "EIXO X: " + Cmd$)
        CASE 4
            Sintax$ = "Titulo do eixo Y do Grafico (max. 20):"
            Cmd$ = ""
            CALL Command(Sintax$, Cmd$)
            TitleY$ = Cmd$
            IF LEN(TitleY$) > 20 THEN
                Mes$ = "TITULO Y MAIOR QUE 20 CARACTERES !!!"
                CALL Mensagem(Mes$)
                EXIT DO
            END IF
            CALL Saymes(15, 5, 0, "EIXO Y: " + Cmd$)
        CASE 5
            EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    LOOP
    ERASE Opset4$
    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
    RETURN
'
'_____________________________________________________________________
GeraGrafo:
    File$ = "DIRGRM.DAT"
    NumberFile = 2
    SizeFile = LEN(TabDir)
    CALL OpenFile(File$, NumberFile, SizeFile)
    NumberOfDir = LOF(2) / LEN(TabDir)
    IF NumberOfDir = 0 THEN
        CALL DirZero
        RETURN
    END IF
    Mes$ = "ESCOLHA O MODULO PARA GERAR O GRAFICO !!"
    CALL Saymes(22, 0, 0, Mes$)
    CALL MostraDir(PointerDir)
    IF PointerDir <> 0 THEN
        Mes$ = "Confirme o modulo (S/N) : " + TabDir.FileName + "  "
        CALL Getmes(22, 0, 0, Mes$, Confirme$)
        IF UCASE$(Confirme$) = "S" THEN
            CALL Getmes(22, 0, 0, "DESCRICAO :", DesFile$)
            File$ = RTRIM$(TabDir.FileName)
            FOR x = 1 TO LEN(File$)
```

```
                    IF MID$(File$, x, 1) = "." THEN
                        File$ = MID$(File$, 1, x - 1)
                        EXIT FOR
                    END IF
                NEXT x
                Modulo1$ = " + " + Unit$ + Dir$ + "\GRAF02.DAT    "
                Modulo2$ = File$ + ".GRA"
                SCREEN 0, 7, 1, 1
                SHELL "COPY " + TabDir.FileName + Modulo1$ + Modulo2$
                COLOR 14, 9, 1: CLS
                SCREEN 0, 7, 0, 0
                CLOSE #2
                File$ = "DIRGRA.DAT"
                NumberFile = 2
                SizeFile = LEN(TabDir)
                CALL OpenFile(File$, NumberFile, SizeFile)
                NumberOfDir = LOF(2) / LEN(TabDir)
                FOR x = 1 TO NumberOfDir
                    GET #2, x, TabDir
                                    IF  RTRIM$(LTRIM$(TabDir.FileName))  =
RTRIM$(LTRIM$(Modulo2$)) THEN
                        TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
                        TabDir.FileName = Modulo2$
                        TabDir.FileDes = DesFile$
                        PUT #2, x, TabDir
                        EXIT FOR
                    END IF
                NEXT x
                IF x > NumberOfDir THEN
                    NumberOfDir = NumberOfDir + 1
                    TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
                    TabDir.FileName = Modulo2$
                    TabDir.FileDes = DesFile$
                    PUT #2, NumberOfDir, TabDir
                END IF
                CLOSE #2
            END IF
        END IF
        CLOSE #2
        RETURN

END SUB
```

NOME DA SUBROTINA: Montador

PARAMETROS: Nenhum

DESCRICAO: Executa a montagem dos arquivos de testes, extensão .BAS, com os módulos editados e gerados pelos editores do ambiente. Salva arquivo montado no disco de trabalho e inclui no arquivo de lista.

```
SUB Montador
Confirme$ = SPACE$(1)
DIM OpMon$(6, 2)
OpMon$(1, 1) = "SETUP"
OpMon$(1,  2) =  "Escolhe o  modulo de SETUP para compor o arquivo de
testes"
OpMon$(2, 1) = "MEDIDA"
OpMon$(2,  2) =  "Escolhe o modulo de MEDIDA para compor o arquivo de
testes"
OpMon$(3, 1) = "GRAFICO"
OpMon$(3,  2) = "Escolhe o modulo de GRAFICO para compor o arquivo de
testes"
OpMon$(4, 1) = "PLANILHA"
OpMon$(4,  2) =  "Escolhe o  modulo de PALNILHA para compor o arquivo
de testes"
OpMon$(5, 1) = "MONTAGEM"
OpMon$(5, 2) = "Executa a montagem do arquivo de testes"
OpMon$(6, 1) = "XFIM"
OpMon$(6, 2) = "FINAL DE OPERACOES"

ModSet$ = ""
ModMed$ = ""
ModGra$ = ""
ModPla$ = ""
ModPri$ = ""

Escolha = 1
DO WHILE Escolha <> 0
    CALL Saymes(8, 0, 0, "MODULO DE SETUP : " + ModSet$)
    CALL Saymes(10, 0, 0, "MODULO DE MEDIDA: " + ModMed$)
    CALL Saymes(12, 0, 0, "MODULO DE GRAFICO: " + ModGra$)
    CALL Saymes(14, 0, 0, "MODULO DE PLANILHA: " + ModPla$)
    FOR x = 17 TO 19
        LOCATE x, 2: PRINT SPACE$(76)
    NEXT x
    CALL Saymes(18, 0, 1, "MODULO PRINCIPAL: " + ModPri$)
    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
    Escolha = Menu(4, 1, OpMon$())
    SELECT CASE Escolha
    CASE 1
        File$ = "DIRSET.DAT"
        GOSUB Modulo:
    CASE 2
        File$ = "DIRMED.DAT"
        GOSUB Modulo:
    CASE 3
        File$ = "DIRGRA.DAT"
        GOSUB Modulo:
```

```
        CASE 4
            File$ = "DIRPLA.DAT"
            GOSUB Modulo:
        CASE 5
            File$ = "DIRBAS.DAT"
            GOSUB GeraBas:
        CASE 6
            EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    LOOP
    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
    CLOSE #2
    ERASE OpMon$
    EXIT SUB
    '-------------------------------------------------------------------------
    ' Busca os modulos de SETUP, MEDIDA, GRAFICO e PLANILHA

    Modulo:
        NumberFile = 2
        SizeFile = LEN(TabDir)
        CALL OpenFile(File$, NumberFile, SizeFile)
        NumberOfDir = LOF(2) / LEN(TabDir)
        IF NumberOfDir = 0 THEN
            CALL DirZero
            RETURN
        END IF
        Mes$ = "ESCOLHA O ARQUIVO PARA MONTAGEM !!"
        CALL Saymes(22, 0, 0, Mes$)
        CALL MostraDir(PointerDir)
        IF PointerDir <> 0 THEN
            Mes$ = "Confirme o arquivo (S/N) : " + TabDir.FileName + " "
            CALL Getmes(22, 0, 0, Mes$, Confirme$)
            IF UCASE$(Confirme$) = "S" THEN
                SELECT CASE Escolha
                CASE 1
                    ModSet$ = TabDir.FileName
                CASE 2
                    ModMed$ = TabDir.FileName
                CASE 3
                    ModGra$ = TabDir.FileName
                CASE 4
                    ModPla$ = TabDir.FileName
                END SELECT
            END IF
        END IF
        CALL LimpaVideo
        RETURN
    '-------------------------------------------------------------------------
    ' Gerador do arquivo de teste
    GeraBas:
        IF ModSet$ = "" THEN
            ModSet$ = UnitQB$ + DirQB$ + "\NOSET.DAT"
        END IF
        IF ModMed$ = "" THEN
            ModMed$ = UnitQB$ + DirQB$ + "\NOMED.DAT"
        END IF
        IF ModGra$ = "" THEN
            ModGra$ = UnitQB$ + DirQB$ + "\NOGRA.DAT"
        END IF
```

```
    IF ModPla$ = "" THEN
        ModPla$ = UnitQB$ + DirQB$ + "\NOPLA.DAT"
    END IF
    ModPri$ = SPACE$(8)
    CALL Getmes(22, 0, 0, "NOME DO ARQUIVO DE TESTE : ", ModPri$)
    IF ModPri$ = "" THEN
        RETURN
    END IF
    ModPri$ = UCASE$(ModPri$)
    CALL Getmes(22, 0, 0, "DESCRICAO :", DesFile$)
    NumberFile = 2
    SizeFile = LEN(TabDir)
    CALL OpenFile(File$, NumberFile, SizeFile)
    NumberOfDir = LOF(2) / LEN(TabDir)
    CALL FIDir(Unit$, Dir$)
    Modulo1$ = UnitQB$ + DirQB$ + "\MENUPRI.DAT"
    Modulo2$ = " + " + LTRIM$(ModSet$)
    Modulo3$ = " + " + LTRIM$(ModMed$)
    Modulo4$ = " + " + LTRIM$(ModGra$)
    Modulo5$ = " + " + LTRIM$(ModPla$)
    Modulo6$ = Unit$ + Dir$ + "\" + ModPri$ + ".BAS"
    Modtemp$ = Unit$ + Dir$ + "\TEMP1.TMP"
    SCREEN 0, 7, 1, 1
    SHELL "COPY " + Modulo1$ + Modulo2$ + Modulo3$ + "  " + Modtemp$
    SHELL "COPY " + Modtemp$ + Modulo4$ + Modulo5$ + "  " + Modulo6$
    SHELL "DEL " + Unit$ + Dir$ + "\*.TMP"
    SCREEN 0, 7, 0, 0: COLOR 14, 9, 1
    FOR x = 1 TO NumberOfDir
        GET #2, x, TabDir
                                IF    RTRIM$(LTRIM$(TabDir.FileName))    =
RTRIM$(LTRIM$(Modulo6$)) THEN
            TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
            TabDir.FileName = Modulo6$
            TabDir.FileDes = DesFile$
            PUT #2, x, TabDir
            EXIT FOR
        END IF
    NEXT x
    IF x > NumberOfDir THEN
        NumberOfDir = NumberOfDir + 1
        TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
        TabDir.FileName = Modulo6$
        TabDir.FileDes = DesFile$
        PUT #2, NumberOfDir, TabDir
    END IF
    CLOSE #2
    ModPri$ = ModPri$ + ".BAS"
    RETURN

END SUB
```

NOME DA SUBROTINA: Operador

PARAMETROS: Nenhum

DESCRICAO: Inclui, exclui ou altera do dados (senha) do(s) operador(es).

```
SUB Operador
ON ERROR GOTO Roterro
NumberFile = 1
SizeFile = LEN(Senha)
File$ = "AIDSENHA.DAT"
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfSenha = LOF(1) / LEN(Senha)

REDIM OpOP$(4)
OpOP$(1) = "INSERIR"
OpOP$(2) = "ALTERAR"
OpOP$(3) = "CANCELAR"
OpOP$(4) = "XFIM"
OpOx = Achoice%(8, 7, OpOP$())

SELECT CASE OpOx
CASE 1
    CALL Addsenha
CASE 2
    CALL AltSenha
CASE 3
    CALL DelSenha(Unit$, Dir$)
END SELECT
CLOSE #1
ERASE OpOP$
END SUB
```

NOME DA SUBROTINA: Planilha
PARAMETROS: Opedi$()
DESCRICAO: Edita o módulo PLANILHA para a subrotina editor. Salvar os valores inseridos no arquivo temporário criado pelo ambiente.

```
SUB Planilha (Opedi$())
Escolha = 1
Confirme$ = SPACE$(1)

DO WHILE Escolha <> 0
    COLOR 14, 9, 1
    FOR x = 7 TO 19
        LOCATE x, 2: PRINT SPACE$(78)
    NEXT x
    Pvideo = 7
    FOR Pfile = 1 TO 4
        GET #1, Pfile, Linein(Pfile)
        LOCATE Pvideo, 2: PRINT Linein(Pfile)
        Pvideo = Pvideo + 1
    NEXT Pfile
    Pvideo = Pvideo - 1
    Pfile = Pfile - 1
    CALL Saymes(11, 5, 0, "ARQUIVO: " + NomePlani$)
    CALL Saymes(11, 40, 0, "COLUNAS: " + STR$(TipoPla))
    CALL Saymes(12, 5, 0, "PRIN.: " + TitlePri$)
    CALL Saymes(13, 5, 0, "Faixa A: " + TitleFA$)
    CALL Saymes(14, 5, 0, "Faixa B: " + TitleFB$)
    CALL Saymes(15, 5, 0, "Faixa C: " + TitleFC$)
    CALL Saymes(16, 5, 0, "Faixa D: " + TitleFD$)
    CALL Saymes(13, 40, 0, "Faixa E: " + TitleFE$)
    CALL Saymes(14, 40, 0, "Faixa F: " + TitleFF$)
    CALL Saymes(15, 40, 0, "Faixa G: " + TitleFG$)
    CALL Saymes(16, 40, 0, "Faixa H: " + TitleFH$)
    FF$ = ""
    FOR f = 1 TO LEN(Faixas$)
        FF$ = FF$ + " - " + MID$(Faixas$, f, 1)
    NEXT f
    CALL Saymes(17, 5, 0, "FAIXAS ATIVAS:" + FF$)
      Mes$  = "INICIO:  " +  STR$(StartFaixa) +  "% "  + SPACE$(23) +
"FINAL: " + STR$(StopFaixa) + "%  "
    CALL Saymes(18, 5, 0, Mes$)
    ON ERROR GOTO Roterro
    Escolha = Menu(4, 1, Opedi$())
    SELECT CASE Escolha
    CASE 1
        GOSUB coluna:
    CASE 2
        GOSUB PlArquivo:
    CASE 3
        GOSUB PlFaixa:
    CASE 4
        GOSUB PlTitulos:
    CASE 5
        GOSUB GeraPlani:
```

```
        CASE 6
            CALL SaveArq(Menux)
        CASE 7
            CALL DeleArq(Menux)
        CASE 8
            CALL RecuArq(Menux)
        CASE 9
           EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    LOOP
    EXIT SUB
    END

    '*****************************************
    '  INICIO DAS SUBROTINAS DO EDITOR DE SETUP
    '
    '-----------------------------------------------------------------
    '
    coluna:
        DIM OpPL$(8)
        FOR x = 1 TO 8
            OpPL$(x) = STR$(x) + " COLUNAS"
        NEXT x
        OpPx = Achoice%(7, 2, OpPL$())
        IF OpPx <> 0 THEN
            TipoPla = OpPx
            Mes$ = "PLANILHA DE " + STR$(OpPx) + " COLUNAS"
            CALL Saymes(14, 0, 2, Mes$)
            SLEEP 1
            Passo1 = TRUE
        ELSE
            BEEP
        END IF
        ERASE OpPL$
        RETURN
    '
    '-----------------------------------------------------------------
    PlArquivo:
        NomePlani$ = SPACE$(8)
        CALL Getmes(22, 0, 0, "NOME DO ARQUIVO : ", NomePlani$)
        IF NomePlani$ = "" THEN
            RETURN
        END IF
        IF LEN(NomePlani$) > 8 THEN
            Mes$ = "NOME DO ARQUIVO MAIOR QUE 8 CARACTERES !!!"
            CALL Mensagem(Mes$)
            RETURN
        ELSE
            SLEEP 1
            LOCATE 22, 2: PRINT SPACE$(76)
        END IF
        NomePlani$ = UCASE$(NomePlani$)
        PlaniSet$ = Unit$ + Dir$ + "\" + NomePlani$ + ".DAD"
        GrafoAtive$ = NomePlani$ + ".DAD"
        Passo2 = TRUE
        RETURN
    '
```

```
'_____________________________________________________________________
P1Faixa:
    DIM OpsetF$(11, 2)
    FOR x = 1 TO 8
        OpsetF$(x, 1) = " " + CHR$(64 + x) + " "
        OpsetF$(x, 2) = "Faixa de dados - " + CHR$(64 + x)
    NEXT x

    OpsetF$(9, 1) = "INICIO%"
    OpsetF$(9, 2) = "Inicio da faixa para o garfico - 0%"
    OpsetF$(10, 1) = "FINAL%"
    OpsetF$(10, 2) = "Final da faixa para o grafico - 100%"
    OpsetF$(11, 1) = "XFIM"
    OpsetF$(11, 2) = "FINAL DE OPERACOES"

    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT

    IF TipoPla <> 0 THEN
          Mes$ = "PLANILHA COM " + STR$(TipoPla) + " COLUNAS, FAIXAS
SEQUENCIAIS"
     ELSE
        Mes$ = "Tipo de PLANILHA nao definida !!!"
        CALL Mensagem(Mes$)
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        ERASE OpsetF$
        RETURN
    END IF
    CALL Saymes(22, 0, 0, Mes$)
    SLEEP 2
    LOCATE 22, 2: PRINT SPACE$(76)
    DO WHILE Escolha <> 0
        IF LEN(Faixas$) = TipoPla THEN
              Mes$ = "FAIXAS JA DEFINIDAS PARA " + STR$(TipoPla) + "
COLUNAS"
            CALL Mensagem(Mes$)
            EXIT DO
        END IF

        Escolha = Menu(4, 1, OpsetF$())
        SELECT CASE Escolha
        CASE 1 TO 8
            Letra$ = CHR$(64 + Escolha)
            GOSUB P1TestaLetra:
        CASE 9
            OldFaixa = StartFaixa
               LOCATE 22, 15: INPUT "Inicio da faixa do grafico (0%):
", StartFaixa
            LOCATE 22, 2: PRINT SPACE$(78)
            IF StartFaixa > StopFaixa THEN
                Mes$ = "DEFINICAO DE FAIXA INCORRETA !!!"
                CALL Mensagem(Mes$)
                StartFaixa = OldFaixa
                EXIT DO
            END IF
            IF StartFaixa < 0 THEN
                StartFaixa = 0
            END IF
                 Mes$  = "INICIO:  " +  STR$(StartFaixa) +  "%   " +
SPACE$(23) + "FINAL: " + STR$(StopFaixa) + "%  "
            CALL Saymes(18, 5, 0, Mes$)
```

```
        CASE 10
            OldFaixa = StopFaixa
              LOCATE 22, 15: INPUT "Final da faixa do grafico (100%):
", StopFaixa
            LOCATE 22, 2: PRINT SPACE$(78)
            IF StopFaixa < StartFaixa THEN
                Mes$ = "DEFINICAO DE FAIXA INCORRETA !!!"
                CALL Mensagem(Mes$)
                StopFaixa = OldFaixa
                EXIT DO
            END IF
            IF StopFaixa > 100 THEN
                StopFaixa = 100
            END IF
                Mes$  = "INICIO:  " +  STR$(StartFaixa) +  "%    " +
SPACE$(23) + "FINAL: " + STR$(StopFaixa) + "% "
            CALL Saymes(18, 5, 0, Mes$)
        CASE 11
            EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    LOOP
    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
    ERASE OpsetF$

RETURN
'
'*********************************
' sub rotina usada pela sub Faixa:
PlTestaLetra:
        FOR f = 1 TO LEN(Faixas$)
            IF MID$(Faixas$, f, 1) = Letra$ THEN
                Mes$ = "Faixa ja selecionada !!!"
                CALL Mensagem(Mes$)
                RETURN
            END IF
        NEXT f
        Faixas$ = Faixas$ + Letra$
        FF$ = ""
        FOR f = 1 TO LEN(Faixas$)
             FF$ = FF$ + " - " + MID$(Faixas$, f, 1)
        NEXT f
        CALL Saymes(17, 5, 0, "FAIXAS ATIVAS:" + FF$)
        Passo3 = TRUE
        RETURN
'-------------------------------------------------------------
PlTitulos:
    DIM Opset4$(10, 2)
    Opset4$(1, 1) = "PRINCIPAL"
    Opset4$(1, 2) = "Titulo principal do grafico"
    FOR x = 2 TO 9
        Opset4$(x, 1) = " " + CHR$(63 + x) + " "
        Opset4$(x, 2) = "Faixa de dados - " + CHR$(63 + x)
    NEXT x
    Opset4$(10, 1) = "XFIM"
    Opset4$(10, 2) = "FINAL DE OPERACOES"

    VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
```

```
DO WHILE Escolha <> 0
    Escolha = Menu(4, 1, Opset4$())

    IF Escolha >= 2 AND Escolha <= 9 THEN
        Letra$ = CHR$(63 + Escolha)
        FlagoFaixa = 0
        GOSUB TestaFaixa:
        IF FlagoFaixa = 1 THEN
            Cmd$ = ""
            EXIT DO
        END IF
        Sintax$ = "Titulo Faixa de dados " + Letra$ + ":"
        Cmd$ = ""
        CALL Command(Sintax$, Cmd$)
        GOSUB SizeCmd:
    END IF
    SELECT CASE Escolha
    CASE 1
        Sintax$ = "Titulo Principal do Grafico :"
        Cmd$ = ""
        CALL Command(Sintax$, Cmd$)
        TitlePri$ = Cmd$
        CALL Saymes(12, 5, 0, "PRIN.: " + Cmd$)
    CASE 2
        TitleFA$ = Cmd$
        CALL Saymes(13, 5, 0, "Faixa A: " + Cmd$)
    CASE 3
        TitleFB$ = Cmd$
        CALL Saymes(14, 5, 0, "Faixa B: " + Cmd$)
    CASE 4
        TitleFC$ = Cmd$
        CALL Saymes(15, 5, 0, "Faixa C: " + Cmd$)
    CASE 5
        TitleFD$ = Cmd$
        CALL Saymes(16, 5, 0, "Faixa D: " + Cmd$)
    CASE 6
        TitleFE$ = Cmd$
        CALL Saymes(13, 40, 0, "Faixa E: " + Cmd$)
    CASE 7
        TitleFF$ = Cmd$
        CALL Saymes(14, 40, 0, "Faixa F: " + Cmd$)
    CASE 8
        TitleFG$ = Cmd$
        CALL Saymes(15, 40, 0, "Faixa G: " + Cmd$)
    CASE 9
        TitleFH$ = Cmd$
        CALL Saymes(16, 40, 0, "Faixa H: " + Cmd$)
    CASE 10
        EXIT DO
    CASE ELSE
        BEEP
    END SELECT
LOOP
ERASE Opset4$
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
RETURN
```

```
'_____________________________________________________________
TestaFaixa:
        IF LEN(Faixas$) <> 0 THEN
            FOR f = 1 TO LEN(Faixas$)
                IF MID$(Faixas$, f, 1) = Letra$ THEN
                    RETURN
                END IF
            NEXT f
            Mes$ = "Faixas nao selecionadas !!!"
            CALL Mensagem(Mes$)
            FlagoFaixa = 1
        END IF
        RETURN


'_____________________________________________________________
SizeCmd:
    IF LEN(Cmd$) > (80 / TipoPla) - 1 THEN
         Mes$ = "Nome da Faixa Maior que " + STR$((80 / TipoPla) - 1)
+ " caracteres !"
        CALL Saymes(22, 0, 0, Mes$)
        SLEEP 2
        LOCATE 22, 2: PRINT SPACE$(76)
        Cmd$ = MID$(Cmd$, 1, (80 / TipoPla) - 1)
    END IF
    RETURN

'
'
'_____________________________________________________________
GeraPlani:
    File$ = "DIRPLM.DAT"
    NumberFile = 2
    SizeFile = LEN(TabDir)
    CALL OpenFile(File$, NumberFile, SizeFile)
    NumberOfDir = LOF(2) / LEN(TabDir)
    IF NumberOfDir = 0 THEN
        CALL DirZero
        RETURN
    END IF
    Mes$ = "ESCOLHA O MODULO PARA GERAR A PLANILHA !!"
    CALL Saymes(22, 0, 0, Mes$)
    CALL MostraDir(PointerDir)
    IF PointerDir <> 0 THEN
        CALL Getmes(22, 0, 0, "DESCRICAO :", DesFile$)
        Mes$ = "Confirme o modulo (S/N) : " + TabDir.FileName + "  "
        CALL Getmes(22, 0, 0, Mes$, Confirme$)
        IF UCASE$(Confirme$) = "S" THEN
            CALL Getmes(22, 0, 0, "DESCRICAO :", DesFile$)
            File$ = RTRIM$(TabDir.FileName)
            File$ = MID$(File$, 1, LEN(File$) - 4)

            Modulo1$ = " + " + Unit$ + Dir$ + "\PLANI2.DAT   "
            Modulo2$ = File$ + ".PLA"
            SCREEN 0, 7, 1, 1
            SHELL "COPY " + TabDir.FileName + Modulo1$ + Modulo2$
            SCREEN 0, 7, 0, 0
            CLOSE #2
            File$ = "DIRPLA.DAT"
            NumberFile = 2
            SizeFile = LEN(TabDir)
            CALL OpenFile(File$, NumberFile, SizeFile)
```

```
                NumberOfDir = LOF(2) / LEN(TabDir)
                FOR x = 1 TO NumberOfDir
                    GET #2, x, TabDir
                        IF RTRIM$(LTRIM$(TabDir.FileName))=RTRIM$(LTRIM$
(Modulo2$)) THEN
                        TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
                        TabDir.FileName = Modulo2$
                        TabDir.FileDes = DesFile$
                        PUT #2, x, TabDir
                        EXIT FOR
                    END IF
                NEXT x
                IF x > NumberOfDir THEN
                    NumberOfDir = NumberOfDir + 1
                    TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
                    TabDir.FileName = Modulo2$
                    TabDir.FileDes = DesFile$
                    PUT #2, NumberOfDir, TabDir
                END IF
                CLOSE #2
            END IF
        END IF
        COLOR 14, 9, 1
        LOCATE 22, 2: PRINT SPACE$(76)
        CLOSE #2
        RETURN
'
'-------------------------------------------------------------------
END SUB
```

NOME DA SUBROTINA: RecuArq
PARAMETROS: Menux%
DESCRICAO: Recupera arquivos e módulos já salvos no disco de trabalho, atualiza os dados no editor selecionado.

```
SUB RecuArq (Menux)
Confirme$ = SPACE$(1)
SELECT CASE Menux
CASE 1
    File$ = "DIRSET.DAT"
    TempFile$ = "SETUP.$$$"
CASE 2
    File$ = "DIRMED.DAT"
    TempFile$ = "MEDE.$$$"
CASE 3
    File$ = "DIRGRM.DAT"
    TempFile$ = "GRAFO.$$$"
CASE 4
    File$ = "DIRPLM.DAT"
    TempFile$ = "PLANI.$$$"
END SELECT

NumberFile = 2
SizeFile = LEN(TabDir)
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfDir = LOF(2) / LEN(TabDir)
```

```
IF NumberOfDir = 0 THEN
    CALL DirZero
    EXIT SUB
END IF
Mes$ = "ESCOLHA O ARQUIVO PARA REINSERIR !!"
CALL Saymes(22, 0, 0, Mes$)
CALL MostraDir(PointerDir)
CLOSE #2
IF PointerDir <> 0 THEN
    Mes$ = "Confirme o arquivo (S/N) : " + TabDir.FileName + "  "
    CALL Getmes(22, 0, 0, Mes$, Confirme$)
    IF UCASE$(Confirme$) = "S" THEN
        IF RIGHT$(FileAtive$, 3) <> "$$$" THEN
            Mes$ = "Executando rotina de Backup de Arquivo !!"
            CALL Saymes(22, 0, 0, Mes$)
            SLEEP 1
            LOCATE 22, 2: PRINT SPACE$(76)
            CALL SaveArq(Menux)
        END IF
        CLOSE #1
        NumberFile = 10
        SizeFile = LEN(Linex)
        File$ = RTRIM$(TabDir.FileName)
        DesFile$ = TabDir.FileDes
        CALL FindBar(File$)
        FileAtive$ = File$
        CALL OpenSeqFile(File$, NumberFile, SizeFile)
        KILL Unit$ + Dir$ + "\" + TempFile$
        NumberFile = 1
        SizeFile = LEN(Linex)
        File$ = TempFile$
        CALL OpenFile(File$, NumberFile, SizeFile)
        Pline = 1
        DO WHILE NOT EOF(10)
            LINE INPUT #10, Linein(Pline)
            PUT #1, Pline, Linein(Pline)
            Pline = Pline + 1
        LOOP
        CLOSE #10
        Pfile = 0
        IF Menux = 3 THEN
            GET #1, 5, Linein(5)
            NomeGrafo$ = MID$(Linein(5), 15, LEN(Linein(5)) - 15)
            GET #1, 6, Linein(6)
            TipoGrafo = VAL(MID$(Linein(6), 13, LEN(Linein(6))))
            GET #1, 7, Linein(7)
            TitlePri$ = MID$(Linein(7), 14, LEN(Linein(7)) - 14)
            GET #1, 8, Linein(8)
            TitleSec$ = MID$(Linein(8), 14, LEN(Linein(8)) - 14)
            GET #1, 9, Linein(9)
            TitleX$ = MID$(Linein(9), 12, LEN(Linein(9)) - 12)
            GET #1, 10, Linein(10)
            TitleY$ = MID$(Linein(10), 12, LEN(Linein(10)) - 12)
            GET #1, 11, Linein(11)
            StartFaixa = VAL(MID$(Linein(11), 14, LEN(Linein(11))))
            GET #1, 12, Linein(12)
            StopFaixa = VAL(MID$(Linein(12), 13, LEN(Linein(12))))
            GET #1, 13, Linein(13)
            Faixas$ = MID$(Linein(13), 12, LEN(Linein(13)) - 12)
            SELECT CASE TipoGrafo
            CASE 1
                TipoGra$ = "X - t"
```

```
            CASE 2
                TipoGra$ = "X - Y"
            END SELECT
        END IF
        IF Menux = 4 THEN
            GET #1, 5, Linein(5)
            NomePlani$ = MID$(Linein(5), 15, LEN(Linein(5)) - 15)
            GET #1, 6, Linein(6)
            TipoPla = VAL(MID$(Linein(6), 11, LEN(Linein(6))))
            GET #1, 7, Linein(7)
            TitlePri$ = MID$(Linein(7), 14, LEN(Linein(7)) - 14)
            GET #1, 8, Linein(8)
            TitleFA$ = MID$(Linein(8), 13, LEN(Linein(8)) - 13)
            GET #1, 9, Linein(9)
            TitleFB$ = MID$(Linein(9), 13, LEN(Linein(9)) - 13)
            GET #1, 10, Linein(10)
            TitleFC$ = MID$(Linein(10), 13, LEN(Linein(10)) - 13)
            GET #1, 11, Linein(11)
            TitleFD$ = MID$(Linein(11), 13, LEN(Linein(11)) - 13)
            GET #1, 12, Linein(12)
            TitleFE$ = MID$(Linein(12), 13, LEN(Linein(12)) - 13)
            GET #1, 13, Linein(13)
            TitleFF$ = MID$(Linein(13), 13, LEN(Linein(13)) - 13)
            GET #1, 14, Linein(14)
            TitleFG$ = MID$(Linein(14), 13, LEN(Linein(14)) - 13)
            GET #1, 15, Linein(15)
            TitleFH$ = MID$(Linein(15), 13, LEN(Linein(15)) - 13)
            GET #1, 16, Linein(16)
            StartFaixa = VAL(MID$(Linein(16), 14, LEN(Linein(16))))
            GET #1, 17, Linein(17)
            StopFaixa = VAL(MID$(Linein(17), 13, LEN(Linein(17))))
            GET #1, 18, Linein(18)
            Faixas$ = MID$(Linein(18), 12, LEN(Linein(18)) - 12)
        END IF
    ELSE
        BEEP
    END IF
ELSE
    LOCATE 22, 2: PRINT SPACE$(76)
END IF
FOR x = 7 TO 19
    LOCATE x, 2: PRINT SPACE$(78)
NEXT x

END SUB
```

NOME DA SUBROTINA: SaveArq

PARAMETROS: Menux%

DESCRICAO: Salva os módulos editados no disco de trabalho. Inclui o nome e descrição do módulo na lista de módulos editados pelo ambiente.

```
SUB SaveArq (Menux%)
SELECT CASE Menux
CASE 1
    File$ = "DIRSET.DAT"
CASE 2
    File$ = "DIRMED.DAT"
CASE 3
    File$ = "DIRGRM.DAT"
CASE 4
    File$ = "DIRPLM.DAT"
END SELECT
NumberFile = 2
SizeFile = LEN(TabDir)
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfDir = LOF(2) / LEN(TabDir)
IF RIGHT$(FileAtive$, 3) = "$$$" THEN
    CALL Getmes(22, 0, 0, "NOME DO ARQUIVO : ", File$)
    IF File$ = "" THEN
        CLOSE #2
        EXIT SUB
    END IF
    IF LEN(File$) > 8 THEN
        Mes$ = "NOME DO ARQUIVO MAIOR QUE 8 CARACTERES !!!"
        CALL Mensagem(Mes$)
        CLOSE #2
        EXIT SUB
    END IF
    File$ = UCASE$(File$)
    SELECT CASE Menux
    CASE 1
        FileSet$ = Unit$ + Dir$ + "\" + File$ + ".SET"
        File$ = File$ + ".SET"
    CASE 2
        FileSet$ = Unit$ + Dir$ + "\" + File$ + ".MED"
        File$ = File$ + ".MED"
    CASE 3
        FileSet$ = Unit$ + Dir$ + "\" + File$ + ".GRM"
        File$ = File$ + ".GRM"
    CASE 4
        FileSet$ = Unit$ + Dir$ + "\" + File$ + ".PLM"
        File$ = File$ + ".PLM"
    END SELECT
    Mes$ = "DESCRICAO :"
    DesFile$ = SPACE$(35)
    CALL Getmes(22, 0, 0, Mes$, DesFile$)
```

```
     IF LEN(DesFile$) > 35 THEN
          Mes$ = "DESCRICAO DO ARQUIVO MAIOR QUE 35 CARACTERES !!!"
          CALL Mensagem(Mes$)
          CLOSE #2
          EXIT SUB
     END IF
     FOR x = 1 TO NumberOfDir
          GET #2, x, TabDir
               IF RTRIM$(LTRIM$(TabDir.FileName))=RTRIM$(LTRIM$(FileSet
$)) THEN
                    Mes$ = "Arquivo ja' existente, Confirme? (S/N) : " +
TabDir.FileName + "  "
               CALL Getmes(22, 0, 0, Mes$, Confirme$)
               IF UCASE$(Confirme$) <> "S" THEN
                    CLOSE #2
                    EXIT SUB
               END IF
               TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
               TabDir.FileName = FileSet$
               TabDir.FileDes = DesFile$
               PUT #2, x, TabDir
               EXIT FOR
          END IF
     NEXT x
     IF x > NumberOfDir THEN
          NumberOfDir = NumberOfDir + 1
          TabDir.OperatorName = RTRIM$(LTRIM$(OpName$))
          TabDir.FileName = FileSet$
          TabDir.FileDes = DesFile$
          PUT #2, NumberOfDir, TabDir
     END IF
     FileAtive$ = RTRIM$(File$)
     FileBak$ = MID$(FileAtive$, 1, LEN(FileAtive$) - 4)
     FileBak$ = UCASE$(FileBak$) + ".BAK"

ELSE
     File$ = FileAtive$
     DesFile$ = TabDir.FileDes
     CALL Saymes(22, 0, 0, "NOME DO ARQUIVO : " + File$)
     SLEEP 2
     LOCATE 22, 2: PRINT SPACE$(76)
     FileSet$ = Unit$ + Dir$ + "\" + File$
     Mes$ = "DESCRICAO : "
     CALL Saymes(22, 0, 0, Mes$ + DesFile$)
     SLEEP 2
     LOCATE 22, 2: PRINT SPACE$(76)
     FileAtive$ = RTRIM$(FileAtive$)
     FileBak$ = MID$(FileAtive$, 1, LEN(FileAtive$) - 4)
     FileBak$ = UCASE$(FileBak$) + ".BAK"
END IF
CLOSE #2
FOR x = 7 TO 19
     LOCATE x, 2: PRINT SPACE$(78)
NEXT x

SELECT CASE Menux
CASE 1, 2
        Linein(1)  =  "'**************  ARQUIVO  TEMPORARIO  DE  "  +
ModuloAtive$ + " ********************"
      Linein(2)  = "'Nome  do  Operador=  "  +  RTRIM$(OpName$)  +  "
Data: " + DATE$ + "        Hora: " + TIME$
     Linein(3) = "'Nome do Arquivo = " + FileAtive$
```

```
    Linein(4) = ModuloAtive$
    FOR x = 1 TO 4
        PUT #1, x, Linein(x)
    NEXT x
CASE 3
      Linein(1)  = "'****************************  MODULO  DE  GRAFICO
********************"
      Linein(2)  = "'Nome  do  Operador=  "  +  RTRIM$(OpName$)  +  "
Data: " + DATE$ + "       Hora: " + TIME$
    Linein(3) = "'Nome do Arquivo = " + FileAtive$
    Linein(4) = "GRAFICO:"
    Linein(5) = "NomeGrafo$ = " + CHR$(34) + NomeGrafo$ + CHR$(34)
    Linein(6) = "TipoGrafo = " + STR$(TipoGrafo)
    Linein(7) = "TiTlePri$ = " + CHR$(34) + TitlePri$ + CHR$(34)
    Linein(8) = "TiTleSec$ = " + CHR$(34) + TitleSec$ + CHR$(34)
    Linein(9) = "TiTleX$ = " + CHR$(34) + TitleX$ + CHR$(34)
    Linein(10) = "TiTleY$ = " + CHR$(34) + TitleY$ + CHR$(34)
    Linein(11) = "StartFaixa = " + STR$(StartFaixa)
    Linein(12) = "StopFaixa = " + STR$(StopFaixa)
    Linein(13) = "Faixas$ = " + CHR$(34) + Faixas$ + CHR$(34)
    FOR x = 1 TO 13
        PUT #1, x, Linein(x)
    NEXT x
CASE 4
      Linein(1)  = "'****************************  MODULO DE PLANILHA
********************"
      Linein(2)  = "'Nome  do  Operador=  "  +  RTRIM$(OpName$)  +  "
Data: " + DATE$ + "       Hora: " + TIME$
    Linein(3) = "'Nome do Arquivo = " + FileAtive$
    Linein(4) = "PLANILHA:"
    Linein(5) = "NomePlani$ = " + CHR$(34) + NomePlani$ + CHR$(34)
    Linein(6) = "Colunas = " + STR$(TipoPla)
    Linein(7) = "TiTlePri$ = " + CHR$(34) + TitlePri$ + CHR$(34)
    Linein(8) = "TitleFA$ = " + CHR$(34) + TitleFA$ + CHR$(34)
    Linein(9) = "TitleFB$ = " + CHR$(34) + TitleFB$ + CHR$(34)
    Linein(10) = "TitleFC$ = " + CHR$(34) + TitleFC$ + CHR$(34)
    Linein(11) = "TitleFD$ = " + CHR$(34) + TitleFD$ + CHR$(34)
    Linein(12) = "TitleFE$ = " + CHR$(34) + TitleFE$ + CHR$(34)
    Linein(13) = "TitleFF$ = " + CHR$(34) + TitleFF$ + CHR$(34)
    Linein(14) = "TitleFG$ = " + CHR$(34) + TitleFG$ + CHR$(34)
    Linein(15) = "TitleFH$ = " + CHR$(34) + TitleFH$ + CHR$(34)
    Linein(16) = "StartFaixa = " + STR$(StartFaixa)
    Linein(17) = "StopFaixa = " + STR$(StopFaixa)
    Linein(18) = "Faixas$ = " + CHR$(34) + Faixas$ + CHR$(34)
    FOR x = 1 TO 18
        PUT #1, x, Linein(x)
    NEXT x
END SELECT

File$ = FileAtive$
NumberFile = 2
SizeFile = LEN(Linex)
CALL OpenSeqFile(File$, NumberFile, SizeFile)
NumberOfLines = LOF(1) / LEN(Linex)
FOR x = 1 TO NumberOfLines + 1
    GET #1, x, Linein(x)
    PRINT #2, Linein(x)
NEXT x
CLOSE #2
Pfile = 0
SCREEN 0, 7, 0, 1: COLOR 14, 9, 1
SHELL "copy " + FileAtive$ + "  " + FileBak$
```

```
SHELL "cls"
SCREEN 0, 7, 0, 0: COLOR 7, 9, 1
END SUB
```

NOME DA SUBROTINA: SetMed
PARAMETROS: Menux%,Opedi$()
DESCRICAO: Ativa os editores de 'SETUP' e MEDIDAS. Cria módulo temporário de acordo com a opção selecionada. Possue o mesmo tipo de pop-menus e linhas de comando.

```
SUB SetMed (Menux%, Opedi$())
Pfile = 0
Escolha = 1
FlagSub = FALSE
Flago = 0
DO WHILE Escolha <> 0
     Pvideo = 7
     IF Pfile = 0 THEN
          Pfile = 1
     ELSE
          Pfile = Pline - 12
     END IF
     IF Pfile < 0 THEN
          Pfile = 1
     END IF
     GET #1, Pfile, Linein(Pfile)
     DO WHILE NOT EOF(1)
               IF Pvideo > 19 THEN
                    EXIT DO
               END IF
               GET #1, Pfile, Linein(Pfile)
               LOCATE Pvideo, 2: PRINT Linein(Pfile)
               Pfile = Pfile + 1
               Pvideo = Pvideo + 1
     LOOP
     Pvideo = Pvideo - 1
     Pfile = Pfile - 1
     ON ERROR GOTO Roterro
     IF FlagSub = FALSE THEN
          Escolha = Menu(4, 1, Opedi$())
          SELECT CASE Escolha
          CASE 1
               GOSUB Grupo1:
          CASE 2
               GOSUB Grupo2:
          CASE 3
               GOSUB Grupo3:
          CASE 4
               GOSUB Qbasic:
          CASE 5
               CALL SaveArq(Menux)
          CASE 6
               CALL DeleArq(Menux)
          CASE 7
               CALL RecuArq(Menux)
```

```
        CASE 8
            EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    ELSE
        ON Flago GOSUB Grupo1, Grupo2, Grupo3, Qbasic
    END IF
LOOP
EXIT SUB
END

'****************************************
'  INICIO DAS SUBROTINAS DO EDITOR DE SETUP
'
'----------------------------------------------------------------
Grupo1:
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT

DIM Opset1$(11, 2)
Opset1$(1, 1) = "IBREAD"
Opset1$(1, 2) = "Executa leitura em um Instrumento"
Opset1$(2, 1) = "IBWHITE"
Opset1$(2, 2) = "Executa escrita em um Instrumento"
Opset1$(3, 1) = "IBSPOOL"
Opset1$(3, 2) = "Executa um Serial Poll em um Instrumento"
Opset1$(4, 1) = "IBCLEAR"
Opset1$(4, 2) = "Executa a inicializacao de um Instrumento"
Opset1$(5, 1) = "IBFIND"
Opset1$(5, 2) = "Espera por um evento pre-definido"
Opset1$(6, 1) = "EDITA"
Opset1$(6, 2) = "Edita uma linha de comando"
Opset1$(7, 1) = "<"
Opset1$(7, 2) = "Retrocede um grupo"
Opset1$(8, 1) = ">"
Opset1$(8, 2) = "Avanca um grupo"
Opset1$(9, 1) = "/\"
Opset1$(9, 2) = "Sobe um linha na tela"
Opset1$(10, 1) = "\/"
Opset1$(10, 2) = "Desce uma linha na tela"
Opset1$(11, 1) = "XFIM"
Opset1$(11, 2) = "FINAL DE OPERACOES"
DO WHILE Escolha <> 0
    ON ERROR GOTO Roterro
    Escolha = Menu(4, 1, Opset1$())
    SELECT CASE Escolha
    CASE 1
        Syntax$ = "Sintaxe: CALL IBRD (DEVICE%, VARIABLE$)"
        Cmd$ = "CALL IBRD("
        CALL Command(Syntax$, Cmd$)
    CASE 2
        Syntax$ = "Sintaxe: CALL IBWRT (DEVICE%, VARIABLE$)"
        Cmd$ = "CALL IBWRT("
        CALL Command(Syntax$, Cmd$)
    CASE 3
        Syntax$ = "Sintaxe: CALL IBRSP (DEVICE%, INTEGER%)"
        Cmd$ = "CALL IBRSP("
        CALL Command(Syntax$, Cmd$)
    CASE 4
        Syntax$ = "Sintaxe: CALL IBCLR (DEVICE%)"
        Cmd$ = "CALL IBCLR("
        CALL Command(Syntax$, Cmd$)
```

```
    CASE 5
        Syntax$ = "Sintaxe: CALL IBFIND(BOARD$,DEVICE%)"
        Cmd$ = "CALL IBFIND("
        CALL Command(Syntax$, Cmd$)
    CASE 6
        Cmd$ = SPACE$(4) + Cmd$
        CALL EditaCmd(Cmd$, Pline, Pfile, Pvideo)
    CASE 7
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        FlagSub = TRUE: Flago = 4: EXIT DO
    CASE 8
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        FlagSub = TRUE: Flago = 2: EXIT DO
    CASE 9          'SOBE
        CALL LineUp
    CASE 10      'DESCE
        CALL LIneDown
    CASE 11
        FlagSub = FALSE
        EXIT DO
    CASE ELSE
        BEEP
    END SELECT
LOOP
ERASE Opset1$
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
RETURN
'-------------------------------------------------------------------
Grupo2:
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
DIM Opset2$(11, 2)
Opset2$(1, 1) = "IBIFC"
Opset2$(1, 2) = "Executa a inicializacao da interface"
Opset2$(2, 1) = "IBTRIG"
Opset2$(2, 2) = "Executa uam sequencial de Trigger em um
Instrumento"
Opset2$(3, 1) = "IBLOC"
Opset2$(3, 2) = "Coloca o Instrumento em modo local"
Opset2$(4, 1) = "IBSRE"
Opset2$(4, 2) = "Coloca o Instrumento em modo remoto"
Opset2$(5, 1) = "IBWAIT"
Opset2$(5, 2) = "Espera por um evento pre-definido"
Opset2$(6, 1) = "EDITA"
Opset2$(6, 2) = "Edita uma linha de comando"
Opset2$(7, 1) = "<"
Opset2$(7, 2) = "Retrocede um grupo"
Opset2$(8, 1) = ">"
Opset2$(8, 2) = "Avanca um grupo"
Opset2$(9, 1) = "/\"
Opset2$(9, 2) = "Sobe um linha na tela"
Opset2$(10, 1) = "\/"
Opset2$(10, 2) = "Desce uma linha na tela"
Opset2$(11, 1) = "XFIM"
Opset2$(11, 2) = "FINAL DE OPERACOES"
```

```
DO WHILE Escolha <> 0
     ON ERROR GOTO Roterro
     Escolha = Menu(4, 1, Opset2$())
     SELECT CASE Escolha
     CASE 1
          Syntax$ = "Sintaxe: CALL IBCLR (DEVICE% or BOARD%)"
          Cmd$ = "CALL IBCLR("
          CALL Command(Syntax$, Cmd$)
     CASE 2
          Syntax$ = "Sintaxe: CALL IBTRG (DEVICE%)"
          Cmd$ = "CALL IBTRG("
          CALL Command(Syntax$, Cmd$)
     CASE 3
          Syntax$ = "Sintaxe: CALL IBLOC (DEVICE%)"
          Cmd$ = "CALL IBLOC("
          CALL Command(Syntax$, Cmd$)
     CASE 4
          Syntax$ = "Sintaxe: CALL IBSRE (DEVICE%, VARIABLE%)"
          Cmd$ = "CALL IBSRE("
          CALL Command(Syntax$, Cmd$)
     CASE 5
          Syntax$ = "Sintaxe:     CALL     IBWAIT     (BOARD%,     MASK%)
MASK%=&H1000"
          Cmd$ = "CALL IBWAIT("
          CALL Command(Syntax$, Cmd$)
     CASE 6
          Cmd$ = SPACE$(4) + Cmd$
          CALL EditaCmd(Cmd$, Pline, Pfile, Pvideo)
     CASE 7
          VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
          FlagSub = TRUE: Flago = 1: EXIT DO
     CASE 8
          VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
          FlagSub = TRUE: Flago = 3: EXIT DO
     CASE 9
          CALL LineUp
     CASE 10
          CALL LIneDown
     CASE 11
          FlagSub = FALSE
          EXIT DO
     CASE ELSE
          BEEP
     END SELECT
LOOP
ERASE Opset2$
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
RETURN
'
'_____________________________________________________
Grupo3:
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT

DIM Opset3$(11, 2)

Opset3$(1, 1) = "PPOLL"
Opset3$(1, 2) = "Executa um Parall Poll nos Instrumentos"
Opset3$(2, 1) = "CPOOL"
Opset3$(2, 2) = "Configura para um Parall Poll"
Opset3$(3, 1) = "UPOOL"
Opset3$(3, 2) = "Desconfigura os Instrumentos apos um Parall Poll"
```

```
Opset3$(4, 1) = "LOCKOUT"
Opset3$(4, 2) = "Inibe painel frontal do Instrumento"
Opset3$(5, 1) = "SENDCMD"
Opset3$(5, 2) = "Envia comandos do GPIB/IEEE488 para os
Instrumentos"
Opset3$(6, 1) = "EDITA"
Opset3$(6, 2) = "Edita uma linha de comando"
Opset3$(7, 1) = "<"
Opset3$(7, 2) = "Retrocede um grupo"
Opset3$(8, 1) = ">"
Opset3$(8, 2) = "Avanca um grupo"
Opset3$(9, 1) = "/\"
Opset3$(9, 2) = "Sobe um linha na tela"
Opset3$(10, 1) = "\/"
Opset3$(10, 2) = "Desce uma linha na tela"
Opset3$(11, 1) = "XFIM"
Opset3$(11, 2) = "FINAL DE OPERACOES"

DO WHILE Escolha <> 0
    ON ERROR GOTO Roterro
    Escolha = Menu(4, 1, Opset3$())
    SELECT CASE Escolha
    CASE 1
        Syntax$ = "Sintaxe: CALL IBRPP (DEVICE%, VARIABLE%)"
        Cmd$ = "CALL IBRPP("
        CALL Command(Syntax$, Cmd$)
    CASE 2
        Syntax$ = "Sintaxe: CALL IBPPC (DEVICE%, 1)"
        Cmd$ = "CALL IBPPC("
        CALL Command(Syntax$, Cmd$)
    CASE 3
        Syntax$ = "Sintaxe: CALL IBPPC (DEVICE%, 3)"
        Cmd$ = "CALL IBPPC("
        CALL Command(Syntax$, Cmd$)
    CASE 4
        Syntax$ = "Sintaxe: CALL IBCMD (DEVICE%, LOCKOUT$)"
        Cmd$ = "CALL IBCMD("
        CALL Command(Syntax$, Cmd$)
    CASE 5
        Syntax$ = "Sintaxe: CALL IBCMD (DEVICE%, COMMAND$)"
        Cmd$ = "CALL IBCMD("
        CALL Command(Syntax$, Cmd$)
    CASE 6
        Cmd$ = SPACE$(4) + Cmd$
        CALL EditaCmd(Cmd$, Pline, Pfile, Pvideo)
    CASE 7
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        FlagSub = TRUE: Flago = 2: EXIT DO
    CASE 8
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        FlagSub = TRUE: Flago = 4: EXIT DO
    CASE 9
        CALL LineUp
    CASE 10
        CALL LIneDown
    CASE 11
        FlagSub = FALSE
        EXIT DO
    CASE ELSE
        BEEP
    END SELECT
LOOP
```

```
ERASE Opset3$
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
RETURN
'
'_____________________________________________________________________
Qbasic:

VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
DIM Opset4$(8, 2)
Opset4$(1, 1) = "QBASIC"
Opset4$(1, 2) = "Executa Comandos e funcoes do QuickBASIC"
Opset4$(2, 1) = "EDITA"
Opset4$(2, 2) = "Edita uma linha de comando"
Opset4$(3, 1) = "AJUDA"
Opset4$(3, 2) = "Ajuda para Comandos e Funcoes do QuickBASIC"
Opset4$(4, 1) = "<"
Opset4$(4, 2) = "Retrocede um grupo"
Opset4$(5, 1) = ">"
Opset4$(5, 2) = "Avanca um grupo"
Opset4$(6, 1) = "/\"
Opset4$(6, 2) = "Sobe um linha na tela"
Opset4$(7, 1) = "\/"
Opset4$(7, 2) = "Desce uma linha na tela"
Opset4$(8, 1) = "XFIM"
Opset4$(8, 2) = "FINAL DE OPERACOES"

DO WHILE Escolha <> 0
    ON ERROR GOTO Roterro
    Escolha = Menu(4, 1, Opset4$())
    SELECT CASE Escolha
    CASE 1
        Syntax$ = "Sintaxe: SINTAXE DOS COMANDOS E FUNCOES DO
QuickBASIC"
        Cmd$ = ""
        CALL Command(Syntax$, Cmd$)
    CASE 2
        Cmd$ = SPACE$(4) + Cmd$
        CALL EditaCmd(Cmd$, Pline, Pfile, Pvideo)
    CASE 3
        Mes$ = "MENU DE AJUDA NAO IMPLEMENTADO !!!"
        CALL Saymes(22, 0, 0, Mes$)
        SLEEP 2
        LOCATE 22, 2: PRINT SPACE$(76)
    CASE 4
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        FlagSub = TRUE: Flago = 3: EXIT DO
    CASE 5
        VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
        FlagSub = TRUE: Flago = 1: EXIT DO
    CASE 6
        CALL LineUp
    CASE 7
        CALL LIneDown
    CASE 8
        FlagSub = FALSE
        EXIT DO
    CASE ELSE
        BEEP
    END SELECT
LOOP
ERASE Opset4$
VIEW PRINT 4 TO 5: COLOR 14, 9, 1: CLS : VIEW PRINT
```

```
RETURN
END SUB
```

NOME DA SUBROTINA: Unidade
PARAMETROS: Unit$, Dir$
DESCRICAO: Define a unidade ativa e o diretório do ambiente e de trabalho. lista diretório da unidade selecionada.

```
SUB Unidade (Unit$, Dir$)
DEFSNG A-Z
ON ERROR GOTO Roterro
CALL FIDir(Unit$, Dir$)
Mes$ = "UNIDADE ATIVA --> " + Unit$ + Dir$
CALL Saymes(16, 0, 0, Mes$)
Mes$ = "UNIDADE DE TRABALHO (A:,B:,C: .. = "
DLetter$ = SPACE$(15)
CALL Getmes(12, 0, 1, Mes$, DLetter$)
Ferro = FALSE
SCREEN 0, 7, 0, 1: COLOR 14, 9, 1
SHELL "DIR " + DLetter$
SCREEN 0, 7, 0, 0: COLOR 7, 9, 1
IF Ferro = TRUE THEN
    Ferro = FALSE
    EXIT SUB
END IF
DriveLetter$ = DLetter$
IF DriveLetter$ <> "" THEN
    IF SetDefaultDri(InRegs, OutRegs, DriveLetter$) = 0 THEN
        BEEP
        Mes$ = "NAO HOUVE ALTERACAO NA UNIDADE !!"
        CALL Mensagem(Mes$)
    ELSE
        IF LEN(DriveLetter$) > 2 THEN
            CHDIR MID$(DriveLetter$, 3, LEN(DriveLetter$) - 2)
        END IF
        CALL FIDir(Unit$, Dir$)
        Mes$ = "NOVA UNIDADE E DIRETORIO --> " + Unit$ + Dir$
        CALL Mensagem(Mes$)
    END IF
ELSE
    BEEP
END IF
END SUB
```

ANEXO - 02

EXEMPLO DE APLICAÇAO:

PROGRAMA DE MEDIDA DE DESVIO

AIDS_TME - Versão 91.01

Neste anexo é apresentado o arquivo de teste gerado pelo ambiente e as subrotinas que o compoem. As funçÕes e subrotinas não listadas aqui fazem parte da biblioteca de funçÕes especiais, desenvolvidas exclusivamente para o programa principal, descritas no ANEXO 03 e contidas no arquivo 'TOOLBOX.LIB' do disco de instalação. (28-30)

PROGRAMA EXEMPLO:

```
REM DEFINICOES DE VARIAVEIS E CONSTANTES
DEFINT A-Z
CONST false = 0, True = NOT false
'=============================================================================
REM $INCLUDE: 'qbdecl4.bas'
' inclusao das declaracoes do GPIB
'
'=============================================================================
'
REM DEFINICAO DAS ESTRUTURAS DOS ARQUIVOS
TYPE SenhaType
     OperatorName AS STRING * 15
     OperatorSenha AS DOUBLE
END TYPE
'
TYPE FileDirType
     OperatorName AS STRING * 15
     FileName AS STRING * 25
     FileDes AS STRING * 35
END TYPE
'
TYPE Regtypex
     ax AS INTEGER
     bx AS INTEGER
     cx AS INTEGER
     dx AS INTEGER
     bp AS INTEGER
     si AS INTEGER
     di AS INTEGER
     flags AS INTEGER
     ds AS INTEGER
     es AS INTEGER
END TYPE
'
TYPE FileSetup
     LineSet AS STRING * 128
END TYPE
```

```
'*************************************************************************
REM DIMENSIONAMENTO DAS VARIAVEIS DO SISTEMA
'
'==================  MODULO DO EDITOR DE SENHA  ==============
'
DIM SHARED SenhaRecord AS SenhaType
COMMON SHARED NumberOfSenha AS INTEGER
DIM SHARED Senha AS SenhaType
DIM SHARED RecordSenha AS SINGLE
COMMON SHARED OpName AS STRING * 15
'
DIM SHARED InRegs AS Regtypex
DIM SHARED OutRegs AS Regtypex
'
'==================  MODULO DO EDITOR DE SETUP  ==============
'
DIM SHARED FileSet AS STRING
DIM SHARED FileTemp AS STRING
COMMON SHARED TabDir AS FileDirType
DIM SHARED NumberOfDir AS SINGLE
DIM SHARED Menux AS INTEGER
COMMON SHARED Cmd AS STRING
DIM SHARED Syntax AS STRING
COMMON SHARED Pfile AS INTEGER
COMMON SHARED Pvideo AS INTEGER
COMMON SHARED Pline AS INTEGER
DIM SHARED FileAtive AS STRING
DIM SHARED ModuloAtive AS STRING
DIM SHARED Unit AS STRING
DIM SHARED Dir AS STRING
DIM SHARED UnitQB AS STRING
DIM SHARED DirQB AS STRING
DIM SHARED PointerDir AS INTEGER
DIM SHARED File AS STRING
'
'==================  MODULO DE ABERTURA DE ARQUIVOS ========
'
DIM SHARED Linex AS FileSetup
DIM SHARED Linein(1000) AS STRING
DIM SHARED Ferro AS INTEGER
'
'==================  MODULO DE GRAFICOS ====================
'
TYPE FileGrafo
     FaixaA AS DOUBLE
     FaixaB AS DOUBLE
     FaixaC AS DOUBLE
     FaixaD AS DOUBLE
     FaixaE AS DOUBLE
     FaixaF AS DOUBLE
     FaixaG AS DOUBLE
     FaixaH AS DOUBLE
END TYPE
'
DIM SHARED Grafox AS FileGrafo
```

```
'
'****************** MODULO DE DEFINICOES DO GRAFICO ************
'grafico:
    DIM Maximo(8) AS SINGLE, Minimo(8) AS SINGLE
    DIM StartX AS SINGLE, StopX AS SINGLE
    DIM StartFaixa AS SINGLE, StopFaixa AS SINGLE
    DIM MaxGra AS SINGLE, MinGra AS SINGLE, PasGra AS SINGLE
    DIM Plow AS SINGLE, Phig AS SINGLE, DeltaGra AS SINGLE
    DIM aa AS SINGLE, bb AS SINGLE, cc  AS SINGLE, dd AS SINGLE
    DIM ee AS SINGLE, ff AS SINGLE, gg  AS SINGLE, hh AS SINGLE
    DIM xx AS SINGLE, yy AS SINGLE, zz AS SINGLE
    DIM RangeGra AS SINGLE, Pxx AS SINGLE, Pasxx AS SINGLE
'planilha:
    DIM Delta AS INTEGER
'=============================================================
REM DEFINICAO DAS FUNCOES E SUBROTINAS EXTERNAS
REM $INCLUDE: 'toolbox.bi'
'
'=============================================================
REM INICIO DA PROGRAMA PRINCIPAL
'*************************************
TIMER OFF
Mous.Exists = 0

'*********************************
REM MENU PRINCIPAL - CHAMADA DOS MODULOS
CALL M.FindMouse
'*********************************
OpName$ = SPACE$(15)
CALL Getmes(12, 0, 2, "NOME DO OPERADOR DO TESTE :", OpName$)
'
'=============================================================
REM MENU PRINCIPAL - CHAMADA DOS MODULOS
DIM SHARED Opcao$(5, 2)
Opcao$(1, 1) = "SETUP"
Opcao$(1, 2) = "Executa modulo de Inicializacao (SETUP)."
Opcao$(2, 1) = "MEDIDA"
Opcao$(2, 2) = "Executa modulo de MEDIDA do Teste e/ou Ensaio."
Opcao$(3, 1) = "GRAFICO"
Opcao$(3, 2) = "Executa modulo de GRAFICO do Teste e/ou Ensaio."
Opcao$(4, 1) = "PLANILHA"
Opcao$(4, 2) = "Executa modulo de PLANILHA do Teste e/ou
Ensaio."
Opcao$(5, 1) = "XFIM"
Opcao$(5, 2) = "FINAL DE OPERACOES"
Escolha = 1
DO WHILE Escolha <> 0
    ON ERROR GOTO Roterro
    COLOR 7, 9, 1: CLS
     Mes$ = "  AMBIENTE INTERERATIVO DE DESENVOLVIMENTO DE TESTES
E MEDIDAS     V0.0/90"
    CALL Saymes(2, 0, 2, Mes$)
    CALL Borda(6, 1, 20, 80, 1)
    CALL Borda(21, 1, 23, 80, 1)
    Escolha = Menu(4, 1, Opcao$())
    SELECT CASE Escolha
    CASE 1
        GOSUB SETUP:
    CASE 2
        GOSUB MEDIDA:
    CASE 3
        GOSUB GRAFICO:
```

```
    CASE 4
        GOSUB PLANILHA:
    CASE 5
        EXIT DO
    CASE ELSE
        BEEP
    END SELECT
LOOP
CLOSE
COLOR 7, 9, 1: CLS
END
'
'*********************************************
' FINAL DO PROGRAMA PRINCIPAL
'
Roterro:
    CALL Roterror(Ferro)
    RESUME NEXT

'*************************************************************
'   INCIO DO PROGRAMA GERADO PELO AMBIENTE
'

'************ ARQUIVO TEMPORARIO DE SETUP: *********************
'Nome  do Operador=  OTAVIO          Data: 04-04-1991       Hora:
14:37:58
'Nome do Arquivo = DESVIO.SET
SETUP:
    BDNAME$ = "D5334A"
    CALL IBFIND(BDNAME$,D5334A%)
    CALL IBCLR(D5334A%)
    CALL IBWRT(D5334A%,"INWA1SM1")
    CALL IBTRG(D5334A%)
    RETURN


'************ ARQUIVO TEMPORARIO DE MEDIDA: ********************
'Nome  do Operador=  OTAVIO          Data: 04-04-1991       Hora:
14:38:24
'Nome do Arquivo = DESVIO.MED
MEDIDA:
      ' DIMENSIONAMENTO DA VARIAVEIS
      DIM ValorDaMedida$ (200)
      DIM TempoDaMedida$ (200)
      '
      '
      TIMER ON
      Mes$  = "INICIO  DAS MEDIDAS  - DIGITE  QUALQUER TECLA PARA
CONTINUAR !!"
      CALL Mensagem(Mes$)
      '
      '      1. CICLO DE 50 MEDIDAS DE 1 SEGUNDO
      '
      RD$ = SPACE$(19)
    FOR X=1 TO 50
      GOSUB LOOPMEDE
      SLEEP 1
    NEXT X
```

```
        '
        '       2. CICLO DE 50 MEDIDAS DE 5 SEGUNDOS
        '
      FOR X=51 TO 100
        GOSUB LOOPMEDE
        SLEEP 5
      NEXT X
        '
        '       3. CICLO DE 50 MEDIDAS DE 10 SEGUNDOS
        '
      FOR X=101 TO 150
        GOSUB LOOPMEDE
        SLEEP 10
      NEXT X
        '
        '       4. CICLO DE 50 MEDIDAS DE 30 SEGUNDOS
        '
      FOR X=151 TO 200
        GOSUB LOOPMEDE
        SLEEP 30
      NEXT X
        '
        Mes$ = "FINAL DO ENSAIO"
        CALL Mensagem(Mes$)
        '
        CALL IBCLR(D5334A%)
        CALL IBONL(D5334A%, 0)
        CALL IBLOC(D5334A%)
        TIMER OFF
      '
        ' SELECAO DA FREQUENCIA CENTRAL
        FreCentral! = 0
         LOCATE 16,10:INPUT  "QUAL A  FREQUENCIA CENTRAL  EM MHz  :
";FreCentral!
        '
        ' ARQUIVO PARA COLOCAR OS DADOS DO ENSAIO
        '
        CALL FIDir (Unit$,Dir$)
        OPEN Unit$+Dir$+"\DESVIO.DAD" FOR OUTPUT AS #3
        FOR x = 1 TO 200
              Valor!=VAL(RIGHT$(ValorDaMedida$(x),LEN(ValorDaMedid
a$(x))-1))
              Desvio! = (FreCentral!*1E6) - Valor!
              Derro! = (Desvio!/(FreCentral!*1E6))*1E4
              WRITE #3,x,Valor!,Desvio!,Derro!,0,0,0,0
      NEXT x
        CLOSE # 3
        '
        ' FINAL DAS MEDIDAS
        '
        RETURN

LOOPMEDE:
            CALL IBRD(D5334A%,RD$)
       CALL IBTRG(D5334A%)
            ValorDaMedida$(x) = RD$
            TempoDaMedida$(x) = TIME$
            LOCATE 10,10:PRINT "NUMERO DA MEDIDA : ";X
            LOCATE 12,10:PRINT "VALOR MEDIDO : ";ValorDaMedida$(x)
            LOCATE 14,10:PRINT "TEMPO ATUAL : "; TempoDaMedida$(x)
        RETURN
```

```
'************************ MODULO DE GRAFICO ********************
'Nome  do Operador=  OTAVIO            Data: 04-11-1991        Hora:
11:14:49
'Nome do Arquivo = DESVIO.GRM
GRAFICO:
NomeGrafo$ = "DESVIO"
TipoGrafo =  2
TiTlePri$ = "MEDIDA DE DESVIO DE BASE - 10 MHz"
TiTleSec$ = "PADRAO HP 5334A"
TiTleX$ = "NUMERO DE MEDIDAS (200)"
TiTleY$ = "DESVIO (Hz)"
StartFaixa =  0
StopFaixa =  100
Faixas$ = "AC"
'*************** MODULO PRINCIPAL DO GRAFICO ***************
' 2. PARTE:
'------------------------ inicio do programa ----------------
Unit$ = GetDefaultDri$(InRegs, OutRegs, DriveLetter$)
Dir$ = GetCurrentDir$(InRegs, OutRegs, DriveLetter$)
File$ = NomeGrafo$ + ".DAD"
IF Dir$ = "\" THEN
     OPEN Unit$ + Dir$ + File$ FOR INPUT AS #3
ELSE
     OPEN Unit$ + Dir$ + "\" + File$ FOR INPUT AS #3
END IF
NuMeasure = 0
DO UNTIL EOF(3)
   INPUT #3, aa, bb, cc, dd, ee, ff, gg, hh
   NuMeasure = NuMeasure + 1
LOOP
'NuMeasure = NuMeasure - 1
CLOSE #3
'$DYNAMIC
REDIM mede(NuMeasure, 8)  AS SINGLE
IF Dir$ = "\" THEN
     OPEN Unit$ + Dir$ + File$ FOR INPUT AS #3
ELSE
     OPEN Unit$ + Dir$ + "\" + File$ FOR INPUT AS #3
END IF
Pm = 1
DO UNTIL EOF(3)
     INPUT  #3, mede(Pm,  1), mede(Pm,  2), mede(Pm, 3), mede(Pm,
4), mede(Pm, 5), mede(Pm, 6), mede(Pm, 7), mede(Pm, 8)
     Pm = Pm + 1
LOOP
CLOSE #3
StartFaixa = INT(NuMeasure * (StartFaixa / 100))
StopFaixa = INT(NuMeasure * (StopFaixa / 100))
IF StartFaixa = 0 THEN
     StartFaixa = 1
END IF
FOR y = 1 TO 8
     Minimo(y) = 1E+33
     Maximo(y) = -1E+33
NEXT y
FOR x = StartFaixa TO StopFaixa
     FOR y = 1 TO 8
          IF Minimo(y) > mede(x, y) THEN
              Minimo(y) = mede(x, y)
          END IF
```

```
            IF Maximo(y) < mede(x, y) THEN
                Maximo(y) = mede(x, y)
            END IF
        NEXT y
    NEXT x
    MinGra = 1E+30: MaxGra = -1E+30
    FOR y = TipoGrafo TO LEN(Faixas$)
        x = ASC(MID$(Faixas$, y, 1)) - 64
        IF MinGra > Minimo(x) THEN
            MinGra = Minimo(x)
        END IF
        IF MaxGra < Maximo(x) THEN
            MaxGra = Maximo(x)
        END IF
    NEXT y
    RangeGra = MaxGra - MinGra
    IF RangeGra = 0 THEN
        RangeGra = ABS(MaxGra * .1)
    END IF
    DeltaGra = .1 * RangeGra
    Plow = MinGra - DeltaGra
    Phig = MaxGra + DeltaGra
    CLS
    COLOR 7, 9, 1
    SCREEN 2
    X1 = 60: Y1 = 170: LX = 640 - X1 - 16: LY = Y1 - 16
    AX = X1 - 5: LAX = LX - 5
    CALL DoXAxis(X1, Y1, LX, 5, 2)
    CALL DoYAxis(X1, Y1, LY, 5, 4)
    XtextArr$(0) = TitleX$
    FOR x = 1 TO 6
        XtextArr$(x)    =    STR$(INT(StartFaixa    +    ((StopFaixa    -
    StartFaixa) / 5) * (x - 1)))
    NEXT x
    YtextArr$(0) = TitleY$
    FOR y = 1 TO 6
        YtextArr$(y) = STR$((Plow + ((RangeGra / 5) * (y - 1))))
    NEXT y
    CALL AnnXaxis(X1, Y1 + 6, LX, 5, XtextArr$(), 0)
    CALL AnnYaxis(X1, Y1 + 6, LY, 5, YtextArr$(), 0)

    VIEW (60, 170)-(640 - 16, 16)
    IF TipoGrafo = 1 THEN
        WINDOW (StartFaixa, Phig)-(StopFaixa, Plow)
        IF Plow < 0 THEN
            LINE (StartFaixa, 0)-(StopFaixa, 0)
        END IF
        PasGra = (StopFaixa - StartFaixa) / NuMeasure
        FOR y = 1 TO LEN(Faixas$)
            x = ASC(MID$(Faixas$, y, 1)) - 64
            FOR zz = StartFaixa TO StopFaixa + PasGra STEP PasGra
                yy = mede(zz, x)
                PSET (zz, yy)
            NEXT zz
        NEXT y
    ELSE
        StartX = Minimo(1)
        StopX = Maximo(1)
        WINDOW (StartX, Phig)-(StopX, Plow)
        IF Plow < 0 THEN
            LINE (StartX, 0)-(StopX, 0)
        END IF
```

```
      PasGra = (StopFaixa - StartFaixa) / NuMeasure
      FOR y = 2 TO LEN(Faixas$)
            x = ASC(MID$(Faixas$, y, 1)) - 64
            FOR zz = StartFaixa TO StopFaixa + PasGra STEP PasGra
                  xx = mede(zz, 1)
                  yy = mede(zz, x)
                  PSET (xx, yy)
            NEXT zz
      NEXT y
END IF
VIEW PRINT 1 TO 24
CALL Saymes(1, 0, 0, TitlePri$)
CALL Saymes(2, 0, 0, TitleSec$)
SLEEP
SCREEN 0
COLOR 7, 9, 1
RETURN

'************************ MODULO DE PLANILHA ********************
'Nome  do Operador=  OTAVIO          Data: 04-11-1991       Hora:
11:16:23
'Nome do Arquivo = DESVIO.PLM
PLANILHA:
NomePlani$ = "DESVIO"
Colunas =  4
TiTlePri$ = "TABELA DE VALORES DA MEDIDA DE DESVIO"
TitleFA$ = "LEITURA"
TitleFB$ = "FREQUENCIA"
TitleFC$ = "DESVIO"
TitleFD$ = "ERRO PPM"
TitleFE$ = ""
TitleFF$ = ""
TitleFG$ = ""
TitleFH$ = ""
StartFaixa =  0
StopFaixa =  100
Faixas$ = "ABCD"
'***************** MODULO 2 DA PLANILHA ************************
'
'------------------------ inicio do programa -------------
Unit$ = GetDefaultDri$(InRegs, OutRegs, DriveLetter$)
Dir$ = GetCurrentDir$(InRegs, OutRegs, DriveLetter$)
File$ = NomePlani$ + ".DAD"
IF Dir$ = "\" THEN
     OPEN Unit$ + Dir$ + File$ FOR INPUT AS #3
ELSE
     OPEN Unit$ + Dir$ + "\" + File$ FOR INPUT AS #3
END IF
NuMeasure = 0
DO UNTIL EOF(3)
   INPUT #3, aa, bb, cc, dd, ee, ff, gg, hh
     NuMeasure = NuMeasure + 1
LOOP
NuMeasure = NuMeasure - 1
CLOSE #3
'$DYNAMIC
REDIM Mede(NuMeasure, 8)  AS SINGLE
IF Dir$ = "\" THEN
     OPEN Unit$ + Dir$ + File$ FOR INPUT AS #3
ELSE
     OPEN Unit$ + Dir$ + "\" + File$ FOR INPUT AS #3
END IF
```

```
Pm = 0
DO UNTIL EOF(3)
     INPUT  #3, Mede(Pm,  1), Mede(Pm,  2), Mede(Pm, 3), Mede(Pm,
4), Mede(Pm, 5), Mede(Pm, 6), Mede(Pm, 7), Mede(Pm, 8)
     Pm = Pm + 1
LOOP
CLOSE #3
StartFaixa = NuMeasure * (StartFaixa / 100)
StopFaixa = NuMeasure * (StopFaixa / 100)
FOR Y = 1 TO 8
     Minimo(Y) = 99999999
     Maximo(Y) = -99999999
NEXT Y
FOR x = StartFaixa TO StopFaixa
     FOR Y = 1 TO 8
          IF Minimo(Y) > Mede(x, Y) THEN
              Minimo(Y) = Mede(x, Y)
          END IF
          IF Maximo(Y) < Mede(x, Y) THEN
              Maximo(Y) = Mede(x, Y)
          END IF
     NEXT Y
NEXT x

' Define passo na planilha
Delta = 1

' Menu principal da planilha
REDIM Titulos$(8)
Titulos$(1) = TiTleFA$
Titulos$(2) = TiTleFB$
Titulos$(3) = TiTleFC$
Titulos$(4) = TiTleFD$
Titulos$(5) = TiTleFE$
Titulos$(6) = TiTleFF$
Titulos$(7) = TiTleFG$
Titulos$(8) = TiTleFH$

REDIM SHARED Opla$(5, 2)
Opla$(1, 1) = "TELA"
Opla$(1, 2) = "Mostra a Planilha na Tela"
Opla$(2, 1) = "ARQUIVO"
Opla$(2, 2) = "Salva a Planilha em um Arquivo com extensao .TXT"
Opla$(3, 1) = "IMPRESSORA"
Opla$(3, 2) = "Imprime a Planilha na Impressora"
Opla$(4, 1) = "PASSO"
Opla$(4,  2) = "Altera o intervalo de Medidas na Planilha (padrao
= 1)"
Opla$(5, 1) = "XFIM"
Opla$(5, 2) = "FINAL DE OPERACOES"

Escolha = 1
DO WHILE Escolha <> 0
     VIEW PRINT 4 TO 5: CLS : COLOR 7, 9, 1: VIEW PRINT
     Escolha = Menu(4, 1, Opla$())
     SELECT CASE Escolha
     CASE 1
          GOSUB PlaTela:
     CASE 2
          GOSUB PlaFile:
     CASE 3
          GOSUB PlaImp:
```

```
      CASE 4
            GOSUB PlaStep:
      CASE 5
            EXIT DO
      CASE ELSE
            BEEP
      END SELECT
LOOP
VIEW PRINT 4 TO 5: CLS : COLOR 7, 9, 1: VIEW PRINT
RETURN

'*********************************
'SUB ROTINAS DA PLANILHA
'
'_____________________________________________________________________
PlaTela:
      Ptela = 1
      TabCol = 80 / Colunas
      SCREEN 0, 7, 1, 1
      COLOR 7, 9, 1: CLS
      CALL Saymes(2, 0, 1, TitlePri$)
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            LOCATE 4, 1 + Y * TabCol: PRINT Titulos$(x)
      NEXT Y
      Linha = 5
      FOR z = StartFaixa TO StopFaixa STEP Delta
            FOR Y = 0 TO Colunas - 1
                  x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
                  IF Colunas = 8 THEN
                        LOCATE  Linha, 1  + Y  * TabCol:  PRINT USING
"+.##^^^^"; Mede(z, x)
                  ELSE
                        LOCATE  Linha, 1  + Y * TabCol: PRINT Mede(z,
x)
                  END IF
            NEXT Y
            Linha = Linha + 1
            IF Linha = 21 THEN
                  Linha = 5
                  Mes$ = ""
                  CALL Mensagem(Mes$)
                  VIEW  PRINT 5  TO 22:  CLS :  COLOR 7,  9, 1: VIEW
PRINT
            END IF
      NEXT z
      Mes$ = ""
      CALL Mensagem(Mes$)
      VIEW PRINT 4 TO 22: CLS : COLOR 7, 9, 1: VIEW PRINT
      Linha = 7
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            LOCATE  Linha, 1  + Y  *  TabCol:  PRINT  "Maximo  "  +
MID$(Faixas$, Y + 1, 1)
      NEXT Y
      Linha = 9
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            IF Colunas = 8 THEN
                  LOCATE   Linha,  1  +  Y  *  TabCol:  PRINT  USING
"+.##^^^^"; Maximo(x)
            ELSE
```

```
                LOCATE Linha, 1 + Y * TabCol: PRINT Maximo(x)
            END IF
        NEXT Y
        Linha = 13
        FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            LOCATE Linha, 1 + Y * TabCol: PRINT "Minimo " +
MID$(Faixas$, Y + 1, 1)
        NEXT Y
        Linha = 15
        FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            IF Colunas = 8 THEN
                LOCATE Linha, 1 + Y * TabCol: PRINT USING
"+.##^^^^"; Minimo(x)
            ELSE
                LOCATE Linha, 1 + Y * TabCol: PRINT Minimo(x)
            END IF
        NEXT Y
         Mes$ = ""
        CALL Mensagem(Mes$)
        CLS
        SCREEN 0, 7, 0, 0
        RETURN

'-------------------------------------------------------------------
PlaFile:
        TabCol = 80 / Colunas
        File$ = NomePlani$ + ".TXT"
        IF Dir$ = "\" THEN
            OPEN Unit$ + Dir$ + File$ FOR OUTPUT AS #3
        ELSE
            OPEN Unit$ + Dir$ + "\" + File$ FOR OUTPUT AS #3
        END IF
        x = (80 - LEN(TitlePri$)) / 2
        PRINT #3, SPACE$(x) + TitlePri$
        PRINT #3, ""
        FOR Y = 1 TO Colunas
            x = ASC(MID$(Faixas$, Y, 1)) - 64
            PRINT #3, Titulos$(x) + SPACE$(TabCol -
LEN(Titulos$(x)) - 1);
        NEXT Y
        PRINT #3, ""
        FOR z = StartFaixa TO StopFaixa STEP Delta
            OutLine$ = ""
            FOR Y = 1 TO Colunas
                x = ASC(MID$(Faixas$, Y, 1)) - 64
                OutLine$ = OutLine$ + STR$(Mede(z, x)) +
SPACE$(TabCol - LEN(STR$(Mede(z, x))) - 1)
            NEXT Y
            PRINT #3, OutLine$
        NEXT z
        PRINT #3, ""
        FOR Y = 1 TO Colunas
            x = ASC(MID$(Faixas$, Y, 1)) - 64
            PRINT #3, "Maximo " + MID$(Faixas$, Y, 1) +
SPACE$(TabCol - 8);
        NEXT Y
        PRINT #3, ""
        FOR Y = 1 TO Colunas
            x = ASC(MID$(Faixas$, Y, 1)) - 64
```

```
            PRINT      #3,      Maximo(x);       SPACE$(TabCol       -
LEN(STR$(Maximo(x))) - 1);
      NEXT Y
      PRINT #3, "": PRINT #3, ""
      FOR Y = 1 TO Colunas
            x = ASC(MID$(Faixas$, Y, 1)) - 64
            PRINT   #3,  "Minimo    "  +   MID$(Faixas$,  Y,    1)  +
SPACE$(TabCol - 8);
      NEXT Y
      PRINT #3, ""
      FOR Y = 1 TO Colunas
            x = ASC(MID$(Faixas$, Y, 1)) - 64
            PRINT      #3,      Minimo(x);       SPACE$(TabCol       -
LEN(STR$(Minimo(x))) - 1);
      NEXT Y
      CLOSE #3
      SCREEN 0, 7, 1, 1
      COLOR 7, 9, 1: CLS
      SHELL "MORE < " + File$
      SLEEP
      SCREEN 0, 7, 0, 0
      RETURN

'_____________________________________________________
PlaImp:
      Mes$  = "Prepare  a impressora  e confirme a impressao (S/N)
?? "
      Confirme$ = ""
      CALL Getmes(22, 0, 0, Mes$, Confirme$)
      IF    Confirme$    =    CHR$(27)    OR    Confirme$    =    ""    OR
UCASE$(Confirme$) <> "S" THEN
            LOCATE 22, 2: PRINT SPACE$(76)
            BEEP
            RETURN
      END IF
      TabCol = 80 / Colunas
      GOSUB Cabeca:
      Linha = 5
      FOR z = StartFaixa TO StopFaixa STEP Delta
            IF Linha = 60 THEN
                  LPRINT CHR$(12)
                  GOSUB Cabeca:
                  Linha = 5
            END IF
            FOR Y = 0 TO Colunas - 1
                  x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
                  IF Colunas = 8 THEN
                        LPRINT USING "+.##^^^^"; Mede(z, x);
                        LPRINT SPACE$(TabCol - 8);
                  ELSE
                       LPRINT     Mede(z,     x);     SPACE$(TabCol      -
LEN(STR$(Mede(z, x))) - 1);
                  END IF
            NEXT Y
            Linha = Linha + 1
      NEXT z
      LPRINT CHR$(12)
      GOSUB Cabeca:
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            LPRINT   "Maximo   "   +  MID$(Faixas$,  Y  +  1,  1)  +
SPACE$(TabCol - 8);
```

```
      NEXT Y
      LPRINT ""
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            IF Colunas = 8 THEN
                  LPRINT USING "+.##^^^^"; Maximo(x);
                  LPRINT SPACE$(TabCol - 8);
            ELSE
                  LPRINT       Maximo(x);       SPACE$(TabCol       -
LEN(STR$(Maximo(x))) - 1);
            END IF
      NEXT Y
      LPRINT "": LPRINT ""
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            LPRINT   "Minimo  "   +  MID$(Faixas$,  Y  +  1,  1)  +
SPACE$(TabCol - 8);
      NEXT Y
      LPRINT ""
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            IF Colunas = 8 THEN
                  LPRINT USING "+.##^^^^"; Minimo(x);
                  LPRINT SPACE$(TabCol - 8);
            ELSE
                  LPRINT       Minimo(x);       SPACE$(TabCol       -
LEN(STR$(Minimo(x))) - 1);
            END IF
      NEXT Y
      LOCATE 22, 2: PRINT SPACE$(76)
      RETURN

'-------------------------------------------------
' Sub rotina de cabecalho
Cabeca:
      x = (80 - LEN(TitlePri$)) / 2
      LPRINT SPACE$(x) + TitlePri$
      LPRINT ""
      FOR Y = 0 TO Colunas - 1
            x = ASC(MID$(Faixas$, Y + 1, 1)) - 64
            LPRINT      Titulos$(x)      +      SPACE$(TabCol      -
LEN(Titulos$(x)));
      NEXT Y
      LPRINT ""
      RETURN

'-------------------------------------------------------------
PlaStep:
      Pstep = 1
      VIEW PRINT 4 TO 5: CLS : COLOR 7, 9, 1: VIEW PRINT
      LOCATE  4, 2:  PRINT "Digite  o  novo  Intervalo  Padrao  de
Medidas (Padrao = 1)"
      LOCATE 5, 2: INPUT "INTERVALO = "; Pstep
      IF Pstep <> 0 THEN
            Delta = Pstep
      ELSE
            Delta = 1
      END IF
      VIEW PRINT 4 TO 5: CLS : COLOR 7, 9, 1: VIEW PRINT
      RETURN
'-------------------------------------------------------------
```

ANEXO 03

BIBLIOTECAS DE FUNÇÕES ESPECIAIS

TOOLBOX.LIB - Versão 91.01

Declarações gerais das subrotinas e funções especiais desenvolvidas para a implementação do AIDS_TME. (28-30)

DECLARAÇÕES GERAIS:

```
'=============================================================================
REM
REM TOOLBOX ------------------------------------   06/09/90
REM
REM BIBLIOTECA DE FUNCOES E PROCEDIMENTOS GERAIS
'=============================================================================

'=== D E C L A R A C O E S   D E   S U B R O T I N A S  ==
DECLARE SUB LineUp ()
DECLARE SUB LIneDown ()
DECLARE SUB EditaCmd (Cmd$, Pline%, Pfile%, Pvideo%)
DECLARE SUB MostraDir (PointerDir%)
DECLARE SUB OpenFile (File$, NumberFile%, SizeFile%)
DECLARE SUB OpenSeqFile (File$, NumberFile%, SizeFile%) DECLARE
SUB Addsenha ()
DECLARE SUB AltSenha ()
DECLARE SUB DelSenha (Unit$, dir$)
DECLARE FUNCTION Findpassword% (Pw$, OpName$)
DECLARE FUNCTION Scramble# (Pw$)            ' craia a senha
DECLARE SUB Getpassword (Pw$)               ' pega password
DECLARE SUB Saymes (Linha%, coluna%, Tipoborda%, Mes$)   'envia
mensagem
DECLARE SUB Getmes (Linha%, coluna%, Tipoborda%, Mes$, Var$)
DECLARE SUB Mensagem (Mes$)                 ' mensagem de rodape
DECLARE SUB Borda (x1%, y1%, x2%, y2%, Tipoborda%)
DECLARE FUNCTION Menu% (Linha%, coluna%, Opçâo$())
DECLARE FUNCTION Achoice% (ax1%, ay1%, Op$())
DECLARE SUB Command (Syntax$, Cmd$)
DECLARE SUB FirstSenha ()
DECLARE SUB Roterror (Ferro%)
'
' new sub declare
DECLARE SUB QBDir (UnitQB$, DirQB$)
DECLARE SUB FIDir (Unit$, dir$)
DECLARE SUB DirZero ()
DECLARE SUB LimpaVideo ()
DECLARE SUB FindBar (File$)
DECLARE SUB DelFileDir (NumberOfDir!, PointerDir%)
'
'
'=== D E C L A R A C O E S   D O   A M B I E N T E =====
DECLARE SUB INTERRUPTX (intnum AS INTEGER, InRegs AS Regtypex,
OutRegs AS Regtypex)
DECLARE FUNCTION GetCurrentDir$ (InRegs AS Regtypex, OutRegs AS
Regtypex, DriveLetter$)
```

```
DECLARE FUNCTION GetDefaultDri$ (InRegs AS Regtypex, OutRegs AS
Regtypex, DriveLetter$)
DECLARE FUNCTION SetDefaultDri (InRegs AS Regtypex, OutRegs AS
Regtypex, DriveLetter$)
'
'===  D E C L A R A C O E S  D O  M O U S E ========
DECLARE SUB M.FindMouse ()
DECLARE SUB M.Flushbuttons ()
DECLARE SUB M.PenOn ()
DECLARE SUB M.PenOff ()
DECLARE SUB M.Movement ()
DECLARE SUB M.GetBtnPress (button%)
DECLARE SUB M.GetBtnRelease (button%)
DECLARE SUB M.TextCursor (cursortype%, scan1%, scan2%)
DECLARE SUB M.WaitClick (button%)
DECLARE SUB M.HorizRange (leftcol%, rightcol%)
DECLARE SUB M.VertRange (upperrow%, lowerrow%)
DECLARE SUB M.GetPos ()
DECLARE SUB M.ShowCursor ()
DECLARE SUB M.HideCursor ()
DECLARE SUB M.Initialize ()
DECLARE SUB M.MoveCursor (RRow%, CCol%)
DECLARE SUB M.Mouse (a%, b%, c%, d%)
'
'==================  M O U S E  ============================
'
CONST TRUE = 1, FALSE = 0, YES = 1, NO = 0, LEFT = 0, RIGHT = 1
CONST BOTH = 2, EITHER = 3, HARDCURSOR = 1, SOFTCURSOR = 0
'
' DEFINIÇAO DO TIPO DE DADOS

TYPE mouseinfo
     Exists AS INTEGER          ' > 0 se mouse existe
     CursorOn AS INTEGER        ' 1 se cursor ativo, 0 desligado
     BtnStatus AS INTEGER       ' status do botão ativo
     BtnClicks AS INTEGER       ' vezes que o botão ativo foi
apertado
     Column AS INTEGER          ' coluna do mouse
     Row AS INTEGER             ' linha do mouse
     HMovement AS INTEGER       ' deslocamento horizontal do mouse
     VMovement AS INTEGER       ' deslocamento vertical do mouse
END TYPE

DIM SHARED button AS INTEGER
DIM SHARED a AS INTEGER, b AS INTEGER, c AS INTEGER, d AS
INTEGER
DIM SHARED cursortype AS INTEGER, scan1 AS INTEGER, scan2 AS
INTEGER
DIM SHARED leftcol AS INTEGER, rightcol AS INTEGER
DIM SHARED upperrow AS INTEGER, lowerrow AS INTEGER
DIM SHARED RRow  AS INTEGER, CCol AS INTEGER

DIM Mous AS mouseinfo
'=============================================================
'==================  MODULO DO EDITOR DE SENHA  ==============
'
DIM SHARED SenhaRecord AS SenhaType
COMMON SHARED NumberOfSenha AS INTEGER
DIM SHARED Senha AS SenhaType
DIM SHARED RecordSenha AS SINGLE
COMMON SHARED OpName AS STRING * 15
'
```

```
DIM SHARED InRegs AS Regtypex
DIM SHARED OutRegs AS Regtypex
'
'================  MODULO DO EDITOR DE SETUP  ============
'
DIM SHARED Unit AS STRING
DIM SHARED dir AS STRING
DIM SHARED FileSet AS STRING
DIM SHARED FileTemp AS STRING
COMMON SHARED TabDir AS FileDirType
DIM SHARED NumberOfDir AS SINGLE
DIM SHARED PointerDir AS INTEGER
COMMON SHARED Cmd AS STRING
COMMON SHARED Pfile  AS INTEGER
COMMON SHARED Pvideo  AS INTEGER
COMMON SHARED Pline AS INTEGER
COMMON SHARED hpos AS INTEGER
'
'================  MODULO DE ABERTURA DE ARQUIVOS ========
'
DIM SHARED Linex AS FileSetup
DIM SHARED Linein(100) AS STRING
'
'================  MODULO DE GRAFICOS ====================
'
TYPE FileGrafo
     FaixaA AS DOUBLE
     FaixaB AS DOUBLE
     FaixaC AS DOUBLE
     FaixaD AS DOUBLE
     FaixaE AS DOUBLE
     FaixaF AS DOUBLE
     FaixaG AS DOUBLE
     FaixaH AS DOUBLE
END TYPE
'
DIM SHARED Grafox AS FileGrafo
```

```
FUNCTION Achoice% (ax1%, ay1%, Op$())
'_____________________________________________________________________
'TITULO: Achoice
'ARGUMENTOS: x1,y1,Op$()
'AÇAO: Monta um menu na vertical com as opçÕes definida em OP$()
e borda dupla, retorna em Achoive o número da opçâo escolhida.
'
'*********************************************************************
DEFSNG A-Z
SHARED Mous AS mouseinfo
DEFINT A-Z
numop = UBOUND(Op$)
REM INICIALIZAÇAO DE X2 E Y2
x2 = ax1 + UBOUND(Op$) + 1
y2 = 0
FOR x = 1 TO UBOUND(Op$, 1)
     temp = LEN(Op$(x))
     IF temp > y2 THEN
          y2 = temp
     END IF
NEXT x
y2 = ay1 + y2 + 2
REM CONFEÇAO DA BORDA DUPLA
LOCATE ax1, ay1: PRINT CHR$(201)
LOCATE ax1, y2: PRINT CHR$(187)
FOR x = ax1 + 1 TO x2 - 1
     LOCATE x, ay1: PRINT CHR$(186);
     LOCATE x, y2: PRINT CHR$(186);
NEXT x
LOCATE x2, ay1: PRINT CHR$(200)
LOCATE x2, y2: PRINT CHR$(188)
FOR Y = ay1 + 1 TO y2 - 1
     LOCATE ax1, Y: PRINT CHR$(205);
     LOCATE x2, Y: PRINT CHR$(205);
NEXT Y
FOR x = 1 TO UBOUND(Op$, 1)
     LOCATE (ax1 + x), ay1 + 1: PRINT Op$(x) + SPACE$((y2 - ay1 -
2) - LEN(Op$(x)))
NEXT x
REM MOVIMENTAÇAO DO CURSOR E ESCOLHA DA OPÇAO
hpos = ax1 + 1: vpos = ay1 + 1
COLOR 4, 7, 1
LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
COLOR 14, 9, 1
IF Mous.Exists THEN
     CALL M.HorizRange(ay1 * 8, (y2 - 2) * 8)
     CALL M.VertRange((ax1) * 8, (x2 - 2) * 8)
     DO
          CALL M.Flushbuttons
          CALL M.ShowCursor
          CALL M.WaitClick(LEFT)
          CALL M.HideCursor
          LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
          hpos = ((Mous.Row / 8) + 1)
          COLOR 4, 7, 1
          LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
          COLOR 14, 9, 1
          Achoice = hpos - ax1
          SLEEP 1
          EXIT DO
     LOOP
```

```
ELSE
DO WHILE r$ <> CHR$(13)
     r$ = ""
     DO WHILE r$ = ""
          r$ = INKEY$
     LOOP
     IF LEN(r$) = 2 THEN
          SELECT CASE ASC(RIGHT$(r$, 1))
               CASE 80: REM seta para baixo
                    LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
                    hpos = hpos + 1
                    IF hpos = x2 THEN
                         hpos = ax1 + 1
                    END IF
               CASE 72: REM seta para cima
                    LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
                    hpos = hpos - 1
                    IF hpos = ax1 THEN
                         hpos = x2 - 1
                    END IF
               CASE 75, 77
                    Achoice = 0
                    EXIT DO
          END SELECT
     ELSE
     ' BUSCA A PRIMEIRA LETRA DA OPÇAO
          IF (r$ >= "A" AND r$ <= "Z") OR (r$ >= "a" AND r$ <=
"z") THEN
               x = 1
               DO WHILE x <= numop
                    Optemp$ = LEFT$(Op$(x), 1)
                    IF r$ = Optemp$ OR ASC(r$) - 32 = ASC(Optemp$)
THEN
                         LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
                         IF hpos - ax1 <> x THEN
                            hpos = ax1 + x
                            EXIT DO
                         END IF
                    END IF
                    x = x + 1
               LOOP
               IF x > numop THEN
                    BEEP
               END IF
          END IF
          IF r$ = CHR$(27) THEN
               Achoice = 0
               EXIT DO
          END IF
     END IF
     COLOR 4, 7, 1
     LOCATE hpos, vpos: PRINT Op$(hpos - ax1)
     COLOR 14, 9, 1
     Achoice = hpos - ax1
LOOP
END IF
REM FINAL DA FUNÇAO Achoice%

END FUNCTION
```

SUB Addsenha

```
'-----------------------------------------------------------------------
'TITULO: AddSenha
'ARGUMENTOS: nenhum
'AÇAO: Adiciona um nova senha no arquivo de senha AIDSENHA.DAT
'
'***********************************************************************
DO
    OpName$ = SPACE$(15): CALL Getmes(12, 0, 2, "NOME DO
OPERADOR : ", OpName$)
    IF LEN(OpName$) > 15 THEN
        Mes$ = "NOME DO OPERADOR MAIOR QUE 15 CARACTERES !!"
        CALL Mensagem(Mes$)
        EXIT DO
    END IF
    Mes$ = " Digite a sua senha !!!"
    CALL Saymes(16, 0, 1, Mes$)
    Pw$ = "": CALL Getpassword(Pw$)
    IF Pw$ = CHR$(27) OR Pw$ = "" THEN
        EXIT DO
    END IF
    Firstpw$ = "": Firstpw$ = Pw$
    Mes$ = "Confirme a sua senha !!"
    CALL Saymes(16, 0, 1, Mes$)
    Pw$ = "": CALL Getpassword(Pw$)
    IF Pw$ = CHR$(27) OR Pw$ = "" THEN
        EXIT DO
    END IF
    IF Firstpw$ <> Pw$ THEN
        Mes$ = "SENHA N&O CONFERE, DIGITE NOVAMENTE"
        CALL Mensagem(Mes$)
    ELSE
        IF Findpassword(Pw$, OpName$) = 1 THEN
            Mes$ = "SENHA JA CADASTRADA, TENTE OUTRA !!"
            CALL Mensagem(Mes$)
        ELSE
            EXIT DO
        END IF
    END IF
LOOP
IF Pw$ <> CHR$(27) AND Pw$ <> "" THEN
    NumberOfSenha = NumberOfSenha + 1
    Senha.OperatorSenha = Scramble(Pw$)
    Senha.OperatorName = UCASE$(OpName$)
    PUT #1, NumberOfSenha, Senha
END IF
END SUB
```

SUB AltSenha

```
'-----------------------------------------------------------------------
'TITULO: AltSenha
'ARGUMENTOS: nenhum
'AÇAO: Altera um senha no arquivo de senha AIDSENHA.DAT.
'
'***********************************************************************
DO
    OpName$ = SPACE$(15): CALL Getmes(12, 0, 2, "NOME DO
OPERADOR :", OpName$)
    IF LEN(OpName$) > 15 THEN
        Mes$ = "NOME DO OPERADOR MAIOR QUE 15 CARACTERES !!!"
```

```
            CALL Mensagem(Mes$)
            EXIT DO
        END IF
        Mes$ = " Digite a sua senha !!!"
        CALL Saymes(16, 0, 1, Mes$)
        Pw$ = "": CALL Getpassword(Pw$)
        IF Pw$ = CHR$(27) OR Pw$ = "" THEN
            EXIT DO
        END IF
        IF Findpassword(Pw$, OpName$) = 1 THEN
            Mes$ = LTRIM$(RTRIM$(Senha.OperatorName))
            PointerSenha = RecordSenha
            CALL Saymes(19, 0, 0, "Nome do Operador para esta Senha
: " + Mes$)
            Mes$ = " Digite a sua NOVA senha !!!"
            CALL Saymes(16, 0, 1, Mes$)
            Pw$ = "": CALL Getpassword(Pw$)
            IF Pw$ = CHR$(27) OR Pw$ = "" THEN
               EXIT DO
            END IF
            IF Findpassword(Pw$, OpName$) = 1 AND
RTRIM$(UCASE$(OpName$)) <> RTRIM$(Senha.OperatorName) THEN
                Mes$ = "SENHA JA CADASTRADA, TENTE OUTRA VEZ !!"
                CALL Mensagem(Mes$)
                EXIT DO
            ELSE
                Firstpw$ = "": Firstpw$ = Pw$
                Mes$ = "Confirme a sua NOVA senha !!"
                CALL Saymes(16, 0, 1, Mes$)
                Pw$ = "": CALL Getpassword(Pw$)
                IF Pw$ = CHR$(27) OR Pw$ = "" THEN
                    EXIT DO
                END IF
                IF Firstpw$ <> Pw$ THEN
                    Mes$ = "SENHA NAO CONFERE, DIGITE NOVAMENTE"
                    CALL Mensagem(Mes$)
                    EXIT DO
                END IF
            END IF
        ELSE
            Mes$ = "SENHA NAO CONFERE, DIGITE NOVAMENTE"
            CALL Mensagem(Mes$)
            EXIT DO
        END IF
        Senha.OperatorSenha = Scramble(Pw$)
        Senha.OperatorName = UCASE$(OpName$)
        PUT #1, PointerSenha, Senha
        EXIT DO
    LOOP
    END SUB
```

SUB Borda (x1, y1, x2, y2, Tipoborda)

```
'-------------------------------------------------------------------------
'TITULO: Borda
'ARGUMENTOS: x1, y1, x2, y2, Tipoborda
'AÇAO: Construe um a borda nas coordenadas x e y, com borda
simples = 1 e borda dupla = 2.
'
'*************************************************************
REM BORDA = 1 - BORDA SIMPLES
```

```
REM BORDA = 2 - BORDA DULPA
LOCATE x1, y1: PRINT CHR$(218 - (Tipoborda - 1) * 17)
LOCATE x1, y2: PRINT CHR$(191 - (Tipoborda - 1) * 4)
FOR x = x1 + 1 TO x2 - 1
    LOCATE x, y1: PRINT CHR$(179 + (Tipoborda - 1) * 7);
    LOCATE x, y2: PRINT CHR$(179 + (Tipoborda - 1) * 7);
NEXT x
LOCATE x2, y1: PRINT CHR$(192 + (Tipoborda - 1) * 8)
LOCATE x2, y2: PRINT CHR$(217 - (Tipoborda - 1) * 29)
FOR Y = y1 + 1 TO y2 - 1
    LOCATE x1, Y: PRINT CHR$(196 + (Tipoborda - 1) * 9);
    LOCATE x2, Y: PRINT CHR$(196 + (Tipoborda - 1) * 9);
NEXT Y
REM FINAL DO PROCEDIMENTO Borda
END SUB
```

SUB Command (Syntax$, Cmd$)

```
'-------------------------------------------------------------------------
'TITULO: Command
'ARGUMENTOS: Syntax$, Cmd$
'AÇÃO: Recebe linha de comando do GPIB de acordo com a sintaxe
'
'*************************************************************
Cmd1$ = ""
VIEW PRINT 4 TO 5: CLS : COLOR 14, 9, 1: VIEW PRINT
LOCATE 4, 2: PRINT Syntax$
LOCATE 5, 2: PRINT Cmd$
LOCATE 5, LEN(Cmd$) + 2: LINE INPUT "", Cmd1$
Cmd$ = Cmd$ + Cmd1$
VIEW PRINT 4 TO 5: CLS : COLOR 14, 9, 1: VIEW PRINT
END SUB
```

SUB DelFileDir (NumberOfDir!, PointerDir%)

```
'-------------------------------------------------------------------------
'TITULO: DelFileDir
'ARGUMENTOS: NumberOfDir!, PointerDir%
'AÇÃO: Deleta arquivo escolhido da lista de diretório e do disco
'de trabalho.
'
'*************************************************************
OPEN Unit$ + dir$ + "\TEMP.DAT" FOR RANDOM AS #3 LEN =
LEN(TabDir)
IF PointerDir = 1 THEN
    FOR RecorDir = 2 TO NumberOfDir
        GET #2, RecorDir, TabDir
        PUT #3, RecorDir - 1, TabDir
    NEXT RecorDir
ELSE
    FOR RecorDir = 1 TO PointerDir - 1
        GET #2, RecorDir, TabDir
        PUT #3, RecorDir, TabDir
    NEXT RecorDir
    IF PointerDir <> NumberOfDir THEN
        FOR RecorDir = PointerDir + 1 TO NumberOfDir
            GET #2, RecorDir, TabDir
            PUT #3, RecorDir - 1, TabDir
        NEXT RecorDir
    END IF
```

```
END IF
CLOSE #3
END SUB
```

SUB DelSenha (Unit$, dir$)

```
'-----------------------------------------------------------------------
'TITULO: DelSenha
'ARGUMENTOS: nenhum
'AÇAO: Deleta uma senha no arquivo de senha AIDSENHA.DAT.
'
'***********************************************************************
DO
    OpName$ = SPACE$(15): CALL Getmes(12, 0, 2, "NOME DO
OPERADOR : ", OpName$)
    IF LEN(OpName$) > 15 THEN
        Mes$ = "NOME DO OPERADOR MAIOR QUE 15 CARACTERES !!"
        CALL Mensagem(Mes$)
        EXIT DO
    END IF
    Mes$ = " Digite a sua senha !!!"
    CALL Saymes(16, 0, 1, Mes$)
    Pw$ = "": CALL Getpassword(Pw$)
    IF Pw$ = CHR$(27) OR Pw$ = "" THEN
        EXIT DO
    END IF
    IF Findpassword(Pw$, OpName$) = 1 THEN
        Mes$ = RTRIM$(LTRIM$(Senha.OperatorName))
        IF UCASE$(RTRIM$(LTRIM$(OpName$))) <> Mes$ THEN
            Mes$ = "ESTA NÃO E' A SUA SENHA !!!"
            CALL Mensagem(Mes$)
            EXIT DO
        END IF
        PointerSenha = RecordSenha
        CALL Saymes(20, 0, 0, "Nome do Operador para esta Senha
: " + Mes$)
        Mes$ = " CONFIRME O CANCELAMENTO DO OPERADOR E SENHA
(S/N) ?"
        Confirme$ = ""
        CALL Getmes(18, 0, 2, Mes$, Confirme$)
        IF Confirme$ = CHR$(27) OR Confirme$ = "" OR
UCASE$(Confirme$) = "N" THEN
            BEEP
            EXIT DO
        ELSE
            OPEN Unit$ + dir$ + "\TEMP.DAT" FOR RANDOM AS #2 LEN
= LEN(SenhaRecord)
            FOR RecordSenha = 1 TO PointerSenha - 1
                GET #1, RecordSenha, Senha
                PUT #2, RecordSenha, Senha
            NEXT RecordSenha
            IF PointerSenha <> NumberOfSenha THEN
                FOR RecordSenha = PointerSenha + 1 TO
NumberOfSenha
                    GET #1, RecordSenha, Senha
                    PUT #2, RecordSenha - 1, Senha
                NEXT RecordSenha
            END IF
            CLOSE #1
            CLOSE #2
            KILL Unit$ + dir$ + "\AIDSENHA.DAT"
```

```
            NAME Unit$ + dir$ + "\TEMP.DAT" AS Unit$ + dir$ +
"\AIDSENHA.DAT"
            EXIT DO
        END IF
    ELSE
        Mes$ = "SENHA NAO CADASTRADA, TENTE NOVAMENTE !!"
        CALL Mensagem(Mes$)
        EXIT DO
    END IF
LOOP
END SUB
```

SUB DirZero

```
'-------------------------------------------------------------------------
'TITULO: DirZero
'ARGUMENTOS: nenhun
'AÇAO: Envia mensagem de diretorio vazio
'
'*************************************************************************
    Mes$ = "NAO EXITE ARQUIVO NO DIRETORIO !!!"
    CALL Mensagem(Mes$)
    CLOSE #2
END SUB
```

SUB EditaCmd (Cmd$, Pline%, Pfile%, Pvideo%)

```
'-------------------------------------------------------------------------
'TITULO: EditaCmd
'ARGUMENTOS: Cmd$, Pline%, Pfile%, Pvideo%
'AÇAO: Edita ultimo comando inserido.
'
'*************************************************************************
VIEW PRINT 4 TO 5: CLS : COLOR 14, 9, 1: VIEW PRINT
LOCATE 4, 2: PRINT "Ultimo comando preparado :"
LOCATE 5, 2: PRINT Cmd$
Mes$ = "Confirme a inclusão da linha no arquivo (S/N) "
Confirme$ = ""
CALL Getmes(22, 0, 0, Mes$, Confirme$)
VIEW PRINT 4 TO 5: CLS : COLOR 14, 9, 1: VIEW PRINT
LOCATE 22, 2: PRINT SPACE$(78)
IF UCASE$(Confirme$) = "S" THEN
    Linein(Pline) = Cmd$
    PUT #1, Pline, Linein(Pline)
    IF Pfile <> Pline THEN
        Pvideo = 20
        Pfile = Pline
    END IF
    IF Pvideo > 19 THEN
        FOR x = 7 TO 19
            LOCATE x, 2: PRINT SPACE$(78)
        NEXT x
        Pvideo = 7
        Pfile = Pline - 12
        DO WHILE NOT EOF(1)
            IF Pvideo > 19 THEN
                EXIT DO
            END IF
```

```
                GET #1, Pfile, Linein(Pfile)
                LOCATE Pvideo, 2: PRINT Linein(Pfile)
                Pfile = Pfile + 1
                Pvideo = Pvideo + 1
            LOOP
            Pfile = Pfile - 1
        ELSE
            LOCATE Pvideo, 2: PRINT Linein(Pline)
            Pvideo = Pvideo + 1
            Pfile = Pfile + 1
        END IF
    Pline = Pline + 1
    END IF
    VIEW PRINT 4 TO 5: CLS : COLOR 14, 9, 1: VIEW PRINT
    END SUB
```

SUB FIDir (Unit$, Dir$)

```
'-------------------------------------------------------------------------------
'TITULO: FIDir
'ARGUMENTOS: Unit$, Dir$
'AÇAO: driver e diretorio doS ARQUIVOS DE DADOS
'
'*******************************************************************
Unit$ = GetDefaultDri$(InRegs, OutRegs, DriveLetter$)
dir$ = GetCurrentDir$(InRegs, OutRegs, DriveLetter$)
IF dir$ = "\" THEN
    dir$ = ""
END IF
END SUB
```

SUB FindBar (File$)

```
'-------------------------------------------------------------------------------
'TITULO: FinBar
'ARGUMENTOS: File$
'AÇAO: Busca "\" no nome do arquivo
'
'*******************************************************************
x = 1
DO WHILE x <> 0
    x = INSTR(1, File$, "\")
    File$ = MID$(File$, x + 1, LEN(File$) - x)
LOOP
END SUB
```

FUNCTION Findpassword% (Pw$, OpName$)

```
'-------------------------------------------------------------------------------
'TITULO:Findpassword
'ARGUMENTOS: Scramble#
'AÇAO: Verifica se a senha e verdadeira.
'
'*******************************************************************
DEFSNG A-Z
BEEP
Findpassword = 0
FOR RecordSenha = 1 TO NumberOfSenha
```

```
    GET #1, RecordSenha, Senha
    IF Senha.OperatorSenha = Scramble(Pw$) THEN
        Findpassword = 1
        OpName$ = Senha.OperatorName
        EXIT FOR
    END IF
NEXT RecordSenha
END FUNCTION
```

SUB FirstSenha

```
'TITULO: FirstSenha
'ARGUMENTOS: Nenhum
'AÇAO: Recebe a primeira Senha do Operador
'
'*************************************************************
SCREEN 0
COLOR 7, 12, 4: CLS
NumberFile = 1
SizeFile = LEN(Senha)
File$ = "AIDSENHA.DAT"
CALL OpenFile(File$, NumberFile, SizeFile)
NumberOfSenha = LOF(1) / SizeFile
IF NumberOfSenha = 0 THEN
    Mes$ = "  *********     PRIMEIRA SENHA     **********"
    CALL Saymes(5, 0, 2, Mes$)
    CALL Addsenha
ELSE
    Tentativa = 0
    DO WHILE Tentativa <> 3
        Mes$ = "DIGITE A SUA SENHA"
        CALL Saymes(10, 0, 2, Mes$)
        Pw$ = "": CALL Getpassword(Pw$)
        IF Findpassword(Pw$, OpName$) = 1 THEN
            COLOR 14, 9, 1: CLS
            CLOSE #1
            EXIT DO
        ELSE
            Tentativa = Tentativa + 1
        END IF
    LOOP
    IF Tentativa = 3 THEN
        COLOR 14, 9, 1: CLS
        CLOSE #1
        END
    END IF
END IF
END SUB
```

FUNCTION GetCurrentDir$ (InRegs AS Regtypex, OutRegs AS Regtypex, DriveLetter$)

```
'TITULO: GetCurrentDir
'ARGUMENTOS: InRegs, OutRegs, DriveLetter
'AÇAO: Retorna a letra do diretório ativo.
'
'*************************************************************
buffer$ = STRING$(64, " ")
```

```
IF DriveLetter$ = "" THEN
    InRegs.dx = 0
ELSE
    InRegs.dx = ASC(UCASE$(DriveLetter$)) - 64
END IF
IF InRegs.dx < 0 OR InRegs.dx > 26 THEN
    InRegs.dx = 0
END IF
InRegs.ax = &H4700
InRegs.ds = VARSEG(buffer$)
InRegs.si = SADD(buffer$)
CALL INTERRUPTX(&H21, InRegs, OutRegs)
IF (OutRegs.flags AND 1) THEN
    GetCurrentDir$ = ""
    EXIT FUNCTION
END IF
GetCurrentDir$ = "\" + LEFT$(buffer$, INSTR(buffer$, " ") - 2)
END FUNCTION
```

FUNCTION GetDefaultDri$ (InRegs AS Regtypex, OutRegs AS Regtypex, DriveLetter$)

```
'-------------------------------------------------------------------------
'TITULO: GetDefaultDri
'ARGUMENTOS: InRegs, OutRegs, DriveLetter
'AÇAO: Retorna o nome do diretorio atual
'
'*************************************************************************
InRegs.ax = &H1900
CALL INTERRUPTX(&H21, InRegs, OutRegs)
GetDefaultDri$ = CHR$((OutRegs.ax AND &HFF) + 65) + ":"
END FUNCTION
```

SUB Getmes (Linha%, coluna%, Tipoborda%, Mes$, Var$)

```
'-------------------------------------------------------------------------
'TITULO: Getmes
'ARGUMENTOS: Linha, Coluna, Tipoborda, Mes$, Var$
'AÇAO: Envia uma mensagem em qualquer parte da tela com borda
simples - 1  borda dupla - 2 ou sem borda - 0, para Coluna = 0 ,
a mensagem e' centralizada e pega a o valor da variavel Var$
'
'*************************************************************************
Linha0 = Linha - 1
Linha1 = Linha + 1
IF coluna = 0 THEN
    Coluna0 = (80 - LEN(Mes$) - LEN(Var$)) / 2 - 2
    Coluna1 = Coluna0 + LEN(Mes$) + LEN(Var$) + 4
    coluna = Coluna0 + 2
ELSE
    Coluna0 = coluna - 2
    Coluna1 = coluna + LEN(Mes$) + LEN(Var$) + 2
END IF
SELECT CASE Tipoborda
    CASE 1
        CALL Borda(Linha0, Coluna0, Linha1, Coluna1, 1)
    CASE 2
        CALL Borda(Linha0, Coluna0, Linha1, Coluna1, 2)
END SELECT
IF Linha = 22 THEN
```

```
    LOCATE 22, 2: PRINT SPACE$(76)
END IF
LOCATE Linha, coluna: PRINT Mes$
LOCATE Linha, coluna + LEN(Mes$): INPUT "", Var$
IF Linha = 22 THEN
    LOCATE 22, 2: PRINT SPACE$(76)
END IF
REM FINAL DO PROCEDIMENTO Getmes
END SUB
```

SUB Getpassword (Pw$)

```
'-------------------------------------------------------------------------------
'TITULO: Getpassword
'ARGUMENTOS: pw$
'AÇÃO: Pega password do teclado sem ecoar na tela.
'
'*******************************************************************************
pa$ = ""
World = 0
DO WHILE pa$ <> CHR$(13)
    pa$ = ""
    WHILE pa$ = "": pa$ = INKEY$: WEND
    IF pa$ = CHR$(8) THEN
        IF LEN(Pw$) > 0 THEN
            Pw$ = LEFT$(Pw$, LEN(Pw$) - 1)
        ELSE
            BEEP
        END IF
    ELSEIF pa$ <> CHR$(13) THEN
        Pw$ = Pw$ + pa$
        World = World + 1
    END IF
LOOP
IF World > 10 THEN
    Mes$ = "MAXIMO DE 10 CARACTERES, VOCE DIGITOU " +
STR$(World) + " CARACTERES !!"
    CALL Mensagem(Mes$)
    Pw$ = ""
END IF
END SUB
```

SUB LimpaVideo

```
'-------------------------------------------------------------------------------
'TITULO: LimpaVideo
'ARGUMENTOS: Nenhum
'AÇÃO: Limpeza do video após a mostra do arquivo
'
'*******************************************************************************
LOCATE 22, 2: PRINT SPACE$(76)
FOR x = 7 TO 19
     LOCATE x, 2: PRINT SPACE$(78)
NEXT
CLOSE #2
ND SUB
```

SUB LIneDown

```
'-------------------------------------------------------------------------------
'TITULO: LineDown
'ARGUMENTOS: Nenhum
'AÇAO: Rola a tela um linha para baixo
'
'*******************************************************************************
IF Pline <= 14 THEN
    EXIT SUB
END IF
FOR x = 7 TO 19
    LOCATE x, 2: PRINT SPACE$(78)
NEXT x
Pvideo = 7
IF Pfile = Pline - 1 THEN
    Pfile = Pline - 13
ELSE
    Pfile = Pfile - 11
END IF
DO WHILE NOT EOF(1)
    IF Pvideo > 19 THEN
        EXIT DO
    END IF
    GET #1, Pfile, Linein(Pfile)
    LOCATE Pvideo, 2: PRINT Linein(Pfile)
    Pfile = Pfile + 1
    Pvideo = Pvideo + 1
LOOP
Pfile = Pfile - 1
END SUB
```

SUB LineUp

```
'-------------------------------------------------------------------------------
'TITULO: LineUp
'ARGUMENTOS: Nenhum
'AÇAO: Rola a tela um linha para cima
'
'*******************************************************************************
IF Pline <= 14 THEN
    EXIT SUB
END IF
FOR x = 7 TO 19
    LOCATE x, 2: PRINT SPACE$(78)
NEXT x
Pvideo = 7
Pfile = Pfile - 13
IF Pfile <= 0 THEN
    Pfile = 1
END IF
DO WHILE NOT EOF(1)
    IF Pvideo > 19 THEN
        EXIT DO
    END IF
    GET #1, Pfile, Linein(Pfile)
    LOCATE Pvideo, 2: PRINT Linein(Pfile)
    Pfile = Pfile + 1
    Pvideo = Pvideo + 1
LOOP
Pfile = Pfile - 1
END SUB
```

SUB M.FindMouse

```
'---------------------------------------------------------------------------
'TITULO: M.FindMouse
'ARGUMENTOS: Nenhum
'AÇÃO: Rotina para verificação da existencia de mouse instalado
'
'***************************************************************************
DEFSNG A-Z
SHARED Mous AS mouseinfo
CALL M.Initialize
Mous.CursorOn = NO
IF Mous.Exists THEN
    Mes$ = " ***  M O U S E   I N S T A L A D O  *** "
    CALL M.MoveCursor(12 * 8, 40 * 8)
ELSE
    Mes$ = " ***  M O U S E   N A O   I N S T A L A D O *** "
END IF
COLOR 14, 9, 1: CLS
CALL Saymes(10, 0, 2, Mes$)
SLEEP 2
COLOR 14, 9, 1: CLS
END SUB
```

SUB M.Flushbuttons

```
'---------------------------------------------------------------------------
'TITULO: M.Flushbuttons
'ARGUMENTOS: Nenhum
'AÇÃO: Reseta o driver interno do mouse para "press" e "release"
' igual a 0
'
'***************************************************************************
    a = 5: b = LEFT: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
    a = 5: b = RIGHT: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
    a = 6: b = LEFT: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
    a = 6: b = RIGHT: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
END SUB
```

SUB M.GetBtnPress (Button%)

```
'---------------------------------------------------------------------------
'TITULO: M.GetBtnPress
'ARGUMENTOS: Button%
'AÇÃO: Rotina para pegar o status do mouse
'
'***************************************************************************
SHARED Mous AS mouseinfo
    a = 5: b = button%: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
    Mous.BtnStatus = a
    Mous.BtnClicks = b
    Mous.Column = c
    Mous.Row = d
```

```
END SUB
```

SUB M.GetBtnRelease (Button%)

```
'-----------------------------------------------------------------------
'TITULO: M.GetBtnRelease
'ARGUMENTOS: Button%
'AÇÃO: Rotina para pegar o status do mouse - release
'
'*****************************************************************
SHARED Mous AS mouseinfo
    a = 6: b = button%: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
    Mous.BtnStatus = a
    Mous.BtnClicks = b
    Mous.Column = c
    Mous.Row = d
END SUB
```

SUB M.GetPos

```
'-----------------------------------------------------------------------
'TITULO: M.GetPos
'ARGUMENTOS: Nenhum
'AÇÃO: Retorna o status do mouse cada vez que e chamada
'
'*****************************************************************
SHARED Mous AS mouseinfo
    a = 3
    CALL M.Mouse(a, b, c, d)
    Mous.BtnStatus = b
    Mous.Column = c
    Mous.Row = d
END SUB
```

SUB M.HideCursor

```
'-----------------------------------------------------------------------
'TITULO: M.HideCursor
'ARGUMENTOS: Nenhum
'AÇÃO: Desliga o cursor se esta ativo
'
'*****************************************************************
SHARED Mous AS mouseinfo
    IF Mous.CursorOn = YES THEN
        a = 2: b = 0: c = 0: d = 0
        CALL M.Mouse(a, b, c, d)
        Mous.CursorOn = NO
    END IF
END SUB
```

```
SUB M.HorizRange (leftcol%, rightcol%)
'-------------------------------------------------------------------------
'TITULO: M.HorizRange
'ARGUMENTOS: LeftCol%, RightCol%
'AÇÃO: Restige o movimento do curso nas colunas definidas
'
'*************************************************************************
    a = 7: b = 0: c = leftcol%: d = rightcol%
    CALL M.Mouse(a, b, c, d)
END SUB


SUB M.Initialize
'-------------------------------------------------------------------------
'TITULO: M.Initialize
'ARGUMENTOS: Nenhum
'AÇÃO: Se o mouse esta presente retorna valor positivo senão
' valor zero
'
'*************************************************************************
SHARED Mous AS mouseinfo
    a = 0: b = 0: c = 0: d = 0
    CALL M.Mouse(a, b, c, d)
    Mous.Exists = a
END SUB


SUB M.Mouse (a%, b%, c%, d%)
'-------------------------------------------------------------------------
'TITULO: M.Mouse
'ARGUMENTOS: a%, b%, c%, d%
'AÇÃO: Rotina de acesso âos registradores do mouse
'
'*************************************************************************
SHARED InRegs AS Regtypex, OutRegs AS Regtypex
InRegs.ax = a
InRegs.bx = b
InRegs.cx = c
InRegs.dx = d
CALL INTERRUPTX(&H33, InRegs, OutRegs)
a = OutRegs.ax
b = OutRegs.bx
c = OutRegs.cx
d = OutRegs.dx
END SUB


SUB M.MoveCursor (RRow%, CCol%)
'-------------------------------------------------------------------------
'TITULO: M.MoveCursor
'ARGUMENTOS: RRow%, CCol%
'AÇÃO: Movimenta o curso para a posição de video especificada
' por Row e Col
'
'*************************************************************************
    a = 4: b = 0: c = CCol%: d = RRow%
    CALL M.Mouse(a, b, c, d)
END SUB
```

SUB M.Movement

```
'-------------------------------------------------------------------------
'TITULO: M.Movement
'ARGUMENTOS: Nenhum
'AÇÃO: Indica o movimento do mouse após uma chamada
'
'*************************************************************************
SHARED Mous AS mouseinfo
    a = 11
    CALL M.Mouse(a, b, c, d)
    Mous.HMovement = c
    Mous.VMovement = d
END SUB
```

SUB M.PenOff

```
'-------------------------------------------------------------------------
'TITULO: M.PenOff
'ARGUMENTOS: Nenhum
'AÇÃO: Desativa a emulação da Light-pen
'
'*************************************************************************
    a = 14
    CALL M.Mouse(a, 0, 0, 0)
END SUB
```

SUB M.PenOn

```
'-------------------------------------------------------------------------
'TITULO: M.PenOn
'ARGUMENTOS: Nenhum
'AÇÃO: Ativa a emulação da Light-pen
'
'*************************************************************************
    a = 13
    CALL M.Mouse(a, 0, 0, 0)
END SUB
```

SUB M.ShowCursor

```
'-------------------------------------------------------------------------
'TITULO: M.ShowCursor
'ARGUMENTOS: Nenhum
'AÇÃO: Ativa o cursor no video
'
'*************************************************************************
SHARED Mous AS mouseinfo
    IF Mous.CursorOn = NO THEN
        a = 1: b = 0: c = 0: d = 0
        CALL M.Mouse(a, b, c, d)
        Mous.CursorOn = YES
    END IF
END SUB
```

```
SUB M.TextCursor (cursortype%, scan1%, scan2%)
'-------------------------------------------------------------------------
'TITULO: M.TextCursor
'ARGUMENTOS: CursorType%, Scan1%, Scan2%
'AÇAO: Ativa o curso para modo texto, tipo do cursor for 0, é
'selecionado o cursor por software. Se o tipo do cursor for 1,
'é selecionado o cursor por hardware
'
'*************************************************************************
    a = 10: b = cursortype%: c = scan1%: d = scan2%
    CALL M.Mouse(a, b, c, d)
END SUB


SUB M.VertRange (upperrow%, lowerrow%)
'-------------------------------------------------------------------------
'TITULO: M.VertRange
'ARGUMENTOS: Upperrow%, Lowerrow%
'AÇAO: Define a linha superior e inferior de movimento do mouse
'
'*************************************************************************
    a = 8: b = 0: c = upperrow%: d = lowerrow%
    CALL M.Mouse(a, b, c, d)
END SUB


SUB M.WaitClick (button%)
'-------------------------------------------------------------------------
'TITULO: M.WaitClick
'ARGUMENTOS: Button%
'AÇAO: Espera por um toque no botão
' se button = LEFT (0) , botão esquerdo
' se button = RIGHT(1) , botão direito
' se button = BOTH (2) , ambos
' se button = EITHER (3), nenhum
'
'*************************************************************************
' DEFINIÇAO DAS CONTANTES
SHARED Mous AS mouseinfo
TIMER ON
Mous.BtnStatus = 0
DO
    M.GetPos
    IF NOT Mous.Exists THEN
        EXIT DO
    END IF
LOOP UNTIL Mous.BtnStatus = 0
IF button < EITHER THEN
    DO
        M.GetPos
        IF NOT Mous.Exists THEN
           EXIT DO
        END IF
    LOOP UNTIL Mous.BtnStatus = (button + 1)
ELSE
    DO
        M.GetPos
        IF NOT Mous.Exists THEN
            EXIT DO
```

```
        END IF
    LOOP UNTIL Mous.BtnStatus > 0
END IF
' Espera pelo botão ser solto
DO
    M.GetPos
    IF NOT Mous.Exists THEN
        EXIT DO
    END IF
LOOP UNTIL Mous.BtnStatus = 0
TIMER OFF
END SUB
```

SUB Mensagem (Mes$)

```
'-------------------------------------------------------------------------
'TITULO: Mensagem
'ARGUMENTOS: Mes$
'AÇAO: Envia uma mensagem para a linha 22 da tela (quadro de
mensagem) e espera por uma tecla para continuar.
'
'*************************************************************************
DEFINT A-Z
coluna = (80 - LEN(Mes$)) / 2
Branco = LEN(Mes$)
LOCATE 22, 2: PRINT SPACE$(76)
CALL Borda(21, 1, 23, 80, 1)
LOCATE 22, coluna: PRINT Mes$
LOCATE 23, 20: PRINT "Digite qualquer tecla para continuar !!!"
k$ = ""
DO WHILE k$ = ""
    k$ = INKEY$
    LOCATE 22, coluna: PRINT SPACE$(Branco)
    FOR T = 0 TO 10000: NEXT T
    COLOR 4, 7, 1
    LOCATE 22, coluna: PRINT Mes$
    COLOR 14, 9, 1
    FOR T = 0 TO 15000: NEXT T
LOOP
COLOR 14, 9, 1
LOCATE 22, coluna: PRINT SPACE$(Branco)
CALL Borda(21, 1, 23, 80, 1)
Mes$ = " Operador: " + RTRIM$(LTRIM$(OpName$)) + " "
CALL Saymes(23, 78 - LEN(Mes$), 0, Mes$)
REM FINAL DA FUNÇAO Mensagem
END SUB
```

FUNCTION Menu% (Linha%, coluna%, Opção$())

```
'-------------------------------------------------------------------------
'TITULO: Menu
'ARGUMENTOS: Linha, Coluna, Opção$()
'AÇAO: Monta um menu horizontal com as opçÕes definas por
Opção$(x,2) e as mensagens de espera de cada um (segunda
dimensão) e retorna em Menu o valor da opção escolhida.
'
'*************************************************************************
DEFINT A-Z
SHARED Mous AS mouseinfo
numop = UBOUND(Opção$)
```

```
sizeop = 0
FOR x = 1 TO numop
    temp = LEN(Opção$(x, 1))
    sizeop = sizeop + temp
NEXT x
IF sizeop > 80 - numop THEN
    Menu = -1
    EXIT FUNCTION
END IF
Spaceop = INT((80 - sizeop - coluna) / (numop - 1))
Colunaop = coluna
FOR x = 1 TO numop
    LOCATE Linha, Colunaop: PRINT Opção$(x, 1)
    Colunaop = Colunaop + LEN(Opção$(x, 1)) + Spaceop
NEXT x
lastcolop = Colunaop - LEN(Opção$(numop, 1)) - Spaceop
Colunaop = coluna
hpos = 1
COLOR 4, 7, 1
LOCATE Linha, Colunaop: PRINT Opção$(hpos, 1)
COLOR 14, 9, 1
LOCATE Linha + 1, coluna: PRINT Opção$(hpos, 2)
IF Mous.Exists THEN
    CALL M.HorizRange(0, 80 * 8)
    CALL M.VertRange((Linha - 1) * 8, (Linha - 1) * 8)
    DO
        CALL M.Flushbuttons
        CALL M.ShowCursor
        CALL M.WaitClick(LEFT)
        CALL M.HideCursor
        IF NOT Mous.Exists THEN
            Menu = 0
            EXIT DO
        END IF
        IF (Mous.Row / 8) + 1 <> Linha THEN
            Menu = 0
            EXIT DO
        END IF
        LOCATE Linha, Colunaop: PRINT Opção$(hpos, 1)
        LOCATE Linha + 1, coluna: PRINT SPACE$(78)
        Fponto = coluna
        ponto = (Mous.Column / 8) + 1
        FOR x = 1 TO numop
            Sponto = Fponto + LEN(Opção$(x, 1))
            IF Fponto <= ponto AND Sponto >= ponto THEN
                hpos = x
                Fpos = hpos
                Colunaop = Fponto
                EXIT FOR
            END IF
            Fponto = Sponto + Spaceop
        NEXT x
        IF x > numop THEN
            Menu = -1
            EXIT DO
        END IF
        COLOR 4, 7, 1
        LOCATE Linha, Colunaop: PRINT Opção$(hpos, 1)
        COLOR 14, 9, 1
        LOCATE Linha + 1, coluna: PRINT Opção$(hpos, 2)
        SLEEP 1
        Menu = hpos
```

```
            EXIT DO
        LOOP
    ELSE
    DO WHILE r$ <> CHR$(13)
        r$ = ""
        DO WHILE r$ = ""
           r$ = INKEY$
        LOOP
        IF LEN(r$) = 2 THEN
            SELECT CASE ASC(RIGHT$(r$, 1))
                CASE 77: REM seta para direita
                    LOCATE Linha, Colunaop: PRINT Opção$(hpos, 1)
                    LOCATE Linha + 1, coluna: PRINT SPACE$(78)
                    hpos = hpos + 1
                    IF hpos > numop THEN
                        Colunaop = coluna
                        hpos = 1
                    ELSE
                        Colunaop = Colunaop + Spaceop + LEN(Opção$
(hpos - 1, 1))
                    END IF
                CASE 75: REM seta para direta
                    LOCATE Linha, Colunaop: PRINT Opção$(hpos, 1)
                    LOCATE Linha + 1, coluna: PRINT SPACE$(78)
                    hpos = hpos - 1
                    IF hpos = 0 THEN
                        hpos = numop
                        Colunaop = lastcolop
                    ELSE
                        Colunaop = Colunaop - Spaceop - LEN(Opção$
(hpos, 1))
                    END IF
            END SELECT
        ELSE
        'Busca a 1. letra da opção
            IF (r$ >= "A" AND r$ <= "Z") OR (r$ >= "a" AND r$ <=
"z") THEN
                x = 1
                DO WHILE x <= numop
                    Optemp$ = LEFT$(Opção$(x, 1), 1)
                    IF r$ = Optemp$ OR ASC(r$) - 32 = ASC(Optemp$)
THEN
                        LOCATE Linha, Colunaop: PRINT Opção$(hpos,
1)
                        LOCATE Linha + 1, coluna: PRINT SPACE$(78)
                        Colunaop = coluna
                        FOR xy = 1 TO x - 1
                            Colunaop = Colunaop + LEN(Opção$(xy, 1))
+ Spaceop
                        NEXT xy
                        IF hpos <> xy THEN
                           hpos = xy
                           EXIT DO
                        END IF
                    END IF
                    x = x + 1
                LOOP
                IF x > numop THEN
                    BEEP
                END IF
            END IF
            IF r$ = CHR$(27) THEN
```

```
                Menu = 0
                EXIT DO
            END IF
        END IF
        COLOR 4, 7, 1
        LOCATE Linha, Colunaop: PRINT Opçâo$(hpos, 1)
        COLOR 14, 9, 1
        LOCATE Linha + 1, coluna: PRINT Opçâo$(hpos, 2)
        Menu = hpos
LOOP
END IF
REM FINAL DA FUNÇAO Menu
END FUNCTION
```

SUB MostraDir (PointerDir%)

```
'-------------------------------------------------------------------------------
'TITULO: MostraDir
'ARGUMENTOS: PointerDir
'AÇAO: Mostra lista de diretório no video
'
'*******************************************************************************
DEFINT A-Z
SHARED Mous AS mouseinfo
FOR x = 7 TO 19
    LOCATE x, 2: PRINT SPACE$(78)
NEXT x
LinhaUp = 7
LinhaDo = 19
PointerDir = 1
NumberOfDir = LOF(2) / LEN(TabDir)
FOR x = LinhaUp TO LinhaDo
    GET #2, PointerDir, TabDir
    LOCATE x, 2: PRINT RTRIM$(LTRIM$(TabDir.OperatorName)),
TabDir.FileName, TabDir.FileDes
    PointerDir = PointerDir + 1
    IF PointerDir > NumberOfDir THEN
        EXIT FOR
    END IF
NEXT x
'
'------- MOVIMENTAÇAO DO CURSOR E ESCOLHA DO ARQUIVO ----------
hpos = LinhaUp
PointerDir = 1
GET #2, PointerDir, TabDir
COLOR 4, 7, 1
LOCATE hpos, 2: PRINT RTRIM$(LTRIM$(TabDir.OperatorName)),
TabDir.FileName, TabDir.FileDes
COLOR 14, 9, 1
IF Mous.Exists THEN
    LOCATE 6, 80: PRINT CHR$(24)
    FOR x = 7 TO 19
        LOCATE x, 80: PRINT CHR$(177)
    NEXT x
    LOCATE 20, 80: PRINT CHR$(25)
    CALL M.HorizRange(1 * 8, (80 - 1) * 8)
    CALL M.VertRange((6 - 1) * 8, (20 - 1) * 8)
    DO
        CALL M.Flushbuttons
        CALL M.ShowCursor
        CALL M.WaitClick(LEFT)
```

```
            CALL M.HideCursor
            LOCATE hpos, 2: PRINT
    LTRIM$(RTRIM$(TabDir.OperatorName)), TabDir.FileName,
    TabDir.FileDes
            hponto = (Mous.Row / 8) + 1
            DeltaPonto = hponto - hpos
            hpos = hponto
            PointerDir = PointerDir + DeltaPonto
            IF PointerDir > NumberOfDir THEN
                PointerDir = 1
                hpos = LinhaUp
            END IF
            IF hpos = LinhaDo + 1 THEN
                hpos = LinhaDo
                FOR x = LinhaUp TO LinhaDo
                    GET #2, PointerDir - (LinhaDo - x), TabDir
                    LOCATE x, 2: PRINT RTRIM$(LTRIM$ (TabDir
    .OperatorName)), TabDir.FileName, TabDir.FileDes
                NEXT x
            END IF
            IF hpos = LinhaUp - 1 THEN
                hpos = LinhaUp
                PointerDir = PointerDir - (DeltaPonto) + 1
                FOR x = LinhaUp TO LinhaDo
                    GET #2, PointerDir + (x - LinhaUp), TabDir
                    LOCATE x, 2: PRINT RTRIM$(LTRIM$(TabDir
    .OperatorName)), TabDir.FileName, TabDir.FileDes
                NEXT x
            END IF
            GET #2, PointerDir, TabDir
            COLOR 4, 7, 1
            LOCATE hpos, 2: PRINT RTRIM$(LTRIM$(TabDir
    .OperatorName)), TabDir.FileName, TabDir.FileDes
            COLOR 14, 9, 1
            EXIT DO
        LOOP
        CALL Borda(6, 1, 20, 80, 1)
    ELSE
    DO WHILE r$ <> CHR$(13)
        r$ = ""
        DO WHILE r$ = ""
            r$ = INKEY$
        LOOP
        IF LEN(r$) = 2 THEN
            SELECT CASE ASC(RIGHT$(r$, 1))
            CASE 80 ' SETA PRA BAIXO
                LOCATE hpos, 2: PRINT LTRIM$(RTRIM$(TabDir
    .OperatorName)), TabDir.FileName, TabDir.FileDes
                IF PointerDir < NumberOfDir THEN
                    hpos = hpos + 1
                    PointerDir = PointerDir + 1
                END IF
                IF hpos = LinhaDo + 1 THEN
                    hpos = LinhaDo
                    FOR x = LinhaUp TO LinhaDo
                        GET #2, PointerDir - (LinhaDo - x), TabDir
                        LOCATE x, 2: PRINT RTRIM$(LTRIM$(TabDir
    .OperatorName)), TabDir.FileName, TabDir.FileDes
                    NEXT x
                END IF
            CASE 72      ' SETA PARA CIMA
```

```
            LOCATE hpos, 2: PRINT RTRIM$(LTRIM$(TabDir
.OperatorName)), TabDir.FileName, TabDir.FileDes
            IF PointerDir <> 1 THEN
                hpos = hpos - 1
                PointerDir = PointerDir - 1
            END IF
            IF hpos = LinhaUp - 1 THEN
                hpos = LinhaUp
                FOR x = LinhaUp TO LinhaDo
                    GET #2, PointerDir + (x - LinhaUp), TabDir
                    LOCATE x, 2: PRINT RTRIM$(LTRIM$(TabDir
.OperatorName)), TabDir.FileName, TabDir.FileDes
                NEXT x
            END IF
        CASE 71              'HOME
            LOCATE hpos, 2: PRINT LTRIM$(RTRIM$(TabDir
.OperatorName)), TabDir.FileName, TabDir.FileDes
            PointerDir = 1
            FOR x = LinhaUp TO LinhaDo
                GET #2, PointerDir, TabDir
                LOCATE x, 2: PRINT RTRIM$(LTRIM$(TabDir
.OperatorName)), TabDir.FileName, TabDir.FileDes
                PointerDir = PointerDir + 1
                IF PointerDir > NumberOfDir THEN
                    EXIT FOR
                END IF
            NEXT x
            hpos = LinhaUp
            PointerDir = 1
        CASE 79              'END
            PointerDir = NumberOfDir - (LinhaDo - LinhaUp)
            IF PointerDir <= 0 THEN
                PointerDir = 1
            END IF
            LOCATE hpos, 2: PRINT LTRIM$(RTRIM$(TabDir
.OperatorName)), TabDir.FileName, TabDir.FileDes
            FOR x = LinhaUp TO LinhaDo
                GET #2, PointerDir, TabDir
                LOCATE x, 2: PRINT RTRIM$(LTRIM$(TabDir
.OperatorName)), TabDir.FileName, TabDir.FileDes
                PointerDir = PointerDir + 1
                IF PointerDir = NumberOfDir THEN
                    EXIT FOR
                END IF
            NEXT x
            hpos = x + 1
        CASE 75, 77
            PointerDir = 0
            FOR x = 7 TO 19
                LOCATE x, 2: PRINT SPACE$(78)
            NEXT x
            EXIT DO
        CASE ELSE
            BEEP
        END SELECT
    ELSE
        IF r$ = CHR$(27) THEN
            PointerDir = 0
            FOR x = 7 TO 19
                LOCATE x, 2: PRINT SPACE$(78)
            NEXT x
            EXIT DO
```

```
        ELSE
            IF r$ <> CHR$(13) THEN
                BEEP
            END IF
        END IF
    END IF
    GET #2, PointerDir, TabDir
    COLOR 4, 7, 1
    LOCATE hpos, 2: PRINT RTRIM$(LTRIM$(TabDir.OperatorName)),
TabDir.FileName, TabDir.FileDes
    COLOR 14, 9, 1
LOOP
END IF
END SUB
```

SUB OpenFile (File$, NumberFile, SizeFile)

```
'-------------------------------------------------------------------------------
'TITULO: OpenFile
'ARGUMENTOS: unidadde,diretorio e arquivo
'AÇAO: Define e abre o arquivo de senha.
'
'*******************************************************************
Unit$ = GetDefaultDri$(InRegs, OutRegs, DriveLetter$)
dir$ = GetCurrentDir$(InRegs, OutRegs, DriveLetter$)
IF dir$ = "\" THEN
   OPEN Unit$ + dir$ + File$ FOR RANDOM AS #NumberFile LEN =
SizeFile
ELSE
    OPEN Unit$ + dir$ + "\" + File$ FOR RANDOM AS #NumberFile
LEN = SizeFile
END IF
END SUB
```

SUB OpenSeqFile (File$, NumberFile%, SizeFile%)

```
'-------------------------------------------------------------------------------
'TITULO: OpenSeqFile
'ARGUMENTOS: unidadde,diretorio e arquivo
'AÇAO: Define e abre o arquivo de entrada sequencial.
'
'*******************************************************************
Unit$ = GetDefaultDri$(InRegs, OutRegs, DriveLetter$)
dir$ = GetCurrentDir$(InRegs, OutRegs, DriveLetter$)
IF NumberFile = 10 THEN
    IF dir$ = "\" THEN
        OPEN Unit$ + dir$ + File$ FOR INPUT AS #NumberFile LEN =
SizeFile
    ELSE
        OPEN Unit$ + dir$ + "\" + File$ FOR INPUT AS #NumberFile
LEN = SizeFile
    END IF
ELSE
    IF dir$ = "\" THEN
        OPEN Unit$ + dir$ + File$ FOR OUTPUT AS #NumberFile LEN
= SizeFile
    ELSE
        OPEN Unit$ + dir$ + "\" + File$ FOR OUTPUT AS
#NumberFile LEN = SizeFile
    END IF
```

```
END IF
END SUB
```

SUB QBDir (UnitQB$, DirQB$)

```
'-------------------------------------------------------------------------------
'TITULO: QBDir
'ARGUMENTOS: UnitQB$, DirQB$
'AÇAO: Retorna a unidade e diretório do ambiente
'
'*******************************************************************
UnitQB$ = GetDefaultDri$(InRegs, OutRegs, DriveLetter$)
DirQB$ = GetCurrentDir$(InRegs, OutRegs, DriveLetter$)
IF DirQB$ = "\" THEN
    DirQB$ = ""
END IF
END SUB
```

SUB Roterror (Ferro%)

```
'-------------------------------------------------------------------------------
'TITULO: Roterror
'ARGUMENTOS: Ferro%
'AÇAO: Subrotina de apresentação de erros
'
'*******************************************************************
    SCREEN 0, 7, 0, 0: COLOR 14, 9, 1
    Ferro = TRUE
    DIM Error$(76)
    Error$(3) = "RETURN without GOSUB"
    Error$(5) = "Illegal function call"
    Error$(6) = "Overflow"
    Error$(7) = "Out of memory"
    Error$(9) = "Subscript out of range"
    Error$(11) = "Divion by 0"
    Error$(14) = "Out of string space"
    Error$(16) = "String formula too complex"
    Error$(19) = "No RESUME"
    Error$(20) = "RESUME without error"
    Error$(24) = "Device timeout"
    Error$(25) = "Device fault"
    Error$(27) = "Out of paper"
    Error$(39) = "CASE ELSE expected"
    Error$(50) = "FIELD overflow"
    Error$(51) = "Internal error"
    Error$(52) = "Bad file name or number"
    Error$(53) = "File not found"
    Error$(54) = "Bad file mode"
    Error$(55) = "File already open"
    Error$(56) = "FIELD statement active"
    Error$(57) = "Device I/O error"
    Error$(58) = "File already exists"
    Error$(59) = "Bad record lenght"
    Error$(61) = "Disk full"
    Error$(62) = "Input past end of file"
    Error$(63) = "Bad record number"
    Error$(64) = "Bad file name"
    Error$(67) = "Too many files"
    Error$(68) = "Device unavalible"
    Error$(69) = "Commnucations buffer overflow"
```

```
    Error$(70) = "Permission denied"
    Error$(71) = "Disk not ready"
    Error$(72) = "Disk media error"
    Error$(73) = "Advanced feature unavalible"
    Error$(74) = "Rename across disks"
    Error$(75) = "Path/File access error"
    Error$(76) = "Path not found"
    IF LEN(Error$(ERR)) > 0 THEN
        Error$(1) = Error$(ERR)
    ELSE
        Error$(1) = "ERRO NAO CATALOGADO"
    END IF
    Mes$ = "ERRO " + STR$(ERR) + " -> " + Error$(1)
    CALL Mensagem(Mes$)
    ERASE Error$
END SUB
```

SUB Saymes (Linha%, coluna%, Tipoborda%, Mes$)

```
'-------------------------------------------------------------------------
'TITULO: Saymes
'ARGUMENTOS: Linha, Coluna, Tipoborda, Mes$
'AÇÃO: Envia uma mensagem em qualquer parte da tela com borda
'simples - 1 borda dupla - 2 ou sem borda - 0, para Coluna = 0 ,
'a mensagem e'centralizada.
'
'*************************************************************************
Linha0 = Linha - 1
Linha1 = Linha + 1
IF coluna = 0 THEN
    Coluna0 = (80 - LEN(Mes$)) / 2 - 2
    Coluna1 = Coluna0 + LEN(Mes$) + 4
    coluna = Coluna0 + 2
ELSE
    Coluna0 = coluna - 2
    Coluna1 = coluna + LEN(Mes$) + 2
END IF
SELECT CASE Tipoborda
    CASE 1
        CALL Borda(Linha0, Coluna0, Linha1, Coluna1, 1)
    CASE 2
        CALL Borda(Linha0, Coluna0, Linha1, Coluna1, 2)
END SELECT
IF Linha = 22 THEN
    LOCATE 22, 2: PRINT SPACE$(76)
END IF
LOCATE Linha, coluna: PRINT Mes$
'
REM FINAL DO PROCEDIMENTO Saymes
END SUB
```

FUNCTION Scramble# (Pw$)

```
'_____________________________________________________________________
'TITULO: Scramble
'ARGUMENTOS: pw$
'AÇÃO: Produz um valor numerico relativo a senha digitada
'
'*********************************************************************
DEFSNG A-Z
RANDOMIZE TIMER
length = LEN(Pw$)
pws# = 0
 FOR x = 1 TO length
     pws = x * pws XOR ASC(CHR$(x + 30)) XOR ASC(MID$(Pw$, x,
1))
NEXT x
Scramble = pws
END FUNCTION
```

FUNCTION SetDefaultDri (InRegs AS Regtypex, OutRegs AS Regtypex, DriveLetter$)

```
'_____________________________________________________________________
'TITULO: SetDefaultDri
'ARGUMENTOS: InRegs, OutRegs, DriveLetters%
'AÇÃO: Ativa a unidade selecionada
'
'*********************************************************************
DEFINT A-Z
InRegs.ax = &HE00
InRegs.dx = ASC(UCASE$(DriveLetter$)) - 65
IF InRegs.dx < 0 OR InRegs.dx > 26 THEN
    SetDefaultDri = 0
    EXIT FUNCTION
END IF
CALL INTERRUPTX(&H21, InRegs, OutRegs)
SetDefaultDri = OutRegs.ax AND &HFF
END FUNCTION
```

A N E X O 04

SISTEMA DE COMUNICAÇAO IEEE-488/78 (20)

1. INTRODUÇAO

O objetivo da NORMA IEEE-488/78 é o de possibilitar a interligação de instrumentos programáveis de diferentes tipos e fabricantes, evitando a necessidade de adaptadores e numerosos tipos de cabos de ligação.

A norma provê um conjunto de regras para estabelecer um elo de COMUNICAÇAO entre instrumentos e controladores, com alto grau de confiabilidade e flexibilidade.

2. RESUMO DA NORMA IEEE-488/78

Este sistema pode ser interpretado como sendo um elo de COMUNICAÇAO entre dois ou mais instrumentos. Este elo de ligação é um barramento onde a transmissão é do formato "byte"-serial, "bit"-paralelo. "Bit"-paralelo refere-se a um conjunto de "bits" de dados sendo transmitidos simultaneamente, e "byte"-serial refere-se às palavras consecutivas transportadas através do elo de dados, na forma serial. O barramento proposto consiste de 16 linhas de transmissão que estão caracterizadas como:

- 8 Linhas de Dados (DIO1-DIO8);

- 3 Linhas de Controle de Transferência de Dados ("Handshake") (DAV,NDAC,NRFD);

- 5 Linhas de Sinais de Gerenciamento Geral da Interface (ATN,IFC,REN,SRQ e EOI).

As 8 linhas de dados são utilizadas para transferir dados entre equipamentos Falantes ("TALKER") e Ouvintes ("LISTENER"). Elas também são utilizadas para transferir comandos do controlador para os demais instrumentos.

Toda transmissão é assíncrona e ocorre de acordo com as Linhas de "Handshake", exceto no Modo Votação (apresentação) paralela ("Parallel Poll").

São com as Linhas de "Handshake" que o Falante ou Controlador sincroniza sua disposição para transmitir dados e com a disposição do Ouvinte para recebê-los.

A qualquer instante de tempo, um dispositivo individual poderá monitorar a atividade do barramento em uma das seguintes situações:

- um falante enviando dados para ouvintes no barramento; ou

- um ouvinte (Controlador) recebendo dados de um falante.

Muitos dispositivos são tanto falantes quanto ouvintes. Um multímetro programavel, por exemplo, será um ouvinte quando estiver recebendo suas instruções de programação e um falante quando estiver enviando dados a outros dispositivos, tais como uma impressora ou unidade de disco.

Pode haver mais de um Ouvinte ao mesmo tempo, mas somente um Falante. Um sistema mínimo não precisa conter um controlador mas pode consistir de somente um Falante e um Ouvinte, como exemplo, um voltímetro dedicado poderia enviar dados para uma impressora. Em tal sistema, serão necessárias para os dois dispositivos, opções de interfaceamento que irão permitir o tráfego de mensagens entre os mesmos e a configuração individual como Falante ou Ouvinte.

A configuração será feita mais freqüentemente quando da inicialização do sistema e não será alterada.

Controladores podem ser omitidos somente em sistemas dedicados, isto é, sistemas onde as funções de Falante ou Ouvinte não são automaticamente reconfiguradas.

O controlador alterna a atividade do barramento pelo envio de comandos ou dados pela interface. É o único dispositivo capaz de enviar comandos de duas maneiras distintas.

a) Mensagens UNILINHA: o controlador pode enviar um comando por uma das 5 linhas de controle, como por exemplo IFC, REN, etc.

b) Mensagens MULTILINHA: o controlador pode enviar um comando pelas 8 linhas de dados, sinalizando através da linha de controle ATN que as 8 linhas de dados contêm um comando multilinha ou um endereço ao invés de dados.

Estas mensagens são comandos de interface e não interagem diretamente com o processo de medição do instrumento.

O propósito primário destas mensagens é executar o protocolo adequado para a inicialização, manutenção e término de um fluxo ordenado de mensagens dependentes dos dispositivos.

Mensagens dependentes do dispositivo referem-se às informações enviadas pelo dispositivo endereçado como ouvinte e não às mensagens utilizadas para controle de interface. As mensagens Multilinha e Unilinha são utilizadas para:

- Endereçar dispositivos como Falantes ou Ouvintes;

- Avisar um instrumento no sentido de ignorar ou não os ajustes do painel frontal;

- Inquerir sobre algum problema que o dispositivo tenha;

- Reinicializar ("Reset") o circuito da interface;

- Inicializar uma medição e outras funções.

Os comandos podem ser categorizados como Comandos Universais, Comandos Endereçados, Comandos Universais Endereçados ou Comandos Secundários.

Endereços são atribuídos a cada dispositivo para que eles possam responder a Comandos Endereçados. Usando este comando, o controlador pode selecionar um dispositivo e instruí-lo para ser um Falante ou um Ouvinte.

Um Falante envia uma palavra de dados para um Ouvinte ou Ouvintes utilizando as 3 linhas assíncronas de "Handshake". A transferência inicia-se quando o falante torna o dado disponível (DAV) e é completada quando o Ouvinte mais lento recebe a palavra de dado (DAC). A terceira linha (NRFD) é utilizada para avisar o Falante que o Ouvinte está pronto para receber dados.

Desde que possa haver muitos Ouvintes (máximo de 14) é possível que alguns respondam mais rápido (exemplo uma unidade de disco) e outros vagarosamente (exemplo um teletipo ou telex) para uma mesma palavra de dados. Logo, a velocidade final de transmissão pelo barramento será estabelecida pela taxa de resposta do ouvinte mais lento.

O objetivo da Norma IEEE-488 é proporcionar um conjunto de regras que garantam a compatibilidade e, por outro lado, permitam uma liberdade de projeto de modo a ajustar um instrumento às suas necessidades específicas.

Para isto, a norma foi desenvolvida em termos das funções da interface (distintas das funções do instrumento), das mensagens e dos diagramas de estado descrevendo o comportamento de cada uma dessas funções. A operação de um instrumento é dividida em dois conjuntos de funções:

- Funções da interface: garantem o comportamento correto do instrumento com respeito às linhas de sinais do barramento IEEE-488;

- Funções do dispositivo: utilizadas para controlar o instrumento específico, como por exemplo, programar uma função de um contador universal (freqüência, período, etc.).

3. FUNÇÕES NORMALIZADAS

A Norma IEEE-488 estabelece um número de 10 funções que devem ser implementadas pela interface, descritas a seguir:

- "Source Handshake" (SH)

Provê a um dispositivo a capacidade de transferência de mensagens Multilinha transmitidas assincronamente pelo barramento. Controla o início e o término de cada transmissão utilizando-se das linhas de "Handshake": DAV, NRFD e DAC.

- "Talker" (T e TE)

Provê a um dispositivo a capacidade de enviar dados (incluindo o "status" durante uma seqüência de "Serial Poll") através da interface para outros dispositivos.

Pode ser implementada em duas versões: "Talker" Normal (T), com um "byte" de endereçamento e "Talker" Estendido (TE), com dois "bytes" de endereçamento. As capacidades em ambas são as mesmas.

- **"Listener" (L e LE)**

Provê a um dispositivo a capacidade de receber dados (incluindo o "status") através da interface de outros dispositivos.

Pode ser implementada em duas versões: "Listener" Normal (L), com um "byte" de endereçamento e "Listener" Estendido (LE), com dois "bytes" de endereçamento. As capacidades em ambas são as mesmas.

- **"Service Request" (SR)**

Provê a um dispositivo a capacidade de requerer serviços assincronamente de um controlador quando operando sobre a interface.

- **"Remote/Local" (RL)**

Provê a um dispositivo a capacidade para selecionar entre duas fontes de entrada de informação (Remota ou Local), qual delas será utilizada.

- **"Parallel Poll" (PP)**

Provê a um dispositivo a capacidade de se apresentar quando um "Parallel Poll" é enviado pelo controlador.

- **"Device Clear" (DC)**

Provê a um dispositivo a capacidade de se inicializar individualmente ou como parte de um grupo.

- **"Device Trigger" (DT)**

Provê a um dispositivo a capacidade de iniciar e executar uma atividade básica individualmente ou como parte de um grupo.

- **"Controller" (C)**

Provê a um dispositivo a capacidade de enviar endereços, comandos universais e comandos endereçados a outro dispositivo através da interface.

A N E X O 05

LINGUAGEM DE PROGRAMAÇAO QuickBASIC (2a-3a)

1 - COMANDOS E DECLARAÇÕES DO QUICKBASIC (QB)

BEEP

Emite um breve sinal sonoro. O mesmo que PRINT CHR$(7).

BLOAD <"arquivo">(,deslocamento)

Carrega para a memória um arquivo imagem de memória, criado BSAVE, a partir do deslocamento especificado. Se o segmento não foi especificado através de um DEF SEG o segmento de default (DS) do QB é usado.

BSAVE <"arquivo">,<deslocamento>,<tamanho>

Grava imagem de memória começando no deslocamento e com o tamanho no arquivo especificado. Se o segmento não foi especificado através de um default (DS) do QB é usado.

CALL <nome> (argumentos)

Chama e transfere o controle para um sub-programa em QB. Os argumentos são nomes das variáveis que serão usadas no módulo e são transferidos como endereços. A palavra CALL é opcional se o procedimento foi declarado anteriormente com um DECLARE, nesse caso omite-se os parênteses em argumentos.

CALL <nome> (BYVAL/SEG) argumentos

Chama e transfere o controle para procedimento escrito em outra linguagem que não o BASIC. BYVAL passa os argumentos como valores, SEG passa o endereço completo segmentado.

CALL ABSOLUTE (argumentos, desloc)

Chama e transfere o controle para um sub-programa em Assembler cujo deslocamento está contido em desloc e cujo segmento foi previamente definido em um DEF SEG. Os argumentos são nomes de variáveis que serão usadas no módulo chamado e são passados como deslocamentos em relação ao segmento default do QB.

CALLS <nome> (argumentos)

Chama e transfere o controle para uma sub-rotina em outra linguagem que não o QB. Os argumentos são passados como endereços completos (segmento e deslocamento).

CALL INTERRUPT <(int, regentrada, regsaida)>

Chama a interrupção do processador definida em int; regentrada é uma variável estrutural (RegType), definindo o conteúdo dos registradores na entrada; regsaida os contém na saída. A definição do RegType:

```
TYPE RegType
     AX      AS INTEGER
     BX      AS INTEGER
     CX      AS INTEGER
     DX      AS INTEGER
     BP      AS INTEGER
     SI      AS INTEGER
     DI      AS INTEGER
     FLAGS   AS INTEGER
END TYPE
```

CALL INTERRUPTX <(int, regentrada, regsaida)>

Igual a CALL INTERRUPT só que também usa o conteúdo dos registradores DS e ES adicionados a estrutura acima.

CHAIN <"programa">

Transfere controle para um outro programa passando eventuais variáveis do programa corrente através da instrução COMMON.

CHDIR <"path">

Muda o diretório corrente para path.

CIRCLE (STEP) (x,y), <raio> (cor,início, fim,aspécto)

Desenha um círculo, uma elipse ou um arco, onde STEP: especifica coordenadas x,y como relativas ao cursor gráfico; x,y: coordenadas do centro da figura; raio: raio da figura; cor: código da cor; início: ângulo inicial em radianos; fim: ângulo final em radianos; aspecto: proporção entre o raio horizontal e vertical.

CLEAR (m)

Fecha todos os arquivos, limpa todas as variáveis do COMMON, reseta todas as variáveis e matrizes numéricas para zero, reseta o "stack" e opcionalmente, altera seu tamanho, reseta todas as variáveis alfanuméricas para nulo e libera todos os "buffers" de disco.

CLOSE (#<n1>,#<n2>,...)

Encerra acesso a arquivos ou dispositivos. Sem nenhum parâmetro fecha todos os arquivos e dispositivos.

CLS (0,1,2)

Limpa o vídeo. Sem argumentos limpa a janela ativa. 0 - limpa tela de texto e gráfico, 1 - limpa tela gráfica, 2 - tela de texto.

COLOR (letra,fundo,borda)

Define as cores das letras, do fundo e da borda no modo texto em SCREEN 0 do CGA, EGA, VGA, MCGA

COLOR (fundo, palheta)

Define a cor de fundo e da palheta no modo gráfico e modo texto 40x25 em SCREEN 1, CGA, EGA, VGA, MCGA.

COLOR (primeiro-plano,fundo)

Define a cor do primeiro plano e do fundo nos modos SCREEN 7 - 10, do EGA, VGA e MCGA.

COLOR (primeiro-plano)
Define a cor do primeiro plano em SCREEN 11 - 13, no VGA e MCGA.

COM (n) ON/OFF/STOP
Ativa (ON), desativa (OFF) ou suspende (STOP) captação de eventos nos dispositivos seriais (n=1,2).

COMMON (SHARED) (nome,grupo) <var AS tipo>, ...
Passa variáveis ou grupo de variáveis do programa corrente para programas executados por CHAIN ou para outros módulos do mesmo programa. Variáveis tipo matriz são especificadas com () ao final de var. Onde, SHARED: dispensa o uso de "shared" no procedimento executado; nome-grupo: identifica um grupo de variáveis. Não usar com CHAIN; AS tipo: tipo da variável (INTEGER,LONG,SINGLE,DOUBLE, STRING, ou tipo definido pelo usuário).

CONST <nome>=<expressão>(nome=expressão)...
Define constantes simbólicas a serem usadas no lugar de valores numéricos ou alfanuméricos.

DATA <constante>, (constante 2)
Armazena uma lista de constantes que serão acessadas pela declaração READ.

DATE$ = x$
Seta a data do sistema(x$ = mm-dd-(aa)aa ou mm/dd(aa)aa).

DECLARE FUNCTION/SUB <nome> (var AS tipo)...
Declara referências a procedimentos do QB e força a verificação da correção dos argumentos. Dispensa o uso de CALL na execução de procedimento.

DECLARE FUNCTION/SUB <nome> (ALIAS "nome")(BYVAL/SEG/ var AS tipo).
Declara seguência de chamada para procedimentos escritos em outra linguagem. FUNCTIO/SUB: se o procedimento externo retorna ou não um valor; CDECL: o procedimento usa convenção C de passagem de parâmetros; ALIAS: indica o procedimento tem nomes no módulo OBJ ou biblioteca; BYVAL/SEG: se a variável é passada pelo seu valor ou pelo seu endereço; AS tipo: o tipo da variável. Pode ser ANY, que é um tipo não especificado.

DECLARE FUNCTION SenoHip CDECL ALIAS "hipisin" (BYVAL Angulo AS DOUBLE)

DEF FN<nome> (argumentos) = <expressão>
Define uma função numérica ou string (em uma única linha).

DEF FN<nome> (var(AS tipo)...)...<FNnome = expressão>... END DEF
Define uma função que retorna um valor numérico ou string podendo usar diversas linhas.

DEF SEG (=<endereço>)
Define o endereço do segmento de memória que vai ser usado subseguentemente em uma função PEEK, ou em um comando BLOAD, BSAVE, CALL ABSOLUTE ou POKE (default = segmento de dados do QB).

DEFDBL letra (-letra)...
Define como dulpa precisão as variáveis que começam com as letras especificadas.

DEFINIT letra1 (-letra2)...
Define como inteiras longas as variáveis que começam com as letras especificadas.

DEFSNG letra1 (-letra2)...
Define como simples precisão as variáveis que começam com as letras especificadas.

DEFSTR letra1 (-letra2)...
Define como string as variáveis que começam com as letras especificadas.

DIM(SHARED) <var> (lim TO lim)(,lim TO lim,...)(AStipo)(,var...)
Declara e aloca memória para uma variável. SHARED: permite todas as procedures de um módulo compartilhar as matrizes e variáveis declaradas; lim TO lim: declara os limites inferior e superior do subscrito;AS tipo: Declara o tipo da variável.

DO (comandos) LOOP (WHILE/UNTIL <cond>)
Repete um bloco de comandos enquanto uma condição for verdadeira (WHILE) ou até que uma condição se torne verdadeira (UNTIL). A condição é verificada após a execução do bloco de comandos.

DO (WHILE/UNTIL <cond> (comandos) LOOP
Repete um bloco de comandos enquanto uma condição for verdadeira (WHILE) ou até que uma condição se torne verdadeira (UNTIL). A condição é verificada antes da execição do bloco de comandos.

DRAW <"string">
Desenha uma figura no vídeo, conforme os macrocomandos em string.

END (DEF/FUNCTION/SELECT/SUB/TYPE)
END: encerra programa, fecha arquivos e retorna ao DOS; END DEF: encerra uma definição DEF FN; FUNCTION: encerra um procedimento FUNCTION; END IF: encerra um bloco de IF..THEN..ELSE; END SELECT: encerra um bloco SELECT CASE; END SUB: encerra um procedimento SUB; END TYPE: encerra uma definição de tipo definido pelo usuário.

ENVIRON <"parâmetro = texto">
Modifica, inclui ou elimina um parâmetro na área "environment".

ERASE <matrizes>
Apaga os elementos de uma matriz estática; se dinâmica,ela deixa de existir. Poderá ser redefinida com DIM ou REDIM.

ERROR <código do erro>
Simula a ocorrência de um erro do BASIC, ou permite ao usuário definir um código de erro.

EXIT DEF/DO/FOR/FUNCTION/SUB
DEF: abandona uma função DEF FN; DO: abandona um laço DO...LOOP;FOR: abandona um laço FOR...NEXT; FUNCTION: abandona um procedimento FUNCTION; SUB: abandona um procedimento SUB.

FIELD(#)<n>,<tamanho>AS<variável$>,...
Aloca espaço para variáveis em um "buffer" de arquivo aleatório. n: número usado na declaração OPEN; tamanho: largura do campo no registro; AS variável: designa nome de um subcampo no registro.

FILES ("arquivo")
Mostra a relação de arquivos na especificação "arquivo".

FOR <variável> = <x> TO <y> (STEP z)
Define o início de um laço que será executado (y-x)/z vezes, desde o valor inicial x até o valor final y, com o incremento z.

FUNCTION nome (var() AStipo,...) (STATIC)
... nome = expressão ...

END FUNCTION
Declara o nome, os parâmetros e expressões que formam uma função. STATIC: as variáveis locais devem ser mantidas entre as chamadas; expressão: o valor de retornado pela função.

GET(#)<n> (,nunreg)(,var)
Transfere um registro nunreg aleatório para variável var ou para buffer. Pode ser usado para ler porta de comunicação.

GET (STEP)<x1,y1)>-(STEP)<(x2,y2)>,<matriz> (índices)
Lê os pontos de uma área do vídeo, armazenando-os na matriz numérica. x1,y1,x2.y2: coordenadas da diagonal da janela; STEP: as coordenadas são relativas ao cursor gráfico; índice: elemento da matriz onde começar o armazenamento.

GOSUB<linha1/rótulo1>.....RETURN (linha2/rótulo2)
Desvia o fluxo do programa para a sub-rotina que começa na linha1 ou rótulo1 especificado. Retorna quando encontrar a instrução RETURN que pode especificar volta à linha2 ou rótulo2 e não à instrução seguinte.

GOTO<linha/rótulo>
Desvia o fluxo do programa para a linha ou rótulo especificado.

IT<condição> THEN <ação1> (ELSE<ação2>)

Se a condição for verdadeira, a ação1 será executada, caso contrário a próxima instrução será executada ou a ação2 será executada caso a cláusula ELSE tenha sido especificada. (Forma obsoleta).

IF<condição> THEN <ações1> ELSEIF THEN <condição2> <ações2> (ELSE<ações3>) END IF

Forma bloqueada da instrução IF, permitindo uma série de instruções em mais de uma linha.

INPUT(;)("mensagem";/,)<variáveis>

Lê os dados do teclado, atribuindo-os às variáveis especificadas.

; Logo depois do INPUT mantém o cursor na linha após resposta do usuário.

; Depois da mensagem gera ponto de interrogação.

, Depois da mensagem suprime o ponto de interrogação.

variáveis: separadas por vírgulas cujo valor será suprido pelo usuário;

string: entre aspas permite registro de vírgulas e CR.

INPUT#n,<variáveis>

Lê dados de um arquivo n atribuindo-os às variáveis especificadas.

IOCTL(#)n,string

Envia caracter ou string de controle ao dispositivo #n se tal for previsto pelo "device driver" do dispositivo.

KEY ON/OFF/LIST

Mostra(ON) ou não(OFF) as funções às teclas F1 a F12.("default" = ON). "List" faz com que as funções completas (até 15 caracteres) sejam listados na tela).

KEY <n>,<"string">

Atribui ao string (até 15 caracteres) às teclas de função (n = 1- 10,30,31).

KEY<n>,CHR$(semáforo)+CHR$(código de varredura0

n: teclas definidas pelo usuário (15-25); semáforo: estado das teclas SHIFT(01), CTRL(4), ALT(8), NUMLOCK(32),CAPSLOCK(64), TECLADO EXTENDIDO(128).

KEY (n) ON/OFF/STOP

Ativa(ON), desativa(OFF) ou adia(STOP) operações com uma tecla de função.

KILL<"arquivo">

Elimina o arquivo do disco, equivale ao DEL ou ERASE do DOS.

(LET)<variável> = <expressão>
Atribui o valor da expressão à variável. Seu uso é optativo

LINE (STEP) (x1,y1) (STEP)(x2,y2) (,(cor)(B(F))(,estilo)
Desenha uma linha ou um quadro no vídeo. STEP: as coordenadas seguintes são relativas ao cursor gráfico; x1,y1: ponto inicial da linha; x2,y2: ponto final da linha; B: desenha um quadro com a diagonal definida; BF: preenche um quadro com a cor especificada; estilo: máscara de 16 bits -bit = 1, pixel ligado.

LINE INPUT (;)("mensagem";)<variável$>
Lê uma linha inteira (até 255 caracteres) do teclado, ignorando delimitadores. O ";" mantém cursor na mesma linha depois da digitação.

LINE INPUT#n,<variável$>
Lê uma linha inteira (até 255 caracteres), ignorando delimitadores internos de um arquivo seqüencial atribuido-a à variável$.

LOCATE(linha)(,(cursor)(,(início)(,(fim)))
Posiciona o cursor na linha e coluna especificadas e opcionalmente define sua visibilidade e tamanho. Default de linha e coluna é a atual. cursor: visível (0); início,fim: definem a linha inicial e final da varredura do cursor.

LOCK (#)n,(registro/(inicio) TO fim)
Controla o acesso a uma parte ou a todo um arquivo em ambiente de rede. registro: número do registro ou byte cujo acesso deve ser bloqueado; início, fim: intervalo de registro ou bytes cujo acesso deve ser bloqueado.

LPRINT (USING "formato";) <expressões>(;/,)
Imprime dados na impressora (LPT1:) e opcionalmente define o formato de impressão.

LSET<variável$> =<expressão$>
Alinha a esquerda a expressão na variável$, definida em buffer de arquivo aleatório. Pode ser usado para alinhar strings normais.

MID$ (x$,começo(,comprimento))=<y$>
Repõe uma parte do string x$ com comprimento caracteres do string y$, a partir da posição começo em x$.

MKDIR <"path">
Cria um novo diretório definido em path.

NAME <"velho"> AS <"novo">
Altera o nome do arquivo em disco permitindo sua realocação em diretório.

NEXT <variáveis>
Indica o fim de um laço FOR-NEXT.

ON COM (n) GOSUB <linha/rótulo>

Desvia para a sub-rotina especificada quando um caractere é detectado na porta serial n. Se linha=0, a detecção é desativada.

ON ERROR GOTO <linha/rótulo>

No caso de erro, salta para a rotina de tratamento de erro especificada em linha ou rótulo. Se linha=0 detecção de erros é desativada.

ON <expressão> GOTO <linha/rótulo>

Salta para linha ou rótulo dependendo do valor de expressão.

ON <expressão> GOSUB <linha/rótilos>

Executa a sub-rotina especificada, de acordo com o valor de expresão.

ON KEY(n) GOSUB <linha/rótulo>

Executa a sub-rotina especificada, quando pressionada a tecla KEY para explicaçṏes detalhadas. Linha = 0 desativa dispositivo.

ON PEN GOSUB <linha/rótulo>

Executa a sub-rotina especificada, quando a caneta ótica é ativada. Se linha = 0, o desvio é desativado.

ON PLAY(n) GOSUB <linha/rótulo>

Desvia para a sub-rotina indicada por linha ou rótulo quando se chega na nota número n na fila de notas. Linha = 0 desativa detecção.

ON STRIG(n) GOSUB <linha/rótulo>

Executa a sub-rotina especificada, quando um dos disparadores de um Joystick é pressionada. Se linha = .0, a detecção é desativada. n = 0: disparador de baixo do Joystick A; 2: disparador de baixo do Joystick B; 4: disparador de cima do Joystick A; 6: disparador de cima do Joystick B.

ON TIMER(n) GOSUB< linha/rótulo>

Desvia para a sub-rotina indicada depois da passagem de n segundos. Linha=0 desativa o dispositivo

OPEN <"arquivo"> (FOR<modo1>) (ACCESS<acesso>) (bloqueio) AS[#]n (LEN=tamreg)

Abre um arquivo ou dispositivo para operaçṏes de entrada/saida.

modo1 - Modo de operação do arquivo (default = RANDOM). OUTPUT: para saída sequencial; INPUT: para entrada sequencial: APPEND: para saída sequencial no final do arquivo; RANDOM: entrada ou saída aleatória; BINARY: permite acesso a qualquer byte do arquivo.

acesso - Define o tipo de acesso permitido. READ: somente para leitura; WRITE: somente para gravação; READ WRITE: tanto leitura quanto gravação.

bloqueio - Define o tipo de bloqueio em ambiente multiusuário (default = acesso somente para este programa). SHARED: acesso livre a qualquer programa; LOCK READ: somente este programa pode ler arquivo; LOCK WRITE: somente este programa pode gravar arquivo; LOCK READ WRITE: bloqueia qualquer acesso externo ao arquivo.

tamreg - Bytes por registro (1-32767, default rand = 128; seq = 512).

OPEN <modo2>, [#]n,<"arquivo"> (tamreg)

Abre um arquivo ou dispositivo para operaçÕes de entrada/saida.

modo2 - Modo de operação do arquivo. A: adiciona entradas no final do arquivo; B: binário; O: sequencial de saída; I: sequencial de entrada; R: aleatório.

tamreg - Bytes por registro (1-32767, default rand = 128;seq = 512).

OPEN <"COMn:opçÕes1 opeçÕes2"> (FOR modo) <AS> [#]<numarq> (LEN = tamreg)

Habilita dispositivos seriais para operaçÕes de entrada/saida. n: número da porta de comunicaçÕes; opçÕes1: (são obrigatórias se itens de opçÕes2 forem utilizados); Veloc: velocidade de transmissão (75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600); parid: S=paridade zero, E=par, O=ímpar, N=não verifica, M=um.; dados: bits de dados (5, 6, 7 ou 8, default=8); parada: bits de parada (1,1,5 ou 2). opçÕes2 :ASC: usa as convençÕes do código ASCII de transmissão de dados; BIN: default, não utiliza as convençÕes ASCII; CD[n]: espera n milessegundos pelo CD antes de gerar timeout; CS[n]: espera n milessegundos pelo CS antes de gerar um timeout; DS[n]: espera n milessegundos pelo CS antes de gerar um timeout; OP[n]: tempo de espera (em milessegundos)para comecar comunicação; RB[n]: tamanho do buffer de transmissão (default = 512; max = 31767); RS: suprime o sinal RTS (Request to Send); TB[n]: tamanho do buffer de transmissão (default = 512; max = 31767).

OPTION BASE <0/1>

Declara o valor mínimo para índices de matrizes.

OUT <porta> , <byte>

Coloca o byte (0-255) na porta (0-65535) especificada.

PAINT (STEP) (x,y) (,(padra-interno)(,(cor-borda)(,padra-externo)))

Preenche uma área do vídeo com o padrão interno e/ou padrão externo especificados a partir das coordenadas x,y até encontrar a cor especificada em cor borda.

STEP: torna as coordenadas x,y relativas ao ponto atual; x,y: coordenadas onde a cor inicia; PAINT:(30,35),6,2.

PALETTE(atributo, cor)
Associa uma cor a um atributo que será utilizado nos comandos gráficos subsequentes que utilizarem o atributo especificado (só com vídeo EGA, VGA ou MCGA).

PALETTE USING A%(5)
Associa uma cor a um atributo especificado na matriz-inteira a partir do subscrito-matriz (só com vídeo EGA, VGA ou MCGA).

Pcopy<página-fonte, página-destino)
Copia a página-fonte do vídeo para a página-destino do vídeo.

PEN ON /OFF/STOP
Ativa (ON), desativa (OFF) ou adia (STOP) operaçÖes com a caneta ótica.

PLAY "subcomandos"
Emite sons musicais .

PLAY ON/OFF/STOP
Ativa (ON), desativa (OFF) ou inibe (STOP) operaçÖes com o comando PLAY.

POKE <deslocamento>, <byte>
Coloca o byte (0-255) no deslocamento (em relação a DS) especificado.

PRESET (STEP) (x,y) (,cor)
Desenha um ponto na coordenada e cor especificados. Se a cor for omitida, assume a cor de fundo, apagando o ponto da tela. STEP: especifica as coordenadas x,y como relativas ao ponto atual; PRESET: (30,50),3.

PRINT(USING "formato";) <lista de expressÖes> (,/;)
Imprime dados no vídeo e opcionalmente define o formato de impressão. "," salta para a próxima zona de impressão,";" imprime sem espaço.

PSET (STEP) (x,y) (,cor)
Desenha um ponto nas coordenadas e na cor especificadas. STEP: especifica as coordenadas x,y como relativas ao ponto atual.

PUT (#) <n> (,nreg) (,var)
Grava o buffer ou var no registro nreg do arquivo aleatório n.

PUT (STEP) (x,y), matriz (índices) (,ação)
Recupera uma imagem do vídeo armazenada em uma matriz. (x,y): coordenadas do campo superior esquerdo do retângulo; matriz: matriz numérica que contÉm as informaçÖes a serem mostradas no vídeo; índices: a partir de que ponto na matriz a imagem deve ser localizada; ação: - PSET: transfere imagem literal da matriz para o vídeo; PRESET: transfere imagem

negativa da matriz para o vídeo; XOR: inverte ponto no vídeo se o ponto existe na matriz; AND: sobrepõe a imagem na matriz na imagem do vídeo; OR: transfere a imagem na matriz somente se a imagem já existe.

RANDOMIZE (n)
Reinicializa o gerador de números aleatórios com a semente n.

READ <variáveis>
Lê sequencialmente os valores contidos nas declarações DATA, atribuindo-os às variáveis especificadas.

REDIM (SHARED) matriz (subscritos..) (AS tipo) (,.....
Redimensiona matriz(es) dinâmica(s). SHARED: permite que procedimentos tenham acesso a(s) matriz(es); AS tipo: define o tipo de variável contida na matriz.

REM (comentários)
Identifica comentários dentro do programa. O mesmo que '.

RESET
Fecha todos os arquivos em aberto.

RESTORE (linha/rótulo)
Permite que os dados das declarações DATA sejam relidos a partir da linha ou rótulo especificado (default=primeira declaração DATA do programa).

RESUME (0/NEXT/linha/rótulo)
Continua a execução do programa após a rotina de tratamento de erros ter sido executada. Sem nenhum parâmentro retorna para a linha onde ocorreu o erro. 0: retorna para a linha onde ocorreu o erro; NEXT: retorna para a linha seguinte a do erro; linha/rótulo: retorna para a linha especificada.

RETURN (linha/rótulo)
encerra a execução de uma sub-rotina, retornando para a instrução seguinte ao último GOSUB ou para a linha ou rótulo especificado.

RMDIR <"path">
Elimina um diretório especificado por path.

RSET <variável$>=<expressão$>

Copia expressão$ na variável$ que pode fazer parte de um buffer de arquivo aleatório ou ser um string normal (alinhamento à direita).

RUN (linha/arquivo)
Executa programa atual na linha especificada ou executa arquivo.

SCREEN (modo),(,opção) (,páginaA) (,páginaV)))
Seleciona atributos do vídeo.

Modo : Modo do vídeo. 0: texto (25 linhas x 40/80 colunas); 1: média resolução (320 x 200), texto 40 colunas x 25 linhas; 2: alta resolução (640 x 200), texto 80 colunas x 25 linhas; 3-13: exige EGA, VGA, Hércules ou MCGA.

Opção : habilita/desabilita cores.
PáginaA: página do vídeo (0-7 para largura=40,0-3 para largura=80).
PáginaV: página visual (página mostrada no vídeo).

SEEK (#)n,posição

Seta a posição no arquivo n para a próxima escrita ou leitura. Em arquivo aleatório, posição é número do registro, nos outros modos posição é o número do byte a contar do início do arquivo.

SELECT CASE<var>:CASE<lista1>(comandos1):(CASE lista2 (comandos2)):.....:(CASE ELSE (comandosm)):END SELECT

var: variável ou expressão a ser testada; lista: lista de valores que acionam os comandos associados, expressão, expressão,...; expressão TO expressão: IS relação; comandos: comando(s) acionados se o CASE for verdadeiro.

```
SELECT CASE I%
CASE 1,2: PRINT " O VALOR E UM OU DOIS".
CASE 3 TO 10: PRINT " O VALOR ESTA ENTRE TRES E DEZ".
CASE IS > 10: PRINT " O VALOR E MAIOR QUE DEZ".
CASE ELSE: "PRINT " O VALOR E MENOR OU IGUAL A ZERO".
```

SHARED<variável> (AS tipo)(,variável(AS tipo))...

Permite que subprogramas compartilhem variáveis do programa sem que estas sejam passadas como parâmetros.

SHELL("comandos")

Permite a execução de programa (extensão .COM, .EXE, .BAT) ou comando interno do DOS, retornando para a instrução seguinte no QuickBasic. Sem argumentos, é necessário digitar EXIT para realizar este retorno.

SOUND<frequência>,<duração>

Emite som com a frequência e duração especificadas.

STATIC<variavéis>

Declara variavéis com locais a um subprograma ou função e mantém seus valores inalterados entre as chamadas..

STOP

Interrompe programa, fecha todos os arquivos abertos e retorna ao DOS. Dentro do QuickBasic, retorna ao ambiente integrado, sem fechar arquivos.

STRIG ON/OFF

Não tem efeito no QuickBasic: mantém compatibilidade com outras versões no BASIC.

STRIG(n) ON/OFF/STOP

Verifica (ON), não verifica(OFF) ou adia a verificação (STOP) das atividades do joystick.

SUB<nome> ((parâmetro)) (STATIC):...:(EXIT SUB)...END SUB
Define um subprograma. nome: uma variável global vale para todo o programa; parâmetros: var[()] (AS tipo),...: dispensa-se dimensões:STATIC: mantém o valor das variavéis entre chamadas, e mais rápido: EXIT SUB: permite abandonar subprograma antes de seu término.

SWAP<variável1>,<variável2>
Intercambia os valores das duas variavéis (devem ser do mesmo tipo).

SYSTEM
Fecha todos os arquivos e retorna ao DOS.

TIME$=<string>
Atribui um horário ao DOS usando notação de 24 horas.

TIMER ON/OFF/STOP
Ativa (ON), desativa (OFF) ou adia (STOP) a execução do comando ON TIMER.

TRON/TROFF
Mostra a linha do programa que está sendo executada (TRON) ou desativa este dispositivo (TROFF).

TYPE<tipo> nome AS tipo: AS tipo:.....END TYPE
Define uma variável estrutural contendo um ou mais elementos.

UNLOCK#n (,registro/(inicio) TO fim)
Libera o acesso a uma parte ou a todo o arquivo. registro: número do registro cujo acesso deve ser liberado; início,fim: intervalo de registro cujo acesso deve ser liberado.

VIEW(SCREEN)((x1,y1)-(x2,y2) (,(cor) (,borda)))
Define janela para saída gráfica (default=tela inteira). SCREEN: define as coordenadas como absolutas em relação a tela; x1,y1,x2,y2: definem as coordenadas esquerda superior e direita inferior da janela gráfica; cor: cor da janela, se omitido janela não e pintada; borda: qualquer linha desenha linha em torno da janela.

VIEW PRINT (linhasup TO linhainf)
Define janela de texto (default=tela inteira). linhasup: linha superior da janela (1-25); linhainf: linha inferior da janela (1-25).

WAIT<porta,expressão-ans> (,expressão-xor)
Suspende a execução do programa até que o conteúdo da porta depois de um AND lógico com expressão-and e um XOR lógico com expressão-xor (cujo default e o 0) resulte em valor diferente de zero.

WHILE<condição>:bloco-de-instruções.....WEND
Executa o bloco-de-instruções seguintes até WEND, enquanto a condição for verdadeira.

WIDTH(colunas) (,linhas)

Define colunas (40 ou 80) e linhas (25, 30, 43, 50 ou 60) do vídeo. Opç$es disponiveis dependem do adaptador (CGA,EGA etc). Default e modo atual.

WIDTH<#n/dispositivo>,<largura>

Define a largura de linha no arquivo #n ou dispositivo aberto como arquivo: a instrução toma efeito imediatamente.

WIDTH LPRINT<largura>

Define a largura de linha da impressora.

WINDOW ((SCREEN) (x1,y1) - (x2,y2))

Define uma area gráfica, que independe do tamanho e tipo de vídeo. SCREEN: inverte as coordenadas cartesianas, y cresce para baixo; x1,y1,x2,y2: números reais definindo os cantos da janela.

WRITE (express$es)

Mostra no vídeo o valor das express$es separadas por virgulas, os strings entre aspas e nenhum espaco entre as express$es.

WRITE #n,<express$es>

Grava o valor das express$es em um arquivo sequencial, inserindo virgulas entre os itens e colacando os strings entre aspas.

A N E X O 06

INSTRUMENTAÇÃO COM INTERFACE GPIB (IEEE-488) (20)

1 - ARQUIVOS DE DECLARAÇÃO PARA AS LINGUAGENS

Apresenta-se a seguir os arquivos de declaração e tipo objeto fornecidos pelo fabricante para as linguagens C, PASCAL, FORTRAN, BASIC, QUICKBASIC e ASSEMBLER para a geração de novos programas e aplicativos.

1.1 - LINGUAGEM BASIC

O arquivo BB.M juntamente com o "Handler" GPIB - BASIC, contêm a interface entre a linguagem BASIC e o cartão STD-8410. O termo BASIC, usado neste trabalho, refere-se ao "IBM INTERPRETATIVE Basic".

PREPARAÇÕES INICIAIS: O arquivo DECL.BAS contém o bloco de códigos, em linguagem Basic, que deverá ser anexado ao início do programa de aplicação do usuário, com os respectivos ajustes dos números das linhas.

ARQUIVO DECL.BAS

```
1 CLEAR ,XXXXX! : IBINIT1=XXXXX! : IBINIT2=IBINIT1+3
: BLOAD "bib.m",IBINIT1

2 CALL IBINIT1  (IBFIND, IBTRG, IBCLR, IBPCT, IBSIC,
IBLOC, IBPPC,  IBBNA, IBONL,  IBRSC,  IBSRE,  IBRSV,
IBPAD, IBSAD,  IBIST, IBDMA,  IBEOS,  IBTMO,  IBEOT,
IBRDF, IBWRTF, IBTRAP)

3 CALL IBINIT2 (IBGTS, IBCAC, IBWAIT, IBPOKE, IBWRT,
IBWRTA, IBCMD,  IBCMDA, IBRD,  IBRDA, IBSTOP, IBRPP,
IBRSP,  IBDIAG,   IBXTRC,  IBRDI,   IBWRTI,  IBRDIA,
IBWRTIA, IBSTA%, IBERR%, IBCNT%)
```

As linhas 1 e 2 do bloco de código deverão conter um valor no lugar de XXXXX. Este valor tem que ser calculado para o sistema no qual o software do STD-8410 esteja sendo usado. O valor encontrado determina a área máxima de trabalho para o Basic e deve ser calculado determinando-se o comprimento em bytes do BB.M e outras rotinas em linguagem de máquina utilizadas no programa do usuário e subtraindo-se este valor do número de bytes ditos livres quando o interpretador Basic foi inicializado.

A listagem do diretório dá o tamanho do arquivo em bytes e pode ser usada para determinar o tamanho de BB.M. Note que não é importante que o BB.M seja carregado no início da área reservada; é essencial, porém, que o endereço de carga seja um múltiplo de 16. Os números da linha 2, caso o início da área reservada e o endereço de carga sejam diferentes, correspondem aos endereços de carga.

A função e os nomes de variáveis definidos nas linhas 2 e 3 do bloco de códigos podem ser mudados se eles conflitarem com os nomes usados no programa de aplicação do usuário.

1.2 - LINGUAGEM C

O arquivo CB.OBJ contém a interface entre a linguagem C e o STD-8410. Programas aplicativos GPIB escritos em linguagem C devem ser ligados com os arquivos CB.OBJ e PCGPIB.OBJ para produzir um arquivo executável.

VARIÁVEIS DE STATUS GLOBAIS: três variáveis declaradas como públicas dentro do CB.OBJ contém as informações de status, erro e contagem de bytes. A declaração destas variáveis como externas no programa de aplicação permite que o usuário se referencie à elas para recuperação de erros e rotinas de diagnósticos.

ARQUIVO DECL.C

```
/* variáveis declaradas publicas pelo cb.obj */
extern  int far ibsta;  /* palavra de status     */
extern  int far iberr;  /* codigo de erro GPIB   */
extern  int far ibcnt; /*numero de bytes enviados */

*/ constantes usadas em programas de exemplos */
#define UNL   0x3f   /* GPIB comando unlisten */
#define UNT   0x5f   /* GPIB comando untalk */
#define GTL   0x01   /* GPIB comando go to local */
#define SDC   0x04   /* GPIB selected device clear */
#define PPC   0x05   /* GPIB parallel poll conf. */
#define GET   0x08   /* GPIB group execute trigger */
#define TCT   0x09   /* GPIB tome o controle */
#define LLO   0x11   /* GPIB local lock out */
#define DCL   0x14   /* GPIB device clear */
#define PPU   0x15   /* GPIB ppoll unconfigure */
#define SPE   0x18   /* GPIB habilitar poll serial */
#define SPD    0x19    /* GPIB desabilitar poll serial
*/
#define PPE    0x60    /* GPIB habilitar poll paralelo
*/
#define  PPD      0x70     /*  GPIB  desabilitar  poll
paralelo*/
#define S       0x08    /* especifica o sentido do PPR
*/
#define REOS 0x400
#define XEOS 0x800
#define BIN  0x1000
#define LF   0x0a
#define TIMO 0x10000
#define SRQI 0x1000
#define CIC  0x20
#define TACS 0x08
#define LACS 0x04
/* variáveis do programa de aplicação passadas  */
/* para as funcoes STD-8410 .                   */
int     bd    = 0;        /* numero do cartao     */
int     cnt;              /* variável cont bytes  */
```

```
int     v;              /* uso geral              */
int     conf;           /* config. de enderecos */
int     disp;           /* dispositivo GPIB       */
char    gpib_buf[512];  /* buffer p/ comunic.     */
char    ppr;            /* byte de poll paralelo*/
char    spbyte;         /* byte para resposta de*/
/* poll serial          */
unsigned int    mask;   /* eventos esperados      */
char    far *pt_buf = gpib_buf; /* ponteiro p/    */
                        /* o buffer utilizado     */
                        /* nos comandos IBCMD,    */
                        /* IBESC, IBLER, IBRD e   */
                        /* IBWRT.                 */
```

1.3 - LINGUAGEM FORTRAN

O arquivo FB.OBJ contém a interface entre o "handler" GPIB e a linguagem FORTRAN. O acesso ao STD-8410 é obtido ligando-se o programa aplicativo compilado com o FB.OBJ e com o PCGPIB.OBJ, gerando assim um arquivo executável.

VARIÁVEIS DE STATUS GLOBAIS: três variáveis declaradas como comuns no arquivo FB.OBJ contém as informaçÕes de status, erro e contagem de bytes. Declarando essas três variáveis como comuns ao seguimento "A" dos programas de aplicação, o usuário poderá referenciá-las para recuperação de erros e rotinas de diagnósticos. Para torná-las variáveis inteiras de 2 butes, o usuário deve incluir a declaração $storage:2 no começo do programa de aplicação.

ARQUIVO DECL.FOR

```
$storage:2
common /ibglob/ ibsta, iberr, ibcnt

    integer*2 bd, v
    integer*2 cmd(512)
    integer*2 rd(512)
    integer*2 wrt(512)
    integer*2 UNL, UNT, GTL, SDC, PPC, GET, TCT, LLO
    integer*2 DCL, PPU, SPE, SPD, PPE, PPD

    character  spbyte
    integer*2  S
    integer*2  REOS, XEOS, BIN, LF
    integer*2  TIMO, SRQI, CIC, TACS, LACS
    integer*2  mask
    character  ppr
    integer*2  cnt

    data UNL/63/,UNT/95/,GTL/01/,SDC/04/,PPC/05/
    data GET/08/,TCT/09/, LLO/17/,DCL/20/,PPU/21/
    data SPE/24/,SPD/25/,PPE/96/,PPD/112/

    data S/08/,REOS/1024/,XEOS/2048/
    data BIN/4096/,LF/10/, TIMO/16384/,SRQI/4096/
    data CIC/32/,TACS/8/,LACS/4/
```

1.4 - LINGUAGEM PASCAL

Idêntico as linguagens anteriores, os programas de aplicação escritos em PASCAL devem ser ligados ao arquivo PB.OBJ e ao PCGPIB.OBJ para a produção do arquivo executável.

VARÁVEIS DE STATUS GLOBAIS: similar as linguagens anteriores, através da declaração de três variáveis públicas o usuário poderá referenciá-las para recuperação de erros e rotinas de diagnósticos. As funções do STD-8410 chamadas pelo programa de aplicação deverão ser declaradas como externas.

ARQUIVO DECL.PAS

```
(* variáveis de status declaradas publicas pelo PIB.OBJ*)
var [extern] ibsta : integer;  (* palavra de status *)
var [extern] iberr : integer;  (* GPIB codigo de erro  *)
var [extern] ibcnt : integer;  (* numero de bytes envi-*)
                               (* ados ou, em casos    *)
                               (* de erro DOS, o codigo*)
                               (* do erro DOS          *)

Const
UNL   =  16#3f;   (* GPIB comando unlisten          *)
UNT   =  16#5f;   (* GPIB comando untalk            *)
GTL   =  16#01;   (* GPIB go to local               *)
SDC   =  16#04;   (* GPIB selected device clear     *)
PPC   =  16#05;   (* GPIB ppoll     configure       *)
GET   =  16#08;   (* GPIB group execute trigger     *)
TCT   =  16#09;   (* GPIB tome o controle           *)
LLO   =  16#11;   (* GPIB local lock out            *)
DCL   =  16#14;   (* GPIB device clear              *)
PPU   =  16#15;   (* GPIB ppoll unconfigure         *)
SPE   =  16#18;   (* GPIB habilitar poll serial     *)
SPD   =  16#19;   (* GPIB desabilitar poll serial   *)
PPE   =  16#60;   (* GPIB habilitar poll paralelo   *)
PPD   =  16#70;   (* GPIB desabilitar poll paralelo*)
S     =  16#08;   (* especifica sentido de PPR      *)

REOS  =  16#400;
XEOS  =  16#800;
BIN   =  16#1000;
LF    =  16#0A;
TIMO  =  16#4000;
SRQI  =  16#1000;
CIC   =  16#20;
TACS  =  16#08;
LACS  =  16#04;


Type     cbuf =  array[1..512] of char;


Var
cmd : cbuf;
rd  : cbuf;
wrt : cbuf;
v   : integer;     (* parametro "valor"              *)
bd  : integer;     (* numero do cartao               *)
```

```
cnt : integer;      (* cont bytes para transferencias  *)
spbyte : char;      (* byte de resposta do poll serial *)
mask : integer;     (* eventos a serem esperados       *)
ppr : char;         (* byte resposta do poll paralelo  *)

(* funcoes STD-8410 declaradas publicas pelo PIB.OBJ *)
procedure     ibcac  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibclr  (bd:integer;
                      disp:integer);
                                    extern;
procedure     ibcmd  (bd:integer;
                      var cmd:cbuf;
                      cnt:integer);
                                    extern;
procedure     ibconf (bd:integer;
                      conf:integer;
                      v:integer);
                                    extern;
procedure     ibdma  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibeos  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibeot  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibesc  (bd:integer;
                      disp:integer;
                      var esc:cbuf;
                      cnt:integer);
                                    extern;
procedure     ibgts  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibist  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibler  (bd:integer;
                      disp:integer;
                      var ler:cbuf;
                      cnt:integer);
                                    extern;
procedure     iblpe  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibonl  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibpad  (bd:integer;
                      v:integer);
                                    extern;
procedure     ibrd   (bd:integer;
                      var rd:cbuf;
                      cnt:integer);
                                    extern;
procedure     ibrpp  (bd:integer;
                      var ppr:char);
                                    extern;
procedure     ibrsc  (bd:integer;
```

```
                        v:integer);
                                              extern;
procedure     ibrsv   (bd:integer;
                       v:integer);
                                              extern;
procedure     ibrtl   (bd:integer);
                                              extern;
procedure     ibsad   (bd:integer;
                       v:integer);
                                              extern;
procedure     ibsic   (bd:integer);
                                              extern;
procedure     ibsre   (bd:integer;
                       v:integer);
                                              extern;
procedure     ibtmo   (bd:integer;
                       v:integer);
                                              extern;
procedure     ibtrg   (bd:integer;
                       disp:integer);
                                              extern;
procedure     ibwait  (bd:integer;
                       mask:integer);
                                              extern;
procedure     ibwrt   (bd:integer;
                       var wrt:cbuf;
                       cnt:integer);
                                              extern;
```

1.5 - LINGUAGEM ASSEMBLER

Programas de aplicação GPIB escritos em linguagem ASSEMBLER devem ser ligados diretamente ao PCGPIB.OBJ para produzir um arquivo executável que tenha acesso ao STD-8410. As chamadas em linguagem ASSEMBLER requerem que os parâmetros das funções sejam colocados em registradores específicos da seguinte forma:

```
Reg.AL       - número do cartão
Reg.BX       - parâmetro valor
Reg.CX       - contador de bytes ou parâmetro
Reg.DI       - endereço do dispositivo
Reg.(ES:BX)  - ponteiro da área de memória
```

Todos ps registradores retornaram intactos, com excessão do Reg.AX, que retornará com uma cópia do status do STD-8410 e do Reg.CX que será alterado no caso de comandos de transferência de dados, e voltará com um número de bytes transmitidos / recebidos.

VARIÁVEIS DE STATUS GLOBAIS: idêntica as linguagens anteriores a linguagem ASSEMBLER utiliza de três variáveis públicas declaradas no arquivo PCGPIB.OBJ para a manipulação de status, erro e contagem de bytes. As funções chamadas pelo programa do usuário deverão ser declaradas como externas e o arquivo de aplicação ligado ao arquivo PCGPIB.OBJ.

ARQUIVO DECL.ASM

```
;
; arquivo: DECL.ASM
;
; Variáveis e funcoes declaradas publicas em
PCGPIB.OBJ
;
extrn   i_sta:word      ; palavra de status
extrn   i_err:word      ; codigo de erro GPIB
extrn   i_cnt:word      ; numero de bytes enviados
extrn   cac:far
extrn   clr:far
extrn   cmd:far
extrn   conf:far
extrn   dma:far
extrn   eos:far
extrn   eot:far
extrn   esc:far
extrn   gts:far
extrn   ist:far
extrn   ler:far
extrn   lpe:far
extrn   lsp:far
extrn   onl:far
extrn   pad:far
extrn   rd:far
extrn   rpp:far
extrn   rsc:far
extrn   rsv:far
extrn   rtl:far
extrn   sad:far
extrn   sic:far
extrn   sre:far
extrn   tmo:far
extrn   trg:far
extrn   wait:far
extrn   wrt:far
;
UNL     equ  03FH ; GPIB comando de unlisten
UNT     equ  05FH ; GPIB comando de untalk
GTL     equ  001H ; GPIB comando de go to local
SDC     equ  004H ; GPIB selected device clear
PPC     equ  005H ; GPIB parallel poll configure
GET     equ  008H ; GPIB group execute trigger
TCT     equ  009H ; GPIB tome o controle
LLO     equ  011H ; GPIB local lock out
DCL     equ  014H ; GPIB device clear
PPU     equ  015H ; GPIB poll unconfigure
SPE     equ  018H ; GPIB habilitar poll serial
SPD     equ  019H ; GPIB desabilitar poll serial
PPE     equ  060H ; GPIB habilitar poll paralelo
PPD     equ  070H ; GPIB desabilitar poll paralelo
S       equ  008H ; especificar sentido de PPR
REOS    equ  00400H ; terminar leitura em byte eos
XEOS    equ  00800H ; enviar EOI com byte eos
BIN     equ  01000H ; usar 8 bits para comparar eos
LF      equ  00AH ; caracter line feed
TIMO    equ  04000H     ; bit da palavra de status
SRDI    equ  01000H     ; bit da palavra de status
CIC     equ  00020H     ; bit da palavra de status
TACS    equ  00008H     ; bit da palavra de status
```

```
LACS    equ  00004H     ; bit da palavra de status
;
;
DATA    segment byte public 'DATA'

BUFFER  db   512  dup (?)
V       dw   ? ; parametro valor
ENDER   dw   ? ; endereco de configuração do STD8410
BD      db   ?    ; numero do cartao
MASK    dw   ?    ; esperar eventos
SPBYTE  db   ?    ; resposta de poll serial
PPR     db   ?    ; byte de resposta de poll parale.
CNT     dw   ?    ; variável para  contagem de bytes
DISP    dw   ?    ; dispositivo

DATA    ends
;
```

1.6 - LINGUAGEM QUICKBASIC

Para o QUICKBASIC as chamadas de funç%es não serão feitas à endereços contidos em variáveis como no Basic interpretado. Elas serão chamadas diretamente pelo nome e não podendo este ser alterado pelo usuário. Somente o nome das variáveis globais definidas poderá ser alterado.

De acärdo com a linguagem QUICKBASIC todos os argumentos de funç%es são variáveis inteiras ou cadeias de caracteres, e seus valores devem ser associados antes que a chamada da função seja efetuada.

O primeiro argumento de todas as funç%es é a variável inteira BD%, que especifica a interface STD-8410 usada. BD% tem que ser igual a zero quando for usado o "handler" configurado para um único cartão. BD% não tem relação com o endereço primário da interface. Os outros argumento poderão ser variáveis inteiras ou cadeias de caracter dependendo de cada função

ARQUIVO DECLQB4.BAS

```
' Declaração para o QuickBASIC Rev. C.5

' NOTA: Este arquivo deverá ser utilizado para
  versão 4.0 ou mais recente.

' Variáveis comuns de estatus do GPIB

COMMON SHARED /NISTATBLK/ IBSTA%, IBERR%, IBCNT%

' Declaração das Sub-rotinas do GPIB

 DECLARE SUB IBBNA    (BD%, BDNAME$)
 DECLARE SUB IBCAC    (BD%, V%)
 DECLARE SUB IBCLR    (BD%)
 DECLARE SUB IBCMD    (BD%, CMD$)
 DECLARE SUB IBCMDA   (BD%, CMD$)
 DECLARE SUB IBDMA    (BD%, V%)
 DECLARE SUB IBEOS    (BD%, V%)
 DECLARE SUB IBEOT    (BD%, V%)
```

```
DECLARE SUB IBFIND   (BDNAME$, BD%)
DECLARE SUB IBGTS    (BD%, V%)
DECLARE SUB IBIST    (BD%, V%)
DECLARE SUB IBLOC    (BD%)
DECLARE SUB IBONL    (BD%, V%)
DECLARE SUB IBPAD    (BD%, V%)
DECLARE SUB IBPCT    (BD%)
DECLARE SUB IBPPC    (BD%, V%)
DECLARE SUB IBRD     (BD%, RD$)
DECLARE SUB IBRDA    (BD%, RD$)
DECLARE SUB IBRDF    (BD%, FLNAME$)
DECLARE SUB IBRDI    (BD%, IARR%(), CNT%)
DECLARE SUB IBRDIA   (BD%, IARR%(), CNT%)
DECLARE SUB IBRPP    (BD%, PPR%)
DECLARE SUB IBRSC    (BD%, V%)
DECLARE SUB IBRSP    (BD%, SPR%)
DECLARE SUB IBRSV    (BD%, V%)
DECLARE SUB IBSAD    (BD%, V%)
DECLARE SUB IBSIC    (BD%)
DECLARE SUB IBSRE    (BD%, V%)
DECLARE SUB IBSTOP   (BD%)
DECLARE SUB IBTMO    (BD%, V%)
DECLARE SUB IBTRAP   (MASK%, MODE%)
DECLARE SUB IBTRG    (BD%)
DECLARE SUB IBWAIT   (BD%, MASK%)
DECLARE SUB IBWRT    (BD%, WRT$)
DECLARE SUB IBWRTA   (BD%, WRT$)
DECLARE SUB IBWRTF   (BD%, FLNAME$)
DECLARE SUB IBWRTI   (BD%, IARR%(), CNT%)
DECLARE SUB IBWRTIA  (BD%, IARR%(), CNT%)

' Declaração das Funções do GPIB

DECLARE FUNCTION ILBNA%    (BD%, BDNAME$)
DECLARE FUNCTION ILCAC%    (BD%, V%)
DECLARE FUNCTION ILCLR%    (BD%)
DECLARE FUNCTION ILCMD%    (BD%, CMD$, CNT%)
DECLARE FUNCTION ILCMDA%   (BD%, CMD$, CNT%)
DECLARE FUNCTION ILDMA%    (BD%, V%)
DECLARE FUNCTION ILEOS%    (BD%, V%)
DECLARE FUNCTION ILEOT%    (BD%, V%)
DECLARE FUNCTION ILFIND%   (BDNAME$)
DECLARE FUNCTION ILGTS%    (BD%, V%)
DECLARE FUNCTION ILIST%    (BD%, V%)
DECLARE FUNCTION ILLOC%    (BD%)
DECLARE FUNCTION ILONL%    (BD%, V%)
DECLARE FUNCTION ILPAD%    (BD%, V%)
DECLARE FUNCTION ILPCT%    (BD%)
DECLARE FUNCTION ILPPC%    (BD%, V%)
DECLARE FUNCTION ILRD%     (BD%, RD$, CNT%)
DECLARE FUNCTION ILRDA%    (BD%, RD$, CNT%)
DECLARE FUNCTION ILRDF%    (BD%, FLNAME$)
DECLARE FUNCTION ILRDI%    (BD%, IARR%(), CNT%)
DECLARE FUNCTION ILRDIA%   (BD%, IARR%(), CNT%)
DECLARE FUNCTION ILRPP%    (BD%, PPR%)
DECLARE FUNCTION ILRSC%    (BD%, V%)
DECLARE FUNCTION ILRSP%    (BD%, SPR%)
DECLARE FUNCTION ILRSV%    (BD%, V%)
DECLARE FUNCTION ILSAD%    (BD%, V%)
DECLARE FUNCTION ILSIC%    (BD%)
DECLARE FUNCTION ILSRE%    (BD%, V%)
DECLARE FUNCTION ILSTOP%   (BD%)
```

```
DECLARE FUNCTION ILTMO%    (BD%, V%)
DECLARE FUNCTION ILTRAP%   (MASK%, MODE%)
DECLARE FUNCTION ILTRG%    (BD%)
DECLARE FUNCTION ILWAIT%   (BD%, MASK%)
DECLARE FUNCTION ILWRT%    (BD%, WRT$, CNT%)
DECLARE FUNCTION ILWRTA%   (BD%, WRT$, CNT%)
DECLARE FUNCTION ILWRTF%   (BD%, FLNAME$)
DECLARE FUNCTION ILWRTI%   (BD%, IARR%(), CNT%)
DECLARE FUNCTION ILWRTIA%  (BD%, IARR%(), CNT%)
```

2 - CHAMADAS DAS FUNÇ®ES EM QUICKBASIC

Nesse trabalho foi utilizado o "Driver" para linguagem QuickBASIC e as funç®es implementadas no AIDS_TME são aquelas definidas no módulo de inicialização fornecido pelo fabricante.

Torna-se necessário a descrição detalhada de cada função do STD-8410 bem como de seus parâmetros para um entendimento da montagem do comando a ser enviado ao instrumento e a sua sintaxe correta. A seguir descreve-se essas funç®es e seus parâmetros. Os termos: BD e V significam cartão de interface ("Board") e/ou dispositivo ("Device"), e variável respectivamente.

2.1 - IBCAC - Tornar controlador ativo

Parâmetros: BD,V

BD especifica o número do cartão interface. Se V for diferente de zero, o STD-8410 toma o controle sincronamente com uma operação de transferência de dados; caso contrário, o STD-8410 toma o controle imediatamente (e possivelmente assincronamente).

Para tomar o controle sincronamente, o STD-8410 ativa o sinal ATN para assegurar que o dado que estiver sendo transferido pelo GPIB seja interrompido. Se um "handshake" de dados estiver em progresso, a ação de tomar o controle será suspensa até que o "handshake" seja completado. Se não existir "handshake" em progresso, a ação de tomar o controle é executada imediatamente.

A tomada de controle sincronizada não é garantida se uma operação de IBRD ou IBWRT tiver terminado com "time-out" ou erro.

A tomada de controle assíncrona deve ser usada em situaç®es em que parece impossível ganhar o controle sincronamente (isto é, após um erro de "time-out").

Normalmente não é necessário usar a função IBCAC na maioria das aplicaç®es. Funç®es como IBCMD e IBRPP, que requerem que o STD-8410 tome o controle, o fazem automaticamente. Ocorrerá o erro se o STD-8410 não for o "Controller-In-Charge".

2.2 - IBCLR - "Apagar" dispositivo

Parâmetros: BD e DISP

BD indica o número do cartão de interface. DISP indica qual o número do dispositivo GPIB que enviará a mensagem. É um valor de dois bytes que contém, no byte inferior, o endereço primário do dispositivo (0 a 1EH), se houver. Caso contrário, deverá ser colocado 0.

A função IBCLR apaga as funções internas de um determinado dispositivo. IBCLR chama a função IBCMD para enviar basicamente os seguintes comandos:

- "Unlisten" e "Untalk";
- endereço "listen" do dispositivo;
- endereço secundário do dispositivo, se houver;
- SDC ("Selected Device Clear").

Consulte a função IBCMD para informações adicionais.

2.3 - IBCMD - Enviar Comandos

Parâmetros: BD, CMD e CNT

BD especifica o número do cartão interface. CMD contém os comandos a serem enviados através do GPIB. CNT especifica o número de bytes a serem enviados.

A função IBCMD é utilizada para transferir comandos via GPIB. Estes comandos incluem os endereços de "listen" e "talk" dos dispositivos, endereços secundários de "poll" paralelo e serial, e instruções de "clear" e "trigger" para os dispositivos. A função IBCMD é usada também para passar o controle do GPIB para outro dispositivo. Esta função não é usada para transmitir instruções para os dispositivos; instruções de programação e outras informações dependentes do dispositivo são transmitidas pelas funções IBRD e IBWRT.

A operação IBCMD termina com a ocorrência de um dos seguintes eventos:

- todos os comandos foram transferidos com sucesso;
- foi detectado erro;
- o limite de "time-out" foi excedido;
- foi enviado o comando "take control" (TCT);
- foi recebida a mensagem "interface clear" (IFC) proveniente do Controlador de Sistema (não o STD-8410).Após o término, o número de comandos enviados é determinado pela variável IBCNT.

Ocorrerá erro se o STD-8410 não for o "Controller-In-Charge". Se o STD-8410 estiver em "Standby" ele toma o controle e ativa ATN (torna-se ativo) antes de enviar os bytes de comando, permanecendo então como o controlador ativo. Mesmo não haja "Listeners", a função IBCMD não causará erros.

2.4 - IBCONF - Reconfigurar por software o endereço e o "settling time" de uma placa STD-8410.

Parâmetros: BS, ENDER,V

BD especifica o número do cartão de interface. ENDER indica qual será o novo endereço da placa. V especifica qual será o "settling time" da placa.

O endereço "default" de configuração dos cartões é 2B8H, para todos os cartões STD-8410. Se o usuário tiver mais de um cartão instalado no seu PC, será necessário especificar endereços diferentes para cada cartão através das "dipswitches" dos cartões e enviar comandos IBCONF para reconfigurar o software de cada cartão.

O "settling time" pode ser alterado para longo ou curto, se V for zero ou diferente de zero, respectivamente.

Após o uso da função IBCONF, o cartão estará "ON LINE" como se a função IBONL tivesse sido usada com parâmetro igual a 1.

2.5 - IBDMA - Habilitar ou desabilitar o DMA

Parâmetro BD,V

BD especifica o número do cartão de interface. Se V for diferente de zero, as transferências via DMA entre o STD-8410 e a memória serão usadas pelas operações IBRD e IBWRT. Se V for igual a zero, a E/S programada será usada no lugar de E/S por DMA.

Quando o DMA é habilitado, é usado o canal selecionado pela configuração de "jumpers" e pelo valor de V, isto é, o canal escolhido pelo usuário (1, 2, 3) deve ser transmitido através de V e as transferências via DMA no canal devem ser sido habilitado na STD-8410. Não é possível mudar o canal em tempo de execução.

2.6 - IBEOS - Mudar ou Desabilitar terminação com byte eos.

Parâmetros BD,V

BD especifica o número do cartão interface . V seleciona o caracter eos e o método de terminação de transferência de dados.

MÉTODO	VALOR DE V	
	Byte + sig.	Byte - sig.
A - Termina a leitura quando for detectado um EOS (IBRD)	XXXO 0100	EOS
B - Compara todos os 8 bits do byte eos em vez dos 7 menos significativos (IBRD, IBWRT)	XXX1 0000	EOS
C - Envia mensagem END quando eos for escrito (IBWRT)	XXXO 1000	EOS

FIGURA 6.1 - MÉTODO DE EOS

Os métodos A e B determinam como as operaçṏes IBRD terminam. Se apenas o método A for escolhido, a operação termina quando os 7 bytes menos significativos do byte de dados casarem com os 7 bits menos significativos do caracter eos. Se os métodos A e B forem escolhidos, será usada comparaçṏes de 8 bits completos.

Os métodos B e C juntos determinam quando as operaçṏes IBWRT enviam mensagem END. Se apenas o método C for escolhido, a mensagem END é enviada automaticamente com o byte eos, isto é, quando os 7 bits menos significativos do byte de dados casarem com os 7 bits menos significativos do caracter eos. Se métodos B e C forem escolhidos, será usada uma comparação de 8 bits completos.

2.7 - IBEOT - Habilitar/Desabilitar envio de mensagem de terminação END em operaçṏes de escrita.

Parâmetros: BD,V

BD especifica o número do cartão interface. Se V for diferente de zero, a mensagem END é enviada automaticamente com o último byte de cada operação IBWRT. Se V for zero, END não é enviada.

A mensagem END é enviada quando o sinal EOI do GPIB for ativado durante uma transferência de dados e é usada para identificar o último byte de uma seqüência de dados sem que seja necessário o uso do caracter eos ("end of string"). IBEOT é usado antes de se enviar dados binários de comprimento variável. Veja também IBEOS e a Tabela VI.2.

2.8 - IBESC - Enviar mensagem para um dispositivo

Parâmetros: BD,DISP,ESC,CNT

BD especifica o número do cartão de interface, ESC aponta para a mensagem a ser enviada e CNT indica o seu número de bytes. DISP indica qual o número do dispositivo que receberá a mensagem. É um valor de dois bytes que

contém, no byte inferior, o endereço primário do dispositivo (0 a 1EH). No byte superior deverá ser colocado o endereço secundário (60H a 7EH), se houver. Caso contrário, deverá ser colocado 0.

Ao executar esta função, o computador envia ao barramento GPIB uma seqüência de comandos:

- "UNL" (unlisten), para retirar todos os "listeners" do barramento;

- "UNT" (untalk), para retirar todos os "talkers" do barramento;

- endereça o dispositivo fornecido com "listener";

- endereça o computador como "talker".

Após esta inicialização, o computador envia ao dispositivo a mensagem, da mesma forma que na função IBWRT, para em seguida executar um IBCAC, para que o computador se torne o controlador ativo. Ao final da execução desta função, o sinal de ATN continua ativo.

6.5.9 - IBGTS - Ir de Controlador Ativo para "Standby"

Parâmetros: BD,V

BD especifica o número do cartão interface. Se V for diferente de zero, o STD-8410 simula os "handshakes" da transferência de dados como se fosse em receptor, e quando a mensagem END for detectada, o STD-8410 entra no estado de "holdoff" do sinal NRFD ("Not Ready for Data") do GPIB. Se V for igual a zero, nenhuma simulação de "handshake" ou "holdoff" será executada.

A função IBGTS faz com que o STD-8410 entre no estado de Controlador em "Standby" e desative o sinal ATN, desde que tenha sido inicialmente o Controlador ativo. IBGTS permite que os dispositivos GPIB transfiram sem a participação do STD-8410.

Se a opção de "handshake" simulado for ativada, o STD-8410 participa do "handshake" de dados como um receptor sem que realmente receba os dados. Ele monitora as transferências até a mensagem END, contendo a partir deste momento as demais transferências. Este mecanismo permite que o STD-8410 tome o controle sincronamente após uma operação do IBCMD ou IBRPP.Ocorrerá erro se o STD-8410 não for o "Controller-In-Charge".Veja também a função IBCAC.

2.10 - IBIST - Ativar/Apagar o Bit de Status Individual para os "Polls" Paralelos.

Parâmetros: BD,V

BD especifica o número de cartão interface. Se V for diferente de zero, o bit de status individual é ativado. Se V for igual a zero, o bit é apagado.

A função IBIST é usada quando o STD-8410 participa de um "poll" paralelo conduzido por um outro dispositivo no papel de Controlador Ativo. O Controlador Ativo conduz o "poll" paralelo ativando o sinal EOI para enviar IDY ou a mensagem "Identify". Enquanto esta mensagem estiver ativa, cada dispositivo configurado para participar de "poll" paralelo responde tornando verdadeira ou falsa uma linha de dados predeterminada do GPIB, dependendo do valor do bit ist local. O STD-8410 pode, por exemplo, ser configurado para tornar a linha DIO3 verdadeira se ist=1 e falsa de ist=0; por outro lado, ele pode ser configurado para tornar DIO3 verdadeira, se ist=0 e falsa se ist=1.

A relação entre o valor de ist, a linha a ser ativada, e o modo no qual a linha será ativada, é determinada pelo PPE ou mensagem "Parallel poll Enable" efetiva em cada dispositivo. O STD-8410 é capaz de receber essa mensagem localmente, através da função IBLPE, ou remotamente através de um comando do Controlador Ativo. No último caso, o comando é automaticamente interpretado e executado pelo hardware sem que haja intervenção do software.

Uma vez executada a mensagem PPE, a função IBIST muda o sentido no qual a linha será ativada durante o "poll" paralelo e desta forma o STD-8410 se mostra como um bit para o Controlador.

2.11 - IBLER - Receber mensagem de um dispositivo.

Parâmetros: BD,DISP,LER,CNT

BD especifica o número do cartão de interface. DISP indica qual o número do dispositivo GPIB que enviará a mensagem. É um valor de dois bytes que contém, no byte inferior, o endereço primário do dispositivo (0 a 1EH); no byte superior deverá ser colocado o endereço secundário (60h a 7EH), se houver. Caso contrário, deverá ser colocado 0. LER aponta para o buffer que receberá a mensagem e CNT indica seu número de bytes.

Ao executar esta função, o computador envia ao barramento GPIB uma seqüência de comandos:

- "UNL" (unlisten), para retirar todos os "listeners" do barramento;

- "UNT" (untalk), para retirar todos os "talkers" do barramento;

- endereça o dispositivo fornecido como "talker";

- endereça o computador como "listener".

Após esta inicialização, o computador recebe a mensagem transmitida pelo dispositivo, como na função IBRD. Logo em seguida ele executa um IBCAC, para se tornar o controlador ativo do barramento. Ao final da execução desta função, o sinal de ATN continua ativo.

2.12 - IBLPE - Mudar localmente a configuração de "poll" Paralelo.

Parâmetros: BD,V

BD especifica o número do cartão interface. V deve ser uma mensagem válida de "Parallel Poll Enable" (PPE) ou "Parallel Poll disable" (PPD) definida na especificação do IEEE-488.

A função IBLPE é usada para enviar as mensagens PPE ou PPD ao hardware do STD-8410 quando este participar de "polls" paralelos conduzidos por outro dispositivo como Controlador Ativo e quando for usada uma configuração remota.

Cada uma das 16 mensagens PPE especifica a linha de dados GPIB (DIO1 e DIO8) e o sentido (1 ou 0) que o dispositivo deve usar para responder a um IDY ou a uma mensagem "Identify" durante um "poll" paralelo. A mensagem atribuída é interpretada pelo hardware do STD-8410 juntamente com o valor do bit individual de status (ist) para determinar se a linha selecionada deve ser tornada verdadeira ou falsa. Por exemplo, se PPE=64H, DIO5 é tornado verdadeiro se ist=0 e falso se ist=1. E se PPE=68H, DIO1 é tornado verdadeiro se ist=1 e falso se ist=0. Qualquer mensagem PPD cancela a mensagem PPE em efeito.

O protocolo do sistema determinado pelo usuário deve conter informaçÖes tais como: se a função IBLPE ou a configuração remota é usada; que mensagem PPE ou PPD é enviada e o que possui significado numa resposta de "poll" paralelo.Veja também a função IBIST e a Tabela VI.2.

2.13 - IBLSP - Executar "serial poll"

Parâmetros: BD,DISP,V

BD especifica o número do cartão de interface. DISP indica o número do dispositivo GPIB que receberá o "serial poll". é um valor de dois bytes que contém no byte inferior, o endereço primário do dispositivo(0 a 1EH); no byte superior, dever' ser colocado o endereço secundário (60H a 7EH) se houver. Caso contrário deverá ser colocado 0.V recebe o resultado do "serial poll".

Esta função se inicia através de uma seqüência de comandos emitidos pelo computador:

- "UNL" (Unlisten), para tirar todos os "listeners" do barramento ;

- "UNT" (Untalk), para tirar todos os "talkers" do barramento;

- endereça o computador como "listener";

- endereça o dispositivo fornecido como "talker";

- "SPE" (Serial Poll Enable), para que o dispositivo fornecido envie o resultado de um "serial poll".

O computador lê então, da mesma forma que no IBRD, um byte enviado pelo dispositivo e envia o comando "SPD" (Serial Poll Disable), que faz com que o dispositivo cesse de enviar o byte de "serial poll". O sinal ATN permanece ativo após a execução da função.

2.14 - IBONL - Colocar STD-8410 "online" ou "offline"

Parâmetros: BD,V

BD especifica o número do cartão interface. Se V diferente de zero, o STD-8410 está habilitado para oprarar (isto é, "online"). Se V for igual a zero, o STD-8410 é mantido em modo "reset", ddesabilitado (offline".

Se o STD-8410 estiver "offline", qualquer chamada de função o colocará automaticamente em "online" antes de se executar a função.

Chamando-se a função IBONL, os parâmetros listados na Tabela VI.2 são carregados com os valores atuais.

2.15 - IBPAD - Mudar o Endereço Primário do STD-8410

Parâmetros: BD,V

BD especifica o cartão interface. V especifica o endereço primário GPIB para o STD-8410.

Apenas os 5 bits menos significativos de V são válidos e não podem ser todos iguais a um, pois existem somente 31 endereços GPIB válidos, na faixa de 0 a 1EH.

O endereço primário é usado para determinar os endereços de "talk" e "listen" do STD-8410. O endereço "listen" do dispositivo é formado somando-se 20H ao endereço primário. O endereço "talk" é formado somando-se 40H ao endereço primário. Consequentemente, um endereço primário 10H corresponde ao endereço "listen" 30H e ao endereço "talk" 50H.

Devido ao fato de o STD-8410 monitorar e atuar em seu próprio endereço quando este for enviado via GPIB, um endereço primário deve ser atribuído ao STD-8410. Se for necessário um endereço diferente do mostrado na Tabela 6.1, a função IBPAD pode ser usada para modificá-lo. Ocorrerá erro se os 5 bits menos significativos de V forem iguais a 1. Veja também as funções IBSAD, OBONL e a Tabela VI.2.

2.16 - IBRD - Ler Dados

Parâmetros: BD,RD,CNT

BD especifica o cartão de interface. RD especifica o "buffer" para armazenamento dos dados que forem lidos do GPIB. CNT especifica o número de dados a serem lidos do GPIB.

A função IBRD é usada para ler um ou mais bytes de dados de dispositivo GPIB. O endereçamento do STD-8410 como "listen" e do outro dispositivo como "talk" deve ser feito separadamente desta função. A função IBCMD pode ser usada para este propósito, desde que o STD-8410 seja o "Controller-In-Charge". Caso um outro dispositivo seja o "Controller-In-Charge", este deverá endereçar.

Se esta função for chamada com o STD-8410 como o Controlador Ativo, este será primeiramente colocado em estado de "Standby" com o sinal ATN desativado e permanecerá neste estado até que a função seja executada.

A operação IBRD termina em um dos seguintes casos:

- o "buffer" alocado fivar cheio;

- erro for detectado;

- o limite de "time-out" for excedido;

- mensagem END for detectada;

- caracter eos for detectado (se esta opção estiver habilitada);

- for recebido um comando de "Device Clear" ou "Selected Device Clear" (SDC) do "Controller-In-Charge" (não do STD-8410).

Após o término, o número de bytes lidos é determinado pela variável IBCNT.

Ocorrerá erro se o STD-8410 como o "Controller-In-Charge" não estiver sido endereçado por si mesmo como "listener" pela função IBCMD, se o STD-8410 não for o "Controller-In-Charge" e não for endereçado como "listener" dentro do limite de "time-out" e também se o "talker" não for endereçado, e/ou a operação não se completar, por qualquer razão, dentro do tempo limite.

2.17 - IBRPP - Conduzir um Poll Paralelo

Parâmetros: BD,PPR

BD especifica o cartão de interface. PPR identifica a variável na qual será armazenada a resposta do "poll" paralelo.

A função IBRPP faz com que STD-8410 conduza o "poll" paralelo de dispositivos previamente configurados e habilitados, enviando a mensagem IDY (os sinais ATN e EOI ativados) e lendo a resposta nas linhas de dados do GPIB.

Ocorrerá erro se o STD-8410 não for o "Controller-In-Charge". Se o STD-8410 for o controlador em "Standby", ele toma o controle e ativa ATN antes de iniciar o "polling", permanecendo após a execução do "polling" como o Controlador Ativo.

2.18 - IBRSC - Requerer ou Liberar o Controle do Sistema

Parâmetros: BD,V

BD especifica o cartão interface. Se V for diferente de zero, as funções que requeiram capacitação de Controlador de Sistema são permitidas. Se V for igual a zero, não são permitidas as funções que requeiram capacitação de Controlador de Sistema

A função IBRSC é usada para permitir ou proibir que o STD-8410 envie mensagens de "Interface Clear" (IFC) ou "Remote Enabl "(REN) a dispositivos GPIB, respectivamente, através das funções IBSIC e IBSRE.

Na maioria da aplicações, o STD-8410 será sempre o Controlador do Sistema. Em casos alternativos, a função IBRSC é usada somente quando o IBM-PC não for o Controlador do Sistema devido à duração da execução de um programa. Enquanto o padrão IEEE-488 especificamente não permite esquemas em que o Controle do sistema seja dinamicamente passado de um dispositivo para outro, a função IBRSC poderia ser usada em tais esquemas. Veja também a Tabela VI.2.

2.19 - IBRSV - Requerer Serviço e/ou ativar ou mudar o Byte de Status do "poll" Serial.

Parâmetros: BD,V

BD especifica o cartão interface. V especifica a resposta ou byte de status que o STD-8410 fornece quando estiver em "poll" serial por um outro dispositivo que seja o "Controller-In-Charge" do GPIB.

Se o bit 40H estiver ativo, o STD-8410 automaticamente requer serviço do Controlador, ativando a linha SRQ do GPIB.

A função IBRSV é usada para requerer serviço do Controlador usando o sinal "Service Request" (SRQ) e fornecer o byte de status quando o controlador executar um "poll" serial no STD-8410.

Não é um erro chamar a função IBRSV quando o STD-8410 for o "Controller-In-Charge", embora isto somente tenha sentido se o controle for futuramente passado a outro dispositivo. Neste caso, a chamada atualiza o byte de status, mas o sinal SRQ é ativado somente se o bit 40H estiver ativo e somente quando o controle for passado. Veja também a Tabela VI.2.

2.20 - IBRTL - Ir de Remoto para Local

Parâmetros: BD

BD especifica o cartão de interface. A função IBRTL pode ser usada para simular a função de controle "retorne para local" do painel frontal de instrumento quando o usuário quiser diferenciar no IBM-PC uma operação "modo-local" de "modo-remoto".

A função IBRTL faz com que o STD-8410 vá do modo remoto para o modo local, desde que o STD-8410 não tenha sido colocado em estado de "lockout". Isto é, a função apaga o bit REM na palavra de status se o bit LOK não estiver ativo.

Após a execução da função IBRTL, utilize a variável para checar o bit de status REM. Se REM=0, o STD-8410 está em "modo-local". Se REM=1, o "Controller-In-Charge" ainda considera que o STD-8410 está em modo de controle remoto e as funções requeridas localmente são ignoradas. A interpretação do "local" e "remoto" é determinada pelo programa do usuário.

2.21 - IBSAD - Mudar ou Desabilitar Endereço Secundário do STD-8410.

Parâmetros: BD,V.

BD especifica o cartão interface. Se V for um número entre 60H e 7EH, este número se torna o endereço secundário GPIB do STD-8410. Se V estiver desta faixa, o endereçamento secundário ou estendido é desabilitado.

A função IBSAD permite ou desabilita endereçamento estendido no GPIB e, quando habilitado, associa o endereço secundário ao STD-8410. Veja também as funções IBPAD, IBONL e a Tabela VI.2.

2.22 - IBSIC - Enviar "Interface Clear" por 100 micro-segundos

Parâmetros: BD

BD especifica o cartão interface. A função IBSIC faz com que o STD-8410 ative o sinal IFC por no mínimo 100 micro-segundos, desde que o STD-8410 esteja com capacidade para Controlador de Sistema. Esta ação inicializa o GPIB e torna o STD-8410 o Controlador Ativo com o sinal ATN ativado.

O sinal IFC apaga somente as funçÕes de interface dos dispositivos e não as funçÕes internas dos dispositivos. As funçÕes internas dos dispositivos são apagadas com os comandos "Device Clear" (DCL) e "Selected Device Clear" (SDC). Para determinar o efeito destas mensagens, consulte a documentação individual do dispositivo.

Ocorrerá erro se o STD-8410 não tiver a capacidade de Controlador do Sistema. Veja também a função IBRSC.

2.23 - IBSRE - Ativar ou Apagar a linha "Remote Enable" (REN)

Parâmetros: BD,V

BD especifica o número do cartão interface. Se V for diferente de zero, o sinal "Remote Enable" (REN) é ativado. Se V for igual a zero, o sinal será apagado.

A função IBSRE liga e desliga o sinal REN. REN é usado pelos dispositivos paea selecionar os modos de operação entre local e remoto. Um dispositivo não entra realmente em modo local até que receba o seu endereço "listen".

Ocorrerá erro se o STD-8410 não tiver capacidade de Controlador de Sistema. Veja também a função IBRSC e a Tabela VI.2.

2.24 - IBTMO - Mudar ou Desabilitar Limite de "Time-out".

Parâmetros: BD,V

BD especifica o número do cartão interface. V especifica o limite em segundos do "time-out". Se V for igual a zero, os "time-outs" são desativados.

A função IBTMO é usada para mudar a quantidade de tempo que as funçÕes IBRD, IBWRT, IBCMD e IBWAIT esperam para terminar a operação ou um evento ocorrer antes de retornar com a indicação de "time-out". A mudança permanece em efeito até que um novo valor seja associado ou a função IBONL seja chamada. Veja também a função IBWAIT e a Tabela VI.2.

2.25 - IBTRG - Executar um "trigger" no dispositivo.

Parâmetros: BD e DISP

BD indica o número do cartão de interface. DISP indica qual o número do dispositivo GPIB que enviará a mensagem. É um valor de dois bytes que contém no byte inferior, o endereço primário do dispositivo (0 a 1EH); no byte superior deverá ser colocado o endereço secundário (60H a 7EH), se houver. Caso contrário, deverá ser colocado 0.

A função endereça e envia um comando de "strigger" para um determinado dispositivo.

IBTRG chama a função IBCMD para enviar basicamente os seguinte comandos:

- "Unlisten" e "Untalk";
- endereço "listen" do dispositivo;
- endereço secundário do dispositivo, se houver;
- GET (Group Execute Trigger").

Consulte a função IBCMD para informaçºes adicionais.

2.26 - IBWAIT - Esperar por Determinado Evento.

Parâmetros: BD,MASK

BD é o número do cartão interface. MASK é a máscara de bits para os mesmos bits da palavra de status, IBSTA. Cada bit da máscara deve ser ativado ou desativado para esperar ou não, respectivamente, que o correspondente evento ocorra.

A função IBWAIT é usada para monitorar os eventos selecionados pela máscara e atrasar o processamento até que algum deles ocorra. A Tabela VI.1 mostra o layout destes bits.

IBWAIT atualiza também todas as condiçºes da palavra de status, que podem ser lidas através da variável IBSTA.

Se a máscara do bit TIMO for 0, ou se o limite de "time-out" for 0, os "time-outs" estão desabilitados. A desabilitação de "time-outs" só pode ser feita quando mask=0 ou quando se tiver certeza da ocorrência do evento; caso contrário, o processador esperará indefinidamente pela ocorrência do evento.

MNEMÔNICO	BIT	DESCRIÇAO
ERR	15	Erro GPIB ou DOS
TIMO	14	Execedeu limite de tempo
END	13	STD-8410 detectou END ou eos
SRQI	12	SRQ detectado
CMPL	8	E/S completada
LOK	7	STD-8410 está em estado de "lockout"
REM	6	STD-8410 está em estado Remoto
SIC	5	STD-8410 é o "Controller-in-Charge"
ATN	4	Sinal ATN ativado
TACS	3	STD-8410 é o "Talker"
LACS	2	STD-8410 é o "Listener"
DTAS	1	STD-8410 em estado de "Device Trigger"
DCAS	0	STD-8410 em estado de "Device Clear"

TABELA VI.1 - MÁSCARAS DE ERRO

2.27 - IBWRT - Escrever Dados

Parâmetros: BD,WRT,CNT

BD especifica o número do cartão interface. WRT contém os dados a serem enviados via GPIB. CNT especifica o número de bytes a serem enviados.

A função IBWRT é usada para enviar um ou mais bytes de dados para um ou mais dispositivos GPIB. O endereçamento do STD-8410 como "talker" e do(s) dispositivo(s) como "listener", deve ser feito separadamente desta função. A função IBCMD é usada para este propósito, desde que o STD-8410 seja o "Controller-In-Charge", este deve fazer o endereçamento, e neste caso a função IBWIT deve ser usada pelo STD-8410 para esperar pelo endereçamento, evitando um possível "time-out" na chamada de IBWRT.

Se esta função for chamada quando o STD-8410 for o Controlador Ativo, o STD-8410 é colocado primeiramente em estado de "standby" com o ATN desligado e permanece neste estado após a execução de função.

A função IBWRT termina em qualquer dos seguintes eventos:

- todos os bytes da cadeia foram transferidos;

- foi detectado um erro;

- o limite de "time-out" foi excedido;

- o comando de "Device Clear" (DCL) ou "Selected Device Clear" (SDC) foi recebido do "Controller-In-Charge" (não do STD-8410).

Após a transmissão, o número de bytes realmente transmitidos pode ser determinado pela variável IBCNT.

Ocorrerá erro se o STD-8410 como "Controller-In-Charge" não tiver sido endereçado por si mesmo para "talk" com a função IBCMD, se o STD-8410 não for o "Controller-In-Charge" e tiver sido endereçado para "talk" dentro do limite de "time-out" ou ainda se nenhum "listener" tiver endereçado quando o primeiro byte foi enviado para o GPIB.

3 - VALORES "DEFAULT" DOS PARAMETROS DE SOFTWARE

Toda a vez que o Sistema Operacional for inicializado no microcomputador ou a função IBONL for executada, alguns parâmetros de software são carregados com os seus valores "default". Esses valores e as respectivas funções usadas para modificá-los estão apresentados na tabela abaixo (VI.2).

PARAMETRO	DEFAULT
DMA ou E/S	DMA 1
Byte fim de Seqüência	0AH
Enviar EOI com o último byte	NAO
Bit de Status Individual	0
Mensagem de Poll Local	0
Endereço Primário	0
Habilitar Controlador do Sistema	SIM
Byte de Status de Poll Serial	0
Endereço Secundário	DESABI.
Sinal de Habilitação Remota	FALSO
"Timeout" em segundos	5
Endereço base p/ os cartões STD-8410	2B8H
"Settling Time" p/ cartões STD-8410	LONGO

TABELA VI.2 - VALORES "DEFAULT" DOS PARAMETROS DE SOFTWARE