Ferramentas Computacionais para Sistemas de Aquisição de Dados Aplicados ao Ensino de Eletrônica

Maurício de Vasconcelos Affonso

Orientador: Prof. Dr. Carlos Ignácio Zamitti Mammana

Banca Examinadora:
Prof. Dr. Carlos Ignácio Zamitti Mammana
Prof. Dr. João Antonio Zuffo
Prof. Dr. Furio Damiani

Este exemplar corresponde à redação final da tese defendida por Maurício de Vasconcelos Affonso e aprovada pela Comissão Julgadora em 2 4 93.

C. I. Z. Mammana
Orientador

Dissertação apresentada à Faculdade de Engenharia Elétrica como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica

Março de 1993

Departamento de Semicondutores, Instrumentos e Fotônica da Faculdade de Engenharia Elétrica

Universidade Estadual de Campinas - UNICAMP

ABSTRACT

Visando contribuir para o enriquecimento dos meios disponíveis para a formação de recursos humanos na área de eletrônica, foi proposto o desenvolvimento de um Sistema de Aquisição de Dados - SAD que, atuando de forma integrada com o Sistema Didático de Projetos - SDP - utilizado nos laboratórios de Microeletrônica da Escola Brasileiro-Argentina de Informática, tem como objetivo compor um ambiente completo - com baixo custo - voltado a laboratórios de ensino de eletrônica.

Este trabalho, inserido nos objetivos mais amplos propostos para o SAD, abrange: **a)** a especificação de linguagens textuais para a descrição de experimentos em circuitos eletrônicos, baseadas em linguagens de descrição de simulações; **b)** a descrição da implementação dos protótipos de duas ferramentas computacionais, a primeira delas voltada à análise gráfica de resultados de simulações e medidas analógicas e paramétricas e a segunda voltada à programação, execução e análise de resultados de testes funcionais de circuitos de lógica digital. Através dessas ferramentas será possível não só a comparação entre os resultados experimentais e de simulação como também a extração de parâmetros de dispositivos e circuitos eletrônicos.

A meus filhos,
Pedro Henrique e Gustavo,

a minha esposa,
Lilaine

e a meus pais,
Sabino e Lúcia

Conteúdo

Relação dos Anexos

I - Introdução

I.1 - Motivação

Foi proposto junto ao Programa Argentino-Brasileiro de Pesquisas e Estudos Avançados em Informática o desenvolvimento do protótipo de um sistema computacional destinado ao ensino de eletrônica, visando à sua eventual reprodução e disseminação pelas universidades brasileiras e argentinas.

Esse sistema deverá ser constituído pela integração de dois outros sistemas. O primeiro deles é um sistema computacional de apoio ao projeto de circuitos integrados, denominado Sistema Didático de Projetos (SDP), descrito em [Mam87a] e [Mam87b], concebido e desenvolvido através de cooperação entre o Instituto de Microeletrônica do CTI, o Departamento de Ciência da Computação do IMECC/UNICAMP e o Departamento de Semicondutores, Instrumentos e Fotônica (DSIF) da FEE/UNICAMP. O SDP tem sido utilizado com êxito nos últimos cinco anos em cursos intensivos ministrados nas Escolas Brasileiro-Argentinas de Informática (EBAI), na UNICAMP e na Universidade Federal do Paraná. O SDP permite que seja realizada a programação de um curso e seja utilizado um conjunto de ferramentas de auxílio ao projeto, visando a aspectos referentes à engenharia de sistemas, engenharia de circuitos (células) e engenharia da pastilha.

O segundo sistema a ser integrado será o Sistema de Aquisição de Dados - SAD, objeto deste trabalho. Através do mesmo será viabilizado um sistema voltado a laboratórios de eletrônica em geral, com a introdução de diversas facilidades e ferramentas de baixo custo voltadas à realização de testes lógicos e paramétricos de circuitos e sistemas, permitindo a realização de projetos complexos com a introdução de experiências que levam em conta as flutuações estatísticas e a interação entre os problemas de análise, síntese e verificação experimental.

Pretende-se portanto, através da agregação do Sistema de Aquisição de Dados - SAD ao Sistema Didático de Projetos - SDP, realizar um grande número de experimentos dos cursos de eletrônica, viabilizando experimentos mais complexos e de uma maneira mais ágil, colocando-se à disposição do aluno facilidades voltadas ao modelamento, simulação, medição, tratamento e análise dos dados, além da integração das atividades do aluno.

I.2 - Introdução ao SAD

O SAD compõe-se, basicamente, da integração entre um sistema de *software* e um conjunto de instrumentos de medição e controle interligados ao microcomputador através de interfaces apropriadas. O sistema de *software* é o responsável pelo controle dos equipamentos e aquisição de dados dos mesmos, provendo também um conjunto de ferramentas de suporte que abrangerá as etapas de programação do experimento, visualização dos resultados dos testes, análises matemático-estatísticas, tratamento dos dados e extração de parâmetros dos dispositivos em estudo.

Considerando que o sistema tem como alvo o uso didático, procurou-se defini-lo de forma que a facilidade de uso fosse uma de suas características principais, sem que isso comprometesse, no entanto, a necessidade de o aluno compreender cada etapa do processo envolvido na experimentação e validação dos circuitos.

O conjunto de operações do SAD pode ser classificado em dois grandes grupos:

- Operações voltadas à programação das medidas
- Operações de análise e tratamento de resultados

As operações voltadas às medidas abrangem testes de lógica funcional, testes de comportamento analógico e testes paramétricos dos circuitos em estudo. Neste grupo de operações estão compreendidas as etapas de programação do experimento, que poderá ser realizada através de uma descrição textual dos testes, segundo uma linguagem de programação definida ou, ainda, por meio de programação realizada de maneira interativa; a execução dos testes em si, na qual o sistema de *software* controlará a aplicação dos estímulos programados ao circuito sob teste e os instrumentos de medição, armazenando apropriadamente os dados de resultados dos experimentos; a visualização dos resultados dos testes, que oferecerá ao usuário, em forma gráfica e textual, os dados de resultados obtidos dos testes.

As operações de análise e tratamento dos resultados dos testes visam oferecer ao usuário facilidades para a avaliação e crítica dos resultados, sob a ótica qualitativa e quantitativa, facilitando ao aluno a exploração da experimentação em si e de modelos matemáticos e estatísticos associados aos dispositivos em estudo e às técnicas de projeto e testes.

Com vistas à validação da lógica digital, o aluno poderá realizar análises comparativas entre os resultados dos testes efetuados e os resultados obtidos através do simulador lógico SIMUL [Pan87a], que é uma das ferramentas já integradas ao SDP.

A definição de linguagens para a programação dos experimentos de testes funcionais, paramétricos e de comportamento analógico, tem como objetivo permitir ao aluno a abstração de detalhes de características particulares do *hardware* voltado à aquisição dos dados, além de uma melhor organização e sistematização dos experimentos. Face à semelhança entre as informações necessárias à definição de simulações e de experimentos de testes e tendo em vista a possibilidade de intercambiar dados entre programas de simulações e de testes, optou-se pela definição de linguagens para a programação dos testes que se assemelhem, semântica e sintaticamente às dos simuladores. Para os testes funcionais de lógica digital, o SAD utilizará uma linguagem semelhante à do simulador SIMUL [Pan87a], e para testes paramétricos e de comportamento analógico a linguagem será semelhante à do SPICE [Vla81].

Para a análise dos resultados de testes de comportamento analógico e de medidas paramétricas, o aluno terá disponível o STAG - Sistema de Tratamento e Análise Gráfica de Dados, que é uma das ferramentas do SAD e deverá atuar de forma a integrar todo o espectro de informações sobre os resultados desses testes, provendo suporte à realização de análises comparativas entre resultados de simulações e medições, extração de parâmetros físicos e elétricos dos dispositivos, dados para otimização do projeto e outras informações de interesse. Através do STAG o aluno poderá realizar o traçado e a sobreposição de curvas lineares ou logarítmicas, medir a distância entre dois pontos da curva e a inclinação da reta que passa por eles, realizar ajustes de curvas, entre outros tratamentos. São oferecidos ainda recursos para análises estatísticas básicas, cálculo de transformadas de Fourier (FFTs) e a aplicação de regressões aos dados. Os dados a serem tratados e analisados através do STAG poderão ser provenientes de resultados de simulações (por exemplo, através do simulador elétrico SPICE), de testes realizados no próprio SAD, ou ainda de outras ferramentas, integradas ou não ao SDP e ao SAD.

I.3 - O Escopo deste Trabalho

Este trabalho, inserido nos objetivos mais amplos propostos para o SAD, abrange os seguintes módulos:

- Especificação de linguagens textuais para a descrição dos experimentos.
- Desenvolvimento do protótipo de um programa de computador voltado ao tratamento e análise de resultados de medidas.
- Desenvolvimento do protótipo de um programa para a definição de experimentos, execução e análise de resultados de testes funcionais.

No capítulo II fazemos uma descrição geral do SAD, abrangendo os objetivos e requisitos propostos para o sistema, propondo também ferramentas e técnicas a serem utilizadas para o desenvolvimento do *software*.

O capítulo III descreve as especificações para as linguagens de programação de experimento definidas para o SAD.

Nos capítulos IV e V são apresentadas as especificações de requisitos, o projeto detalhado e as principais características da atuais implementações do STAG-Sistema para Tratamento e Análise de Dados e do STEF-Sistema Testes e Análises Funcionais, respectivamente.

No capítulo VI apresentamos as análises e conclusões, além de algumas sugestões de evoluções e melhorias futuras.

II - Descrição Geral do SAD

II.1 - Introdução

O Sistema de Aquisição de Dados - SAD tem por objetivo implementar um conjunto de operações voltadas à caracterização de circuitos e dispositivos eletrônicos, abrangendo a realização de testes funcionais, paramétricos e de comportamento analógico. A partir dos resultados dos testes o sistema dará suporte à aplicação de um amplo conjunto de operações voltadas ao tratamento e análise dos dados, permitindo a visualização gráfica dos resultados, análises comparativas entre simulações e medições realizadas, extração de parâmetros para dispositivos e dados para otimizações, entre outros.

O SAD destina-se a um ambiente didático e deverá, em conjunto com o Sistema Didático de Projetos-SDP, implementar um ambiente completo voltado a laboratórios de eletrônica em geral. Nesse ambiente o aluno terá suporte a praticamente todas as etapas de concepção do circuito, desde o estabelecimento dos requisitos básicos até os testes finais de homologação do protótipo, tomando como base a metodologia descrita em [Mam87].

Uma das principais preocupações na especificação do SAD foi a de definir um sistema que com um custo de *hardware* relativamente baixo possa oferecer ao aluno um amplo conjunto de funções para a aquisição e tratamento de dados. Dessa forma, a disseminação do sistema entre as universidades não encontraria nos custos um fator impeditivo à sua implantação.

Discutiremos neste capítulo as principais características do SAD. Faremos inicialmente uma breve discussão sobre as características da plataforma de execução do sistema, no que diz respeito ao *hardware* e ao *software* básico. Em seguida discutiremos as principais características do ambiente de desenvolvimento. Por fim, faremos a especificação dos requisitos funcionais do sistema de *software* para o SAD.

A figura 1 apresenta um diagrama esquemático genérico dos processos e dados relacionados ao ambiente proposto.

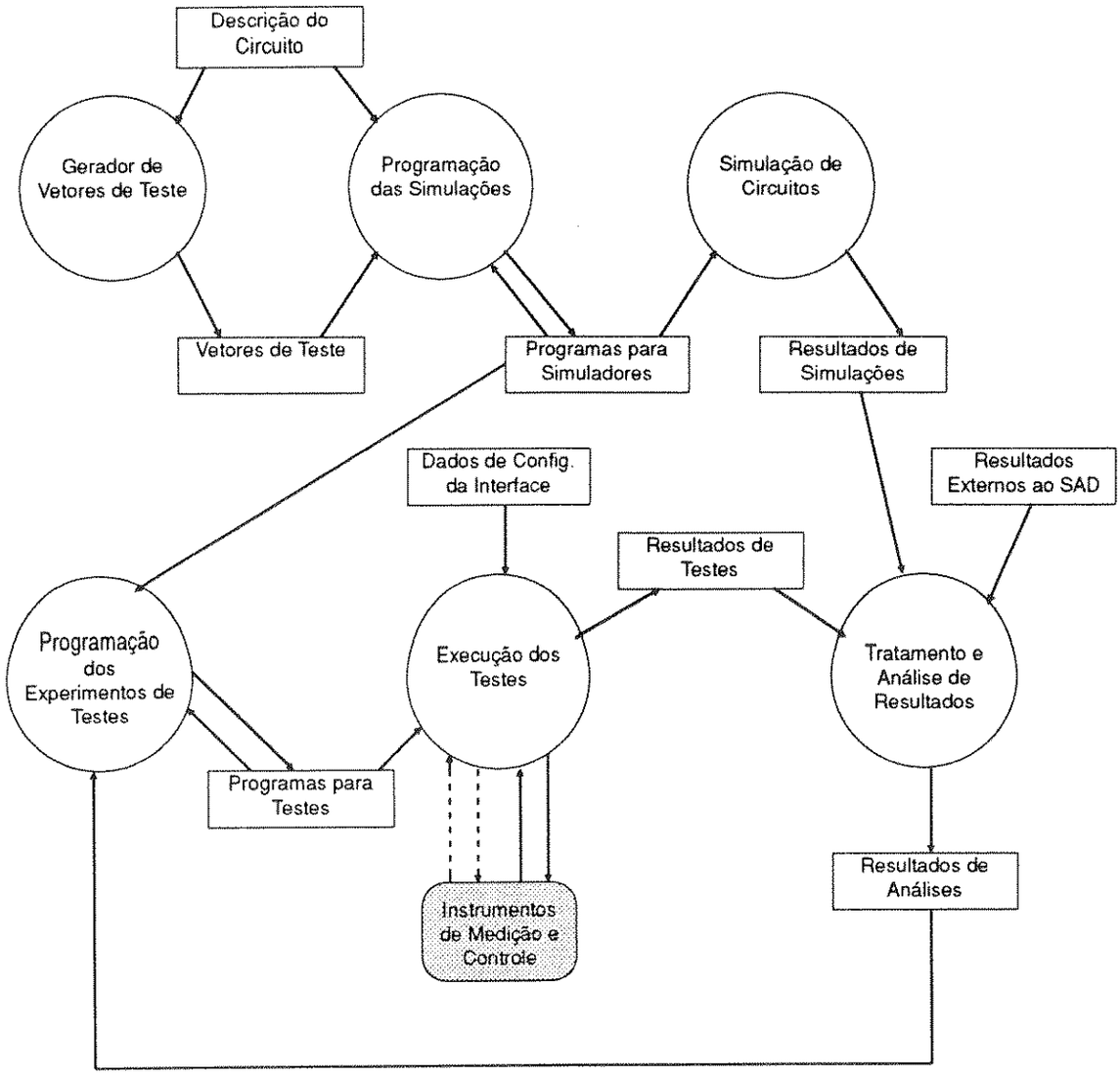


Figura 1 - Diagrama Genérico dos Processos do SAD

II.2 - Características Ambientais para o SAD

Descreveremos nesta seção as principais características ambientais ligadas ao SAD, abrangendo os aspectos de *hardware* e *software*, tanto no que diz respeito à plataforma de execução do sistema quanto no que se refere à plataforma de desenvolvimento e que constituem condições de contorno para o desenvolvimento deste trabalho.

II.2.1 - Plataforma de *Hardware*

A plataforma de *hardware* para o SAD compõe-se basicamente de um microcomputador interligado através de interfaces de aquisição de dados adequadas a um conjunto de instrumentos de medição e controle controlados pelo microcomputador e conectados a este através de conectores de expansão do barramento do mesmo.

São utilizadas pelo sistema duas interfaces de aquisição de dados, uma dedicada aos testes de lógica funcional e a outra dedicada aos testes paramétricos e de comportamento analógico.

A interface para testes de lógica funcional [Qua88] tem por objetivo a simples aplicação de vetores de teste ao circuito e, em seguida, a observação das saídas de maneira estática.

A interface para os testes analógicos tem como objetivo realizar o controle dos diversos equipamentos de medição e controle disponíveis e efetuar a aplicação de estímulos e a medição das respostas analógicas, através do uso de conversores digital-analógicos e analógico-digitais e interfaces adequadas.

Pretende-se permitir ao SAD a utilização de diferentes tipos de interfaces de aquisição de dados, oferecendo-se assim uma maior flexibilidade aos usuários do sistema com relação à acuidade e capacidades específicas nos testes.

Os requisitos mínimos exigidos para o microcomputador e para as interfaces de aquisição de dados são descritos nas seções II.2.1.1 a II.2.1.3.

II.2.1.1 - Requisitos para o Microcomputador

O SAD foi projetado e desenvolvido para operar em computadores da linha PC-xt ou compatível, com os seguintes requisitos mínimos:

- Unidade Central de Processamento (CPU) de 16 bits compatível com os microprocessadores da família 80x86, da Intel.
- 640 kbytes de memória principal.
- Controlador de vídeo gráfico com resolução mínima de 640x200 pontos (padrão CGA) ou superior.
- Disco rígido de 20 Mbytes.
- 01 interface paralela.
- 01 interface serial, padrão RS-232.
- Impressora (opcional)
- Mouse (opcional)

Em função do volume de processamento e de aspectos da interação usuário-software, uma configuração aconselhável consistiria num computador com CPU compatível com o 80286 ou 80386, memória principal de 1 Mbyte, disco rígido de 40 Mbytes e monitor de vídeo padrão EGA ou VGA.

A escolha dessa família de microcomputadores foi motivada principalmente pelo baixo custo do equipamento e pelo parque instalado desse tipo de máquina. A proposta do SAD tem como premissa a criação de um sistema de baixo custo, de forma a viabilizar a disseminação do sistema em universidades e outras entidades de ensino. Apesar da adoção dessa plataforma, existe interesse de, em versões futuras do sistema, migrar para plataformas mais eficientes, mantendo porém a disponibilidade do sistema de baixo custo.

Outros fatores que influenciaram também na escolha do microcomputador foram a compatibilidade com a plataforma de execução do SDP; a escalabilidade da capacidade de processamento, que permite a migração para um equipamento superior, da mesma família; facilidade de obtenção de interfaces e componentes de *hardware* no mercado; quantidade e qualidade de ferramentas para o desenvolvimento de *software*.

II.2.1.2 - Interface para Testes Funcionais da Lógica Digital

A interface para testes funcionais da lógica digital deverá permitir a aplicação de um conjunto pré-definido de estímulos (vetores de teste) aos circuitos e realizar a leitura dos estados lógicos digitais presentes em pontos determinados (saídas) do circuito, definidos pelo usuário.

Dadas as características da aplicação deste tipo de teste, foi estabelecida uma taxa mínima de aquisição dos dados de cem leituras por segundo, o que simplifica a implementação do circuito e do *software*. A interface deverá ainda ser capaz de aplicar estímulos ou medir respostas de, no mínimo, 24 pontos do circuito.

Um protótipo de interface para este tipo de medidas foi projetado e desenvolvido por M.A.Quatorze [Qua88], tendo sido apresentado e utilizado durante a IV EBAI. Esse protótipo comporta até 24 sinais de entrada ou saída compatíveis com os níveis TTL. Para o controle da interface, são utilizadas 8 portas de entrada e saída do computador, configuráveis através de chaves seletoras (*dip-switches*).

A interface de aquisição de dados para testes funcionais, no protótipo de Quatorze, está interligada a um conector universal, ao qual o usuário deverá interligar o circuito sob testes. No anexo VI é apresentado o diagrama esquemático do circuito desse protótipo.

II.2.1.3 - Interface para Medidas de Desempenho Analógico

A interface para medidas de comportamento analógico estará interligada a um conjunto de instrumentos dedicados de controle e medição, os quais deverão ser por ela controlados. Através do sistema de *software* deverá ser possível o controle e a programação de cada um dos instrumentos.

Descreveremos a seguir os requisitos estabelecidos para a interface, bem como para os instrumentos a ela interligados.

II.2.1.3.1 - Características para o Conversor Analógico-digital

A placa conversora analógico-digital deverá receber como entradas tensões contínuas ou alternadas (até 200 V), correntes DC ou AC até algumas dezenas de miliamperes, tensões propocionais à temperatura e tensões proporcionais à resistência (obtidas através da injeção de corrente).

Definem-se abaixo os requisitos mínimos para o conversor analógico digital da interface de aquisição de dados:

- Resolução de Leitura: 10 bits.
- Taxa de Amostragem: 10000 conversões/segundo.
- Entradas de tensão:No mínimo 4, no máximo 8.
- Entradas com conversor de corrente:mínimo 2, máximo 4.
- Entradas com conversão de temperatura:1.
- Entradas com conversão de resistência:1.

Placas de aquisição de dados com as características acima descritas são hoje facilmente encontradas no mercado, a preços acessíveis. Eventualmente poderão ser desejáveis placas com características mais sofisticadas com relação a desempenho e precisão. O sistema de *software* deve, portanto, ser desenvolvido prevendo a possibilidade desse tipo de evolução.

II.2.1.3.2 - Características dos Instrumentos de Controle e Medição

Paralelamente a este trabalho estão sendo desenvolvidos instrumentos de controle e medição que deverão estar conectados à placa de aquisição dos dados. Esse conjunto de instrumentos tem como requisitos básicos as seguintes características:

a) Geração de Estímulos

- Fontes Programáveis
  - Fonte Simétrica: 0 a ± 10 V, 0.50 A
  - Fonte de baixa Corrente: -10 a +10 V, 50 mA
  - Fonte de tensão constante: 5.0 V, 1.0 A
- Conversor D/A até 100 kHz
- Gerador de Funções
- Controlador de Temperatura Programável
  - Faixa de temperaturas: -50° a +150° C

b) Medição de Respostas

- Multímetro
  - Medição AC e DC
  - Número de canais: 8, selecionáveis por *software*
  - Ajuste de escalas: Automático
  - Impedância de Entrada: Maior que 10 MOhm
  - Resolução: 10 ou 12 bits
  - Escalas de Tensão: 5, com fundos de escala de ± 20 mV a ± 200 V
  - Conversores corrente-tensão:4, com fundos de escalas de 0.1 μA a 20 mA
  - Ohmímetro: com fundos de escala de 10 Ω a 100 MΩ
  - Termômetro : Faixa de -50°C a +150°C

c) Medição de Parâmetros

- Capacímetro de pequenos sinais
  - 4 faixas de medição: 1 pF a 1 ηF
  - Freqüência de medição: 1 MHz
  - Tensão de pico da excitação: 10 mV
  - Polarização CC através de fonte externa

A alternativa de utilizar equipamentos autônomos interligados, por exemplo, através de uma rede IEEE-488, poderia também ser considerada. Trabalho nesse sentido foi realizado por J. O. Simões [Sim91]. A adoção dessa alternativa, porém, levaria o custo global do sistema a um patamar extremamente elevado para a aplicação a que se destina, podendo inviabilizar a disseminação do sistema. Além disso, dado o caráter didático da aplicação, a precisão e resolução desejáveis são bastante modestos e, portanto, os ganhos obtidos com o uso de equipamentos autônomos não são significaticos para a aplicação.

II.2.2 - Plataforma de *Software*

Na definição do ambiente básico de *software* para o SAD, julgou-se conveniente estabelecer plataformas idênticas para o desenvolvimento e para a execução do sistema, visando com isso facilitar alguns aspectos ligados à implementação, testabilidade e manutenção do sistema.

Os principais fatores considerados na escolha do ambiente de *software* para o sistema foram: compatibilidade com o SDP, a fim de viabilizar a integração entre os dois sistemas; a quantidade de recursos voltados ao desenvolvimento oferecidos; recursos disponíveis ao usuário final; custo do sistema básico; parque instalado e disseminação da cultura.

A seguir serão discutidos os principais aspectos ligados ao ambiente básico de *software* para o SAD.

II.2.2.1 - Sistema Operacional

O sistema operacional sob o qual o SAD é executável deve ser compatível com o MS-DOS 3.1 ou superior. Essa família de sistemas operacionais é a mais difundida em microcomputadores e provê os recursos necessários à implementação do SAD. A disponibilidade de um grande número de ferramentas e aplicativos existentes no mercado para o MS-DOS é, também, importante fator a ser considerado no desenvolvimento do sistema.

II.2.2.2 - Linguagem de Programação do Sistema

A linguagem de programação escolhida para a implementação do SAD foi o Pascal. A escolha dessa linguagem teve como um dos principais fatores tratar-se de uma linguagem de alto nível e de excelente estruturação, de grande importância durante o desenvolvimento com relação a aspectos de modularidade, reutilizabilidade e manutenção do sistema, influenciando diretamente a qualidade global do mesmo. Além disso, essa é uma linguagem bastante difundida no meios acadêmicos, onde o SAD vem sendo desenvolvido e pelo qual deverá ser realizada a implementação de novos recursos e manutenção do sistema.

O compilador escolhido foi o Turbo Pascal, da Borland, bastante difundido e sem dúvida um dos melhores compiladores desta linguagem disponíveis para o sistema operacional adotado. Esse compilador possui vasta biblioteca de funções padronizadas e incorpora, ainda, em suas versões recentes, ferramentas de suporte à programação como, por exemplo, um depurador interativo a nível de programa-fonte ("*source debugger*"), ferramenta importante com relação à testabilidade do sistema. Ainda, como a maior parte das ferramentas do SDP foi desenvolvida utilizando-se o mesmo compilador, viabilizou-se a criação e utilização de bibliotecas particulares de funções comuns a diversas ferramentas.

Deve-se ressaltar que, embora as ferramentas do SDP e do SAD sejam quase todas escritas em Pascal, não há restrições quanto à utilização de ferramentas escritas em outras linguagens, desde que se observem as boas práticas de programação.

Como outros programas para o MS-DOS, o SAD poderá ser executável a partir do ambiente *MS-Windows* (via *DOS-Shell*), com algumas restrições relativas às interfaces de aquisição de dados. Não se obtêm, no entanto, os principais benefícios oferecidos pelo *Windows*, como a interface gráfica padronizada e o gerenciamento dos periféricos disponíveis.

Encontra-se em estudo de viabilidade a criação de uma versão do SAD dedicada ao ambiente *Windows*, que poderia trazer sem dúvidas grandes melhorias sob vários aspectos, como por exemplo, a interatividade e a capacidade de interfaceamento com diferentes periféricos. Como desvantagens, teríamos principalmente o fator custo do sistema, dado que o ambiente *Windows* requer maior capacidade de processamento do computador, monitor de vídeo de alta resolução, entre outros.

II.3 - Técnicas e Ferramentas de Suporte ao Desenvolvimento

As ferramentas e técnicas de suporte ao desenvolvimento do projeto constituem um importante fator na definição do ambiente de desenvolvimento de *software*, dado que as mesmas têm forte influência na determinação da qualidade global do sistema.

As técnicas de projeto estruturado oferecem recursos baseados num conjunto de processos heurísticos de desenvolvimento e prevêem ferramentas capazes de auxiliar nas diversas fases do projeto, atuando desde a etapa de especificação dos requisitos básicos para o sistema até as etapas de implementação e manutenção do mesmo. São incluidas nessas metodologias técnicas voltadas à partição do sistema em módulos funcionais de processamento, de forma a minimizar e padronizar as interfaces entre os módulos, com a finalidade de se obter o máximo de independência funcional entre os mesmos. Estão disponíveis, também, técnicas para o modelamento lógico e físico dos dados. Ferramentas de suporte ao modelamento, como de diagramas gráficos para a descrição dos processos e dos dados, o uso de um dicionário de dados do sistema, entre outros, auxiliam na verificação da consistência global do projeto, bem como na predição da sua qualidade. O uso das técnicas de projeto estruturado apresenta como resultado um sistema mais robusto, eficiente e adequado.

A documentação do sistema apresenta-se, também, como fator determinante da qualidade e facilidade de manutenção do sistema. A documentação deve abranger os diversos níveis de visão do sistema (visão geral, de cada módulo, de procedimentos e de dados), bem como as diversas etapas de desenvolvimento do projeto. A utilização de diagramas esquemáticos do sistema, de dicionários de dados, de descrição das interfaces de módulos e rotinas e de restrições verificadas na etapa de implementação são de grande importância para a posterior evolução e manutenção do sistema.

Para a documentação da etapa de implementação, propusemos em [Aff89] uma padronização da codificação para programas em Pascal, baseada em [Par72] e [Gui86], onde são estabelecidas diretrizes básicas para a especificação de interfaces de rotinas e entre módulos. São estabelecidas também convenções para a codificação e para nomenclaturas de nomes e identificadores. Ainda, são definidas padronizações para comentários a respeito dos módulos e rotinas do programa, com a finalidade de permitir, via programa utilitário, a extração automática de documentação dos programas-fonte.

Foram implementadas [Aff91], para dar suporte ao desenvolvimento do sistema, ferramentas automatizadas para a criação e manutenção dos sistemas de *help* do SAD (que são implementados na forma de hipertexto) e um extrator automático de documentação de programas-fonte [Aff92]. Essas ferramentas foram construídas de forma a serem genéricas, podendo portanto ser utilizadas no desenvolvimento de outros sistemas. Uma discussão mais detalhada dessas ferramentas é apresentada no anexo II.

Outras ferramentas automatizadas teriam sido desejáveis e poderão ser incorporadas e/ou implementadas futuramente, para dar suporte às técnicas de modelamento empregadas, como um dicionário de dados interativo capaz de realizar consistências automáticas, um editor gráfico de diagramas esquemáticos integrado a um módulo para simulação do modelo do sistema, entre outras. A utilização desse tipo de ferramentas poderia, sem dúvida ter aumentado significativamente a produtividade em algumas das etapas de desenvolvimento e manutenção do sistema. Teria ainda sido de grande valia a utilização de ferramentas para controle de versões dos programas.

II.4 - Requisitos Funcionais de *Software* para o SAD

II.4.1 - Introdução

Como descrevemos anteriormente, o SAD será composto através da integração entre um sistema de *software* e um conjunto de instrumentos de medição e controle, os quais deverão estar conectados ao computador através de interfaces apropriadas. O sistema de *software* será o responsável pelo controle dos instrumentos, oferecendo ainda um conjunto de ferramentas de alto nível que propicie ao aluno recursos voltados à aquisição, tratamento e análise dos dados, além de uma melhor organização de suas atividades e documentos.

Especificaremos nas seções seguintes os principais requisitos funcionais do sistema de *software* para o SAD. Faremos inicialmente uma descrição geral do sistema, passando em seguida à especificação dos principais requisitos de *software*, à definição das prioridades de implementação e, por fim, à descrição das principais evoluções previstas para versões futuras do sistema.

II.4.2 - Descrição Geral do SAD

II.4.2.1 - Perspectivas do Sistema

O SAD tem como ambiente-alvo laboratórios de eletrônica de escolas técnicas e universidades, destinando-se à programação e análise de resultados de medidas realizadas em circuitos eletrônicos. Atuando de forma integrada com o SDP, pretende-se obter um ambiente que, com baixo custo de equipamentos, auxilie o aluno durante praticamente todas as etapas da concepção e validação de projetos de circuitos e dispositivos eletrônicos.

II.4.2.2 - Características dos Usuários

O grupo de usuários previsto para o sistema compõe-se por alunos de graduação e pós-graduação na área de eletrônica. A implantação de um sistema como o SAD pode oferecer um incremento na quantidade e qualidade de experimentos realizados nos cursos e, ainda, propiciar a ambientação dos estudantes com ferramentas automatizadas, as quais têm se difundido rapidamente no mercado de trabalho. Ainda, os alunos dessa área têm conhecimento e alguma cultura na utilização de sistemas de computação, o que torna menos traumática a implantação do sistema.

II.4.2.3 - Suposições, Dependências e Restrições

O desenvolvimento do sistema foi proposto supondo-se ter como ambiente de execução aquele especificado na seção II.2 deste trabalho.

Mesmo com o uso das mais diversas técnicas de análise e programação, a implementação do sistema apresentará particularidades ligadas ao sistema de *hardware* voltado à aquisição dos dados. Cuidado especial deve ser tomado na especificação das interfaces de rotinas ligadas às funções de aquisição de dados, de forma que o esforço para a implementação de suporte a novos tipos de interfaces de aquisição de dados seja minimizado. Deverá ser previsto suporte a placas nas quais os dados sejam lidos através das portas de entrada e saída ou por acesso direto à memória (DMA). As estruturas de dados envolvidas na aquisição de dados devem prever resoluções de até 16 bits (ou maiores).

O desempenho de algumas das funções de análise de resultados dos testes poderá ser limitado devido à velocidade de processamento do computador utilizado, não devendo ser, no entanto, um aspecto crítico.

Para a análise gráfica de resultados de testes, a resolução do monitor de vídeo poderá apresentar limitações na visualização gráfica dos dados.

II.4.3 - Especificação de Requisitos

II.4.3.1 - Requisitos Funcionais

O SAD tem como objetivo básico auxiliar o aluno na realização de duas classes distintas de testes:

- Testes de Lógica Funcional
- Medidas Paramétricas e de Comportamento Analógico

Embora o nível de abstração de detalhes dos circuitos em estudo sejam diferentes para essas duas classes de testes, implicando inclusive no uso de interfaces distintas, em ambos os casos podemos considerar para o SAD os seguintes módulos básicos:

- Programação de Experimentos
- Execução de Testes e Medidas
- Tratamento e Análise de Resultados de Testes e Simulações

Integrando esses três processos básicos, o sistema deverá possuir um módulo gerenciador de projetos, cujo objetivo básico será facilitar ao aluno a organização dos dados envolvidos em cada projeto. Teremos ainda um módulo dedicado ao tratamento da interface do sistema com o usuário, que deverá oferecer uma melhor uniformidade a essa interface.

Na figura 2 apresentamos um diagrama do sistema, destacando os módulos acima descritos, bem como os dados a eles associados. Discutiremos a seguir cada um dos ítens acima citados.

II.4.3.1.1 - Descrição dos Processos Básicos do SAD

A) Programação do Experimento

Através das operações de programação do experimento, o aluno poderá definir os conjuntos de estímulos a serem aplicados ao circuito em estudo ao longo do tempo e as medições a serem efetuadas nos pontos do circuito a serem monitorados, além de condições ambientais para a realização do experimento.

A programação do experimento poderá ser realizada através de linguagens de programação definidas especificamente para os diferentes tipos de testes a serem realizados.

A descrição do experimento poderá também ser realizada de forma interativa, através do uso de ferramentas de *software* projetadas especialmente para essa finalidade. No caso da programação interativa, os vetores de testes gerados pelo sistema devem ser compatíveis com a linguagem de programação do experimento. A utilização de ferramentas voltadas à descrição interativa do experimento tornam a atividade do aluno mais agradável e produtiva.

Devemos considerar também que os vetores de testes a serem utilizados para a execução do experimento poderão ser gerados através de outras ferramentas de *software*. Embora a versão atual do SDP não possua geradores automáticos para os vetores de teste, deve-se prever suporte à existência futura desse tipo de ferramenta, nesse sistema ou fora dele.

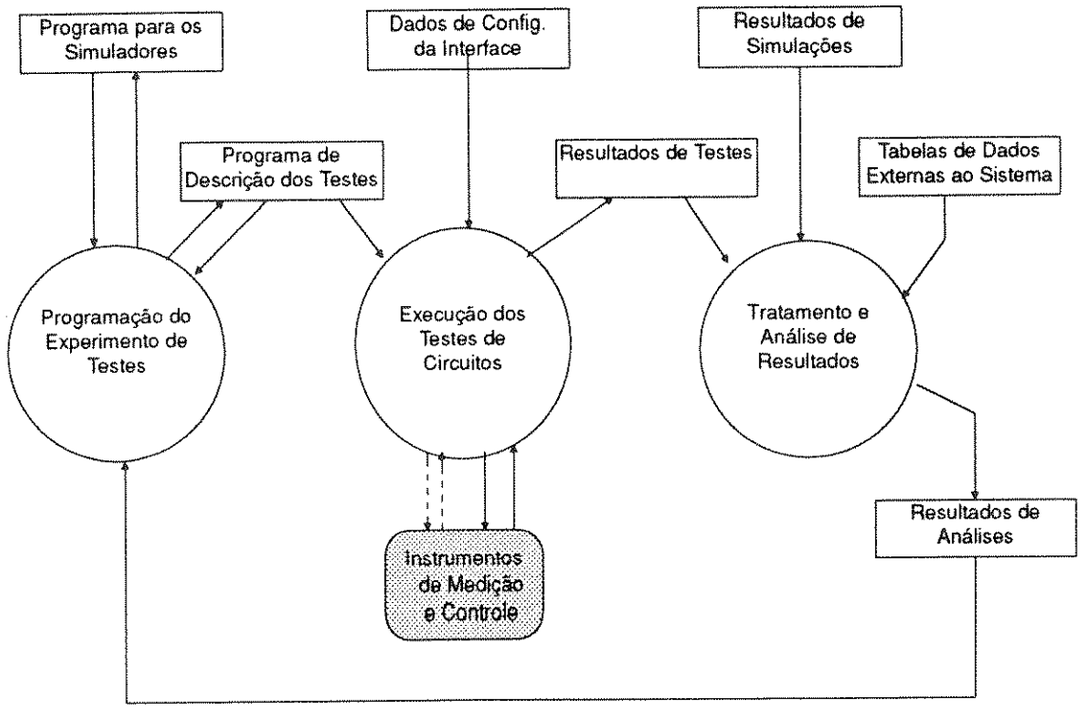


Figura 2 - Visão Geral dos Processos Voltados aos Testes

É conveniente, ainda, que as linguagens de programação do experimento sejam compatíveis, na medida do possível, com as linguagens utilizadas pelos simuladores integrados ao SDP (o SIMUL, no caso da simulação lógica e o SPICE, no caso de simulação elétrica), permitindo assim a criação de funções que permitam que os programas de testes gerados possam ser utilizados também para a simulação e vice-versa, evitando que o aluno necessite duplicar seus esforços definindo um programa para simulação e outro para testes. Além disso, compatibilizando testes e simulações, o usuário será capaz de realizar comparações entre os resultados dessas duas ferramentas de validação do projeto.

Inicialmente, poderíamos considerar a conveniência da definição de uma linguagem única para a definição de experimentos de validação da lógica funcional, desempenho analógico e testes paramétricos. Nota-se, porém, que o nível de abstração para os testes de lógica funcional difere dos testes elétricos, assim como o conjunto de informações necessárias a cada um deles e as próprias estruturas das linguagens dos simuladores lógico e elétrico, tornando inviável a adoção de uma linguagem única.

Para testes de lógica funcional estabeleceu-se que a geração de vetores de testes deveria manter formato compatível com os vetores gerados para o simulador lógico SIMUL [Pan87a], que é o simulador lógico atualmente integrado ao SDP. Uma ferramenta de apoio para a definição de vetores de teste para o SIMUL, o EDTES [Pan87b], permite que o usuário defina interativamente e de forma gráfica os vetores. A linguagem utilizada para a definição dos vetores de testes para o SIMUL será discutida no capítulo III. O projeto detalhado e a implementação do sistema de testes funcionais deverão considerar a possibilidade de dar suporte a outros simuladores de lógica digital.

Para as medidas paramétricas e de comportamento analógico a geração de vetores de testes é mais complexa que para o caso de testes de lógica funcional. A utilização de uma linguagem de programação semelhante à utilizada pelo SPICE mostra-se bastante conveniente quando consideramos os seguintes aspectos: a) o SPICE é o simulador elétrico mais difundido nos meios de engenharia; b) embora a linguagem utilizada pelo SPICE não seja tão interessante do ponto de vista sintático, ela apresenta um poder de síntese muito grande na descrição da programação da simulação (ou do teste, no nosso caso); c) o fato das linguagens de descrição da simulação e de programação do experimento serem intercambiáveis faz com que o aluno não tenha que realizar duas programações distintas, uma para as validações de simulação e outra para os testes; d) o uso do mesmo programa para a simulação e para os testes induz o aluno à comparação dos resultados das simulações e das medições.

A descrição de testes paramétricos e de comportamento analógico por meio de uma ferramenta interativa é, sem dúvida, de grande utilidade ao aluno. Para essa descrição, é conveniente que o sistema apresente na tela do computador uma descrição esquemática do circuito e dos instrumentos de medição e controle disponíveis. A partir dessa tela, o aluno deverá "apontar" para os pontos do circuito onde ele deseja aplicar ou monitorar os sinais para, após selecionar cada ponto, descrever o instrumento ao qual o ponto estará associado (por exemplo, um voltímetro ou um gerador de sinais) e as características do sinal a ser aplicado ou medido. Para a aplicação de sinais ao circuito, deverá ser permitido ao aluno definir os estímulos a serem aplicados como um conjunto de valores discretos ao longo do tempo, através de uma equação matemática segundo a qual o sinal deverá variar ou ainda através de uma função interpoladora para valores de estímulos predefinidos em função do tempo.

Embora as linguagens para a programação do experimento tenham sido definidas, neste trabalho, com base nas linguagens dos simuladores, notou-se a necessidade delas incorporarem algumas características adicionais, principalmente voltadas ao controle de execução do experimento e preparação dos dados para análise. Com isso torna-se necessária a conversão de programas dos simuladores para programas de testes e vice-versa. Essas conversões deverão ser realizadas pelo SAD de forma automática e transparente ao usuário.

Descreveremos no capítulo III uma linguagem para a programação de experimentos voltados à realização de testes paramétricos e de comportamento analógico.

A figura 3 apresenta um diagrama esquemático dos principais processos envolvidos na programação do experimento. A divisão dos processos apresentada nessa figura é válida tanto para a programação de testes funcionais quanto para a programação de medidas analógicas e paramétricas.

B) Execução do Experimento

A etapa de execução do experimento é aquela na qual o sistema de *software*, através do controle adequado dos instrumentos de medição e controle, permitirá a aplicação dos estímulos aos circuitos e a realização das medições desejadas.

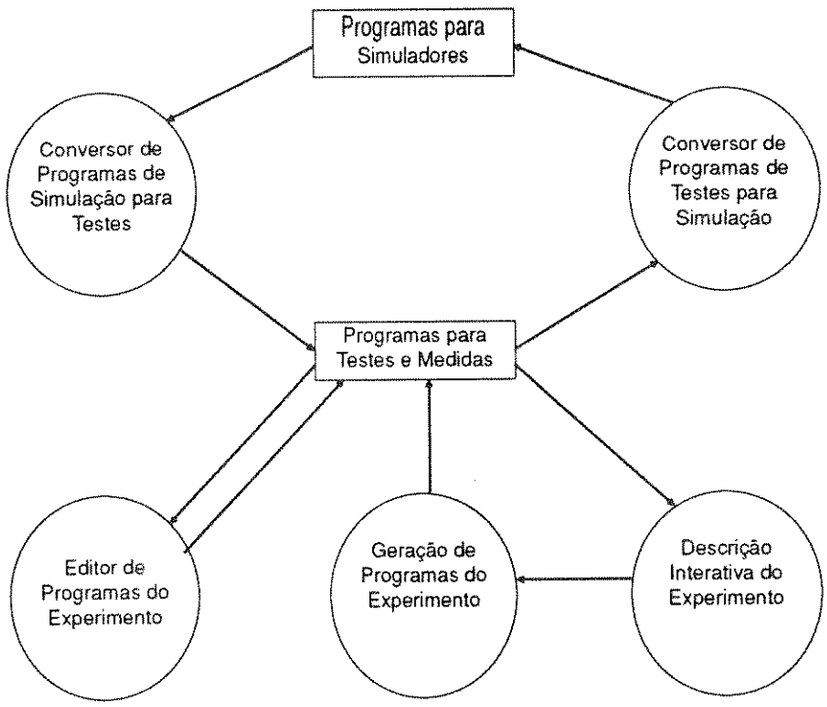


Figura 3 - Processos na Programação do Experimento

A aplicação dos estímulos externos ao circuito e a aquisição dos dados das medições efetuadas serão realizados de acordo com o programa do experimento previamente definido pelo usuário.

Os resultados medidos deverão ser armazenados apropriadamente, em memória ou em disco, de acordo com o tipo de testes em execução ou conforme for definido pelo usuário.

Um dos objetivos precípuos na definição da etapa de execução do experimento foi a obtenção de um alto grau de interatividade do usuário com o processo de execução dos testes. Para tal, o sistema deverá prover facilidades voltadas ao controle da execução, que podem ser ativadas pelo programa do experimento ou interativamente, durante a execução do experimento.

O aluno poderá realizar a monitoração de sinais aplicados ou medidos em tempo de execução, como se estivesse observando os próprios instrumentos de medição ou controle, ou poderá selecionar a visualização somente após o término dos testes, quando serão traçadas curvas de comportamento dos sinais.

Na programação do experimento, deve ser permitida a definição da aplicação de estímulos de inicialização ao circuito, que deverão definir um conjunto de estímulos de maneira análoga aos estímulos de testes, mas que visam garantir que o circuito esteja numa condição inicial adequada quando do início dos testes. Para a inicialização do circuito o aluno poderá definir, além dos estímulos a serem aplicados, condições de parada.

Será permitido ao operador interromper temporariamente ou cancelar a execução de um experimento antes de seu término, através de comando específico no programa do experimento ou através do teclado do computador, sem que os dados coletados até esse instante sejam perdidos. Ao interromper a execução dos testes, será possível ao aluno redefinir as grandezas físicas e elétricas a serem monitoradas, ou mesmo redefinir sinais a serem aplicados ao circuito. No primeiro caso, a execução dos testes é reassumida a partir da condição existente quando o experimento foi interrompido. No segundo caso, a execução é retomada desde a etapa de inicialização do circuito.

A interação do aluno com o *software* para a redefinição do experimento deverá ocorrer de forma semelhante àquela da programação do experimento, atuando como se o sistema combinasse os modos de programação e execução do experimento.

Será possível também a execução passo a passo de um experimento. Nesse caso, o aluno poderá definir os intervalos para a interrupção da execução, em função do tempo ou de valores de um sinal. Deve-se observar que neste modo de execução a base de tempo não é fixa e o aluno deverá atentar aos aspectos funcionais do circuito antes de selecioná-lo.

Embora os recursos que permitem que os experimentos sejam interrompidos possam ser úteis em muitos casos de teste, há casos onde a interrupção do teste não é conveniente. Para esses casos, o aluno poderá definir que os testes são ininterruptíveis. O sistema assume como *default* que um teste pode ser interrompido.

Na figura 4 é apresentado o diagrama dos processos relacionados à etapa de execução do experimento, para testes funcionais e medidas analógicas.

C) Tratamento e Análise de Resultados

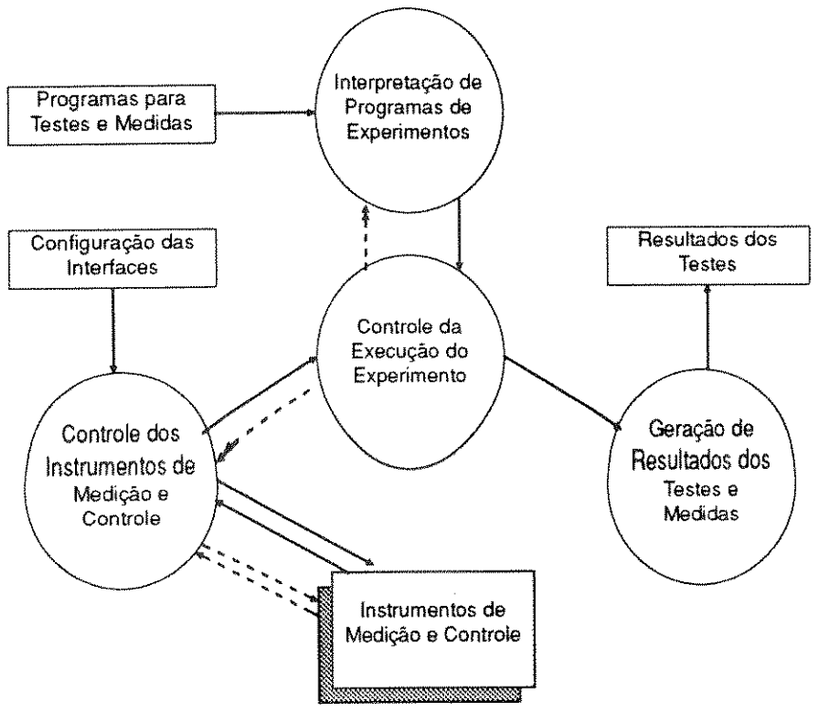


Figura 4 - Principais Processos na Execução do Experimento

O tratamento e análise de resultados é, do ponto de vista do aluno, a ferramenta mais importante no processo de validação dos circuitos em estudo. Estará disponível ao aluno um conjunto de recursos voltados ao pós-processamento dos dados de resultados de medições e simulações, visando principalmente ao enfoque analítico e incluindo análises comparativas entre os resultados de testes e simulações.

Devemos considerar, neste ponto, que os tipos de análises de interesse por parte do aluno com relação aos testes de lógica funcional serão bastante diferentes daquelas voltadas aos dados analógicos (paramétricos e de comportamento analógico). Discutiremos a seguir as análises voltadas a cada um desses conjuntos de dados.

C.1) Tratamento e Análise de Dados de Lógica Funcional

Os tratamentos e análises voltados à validação da lógica funcional prevêem, no SAD, a realização do traçado dos diagramas de estados lógicos, a partir dos dados obtidos das medições e simulações da lógica funcional. A análise comparativa dos resultados poderá ser realizada através da superposição dos dados das medições e das simulações.

Para os testes de lógica funcional, a simples visualização gráfica dos resultados, permitindo que sejam selecionados os sinais a serem visualizados e os intervalos de tempo de interesse, e as comparações com os resultados das simulações mostram-se suficientes ao aluno, que a partir daí poderá realizar as análises voltadas à depuração do circuito.

O traçado dos diagramas de estados lógicos poderá ser realizado a partir de resultados das medições efetuadas, a partir de dados gerados pelo SIMUL ou de dados externos ao sistema, desde que estes possuam um formato conhecido, conforme especificado no anexo I.

Além da visualização dos resultados na tela, deverão estar disponíveis opções específicas para saída em outros periféricos, como impressora e traçador gráfico (*plotter*), através de *drivers* adequados.

Versões futuras do SAD poderão oferecer ao aluno funções de crítica e aconselhamento na depuração do circuito. Por exemplo, a partir dos resultados das medições efetuadas e conhecendo-se os sinais aplicados e a descrição lógica do circuito, pode-se indicar automaticamente ao usuário as porções do sinal de saída que não apresentaram o comportamento esperado e, mais que isso, indicar no próprio diagrama esquemático do circuito os componentes que são suspeitos de mau funcionamento, sugerindo ainda conjuntos de vetores de testes que permitiriam ao aluno isolar o problema detectado. Para tal, o SAD deveria implementar um sistema especialista que aplicasse diferentes métodos de validação, como o da diferença booleana [Sel68], o algoritmo *D* [Rot67] e suas extensões [Bre80], [Lev82], diagramas de decisão binária [Ake78], [Ake80], entre outros.

A figura 5 mostra os principais módulos funcionais relacionados ao tratamento e análise de resultados de testes e simulações funcionais.

C.2) Tratamento e Análise de Dados Analógicos

O tratamento e a análise dos resultados de testes e simulações paramétricos e de comportamento analógico deverão implementar um conjunto de funções que, atuando de forma integrada, permitam a visualização gráfica dos dados e ofereçam ao aluno um ambiente no qual ele possa realizar as seguintes operações:

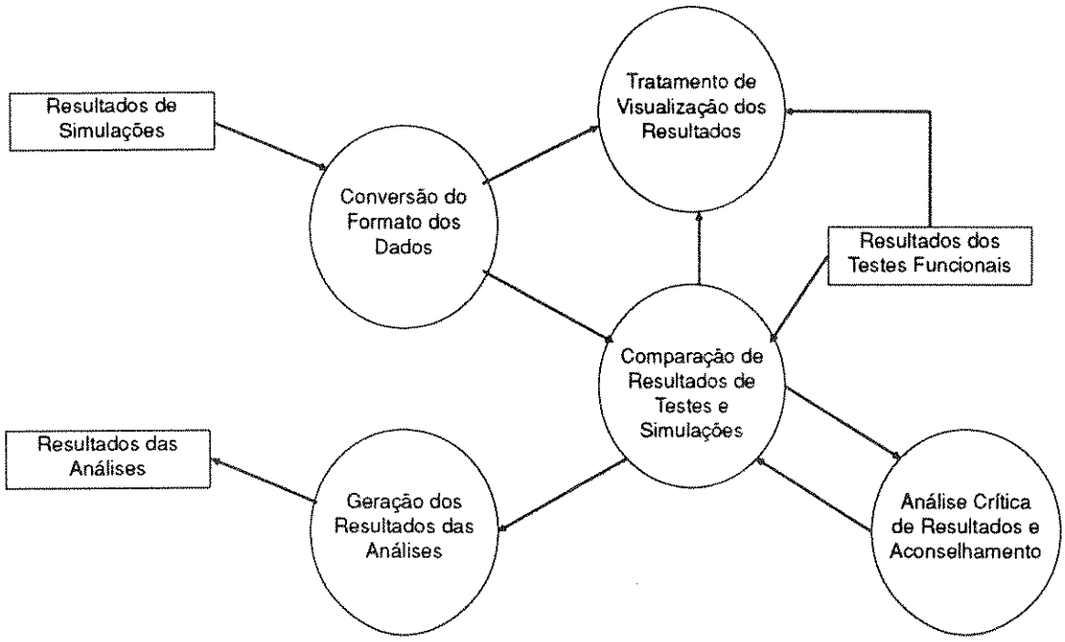


Figura 5 - Processos para Análise de Testes Funcionais

- Traçado de Curvas
- Cálculos de Regressões
- Cálculos de Transformadas de Fourier
- Cálculos e Análises Estatísticas
- Extração de Parâmetros de Dispositivos
- Extração de Dados para Otimização

Uma importante característica considerada para esta etapa é a capacidade de manipulação dos dados e dos tratamentos de visualização. Para tal, a interface com o usuário deve ser implementada na forma de uma prancheta eletrônica, onde o aluno possa atuar como se estivesse utilizando sua prancheta de projeto, tendo como vantagem a disponibilidade de um conjunto de ferramentas de alto nível atuando de maneira integrada, propiciando considerável redução dos esforços dedicados a essa atividade.

Como elemento básico à prancheta eletrônica, o aluno terá um cursor que permitirá que ele "caminhe" pelos gráficos e execute operações como a visualização panorâmica ou de detalhes das curvas, medição de distância entre pontos do gráfico e cálculo da inclinação da reta que passa por eles, translação de eixos, seleção de escalas lineares ou logarítmicas, saídas em impressora ou *plotter*, entre outras.

A fim de não restringir o uso deste tipo de ferramenta ao tratamento de dados obtidos através do próprio sistema, foi estabelecido que deverão ser fornecidas interfaces de *software* que permitam a importação de dados de arquivos em formatos específicos. Serão aceitos como dados de entrada os arquivos gerados como saída do SPICE, arquivos de resultados de testes gerados pelo próprio SAD ou ainda arquivos externos ao sistema. Neste último caso, os arquivos deverão estar em formato ASCII, obedecendo a formatação específica, descrita no anexo I.

Para quaisquer dos tipos de análises realizadas, o aluno poderá visualizar os resultados em tela, em forma gráfica ou textual. Deverá ser possível também que esses resultados sejam armazenados em um arquivo específico, de anotações do projeto.

A operação de **traçado de curvas** permitirá ao aluno a visualização gráfica de curvas paramétricas relacionando tensões, correntes, capacitâncias e impedâncias ou curvas de desempenho analógico nos domínios do tempo e da freqüência, entre outras. Na versão inicial do sistema serão traçadas apenas curvas bidimensionais, permitindo portanto a representação de funções de uma variável. Funções com mais que uma variável poderão também ser representadas, sob a forma de famílias de curvas, através da parametrização das demais variáveis.

Um dos recursos oferecidos com vistas à análise é a superposição de diversas curvas ou famílias de curvas num mesmo gráfico, permitindo a comparação visual das mesmas. Deverá ser possível a superposição de no mínimo 10 curvas simultaneamente.

As operações de **regressões** permitem que se estime o valor de uma variável (chamada de variável dependente) a partir de uma ou mais variáveis correlatas (chamadas de variáveis independentes). O aluno poderá realizar a aplicação de regressões simples (envolvendo apenas duas variáveis), devendo para tal selecionar uma determinada curva do gráfico e então escolher o tipo de regressão a ser aplicada. Estarão disponíveis a ele regressões lineares, polinomiais, geométricas ou exponenciais. A aplicação de regressões é de grande utilidade no auxílio à determinação de parâmetros físicos e elétricos dos dispositivos.

As **transformadas rápidas de Fourier** (FFTs) são aplicáveis a sinais analógicos discretizados e permitem que, a partir de uma seqüência de amostras (medições) associadas a um sinal analógico, este sinal seja decomposto em função de um conjunto de sinais senoidais com freqüência e fase definidos e tais que a somatória destes seja equivalente ao sinal analógico tomado inicialmente. Os resultados do cálculo da FFT deverão ser mostrados graficamente na tela ou em forma textual.

As operações de **análises estatísticas** permitirão ao aluno realizar cálculos estatísticos acerca de um determinado conjunto de dados por ele selecionado. Numa versão inicial estarão disponíveis cálculos estatísticos básicos, como moda, média, mediana, desvio médio, desvio padrão, variância e coeficiente de variabilidade. Será possível também a visualização de gráficos de distribuição de freqüências. Esse conjunto de funções foi considerado como suficiente para a primeira versão do sistema. O uso do sistema poderá determinar o emprego de outros tipos de análises estatísticas voltadas, por exemplo, a determinações de qualidade de dispositivos.

A **extração de parâmetros** de dispositivos é um dos recursos mais importantes ao aluno, especialmente no âmbito da microeletrônica, onde muitas vezes se deseja determinar ou verificar parâmetros físicos e elétricos que não são mensuráveis diretamente. Através da extração de parâmetros dos dispositivos, associada ao tratamento estatístico, o aluno poderá realizar a verificação de viabilidade de modelos, de projetos de pior caso e aferir a qualidade de componentes e de processos de fabricação dos mesmos. Para a determinação de um parâmetro do dispositivo o aluno deverá selecionar o conjunto de dados de interesse (uma curva ou um segmento dela) e, aplicando as funções de tratamento e os cálculos adequados, determinar o valor do parâmetro. Será possível ao aluno armazenar os valores de parâmetros no arquivo de anotações do projeto. Em versões futuras do sistema, poderão estar disponíveis bilbiotecas de dispositivos que incorporem os modelos matemáticos associados à determinação dos parâmetros, reduzindo a quantidade de operações que o aluno precisa realizar para a determinação de um parâmetro.

A operação **Dados para Otimização** permitirá que o aluno, através da análise comparativa entre amostras de dados obtidas para diferentes valores de um parâmetro, determine o valor ótimo do mesmo.

As operações de análise e tratamento de dados paramétricos e de comportamento analógico dos dispositivos buscam, na versão inicial do sistema, estabelecer um conjunto mínimo de ferramentas para o estudo dos circuitos e dispositivos. Versões futuras poderão implementar conjuntos adicionais de facilidades, suportando um maior número de modelos estatísticos e matemáticos, aumentando o grau de automatização dos processos de extração de parâmetros de dispositivos e de dados para otimização e ampliando os recursos de tratamento de visualização, permitindo, por exemplo, o traçado de gráficos tridimensionais.

A figura 6 apresenta um diagrama dos processos para a análise e tratamento de resultados de medidas analógicas.

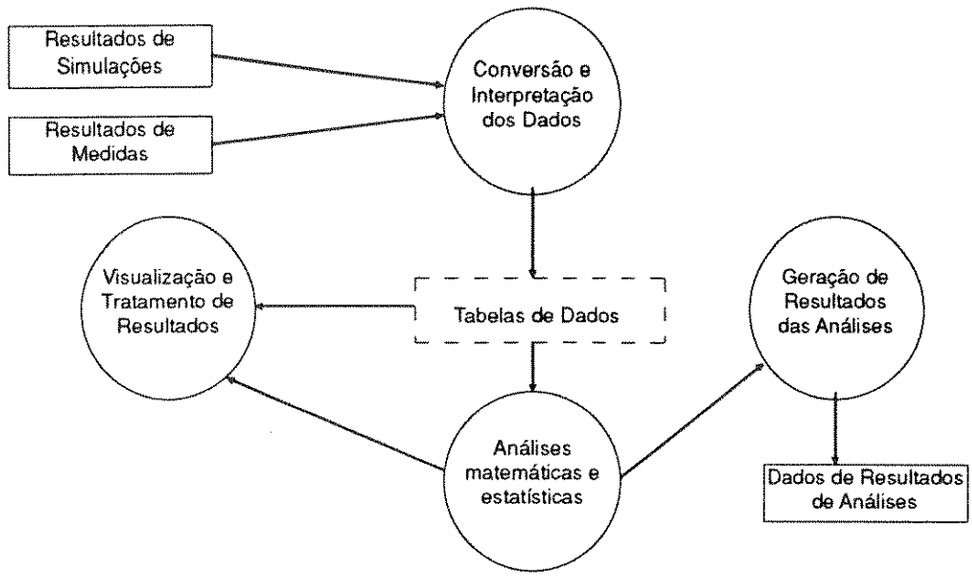


Figura 6 - Processos para Análise de Medidas Analógicas

II.4.3.1.2 - Bases de Dados para o SAD

As bases de dados utilizadas pelo SAD serão geradas através de ferramentas do SDP, através do próprio SAD e por dados externos a esses sistemas. Do ponto de vista funcional, podemos classificar as bases de dados do SAD nos seguintes grupos:

- Dados de Descrição dos Circuitos
- Dados de Resultados de Simulações
- Dados de Medidas Efetuadas
- Dados de Programação de Simulações
- Dados de Programação do Experimento
- Dados de Configuração do Sistema
- Dados de Resultados de Tratamentos Matemáticos e Análises

As bases de dados acima descritas devem ser, na medida do possível, passíveis de tratamentos independentes umas das outras. Além disso devem estar estruturadas de forma que se possa definir interrelacionamentos entre elas, a fim de obtermos melhor integração das operações do sistema e das atividades do aluno.

Descreveremos, a seguir, cada uma das classes de dados acima citadas. No anexo I são apresentadas as informações sobre os formatos das bases de dados utilizadas pelo SAD.

A) Dados de Descrição dos Circuitos

A descrição dos circuitos é realizada com diferentes objetivos e níveis de detalhamento ao longo do projeto, utilizando-se para tal diferentes tipos de representações. Por exemplo, para a validação da lógica de circuitos digitais é conveniente que o circuito seja descrito em termos de portas lógicas; para validação do comportamento elétrico, a descrição é feita em termos de componentes eletrônicos elementares, como transistores, resistores e capacitores. Para caracterização de projetos de circuitos integrados, a descrição dos componentes pode ser feita a nível de dimensões e parâmetros físicos (modelos de dispositivos).

A descrição do circuito a nível de portas lógicas é utilizada pelo simulador lógico (SIMUL), definindo os componentes lógicos do circuito e a rede de interconexões dos mesmos. O mesmo esquema de representação deverá ser utilizado pelo módulo de testes funcionais, permitindo ao aluno visualizar o esquema lógico do circuito durante a definição dos testes a serem executados. A descrição do circuito também será útil em etapas posteriores, quando da aplicação de métodos voltados à localização de falhas no mesmo. A fim de manter compatibilidade entre as informações disponíveis ao simulador lógico e aos módulos do SAD relacionados a testes funcionais, o formato dos dados de descrição lógica do circuito deverá ser idêntico ao utilizado pelo programa de entrada esquemática - PESQA [Pan87c].

A descrição a nível de componentes eletrônicos discretos é utilizada para a simulação elétrica. O SPICE permite que a descrição dos componentes dos circuitos seja realizada também a nível de parâmetros físicos e elétricos (através do comando *".MODEL"*). A descrição a nível dos componentes discretos do circuito será de interesse ao SAD para a programação de experimentos de testes paramétricos e de desempenho analógico, permitindo que o aluno visualize o esquema elétrico do circuito e "aponte" os pontos do circuito onde deverão ser realizadas as medições. Por outro lado, o SAD poderá também gerar informações a respeito de parâmetros dos circuitos obtidos através de medições e análises matemáticas, realimentando assim o processo de validação do circuito. Na descrição elétrica do circuito no SPICE, o *net-list* não contém nenhuma informação posicional sobre os elementos do circuito, sendo necessária, portanto, a criação de uma extensão dessa descrição a fim de que se possam desenhar os diagramas elétricos.

As listas de interconexões (*"net-lists"*) dos circuitos poderão também ser utilizadas pelo aluno como auxílio durante a montagem do protótipo a ser testado.

B) Dados de Resultados de Simulações

Os dados de resultados de simulações serão utilizados pelo SAD durante o processo de tratamento e análise dos resultados de testes, visando à verificação da conformidade ou de diferenças de resultados entre as simulações e os testes.

As informações geradas através dos diferentes tipos de simulações dizem respeito a aspectos distintos das características dos circuitos em estudo. Por conseqüência, o conteúdo dos dados de resultados das simulações elétricas e lógicas serão também distintos, bem como os formatos dos arquivos gerados como resultados de simulações pelo SPICE e pelo SIMUL.

Os resultados de simulações da lógica funcional descreverão apenas os níveis lógicos de sinais aplicados às entradas e verificados (calculados) às saídas do circuito ao longo do tempo. O SAD será capaz de interpretar arquivos de resultados de simulações gerados pelo SIMUL ou ainda arquivos em formato ASCII padrão, conforme formato especificado no anexo I.

Os resultados das simulações elétricas compreenderão informações a respeito do comportamento de pontos definidos do circuito nos domínios do tempo e/ou da freqüência. O SAD será capaz de interpretar arquivos de dados no formato de saída do SPICE ou tabelas em formato ASCII, conforme padrão definido no anexo I. Para os arquivos de saída do SPICE serão interpretadas uma ou mais tabelas de dados geradas pelo comando *".PRINT"* do simulador.

A capacidade de interpretação de arquivos em formato ASCII permite ao SAD utilizar dados gerados por ferramentas externas ao sistema, como, por exemplo, planilhas eletrônicas.

C) Dados de Medidas Efetuadas

Os dados de medidas efetuadas serão gerados pelo SAD como saída do processo de execução do experimento e serão utilizados como entrada para o tratamento e análise de resultados dos testes.

A exemplo do que ocorre com os resultados das simulações elétricas e lógicas, os resultados de testes paramétricos e de comportamento analógico diferem, a nível de conteúdo, dos resultados de testes funcionais.

Os conteúdos das bases de dados de medidas efetuadas e de resultados das simulações serão bastante semelhantes, tanto para dados funcionais do circuito quanto para os dados paramétricos e de desempenho analógico. Para resultados de testes funcionais, o formato dos arquivos gerados pelo SAD deverá ser o mesmo dos arquivos gerados para os resultados de simulações pelo SIMUL. Para resultados de testes paramétricos e de desempenho analógico, o arquivo de resultados dos testes será do tipo texto (ASCII), gerado na forma de tabelas com os valores das variáveis medidas. Embora a utilização de arquivos em formato ASCII possa não ser a mais eficiente em termos de espaço utilizado para o armazenamento, ela se mostra bastante adequada quando vista sob a ótica da compatibilidade e intercambiabilidade de dados entre os diferentes módulos do sistema.

D) Dados de Programação de Simulações

Os dados de programação das simulações descrevem basicamente uma seqüência de casos de teste a serem simulados para o circuito em estudo. Na descrição dos casos de teste incluem-se os estímulos a serem aplicados ao circuito e as saídas a serem monitoradas. Dependendo do tipo de simulação a ser realizada, essa programação pode definir também outros parâmetros incluindo, por exemplo, condições ambientais a serem simuladas.

No mesmo arquivo deverá estar contida também, no nível de detalhamento apropriado à simulação, a descrição do circuito (*net-list*), em termos dos componentes básicos e das interconexões entre eles.

A partir dos dados de programação das simulações, o sistema realizará a geração, de forma automática, de programas voltados aos testes do circuito. Deve-se observar, porém, que nem sempre o programa da simulação refletirá todos os aspectos necessários à realização de um teste equivalente. O aluno deverá estar sempre atento a este tipo de detalhe e poderá, através de ferramentas de *software* adequadas, modificar o programa de testes.

De maneira análoga, o sistema deverá prover meios para, a partir de dados de programação do experimento, gerar os programas para os simuladores.

As linguagens utilizadas para a programação das simulações elétricas e lógicas são discutidas respectivamente em [Vla81] e [Pan87a].

E) Dados de Programação do Experimento

Através da programação do experimento, o aluno definirá os casos de teste a serem executados, definindo os estímulos a serem aplicados às entradas e os sinais a serem monitorados nas saídas do circuito. Para testes paramétricos e de desempenho analógico, o aluno poderá definir também alguns parâmetros adicionais, como por exemplo a temperatura em que o teste deve ser realizado.

Como já foi descrito anteriormente, o SAD proverá ferramentas automatizadas para, a partir de programas de simulações, gerar os programas do experimento (programas de testes) e vice-versa.

As linguagens utilizadas para a programação dos testes serão, a exemplo das de programação das simulações, diferenciadas para testes de lógica funcional e testes paramétricos e de desempenho analógico. A definição dessas linguagens será feita no capítulo III.

F) Dados de Configuração do Sistema

Os dados de configuração definem características ambientais e operacionais do sistema e visam basicamente à flexibilidade de uso do mesmo.

Algumas características do sistema serão autoconfiguráveis, como por exemplo o tipo de monitor de vídeo utilizado.

Outras características serão configuráveis pelo usuário, como cores a serem utilizadas em tela; idioma usado para a interface usuário-software; tipos e modelos de periféricos de saída, como impressoras e *plotters*; dispositivos de entrada, como *mouse* e mesa digitalizadora.

O aluno poderá ainda, através de uma opção de configuração de projeto, definir quais os arquivos de dados envolvidos para as simulações, para os testes, para as análises e tratamentos e para as anotações referentes a um mesmo projeto, estabelecendo assim uma forma eficaz de oferecer ao aluno controle sobre os diversos arquivos de dados envolvidos num projeto.

Os dados de configuração do sistema deverão ainda conter informações a respeito dos tipos de interfaces para aquisição de dados utilizadas e do conjunto de instrumentos a elas interligados e suas características.

II.4.3.2 - Requisitos de Interface

II.4.3.2.1 - Interface com o Usuário

Considerando-se o fato do SAD ser um sistema destinado ao uso em um ambiente didático, um dos principais aspectos a serem considerados na definição dos requisitos do sistema é o mecanismo de interação entre o aluno e o *software*.

A interface usuário-software deve ser amigável e capaz de direcionar o aluno durante a utilização do sistema. Ainda, deve ser previsto o uso de mensagens explicativas sempre que necessário e um sistema de auxílio ao usuário (*help*) sensível ao contexto operacional e que permita que informações acerca de outros ítens associados ao contexto atual sejam visualizadas. Devem também ser oferecidos recursos para facilitar a definição de informações acerca do experimento, como a geração de vetores de testes, a programação de instrumentos de medição, entre outros, a fim de minimizar o esforço do aluno em algumas etapas que são, por natureza, tediosas. Não se deve perder de vista, porém, a finalidade didática do sistema, devendo ser considerado que algumas etapas, embora sejam passíveis de automatização, devem ser realizadas manualmente, a fim de que o aluno visualize e participe daquela determinada etapa do processo.

As ferramentas do SAD utilizam basicamente menus de opções como forma de direcionar o uso do sistema pelo aluno. O menu inicial do sistema (ou menu principal) agrupa as funções do sistema em classes de operações e os submenus de cada uma dessas opções dão acesso às operações básicas do sistema. Estabeleceu-se como regra básica que o acesso a qualquer opção do sistema não deveria exigir mais que três níveis de submenus, evitando assim a navegação excessiva através dos menus. Foi estabelecido também que, sempre que conveniente, as operações do sistema deveriam ser acessíveis através de teclas especiais (chamadas de *"hot keys"* ou *"shortcut keys"*), ativando a função diretamente, sem a necessidade da navegação nos menus.

O sistema de auxílio ao usuário (*help*) está disponível ao usuário a qualquer instante, através da tecla <F1> e atua de forma a ser sensível ao contexto. Por exemplo, estando numa determinada sub-opção de um menu e teclando <F1> o usuário obterá a tela de auxílio referente àquela opção do menu. A implementação das telas de auxílio foi projetada usando o conceito de hipertexto, mostrando algumas palavras-chave destacadas nas telas de *help*. Essas palavras em destaque referenciam outros tópicos do sistema de auxílio ao usuário e, quando selecionadas, trazem à tela as informações referentes àquele tópico. Isso permite que o usuário "caminhe" através do sistema de *help* do SAD.

Foi desenvolvida, com a finalidade de dar suporte à criação e manutenção dos hipertextos, uma ferramenta interativa, baseada em [Ges90] e descrita em [Aff91].

A configuração de idioma foi considerada também como um dos pontos importantes do ponto de vista de interação com o sistema, dado que o sistema se destina ao uso didático em países de diferentes idiomas. Embora na versão atual das ferramentas do SAD o idioma suportado seja apenas o Português, o sistema prevê, para o futuro, o suporte a outros idiomas. Outras opções de configuração, como a de cores e de dispositivos de entrada e saída, estarão também disponíveis.

II.4.3.2.2 - Interfaces de *Software*

O SAD deverá prever interfaces de *software* com as diversas ferramentas do SDP, como o SIMUL, o EDTES, o PESQA, geradores de vetor de testes, etc...

Deverá também ser prevista interface com os programas de simulações e dados de saída do SPICE, a fim de permitir as análises de dados gerados por esse simulador.

Com o objetivo de flexibilizar o uso do sistema, deve-se procurar também oferecer suporte a dados gerados por ferramentas externas ao SAD e ao SDP.

II.4.3.2.3 - Interfaces de *Hardware*

As interfaces básicas de *hardware* para o sistema são aquelas especificadas na seção II.2 deste trabalho. Deve-se procurar também oferecer recursos para a utilização de diferentes tipos e modelos de periféricos que são opcionais ao sistema, como impressoras e traçadores gráficos (*plotters*).

Com relação às interfaces de aquisição de dados, deve-se levar em consideração que os requisitos estabelecidos são os mínimos. O projeto do sistema deve se preocupar com a capacidade de manipulação de placas mais sofisticadas, permitindo por exemplo uma melhor resolução de leitura e maior taxa de amostragem, entre outros.

Com relação à forma da leitura dos dados, as interfaces de aquisição de dados se dividem basicamente em dois grupos: as que permitem acesso por endereçamento de entrada e saída (*I/O ports*) e as que usam acesso direto à memória (DMA). Atualmente várias placas permitem os dois métodos para a leitura dos dados. O SAD deverá permitir o uso de qualquer um desses tipos de placa.

II.4.4 - Prioridades de Implementação

Com vistas à implementação do SAD, decidiu-se pela criação de três subsistemas, cada um dos quais deverá ter a capacidade de atuar de maneira independente dos demais. O primeiro deles, o STEF, deverá abranger as três etapas envolvidas nos testes funcionais de circuitos: programação, execução e análise de resultados. O segundo subsistema deverá cobrir as etapas de programação e execução de medidas analógicas e paramétricas. Por fim, o terceiro subsistema será o STAG, que terá como finalidade permitir o tratamento e análise gráfica de resultados de medidas analógicas e paramétricas.

Foi estabelecido como estratégia para a implementação o desenvolvimento paralelo dos três subsistemas acima descritos. Numa etapa inicial, cada um dos subsistemas terá somente as funções consideradas essenciais, visando à obtenção de um protótipo de cada um deles. Na segunda etapa, outras funções adicionais serão acrescentadas. Uma terceira etapa de implementação é ainda prevista, quando deverão ser implementados aprimoramentos da interface usuário-software e, eventualmente, adicionadas algumas funções, tendo como base as críticas e sugestões formuladas pelos usuários do sistema.

A figura 7 mostra um diagrama geral do SAD, onde são destacadas as atividades relacionadas a cada um dos subsistemas.

II.4.5 - Evoluções e Melhorias Previstas

As propostas de evoluções e melhorias descritas nesta seção deverão ser consideradas em todas as etapas seguintes do projeto e da implementação, de forma a evitar futuros problemas e restrições quanto à evolução do SAD.

A evolução do Sistema de Testes Funcionais-STEF deverá compreender a implementação de funções voltadas à comparação crítica dos resultados de testes e simulações, onde deverão ser incluidas funções de aconselhamento ao aluno, indicando a ele, por exemplo, regiões e componentes do circuito que são suspeitas de mau funcionamento e indicando vetores de testes adequados ao isolamento dos erros.

Uma evolução natural, abrangendo a todos os módulos do SAD, deverá ser a implementação de suporte a um conjunto maior de periféricos, tanto no que diz respeito ao tipo de periférico (por exemplo mesas digitalizadoras, caneta ótica, etc...) quanto aos modelos dos mesmos.

Pretende-se futuramente dar suporte a outros tipos de interfaces de aquisição de dados, proporcionando aos usuários uma maior flexibilidade quanto à determinação do desempenho, da precisão das medidas e do conjunto de medidas suportadas pelo SAD.

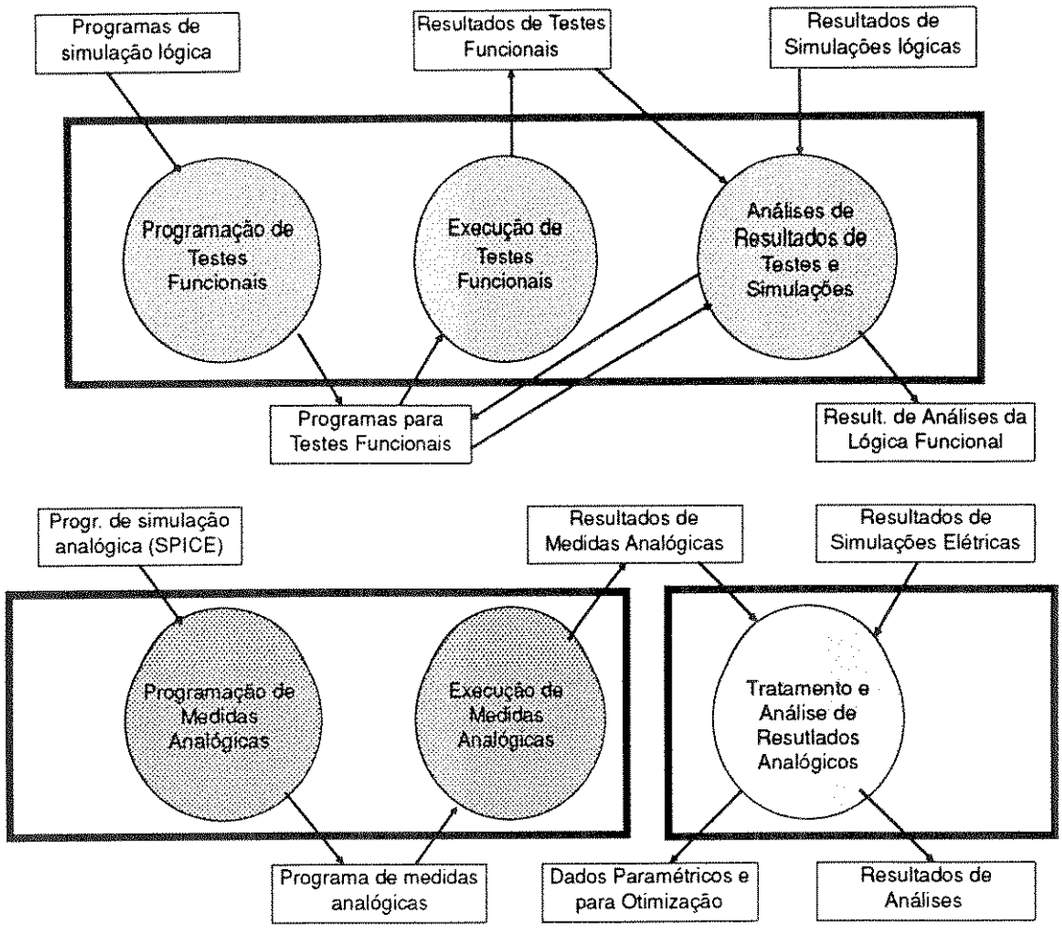


Figura 7 - Módulos Básicos de Implementação do Sistema

Para o STAG, deve-se prever a implementação de um maior número de funções de análises, as quais serão definidas de acordo com o retorno obtido dos alunos a partir de cursos ministrados com o uso do sistema. Outra característica desejável para futuras versões do STAG é a capacidade de traçar gráficos tridimensionais.

É prevista a criação futura de funções voltadas à avaliação de desempenho dos alunos, as quais deverão ser acessíveis somente ao professor.

Encontra-se ainda em estudo a conveniência da criação de uma versão do sistema para o ambiente *Windows*. A criação de uma versão do sistema para esse ambiente poderia, certamente, trazer-nos grandes ganhos em termos da qualidade da interface usuário-*software*,bem como na capacidade de configuração de periféricos.

III - Linguagens de Programação para o Experimento

III.1 - Introdução

A definição de linguagens voltadas à programação dos experimentos a serem realizados através do SAD permite uma melhor organização e sistematização das atividades do aluno.

Com o uso de uma linguagem de programação, o aluno poderá abstrair detalhes de características particulares do *hardware* para aquisição de dados e dos instrumentos de medição e controle.

Como já foi discutido anteriormente, a validação da lógica funcional de um circuito e a validação paramétrica e de desempenho analógico apresentam enfoques e características bastante diferentes. Essas diferenças são reveladas através das próprias linguagens adotadas pelos simuladores.

Outro ponto considerado diz respeito à conveniência da adoção de uma linguagem de programação para os testes que se assemelhe ao máximo à linguagem do simulador, dado que o tipo de informações de entrada para os simuladores e para os sistemas de testes também se assemelham. Ainda, isso propicia ao sistema a possibilidade de intercambiar dados entre programas de simulação e de testes, facilitando, por exemplo, a geração de casos de falhas idênticas para o simulador e para o sistema de testes.

Ainda, para o aluno, a uniformização das linguagens evita que ele tenha que aprender "mais uma linguagem".

Adotou-se para o SAD o uso de duas linguagens distintas de programação do experimento, semelhantes às linguagens dos simuladores SPICE e SIMUL.

Este capítulo apresentará cada uma dessas linguagens. Inicialmente introduziremos alguns conceitos e definições básicos que serão utilizados na descrição das linguagens, passando em seguida à descrição das linguagens de programação de experimentos propriamente ditas.

III.2 - Conceitos e Definições

Discutiremos brevemente nesta seção os principais conceitos e definições usados para o estabelecimento das linguagens de programação. Discussões mais amplas acerca desse assunto podem ser encontradas em [Kow79] e [Aho85].

As linguagens de programação podem ser definidas através de dois aspectos básicos: a sua **sintaxe**, ou seja, *como* seus programas são representados; e a sua **semântica** ou, *o que* os programas descrevem. Um terceiro aspecto refere-se ao problema de integração da linguagem a um determinado ambiente hospedeiro.

Uma **gramática livre de contexto** [Cho59], que ao longo do texto chamaremos simplesmente de gramática, permite a descrição de uma linguagem através de definições indutivas. A classificação de *livre de contexto* define uma classe especial de gramáticas onde as regras de transição para um símbolo terminal não dependem do contexto onde o símbolo se encontra.

A gramática livre de contexto tem quatro componentes básicos:

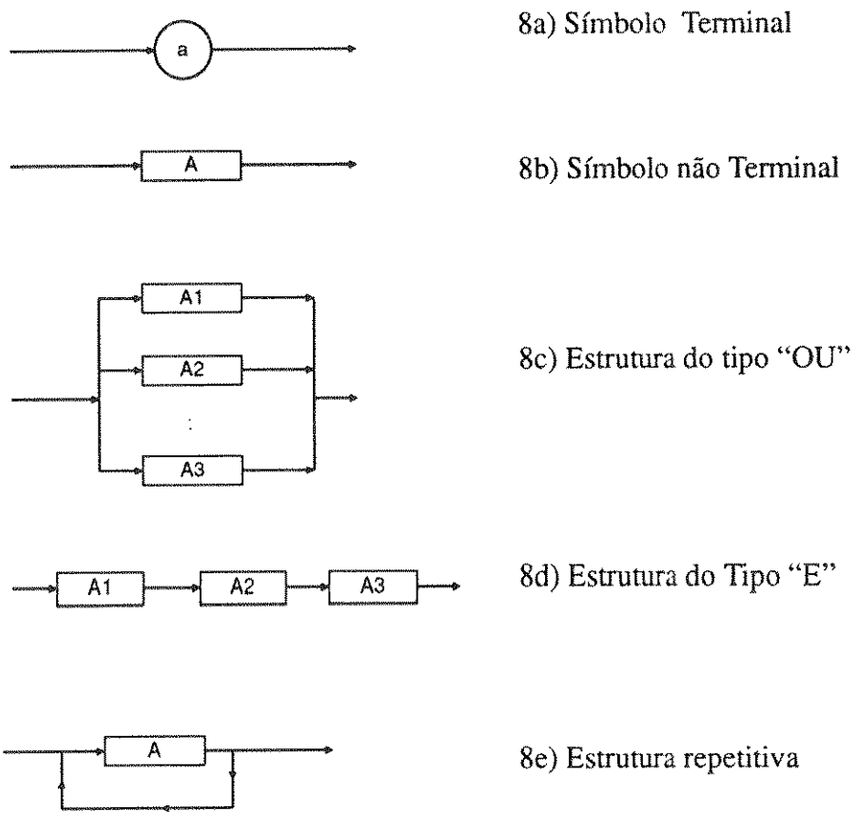
- Um conjunto de **símbolos terminais**, também chamados de *tokens*, que serão tratados como símbolos indivisíveis ou atômicos.
- Um conjunto de **símbolos não terminais**.
- Um conjunto de **produções** (regras de transição), onde cada produção consiste na definição de um símbolo não terminal através de uma sequência de *tokens* e/ou símbolos não terminais.
- A designação de um dos símbolos não terminais, chamada de **símbolo inicial**.

Uma das formas mais comumente utilizadas para a descrição da gramática de linguagens de programação é através das *cartas sintáticas* (também chamadas de *diagramas sintáticos* ou *diagramas de Conway*). Outra notação, também muito difundida é a BNF (*Backus-Naur Form*). Utilizaremos neste texto a representação por cartas sintáticas.

Numa carta sintática, cada símbolo não terminal da gramática é descrito através de um grafo orientado onde cada um de seus nós é rotulado com os símbolos terminais e não terminais da gramática.

Os símbolos utilizados para a representação dos diferentes elementos e estruturas nesses grafos são apresentados na figura 8.

Para a descrição da semântica das linguagens de programação, diversas tentativas de formalização foram realizadas mas trouxeram, na prática, poucas contribuições, devido à grande variação entre as linguagens. Com isso a descrição semântica é feita ainda, na maioria das vezes, de maneira informal.



8a) Símbolo Terminal

8b) Símbolo não Terminal

8c) Estrutura do tipo "OU"

8d) Estrutura do Tipo "E"

8e) Estrutura repetitiva

Figura 8 - Símbolos Utilizados para a Carta Sintática

III.3 - A Linguagem para Descrição dos Testes Funcionais

A linguagem para descrição dos testes funcionais foi feita com bastante aderência à que é utilizada para a descrição de programas de simulações através do SIMUL, acrescentando-se a ela algumas extensões visando a aspectos particulares dos testes. Discutiremos, a seguir, a gramática e a semântica da linguagem proposta.

III.3.1 - Descrição Sintática

Os *tokens* (símbolos terminais) utilizados para os programas de experimentos funcionais são as menores unidades de texto dos programas.

Os programas do experimento serão compostos por *tokens* e separadores, sendo estes últimos classificados em dois grupos: separadores de comandos e separadores de parâmetros. Os separadores de comandos são definidos pelo caracter *return* (ASCII 13, que denotaremos por **<CR>**), enquanto os separadores de parâmetros são definidos por um ou mais caracteres brancos (ASCII 32) ou de tabulação (ASCII 09). Separadores não poderão estar contidos nos *tokens*.

A linguagem utilizará os seguintes subconjuntos do conjunto de caracteres ASCII:

- **Letras:** Os caracteres alfabéticos, maiúsculos ou minúsculos.
- **Dígitos:** Numerais arábicos de 0 a 9.
- **Dígitos-Hexa:** Numerais arábicos de 0 a 9, letras de *A* até *F* e de *a* até *f*.
- **Dígitos-binários:** Numerais arábicos 0 ou 1.
- **Brancos:** O caracter branco (ASCII 32) e o caracter de tabulação (ASCII 09).

A figura 9 apresenta os diagramas sintáticos para cada um desses subconjuntos de caracteres.

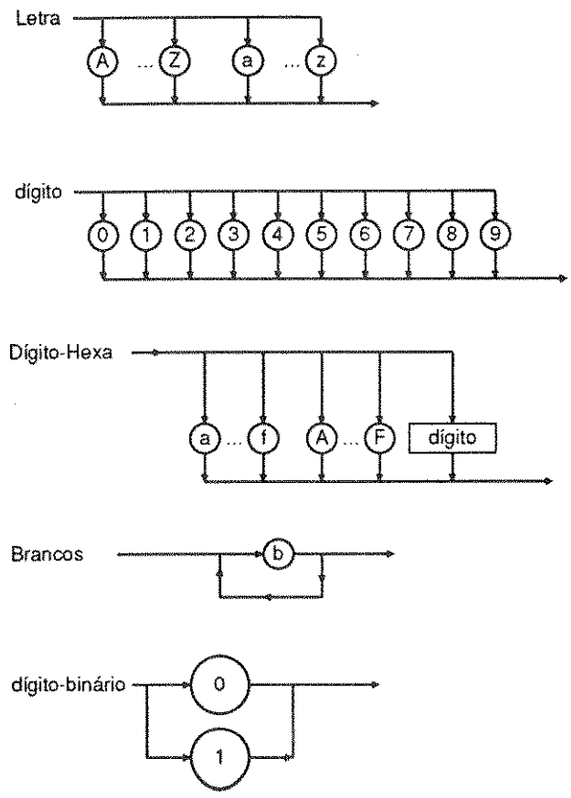


Figura 9 - Diagrama Sintático dos conjuntos de Caracteres

Símbolos especiais e palavras reservadas são os caracteres (ou sequências de caracteres) que têm um ou mais significados fixos. As tabelas 1 e 2 apresentam respectivamente os símbolos especiais e as palavras reservadas.

=
*
.
+
!

Tabela 1
Símbolos Especiais

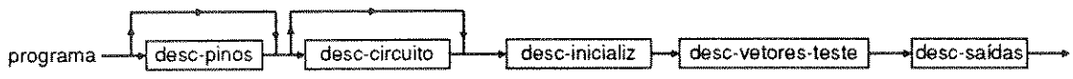
.ARQ	.COND	.INCR
.MOSTR	.PASSO	.PAUSA
.TFIM	.TINIC	AND
DFF	ELE	EST
FIM	JKFF	INI
NAND	NOR	NOT
OR	PIN	SAI
XNOR	XOR	

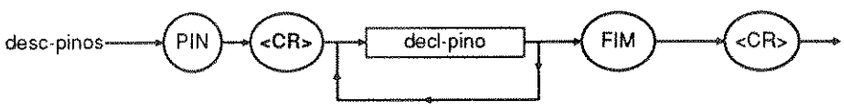
Tabela 2
Palavras Reservadas

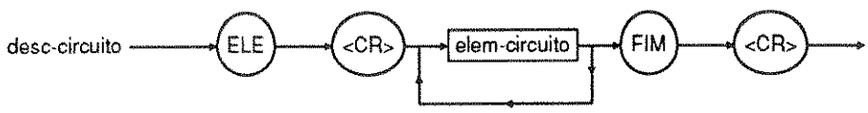
Segue abaixo a descrição da carta sintática que define a linguagem para experimentos de validação da lógica funcional.

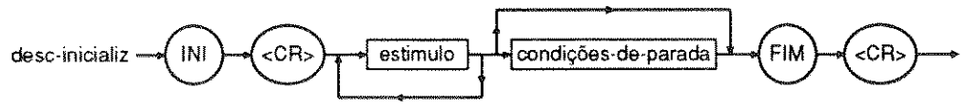
III.3.1.1 - Programa e Blocos de Programa

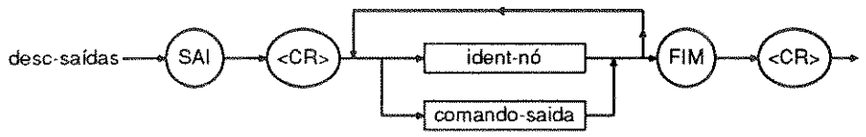
Um programa para a descrição dos testes funcionais é descrito como uma sequência de blocos, cada um deles voltado à definição de diferentes características ligadas ao experimento.



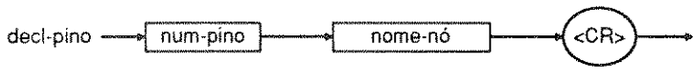


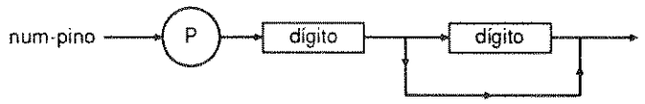


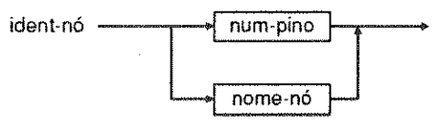




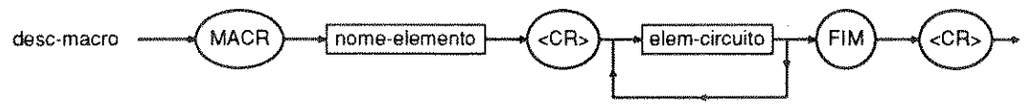
III.3.1.2 - Descrição dos Pinos de Entrada e Saída

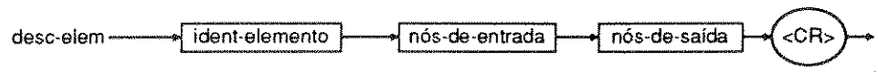


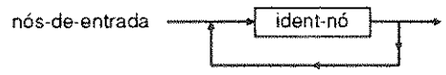


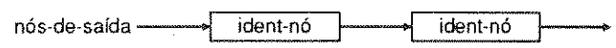


III.3.1.3 - Descrição do *Net-List* do Circuito

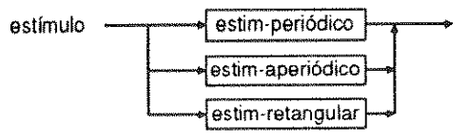


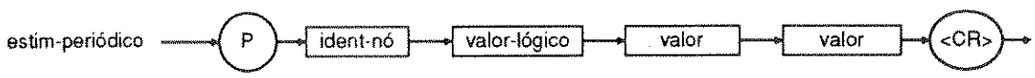


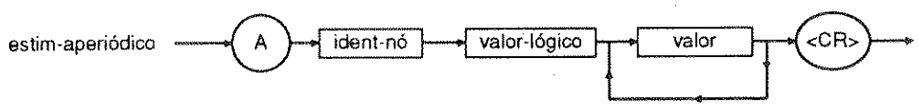


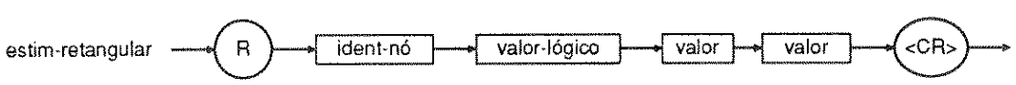


III.3.1.4 - Estímulos de Inicialização e Vetores de Teste

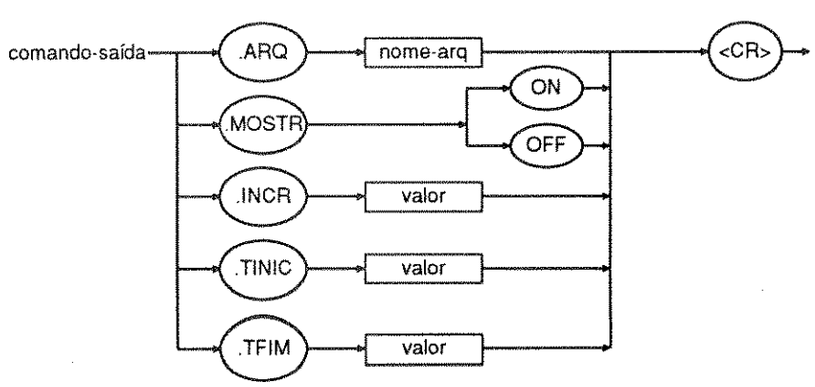




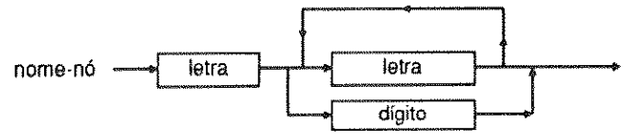


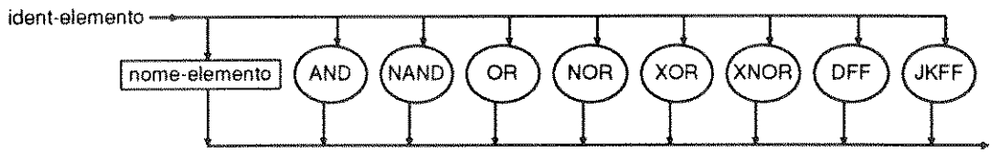


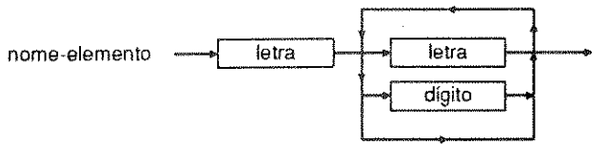
III.3.1.5 - Comandos de Definição dos Dados de Saída

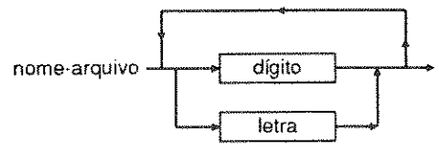


III.3.1.6 - Nomes dos Nós, Elementos do Circuito e de Arquivos

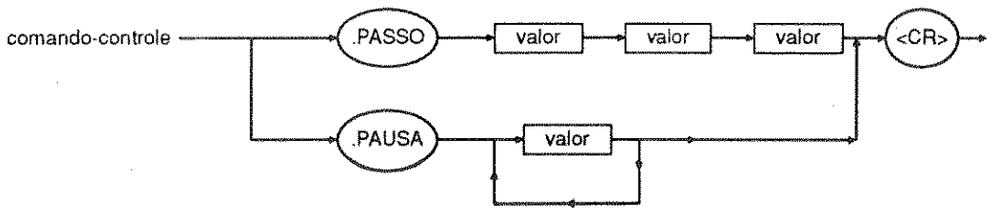


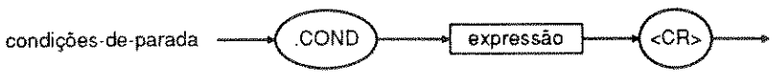




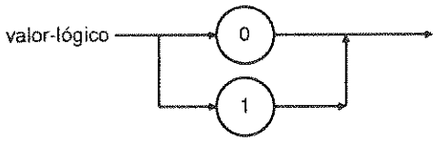


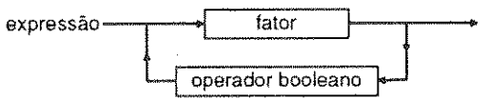
III.3.1.7 - Comandos de Controle da Execução e Condição de Parada

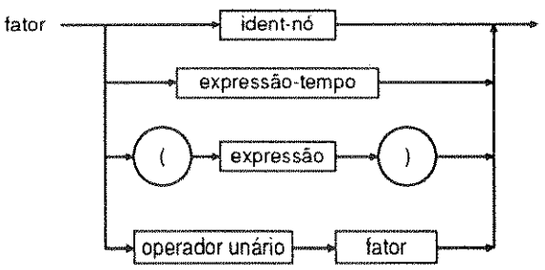


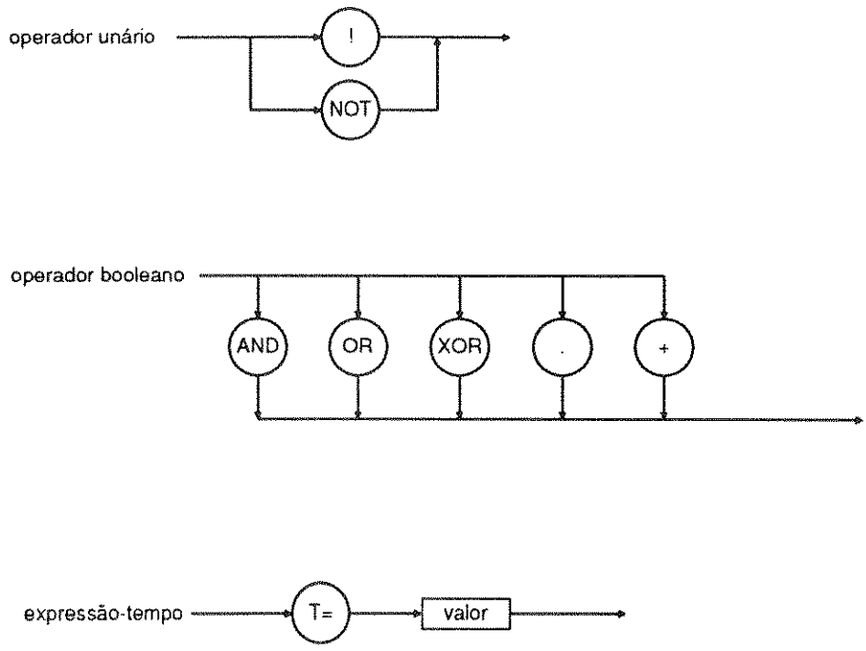


III.3.1.8 - Expressões, Valores e Números









III.3.2 - Descrição Semântica

Descreveremos nesta seção o aspecto semântico da linguagem para programação do experimento. Faremos também algumas análises comparativas entre a linguagem utilizada para o simulador SIMUL e aquela proposta neste trabalho.

III.3.2.1 - Programas e Blocos

Podemos classificar as informações contidas num programa para testes funcionais em quatro grupos distintos:

- Descrição do Circuito
- Descrição dos Estímulos a serem Aplicados ao Circuito
- Descrição das Saídas a serem Monitoradas
- Comandos de Controle da Execução

Procuramos definir a linguagem para os testes funcionais de forma que fosse mantida compatibilidade com os programas para o SIMUL e, ainda, de forma que os programas mantivessem isoladas, em diferentes blocos (ou seções), as diferentes classes de informações acima descritas.

Um programa para testes funcionais é composto por cinco seções básicas (ou blocos de programa), a saber: descrição dos pinos de entrada e saída, descrição do circuito sob testes, descrição dos estímulos de inicialização do circuito, descrição dos vetores de teste e descrição dos sinais a serem monitorados.

Os blocos de descrição dos pinos de entrada e saída e de descrição do circuito conterão as descrições, a nível lógico, do circuito sob testes. Nos blocos de descrição dos estímulos de inicialização e descrição dos vetores de teste são definidos todos os estímulos a serem aplicados ao circuito, além de comandos opcionais para o controle da execução (que estão diretamente relacionados aos estímulos). No bloco de descrição dos sinais de saída são definidos os sinais a serem monitorados e, opcionalmente, comandos adicionais, que definem alguns parâmetros para a monitoração das saídas.

O bloco de descrição dos pinos de entrada e saída permite a declaração de nomes simbólicos associados a cada um dos pinos de entrada e saída da interface de testes. Através destas declarações o aluno poderá referenciar esses nomes simbólicos ao invés dos números dos pinos durante o experimento.

O bloco de descrição do circuito permite que seja definido, a nível de lógica digital, o circuito sob testes. Além da lista de conexões do circuito (*net-list*) poderá também estar descrito neste bloco do programa o diagrama lógico esquemático do circuito. O diagrama esquemático do circuito será apresentado ao aluno durante a execução do experimento, permitindo que, através de uma interface usuário-software apropriada, o aluno redefina, de maneira interativa, o programa do experimento.

No bloco de descrição dos estímulos de inicialização estará descrita uma sequência de estímulos a serem aplicados ao circuito e as condições lógicas (conjunto de estados lógicos no circuito) que definem o término da aplicação dos sinais de inicialização (condições de parada).

O bloco de descrição dos estímulos para testes deverá conter as definições dos vetores de teste a serem aplicados ao circuito, podendo opcionalmente conter também alguns comandos voltados ao controle da execução dos testes. Poderá estar definido também um comando estabelecendo as condições de parada dos testes.

O bloco de descrição dos sinais de saída conterá a descrição dos pontos do circuito a serem monitorados, podendo também conter alguns comandos adicionais, que definem características como instantes inicial e final para a monitoração, se os dados medidos devem ser mostrados em tempo de execução ou somente ao final dos testes, etc...

Descreveremos a seguir cada um dos grupos de informações introduzidos no início desta seção.

III.3.2.2 - Descrição do Circuito

A descrição do circuito será realizada por meio de dois blocos do programa de testes funcionais: o bloco de descrição dos pinos de entrada e saída e o bloco de descrição do circuito propriamente dito.

III.3.2.2.1 - Descrição dos Pinos de Entrada e Saída

A interface para aquisição de dados de lógica funcional estará, como descrevemos anteriormente, interligada a um conector universal, ao qual o aluno deverá interligar os pontos do circuito aos quais serão aplicados estímulos (entradas), bem como as saídas a serem monitoradas. Os pinos desse conector universal são numerados sequencialmente. Durante a descrição do programa do experimento, bem como durante a execução do mesmo, é conveniente que o aluno possa referenciar os nomes lógicos dos sinais ou nós do circuito em estudo.

O objetivo do bloco de descrição dos pinos de entrada e saída é permitir a declaração de nomes lógicos associados a cada um dos pinos de entrada e saída do equipamento de aquisição de dados.

A descrição dos pinos de entrada e saída é opcional num programa de testes funcionais e, em programas onde não houver esta descrição, os canais de entrada e saída do equipamento de aquisição de dados poderão ser referenciados somente pelos números dos pinos do conector (por exemplo P01, P05 e P10, para os pinos de número 1, 5 e 10 respectivamente). Devemos ressaltar que este bloco do programa é estritamente declarativo.

Nos programas do SIMUL este bloco do programa não é definido, já que os estímulos e respostas são simulados.

III.3.2.2.2 - Descrição do *Net-List* do Circuito

Para os programas do experimento, o bloco de descrição do circuito será opcional, dado que o circuito está fisicamente montado. Apesar da opcionalidade dessa descrição, ela poderá ser de grande utilidade ao aluno, considerando-se que com ela o aluno terá acesso a um conjunto de facilidades adicionais durante os testes.

Se for definida, a descrição do circuito permitirá ao aluno visualizar o diagrama lógico do circuito e selecionar pontos a serem monitorados ou estimulados, permitindo assim a programação de forma interativa. A descrição de macros (subcircuitos) será tratada como uma descrição hierárquica do circuito, permitindo ao aluno a visualização em diversos níveis de detalhamento.

A descrição lógica do circuito será também de grande utilidade para funções a serem implementadas em versões futuras do STEF, que implementarão recursos voltados ao aconselhamento do aluno e análise de falhas.

A descrição do circuito é sintaticamente semelhante àquela utilizada para os programas do SIMUL, acrescentando-se somente informações posicionais sobre cada elemento do circuito, voltadas ao traçado do diagrama lógico. A nível da descrição lógica do circuito, manteve-se a compatibilidade entre as duas ferramentas.

III.3.2.3 - Descrição de Estímulos a serem Aplicados

Os estímulos a serem aplicados ao circuito serão descritos em dois blocos do programa de testes funcionais: o bloco de estímulos de inicialização e o bloco de descrição dos vetores de testes.

Em programas para o SIMUL não há a definição de estímulos de inicialização do circuito. Isso ocorre porque, nas simulações, pode-se definir as condições iniciais do circuito.

Para testes de circuitos digitais, essas condições iniciais são obtidas através da aplicação de um conjunto de estímulos às entradas do circuito e da verificação de suas saídas, até que seja atingida a condição desejada.

A descrição de cada estímulo, tanto no que se refere à inicialização do circuito, quanto no que diz respeito aos vetores de teste propriamente ditos, mantém sintaxe idêntica ao SIMUL e, em termos semânticos a diferença é que, ao invés dos estímulos serem simulados, eles serão efetivamente aplicados ao circuito sob testes. Na definição dos estímulos são permitidos três tipos de sinais: periódicos, aperiódicos e retangulares. Descrevemos a seguir cada um deles.

a) Estímulos periódicos

P *ident_nó valinic Tinic increm*

onde:

ident_nó =Identificador do nó a aplicar o sinal
valinic = Estado lógico inicial
Tinic = Instante inicial da aplicação do sinal
increm = Intervalo entre 2 transições (metade do período)

b) Estímulos aperiódicos

A *ident_nó valinic Tinic* T_1 [T_2 [T_3 [T_4 .... [T_n]]]]

onde:

ident_nó =	Identificador do nó a aplicar o sinal
valinic =	Estado lógico inicial
Tinic =	Instante inicial da aplicação do sinal
$T_1, T_2, \ldots T_n$ =	Instantes para as transições

c) Estímulos retangulares

R *ident_nó valinic Tinic largura gap*

onde:

ident_nó =	Identificador do nó a aplicar o sinal
valinic =	Estado lógico inicial
Tinic =	Instante inicial para aplicação do sinal
largura =	Largura do pulso (vide figura 10)
gap =	Tempo entre final de 1 pulso e o início do próximo (vide figura 10)

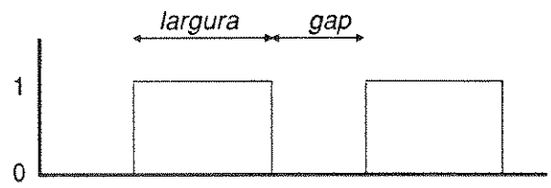


Figura 10 - Exemplo de um Sinal Retangular

Como extensão à linguagem do SIMUL, poderão ser definidos nos programas de testes funcionais alguns comandos voltados ao controle da execução dos testes. Tanto para estímulos de inicialização quanto para os vetores de teste, através do comando "**.COND**", poderão ser definidas condições de parada, em função do tempo e/ou de uma combinação de estados lógicos em determinados nós do circuito. No bloco de descrição dos vetores de teste poderão ainda ser utilizados os seguinte comandos: "**.PAUSA**", que define um conjunto de instantes para pausas no experimento; "**.PASSO**", que define que o experimento será executado passo a passo, durante um determinado intervalo de tempo. A descrição detalhada de cada um desses comandos será realizada na seção III.3.2.5.

III.3.2.4 - Descrição dos Sinais de Saída a serem Monitorados

A descrição dos sinais a serem monitorados manterá, para os programas de testes funcionais, a mesma sintaxe e semântica observadas para o SIMUL.

Além da definição dos sinais a serem monitorados, será possível definir também alguns comandos adicionais, voltados à definição de características da monitoração de sinais, como instante inicial e final para a monitoração, armazenamento de resultados em disco, etc... Descreveremos a seguir cada um desses comandos.

O comando "**.TINIC**" define o instante inicial para a visualização em tela ou armazenamento em disco dos dados de resultados dos testes. Caso este comando seja omitido o sistema assumirá que o instante inicial é igual a zero. A sintaxe para o comando é a seguinte:

.TINIC *instante*

onde

instante = Instante inicial para visualização ou armazenamento de resultados em disco.

O comando "**.TFIM**" define o instante final para visualização ou armazenamento dos dados. A sintaxe para o comando é descrita a seguir.

.TFIM *instante*

onde

instante = Instante final para visualização ou armazenamento de resultados em disco.

O comando "**.INCR**" define o incremento de tempo entre duas amostragens do sinal, a partir do instante inicial definido. Caso este comando não seja definido, o sistema assumirá o valor 1 (um) como *default*. A sintaxe do comando deverá obedecer ao seguinte formato:

.INCR *incremento*

onde

incremento = Incremento de tempo para amostragem do sinal.

Através do comando "**.MOSTR**" pode-se definir que os sinais serão apresentados ao usuário em tempo de execução ou imediatamente após o final dos testes. A forma de uso do comando é descrita abaixo.

.MOSTR ON | OFF

onde

ON = resultados devem ser mostrados em tempo real.
OFF = resultados devem ser mostrados somente após o término dos testes.

O comando "**.ARQ**" permitirá que sejam armazenados em disco os valores lógicos dos sinais monitorados. O parâmetro deste comando deverá definir o nome do arquivo a ser gerado, conforme mostrado a seguir.

.ARQ *nomearq*

onde

nomearq = nome do arquivo a ser gerado.

Observação: Os caracteres permitidos para a definição do nome do arquivo poderão variar de acordo com o sistema operacional utilizado. A implementação deverá levar em conta esse aspecto.

III.3.2.5 - Comandos de Controle da Execução

Os comandos de controle da execução do experimento têm por objetivo oferecer recursos que permitam interromper ou suspender temporariamente o processo de testes, de forma que o usuário possa intervir, seja para realizar análises parciais dos resultados, seja para redefinir alguma condição de teste.

Quaisquer dos comandos de controle de execução do experimento deverão ser definidos juntamente aos estímulos externos a serem aplicados ao circuito.

O comando "**.PAUSA**" permite ao usuário interromper os testes em determinados instantes da execução. O formato estabelecido para o comando é o seguinte:

.PAUSA $inst_1$ [, $inst_2$ [, $inst_3$ [... $inst_n$]]]

onde:

$inst_1$, $inst_2$, $inst_n$ = Instantes em que os testes devem ser interrompidos.

O comando **.PASSO** estabelece uma forma de permitir ao usuário a execução de um experimento, ou parte dele, passo a passo. Este comando é de grande interesse quando da necessidade de depuração de falhas já bem localizadas com relação ao instante de execução. Descrevemos a seguir o formato para o comando.

.PASSO *Tinic Tfim increm*

onde

Tinic = instante inicial para a execução passo a passo
Tfim = instante final para execução passo a passo
increm = incremento para pausa do experimento (passo)

Através do comando "**.COND**" o usuário pode definir condições de parada para a aplicação dos estímulos de inicialização do circuito ou para os estímulos de teste. A condição de parada é definida por uma expressão lógica que define uma combinação de estados lógicos em nós do circuito e/ou um determinado instante da execução.

III.3.3 - Aspectos da Implementação

A linguagem para a descrição dos testes funcionais apresenta-se como uma forma textual para a descrição de um experimento de testes. Considerando-se aspectos de facilidade de uso do sistema, a existência de uma ferramenta interativa voltada à descrição do experimento é mandatória. Essa ferramenta deverá, entretanto, gerar um arquivo com o programa do experimento definido segundo a linguagem aqui proposta.

O STEF, como discutimos anteriormente, será o sub-sistema do SAD que permitirá ao usuário programar, executar e realizar análises de resultados dos testes funcionais.

Um interpretador para a linguagem foi desenvolvido e incorporado ao STEF, e será descrito com mais detalhes no capítulo V. Na versão atual, o interpretador apresenta restrições quanto à interpretação da descrição do circuito. Somente as descrições dos pinos de entrada e saída são interpretadas, não estando incluída a descrição do *net-list* do circuito.

A programação do experimento no STEF poderá ser realizada tanto na forma textual quanto de maneira interativa. Os comandos de controle da execução permitidos no programa do experimento deverão remeter o usuário ao modo interativo.

Maiores detalhes sobre a implementação do interpretador para os programas de testes funcionais podem ser obtidos no capítulo V, onde é descrito o STEF.

III.3.4 - Exemplo de um Programa para Testes Funcionais

Apresentamos nesta seção um exemplo de programa para testes funcionais, procurando utilizar um circuito bastante simples e apresentar os recursos disponíveis através da linguagem de definição do experimento. O circuito utilizado para o exemplo é apresentado na figura 11.

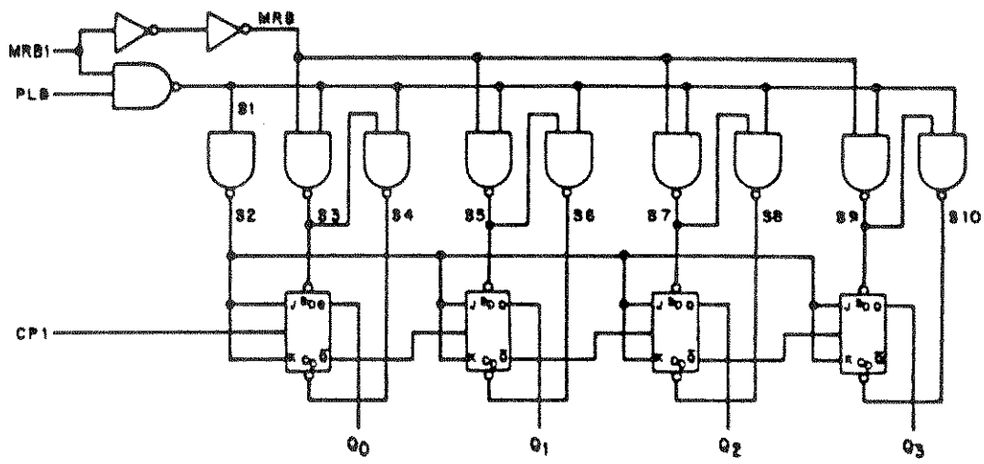


Figura 11- Circuito-Exemplo do Programa de Testes Funcionais

```
PIN
        P1    MRB1
        P2    PLB
        P3    CP1
        P10   Q3
        P11   Q2
        P12   Q1
        P13   Q0
FIM

ELE
        NOT     MRB1    0     0     0     0     M1A   0
        NOT     M1A     0     0     0     0     M1A   0
        NAND    MRB1    PLB   0     0     0     S1    0
        NOT     S1      0     0     0     0     S2    0
        NAND    MRB     S1    0     0     0     S3    0
        NAND    S1      S3    0     0     0     S4    0
        JKFF    S2      S2    CP1   S4    S3    Q0    Q0B
        NAND    MRB     S1    0     0     0     S5    0
        NAND    S1      S5    0     0     0     S6    0
        JKFF    S2      S2    Q0B   S6    S5    Q1    Q1B
        NAND    MRB     S1    0     0     0     S7    0
        NAND    S1      S7    0     0     0     S8    0
        JKFF    S2      S2    Q1B   S8    S7    Q2    Q2B
        NAND    MRB     S1    0     0     0     S9    0
        NAND    S1      S9    0     0     0     S10   0
        JKFF    S2      S2    Q2B   S10   S9    Q3    Q3B
FIM
```

```
INI
        A     MRB1      0     0     1     10
        A     PLB       1     0
FIM

EST
        A     MRB1      1     0
        R     CP1       0     0     5     5
FIM

SAI
        MOSTR     ON
        INCR      5
        TINIC     0
        TFIM      1000
        ARQ       teste.txt
        MRB1      PLB  CP1  Q3    Q2    Q1    Q0
FIM
```

III.4 - A Linguagem para Descrição das Medidas Analógicas

A proposta preliminar da linguagem para a descrição de experimentos de medidas analógicas foi realizada apoiando-se na linguagem utilizada pelo simulador SPICE. Como já discutimos anteriormente, a capacidade de estabelecer compatibilidade entre programas de simulação e programas de testes é bastante conveniente, tanto para a redução de custos quanto para reduzir a probabilidade de erros.

Neste capítulo procuramos definir alguns conceitos preliminares para a especificação da linguagem, como ponto de partida para sua implementação.

A linguagem utilizada pelo SPICE [Vla81] tem como uma de suas maiores qualidades o forte poder de síntese na descrição dos comandos para a simulação. Criamos algumas extensões dessa linguagem, face à necessidade de termos disponíveis para os testes alguns recursos adicionais para o controle da execução e para a aquisição e armazenamento de dados em si. Devemos ressaltar também, que há comandos para a simulação que não têm sentido quando da execução dos testes e que deverão ser desconsiderados.

Tanto no que diz respeito à sintaxe, quanto no que se refere à semântica da linguagem para definição de medidas analógicas procuramos preservar, na medida do possível, características idênticas à linguagem do SPICE, criando apenas alguns comandos adicionais. As seções seguintes apresentam os aspectos sintáticos e semânticos da linguagem proposta.

A definição da linguagem toma como premissa que o sistema de *software* voltado à execução incluirá os seguintes módulos:

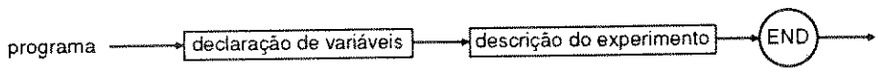
- um interpretador para a linguagem de descrição do experimento.
- um conjunto de *drivers* voltados ao controle e aquisição de dados dos instrumentos.
- um configurador através do qual o usuário possa definir os equipamentos disponíveis e suas respectivas características.
- um modo interativo de operação, através do qual o usuário poderá modificar dados de programação do experimento, visualizar diagramas do circuito e dados de resultados, controlar (interromper e reiniciar) a execução do experimento.

III.4.1 - Descrição Sintática

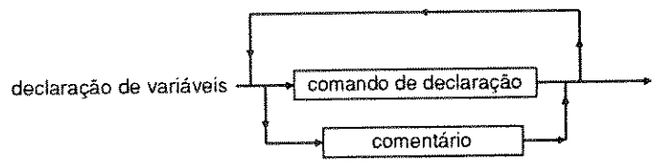
A sintaxe usada na definição de programas para execução de medidas analógicas deverá, de maneira geral, ser a mesma utilizada para os programas do SPICE. Foram definidos alguns comandos voltados especificamente para a linguagem de descrição das medidas analógicas, os quais têm como objetivo dar suporte a características particulares vinculadas aos instrumentos de medida, ao controle da execução das medidas e ao armazenamento de dados de resultados das mesmas.

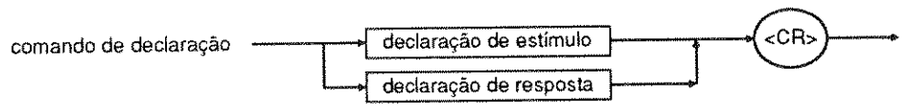
Apresentaremos nesta seção somente as definições essenciais à descrição dos comandos específicos à programação do experimento de medidas analógicas e dos comandos para o SPICE que tenham sofrido modificações sintáticas.

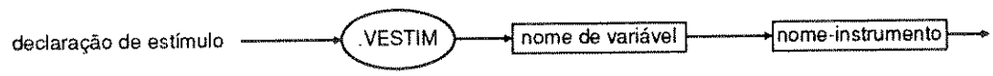
III.4.1.1 - Programas

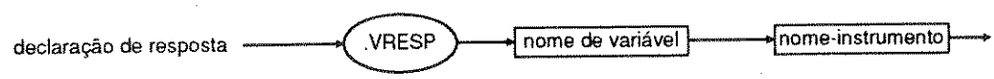


III.4.1.2 - Declarações de Variáveis

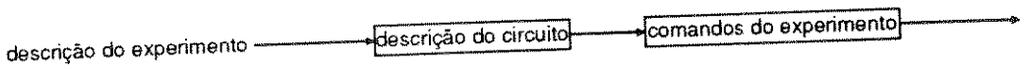


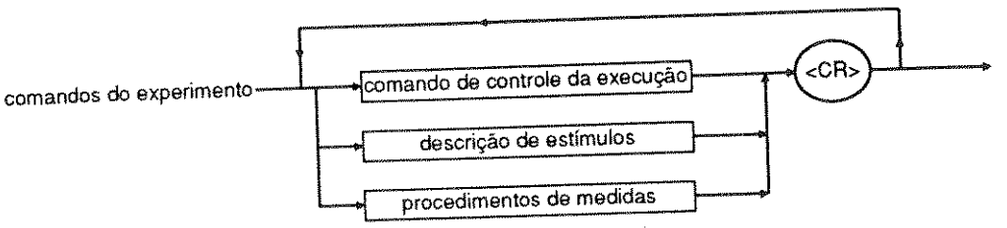






III.4.1.3 - Descrição do Experimento



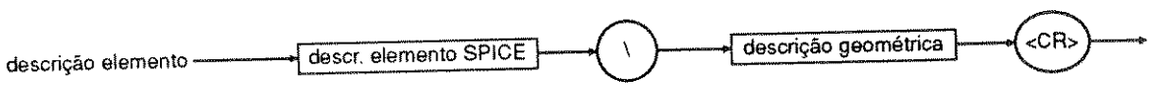


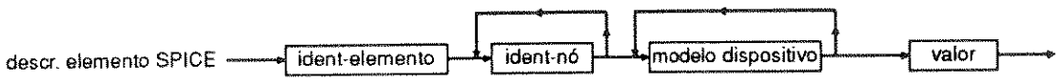
III.4.1.4 - Descrição do Circuito

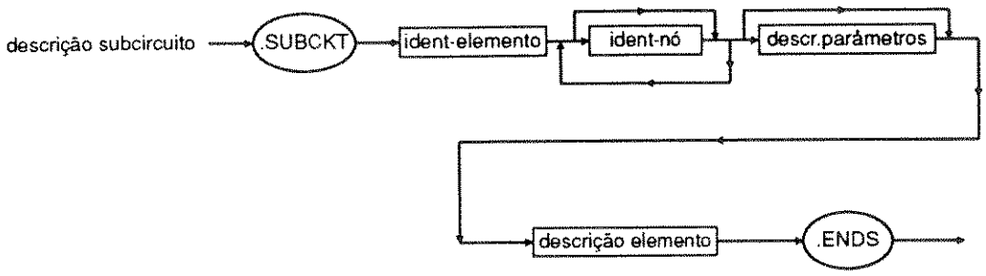
A descrição do circuito nos programas para medidas analógicas será utilizada somente para a visualização do esquema elétrico do circuito e identificação dos nós aos quais se aplicam estímulos e/ou nos quais são medidas respostas. É importante notar que, ao se acoplarem instrumentos ao circuito, o mesmo é alterado.

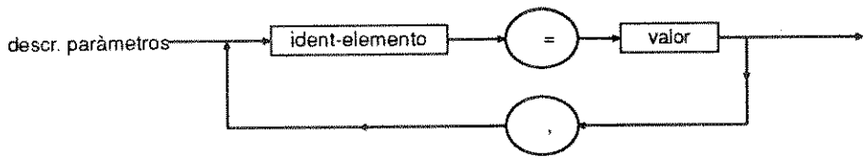
A sintaxe para a descrição do circuito será a mesma utilizada para o SPICE, sendo acrescentadas, ao final da descrição de cada elemento, informações para descrição do mesmo no diagrama esquemático.

Omitiremos na descrição sintática do circuito as produções para as quais a sintaxe seja idêntica à do SPICE.



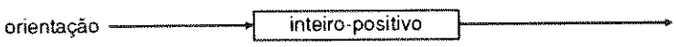

descr. elemento SPICE
ident-elemento
ident-nó
modelo dispositivo
valor

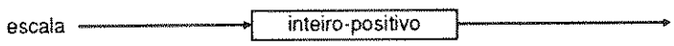

descrição subcircuito
.SUBCKT
ident-elemento
ident-nó
descr.parâmetros
descrição elemento
.ENDS


descr. paràmetros
ident-elemento
=
valor
,

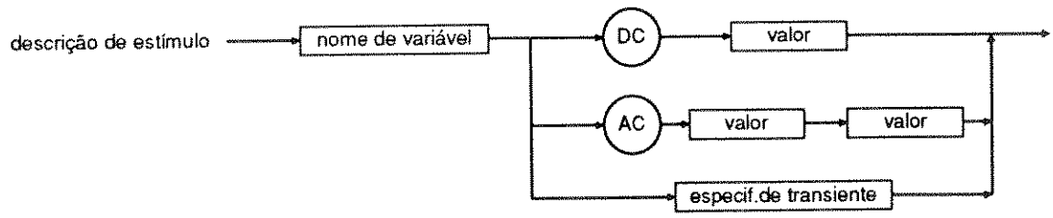

descrição geométrica
coordenadas-origem
orientação
escala

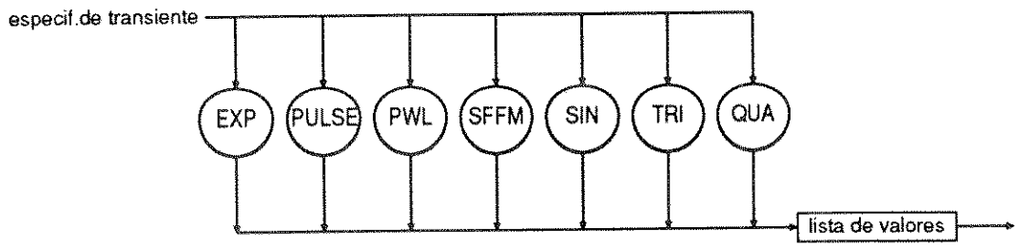

coordenadas-origem
inteiro-positivo
inteiro-positivo


orientação
inteiro-positivo

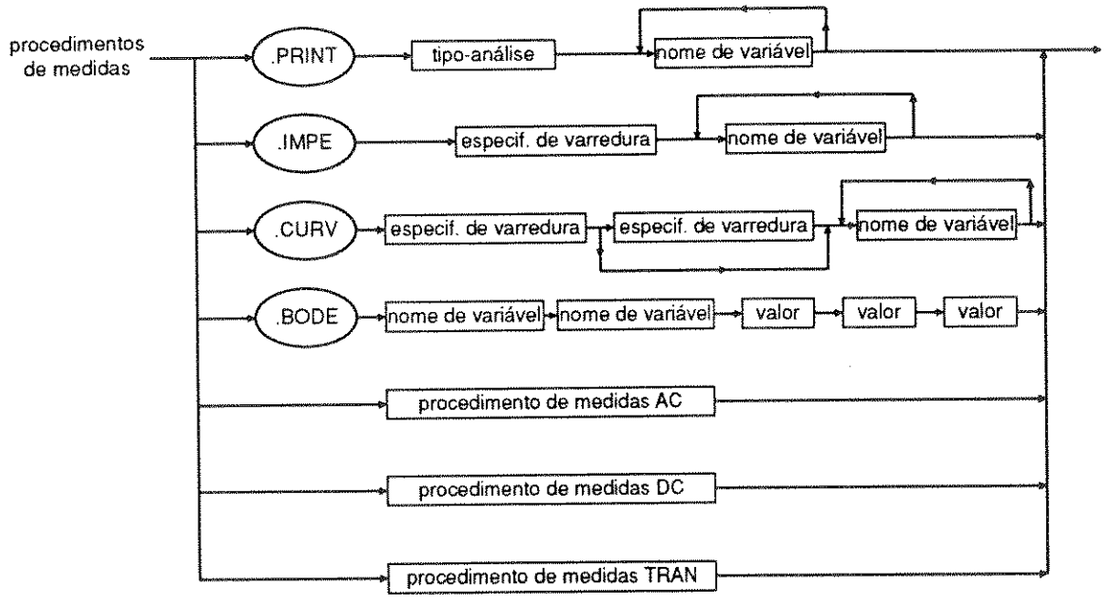

escala
inteiro-positivo

III.4.1.5 - Comandos de Descrição de Estímulos

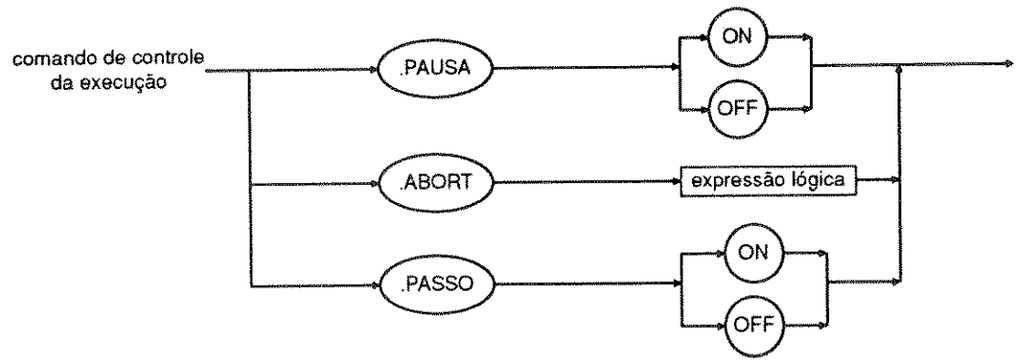




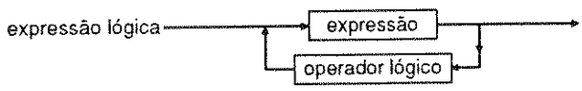
III.4.1.6 - Comandos de Procedimento de Medidas

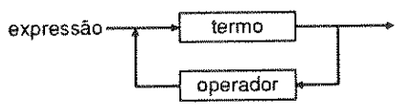


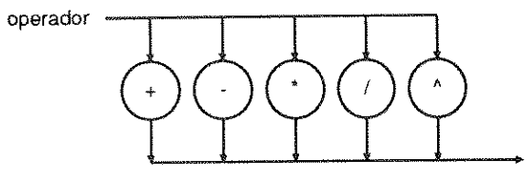
III.4.1.7 - Comandos de Controle da Execução

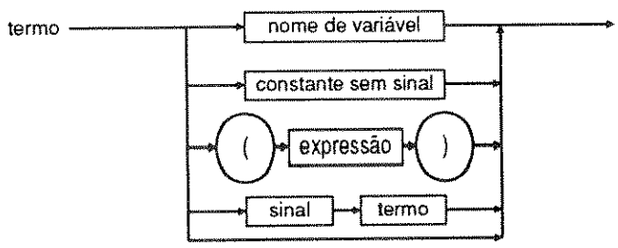


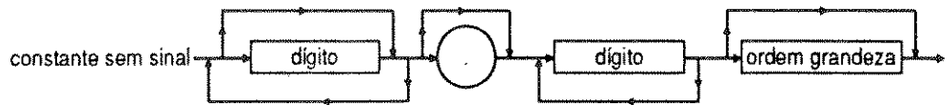
III.4.1.8 - Expressões, Listas de Valores e Valores

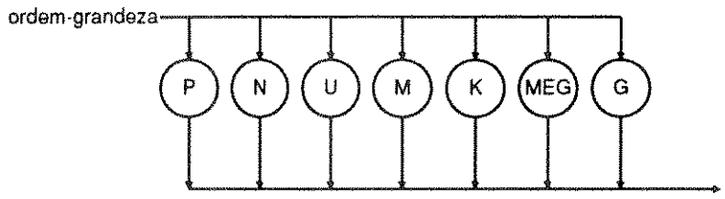


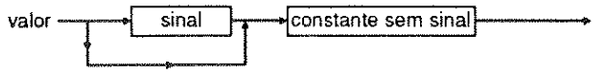


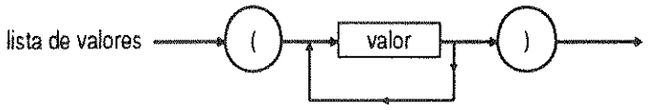


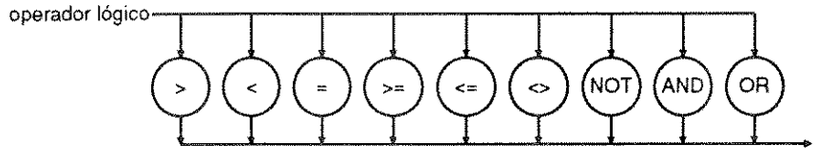




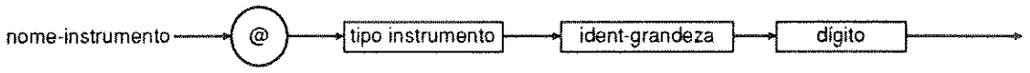


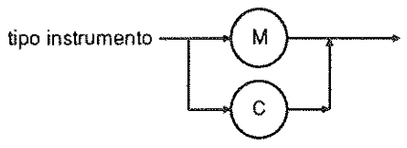


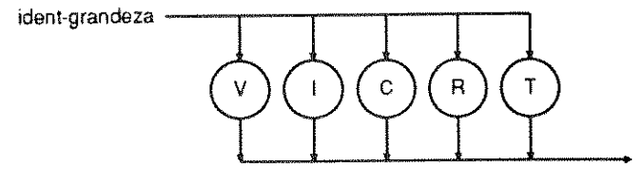


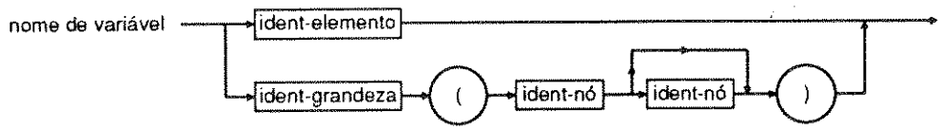


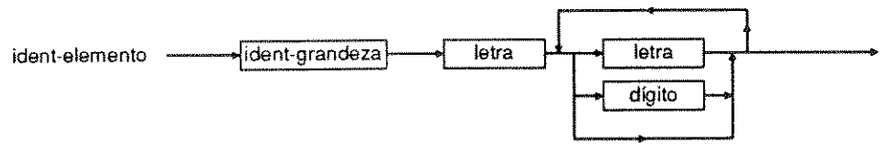
III.4.1.9 - Nomes de Variáveis e Identificadores de Instrumentos

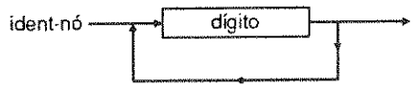




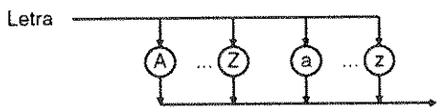


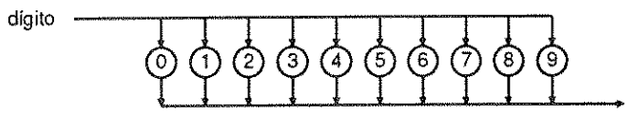


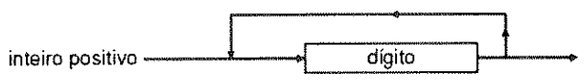




III.4.1.10 - Dígitos, Letras e outros Símbolos Básicos







III.4.2 - Descrição Semântica

Descreveremos nesta seção a semântica proposta para a linguagem de descrição de medidas analógicas. A discussão estará concentrada essencialmente nos comandos criados como extensão à linguagem do SPICE e nos comandos do SPICE que sofreram alterações semânticas.

III.4.2.1 - Programa

Um programa para medidas analógicas pode ser definido com um conjunto de declarações de variáveis, onde são definidos nomes lógicos para os equipamentos de medição e controle utilizados no experimento, seguido de uma seqüência de comandos, que definem o experimento propriamente dito.

As informações contidas nos programas para descrição das medidas analógicas podem ser classificadas nos seguintes grupos:

- Declarações de Variáveis
- Descrição do Circuito
- Descrição dos Estímulos a serem Aplicados
- Descrição das Medidas a Serem Efetuadas
- Comandos de Controle da Execução

Deve-se notar que as informações definidas por um comando podem se enquadrar em mais que um dos grupos acima descritos.

III.4.2.2 - Declarações de Variáveis

A declaração de variáveis estabelece uma associação entre elementos do circuito (fontes de alimentação, nós ou parâmetros) e os instrumentos de medição e controle, de forma que o sistema de *software* possa controlar adequadamente a aplicação de estímulos e a monitoração de sinais no circuito.

Para tal, é conveniente que cada instrumento de medição ou controle utilizado tenha associado a si um nome lógico, através do qual poderá ser referenciado no corpo de descrição do experimento. A quantidade e os tipos de instrumentos disponíveis poderão variar de acordo com o *hardware* de aquisição de dados utilizado. Portanto, as características e a identificação particulares de cada instrumento poderão ser configuradas e/ou definidas pelos usuários. A programação do experimento deve atentar a esse fato quando da declaração de variáveis, pois caso contrário o interpretador emitirá a mensagem de erro correspondente.

A declaração de variáveis é obrigatória para todos os instrumentos utilizados no experimento e deve ser realizada antes da descrição do experimento propriamente dito. O usuário poderá definir dois tipos distintos de variáveis:

- Estímulos a serem aplicados ao circuito
- Pontos do circuito a serem monitorados (respostas).

A declaração de variáveis referentes a estímulos a serem aplicados ao circuito será feita através do comando **.VESTIM**. O comando **.VESTIM** identifica o instrumento e define um nome simbólico para que ele possa ser referenciado no corpo de descrição do experimento. Os parâmetros dos sinais de estímulos (amplitude, forma de onda, etc...) serão definidos no corpo de descrição do experimento, através dos comandos apropriados (de descrição de estímulos ou de descrição de procedimentos). Cada comando de declaração de variável associa uma única variável a um único instrumento.

O comando **.VRESP** permite a declaração de variáveis que, no corpo de descrição do experimento, definirão os nós ou ramos do circuito onde serão realizadas medições (pontos que são variáveis dependentes dos estímulos), associando um instrumento de medição específico a essa variável.

A declaração de uma variável é composta pelo comando de declaração seguido do nome da variável e do nome (identificação) do instrumento que estará associado à variável durante a execução do experimento.

Os identificadores de instrumentos serão iniciados pelo caracter "@" seguido de um caracter que identifica o tipo de instrumento ("M" para equipamentos de medição e "C" para os de controle), seguido de um caracter que identifica a grandeza física medida pelo equipamento e do número do mesmo.

Deve-se observar que na declaração de variáveis não há nenhuma informação sobre características particulares dos instrumentos (como, por exemplo, resolução, faixas de operação, etc...). Essas informações, por pressuposto, serão de conhecimento do sistema de *software* responsável pela execução do experimento, sendo definidas através de opção específica para configuração de equipamentos.

A declaração de variáveis faz com que modificações num experimento decorrentes da simples troca de um instrumento de medição ou controle não afetem o corpo de descrição do experimento, já que os instrumentos serão sempre referenciados pelos nomes de variáveis.

III.4.2.3 - Descrição do Circuito

A descrição do circuito, quando da programação e execução de testes analógicos, será de interesse para permitir ao aluno escolher e selecionar com maior facilidade os pontos do circuito onde se deseja aplicar ou medir sinais. A visualização do diagrama esquemático do circuito em tela poderá ser um auxílio para checar, por exemplo, se a montagem está correta.

O *net-list* do circuito está incluído na descrição de programas para o SPICE. Essa descrição, porém, não inclui algumas informações adicionais necessárias à descrição de um diagrama, como as posições e escalas de cada elemento do circuito no diagrama esquemático. Nossa proposta inicial foi adicionar-se, ao final de uma linha de descrição de um elemento do circuito, essas informações. Estudos mais detalhados nesse sentido deverão ser promovidos a fim de verificar a adequação dessa descrição, dado que diversos simuladores comercialmente disponíveis incorporam atualmente programas de captura esquemática.

A descrição de um circuito para o SPICE poderá incluir também a definição de modelos de dispositivos (descritos através do comando **.MODEL**). Esse tipo de definição não será de interesse ao SAD quando da execução do experimento e não será utilizado pelo programa de medidas.

Na visualização do diagrama elétrico, os subcircuitos serão apresentados como blocos, devendo ser possível a visualização do conteúdo de cada bloco através de operações de *Zoom*.

III.4.2.4 - Descrição dos Estímulos a serem Aplicados

Os estímulos a serem aplicados ao circuito são descritos de maneira análoga à do SPICE, sendo necessário que as variáveis sejam previamente declaradas, conforme descrito na seção III.4.2.1.

Para a definição dos estímulos, devem ser consideradas as características do *hardware* instalado, pois isso afetará, por exemplo, as faixas de valores para amplitudes e outros parâmetros permitidos para os estímulos.

Os estímulos serão descritos de três formas distintas: **a)** pela definição de fontes de estímulos; **b)** através de parâmetros que definem o controle dos instrumentos durante os procedimentos de medidas; **c)** pela definição de variáveis não elétricas (como, por exemplo, temperatura).

É importante observar que alguns dos comandos propostos apresentarão resultados que poderiam ser obtidos através de outros comandos mais genéricos. A inclusão desses comandos teve duas motivações básicas: **a)** facilitar ao usuário a descrição de algumas das medições mais freqüentemente utilizadas; **b)** oferecer maiores facilidades para a compatibilização entre programas de simulação e de medições.

a) Fontes de Alimentação e Sinais

A descrição de estímulos (fontes de alimentação e geradores de sinais) inclui fontes de tensão ou de corrente e a sintaxe desses comandos será idêntica à do SPICE, com exceção das especificações de sinais transientes. Nesse caso, além das opções disponíveis no SPICE (EXP, PULSE, PWL, SFFM e SIN), foram incluídas opções específicas para a definição de formas de onda quadradas ("QUA") e triangulares ("TRI"). O formato para a definição desses transientes é descrito a seguir.

TRI (*val_dc amplitude freqüência*)

onde

val_dc =	valor DC da tensão ou corrente.
amplitude =	Amplitude da tensão ou corrente.
freqüência =	freqüência do sinal (em Hertz).

QUA (*val_dc amplitude freqüência*)

onde

val_dc =	valor DC da tensão ou corrente.
amplitude =	Amplitude da tensão ou corrente.
freqüência =	freqüência do sinal (em Hertz).

Mais uma vez, devemos ressaltar que o nome da variável deve estar associado a um equipamento específico, já que diferentes tipos de estímulos poderão corresponder a diferentes equipamentos.

b) Comandos de Procedimento de Medidas

Os comandos de procedimento de medidas definem um conjunto de estímulos a serem aplicados ao circuito, um conjunto de respostas e serem medidas, o armazenamento de resultados e pausas do experimento (por exemplo, para o acoplamento de instrumentos ou para ajustes de escalas dos mesmos).

Sob a óptica da definição da linguagem, os comandos de procedimento de medidas podem ser divididos em dois subgrupos. O primeiro deles reúne comandos derivados de comandos de análise do SPICE (.AC, .DC, .TRAN). O segundo grupo reúne os comandos definidos especificamente para a linguagem de descrição dos experimentos, visando basicamente à medição de grandezas que para o SPICE são definidas como parâmetros ou constantes e à facilidade de programação das medidas mais comumente realizadas.

Um comando de procedimento de medidas corresponderá, em tempo de execução do experimento, à seguinte seqüência de etapas (simplificada):

1) o sistema de *software*:
    1.1) Interpreta um comando de procedimento de medidas.
    1.2) Se necessário, busca e interpreta o comando de definição de respostas (.PRINT) associado a ele.
    1.3) informa ao usuário as montagens necessárias à execução das medições.
    1.4) aguarda confirmação do usuário de que a montagem foi completada.
2) o aluno
    2.1) faz as montagens e ajustes de instrumentos indicados pelo sistema de *software*.
    2.2) confirma (ao sistema de *software*) que a montagem está completa.
3) o sistema de *software*
    3.1) faz a aplicação de estímulos ao circuito.
    3.2) mede as respostas necessárias.
    3.3) armazena os resultados.
    3.4) Volta ao item 3.1.

Podemos observar que as etapas do passo 3 correspondem à execução do experimento em si. O sistema de *software* responsável pela execução do experimento deve prever a interrupção (e eventual retomada) do experimento em qualquer uma dessas etapas. Essas interrupções poderão ser ocasionadas por: **a)** comandos de controle de execução definidos no programa do experimento; **b)** interrupção interativa do experimento, através do teclado; **c)** necessidade de ajuste de escalas dos instrumentos (de forma interativa ou através do controle automático dos instrumentos pelo próprio sistema de *software*).

Em qualquer um dos casos de interrupção do experimento acima descritos, será possível retomar a execução do experimento desde o início ou a partir do ponto em que este foi interrompido.

A motivação básica da utilização dos comandos derivados do SPICE foi a compatibilização entre as linguagens de simulação e de descrição dos experimentos. Em geral, os comandos de análise do SPICE são a essência das simulações realizadas e, portanto, é de grande valia a capacidade de intercambiar esses comandos com a linguagem de descrição dos experimentos.

A proposta da linguagem inclui os comandos **.AC**, **.DC** e **.TRAN**, que terão sintaxe idêntica àquela utilizada pelo SPICE, com modificações semânticas impostas à realização dos experimentos, as quais serão descritas mais adiante.

Deve-se observar que não foram incluídos os comandos **.DISTO** e **.NOISE**. Para esses comandos haveria a necessidade da utilização de instrumentos especializados para a realização do experimento e/ou da aplicação de tratamentos matemáticos aos resultados. Estudos futuros poderão avaliar a viabilidade e conveniência da implementação desses comandos.

Os comandos de procedimento de medidas derivados dos comandos de análise do SPICE, quando utilizados, deverão obrigatoriamente ser usados em conjunção com o comando **.PRINT**, através do qual serão definidos os resultados a serem armazenados.

O segundo grupo de comandos de procedimento de medidas, que inclui os comandos definidos especificamente para a linguagem de descrição do experimento, foi definido de forma que um único comando inclua a descrição dos estímulos a serem aplicados ao circuito e as respostas a serem medidas (deve-se notar que no SPICE usa-se, em geral, um comando de análise seguido de um comando .PRINT ou .PLOT). A descrição de estímulos e respostas num mesmo comando facilita a programação do experimento.

O comando "**.CURV**" permite a aplicação de sinais a pontos específicos do circuito e a medição de dados e geração de tabelas para outras grandezas mensuráveis pelo sistema. O usuário deverá definir uma ou duas variáveis de varredura (que correspondem aos sinais a serem aplicados) e até dez variáveis a serem medidas (respostas aos estímulos ou parâmetros). As variáveis a serem medidas poderão ser tensões, correntes ou parâmetros medidos diretamente. Este comando é semelhante ao comando "**.DC**", com as seguintes diferenças: **a)** não há a necessidade de definir-se um comando adicional ("**.PRINT**") para a descrição das respostas a serem medidas ; **b)** as variáveis medidas podem incluir quaisquer parâmetros diretamente mensuráveis pelo sistema. São descritos a seguir os parâmetros utilizados para este comando.

.CURV *var1 min1 max1 nptos1 var2 min2 max2 nptos2 (par1 [par2 [...[parn]]])*

onde

varx = nome lógico da variável de varredura (eixo X).
min = valor mínimo para a variável de varredura.
max = valor máximo para a variável de varredura.
nptos = número de pontos (amostras) a serem medidos.
par1, par2, parn = nomes lógicos das variáveis a serem medidas

O comando "**.BODE**" permitirá a medição de dados que determinem a resposta de um sinal em função da freqüência (curvas de Bode). A tabela de saída gerada por este comando deverá relacionar a freqüência contra (*var2 / var1*). Este comando é uma particularização do comando "**.AC**", tendo sido incluído por ser um tipo de medição muito freqüentemente realizada. O formato do comando é descrito a seguir.

.BODE *var1 var2 freq1 freq2 nptos*

onde

var1 = variável de entrada para medição da resposta em freqüência (módulo e fase).
var2 = variável do ponto de saída (módulo e fase).
freq1 = freqüência inicial para as medições.
freq2 = freqüência final para as medições.
nptos = número de pontos medidos por década. (Vide observações).

O comando "**.IMPE**" permitirá o controle de equipamentos para realizar medições voltadas à verificação de valores de impedância entre dois nós do circuito em função da freqüência aplicada ou outro parâmetro, como tensão ou temperatura. Não há correspondência deste comando com nenhum comando do SPICE. Os parâmetros para o comando são descritos a seguir.

.IMPE *entr param1 param2 nptos sai1 [sai2 [... [saiₙ]]]*

onde

entr = nome da variável associada ao sinal de entrada.
param1 = Valor inicial do parâmetro de estímulo para a análise.
param2 = Valor final do parâmetro de estímulo para a análise.
nptos = número de pontos medidos. (Vide observações).
sai1, sai2, saiₙ = nomes das variáveis associadas aos nós a serem medidos.

Observações: Para os comandos "**.BODE**" e "**.IMPE**", o parâmetro *nptos* define o número de pontos a serem medidos. O usuário pode definir o número de pontos por décadas (que é o *default*), por oitavas ou em termos absolutos (linear), acrescentando-se uma letra imediatamente após o valor do número de pontos. Serão aceitas as letras "D", "L" e "O", representando respectivamente décadas, linear e oitavas.

c) Definição de Parâmetros

A definição de parâmetros refere-se à aplicação de estímulos não elétricos ao circuito, como é o caso da temperatura.

A aplicação desse tipo de estímulo ao circuito poderá ser realizada através do comando "**.TPARAM**", análogo ao comando **.PARAM** do SPICE. Ao se aplicar um estímulo não-elétrico ao circuito, o sistema de *software* responsável pela execução do experimento deverá interromper temporariamente o experimento até que a variável associada ao parâmetro atinja seu valor nominal e se estabilize. Por exemplo, se o parâmetro for a temperatura, pode ser necessário um intervalo de tempo razoavelmente grande até que o circuito atinja a temperatura definida no programa e, nesse caso, os testes não devem prosseguir antes disso pois, dessa forma, estaríamos realizando medições espúrias.

No caso específico da temperatura, pode-se definir também o comando "**.TEMP**", que terá sintaxe idêntica à do SPICE.

III.4.3.6 - Comandos de Controle da Execução

Na definição dos programas para medidas analógicas será bastante comum que o usuário agregue, em um único programa, mais que um tipo de teste. Por exemplo, ele pode realizar, no mesmo programa, a medida do ponto de operação e do ganho de um circuito.

É, portanto, de interesse do aluno que existam recursos que permitam a ele interromper temporariamente a execução de um determinado teste, devendo haver a possibilidade de retomá-lo ou de abandoná-lo.

Os comandos de controle da execução do experimento implementarão esses recursos, permitindo que se interrompa ou cancele um experimento antes de seu término ou ainda que se execute passo a passo o experimento.

Para que os comandos de controle da execução tenham efeito, é necessário que o sistema de execução das medidas possuam, além do modo de operação automático, um modo de operação manual ou interativo.

Ao se interromper um teste, o sistema deve migrar do modo automático para o modo de programação interativa, permitindo que o usuário redefina características dos testes, realize análises parciais dos resultados, modifique o circuito e, eventualmente, retome a execução.

O comando "**.PAUSA**" permitirá ao operador definir pontos específicos para pausas (*"breakpoints"*) no programa. Se utilizado com os parâmetros *ON* ou *OFF*, definirá se o programa é ou não interruptível através do teclado. A interrupção do experimento por meio do teclado será realizada pressionando-se uma combinação específica de teclas.

O comando **".PASSO"** permitirá ao usuário executar passo a passo um determinado trecho do programa do experimento. Com o modo passo a passo ativo, o sistema interromperá a execução dos testes a cada linha de comando executada. O estado *default* para o sistema será a execução passo a passo desabilitada.

O comando "**.ABORT**" permite que o usuário defina um conjunto de condições que, caso ocorram, provoquem o término imediato do experimento. Este comando permite, por exemplo, que sejam evitadas condições que possam vir a danificar o circuito, como *overflows* e *underflows*. A expressão que define a condição de parada poderá conter como operandos valores absolutos, valores de parâmetros e variáveis do circuito.

Deve-se observar que, durante a execução dos experimentos, o sistema de *software* poderá (e deverá) realizar interrupções automáticas do programa, seja para o acoplamento de instrumentos ao circuito, para ajustes de escalas de equipamentos ou por outros motivos de natureza semelhante. Essas interrupções do experimento podem ou não ser preceptíveis pelo usuário.

III.4.3 - Aspectos da Implementação

Na implementação do interpretador para a linguagem de medidas analógicas, alguns aspectos referentes ao *hardware* voltado à aquisição dos dados podem influenciar em alguns fatores como, por exemplo, o número de equipamentos disponíveis, as faixas de valores aceitos, etc... Sugere-se que na implementação do sistema de execução de medidas analógicas, as informações sobre os equipamentos disponíveis e suas características sejam configuráveis pelo usuário e que essa informações de configuração sejam acessíveis ao interpretador do programa do experimento, a fim de permitir que ele acuse erros na definição de equipamentos.

Conforme foi citado anteriormente, na geração de programas para medidas analógicas a partir de programas do SPICE e vice-versa, alguns comandos serão desconsiderados, em função das diferentes características entre simulações e medidas. Apesar dessas informações serem desconsideradas, elas deverão estar presentes nos programas gerados (por exemplo, na forma de comentários com delimitadores especiais), a fim de possibilitar uma posterior conversão no sentido inverso sem que haja perda de informações do programa original.

IV - Tratamento e Análise de Resultados (STAG)

O STAG - Sistema de Tratamento e Análise Gráfica de Dados é o subsistema do SAD que dá apoio ao tratamento e análise de resultados de medidas.

Através do STAG, o aluno poderá visualizar graficamente resultados de medidas e simulações e terá à sua disposição um conjunto de ferramentas voltado à análise desses resultados.

Atuando de forma interativa e com as ferramentas de análise, o sistema deverá permitir ao usuário ampla capacidade de manipulação dos dados, tanto no que se refere à visualização gráfica quanto na realização de análises comparativas entre resultados de testes e simulações, análises estatísticas, extração de parâmetros de dispositivos, entre outros.

Uma versão inicial do STAG foi implementada e encontra-se atualmente em fase de beta-testes. Para essa versão inicial estão disponíveis somente as operações consideradas essenciais à utilização do sistema. Algumas funções adicionais encontram-se em fase de testes de integração. Apresentamos no anexo III o manual de usuário do STAG para a versão inicial.

Faremos neste capítulo uma descrição geral do STAG, passando em seguida à especificação de requisitos de *software*, à descrição das características mais relevantes da implementação e, por fim, à discussão das principais evoluções previstas para versões futuras.

IV.1 - Descrição Geral do STAG

IV.1.1 - Perspectivas do Sistema

O STAG destina-se ao tratamento e análise de resultados de medidas e de simulações de desempenho analógico e paramétricas, tendo como ambiente-alvo os laboratórios de eletrônica de universidades e escolas técnicas. O STAG inclui-se como uma das várias ferramentas que comporão o SAD - Sistema de Aquisição de Dados. Atuando de forma integrada, o SAD e o SDP - Sistema Didático de Projetos, deverão compor um ambiente completo voltado aos laboratórios.

Dentre as análises permitidas através do STAG, destacam-se aquelas voltadas à comparação entre resultados obtidos através das simulações e resultados das medidas efetuadas.

O grupo de usuários previsto para o STAG deverá ser composto por alunos de graduação e de pós-graduação de cursos ligados à área de eletrônica.

IV.1.2 - Conceitos e Definições

Durante a realização de testes e simulações de um circuito é bastante comum que o aluno gere um número relativamente grande de tabelas de resultados para o posterior levantamento de curvas, tabelas estas que dizem respeito a diferentes estudos acerca do circuito e que estarão contidas em diversos arquivos de dados.

A fim de permitir ao aluno uma melhor organização durante a realização dos tratamentos e análises de resultados, criou-se no STAG uma estruturação hierárquica para a organização das informações pertinentes a um mesmo projeto.

Estabeleceremos a seguir os conceitos e definições associados à hierarquização dos dados realizada pelo STAG para os experimentos.

- **Projeto:** Para o STAG, um projeto tem associados a si um conjunto de **arquivos de dados** a serem analisados e um arquivo onde o aluno poderá efetuar anotações referentes ao projeto (chamado de **arquivo de anotações**). Na versão atual do sistema, um projeto pode ter associados a si até dez arquivos de dados e um único arquivo de anotações.

- **Arquivos de Dados:** Estes arquivos deverão conter basicamente tabelas de dados com os resultados de simulações e medições realizadas pelo aluno. Na versão atual do sistema, cada arquivo de dados poderá conter até dez tabelas diferentes. O sistema será capaz de interpretar arquivos de dados no formato de saída do SPICE, arquivos de resultados de medidas do próprio SAD ou arquivos gerados por ferramentas externas, conforme formato específico definido no anexo I.

- **Arquivo de Anotações:** Este arquivo deverá ser utilizado pelo aluno para registrar dados e observações de interesse acerca das análises por ele realizadas. As anotações poderão ser realizadas de forma automática pelo STAG ou manualmente, através de um editor de textos.

- **Tabelas de Dados:** As tabelas de dados deverão relacionar um conjunto de valores para duas ou mais variáveis correlatas. As tabelas poderão representar uma curva, um conjunto de curvas ou ainda famílias de curvas. A organização da tabela deverá ser feita na forma de linhas e colunas, sendo que cada coluna corresponderá a uma variável. Para o traçado das curvas, a variável correspondente à primeira coluna da tabela será tomada como a variável de varredura. As demais colunas serão tomadas, uma a uma, como funções da variável de varredura. Tratamentos voltados à representação gráfica de funções de mais de uma variável serão realizados na forma de famílias de curvas, conforme discutido a seguir. Na versão atual do sistema, uma tabela de dados poderá conter até dez variáveis.

- **Famílias de Curvas**: Como discutimos anteriormente, o STAG permite somente o traçado de curvas em duas dimensões (funções de uma variável). É conveniente, porém que seja possível ao aluno realizar análises sobre funções de mais de uma variável. Uma função de *n* variáveis, pode porém ser representada em duas dimensões, através da parametrização de *(n-1)* variáveis. Como resultado dessa parametrização obtém-se um conjunto de curvas, uma para cada combinação de valores das variáveis parametrizadas. Esse conjunto de curvas é chamado de **família de curvas**.

- **Curvas:** Para o STAG, define-se como uma curva a representação gráfica de uma sequência de coordenadas (x,y), que em geral expressam uma função de uma variável (y= F(x)).

A figura 13 apresenta um diagrama da hierarquia estabelecida entre os diversos elementos acima definidos.

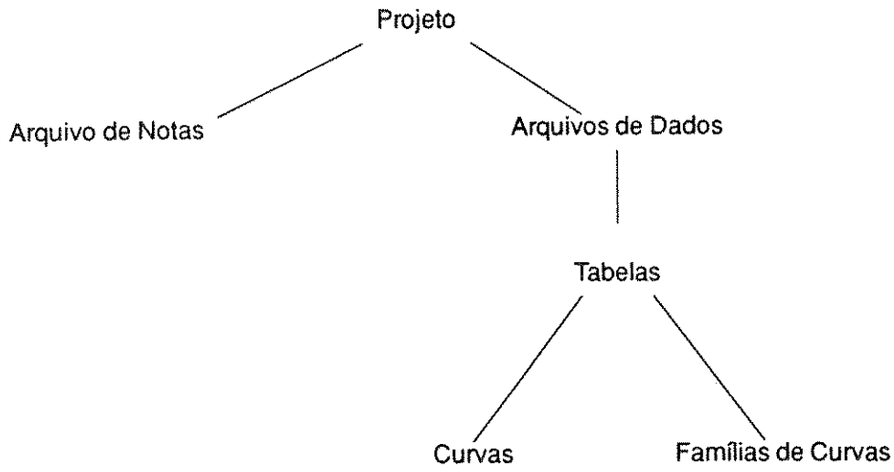


Figura 13 - Hierarquia entre Elementos de Projeto no STAG

IV.1.3 - Suposições, Dependências e Restrições

O STAG foi proposto tomando como suposição básica que o ambiente de execução será aquele definido na seção II.2 deste trabalho.

Considerando-se que o STAG deverá operar basicamente em modo gráfico, deve-se considerar que o uso de monitores de vídeo de baixa resolução poderá afetar de forma significativa a capacidade de visualização gráfica dos dados.

É suposto também que os usuários do STAG tenham conhecimentos básicos de operação de microcomputadores.

IV.2 - Especificação de Requisitos

IV.2.1 - Requisitos Funcionais

Conforme foi citado anteriormente, o STAG deverá, a partir de dados de resultados de testes e simulações, permitir ao aluno executar um conjunto de funções de tratamento e análise, através dos quais poderão ser extraídas informações sobre o comportamento analógico dos circuitos e, a partir destas, dados sobre parâmetros físicos e elétricos de dispositivos, bem como dados para a otimização dos mesmos. Além disso, o STAG deverá permitir também a realização de análises comparativas entre resultados de testes e simulações.

Do ponto de vista de organização interna de suas funções, o STAG pode ser particionado nos seguintes módulos:

- Leitura e Conversão de Formato de Dados de Entrada
- Traçado e Tratamento Gráfico de Curvas
- Cálculos e Análises matemático-estatísticas
- Gerenciamento do Projeto
- Extração de Dados Paramétricos e para Otimização

O diagrama apresentando os relacionamentos entre os módulos acima descritos e os dados de entrada e saída do sistema é apresentado na figura 14.

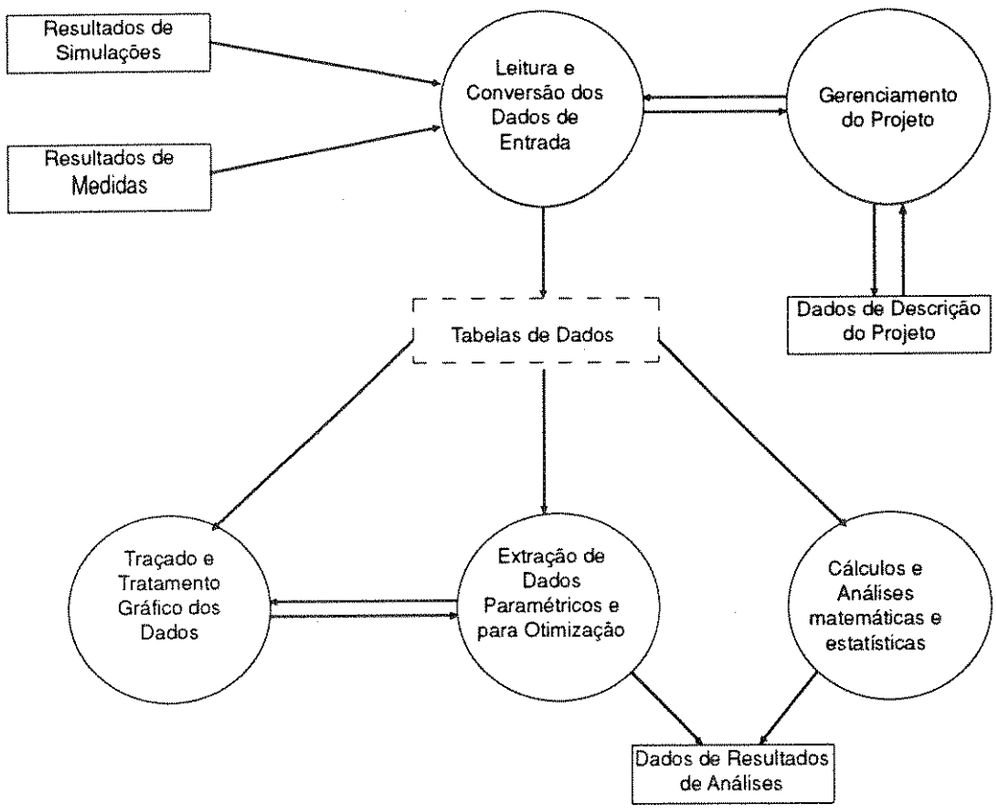


Figura 14 - Principais Módulos Funcionais do STAG

IV.2.1.1 - Descrição das Operações Básicas

O STAG deverá oferecer ao aluno um conjunto de funções que lhe permitam realizar as seguintes operações:

- Traçado e Tratamento de Curvas
- Cálculos de Regressões
- Cálculo de Transformadas de Fourier
- Cálculos e Análises Estatísticas Básicas
- Comparações de Resultados de Testes e Simulações
- Extração de Parâmetros de Dispositivos
- Extração de Dados para Otimização
- Configuração do Sistema

Com vistas a uma melhor organização das atividades do aluno, as operações acima descritas foram agrupadas da seguinte maneira:

- Operações para Definição do Projeto
- Operações sobre Arquivos
- Operações de Tratamento dos Gráficos
- Operações de Análise dos Dados
- Operações de Configuração do Sistema

Descreveremos a seguir cada um desses grupos de funções, não nos preocupando, por enquanto, com detalhes do aspecto de interação entre o aluno e o sistema. Esses detalhes serão discutidos na seção IV.3, que trata da implementação do STAG.

a) Operações para Definição do Projeto

As operações voltadas à definição do projeto permitirão que o usuário defina os conjuntos de dados a serem analisados e características particulares desses dados, além dos relacionamentos entre eles.

Estarão disponíveis ao aluno as operações básicas de criação, modificação, exclusão (eliminação) e consulta aos dados de um projeto.

Um projeto tem associados a si um conjunto de até dez arquivos de dados e um único arquivo de anotações. Cada um dos arquivos de dados poderá conter até dez tabelas de dados, e cada uma das tabelas de dados poderá descrever uma curva, um conjunto de curvas distintas ou uma família de curvas. O arquivo de anotações permitirá que o aluno realize anotações de informações acerca dos estudos e análises realizadas.

Considerando-se que a definição do projeto tem como único objetivo permitir ao aluno uma melhor organização das informações durante a análise de resultados, ela não será obrigatória para a utilização do sistema. Considerando-se porém que, em geral, a quantidade de tabelas e arquivos de dados envolvidas no processo de análise é grande, a definição do projeto é fortemente recomendada.

b) Operações sobre Arquivos

O conjunto de operações sobre arquivos reúne as operações voltadas à manipulação dos arquivos de dados e de anotações, além de algumas funções acessórias. Estarão disponíveis as seguintes operações:

- Ler Dados
- Gravar Dados
- Anotações
- Consultar Diretório

A operação de leitura dos dados permitirá que sejam lidos arquivos de dados no formato de saída do simulador SPICE ou no formato gerado pelo SAD. Para arquivos gerados pelo SPICE, o sistema é capaz de extrair automaticamente as tabelas de dados geradas para os diferentes tipos de análise (AC, DC, transientes, distorção e ruído). A implementação dessas funções deve ser realizada de forma a permitir que futuramente sejam suportados outros formatos dos dados de entrada. Ao ser lida uma tabela de dados, as curvas nela definidas são automaticamente apresentadas em tela.

Um dos compromissos básicos do STAG é permitir ao aluno realizar análises comparativas entre resultados de medidas efetuadas e simulações. Para tal, o sistema permitirá que sejam lidos e mantidos simultaneamente em memória dados de diversos arquivos (no máximo 10, na versão inicial).

Através da operação de gravação de dados é possível ao aluno armazenar num arquivo em disco uma tabela em formato texto contendo os dados atualmente apresentados em tela.

A operação de Anotações dá acesso a um editor de textos, onde o aluno poderá modificar o arquivo de anotações associado ao projeto atualmente carregado. O arquivo de anotações poderá conter informações geradas automaticamente pelo STAG (como resultados de operações de análises) e dados digitados pelo usuário, através do uso do editor de textos.

A operação de consulta a diretórios permitirá ao aluno verificar arquivos presentes num determinado subdiretório em disco.

c) Operações de Tratamento dos Gráficos

As operações voltadas ao tratamento dos gráficos permitem o traçado de curvas na tela e a realização de um conjunto de operações de tratamento visual dessas curvas. Incluímos também neste grupo de funções, alguns recursos que não manipulam diretamente as curvas, mas que estão associados à visualização dos gráficos.

Diversas operações de tratamento dos dados podem também ser utilizadas no auxílio à análise de características específicas das curvas. A figura 15 apresenta um exemplo de tela do STAG, destacando as principais características ligadas às operações de tratamento dos gráficos.

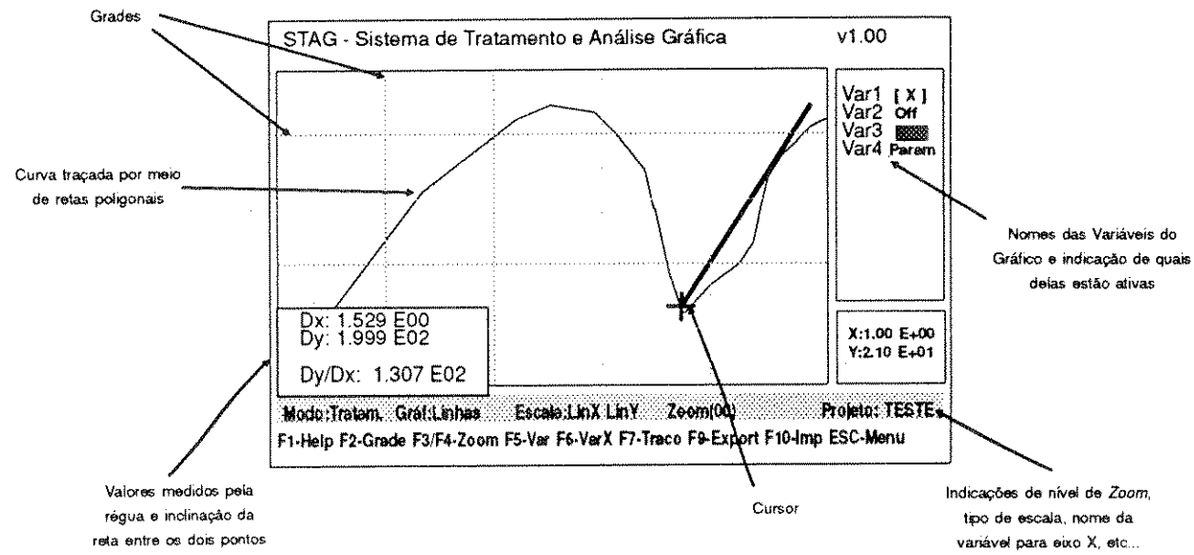


Figura 15 - Detalhes de tela para operações de tratamento

c.1) Cursor

O cursor será um dos recursos básicos às operações de tratamento e análise dos gráficos.

O usuário poderá selecionar dois métodos diferentes de **movimentação do cursor**. O primeiro deles, que chamaremos de movimentação **normal** permite que o cursor se desloque livremente pela região do gráfico na tela. O segundo restringe a movimentação do cursor, permitindo que se caminhe somente sobre **pontos das curvas** traçadas. Este segundo método de deslocamento do cursor é bastante útil quando o usuário quer verificar valores exatos das coordenadas dos pontos das curvas, dado que a apresentação das mesmas na tela é limitada pela resolução gráfica do monitor de vídeo e pode introduzir imprecisões oriundas de arrendondamentos de valores.

Outra função relacionada ao cursor é a definição do **tipo de cursor**. Esta função permite que o usuário escolha o formato do cursor a ser mostrado na tela. A versão inicial do STAG possibilita o uso de um cursor em formato de cruz (que é o *default* do sistema), em formato de circunferência ou na forma de linhas paralelas aos eixos do gráfico.

c.2) Grade

A função **grade** não afeta as curvas propriamente ditas, mas apresenta-se como um recurso de interesse na visualização das mesmas. A grade é um conjunto de linhas tracejadas paralelas aos eixos X e Y, usadas para facilitar a associação visual de valores de pontos das curvas com as escalas dos eixos. O aluno poderá ativar ou desativar a grade a qualquer instante durante a visualização do gráfico, conforme julgar conveniente.

c.3) Tipo de Traçado

Deverá ser permitido ao aluno escolher a forma como os dados são representados graficamente na tela. O STAG deverá oferecer três tipos de traçado:

- Traçado por Pontos
- Traçado Poligonal
- Traçado Interpolado

No traçado **por pontos** são mostrados na tela somente os pontos definidos na tabela de dados. O traçado **poligonal** mostra os dados interligando pontos consecutivos da tabela através de segmentos de reta e o traçado **interpolado** é realizado calculando-se uma função B-Spline a partir dos pontos da tabela e depois traçando-se a curva a partir dessa função.

O uso de diferentes tipos de traçados é de importância no sistema, pois dependendo do tipo de análise que o aluno deseje, há a conveniência de um ou outro traçado. Por exemplo, para análises de dispersão o aluno poderia selecionar o traçado por pontos, para o traçado de uma curva suave o uso de retas poligonais pode ser interessante e para curvas onde haja poucos pontos na tabela o traçado interpolado pode ser desejável.

c.4) Tipo de Escala

A função **Tipo de Escala** permite ao aluno selecionar escalas lineares ou logarítmicas para os eixos X e Y do gráfico. Esse recurso oferece ao usuário maior facilidade na análise de curvas de funções logarítmicas.

c.5) Variáveis para o Gráfico

No tratamento dos gráficos, será comum que o usuário tenha várias curvas (correspondentes a uma ou a várias tabelas) simultaneamente na tela. Num dado instante porém, é comum que ele deseje analisar o comportamento de apenas algumas delas, caso onde um número excessivo de curvas pode tornar desconfortável a análise. A função **Variáveis do Gráfico** permite que sejam ativadas ou desativadas uma a uma as tabelas e variáveis a serem mostradas nos gráficos. Essa opção é bastante útil na análise comparativa entre resultados de simulações e de medidas efetuadas.

c.6) Variável para o Eixo X

Como discutimos anteriormente, na leitura dos dados pelo STAG, a variável correspondente à primeira coluna de uma tabela será tomada como a variável para o eixo X e as demais para o eixo Y, tomadas portanto como função de X (ou como parâmetros). Pode ser desejável, ao longo das análises, traçar uma curva que correlacione duas variáveis que estejam no mesmo eixo. A função **Variável para o Eixo X** permite ao aluno trocar a variável do eixo X por qualquer outra variável do eixo Y pertencente à mesma tabela de dados. Esse recurso é de grande importância, dado que sem ele o usuário teria a necessidade de utilizar diversas tabelas para realizar essas mesmas operações.

c.7) Variáveis Paramétricas

A representação de funções de mais de uma variável será feita, conforme discutimos anteriormente, através de famílias de curvas. Para uma função de *n* variáveis, deverá ser feita a parametrização de *(n-1)* delas, definindo assim a família de curvas paramétricas.

A operação **variáveis paramétricas** permitirá que o usuário escolha, para uma determinada tabela de dados, quais as variáveis que devem ser tomadas como paramétricas para o traçado do gráfico.

c.8) *Zoom* e Visão Panorâmica (*Pan*)

A operação "**Zoom**" é subdividida em duas funções: **Ampliar** e **Reduzir**. A função **ampliar** permite ao usuário selecionar uma região de interesse específico do gráfico e visualizá-la na tela toda, tornando possível o estudo detalhado da região selecionada. É permitida a realização de diversas ampliações sucessivas, viabilizando assim a visualização do gráfico em diversos níveis de detalhamento. A função **reduzir** efetua a operação inversa da ampliação, retornando à visualização anterior à última ampliação. O STAG deverá ser capaz de suportar diversos níveis de *Zoom*.

Uma função complementar ao *Zoom* é a redefinição dos **limites para o gráfico**. Para o traçado das curvas, o STAG define automaticamente os limites inferiores e superiores para os eixos X e Y do gráfico, de maneira que as curvas sejam apresentadas ocupando a maior área possível na tela. Através desta função o usuário pode redefinir os limites do gráfico, podendo obter um efeito semelhante ao do *Zoom* ou ainda obter o efeito de visão panorâmica do gráfico.

c.9) Régua

Através da função **régua** o sistema permite que o aluno meça a distância entre dois pontos quaisquer do gráfico, além da inclinação da reta que passa por eles. Essa função pode ser útil, por exemplo, na determinação aproximada da derivada de uma curva, com a finalidade de estimação do valor de um parâmetro do dispositivo em estudo.

c.10) Imprimir Dados

A função **Imprimir** não diz respeito diretamente ao tratamento gráfico dos dados mas foi incluída neste grupo de funções. Por meio desta opção o usuário pode imprimir somente o subconjunto de dados que está atualmente sendo visualizado. Os dados referentes às variáveis marcadas como inativas e, caso o *zoom* esteja ativo, os dados que não pertencem à janela de *zoom* não serão impressos. A função imprimir permitirá ao aluno imprimir um *hardcopy* do gráfico mostrado na tela ou ainda uma tabela de dados correspondente aos pontos atualmente visualizados.

d) Operações de Análise dos Dados

As operações de análise dos dados devem permitir ao aluno a aplicação de diferentes métodos matemáticos e estatísticos sobre os dados em estudo, visando à análise dos resultados e à extração de parâmetros de dispositivos.

Para a versão inicial do STAG foi definido um conjunto considerado mínimo de operações de análise, o qual poderá ser ampliado conforme as necessidades verificadas com a utilização do sistema nos cursos. Dentre as operações a serem implementadas futuramente, destacam-se aquelas voltadas à extração de parâmetros de dispositivos e de dados para otimização de forma mais automática, incorporando, por exemplo, informações sobre diferentes modelos de dispositivos. A implementação deve preocupar-se com a modularidade, de forma a facilitar a ampliação das funções de análise.

As operações de análise dividem-se em três grupos:

- Regressões
- FFTs
- Estatísticas Básicas

Para quaisquer das análises acima definidas, o usuário poderá aplicar a operação a uma única curva por vez.

Os resultados das operações de análise serão apresentados graficamente na tela e, caso seja conveniente, poderão ser também visualizados em formato texto e/ou armazenados em disco, no arquivo de anotações.

Para a extração de parâmetros de dispositivos e de dados para otimização, o aluno deverá efetuar as análises e tratamentos dos gráficos e, em seguida registrar os valores obtidos no arquivo de anotações, através do editor de textos. Versões futuras do STAG deverão oferecer recursos para simplificar a extração de parâmetros e dados para a otimização, incorporando, por exemplo, definições de modelos de dispositivos.

d.1) Regressões

As operações de regressões permitem que, a partir de um determinado conjunto de pontos de uma curva, sejam realizadas extrapolações que podem, por exemplo, estimar valores de parâmetros dos dispositivos sob estudo.

A figura 16 apresenta um exemplo da tela obtida após a realização de uma operação de regressão.

Para as operações de regressões estarão disponíveis quatro métodos básicos:

- Regressões Lineares
- Regressões Polinomiais
- Regressões Geométricas
- Regressões Exponenciais

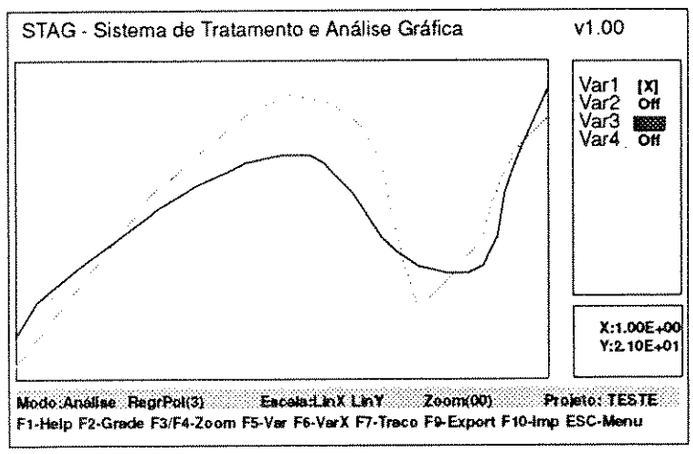


Figura 16 - Exemplo de Tela com Resultados de Regressões

Para qualquer uma das regressões, após terem sido efetuados os cálculos, o sistema mostrará na tela a curva com os valores tabulados da variável escolhida, além da curva obtida através da regressão. Opcionalmente, o aluno poderá verificar as equações que definem as curvas calculadas através das regressões, sendo possível registrar, de forma automática, esses resultados no arquivo de anotações.

A regressão linear ajusta as coordenadas da curva escolhida pelo aluno através da equação de uma reta. O método aplicado para o ajuste da curva é o dos mínimos quadrados.

A regressão polinomial realiza o ajuste da curva por um polinômio de grau **n**. Ao selecionar este tipo de regressão o usuário deverá definir o grau do polinômio a ser calculado. Na versão inicial o grau de polinômio deverá ser menor ou igual a 10.

Na regressão geométrica, o método aplicado é também o dos mínimos quadrados e o ajuste da curva é realizado segundo uma equação do tipo:

$$y = \mathbf{A}.x^{\mathbf{B}}$$

onde

A e B são os coeficientes a serem determinados pela regressão.

Para a regressão exponencial, será calculada a função exponencial que melhor aproxima os pontos da curva escolhida pelo usuário. A função exponencial pode ser descrita pela equação:

$$y = A.e^{(B.X)}$$

onde

e é a base neperiana e A e B são os coeficientes que definem a exponencial.

d.2) Transformadas de Fourier (FFTs)

As análises pelas Transformadas de Rápidas de Fourier (FFTs) permitem que, a partir de uma sequência de amostras de um sinal analógico discretizado, esse sinal seja decomposto em função de um conjunto de sinais senoidais com frequência e fase definidos e tais que a somatória deles seja equivalente ao sinal analógico tomado inicialmente.

O STAG permitirá ao aluno o cálculo de FFT direta ou inversa de um sinal. Os resultados da FFT são mostrados graficamente na tela e, caso o usuário deseje, poderão ser automaticamente registrados, em formato texto, no arquivo de anotações do projeto.

Versões futuras do STAG deverão permitir um maior número de cálculos associados às FFTs, como convolução e correlação.

A figura 17 apresenta o formato geral da tela em que serão apresentados os resultados gráficos dos cálculos das FFTs.

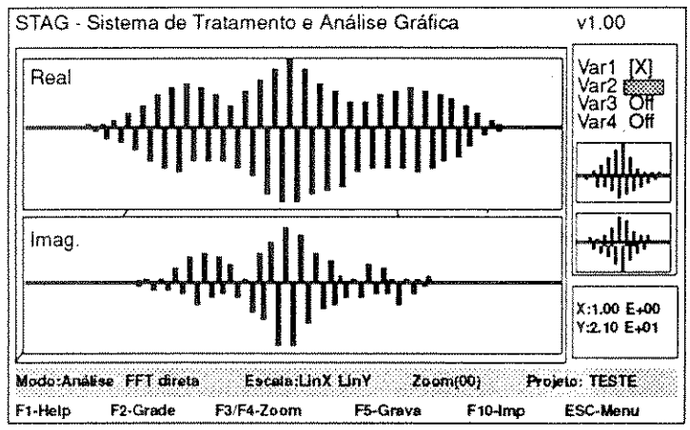


Figura 17 - Exemplo de Tela com Resultados de FFTs

d.3) Operações Estatísticas

As análises estatísticas permitem ao aluno realizar um conjunto de operações estatísticas básicas, visando à realização de medidas de tendência central e de dispersão das amostras em estudo.

Na versão atual foi estabelecido um conjunto mínimo de operações estatísticas, para distribuições gaussianas. São elas:

- Média
- Moda
- Mediana
- Desvio Médio
- Desvio Padrão
- Variância
- Coeficiente de Variação
- Distribuição de Frequências

Ao ser selecionada a opção de análises estatísticas, o sistema calculará automaticamente todos os valores descritos acima, mostrando graficamente na tela os valores da média e da mediana.

O usuário poderá selecionar também a visualização de um gráfico de barras (histograma) mostrando a distribuição de frequências da variável sob análise. Ele poderá definir os limites inferior e superior para o gráfico, bem como o número de intervalos (número de barras no gráfico).

Em versões futuras poderão, por exemplo, estar disponíveis funções para cálculos de médias e desvios entre diversas curvas e análises de dispersão mais sofisticadas. Poderão ser realizados também cálculos estatísticos com base em outros tipos de distribuição, através da aplicação de outros métodos, como por exemplo o de Weibull [Wei51], que é amplamente utilizado na área de qualidade.

As figuras 18 e 19 apresentam respectivamente os resultados gráficos e numéricos (em formato textual) das análises estatísticas básicas. A figura 20 apresenta um exemplo do gráfico de distribuição de frequências.

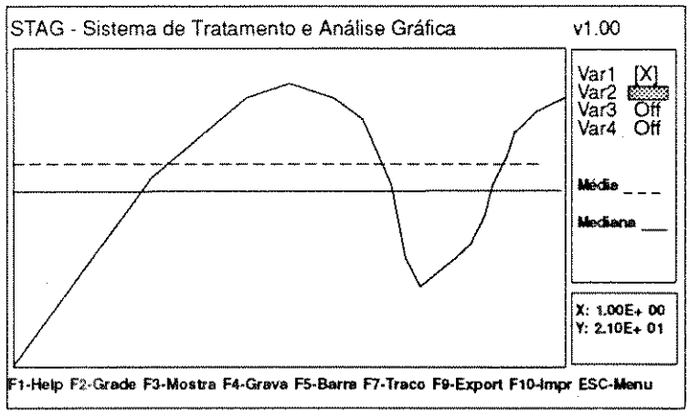


Figura 18 - Resultados Gráficos das Estatísticas Básicas

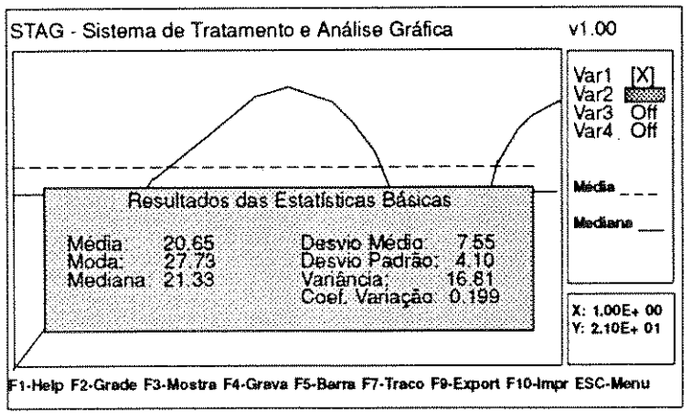


Figura 19 - Resultados Numéricos das Estatísticas Básicas

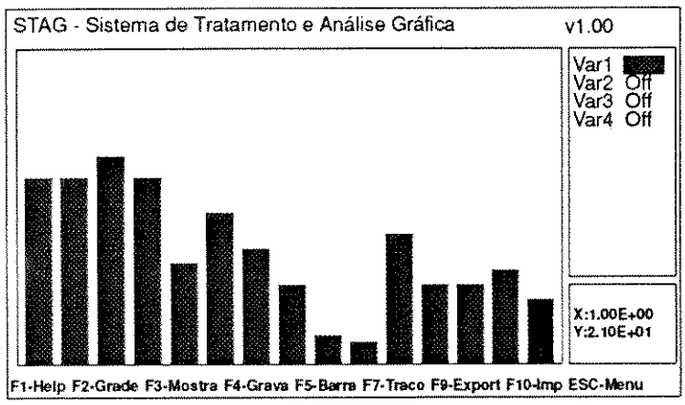


Figura 20 - Gráfico de Distribuição de Frequências

e) Operações de Configuração do Sistema

As operações de configuração permitem ao usuário redefinir diversas características ambientais do STAG, conferindo assim uma maior flexibilidade de uso do sistema.

O STAG deverá oferecer as seguintes opções de configuração:

- Configuração de Cores
- Configuração de Idioma
- Configuração do Editor de Textos
- Configuração de Periféricos

A **configuração de cores** permite que o aluno defina as cores a serem utilizadas em tela pelo sistema. O usuário poderá selecionar e modificar individualmente as cores utilizadas para cabeçalhos, rodapés, mensagens de erro e advertência, para as variáveis e legendas do gráfico, para os menus, etc...

A operação **configuração de idioma** deverá permitir que seja selecionado o idioma a ser utilizado pelo sistema para todas as mensagens por ele emitidas. Esta operação é de importância em um sistema que pretende ser utilizado em países de diferentes línguas.

Através da operação de **configuração do editor de textos**, o usuário poderá definir qual o editor de textos que será utilizado pelo sistema, para a edição do arquivo de anotações. Será utilizado como *default* do sistema o mesmo editor de textos disponível no SDP. Deve-se ressaltar que, dependendo do editor de textos escolhido, o usuário pode ter problemas de falta de memória e, nesse caso, a solução será a utilização do editor *default* do sistema.

A **configuração de periféricos** deverá permitir ao aluno definir quais os tipos e modelos dos periféricos disponíveis para uso no sistema. Dependendo do tipo de periférico, o aluno deverá definir também um conjunto adicional de características do mesmo, como por exemplo o dispositivo lógico de entrada e saída utilizado, velocidade de comunicação da interface, etc... Inicialmente o usuário terá disponivel opções para a configuração de impressoras apenas.

Quaisquer alterações realizadas na configuração do sistema serão válidas para a atual sessão de uso do sistema. No instante em que o usuário for sair do sistema, se houver ocorrido alguma modificação de configuração, o sistema deverá perguntar a ele se deseja ou não salvar as alterações.

IV.2.1.2 - Bases de Dados

O projeto de dados de um sistema é um ponto de grande importância na sua definição, estando estreitamente ligado aos aspectos de sua eficiência e consistência, influenciando diretamente a qualidade global do sistema.

Descreveremos, nas seções seguintes, as bases de dados utilizadas e geradas pelo STAG. As principais estruturas de dados utilizadas para a representação interna dos dados serão descritas nas seções referentes à implementação do sistema.

As bases de dados utilizadas como entradas para o STAG compreendem basicamente dados gerados como saídas dos testes e simulações analógicos. A partir desse dados, serão geradas bases de dados de saída contendo resultados das análises e que poderão ser usados em outras etapas da validação do circuito. Teremos ainda uma terceira classe de bases de dados que será de uso exclusivo do STAG (que chamaremos de dados proprietários). Neste último grupo estarão presentes, por exemplo, os dados de configuração do sistema. A figura 21 mostra um diagrama representando o relacionamento do STAG com suas bases de dados.

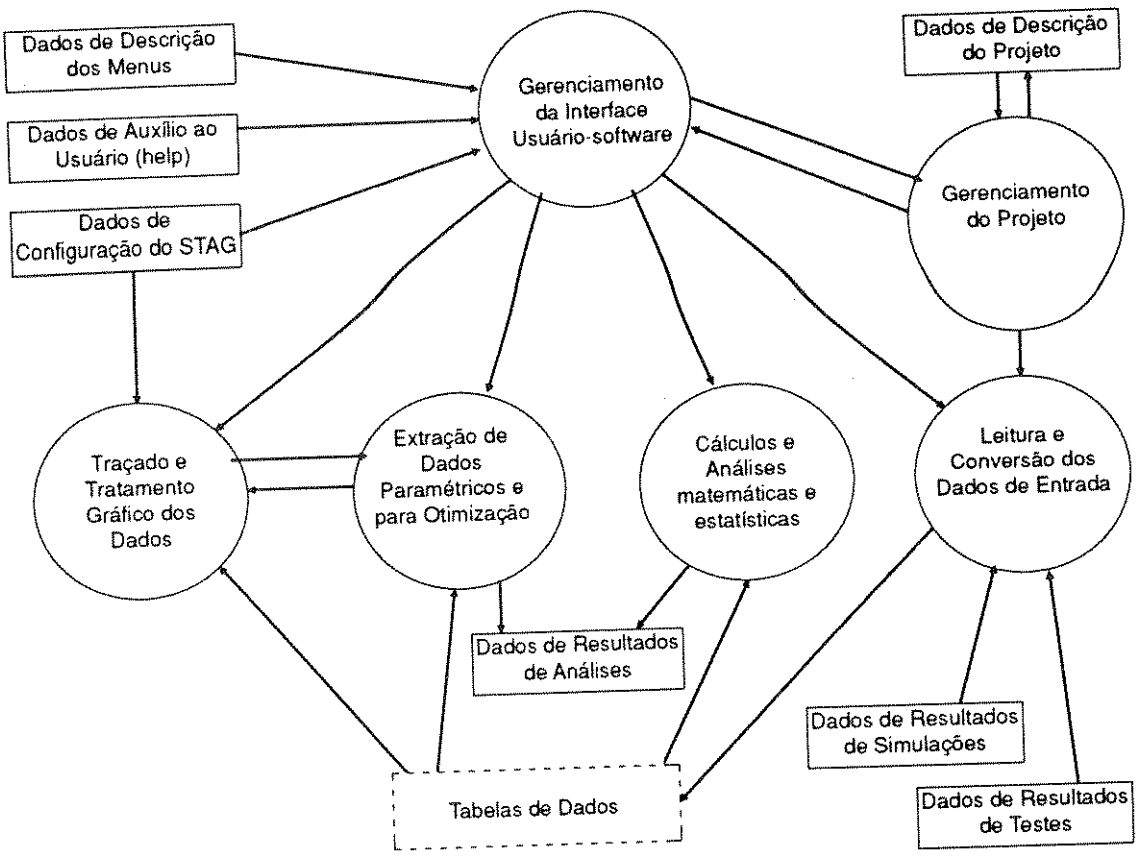


Figura 21 - Relacionamentos do STAG com suas Bases de Dados

As bases de dados utilizadas pelo STAG serão compostas por um ou mais arquivos, que estarão armazenados em disco.

a) Bases de Dados de Entrada

As bases de dados tomadas como entrada pelo STAG compreendem dados de resultados de simulações, gerados pelo simulador SPICE, dados de resultados de testes, gerados pelo SAD e, eventualmente, tabelas de dados geradas por outras ferramentas externas ao SAD e ao SDP, desde que esses dados obedeçam ao formato descrito no anexo I.

Os dados dos arquivos de entrada de interessse para o STAG serão basicamente um conjunto de tabelas de dados que descrevem valores de variáveis de uma simulação ou medição. Cada tabela de dados deverá relacionar valores de no mínimo duas e no máximo dez variáveis e poderá conter até 1024 valores para cada variável.

Para arquivos gerados pelo SPICE, além das tabelas de dados propriamente ditas, o arquivo conterá também informações sobre a descrição do circuito, das análises a serem aplicadas e de outros resultados de simulações que não serão de nosso interesse. Para esses arquivos, o STAG é capaz de verificar automaticamente as análises (AC, DC, de transientes, de distorção ou de ruídos) para as quais foram geradas as tabelas de dados. As tabelas de dados capturadas pelo STAG serão sempre resultantes de comandos **.PRINT** definidos no programa do SPICE.

Para os arquivos em formato compatível com o SAD teremos somente as tabelas de resultados, o que facilita o tratamento por parte do STAG.

b) Bases de Dados de Saída

Os dados gerados como saída pelo STAG podem ser classificados em dois grupos: dados de anotações de projeto e tabelas de dados.

Para um determinado projeto, os dados de notas de projeto estarão armazenados num único arquivo, em formato texto. O arquivo de anotações poderá conter informações geradas automaticamente pelo STAG, como é o caso de resultados de análises (parâmetros derivados da aplicação de regressões, FFTs ou estatísticas), e informações digitadas manualmente pelo usuário, através do editor de textos. O objetivo do arquivo de anotações é permitir ao aluno manter um registro histórico das informações extraídas através do tratamento e análise dos gráficos.

O STAG permite também que sejam geradas tabelas de dados com os valores correspondentes às coordenadas das curvas apresentadas na tela. As tabelas de dados são geradas em formato texto, sendo compatíveis com os dados de entrada do STAG. As tabelas de dados serão geradas sob comando do operador e é ele quem definirá os nomes dos arquivos para as mesmas.

c) Bases de Dados Proprietárias

As bases de dados consideradas proprietárias agrupam os dados que são de uso exclusivo do STAG. Essas bases de dados são voltadas basicamente à organização e flexibilidade do sistema.

Incluem-se aqui as bases de dados de descrição do projeto, os dados de configuração do sistema, dados de descrição dos menus de opções e os dados de auxílio ao usuário (*help*). O formato dos arquivos descritos nesta seção é apresentado em detalhes no anexo I.

A base de dados de descrição do projeto estará armazenada num arquivo texto e conterá as informações sobre os arquivos de dados e o arquivo de anotações associados ao projeto. Para cada projeto definido pelo usuário teremos um arquivo de descrição do projeto.

Na base de dados de configuração do sistema estarão descritas todas as informações necessárias para o sistema acerca de sua configuração atual, como as cores a serem utilizadas para a tela, o idioma, os dispositivos de entrada e saída disponíveis, etc... Esta base de dados estará contida num único arquivo, o qual poderá ser modificado pelo usuário através das operações de configuração do STAG.

A base de dados de *help* será composta por um arquivo em disco para cada idioma disponível ao sistema. No arquivo de *help* estarão armazenadas, em formato específico, todas as telas de auxílio ao usuário disponíveis para o STAG. Este arquivo será utilizado pelo STAG somente para leitura. A alteração dos dados contidos neste arquivo será realizada através de um utilitário desenvolvido especificamente para esta finalidade [Aff91]. No próprio arquivo estarão descritas também todas as informações referentes às palavras-chave e às ligações com outras páginas (*hotlinks*), que estabelecem os diversos contextos de *help* e a estrutura do hipertexto.

A maneira como foram estruturados os arquivos de *help* do sistema teve forte influência de alguns aspectos voltados à manutenção e evolução do sistema. O fato de todas as informações que descrevem o hipertexto estarem descritas no mesmo arquivo possibilitam que a alteração dos textos que descrevem determinados contextos do *help* e, em alguns casos até a criação de novos contextos, sejam feitas sem a necessidade de modificação e recompilação do código do STAG.

A base de dados de descrição dos menus será composta por um arquivo para cada idioma disponível no sistema. Cada arquivo conterá todos os dados necessários à descrição dos menus, incluindo as mensagens referentes a cada opção, o número de subopções, o contexto de help associado, etc... A exemplo do que ocorre com os arquivos de *help*, diversas alterações nos menus podem eventualmente ser realizadas sem a necessidade de alteração e recompilação do código do sistema.

O modo como foram estruturadas as bases de dados para o *help* e para a descrição das estruturas dos menus trazem consigo ainda a possibilidade de que se acrescente um novo idioma ao sistema praticamente sem a necessidade de modificações do código, reduzindo enormemente os esforços envolvidos nessa tarefa e dispensando a necessidade da alocação de um programador que tenha algum conhecimento do sistema para essa tarefa.

IV.2.2 - Requisitos de Interface

IV.2.2.1 - Interface Usuário-Software

Como discutimos anteriormente, o caráter didático proposto para o sistema exige uma interface usuário-software que, além de propiciar ao aluno a facilidade de uso, seja capaz de direcioná-lo durante a utilização do sistema. Ainda, deverão estar disponíveis, a qualquer instante, informações que o auxiliem, tanto no que diz respeito à operação do sistema quanto no que se refere a conceitos envolvidos nas suas atividades.

O STAG utilizará menus de opções como o mecanismo básico de interação do sistema com o usuário. Através dos menus, o aluno poderá selecionar quaisquer opções disponíveis no sistema. A organização dos menus deverá agrupar as operações de forma a facilitar o direcionamento das atividades do aluno. Uma forma alternativa de interação com o sistema - aplicável principalmene a usuários já habituados ao uso do STAG - deverá ser feita através de teclas especiais (*hot-keys*). Através das *hot-keys* o aluno poderá ter acesso direto à maioria das funções disponíveis no STAG, sem a necessidade da navegação nos menus. O mecanismo de menus de opções levou em consideração o aspecto de uniformidade com outras ferramentas do SAD e do SDP, tendo funcionamento semelhante às demais ferramentas.

Deverão estar disponíveis a qualquer instante, informações de auxílio ao usuário (*help*), através da tecla <F1>. O sistema de auxílio ao usuário deverá ser implementado na forma de hipertexto, a partir das ferramentas propostas em [Aff91]. Dessa forma, o aluno poderá "caminhar" através das telas de *help* do sistema tendo, a partir de uma determinada tela, acesso a outras telas referentes a tópicos relacionados ao contexto atual. As informações de auxílio deverão abranger não somente aspectos da operação do sistema, mas também informações referentes a conceitos envolvidos nas atividades de análise. Teclando <Alt-F1>, o aluno poderá obter informações sobre as teclas especiais (*hot-keys*) disponíveis naquele momento.

A capacidade de configuração de diferentes idiomas é também uma característica desejável à interface usuário-software, considerando-se a possibilidade de disseminação do sistema em outros países. Deverá ser prevista a configuração de idiomas de duas formas diferentes: interativamente, durante uma sessão de uso do sistema ou durante o processo de instalação do sistema.

IV.2.2.2 - Interfaces com outros Sistemas

Como discutimos anteriormente, o STAG deve ser capaz de interpretar dados gerados como resultados das simulações pelo SPICE e dados gerados como resultados de testes analógicos e paramétricos através do próprio SAD. O STAG deverá implementar um módulo dedicado especialmente à interpretação e conversão desses dados para o formato interno de armazenamento. Esse módulo poderá futuramente ser ampliado, de forma a suportar dados em outros formatos conhecidos.

Também em versões futuras do sistema, dados de parâmetros de dispositivos e dados para otimização poderão ser gerados em formatos compatíveis com os programas do SPICE, melhorando o nível de integração das atividades do aluno.

IV.2.2.3 - Interfaces de *Hardware*

As interfaces de *hardware* previstas para o STAG dizem respeito somente à utilização de diferentes tipos de periféricos de entrada e saída.

Deverão estar disponíveis, no mínimo, uma saída paralela e uma saída serial (RS-232).

Inicialmente o STAG deverá prover suporte a *mouse* e impressoras. Futuramente poderão ser implementadas funções adicionais para dar suporte a outros dispositivos, como mesas digitalizadoras, canetas óticas, traçadores gráficos, etc...

IV.3 - Descrição da Implementação

IV.3.1 - Módulos de Implementação

Na definição dos módulos de implementação do STAG tomou-se como principais critérios a obtenção de independência funcional entre os diversos módulos, a criação de biblioteca de rotinas básicas comuns a eles e a testabilidade das funções de cada módulo. Buscou-se com isso otimizar aspectos de estruturação do sistema e, consequentemente, da facilidade de manutenção e ampliação do STAG.

Em especial, a criação de bibliotecas de rotinas comuns foi de grande utilidade, facilitando a programação e permitindo uma melhor padronização do código e das interfaces com o usuário. Ainda, o código contido nessas bibliotecas é totalmente reutilizável.

O STAG foi particionado nos seguintes módulos de implementação:

- **STAG** - É o módulo principal do sistema. Realiza a integração dos diversos módulos de implementação. É também o responsável pelo traçado das curvas.
- **GSARQ** - Implementa as rotinas de leitura dos dados de projeto e interpretação dos arquivos de entrada, de gravação de dados e de gravação das anotações automáticas realizadas pelo STAG.
- **GSESTAT** - Este módulo contém as rotinas para os cálculos estatísticos e das análises de regressões.
- **GSFFTB2** - Contém as rotinas para cálculo das transformadas rápidas de Fourier (FFTs). As rotinas básicas deste módulo foram adaptadas a partir das rotinas do *Turbo Numerical Methods Toolbox* [Bor87].
- **GSTELA** - Contém rotinas básicas para tratamento dos gráficos e para apresentação de resultados de análises em tela.
- **BGGRAF** - Implementa a biblioteca de rotinas básicas de visualização gráfica, incluindo também tratamento de alto nível do gerenciamento de menus.
- **BGVARS** - Contém as declarações e rotinas de inicialização das variáveis globais da biblioteca de rotinas básicas (BGGRAF).
- **BGEXCMNU** - Implementa o tratamento de médio nível de gerenciamento dos menus de opções.
- **GSMENU** - Contém as rotinas associadas à ativação das opções que são folhas na árvore de menus.
- **GSVARS** - Contém a declaração e inicialização de variáveis globais do STAG. É utilizado por todos os módulos do STAG, exceto aqueles que implementam bibliotecas de rotinas comuns.
- **BGHELP** - Implementa as rotinas básicas de gerenciamento e ativação do sistema de auxílio ao usuário.

IV.3.2 - Organização da Interface Usuário-Software

As telas são, para o STAG, o principal meio de interface com o usuário. Elas são organizadas de forma a definir um conjunto de regiões, conforme mostrado na figura 22. O posicionamento das informações na tela segundo a definição das regiões busca oferecer ao usuário maior conforto durante a utilização do sistema, evitando que este precise "procurar" determinadas informações na tela toda.

Descrevemos a seguir cada uma das regiões da tela.

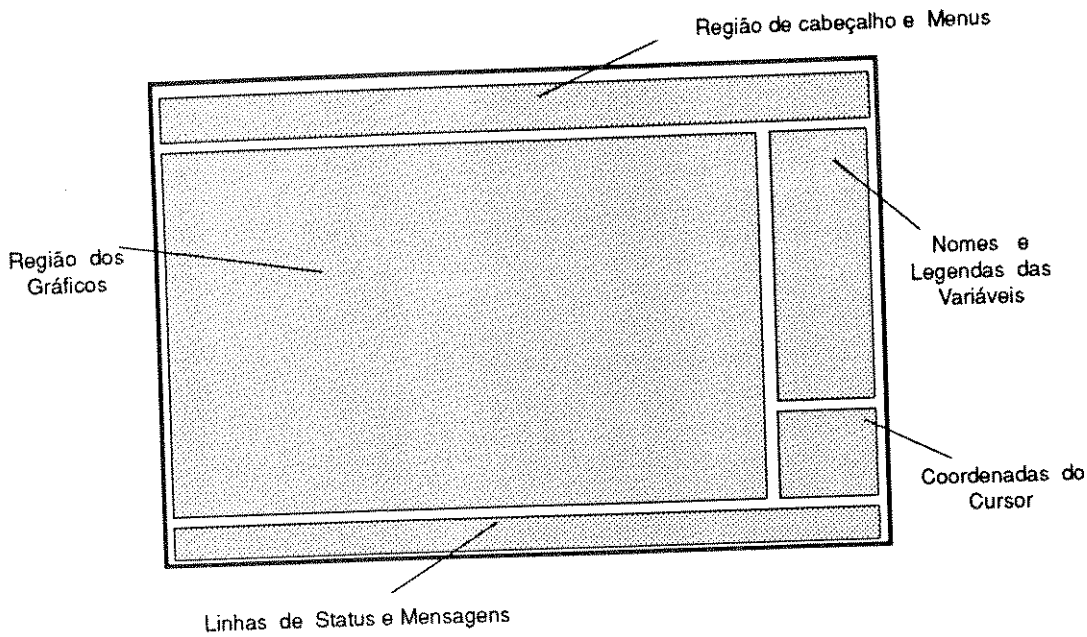


Figura 22 - Regiões da Tela para o STAG

- **Região dos Gráficos:** Nesta região serão mostrados os resultados gráficos dos tratamentos e análises aplicados aos dados em estudo. O usuário terá disponível nesta região um cursor, através do qual ele poderá realizar a manipulação das curvas. Nesta região serão apresentadas também as informações de auxílio ao usuário (*help*), quando solicitadas.

- **Região de Cabeçalho e Menus:** Nesta região da tela é apresentado ao usuário o menu principal do STAG, a partir do qual ele terá acesso às diversas operações do sistema. O menu principal não permanece presente na tela o tempo todo. Para ativar os menus o usuário deve, estando na região dos gráficos, pressionar a tecla <ESC >. Para abandonar os menus, retornando à região dos gráficos, deve-se teclar <ESC> a partir do menu principal do sistema.

- **Região das Linhas de Status e Mensagens:** Nesta região o sistema apresenta ao usuário mensagens explicativas e/ou de advertência. É apresentada também uma linha mostrando o estado operacional do sistema, com informações como o nome do projeto, o tipo de escalas dos gráficos (linear ou logarítmica), nível de **Zoom**, tipos de escalas, etc...

- **Região de Nomes e Legendas das Variáveis:** Serão mostradas nesta região os nomes das variáveis atualmente em estudo, assim como as suas respectivas legendas. Entre duas tabelas de dados é utilizado um separador, na forma de linha horizontal, para facilitar ao usuário a identificação das tabelas. Para as variáveis que forem desativadas pelo operador, ao invés da legenda será mostrada a palavra *OFF*. As variáveis do eixo X serão sempre a primeira variável da tabela e serão identificadas pelo texto *"[X]"* após o nome da variável. Variáveis paramétricas serão identificadas pela palavra *"Param"* após o nome da variável. Algumas operações permitirão que o usuário selecione variáveis desta lista como se estivesse utilizando um menu de opções.

- **Região das Coordenadas do Cursor:** Esta região apresentará ao usuário os valores das coordenadas X e Y correspondentes à posição atual do cursor no gráfico.

IV.3.3 - Estruturas de Dados

Descreveremos nesta seção as principais estruturas de dados utilizadas para a organização das informações no STAG. Utilizaremos um pseudo-código semelhante ao do Pascal para a descrição dos campos de dados de cada estrutura e, quando conveniente, usaremos também exemplos e diagramas.

a) Estrutura de Dados para os Menus

A estrutura de dados dos menus representa a árvore de opções dos menus e características particulares a cada uma das opções.

```
Const
        MAXOPCMENU = 80;

Type
        RegMenu = Record
                Msg: String[40];
                NOpc: Integer;
                Ind: Integer;
                Prior: Integer;
                Hlp: Integer
                end;
Var
        Menu: Array[1..MAXOPCMENU] of RegMenu:
```

Na estrutura acima o campo *Msg* contém o texto de uma opção do menu.

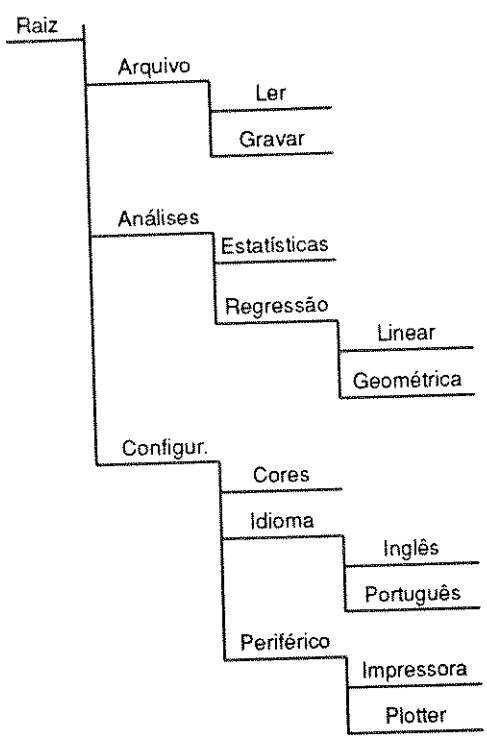
O campo *NOpc* representa o número de sub-opções para essa opção, sendo válidos os valores entre 0 e 16. Caso o valor seja zero, isso significa que a opção é uma folha na árvore de menus.

O campo *Ind* indica qual a posição no vetor onde se iniciam as sub-opções (caso o valor do campo *NOpc* seja maior que zero) ou o procedimento a ser executado (caso *NOpc* tenha o valor 0). São válidos quaisquer valores inteiros não nulos entre -32000 e MAXOPCMENU, sendo que valores negativos indicam um procedimento a ser executado pelo programa. Note-se que as opções referentes a um mesmo menu deverão estar em posições contíguas no vetor *Menu*.

O campo *Prior* designa qual a prioridade de acesso mínima para que o usuário tenha acesso a esta opção. Na versão atual este campo não tem utilidade, mas prevê, para versões futuras, a criação de opções que tenham acesso restrito a determinados grupos de usuários.

O campo *Hlp* estabelece qual o contexto de *help* associado à opção do menu.

Ilustraremos a seguir um exemplo simples de uma árvore de opções e da estrutura de dados segundo a qual ela pode ser descrita.



Msg	NOpc	Ind	Prior	Hlp
Raiz	3	2	0	1
Arquivo	2	5	0	2
Análises	2	7	0	5
Configuração	3	9	0	10
Ler	0	-11	0	3
Gravar	0	-12	0	4
Estatísticas	0	-21	0	6
Regressões	2	12	0	7
Cores	0	-31	0	21
Idioma	2	14	0	22
Periféricos	2	16	0	23
Linear	0	-221	0	8
Geométrica	0	-222	0	9
Inglês	0	-321	0	11
Português	0	-322	0	12
Impresora	0	-331	0	15
Plotter	0	-332	0	16

b) Estruturas para o Sistema de *Help*

Cada contexto do sistema de auxílio ao usuário (*help*) define um texto explicativo sobre um determinado tópico do sistema, bem como suas ligações com outros contextos. As estruturas de dados que definem cada contexto de *help* são descritas a seguir.

```
Const
        MaxLinHlp  = 12;
        MaxColHlp  = 50;

Type
        RegHelp= Record
             LinhaHelp: Array[1..MaxLinHlp] of String[MaxColHlp+30];
             end;
```

A estrutura *RegHelp* define um registro de dados correspondente a um contexto de *help* de 12 linhas de 50 colunas cada. No vetor *LinhaHelp*, cada *string* é definida com um espaço adicional (+30), a fim de permitir que os *hot-links* estejam definidos juntamente com o texto.

O texto de *help*, na versão atual, será composto apenas por caracteres ASCII padrão (i.e., caracteres de 7 bits). As palavras que definem *hot-links* são marcadas somando-se 128 ao código ASCII de cada caracter da palavra (ou seja, colocando o bit mais significativo em "1"). O contexto "apontado" por cada palavra marcada como *hot-link* é definido nos primeiros bytes imediatamente após o final do texto da linha. Para cada *hot-link* são utilizados dois bytes. O primeiro byte tem o valor NULO e o segundo byte indica o número do contexto associado ao *hot-link*. Considerando-se que cada linha tem um espaço adicional de 30 bytes, poderíamos definir então 15 *hot-links* para cada linha de uma tela de *help*.

Em tempo de execução do sistema, as informações de *help* terão somente acesso para leitura. A criação e manutenção dos sistemas de *help* é feita através de utilitários desenvolvidos especificamente para essa finalidade.

Os anexos I e II apresentam informações complementares a respeito do sistema de *help*.

c) Estruturas para Descrição das Variáveis do Gráfico

Cada variável para o gráfico é descrita por um vetor contendo um conjunto de valores lidos da tabela de entrada e um conjunto de atributos que podem ser modificados pelo usuário durante os tratamentos e análises.

As estruturas que descrevem uma variável para o gráfico são mostradas a seguir.

```
Const
        MAX_TAM_VET = 1023;     { Número máximo de pontos para 1 variável    }
        MAX_VET_GR = 20;        { Número máximo de variáveis simultâneas     }

Type

        VetPtos = Array [0..MAX_TAM_VET] of Real;

        VarGrafico = Record
              V: VetPtos;
              EhParam: Boolean;
              NomVar: String[8];
              Ativa: Boolean;
              end;
```

A estrutura *VarGrafico* descreve uma variável. Nela, o campo *V* descreve o vetor dos valores para a variável, o campo *EhParam* identifica se a variável está atualmente configurada como um parâmetro ou não, o campo *NomVar* define o nome da variável e o campo *Ativa* identifica se a variável está ativa para visualização ou não.

d) Estruturas para Descrição das Tabelas de Dados.

Cada uma das tabelas de dados lidas descreve uma variável de varredura (a variável do eixo X) e uma ou mais variáveis para o eixo Y. Essas tabelas serão descritas pela seguinte estrutura:

```
Const  MAXVARY = 10;
Type
        PtrVarGraf = ^VarGrafico;
        DscTabela = Record
              NomTab: String[40];
              VarX: PtrVarGraf;
              VarY: Array[1..MAXVARY] of PtrVarGraf;
              TabAtiva: Boolean;
              TotPtos: Integer;
              Familia: Byte
              end;
```

Na estrutura *DscTabela* o campo *NomTab* conterá um nome lógico, definido pelo usuário, para a tabela. Os campos *VarX* e *VarY* serão apontadores para descritores de variáveis para o eixo X (uma única variável) e Y (até 10 variáveis) respectivamente. O campo *TabAtiva* identifica se a tabela está ou não ativa para a visualização. O campo *TotPtos* contém o número de linhas da tabela (ou seja, o número de valores para cada variável). O campo *Familia* descreve o número da familia de curvas (usado internamente pelo STAG).

e) Estruturas para Descrição de Arquivos de Dados

A descrição dos arquivos de dados deverá identificar o nome do arquivo, seu formato (SPICE ou SAD) e as tabelas de dados nele contidas, observada a restrição do número máximo de dez tabelas de dados por arquivo. A estrutura de dados associada será a seguinte:

```
Const
      MAXTABARQ = 10;   { Número máximo de tabelas de dados num arquivo }
Type
      DscArqDados = Record
            NomArq: String[32];
            Arq: Text;
            TipoArq: Byte;
            Ativo: Boolean;
            TabDad: Array[1..MAXTABARQ] of DscTabela;
            Ativo: Boolean
            end;
```

O campo *NomArq* conterá o *pathlist* completo do arquivo (diretório e nome do arquivo). O campo *TipoArq* identificará o formato do arquivo (1=SAD, 2=SPICE). O vetor *TabDados* conterá as descrições de cada uma das tabelas de dados contidas no arquivo. O campo *Ativo* indica se o arquivo está atualmente ativo ou não.

f) Estruturas para a Descrição do Projeto

A descrição do projeto associará um conjunto de arquivos de dados e um arquivo de anotações a um nome de projeto, conforme já foi discutido anteriormente. A estrutura de dados para a descrição do projeto é definida a seguir.

```
Const
        MAXARQENTR = 10;
Type
        DscProjeto = Record
                NomProj: String[32];
                ArqDad: Array[1..MAXARQENTR] of DscArqDados;
                NomArqNotas: String[32];
                ArqNotas: Text;
                end;
```

O campo *NomProj* define o nome do projeto. O vetor *ArqDad* contém as estruturas que descrevem os arquivos de dados e o campo *NomArqNotas* contém o nome do arquivo de anotações para o projeto. O campo *ArqNotas* contém o descritor do arquivo de anotações.

g) Estruturas para Descrição de Curvas

Cada curva a ser traçada na tela será descrita através da associação entre um subconjunto dos valores de duas variáveis pertencentes a uma mesma tabela de dados. A estrutura de dados utilizada para descrever uma curva é descrita a seguir.

```
DscCurva = Record
        X: PtrVargraf;
        Y: PtrVarGraf;
        PInic: Integer;
        PFim: Integer;
        Cor: Byte;
        Legenda: Byte;
        end;
```

Na estrutura de dados acima, os campos *X* e *Y* são apontadores para os descritores de duas variáveis pertencentes a uma mesma tabela, correspondendo respectivamente às variáveis para os eixos X e Y. Os campos *PInic* e *PFim* definem as posições inicial e final no vetor de descrição das variáveis. Os campos *Cor* e *Legenda* definem respectivamente a cor e o tipo de legenda a serem utilizados para o traçado da curva.

h) Estruturas para Descrição de Famílias de Curvas

As famílias de curvas são obtidas através da parametrização de funções de mais de uma variável, de forma que essas possam ser representadas através de um conjunto de curvas que descrevam funções de uma só variável. A estrutura de dados usada para representar uma família de curvas é descrita a seguir.

```
Const
        MAXCURVASFAM = 20;
        MAXPARAM = 5;
Type
        DscFamilia = Record
                Existe: Boolean;
                Curv: Array [1..MAXCURVASFAM] of DscCurva;
                Param: Array [1..MAXPARAM] of ^VarGrafico;
                end;
```

Na estrutura acima o campo *Curv* é um vetor onde cada posição contém o descritor de uma curva. O campo *Param* é um vetor de apontadores para as variáveis definidas como parâmetros para a família de curvas. O campo *Existe* identifica se a estrutura contém ou não a descrição de uma família de curvas.

i) Interrelacionamentos entre Estruturas

As estruturas de dados descritas nos ítens c) até h) são todas estruturas voltadas à representação interna dos dados ligados à descrição das curvas a serem traçadas e manipuladas pelo sistema e mantêm entre si fortes relacionamentos.

A figura 23 mostra um diagrama da organização e dos relacionamentos das estruturas de dados voltadas à definição das curvas.

IV.3.4 - Métodos e Algoritmos

Esta seção descreve os principais métodos e algoritmos utilizados na versão atual do STAG. No anexo V apresentaremos as listagens das principais rotinas que implementam esses algoritmos e métodos.

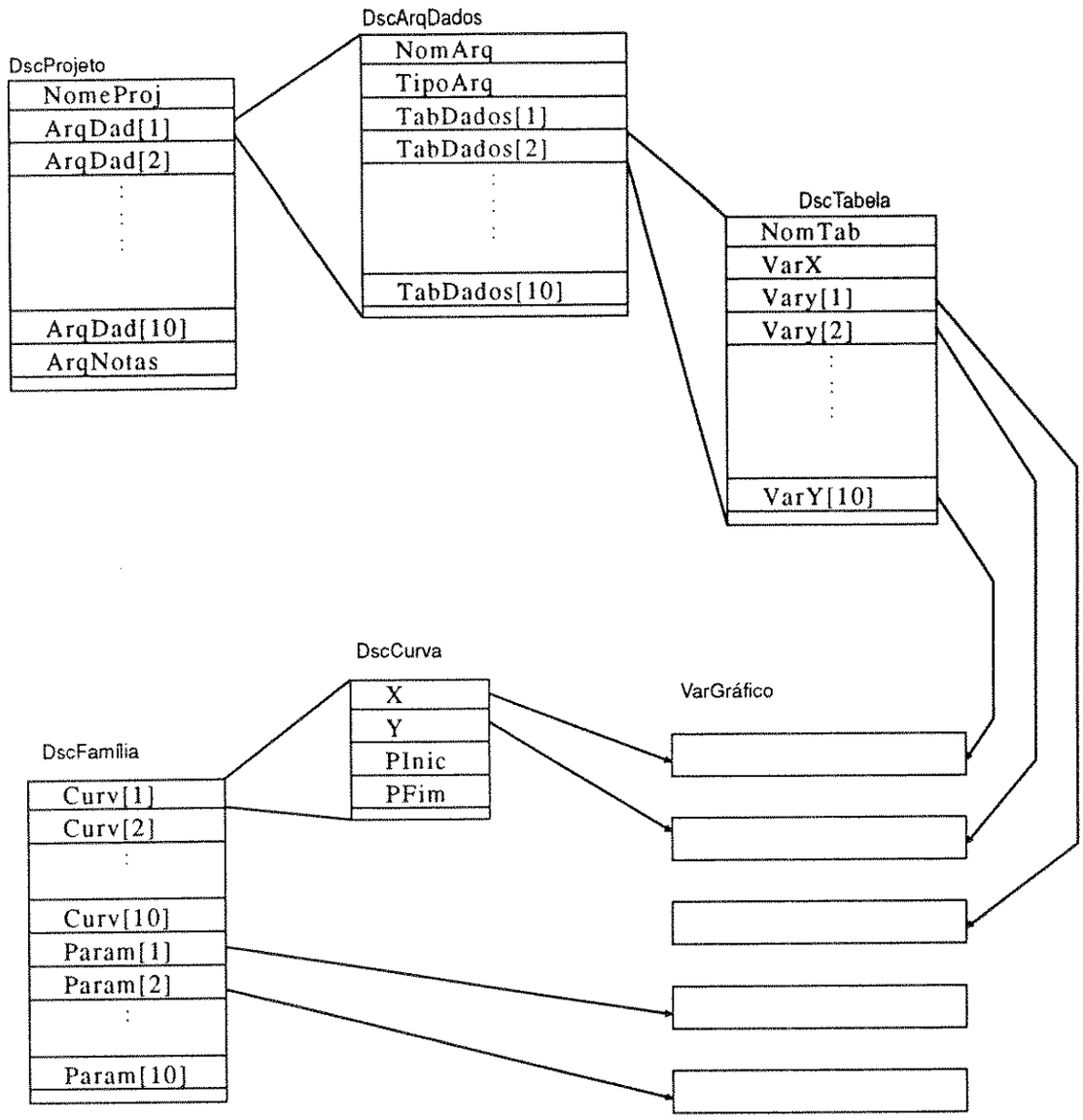


Figura 23 - Relacionamento entre Estruturas para Curvas

a) Leitura e Interpretação dos Arquivos de Entrada

A leitura e interpretação de dados de entrada é bastante simples, dado que os arquivos de entradas estão em formato ASCII e na forma de tabelas. Os arquivos gerados pelo SAD contém simplesmente as tabelas de dados. Os arquivos gerados como saída do SPICE, contêm informações adicionais, de descrição do circuito e das simulações.

Para os arquivos gerados pelo SPICE, o STAG realiza inicialmente a leitura do programa da simulação, identificando os comandos **.PRINT** nele contidos. Em seguida, ele apresenta ao usuário um menu de opções contendo os nomes das variáveis de cada comando **.PRINT**. O usuário deverá selecionar uma das opções do menu e o STAG extrairá então a tabela de dados correspondente à opção selecionada, fazendo em seguida a leitura da tabela de dados.

b) Interface com o Usuário

Na interface com o usuário merecem destaque os algoritmos utilizados para a implementação dos menus de opções e para o sistema de auxílio ao usuário.

b.1) Gerenciamento dos Menus de Opções

O gerenciamento dos menus de opções é feito através das seguintes rotinas básicas: *BGInicMenu, BGFinalizaMenu, BGLeDefMenu, BGHotMenu, BGMenu* e *BGExecMenu.*

A rotina *BGInicMenu* é responsável pela alocação de memória e inicialização das variáveis e estruturas de dados que descrevem a árvore de opções dos menus. A rotina *BGFinalizaMenu* libera a memória alocada aos menus. A rotina *BGLeDefMenu* faz a leitura do arquivo de descrição dos menus e monta a árvore de opções dos menus (no vetor *Menu*).

A ativação e gerenciamento dos menus são feitos através das funções *BGMenu* e *BGHotMenu.* A execução das operações que são folhas na árvore de opções é ativada pela rotina *BGExecMenu.*

A função *BGMenu* recebe como parâmetros as coordenadas da tela onde deve ser mostrado o menu, a posição do vetor *Menu* correspondente à primeira opção a ser mostrada e o número de opções do menu. O valor retornado será sempre nulo ou negativo. Sendo nulo, indica que a rotina terminou com o usuário teclando <ESC> no menu inicial. Se o valor retornado for negativo, isso indica que foi ativada uma folha da árvore de opções.

O algoritmo simplificado para a rotina *BGMenu* é mostrado a seguir.

```
01:   Function BGMenu
02:   Se chegou a uma folha da árvore de opções, então sai.
03:   Salva Região da tela onde será mostrado o menu.
04:   Repita
05:         Mostra o Menu.
06:         Le opção escolhida no Menu
07:         Se selecionou opção, então
08:               Procura sub-menu correspondente à opção escolhida.
09:               Se encontrou sub-menu, então
10:                     Chama rotina BGMenu para o sub-menu.
11:               senão       (* é uma folha na árvore de opções *)
12:                     Chama rotina BGExecMenu para a opção escolhida
13:                     Valor de Retorno Número da Opção (*Negativo *)
14:         senão             (* não selecionou opção, ou seja, teclou ESC *)
15:               Restaura um nível nos menus.
16:               Retorna 0
17:   Até que Valor_de_Retorno <0
18:   Restaura um nível nos menus.
19:   Retorna (Valor_de_Retorno)
20:   FIM (* BGMenu *)
```

Como podemos observar no algoritmo acima, o controle dos "caminhos" escolhidos na árvore de opções é realizado através de chamadas recursivas (linha 10). A cada chamada recursiva, o usuário desce um nível nos menus de opções. A cada retorno da rotina, ele retorna ao nível imediatamente superior dos menus. Ao ser executada uma operação que é folha na árvore de opções, a ativação (*instância*) atual termina pela saída normal da rotina (linha 19) retornando também de todas as instâncias superiores (pela linha 2).

A rotina *BGHotMenu* tem objetivo semelhante à *BGMenu*. Esta rotina é utilizada para a ativação de apenas uma sub-árvore das opções dos menus. Ela recebe como parâmetros as coordenadas da tela onde o menu deve ser apresentado e o **nome** do submenu inicial. Ao ser ativada, ela busca o nome recebido como parâmetro no vetor *Menu* e então ativa a rotina *BGMenu* para esse submenu.

b.2) Sistema de Auxílio ao Usuário

O sistema de auxílio ao usuário é ativado através da chamada à rotina *BGAtivaHelp*. Essa rotina recebe como parâmetros o nome do arquivo de *help*, o contexto atual a ser apresentado e o contexto inicial (quando foi ativado o *help*).

Ao ser ativada, a rotina fará a leitura do contexto atual no arquivo de *help* e o apresentará na tela. As palavras que definem ligações com outros contextos (*hot-links*) serão mostradas em destaque. O primeiro *hot-link* do contexto é mostrado em texto reverso. Através das setas o usuário poderá selecionar qualquer um dos *hot-links* do contexto atual e teclando <ENTER> visualizará o contexto referente ao *hot-link* atualmente selecionado.

Como o arquivo de *help* tem registros de tamanhos fixos e cada registro desse arquivo define um contexto de *help*, o acesso ao arquivo é feito de maneira direta, a cada ativação de um novo contexto. Isso simplifica as estruturas internas de dados sem prejuízo no tempo de acesso às informações de auxílio.

Internamente, a rotina *BGAtivaHelp* mantém uma lista com a sequência de contextos visualizados na atual ativação do sistema de *help*. Dessa forma, o usuário pode navegar por uma sequência de contextos qualquer e, em seguida, ir voltando aos contextos anteriormente visualizados.

c) Traçado de Curvas

Nas operações voltadas ao traçado de curvas foram utilizadas, via de regra, funções oferecidas através das próprias bibliotecas de rotinas gráficas do Turbo Pascal.

Uma exceção é o traçado interpolado de curvas. Esse traçado é realizado através de B-Splines. A partir do conjunto de coordenadas *(x,y,)* que definem os pontos da curva, é calculada uma função interpoladora (a chamada B-Spline). Após realizado o cálculo da B-Spline, é traçada a curva por ela definida. A implementação utilizada na versão atual foi adaptada a partir de rotina disponível em [Bor87].

d) Análises

Para as operações de análise são relevantes os algoritmos e métodos utilizados para os cálculos de regressões, FFTs e das estatísticas básicas.

Deve-se destacar que, para todas as operações de análise, os cálculos são aplicados sempre sobre o conjunto de pontos da atual visualização das curvas, de forma que, ativando ou desativando o *Zoom* o usuário poderá obter resultados diferentes. Isso oferece ao usuário maior flexibilidade na análise de regiões de interesse específico de uma curva.

Descreveremos a seguir cada um desses pontos.

d.1) Regressões

As regressões lineares fazem a aproximação da curva pela reta que melhor aproxima os dados definidos pelas coordenadas *(x,y)* que descrevem a curva. O método utilizado para o cálculo das regressões lineares é o dos mínimos quadrados.

Para as regressões polinomiais a curva é aproximada por um polinômio de grau *n*. Na versão atual o maior grau permitido para o polinômio é 10 (dez). O método utilizado é também o dos mínimos quadrados.

As regressões exponenciais aproximam a curva através de uma curva exponencial definida por:

$$y = A.\ e^{(Bx)}$$

O método utilizado consiste em realizar a linearização da equação acima por:

$$\ln(y) = \ln(A) + Bx$$

Como podemos obter valores inválidos para *y* e para *A*, é necessário que antes de aplicarmos a linearização, seja feito o deslocamento da curva (translação) para que os valores da tabela estejam todos no primeiro ou todos no quarto quadrante. Após a linearização é aplicado o método dos mínimos quadrados.

As regressões geométricas aproximam a curva segundo uma equação do tipo:

$$y = A.x^{B}$$

Analogamente às regressões exponenciais, é realizada também a linearização da equação acima por:

$$\ln(y) = \ln(A) + B\ \mathrm{Ln}(x)$$

Neste caso, os valores de *x* têm que ser todos positivos e os de *y* têm que ter sempre o mesmo sinal (positivo ou negativo). Após a linearização é aplicado o método dos mínimos quadrados.

d.2) Transformadas de Fourier (FFTs)

Para as transformadas rápidas de Fourier (FTTs) estão disponíveis opções para cálculo FFTs diretas e inversas. O cálculo é feito através da aplicação do algoritmo de Cooley-Tukey [Coo75] (com redução da FFT em base 2), com algumas otimizações voltadas a reduzir o número de multiplicações. As rotinas implementadas na versão atual foram extraídas do *Turbo Numerical Methods Toolbox* [Bor87] e adaptadas às nossas necessidades.

d.3) Cálculos das Estatísticas Básicas

Na versão atual, as estatísticas básicas incluem o cálculo da média, moda, mediana, desvio médio, desvio padrão, variância e coeficiente de variação para distribuição gaussiana.

Como descrevemos anteriormente, os valores calculados serão referentes somente aos pontos da curva atualmente visualizados.

O cálculo da **média** ($\overline{X}$) é dado pela média aritmética dos pontos, calculada pela fórmula:

$$\overline{X} = \sum_{i=1}^{n} (x_i) / N$$

A **moda** é definida como o valor de maior frequência no conjunto de pontos selecionado. Como as curvas serão definidas geralmente por valores reais, calculamos o intervalo de variação da amostra e dividimo-lo em 100 sub-intervalos de igual amplitude. O valor da moda é então calculado como sendo o ponto médio do sub-intervalo com maior número de pontos.

A **mediana** de um conjunto de números ordenados em ordem crescente (i.e., em um rol) é definida como o valor central (se o número de valores for ímpar) ou a média dos dois valores centrais (caso o número de valores seja par) desse rol. Geometricamente a mediana define uma reta que divide um histograma em duas partes de áreas iguais.

O **desvio médio** de um conjunto de *N* números $X_1, X_2, \ldots X_n$ é dado pela fórmula:

$$DM = \frac{\Sigma | x_i - \overline{x} |}{N}$$

O **desvio padrão** é calculado pela fórmula:

$$s = \sqrt{\frac{\Sigma | x_i^2 - \overline{x}^2 |}{(n-1)}}$$

Note-se que o cálculo do desvio padrão é feito com relação à amostra colhida e não com relação à população.

A **Variância** define-se como o quadrado do desvio padrão e pode ser obtida pela fórmula:

$$s^2 = \frac{\Sigma | x_i^2 - \overline{x}^2 |}{(N-1)}$$

O **Coeficiente de variação** é dado pela razão entre o desvio padrão e o valor da média, ou seja:

$$V = s / \overline{X}$$

IV.4 - Evoluções e Melhorias Futuras

A versão inicial do STAG teve como objetivo a criação de um protótipo da ferramenta voltada ao tratamento e análise gráfica dos resultados de testes e simulações. Para tal, determinamos um conjunto considerado mínimo de operações e recursos de tratamento e análise. Versões futuras deverão estender esse conjunto de operações.

Dentre as principais evoluções previstas para o STAG, devemos destacar aquelas voltadas à extração de parâmetros de dispositivos e de dados para otimização. Na versão atual, essas operações são realizadas de forma indireta, com a necessidade do aluno realizar cada passo voltado à extração de um determinado parâmetro e em seguida anotar manualmente no arquivo de anotações os resultados obtidos. Em versões futuras, o STAG poderá incorporar bibliotecas de parâmetros de dispositivos e um conjunto de funções com maior grau de automação para essa atividade.

Ainda com relação às operações de análise, nos cálculos de transformadas de Fourier poderão estar disponíveis futuramente operações de convolução e correlação.

As operações estatísticas básicas poderão ser também ampliadas permitindo, por exemplo, cálculos de desvios, médias, etc, entre diferentes curvas e outras análises de tendência central e dispersão mais sofisticadas. Poderão também ser incorporados outros modelos de distribuição estatística, como o de Weibull [Wei51], que se aplica a processos estocásticos e que é amplamente utilizado na engenharia de qualidade e confiabilidade.

No tratamento de visualização poderão ser incorporadas facilidades voltadas ao traçado de curvas tridimensionais.

É previsto também um maior número opções de configuração do sistema, no que se refere a tipos e modelos de periféricos. Seria conveniente, por exemplo, o suporte a traçadores gráficos e um maior número de impressoras. Para dispositivos de entrada, seria de interesse o suporte a canetas óticas (*light-pen*), mesas digitalizadoras, etc...

Com relação à interface usuário-software, é previsto suporte a um maior número de idiomas. Para o sistema de auxílio ao usuário, podem também ser realizadas melhorias como, por exemplo, permitir a inclusão de figuras nas telas de *help* e permitir buscas por assunto (índice).

Encontra-se também em andamento um estudo de viabilidade da criação de uma versão do sistema para o ambiente *Windows*. Nesse ambiente poderíamos, sem sombra de dúvida, ter uma melhor interface entre usuário e *software* e várias facilidades adicionais no tratamento gráfico, no suporte a diferentes tipo de periféricos e no estabelecimento de vínculos entre diferentes atividades do projeto do circuito. Um fator contrário à utilização desse ambiente é que, para obtermos um desempenho satisfatório do sistema, tem-se a necessidade de máquinas com maior capacidade de processamento, monitores de vídeo de melhor resolução, entre outros fatores que podem comprometer nosso compromisso com a obtenção de um sistema de baixo custo.

V - Sistema para Testes Funcionais (STEF)

O STEF - Sistema para Testes Funcionais da Lógica Digital será o sub-sistema do SAD responsável pela integração das funções de programação, execução e análise de resultados de testes de lógica funcional de circuitos digitais.

A partir de programas para o simulador lógico ou através de programação específica, em forma textual ou interativa, será possível ao aluno gerar programas para testes de validação da lógica funcional dos circuitos em estudo.

A partir do programa do experimento, o STEF controlará a aplicação de estímulos ao circuito e a leitura de respostas, as quais poderão ser monitoradas em tempo real.

Ao final da execução de testes, o aluno terá disponível um conjunto de operações de análise, quando poderá então realizar comparações entre os resultados obtidos através de simulações e de testes.

Uma versão inicial do STEF foi implementada e encontra-se atualmente em fase de testes de integração das funções. Essa versão inicial compreende apenas um conjunto mínimo de operações, contendo funções básicas voltadas à programação do experimento, as funções completas referentes à execução do experimento e um conjunto mínimo de operações de análises dos resultados. Funções adicionais encontram-se em fase de implementação. O anexo IV apresenta o manual do usuário para esta versão inicial do STEF.

Neste capítulo faremos a descrição geral do STEF, a especificação de seus requisitos de *software*, a descrição de aspectos relevantes da implementação e, por fim, discutiremos as principais evoluções e melhorias previstas para versões futuras.

V.1 - Descrição Geral do STEF

V.1.1 - Perspectivas do Sistema

O STAG destina-se às atividades de programação, execução e análise de resultados de testes funcionais de circuitos digitais. Em conjunto com as demais ferramentas do SAD, o qual deverá atuar de forma integrada com o SDP - Sistema Didático de Projetos, pretende-se obter um sistema completo voltado a laboratórios de ensino de eletrônica em universidades.

V.1.2 - Características dos Usuários

O grupo de usuários previsto para o STEF compõe-se por alunos de cursos de graduação e pós-graduação nas áreas de engenharia elétrica e de computação. Supõe-se que esses usuários tenham alguma familiaridade no uso de microcomputadores e que tenham os conhecimentos teóricos básicos necessários às atividades de teste funcionais de circuitos.

V.1.3 - Suposições, Dependências e Restrições

Tomou-se como suposição básica para o desenvolvimento do STEF que o ambiente de execução será aquele apresentado na seção II.2 deste trabalho e que os usuários do sistema serão aqueles que acabamos de descrever.

Na versão atual, o SIMUL é o simulador lógico para o qual o STEF é capaz de converter programas de simulação para programas de testes funcionais e a interface de aquisição de dados suportada é aquela desenvolvida por [Qua88], embora no projeto e implementação do STEF tenha havido a preocupação com a independência funcional do sistema com relação ao simulador lógico utilizado e com relação ao tipo de interfaces de aquisição de dados suportadas.

No que diz respeito ao desempenho do sistema, deve-se considerar que a interface de aquisição de dados utilizada, bem como a capacidade de processamento do computador utilizado, serão fatores determinantes.

Considerando-se que o STEF atuará essencialmente em modo gráfico, torna-se aconselhável o uso de monitores de vídeo com melhor resolução. Uma resolução de 640 por 350 pontos pode ser considerada satisfatória. O uso de monitores de baixa resolução poderá tornar desagradável a utilização do sistema pelo aluno.

V.2 - Especificação de Requisitos

V.2.1 - Requisitos Funcionais

Conforme citamos anteriormente, o STEF deverá integrar as atividades voltadas à realização de testes funcionais de circuitos digitais, abrangendo as etapas de descrição (programação) do experimento, execução e de análises de resultados desse tipo de testes.

Do ponto de vista funcional, o STEF pode ser particionado nos seguintes módulos:

- Tradução de Programas de Simulação para Programas de Experimento e vice-versa.
- Programação Interativa dos Experimentos de Testes Funcionais
- Execução Interativa dos Programas de Experimento
- Apresentação e Tratamento dos Resultados de Testes
- Análises de Resultados dos Testes Funcionais

A figura 24 apresenta um diagrama lógico dos processos acima descritos e seus relacionamentos com as bases de dados utilizadas pelo STEF.

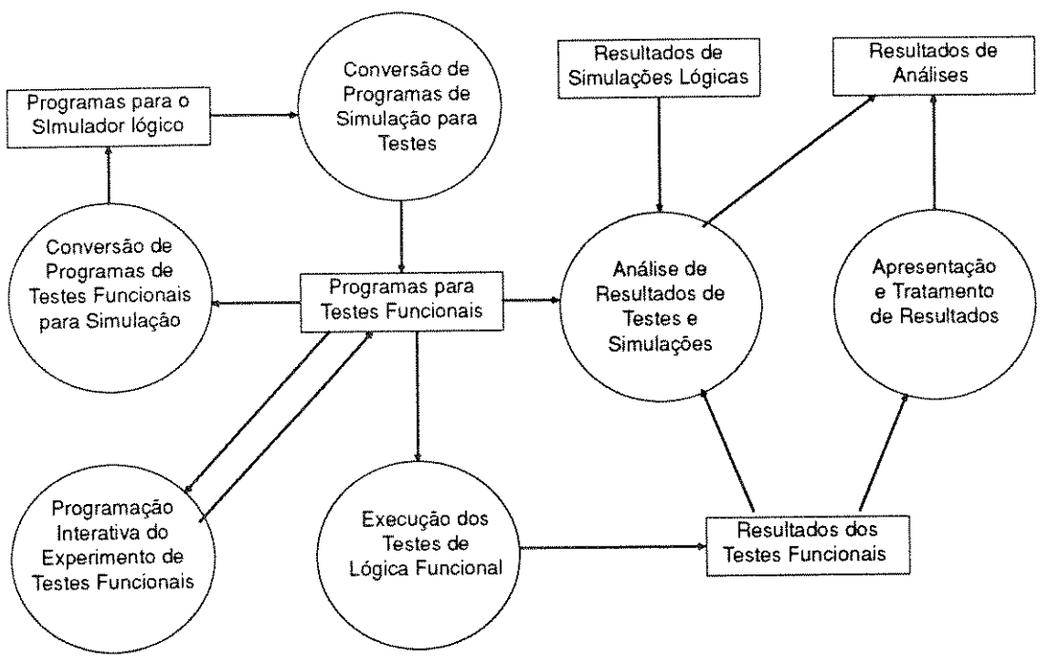


Figura 24 - Principais Módulos Funcionais do STEF

V.2.1.1 - Descrição das Operações Básicas

Visando a uma melhor organização das atividades do aluno durante a realização de testes funcionais, as operações disponíveis ao STEF foram organizadas em quatro grupos básicos:

- Operações de Definição do Experimento
- Operações de Execução dos Testes Funcionais
- Operações de Visualização e Análise de Resultados
- Operações de Configuração do Sistema

Descreveremos a seguir cada um desses grupos de operações, dando destaque aos aspectos de funcionalidade de cada operação. Detalhes dos aspectos de interação entre o usuário e o sistema serão descritos mais adiante, na seção que trata da implementação do STEF.

a) Operações para Definição do Experimento

As operações voltadas à definição do experimento permitirão ao aluno realizar as operações relacionadas à programação dos testes funcionais. Como resultado das operações de definição do experimetno será gerado um programa de experimento, segundo a linguagem descrita no capítulo III.

Estarão disponíveis ao aluno cinco operações básicas:

- Identificação do Projeto
- Descrição do Circuito
- Descrição dos Estimulos para Inicializacao
- Descrição dos Estimulos para Testes
- Definição dos Sinais de Saida

A **identificação do projeto** será feita através da definição de um nome lógico para o projeto. Após definido um nome para o projeto, os arquivos de programas e de resultados gerados pelo STEF utilizarão esse mesmo nome, sendo diferenciados pelas suas extensões. Para a geração de programas de testes funcionais a partir de programas para o SIMUL, o nome do projeto deverá corresponder ao nome do arquivo que contém o programa do SIMUL a ser convertido. Nesse caso a conversão do programa será feita de forma automática. O objetivo básico da operação de identificação do projeto é facilitar a organização das atividades do aluno. A definição do nome do projeto não é obrigatória. Caso o usuário não defina o nome do projeto, o STEF assumirá um nome *default* e, ao final da sessão de uso do sistema perguntará ao aluno se ele deseja ou não armazenar os dados referentes ao projeto em disco.

A **descrição do circuito**, na versão inicial do STEF, não estará disponível. Para a execução dos testes envolvendo apenas variáveis terminais, a descrição do *net-list* do circuito não é necessária. Nosso interesse na descrição do circuito será voltado principalmente à visualização de diagramas esquemáticos, oferecendo ao usuário maiores facilidades na descrição interativa do experimento. Em versões futuras, as informações sobre o *net-list* do circuito serão também de interesse para a execução de operações mais avançadas, voltadas à análise crítica de falhas do circuito e ao aconselhamento do aluno na depuração. A operação de descrição do circuito deverá permitir ao aluno descrever, de forma gráfica e interativa o circuito sob testes, em termos de componentes lógicos básicos.

Quando da **execução dos testes funcionais**, pode haver a necessidade de que circuito possua um conjunto de condições lógicas iniciais antes da aplicação dos vetores de teste propriamente ditos. Através da operação de descrição dos estímulos para inicialização o aluno poderá definir um conjunto de estímulos a serem aplicados ao circuito e o conjunto de condições lógicas iniciais a serem atingidas.

A **descrição dos vetores de inicialização** poderá ser feita na forma textual, conforme descrito no capítulo III ou de maneira interativa, através de um editor gráfico de estímulos incorporado ao STEF.

As condições de parada para a aplicação dos estímulos de inicialização serão definidas interativamente através de uma expressão lógica. O STEF deverá realizar a interpretação dessa expressão imediatamente após sua digitação, devendo acusar eventuais erros sintáticos na sua definição.

A operação de **descrição dos estímulos para testes** permite que o aluno defina os vetores de teste propriamente ditos e um conjunto de condições para a parada dos testes. Tanto a definição dos estímulos (vetores de teste) quanto a das condições de parada dos testes serão realizadas de maneira análoga àquela descrita para os estímulos de inicialização do circuito.

A operação de **definição dos sinais de saída** permitirá ao aluno selecionar, interativamente, os sinais a serem monitorados como resultados dos testes. Para a seleção dos sinais a serem monitorados, o STEF apresentará um diagrama com os pinos do soquete de testes. Através de um cursor, o usuário deverá selecionar cada um dos pinos onde deverão ser medidos os estados lógicos de saídas.

Além dessas operações básicas, teremos uma série de operações que serão executadas automaticamente pelo sistema, sem a intervenção do usuário. Entre elas estão a conversão automática de programas do SIMUL para o STEF e vice-versa e a geração do arquivo de programa do experimento conforme a linguagem definida no capítulo III.

Embora as operações que acabamos de descrever sejam realizadas essencialmente de forma interativa o aluno poderá, opcionalmente, descrever o experimento através da linguagem de programação. Para tal, o STEF deverá ser capaz de dar acesso a um editor de textos. O aluno poderá definir o editor de textos de textos a ser utilizado através de opção específica na configuração do STEF. Como *default* o STEF utilizará o mesmo editor de textos do SDP.

b) Operações para Execução dos Testes Funcionais

O grupo de operações para execução do experimento contém, além da operação de execução propriamente dita, operações voltadas à descrição dos sinais de saída a serem monitorados e de alguns parâmetros para a execução dos testes.

Estarão disponíveis as seguintes operações:

- Definição do Método de Armazenamento dos Resultados
- Definição da Visualização de Resultados
- Executar Experimento

Através da operação de definição do método de armazenamento dos resultados o aluno definirá se os dados, durante a execução, devem ser armazenados em memória ou diretamente em arquivos. Caso seja selecionada a opção de armazenamento em memória deverá ser possível, ao final dos testes, gerar os resultados num arquivo em memória de massa. O STEF utilizará como *default* o armazenamento dos resultados em memória. Deve-se observar que o armazenamento dos dados diretamente em arquivos pode tornar mais lenta a execução dos testes.

A definição da visualização de resultados permitirá ao aluno escolher se os resultados dos testes devem ser apresentados em tempo de execução ou somente ao final dos testes. A apresentação de resultados em tempo de execução permite um melhor acompanhamento da execução dos testes e será, portanto, o *default* para o STEF. Deve-se considerar que a apresentação dos dados em tempo real pode tornar a execução dos testes um pouco mais lenta, dependendo do computador utilizado. Os resultados serão mostrados graficamente, na forma de diagramas de estados dos sinais ao longo do tempo.

A operação de execução do experimento realizará o controle das interfaces para a aplicação de estímulos ao circuito e a leitura dos estados lógicos observados à sua saída. Antes de iniciar a execução propriamente dita, o sistema solicitará ao aluno algumas informações sobre o experimento, como os instantes inicial e final para a monitoração dos resultados, o incremento para cada leitura e o instante final (máximo) para a execução dos testes.

Durante a execução do experimento o aluno poderá, através do teclado, interromper a realização dos testes. Após essa interrupção, o aluno poderá redefinir características sobre o experimento e em seguida retomar os testes. Os testes poderão ser retomados a partir do início ou do ponto onde foram interrompidos.

c) Operações de Visualização e Análise de Resultados

Este grupo de operações permitirá ao aluno verificar graficamente os resultados dos testes e realizar comparações entre diferentes baterias de testes e entre resultados de testes e simulações. Em versões futuras do sistema poderão ser incluídas funções mais sofisticadas que, a partir dos resultados de comparações, realizem críticas e aconselhamento, trazendo ao aluno conceitos e técnicas ligados a metodologias de depuração e verificação de falhas de circuitos.

Na versão inicial, o STAG permitirá a realização de duas operações:

- Análises Simples
- Análises Comparativas

A operação de análises simples permitirá a simples visualização dos resultados obtidos dos testes. Será apresentado ao aluno o diagrama de estados lógicos dos sinais ao longo do tempo e serão permitidas operações básicas de manipulação desses dados, como a visualização de detalhes e a impressão dos dados da tela.

Para as análises comparativas, o aluno deverá fornecer ao sistema o nome do arquivo de resultados (de testes ou simulações) a ser comparado com os dados atualmente disponíveis em memória. Serão apresentados em seguida esses dois conjuntos de dados, sendo mostradas em destaque as diferenças entre os dois conjuntos de resultados e sendo também permitidas as mesmas manipulações de dados disponíveis para as análises simples.

As funções de crítica e aconselhamento, a serem incluídas em versões futuras do sistema, terão como objetivo auxiliar o usuário na depuração de falhas do circuito. Por exemplo, poderão estar disponíveis funções que, dada uma falha, apresentem gradativamente ao aluno as ramificações do circuito que podem ter gerado tal falha, sugira vetores de testes que possam auxiliar na sua localização, etc... Essas funções deverão utilizar-se de técnicas e métodos de validação de circuitos digitais já estabelecidos. Estudos mais detalhados deste grupo de funções deverão ser realizados, a fim de determinar as funções de maior interesse ao aprendizado e as melhores formas de induzir o aluno à exploração do experimento. Os estudos deverão levar também a definições sobre a forma de implementação das funções, estabelecendo, por exemplo, a conveniência ou não de um sistema especialista e outros aspectos da implementação.

d) Operações de Configuração do Sistema

As operações de configuração do sistema deverão permitir ao aluno definir características do ambiente de execução do sistema. As operações disponíveis serão:

- Configuração de Interface
- Configuração de Cores
- Configuração de Idioma
- Configuração do Editor de Textos
- Configuração de Periféricos

A operação de configuração de interface permitirá que o aluno defina o tipo de interface para testes instalado e características particulares ao tipo de interface selecionado. Atualmente. o STEF suporta um único tipo de interface [Qua88]. Deve-se ressaltar que esta operação deverá ser utilizada com cuidado, devendo ser observados detalhes da configuração da interface de testes e do computador, a fim de evitar conflitos que podem posteriormente vir a "travar" o sistema. Após configurar um novo tipo de interface de testes pode ser necessário que o usuário reinicialize o sistema.

Por meio da operação de configuração de cores o aluno poderá definir as cores utilizadas em tela pelo STEF, como as cores para cabeçalhos, mensagens, para as variáveis do gráfico, etc...

A configuração de idiomas permitirá que o usuário selecione o idioma a ser utilizado pelo sistema para a apresentação dos menus de opções, das mensagens de erro e advertência e para o sistema de *help*. Embora esse tipo de operação seja importante em um sistema que pretende ser utilizado em países de diferentes idiomas, na versão inicial o Português será o único idioma disponível. As versões futuras deverão dar suporte a outros idiomas, como o espanhol, o inglês, o italiano, etc...

Através da operação de configuração do editor de textos o aluno poderá definir o editor a ser ativado pelo STEF para a edição dos programas de experimento. Como *default* o STEF utilizará o mesmo editor de textos disponível ao SDP. Deve-se observar, para esta operação, que dependendo do editor de textos escolhido, pode ocorrer falta de memória para a sua execução. Nesse caso, aconselha-se a utilização do próprio editor de textos do SDP.

A operação de configuração de periféricos deverá permitir a definição dos tipos e modelos de periféricos disponíveis para o sistema, como impressoras, traçadores gráficos (*plotters*), caneta ótica, *tablet*, etc... Na versão inicial o STEF dará suporte somente a um número mínimo de impressoras.

Quaisquer modificações na configuração do sistema, exceto a configuração da interface de testes, serão válidas somente para a atual sessão de uso do sistema. Caso tenham sido realizadas modificações na configuração, antes de terminar a sessão o STEF perguntará ao usuário se as configurações devem ser salvas ou não. Caso a resposta seja afirmativa, os dados da nova configuração são gravados em disco, tornando-se permanentes.

V.2.1.2 - Bases de Dados

As bases de dados utilizadas pelo STEF incluirão dados de uso comum ao STEF e a outras ferramentas do SAD e dados de uso exclusivo. Elas estarão organizadas nos seguintes grupos:

- Programas de Testes Funcionais
- Descrição Esquemática do Circuito sob Testes
- Programas para Simuladores Lógicos
- Dados de Resultados de Testes
- Dados de Resultados de Simulações Lógicas
- Dados de Configuração do Sistema
- Dados de Auxílio ao Usuário
- Dados de Descrição dos Menus

Discutiremos a seguir cada um desses grupos. Detalhes sobre o formato de cada uma das bases de dados são apresentados no anexo I. A representação interna dos dados será descrita na seção V.3.3, que trata das estruturas de dados.

a) Programas de Testes Funcionais

Os programas para testes funcionais poderão ser gerados pelo STEF de duas formas distintas: **a)** a partir de programas para simuladores lógicos, através de tradutores incorporados ao STEF; **b)** a partir da descrição interativa do experimento de testes funcionais.

Um programa para testes funcionais será descrito num arquivo em formato texto segundo a linguagem de programação especificada no capítulo III. Nesse programa estarão descritos estímulos de inicialização e os vetores de teste a serem aplicados ao circuito, condições de parada para a inicialização e para os testes, os sinais a serem monitorados e comandos específicos para a definição de parâmetros de execução e controle da mesma.

b) Descrição Esquemática do Circuito

Os dados de descrição esquemática do circuito serão gerados pelo STEF, a partir da descrição interativa do circuito, ou a partir de um arquivo gerado pelo PESQA.

A utilização da descrição esquemática do circuito é de interesse, inicialmente, somente para a visualização do circuito durante as operações de definição dos vetores de testes e dos sinais a serem monitorados.

Em versões futuras, o *net-list* do circuito será de interesse para dar suporte à implementação de funções de aconselhamento voltadas ao auxílio na depuração de falhas em circuitos.

Para a versão inicial do STEF a descrição do circuito não será utilizada, dado que não estarão disponíveis as funções de auxílio à depuração e localização de falhas.

Estudos dirigidos poderão ser realizados para avaliar a viabilidade e conveniência de incluir-se informações para a descrição esquemática do circuito no próprio programa do experimento.

c) Programas para Simuladores Lógicos

Os programas para simuladores lógicos serão utilizados como entrada pelo STEF para a geração, de forma automática, dos programas de testes funcionais.

De forma análoga, o STEF deverá ser capaz de, a partir de um programa de testes funcionais, gerar programas para simuladores lógicos.

Inicialmente o STEF prevê a conversão de programas de simulação escritos para o SIMUL, que é o simulador lógico incorporado ao SDP.

Deve-se notar que, como a linguagem proposta para a descrição de testes funcionais é uma extensão da linguagem do SIMUL, na tradução de programas de testes para programas de simulação alguns comandos (de descrição dos pinos do circuito e de controle da execução) serão desconsiderados. Ainda, se um programa de testes não contiver a descrição do *net-list* do circuito, será impossível realizar a tradução do programa, dado que o SIMUL precisaria dessas informações para realizar as simulações.

d) Dados de Resultados de Testes

Os dados de resultados de testes serão gerados pelo STEF com a finalidade de manter registros históricos dos experimentos realizados. Esses arquivos serão gerados em formato texto e neles estarão contidos, além dos dados de resultados dos testes, também os dados de descrição do circuito e do experimento (i.e., o programa do experimento). Isso faz com que o usuário tenha posteriormente as informações completas sobre os testes e seus resultados.

Os dados de resultados de testes poderão ser utilizados também como entrada para o sistema, nas operações de análises comparativas.

e) Dados de Resultados de Simulações

Os dados de resultados de simulações serão utilizados pelo STEF para a comparação com resultados de testes.

Esses dados serão lidos pelo STEF e, em seguida, convertidos para um formato interno. Na sua versão inicial, o STEF será capaz de ler e converter somente dados gerados como saídas do SIMUL.

f) Dados de Configuração do Sistema

Os dados de configuração do sistema estarão presentes num único arquivo e serão lidos pelo STEF durante o processo de inicialização do sistema. Esses dados descreverão informações básicas para a execução do STEF como, por exemplo, o tipo de interface de testes utilizada, idioma, tipos e modelos dos dispositivos de entrada e saída disponíveis, cores a serem utilizadas nas telas, etc...

Caso o usuário realize modificações na configuração do sistema o STEF, ao final da sessão de uso, perguntará se os dados devem ser atualizados no arquivo de configuração ou não.

g) Dados de Auxílio ao Usuário (*Help*)

A base de dados de auxílio ao usuário (*help*) será composta por um conjunto de arquivos, cada qual contendo as informações de *help* em um idioma diferente. Um arquivo de dados de *help* conterá os textos de todas as telas de auxílio disponíveis ao STEF, bem como as informações que definem ligações (*hot-links*) entre diferentes telas. O STEF fará acesso a esses arquivos somente para leitura. A atualização desses arquivos será feita através de um programa utilitário, descrito no anexo II, desenvolvido especificamente para a finalidade de criação e manutenção dos sistemas de *help*.

Como descrevemos anteriormente, a forma como foram estruturados os arquivos de *help* permitem um acesso rápido a qualquer informação ali contida e, principalmente, permite que as telas de *help* sejam modificadas sem a necessidade de recompilação do STEF. Ainda, pode-se acrescentar ao sistema dados de *help* em um novo idioma pela simples definição do arquivo de *help*.

h) Dados de Descrição dos Menus

A base de dados de descrição dos menus será composta por um arquivo de dados para cada idioma disponível ao STEF. Cada um desses arquivos conterá a descrição completa da árvore de opções dos menus do sistema, das operações a serem executadas pelas opções que são folhas nessa árvore e dos contextos de *help* associados às opções.

A utilização de arquivos de dados para a descrição dos menus de opções do sistema confere-nos bastante facilidade para a criação de menus de opções para outros idiomas, para a modificação da estrutura da árvore de menus ou mesmo para a criação de opções adicionais para o sistema. Na maioria das vezes, poderão ser realizadas alterações nos menus sem a necessidade de recompilação dos programas-fonte do STEF.

V.2.2 - Requisitos de Interface

V.2.2.1 - Interface Usuário-Software

A interface usuário-software do STEF deverá, em linhas gerais, manter os mesmos padrões propostos para as demais ferramentas que comporão o SAD. A uniformidade das interfaces usuário-software dos vários sub-sistemas é um ponto básico à facilidade de uso do SAD como todo.

A utilização de bibliotecas de funções que implementam os elementos básicos da interface usuário-software contribuiu sobremaneira para a obtenção da uniformidade entre as diversas ferramentas. Nessas bibliotecas de funções estão contidas as rotinas de ativação do sistema de auxílio ao usuário (*help*), implementado na forma de hipertexto e com sensibilidade ao contexto operacional, as rotinas para gerenciamento dos menus de opções, emissão de mensagens em tela, etc...

A forma básica de interação do usuário com o sistema será realizada por meio de menus de opções, através dos quais estarão disponíveis todas as operações do sistema. As operações estarão agrupadas de modo a direcionar as atividades do aluno e fazer com que ele perceba as distinções entre as atividades de programação, execução e análise de resultados dos testes.

A qualquer instante estarão disponíveis, através da tecla <F1>, informações de auxílio ao usuário. O sistema de *help* será sensível ao contexto de forma que, sempre que for ativado, mostrará informações referentes às atividades atuais. Essas informações deverão abranger não somente aspectos da operação do sistema, mas também informações conceituais ligadas às atividades em execução. A implementação na forma de hipertexto permitirá também que o aluno "caminhe" através do sistema de *help*, visualizando informações sobre outros tópicos associados ao tópico atual.

Outro aspecto relevante da interface usuário-software é a capacidade de utilização de diferentes idiomas, prevendo a possibilidade de utilização do sistema em países de outros idiomas.

Os principais aspectos referentes à interface com o usuário para a realização das operações específicas do STEF serão discutidos na seção V.3.2.

V.2.2.2 - Interfaces com Outros Sistemas

Considerando-se que o STEF é uma ferramenta que implementa as etapas de programação, execução e análises de resultados de testes de lógica funcional de circuitos digitais, ele poderá ser utilizado como um sistema independente de outras ferramentas do SAD e do SDP.

Apesar do STEF possuir essas características, devemos considerar que um dos pontos mais importantes na definição do SAD é a integração das atividades relacionadas às várias etapas da concepção dos circuitos.

O STEF deverá, portanto, possuir interfaces com as seguintes ferramentas: o SIMUL, responsável pela simulação lógica dos circuitos; o PESQA, ferramenta para a entrada esquemática de circuitos digitais; o EDTES, editor de estímulos digitais.

As interfaces do STEF com esses sistemas serão estabelecidas através dos arquivos de dados gerados por essas ferramentas e por bases de dados geradas pelo STEF.

A interface com o SIMUL será feita através dos programas de simulação. A partir de programas de simulações, o STEF deverá ser capaz de gerar programas de testes funcionais. Poderá também ser realizada a operação inversa, ou seja, a geração de programas para o SIMUL a partir dos programas de testes funcionais. O STEF deverá ainda ser capaz de ler dados de resultados das simulações realizadas através do SIMUL.

Para a interface com o PESQA, o STEF deverá ser capaz de interpretar os arquivos de descrição de circuitos gerados por essa ferramenta. Na versão inicial do STEF, esta interface não estará implementada.

A interface com o EDTES será realizada através dos arquivos de descrição de estímulos. A descrição de estímulos será sintática e semanticamente idêntica para o STEF e o EDTES. O STEF poderá utilizar dados gerados pelo EDTES para descrições de estímulos de inicialização do circuito e para os vetores de testes propriamente ditos.

A implementação das interfaces do STEF com outras ferramentas deverá ter como meta básica a modularidade, permitindo dessa forma que futuramente sejam implementadas interfaces com outros tipos de simuladores, editores esquemáticos e geradores de vetores de testes sem que isso afete outros módulos de implementação do STEF.

V.2.2.3 - Interfaces de *Hardware*

As interfaces de *hardware* utilizadas pelo STEF podem ser classificadas em dois grupos distintos:

- Interface para Testes Funcionais
- Interfaces com periféricos de Entrada e Saída

As interfaces para os testes funcionais, especificadas na seção II.2.1.2 serão responsáveis pela aplicação dos estímulos ao circuito sob testes e pela aquisição dos dados verificados às saídas do mesmo.

Com relação às interfaces com periféricos de entrada e saída, as características mínimas, definidas na seção II.2.1.1, são suficientes ao STEF.

Considerando-se que o sistema faz uso contínuo de telas gráficas para a interação com o usuário, o uso de um monitor de vídeo com padrão EGA ou superior é aconselhável.

V.3 - Descrição da Implementação

Conforme foi citado anteriormente, a versão do STEF atualmente disponível é um protótipo que reúne somente as operações essenciais à sua utilização e que se encontra em fase final de integração.

Descreveremos nesta seção os principais aspectos relacionados à implementação da versão atual do STEF.

V.3.1 - Módulos de Implementação

Um dos principais critérios na definição dos módulos de implementação para o STEF foi com relação à definição e isolamento das rotinas que implementam as interfaces com outras ferramentas e com as interfaces de *hardware* para testes. Buscou-se assim permitir que o STEF possa futuramente, sem grandes esforços, suportar diversos tipos de interfaces de testes e compartilhar dados com um maior número de ferramentas de software.

Os módulos de implementação definidos para a atual versão do STEF são relacionados a seguir.

- **STEF** - É o módulo principal do sistema, implementando o controle geral da execução do sistema, atuando como o integrador dos demais módulos de implementação.
- **BGGRAF** - Biblioteca de rotinas básicas de visualização gráfica e tratamentos de alto nível do gerenciamento de menus de opções.
- **BGVARS** - Contém as declarações de constantes, tipos e variáveis globais da biblioteca de rotinas gráficas (BGGRAF) e as respectivas inicializações.
- **BGEXCMNU** - Implementa o tratamento de médio nível de gerenciamento de menus de opções.
- **BGHELP** - Implementa as rotinas básicas de gerenciamento e ativação do sistema de auxílio ao usuário (*help*).
- **TDVARS** - Contém as declarações e rotinas de inicialização de variáveis globais ao STEF. Este módulo é utilizado por todos os outros módulos do STEF, exceto aqueles que implementam as bibliotecas de rotinas comuns (i.e., aqueles cujos nomes têm o prefixo "BG").
- **TDARQ** - Implementa as rotinas relacionadas ao tratamento de entrada e saída através de arquivos, como leitura e conversão dos programas de testes e para o SIMUL, geração das saídas do STEF, etc...
- **TDGERAL** - Rotinas de uso geral do STEF, incluindo rotinas para a interface usuário-software e a camada de mais alto nível das rotinas de ativação dos testes funcionais.
- **TDINTERF** - Contém as rotinas que fazem o acesso e controle das interfaces de testes funcionais. Na versão atual controla apenas um tipo de interface, mas prevê o suporte a outros tipos.
- **TDMENU** - Contém as rotinas de ativação das opções que são folhas de na árvore de opções dos menus.
- **TDPARSE** - Implementa as rotinas de *parsing* e de avaliação dos comandos que definem condições de parada para a inicialização e testes dos circuitos.

V.3.2 - Características da Interface Usuário-software

A interface usuário-software do STEF utiliza a tela e o teclado (ou *mouse*) como os principais meios de interação.

A tela principal do STEF apresentará, ao alto, o menu principal de opções do sistema. Na metade superior da tela, imediatamente abaixo do menu principal, será apresentado o desenho do soquete da interface de testes funcionais. A metade inferior da tela será reservada para outras operações, como a definição de estímulos a aplicar ao circuito e das condições de parada. A figura 25 apresenta um esboço das regiões da tela principal do STEF.

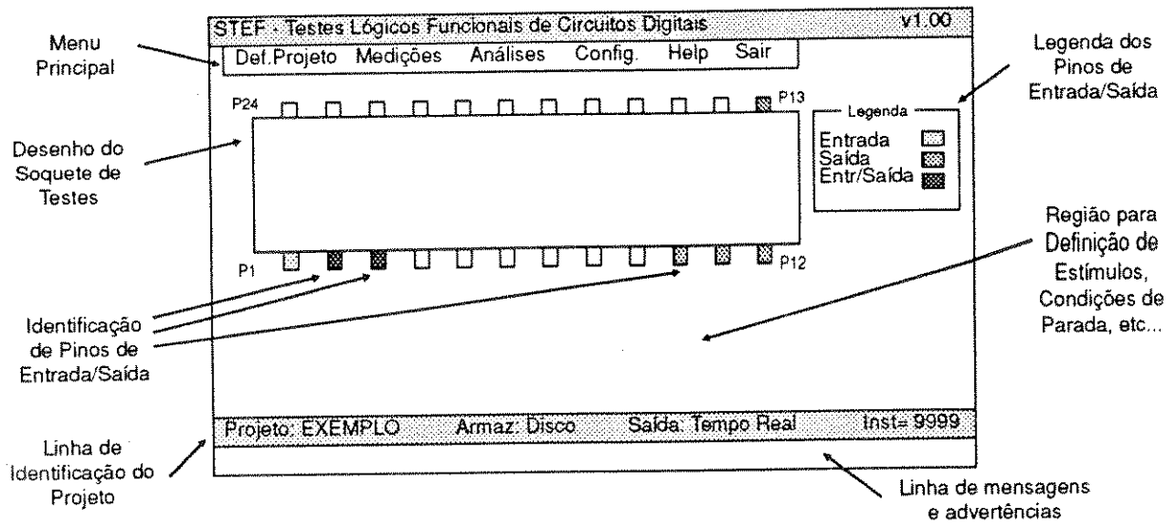


Figura 25 - Regiões da Tela Principal do STEF

Em versões futuras do sistema, quando estiverem implementadas as funções voltadas à representação do diagrama esquemático do circuito, o diagrama deverá ser mostrado dentro da região definida para o desenho do soquete. Deverão ser permitidas, a partir daí, operações de tratamento de visualização do circuito.

O menu principal de opções estará sempre presente na tela, imediatamente abaixo da linha de cabeçalho. A linha de mensagens, na parte inferior da tela, assim como a linha de informações sobre o projeto, também estarão presentes na tela a todo instante.

O desenho do soquete da interface de testes estará presente, além da tela principal, nas telas das operações de programação do experimento. Selecionando as operações de definição de estímulos ou de definição dos sinais a serem monitorados, o aluno passará a ter disponível um cursor em forma de cruz que se movimenta somente sobre os pinos do desenho do soquete de testes. Posicionando o cursor sobre um determinado pino e teclando <INS> ele poderá definir ou modificar informações sobre estímulos ou sobre a monitoração de sinais desse pino. Teclando <DEL> ele excluirá aquele pino da aplicação ou monitoração de sinais.

Para a definição de estímulos (de inicialização ou dos vetores de teste), após selecionado o pino, o usuário deverá selecionar, através de um menu de opções, o tipo de sinal: periódico, aperiódico ou retangular. Em seguida, o STEF apresentará graficamente, na metade inferior da tela, o editor de estímulos, contendo o sinal a ser editado. O aluno poderá movimentar o cursor horizontalmente e *clicando* o mouse ou teclando <ENTER>, ele poderá inserir transições no sinal. A figura 26 é um exemplo da tela apresentada para a edição de estímulos.

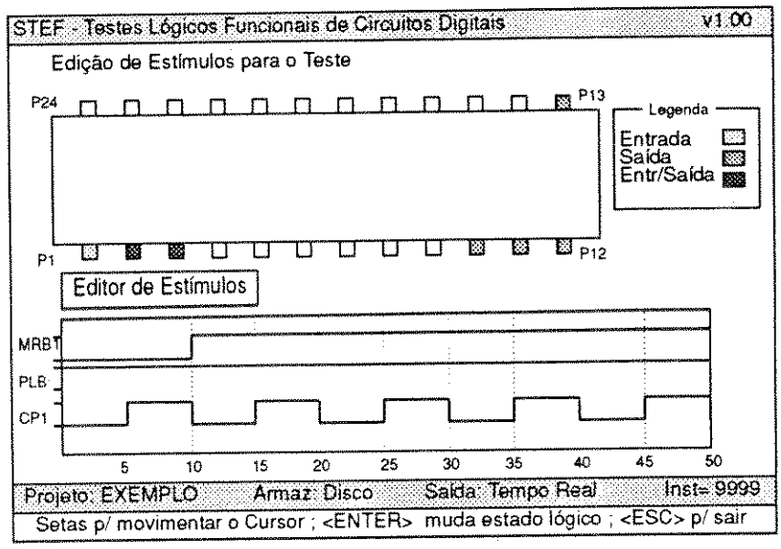


Figura 26 - Tela para Edição de Estímulos no STEF

Para a definição das condições de parada, o aluno terá, na metade inferior da tela, uma janela onde deverá digitar a expressão que define as condições de parada, conforme apresentado na figura 27. A definição da expressão deve obedecer à sintaxe descrita no capítulo III.

Durante a execução do experimento, caso o aluno tenha selecionado a opção de visualização de resultados em tempo real, estes serão apresentados na forma de um diagrama de estados, conforme mostra a figura 28. Caso tenha selecionado a visualização somente ao final dos testes, a tela será a mesma da figura 25, acrescentando-se somente a indicação do instante atual no canto superior direito.

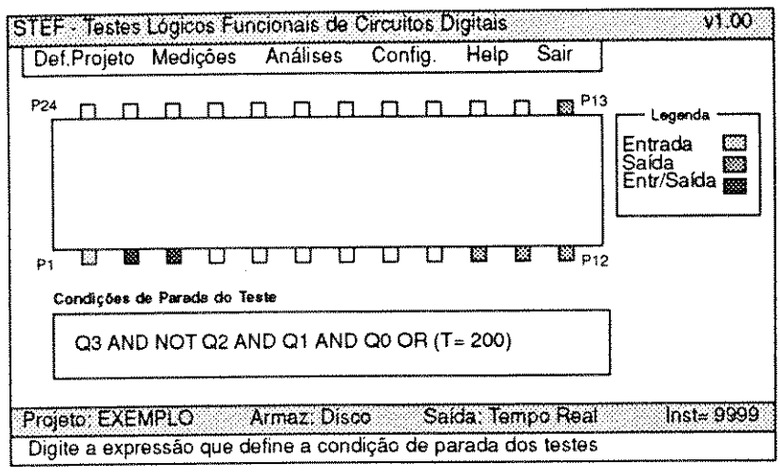


Figura 27 - Tela para Definição de Condições de Parada

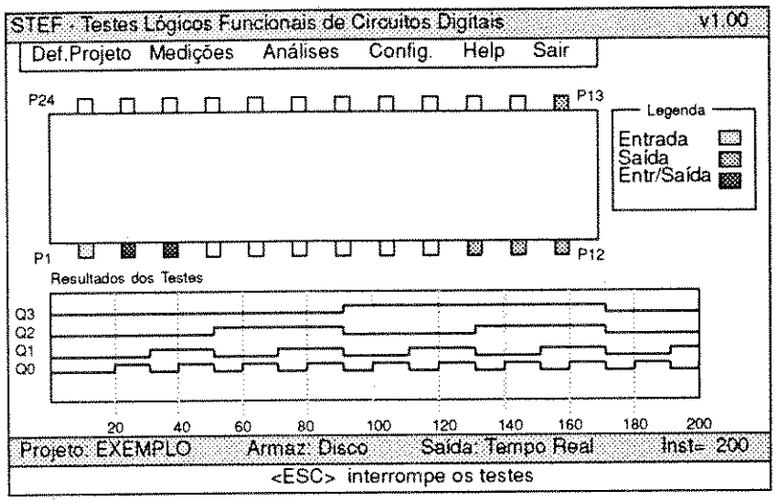


Figura 28 - Tela para Monitoração dos Resultados de Testes

Para as operações de análise, no caso da análise simples, a tela será semelhante à da figura 28, com a diferença de que o usuário terá disponível um cursor na região do diagrama de estados, através do qual ele poderá avançar ou retroceder no gráfico. Algumas operações adicionais permitirão também a visualização de detalhes, entre outros. No caso de análises comparativas, serão apresentados os diagramas de estados dos dois conjuntos de respostas em comparacão, sendo apresentadas em destaque as diferenças entre eles. O usuário terá também um cursor, para avançar ou retroceder na visualização dos dados, comandos para a visualização de detalhes e para selecionar somente os intervalos onde há diferenças entre os dois conjuntos de dados. A figura 29 ilustra um exemplo da tela apresentada para as análises comparativas.

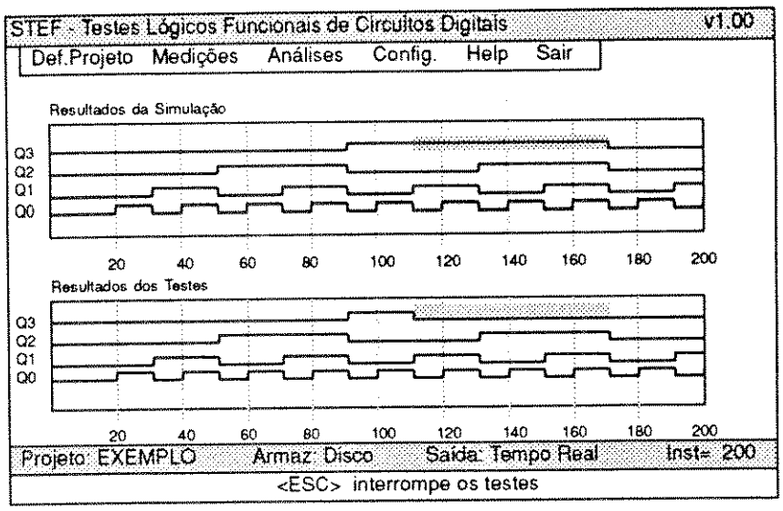


Figura 29 - Análise Comparativa de Resultados no STEF

V.3.3 - Estruturas de Dados

Nesta seção descreveremos as principais estruturas de dados utilizadas pelo STEF.

Omitiremos nesta seção as discussões sobre as estruturas de dados utilizadas para a representação dos menus de opções do sistema e para o sistema de *help*. Essas estruturas são as mesmas descritas para o STAG, na seção IV.3.3.

Utilizaremos em nossa discussão um pseudo-código do Pascal para a descrição das estruturas de dados. Quando necessário, utilizaremos também diagramas e exemplos.

a) Estruturas para a Descrição dos Pinos de Entrada e Saída

As estruturas para descrição dos pinos de entrada e saída relacionam os pinos do soquete de teste com os sinais de entrada e saída do circuito sob testes. Segue abaixo o pseudo-código dessas estruturas de dados.

```
Type

      Pinagem = record              { Identificacao da pinagem do circuito }
            Num: Byte;              { Numero do pino }
            Nome: string[5];        { Nome do pino }
            EhSaida: Boolean;       { Indica se pino deve ser monitorado (saida) }
            EhEntr: Boolean;        { Indica se pino tem estimulos a aplicar }
      end;
Var
      Pinos: array[1..Maxpinos] of Pinagem;   { Descricao dos pinos do Chip }
```

Nas definições acima, a variável *Pinos*, que tem a estrutura definida pelo registro *Pinagem*, associa, para cada pino do soquete de testes, os sinais de entrada e saída para os testes.

O campo *Num* identifica o número do pino no soquete. O campo *Nome* contém o nome do sinal a ser aplicado ou monitorado. Os campos *EhSaida* e *EhEntr* identificam respectivamente se os sinais associados a um pino são de saida e/ou entrada.

b) Estrutura para a Descrição das Condições de Parada

As condições de parada definidas na programação do experimento serão armazenadas na forma de uma árvore de expressões. A estrutura *RegOp*, mostrada a seguir, representa um nó dessa árvore.

```
Type

      RegOp = Record
            Operador : Char;
            Operando1: Integer;
            Operando2: Integer;
            PEsq     : Pointer;
            PDir     : Pointer;
      end;

      RegOpPtr = ^RegOp;

Var

      CondInic: RegOpPtr;         { Cabeca da arvore para cond. inicializacao }
      CondTeste: RegOpPtr;        { Cabeca arvore p/ condicao parada do teste }
```

Na estrutura *RegOp*, cada nó representa uma expressão, que contém um operador e um ou dois operandos, representados pelos campos *Operando1* e *Operando2*. Cada um dos operandos pode ser também uma outra expresssão, nesse caso apontada pelos ponteiros *PEsq* (para o primeiro operando) ou *PDir* (para o segundo operando). Dessa forma, a árvore que define a expressão contém, nos nós que são folhas, somente expressões elementares cujos operandos são indivisíveis. Para os nós que não são folhas, pelo menos um dos operandos é uma sub-expressão.

As condições de parada são definidas por duas expressões distintas, uma delas definindo a condição de parada de inicialização e a outra definindo a condição de parada para os testes. Essas árvores são apontadas respectivamente pelos ponteiros *CondInic* e *CondTeste*.

A figura 30 apresenta um exemplo da árvore que define uma expressão de condição de parada.

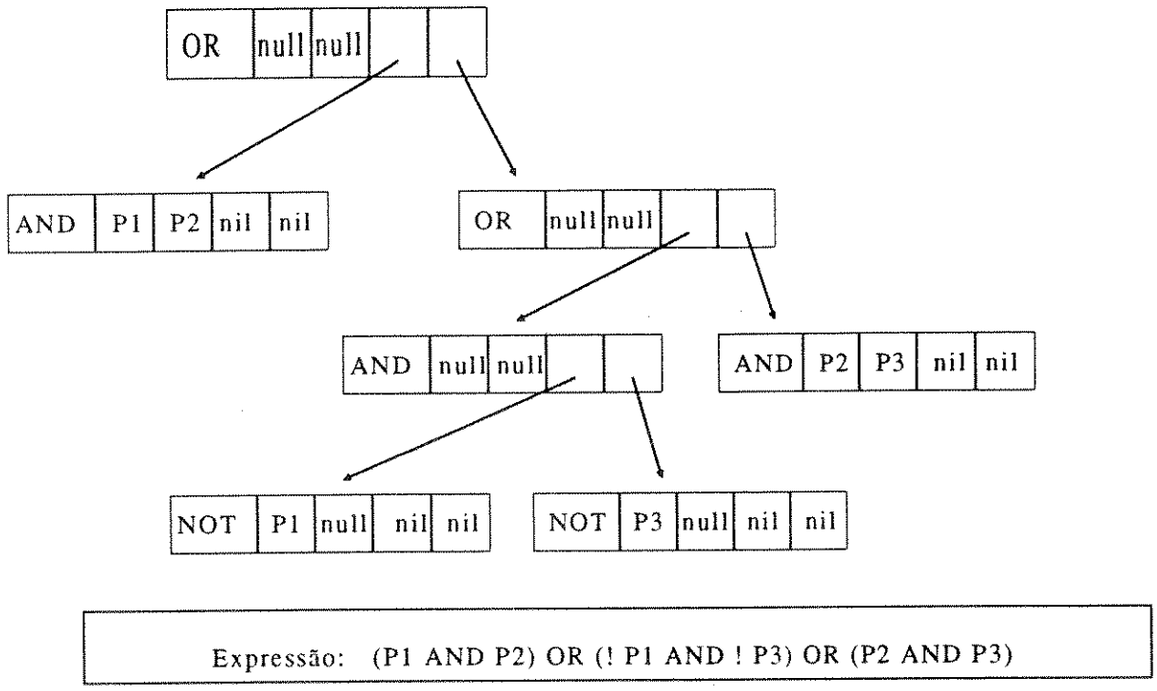


Figura 30 - Árvore para Expressões de Condições de Parada

c) Estruturas para a Descrição de Estímulos

As estruturas para descrição de estímulos são idênticas para os casos dos estímulos de inicialização do circuito e para o caso dos vetores de teste propriamente ditos. O vetor *RgEst* conterá as descrições dos estímulos a serem aplicados ao circuito. As estruturas de dados associadas são descritas a seguir.

```
Const
        MAXTRANS = 32;              { Máximo de Transições p/ sinal aperiódico }
        MAXESTIM = MAXPINOS;

Type

        RegEstim= record            { Registro para descrição de um sinal }
              Tipo: Char;           { Tipo de sinal ( A, P ou R ) }
              Pino: byte;           { Numero do pino a aplicar o sinal }
              Estado_Inic: byte;    { Estado Inicial do Sinal }
              Estado_Atual: byte;   { Estado (nivel logico) atual }
              Tempo_Inic: integer;  { Instante Inicial da aplic. sinal }
              Transic: array[1..Maxtrans] of integer;  { Inst. Chaveam. do Sinal }
              Prox_Trans: integer;  { Prox. instante para chaveamento }
              end;

        RegCondEstim= Record        { Registro p/ descricao estimulos a serem }
              Estim: Array[1..MAXEST] of RegEstim; { aplicados ao circuito }
              PCond: RegOpPtr;
              End;

Var
        RgEst : array[0..1] of RegCondEstim;
```

A variável *RgEst[0]* conterá a descrição dos estímulos definidos nos vetores de testes, enquanto *RgEst[1]* conterá a descrição dos estímulos de inicialização do circuito.

A estrutura *RegCondEstim* define, através do vetor *Estim*, o conjunto de estímulos. O campo *PCond* é um apontador para a descrição das condições de parada.

A estrutura *RegEstim* é utilizada para a descrição de um estímulo. O campo *Tipo* define o tipo de estímulos (periódico, aperiódico ou retangular). O campo *Pino* define o número do pino a ser aplicado o sinal. Os campos *Estado_Inic* e *Estado_Atual* definem respectivamente o estado lógico inicial para a aplicação do sinal e o estado lógico atual. O campo *Tempo_Inic* define o instante inicial para a aplicação do sinal. O campo *Transic* é utilizado para a descrição dos instantes de transição do sinal. Por fim, o campo *Prox_Trans* é utilizado para armazenar o instante da próxima transição do sinal.

d) Estruturas para Identificação e Acesso à Interface de Testes

As estruturas para a identificação e o acesso à interface de testes serão, em sua grande maioria, particulares a cada tipo de interface suportada pelo sistema. De uso comum a todas as interfaces, temos apenas as variáveis *CFNumPinos*, que identifica o número de pinos disponíveis para a interface; *CFTipoInterface*, que identifica o tipo de Interface; e *IdentInterface*, que define o nome da interface.

Conforme discutimos anteriormente, a versão atual do STEF suporta apenas um tipo de interface de testes [Qua88]. Para essa interface são utilizadas as variáveis *CFEnderBase*, *Porta*, *S_Contr*, *S_Estim* e *S_Entr*.

As respectivas estruturas são descritas a seguir.

```
Const
        MAXNUMPORTAS = 8-1;
        MAXBYPIN = (MAXPINOS -1) DIV 8 + 1;
Var
        CFNumPinos: Byte;          { Numero de Pinos do Circuito          }
        CFTipoInterface: Integer;  { Tipo de Interface utilizada pelo sistema }
        IdentInterface: String[16]; { Nome (identificacao) da interface   }


        CFEnderBase:Integer; { Endereco base para a placa testes digitais }
        Porta: Array [0..MaxNumPortas] of Integer;   { Portas de I/O a usar }

        S_Contr:   array[0..MAXBYPIN] of byte;     { Sinais de controle    }
        S_Estim:   array[0..MAXBYPIN] of byte;     { Sinais a Aplicar      }
        S_Entr:    array[0..MAXBYPIN] of byte;     { Sinais lidos nos pinos }
```

A interface de testes do STEF realiza o acesso através de oito das portas de entrada e saída do computador. O endereço das portas utilizadas é configurável na placa através de *dip-switches*. A variável *CFEnderBase* deverá conter o valor do endereço-base configurado. A variável *Porta* contém os valores dos dados lidos ou a serem enviados pelas portas de E/S.

As variáveis *S_Contr*, *S_Estim* e *S_Entr* conterão respectivamente os valores dos sinais de controle a serem enviados à interface, dos estímulos e dos sinais a serem lidos. Cada *bit* dessas variáveis corresponde a um pino do soquete de testes. Através da variável *S_Contr* o STEF habilita ou desabilita a leitura ou escrita dos dados para cada pino, através da variável *S_Estim* define os estados lógicos dos estímulos a aplicar e na variável *S_Entr* armazena os estados lógicos de cada pino.

e) Estruturas para Descrição dos Resultados de Testes

Os resultados de testes serão sempre armazenados em memória durante a execução dos testes. Os dados serão armazenados para todos os pinos suportados pela interface, independentemente destes serem ou não definidos como sinais a serem monitorados. As estruturas de dados que armazenam resultados de testes são utilizadas também para armazenar dados de resultados de simulações ou de testes anteriores, na operação de análises comparativas. O pseudo-código das estruturas de dados para descrição dos resultados de testes é apresentado a seguir.

```
Const
        MAXPTOSMEM = 4096;

Type
        TVetOut  = Array [0..MAXPTOSMEM] of Byte;
        VetSinais= Array [1..MAXPINOS] of Byte;

Var
        VetSai: Array[1..MAXPINOS] of ^TVetOut;
        Anter, Sinal: VetSinais;
```

As variáveis *Anter* e *Sinal* indicam respectivamente os valores lógicos dos sinais no instante anterior e no instante atual. Elas são utilizadas basicamente durante o traçado dos resultados dos testes, para o desenho das bordas de subida e descida dos sinais. Cada posição nesse vetor corresponde a um pino do soquete de testes.

A variável *VetSai* armazena os valores lógicos dos sinais monitorados pela interface de testes. Cada posição desse vetor se refere a um pino do soquete de testes e é um ponteiro para outro vetor, descrito pelo tipo *TVetOut*.

No vetor *TVetOut*, os estados lógicos em cada instante serão representados na forma de um bit. Assim, cada posição nesse vetor conterá informações referentes a oito instantes consecutivos dos testes. O armazenamento, desta forma, permite que sejam monitorados, na versão atual, até 32768 instantes em cada teste. Com a interface atual, que possui 24 canais de dados, o espaço de memória ocupado pelos vetores de resultados de testes equivale a 4096*24 = 96 kbytes.

V.3.4 - Métodos e Algorítmos Utilizados

Nesta seção procuraremos destacar os principais métodos e algoritmos implementados na atual versão do STEF. O código-fonte referente a eles é apresentado no anexo V.

Entre os algoritmos de maior interesse incluem-se, além dos discutidos a seguir, aqueles voltados ao gerenciamento de menus de opções e do sistema de auxílio ao usuário, já discutidos no capítulo IV e que serão portanto omitidos.

a) Conversão entre Programas de Simulação e Testes

A conversão entre programas de simulação e programas de testes funcionais é realizada automaticamente pelo STEF, através das operações de definição do projeto.

Ao se selecionar o nome do projeto o STEF verifica a existência de um programa de testes funcionais com aquele nome. Caso não exista, ele verifica a existência de um programa para o SIMUL de mesmo nome e, se existir, realiza automaticamente a conversão do programa, gerando um arquivo com o programa para testes funcionais.

Ao se realizar operações de definição de estímulos ou definição dos sinais de saída, o STEF atualiza automaticamente o programa de testes funcionais em disco.

Ao término de uma sessão de uso do STEF, o sistema pergunta ao aluno se ele deseja ou não gerar um novo programa para o SIMUL, a partir do programa de testes funcionais.

A conversão de programas do SIMUL para o STEF é uma simples cópia, dada a compatibilidade das linguagens. Eventualmente o aluno terá a necessidade de definir alguns comandos adicionais no programa do experimento, não incluídos nos programas do SIMUL, como a definição de estímulos e condições de parada de inicialização do circuito e comandos de controle da execução.

Para a conversão de programas do STEF para o SIMUL, o processo é ligeiramente diferente. Como temos comandos na linguagem do STEF que não têm correspondência no SIMUL, esses deverão ser suprimidos quando da conversão. Caso tivéssemos, na linguagem do SIMUL, a possibilidade de definir comentários ao programa, seria conveniente que os comandos, ao invés de suprimidos, fossem convertidos na forma de comentários, permitindo assim a posterior recuperação dessas informações. Aparentemente, a modificação no SIMUL para dar suporte a comentários seria bastante simples.

b) Interpretação e Avaliação das Condições de Parada

b.1) Interpretação das Condições de Parada

A interpretação das expressões que definem condições de parada de inicialização ou testes será realizada através da rotina *TDParseCond*. A declaração da rotina tem a seguinte forma:

```
Procedure TDParseCond(Linha:String; Operac:RegOpPtr; Var Operando:Integer);
```

O parâmetro *Linha* conterá o texto da expressão a ser avaliada, o campo *Operac* é um apontador para um nó da árvore que define a expressão e o parâmetro *Operando* retornará o valor do operando, se a expressão contiver somente um operando (isto é, se ela for um átomo).

Na ativação inicial, a rotina receberá como parâmetros a expressão completa a ser avaliada e um apontador para a cabeça da árvore da expressão.

Considerando-se que todos os operadores têm a mesma precedência, o algoritmo torna-se bastante simples, dado que a única análise de precedência a ser realizada diz respeito às sub-expresssões definidas entre parênteses.

O algoritmo da interpretação da expressão consiste na identificação do primeiro operador da expressão (considerando-se as precedências definidas por parênteses).

Caso o operador não tenha à sua esquerda nenhum operando, isso indica que ele é um operador unário (i.e., o operador de negação). Ele é então atribuído ao campo *Operador* (do registro RegCond), é criado um novo nó na árvore, o campo *PDir* apontará para esse nó, e a rotina *TDParseCond* é ativada recursivamente, tendo como parâmetros a sub-expressão que está à direita do operador e um ponteiro para o novo nó.

Caso o operador não seja unário, ele será atribuído ao campo *Operador* e serão criados dois novos nós, que serão apontados por *PEsq* e *PDir*. São então realizadas duas ativações recursivas à rotina *TDParseCond*, uma para a sub-expressão à esquerda do operador e a outra para a sub-expressão à direita do mesmo.

Caso não seja encontrado nenhum operador, a expressão é atômica e então a rotina *TDParseCond* retornará, através do parâmetro *Operando*, um valor numérico correspondente ao pino do soquete de testes definido pelo operando.

No retorno da chamadas recursivas, é verificado se o valor retornado no parâmetro *Operando* é maior que zero. Caso seja, a rotina *TDParseCond* atribuirá esse valor ao campo *Operando1* ou *Operando2* e atribuirá "NIL" ao ponteiro correspondente (*PEsq* ou *PDir*).

b.2) Avaliação das Condições de Parada

A avaliação das condições de parada é implementada através da rotina *TDAvaliaCond*, que recebe como parâmetro o identificador da expressão a ser avaliada. A declaração dessa rotina tem a seguinte forma:

```
Function TDAvaliaCond(NCond: Integer): Boolean;
```

Conforme discutimos anteriormente, as expressões que definem as condições de parada são armazenadas na forma de árvores binárias onde cada nó associa um operador a seus respectivos operandos.

A rotina *TDAvaliaCond* simplesmente ativa a rotina *TDCalcula_Nó*, passando como parâmetro o ponteiro para a cabeça da árvore da expressão. A rotina *TDCalcula_Nó* é quem efetivamente avalia a expressão.

A avaliação de um nó consiste na avaliação de cada um dos operandos seguida da aplicação do operador correspondente.

Na avaliação de um operando é verificado inicialmente se ele é ou não atômico. Se for, então é atribuído a ele o valor lógico do sinal a ele associado. Se o operando não é atômico, a rotina *TDCalcula_Nó* é ativada recursivamente, avaliando a sub-expressão definida para o operando em questão e retornando o resultado dessa avaliação.

A partir da discussão acima, podemos observar que embora a ativação seja feita a partir da raiz da árvore, a avaliação da expressão é feita a partir dos nós que são folhas da árvore, cujos operandos são sempre atômicos.

c) Acesso à Interface de Testes

O acesso à interface de testes atualmente utilizada pelo STEF é feito através das portas de entrada e saída do computador. São utilizadas oito portas de entrada e saída, com endereços consecutivos. O endereço-base é configurável através de estrapes na própria placa. A utilização de cada uma das portas é descrita na tabela 3.

PORTA	UTILIZAÇÃO
endereço base + 0	Leitura/Escrita
endereço base + 1	Leitura/Escrita
endereço base + 2	Escrita
endereço base + 3	Controle
endereço base + 4	Controle
endereço base + 5	Controle
endereço base + 6	Controle
endereço base + 7	Sem uso

Tabela 3 - Portas de E/S usadas pela Interface de Testes

Para fins de discussão, chamaremos as portas por #0 a #7.

A escrita de dados na interface (isto é, a aplicação de estímulos) é implementada através das rotinas *TDPreparaEstímulos* e *TDMandaEstímulos*. Como a aplicação de sinais é seletiva, isto é, não desesejamos aplicar sinais a todos os pinos do soquete de testes, as portas #3, #4 e #5 são utilizadas para mascarar os sinais a serem aplicados. A rotina *TDPreparaEstímulos* faz esse mascaramento, escrevendo um byte em cada uma dessas portas, sendo que cada bit corresponde a um pino do soquete de testes. Para cada bit, o valor "1" habilita a escrita e o valor "0" desabilita. Em seguida ao mascaramento dos pinos é realizada a escrita dos dados nas portas #0, #1 e #2, através da rotina *TDMandaEstímulos*. É utilizado também neste caso um bit de dado para cada pino do soquete de testes. A figura 31 apresenta os pinos associados a cada um dos bits dessas portas para a operação de aplicação de estímulos ao circuito.

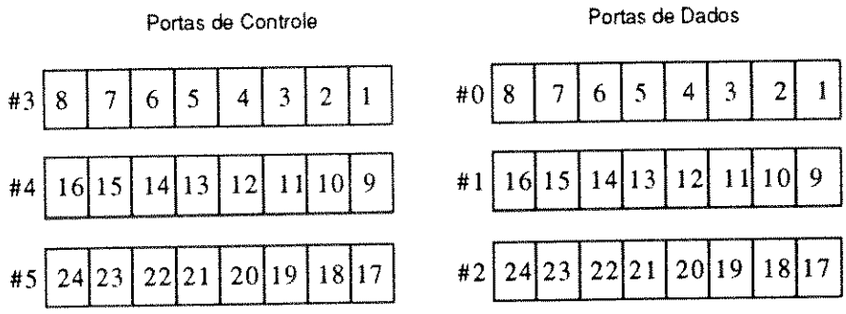


Figura 31- Pinos Associados às Portas na Operação de Escrita

A leitura de dados é realizada pela rotina *TDLeSinais*. Para a operação de leitura não é realizado o mascaramento dos sinais. Os estados lógicos dos pinos do soquete de testes são lidos das portas #0 e #1. Como podemos observar, com apenas duas portas podemos ler os estados de no máximo 16 pinos. A leitura dos dados é portanto realizada em duas etapas. Através da escrita na porta #6 é implementado o controle dos sinais a serem lidos. Em seguida à essa escrita é feita a leitura dos dados nas portas #0 e #1. A figura 32 relaciona os estados da porta #6 com os pinos lidos pelas portas #0 e #1.

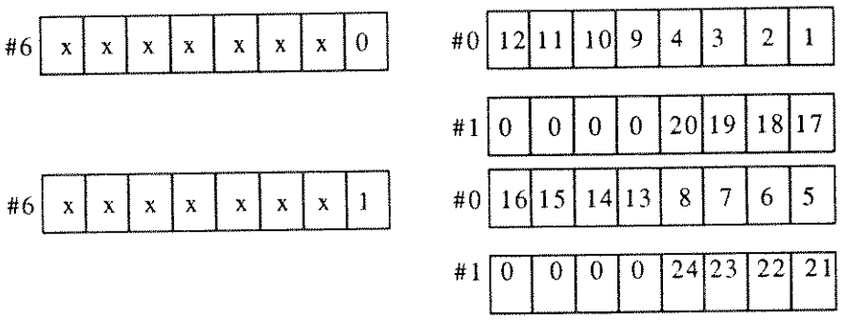


Figura 32 - Pinos Associados às Portas na Leitura dos Dados

d) Armazenamento dos Resultados de Testes

O STEF realiza o armazenamento dos resultados de testes em memória no vetor *VetSai* e, opcionalmente, em disco.

O armazenamento em memória utiliza apenas um bit por pino para cada instante de leitura. A fim de permitir maior flexibilidade em análises posteriores ao teste, são armazenados os dados lidos de todos os pinos do soquete de teste.

Caso o usuário tenha selecionado a opção de armazenamento dos resultados em disco, o STEF fará, durante a execução dos testes, a gravação dos dados num arquivo em formato binário. A fim de evitar um número excessivo de operações de escrita em disco, os dados serão gravados em blocos de 128 bytes para cada pino do circuito, o que equivale a 1024 instantes de leitura.

V.4 - Evoluções e Melhorias Futuras

Conforme discutimos anteriormente, a versão atual do STEF é um protótipo do sistema de testes funcionais e encontra-se ainda em fase de testes de integração, tendo sido implementado um conjunto de operações considerado como mínimo à utilização do sistema.

Dentre as melhorias futuras propostas para o STEF destaca-se, em especial, a inclusão de um conjunto de operações voltadas ao aconselhamento do aluno, a partir de diferenças entre resultados de simulações e dos testes. Essas funções de aconselhamento deveriam aplicar técnicas de validação de circuitos digitais e, a partir disso, indicar ao usuário componentes ou regiões do circuito suspeitos de mau-funcionamento, sugerir conjuntos de vetores de testes que auxiliem no isolamento dos problemas detectados e outras funções voltadas à depuração do circuito.

A visualização em tela do diagrama esquemático do circuito é também uma operação que poderá ser bastante útil às etapas de programação do experimento e nas análises de resultados. Na programação do experimento, o diagrama esquemático facilitaria a definição dos pontos do circuito aos quais seriam aplicados sinais ou monitorados resultados. Para as operações de análise, a simples visualização do diagrama esquemático pode auxiliar o aluno na depuração do circuito sob testes. Destacando-se visualmente regiões do circuito apontadas como suspeitas pelas operações de aconselhamento tornariam a depuração ainda mais interessante.

Com relação à interface usuário-software, o suporte a diferentes idiomas é um ponto considerado de importância. Também o sistema de auxílio ao usuário poderia sofrer modificações para permitir a inclusão de figuras, buscas por índices, etc... A utilização do STEF nos cursos de eletrônica poderá determinar também melhorias a nível de conteúdo do sistema de *help*.

O suporte a um maior número de periféricos de entrada e saída é também desejável, como diferentes modelos de impressora, suporte a *plotters*, canetas óticas e mesas digitalizadoras.

A implementação de uma versão do STEF para o ambiente *MS-Windows* é uma possibilidade em estudos. Sem dúvidas, a interface usuário-software e o suporte a periféricos de entrada e saída poderiam apresentar grandes melhorias nesse ambiente. Por outro lado, deve-se considerar que seriam exigidos para tal, maior capacidade de processamento e um monitor de vídeo com melhor resolução, o que implicaria num aumento do custo final do ambiente. Nesse caso seria ainda conveniente que todas as demais ferramentas do SAD e do SDP tivessem as suas versões para o *Windows*.

VI - Conclusões

Foram obtidos, como resultados deste trabalho:

- a descrição, em linhas gerais, de uma proposta para um conjunto de ferramentas computacionais voltadas a um ambiente didático dedicado à realização de testes e medidas em eletrônica.
- a definição de uma linguagem textual para a descrição de experimentos de testes funcionais de circuitos digitais.
- a definição do arcabouço de uma linguagem textual voltada à descrição de experimentos para medidas analógicas e paramétricas.
- a especificação de um programa de computador voltado ao tratamento e análise de resultados de medidas e simulações - o STAG - e a implementação de um protótipo desse programa, atualmente em fase de beta-testes.
- a especificação de um programa de computador destinado à descrição, execução e análise de resultados de testes funcionais, denominado STEF, para o qual foi implementado um protótipo, atualmente em fase final de integração.

A proposta do conjunto de ferramentas para o ambiente do sistema tomou como premissa básica a obtenção de um sistema de baixo custo, de forma a viabilizar sua disseminação em universidades e outras entidades de ensino, buscando-se também estabelecer uma plataforma de execução já disponível nos ambientes aos quais o sistema se destina. Buscou-se também manter compatibilidade com o ambiente de execução do Sistema Didático de Projetos - SDP [Mam87a], ao qual será agregado.

A plataforma de execução escolhida - microcomputadores da família PC-xt/AT com sistema operacional MS-DOS - atende aos requisitos propostos inicialmente. Foi proposta, ainda, a realização de estudos de viabilidade visando à implementação de uma versão do sistema para o ambiente *Windows*. Nesse ambiente poderíamos ter, entre outros, uma melhor interface com o usuário, melhor suporte no acesso a diferentes dispositivos de entrada e saída e uma melhor conectividade com outros equipamentos. A desvantagem seria o aumento do custo final do sistema, devido aos requisitos de *hardware* impostos por esse ambiente.

A linguagem de descrição de testes funcionais conseguiu prover um bom grau de compatibilidade com os programas de simulação do SIMUL e, numa primeira avaliação, mostrou-se adequada às aplicações a que se destina. Considerando-se a tendência crescente do uso de ferramentas interativas para a descrição de testes funcionais e de simulações lógicas, as linguagens textuais tendem a ser mais utilizadas para o armazenamento e intercâmbio de informações pelos sistemas, embora ainda importantes como linguagens.

Para a descrição de experimentos de medidas paramétricas e de desempenho analógico, definimos um arcabouço da linguagem, que poderá ser tomado como base para futuras implementações. As principais dificuldades a serem encontradas para a especificação de uma versão final para essa linguagem dizem respeito à obtenção de um bom grau de compatibilidade com a linguagem do SPICE.

A especificação do STAG - Sistema de Tratamento e Análise Gráfico de resultados - buscou estabelecer uma ferramenta que contenha um conjunto básico de operações voltadas aos tratamentos e análises e que esse conjunto de operações possa ser facilmente expandido.

O protótipo do STAG inclui as funções que foram consideradas essenciais à sua utilização e encontra-se atualmente em fase de beta-testes. Paralelamente a isso, têm-se trabalhado no sentido da implementação de novas funcionalidades. Atualmente os esforços têm-se concentrado principalmente na melhoria da interface usuário-software e no aprimoramento do conteúdo do sistema de *help*.

Para versões futuras do STAG, uma das principais metas será a incorporação de informações sobre modelos de dispositivos às funções de análise, oferecendo ao usuário um conjunto de funções de suporte voltadas à extração de parâmetros dos dispositivos e de dados para otimização. Outras funções atualmente previstas incluem análises de correlação entre curvas (ou amostras), incorporação de outros modelos de distribuição estatística, como o de Weibull, permitir o tratamento tridimensional de curvas, entre outros.

A versão inicial do STAG já foi incorporada ao SDP e planeja-se utilizá-la num curso de eletrônica básica (EE-722) da Faculdade de Engenharia Elétrica da Unicamp, no primeiro semestre de 1993.

Na especificação do STEF - Sistema para Testes Funcionais de Circuitos Digitais - procurou-se agregar, num único programa, um conjunto de funções que permitisse a descrição do experimento de testes, a execução dos testes funcionais e ainda um conjunto de funções voltadas à visualização e análises comparativas de resultados.

Foi implementado um protótipo do STEF, incorporando as funções básicas à sua utilização. Esse protótipo encontra-se atualmente numa etapa final de testes de integração, sendo prevista a disponibilidade de uma versão inicial (versão beta) para abril de 1993. O protótipo do STEF suporta, atualmente, um único tipo de interface para a realização dos testes [Qua88]. Em versões futuras poderá ser incorporado suporte a outros tipos de interfaces.

Sugere-se, por exemplo, para as próximas versões do sistema, a inclusão de recursos mais sofisticados para a análise de resultados. Com uma base de conhecimentos sobre métodos de validação de circuitos incorporada ao sistema, poderão estar disponíveis funções voltadas à comparação de resultados, crítica e aconselhamento do aluno, auxiliando, por exemplo na detecção de ramos do circuito suspeitos de mau-funcionamento, na geração de vetores de testes que possam isolar as regiões defeituosas, etc... Apesar da inclusão desse tipo de funções, o sistema deve preocupar-se ainda em preservar seu caráter didático.

Um primeiro protótipo do STEF, ainda sem funções para a programação interativa do experimento e para as análises comparativas de resultados, foi apresentado e utilizado durante a IV EBAI, realizada em Santiago del Estero, Argentina, em 1989.

Outras evoluções futuras, comuns ao STEF e ao STAG, são o aprimoramento da interface usuário-software e dos conteúdos dos sistemas de *help*, o suporte a um maior número de dispositivos de entrada e saída e o suporte a outras interfaces de aquisição de dados.

Glossário

Descrevemos nesta seção os principais termos utilizados neste trabalho, a fim de precisar e esclarecer a terminologia.

ASCII - *"American Standard Code for Information Interchange"*.
Tabela de conversão de caracteres alfabéticos, numéricos, símbolos e instruções de controle para código binário de 7 bits.

BNF - *"Backus-Naur Form"*.
Notação baseada em definições indutivas largamente utilizada para a especificação da sintaxe de linguagens.

Cartas Sintáticas - Vide *diagramas sintáticos*.

CGA - *"Color Graphics Adapter"*.
É um dos padrões disponíveis para interfaces controladoras de vídeo em microcomputadores da linha PC-xt/AT.

Diagramas de Conway - Vide *diagramas sintáticos*.

Diagramas Sintáticos - Notação gráfica baseada em definições indutivas utilizada para a especificação da sintaxe de linguagens.

Dip-switch - Um tipo específico de chaves liga/desliga, utilizado com frequência em placas de computadores.

DMA - *"Direct Memory Acess"*.
Acesso direto à memória. Mecanismo que possibilita que um dispositivo de entrada ou saída faça leitura ou escrita diretamente na memória.

Driver - Conjunto de rotinas quepermitem o controle de unidades periféricas individuais num computador.

EDHELP - Programa utilitário voltado à edição e manutenção de hipertextos para sistemas de auxílio ao usuário [Aff91].

EDTES - Editor de Estímulos Externos [Pan87b]. Programa que permite a definição, de forma gráfica e interativa, de estímulos externos a serem aplicados a circuitos de lógica digital. Atualmente é uma das ferramentas incorporadas ao SDP.

EGA - *"Enhanced Graphics Adapter"*. É um dos padrões disponíveis para interfaces controladoras de vídeo em microcomputadores da linha PC-xt/AT.

EXTRADOC - Programa utilitário para a extração automática de documentação de programas-fonte na linguagem Pascal [Aff92].

FFT - *"Fast Fourier Transform"*.
Transformada Rápida de Fourier.

Gramática Livre de Contexto - Classe de gramática onde as regras de transcrição para um símbolo não terminal não depende do contexto em que o símbolo se encontra.

Hardware - conjunto dos componentes e dispositivos eletrônicos, elétricos e mecânicos que compõem um computador e seus periféricos.

Help - Sistema de Auxílio ao usuário.

Hot Keys - Teclas utilizadas em programas de computador para permitir acesso rápido a determinadas operações.

Hot link - Ligação lógica entre dois contextos em um sistema de hipertexto.

IEEE-488 - Um padrão de interfaces para a conexão de dispositivos a um computador.

Mouse - Dispositivo localizador e de entrada de dados, operado através de sua movimentação sobre uma superfície.

MS-DOS - Sistema operacional para microcomputadores da linha IBM PCxt/AT, comercializado pela Microsoft, Inc.

MS-Windows - Ambiente operacional gráfico e baseado em janelas para microcomputadores da linha IBM PCxt/AT, comercializado pela Microsoft, Inc.

Net List - lista de descrição dos componentes de um circuito e das interconexões entre esses componentes.

Overflow - Ultrapassagem de um limite superior.

Parsing - Análise sintática de um texto numa determinada linguagem.

PESQA - Programa para Entrada Esquemática [Pan87c] para circuitos de lógica digital, ferramenta gráfica interativa atualmente incorporada ao SDP. Permite a definição, de forma interativa, de net-lists de circuitos para o SIMUL e para o STEF.

Plotter - traçador gráfico. Dispositivo periférico utilizado para saída gráfica de dados.

Produções - Regras de Transcrição da gramática de uma linguagem.

SAD - "Sistema de Aquisição de Dados". Ambiente composto por um conjunto de ferramentas computacionais e interfaces de aquisição de dados. Destina-se à Engenharia de Testes de Circuitos, abrangendo o planejamento, execução e análises de resultados de testes.

SDP - "Sistema Didático de Projetos". Sistema computacional voltado ao ensino de projetos em eletrônica e microeletrônica [Mam87a] [Mam87b], que integra um grande número de ferramentas voltadas à concepção de circuitos.

Shortcut Keys - Vide *"hot keys"*.

SIMUL - Simulador Lógico para CIrcuitos Digitais [Pan87a], ferramenta atualmente incorporada ao SDP e que permite a simulação de circuitos de lógica digital, considerando-se atrasos unitários.

Software - programa (ou conjunto de programas) de computador.

Source Debugger - depurador simbólico a nível de programas-fonte.

SPICE - Simulador de circuitos elétricos com ênfase em circuitos integrados [Vla81].

STAG - "Sistema de Tratamento e Análise Gráfica de Dados". Ferramenta computacional voltada à visualização, manipulação e análises gráficas de resultados de medidas analógicas e paramétricas, simulações ou tabelas de dados em geral. É uma das ferramentas que compõem o SAD.

STEF - "Sistema para Testes Funcionais de Circuitos Digitais". É o subsistema do SAD responsável pela realização das atividades de programação, execução e análises de resultados de testes de lógica funcional de circuitos digitais.

String - Cadeia de caracteres.

Tablet - Mesa digitalizadora. Tipo de dispositivo para entrada gráfica de dados.

Underflow - ultrapassagem de limite inferior.

VGA - É um dos padrões disponíveis para interfaces controladoras de vídeo em microcomputadores da linha PC-xt/AT.

Windows - Termo utilizado de maneira genérica para descrição de ambientes operacionais gráficos baseados em janelas.

Bibliografia

[Abb84] R.J.Abbott, "An Integrated Approach to Software Development", *Wiley-Interscience*, 1984.

[Aff89] M.V.Affonso, "Uma Proposta para a Padronização de Documentação de Implementação de SW", *Notas Pessoais*, 1989.

[Aff91] M.V.Affonso, "Ferramentas para Manutenção e Criação de Hipertextos", *Notas Pessoais*, 1991.

[Aff92] M.V.Affonso, "Um Utilitário Extrator de Documentação de Programas-fonte do Pascal", *Notas Pessoais*, 1992.

[Aff92a] M.V.Affonso, "STAG-Manual do Usuário", *DSIF/FEE/UNICAMP*, 1992.

[Aff92b] M.V.Affonso, "STEF-Manual do Usuário", *DSIF/FEE/UNICAMP*, 1992.

[Aho85] A.V.Aho, R.Sethi e J.D.Ullman, "Compilers: Principles, Techniques ans Tools", *Addison-Wesley*, 1985.

[Ake78] S.B.Akers, "Functional Testing with Binary Decision Diagrams", *8th Fault-Tolerant Computing Symposium*, Toulose, Jun.78, pp 75-82.

[Ake80] S.B.Akers, "A Procedure to Functional Design Verification", *10th Fault-Tolerant Computing Symposium*, Kyoto, Out.80 pp 65-67.

[Bor87] "Turbo Pascal Numerical Methods Toolbox", *Borland International Inc.*, 1987.

[Bor89] "Turbo Pascal - Reference Guide", *Borland International Inc.*, 1989.

[Bre80] M.A.Breuer e A.D.Friedman, "Functional Level Primitives in Test Generation", *IEEE Transactions on Computers*, Mar.80, pp 223-235.

[Cho59] N.Chomsky, "On Certain Formal Properties of Grammars", *Information and Control 2*, 1959.

[Dav87] M. Davis, "Análise e Projeto de Sistemas", *Livros Técnicos e Científicos Editora*, 1987

[DeM78] T. DeMarco, "Structured Analysis and Systems Specification", *NY:Yourdon*, 1978.

[Gan79] Chris Gane e Trish Sarson, "Structured Systems Analysis: Tools and Techniques", *NJ:Prentice-Hall*, 1979.

[Ges90] Rick Gessner, "Building a Hypertext System", *Dr. Dobb's Journal*, june 1990.

[Gui86] C.C.Guimarães, "Documentação de Módulos de Software" , *Comunicação Interna: PROMON*, 1986.

[IEEE84] "IEEE Guide to Software Requirements Specifications", IEEE Std 830-1984, *IEEE Computer Society*, 1984.

[Kow79] T.Kowaltowsky, "Implementação de Linguagens de Programação", *I Escola de Computação : USP*, 1979.

[Lev82] Y.H.Levendel, P.R.Menon "The *-algorithm: Critical Errors for Funtion and CDHL Construct", *IEEE Transactions on Computers*, jul.82, pp 577-588.

[Mam87] C.I.Z.Mammana e A.P.Mammana,"Introdução ao Projeto de Circuitos Integrados", *DSIF/FEE/UNICAMP*, 1987.

[Mam87a] C.I.Z.Mammana e S.H.M.Oliveira, "Manual do Sistema Didático de Projeto de Circuitos Integrados", UNICAMP/IM-CTI, 1987.

[Mam87b] C.I.Z.Mammana e S.H.M.Oliveira, "SDP: Um Sistema de Projetos para Circuitos Integrados Dedicado ao Ensino", *II Congresso da Sociedade Brasileira de Microeletrônica", 1987.*

[Pan87a] R.Pannain, "Manual do Simulador Lógico - SIMUL", *IM/CTI*, 1987

[Pan87b] R.Pannain, "Manual do Editor Gráfico de Estímulos Externos - EDTES", *IM/CTI*, 1987

[Pan87c] R.Pannain, "Manual do Programa de Entrada Esquemática - PESQA", *IM/CTI*, 1987.

[Par72] D.L. Parnas, "On the Criteria to be Used on Decomposing Systems into Modules", *Communications of ACM Vol.15, no 12.*, 1972

[Qua88] M.A.T.Quatorze, "Interface para Testes de Lógica Funcional de Dispositivos Digitais" , *Comunicação não Publicada*, 1988.

[Roc87] A.R.C.Rocha, "Análise e Projeto Estruturado de Sistemas", *Campus*, 1987.

[Rot67] J.P.Roth, W.G.Bouricius e P.R.Schneider, "Programmed Algorithms to Computer Tests to Detect and Distinguish between Failures in Logic Circuits", *IEEE Transactions on Computers*, Out.1967, pp 567-580.

[Sel68] F.F.Seller, M.Y.Hsiao e L.W.Berarnson, "Analyzing Errors with the Boolean Difference", *IEEE Transactions on Computers*, Jul.1968, pp 676-683.

[Sim91] J.O.Simões, "AIDS-TME: Ambiente Interativo para desenvolvimento de sofware para testes e medidas elétricas, utilizando instrumentação com interface GPIB (IEEE-488)", *Dissertação de Mestrado: FEE/UNICAMP*, 1991.

[Vla81] A. Vladimirescu, A. R. Newton e D. O. Peterson, "SPICE Version 2G.0 - User's Guide", *Dept. of Electrical Engineering and Computer Science*, Univ. of California, Berkeley-CA, Aug. 1981

[Wag88] F.R.Wagner et al., "Métodos de Validação de Sistemas Digitais", Campinas:UNICAMP:IMECC, 1988

[War85] P.T.Ward e S.J.Mellor, "Structured Development for Real-Time Systems", *New York:Yourdon*, 1985.

[War86] P.T.Ward e S.J.Mellor, "", *IEEE Transactions on Software Engineering*, Vol.SE-12, nº2 ,Feb/1986.

[Wei51] W. Weibull, "A Statistical Distribution Function of Wide Applicability", *Journal of Applied Mechanics*, vol 18- nº3, sept/1951.

Anexo I - Descrição das Bases de Dados para o SAD

Este documento descreve as bases de dados utilizadas pelos subsistemas do SAD atualmente implementados, ou seja, o STAG e o STEF. Serão discutidos os conteúdos de dados de cada uma delas, o formato dos dados, dispositivos de armazenamento utilizados e as técnicas de acesso empregadas.

Algumas das bases de dados terão formato comum para os diversos subsistemas, enquanto outras serão utilizadas por um único subsistema, podendo ou não ser de uso exclusivo deste. Nossa discussão será, portanto, organizada conforme os seguintes ítens:

- Bases de Dados de Uso Comum
- Bases de Dados para o STAG
- Bases de Dados para o STEF

A) Bases de Dados de Uso Comum

A.1) Bases de Dados para o Sistema de *Help*

A base de dados para o sistema de *help* do SAD conterá os textos a serem apresentados para cada uma das telas de auxílio ao usuário, bem como as informações adicionais que definem a estrutura do hipertexto.

Os dados de *help* serão armazenados em arquivos, sendo utilizado um arquivo para cada idioma de cada subsistema. Definiu-se uma padronização para os nomes dos arquivos, a fim de facilitar o gerenciamento de diferentes versões e idiomas. Os quatro primeiros caracteres do nome do arquivo deverão identificar qual o subsistema a que ele pertence, sendo seguidos de um ou mais caracteres que identificam o idioma. A extensão dos arquivos deverá ser ".HLP". Por exemplo, para o arquivo de *help* do STAG, na versão em Português, o nome do arquivo seria "STAGPORT.HLP". Para identificação do idioma deverá ser seguida a seguinte nomenclatura: "PORT", para arquivos em Português; "INGL", para arquivos em Inglês; "ESP", para o Espanhol; "FRAN" para Francês, "ITA" para o Italiano e "ALEM" para o Alemão.

Os arquivos serão organizados como uma coleção de registros de tamanho fixo, onde cada registro corresponde a um contexto de *help*. A adoção de registros de tamanho fixo não otimiza o aproveitamento de espaço em disco mas, em compensação, permite maior rapidez no acesso às telas de *help*, através de acesso direto aos registros do arquivo, evitando buscas sequenciais, indexações ou pré-processamento dos dados.

A definição, em Pascal, das estruturas de dados de um registro são descritas a seguir.

```
Const
      MaxLinHlp  = 12;
      MaxColHlp  = 50;

Type
      RegHelp= Record
           LinhaHelp: Array[1..MaxLinHlp] of String[MaxColHlp+30];
           end;
```

A estrutura acima define o registro de dados correspondente a um contexto de *help* de 12 linhas de 50 colunas cada. Note-se que na definição do vetor **LinhaHelp**, cada *string* foi definida com um espaço adicional (+30). Isso é feito para permitir que os *hot-links*, que definem as ligações de palavras-chave com outros contextos, sejam definidos juntamente com o texto.

Podemos também notar que o número do contexto não é definido explicitamente. Isso ocorre porque existe uma correpondência direta entre a posição do registro no arquivo e o número do contexto. Por exemplo, o primeiro registro se refere sempre ao contexto nº1, o n-*ésimo* registro ao n-*ésimo* contexto, etc...

Cada linha de um registro do arquivo contém o texto, com as palavras chaves marcadas por delimitadores específicos (na versão atual, os caracteres das palavras-chave têm o sétimo bit "setado") e, ao final da linha, os ponteiros para os contextos de *help* associados às palavras-chave.

A figura A1.1 ilustra a definição de um contexto do *help*. As palavras sublinhadas são as palavras que definem *hot-links*.

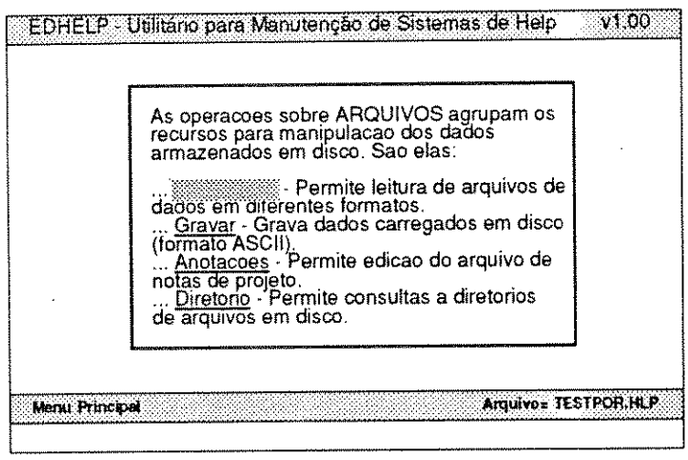


Figura A1.1 - Exemplo de um Contexto de *Help*

A.2) Bases de Dados de Configuração do Sistema

As bases de dados de configuração do sistema são armazenadas em disco em arquivos de acesso sequencial, em formato texto, sendo lidas durante o procedimento de carga do sistema e atualizadas quando da modificação de dados de configuração.

Atualmente, cada subsistema possui seu próprio arquivo de dados de configuração, embora diversas dessas informações sejam comuns a eles. A opção por manter os dados de configuração isolados entre os diversos subsistemas foi feita a fim de manter independência entre as configurações, principalmente no que se refere a cores. O uso do sistema deverá determinar os benefícios e deficiências dessa estratégia.

Para o STAG, o arquivo de configuração terá o nome de "STAG.CFG". A primeira linha do arquivo conterá a identificação do idioma a ser usado na interface usuário-software (PORT, INGL, FRAN, ESP, etc...). A segunda linha conterá os códigos decimais das cores utilizadas em tela. A terceira linha conterá informação referente ao tipo de impressora utilizada. Na quarta linha estará descrito o tipo de *plotter* disponível. A quinta linha conterá o *pathlist* do diretório de dados do sistema. A sexta linha conterá o *pathlist* do editor de textos utilizado.

Para o STEF, o nome do arquivo de configuração será "STEF.CFG". Nesse arquivo, as informações contidas nas primeiras seis linhas são as mesmas que as acima definidas para o arquivo de configuração do STAG. Além dessas informações teremos também, na sétima linha do arquivo, a informação sobre o tipo de interface de aquisição dos dados em uso pelo sistema e características da sua configuração.

Apresentamos a seguir um exemplo de arquivo de configuração para o STEF.

```
PORT
01 01 02 03 04 01 09 15 11 07 13 11 06 08 12 05
01
0
C:\TESTES\STEF
C:\EDIT\NED.EXE
14          $2F0
```

A.3) Bases de Dados de Descrição dos Menus

A base de dados de descrição dos menus contém toda a informação necessária à definição da árvore de opções do sistema. Essa base de dados estará descrita num arquivo em formato texto e de acesso sequencial. Cada subsistema terá acesso exclusivo aos seus arquivos de descrição de menus, utilizando um arquivo para cada idioma disponível.

Os nomes dos arquivos de descrição dos menus serão STEF.MNU e STAG.MNU.

Cada linha do arquivo de descrição dos menus corresponde a uma opção de menu do sistema. Cada linha conterá diversos campos de dados, separados por vírgulas, conforme mostrado no exemplo abaixo.

```
Raiz          ,  6,   1,    0,  23
Arquivos      ,  4,   7,    0,  02
Graficos      ,  9,  19,    0,  27
Analises      ,  3,  42,    0,  07
Config.       ,  2,  51,    0,  20
Help          ,  0,  -6,    0,  26
Sair          ,  2,  53,    0,  34
Ler Dados     ,  2,  11,    0,  03
Gravar        ,  0, -12,    0,  04
Anotacoes     ,  0, -13,    0,  05
Diretorio     ,  0, -14,    0,  06
Formato ASCII ,  0, -111,   0,  30
Formato SPICE ,  5,  14,    0,  29
      :          :
      :          :
```

O primeiro campo de dados contém a mensagem a ser apresentada no menu (exceto para a primeira linha, que define o menu principal). O campo seguinte define o número de sub-opções para essa opção. O terceiro campo define ou a posição de início do submenu da opção atual (se número de sub-opções era maior que zero) ou um número ao qual estará associado um procedimento do programa. O quarto campo não tem uso na versão atual, destinando-se a futuras implementacões de controle de acesso a determinadas opções. Por fim, o quinto campo define qual o contexto de *help* associado à opção atual.

Como podemos observar, a estrutura de menus de opções pode ser facilmente modificada sem que haja a necessidade de alteração e recompilação dos programas-fonte. O único caso onde isso se faria necessário seria quando da criação de novas funções, que obviamente implicariam na criação de procedimentos específicos no programa.

A adoção da descrição de menus através de arquivos em formato texto é de grande utilidade durante as primeiras versões do sistema, onde são mais frequentes modificações da interface usuário-software.

B) Bases de Dados para o STAG

B.1) Dados de Descrição do Projeto

A base de dados de descrição do projeto tem como objetivo básico reunir informações que permitam ao aluno uma melhor organização de suas atividades durante a análise de resultados de testes e simulações, através da definição de projetos.

A cada projeto definido pelo usuário estará associado um arquivo em disco, o qual conterá as informações sobre os nomes dos arquivos de dados e de anotações associados ao projeto. O nome do arquivo será o próprio nome do projeto com a extensão ".PRJ".

Os arquivos de descrição do projeto terão acesso sequencial e estarão armazenados em formato texto. Neles, cada linha conterá o nome de um arquivo. O 1º caracter da linha identificará o tipo de arquivo ("D" para arquivos de dados ou "N" para o arquivo de anotações). Em seguida à identificação do tipo de arquivo teremos o nome do arquivo propriamente dito.

```
N    EXEMPLO1.NT
D    ARQ1.OUT
D    ARQ1.SAD
D    ARQ2.OUT
D    ARQ2.SAD
```

B.2) Dados de Anotações de Projeto

Os dados de anotações de projeto serão armazenados em arquivos de formato texto e poderão ter características particulares de acordo com o editor de textos utilizado. Se for utilizado o editor de textos *default* do sistema, o arquivo obedecerá ao formato ASCII padrão, sendo permitido o uso dos caracteres da tabela ASCII extendida para a acentuação.

B.3) Dados de Entrada para o STAG (tabelas de Dados)

Os dados de entrada para o STAG serão, tipicamente, dados gerados como saídas do simulador SPICE ou dados gerados pelo subsistema de aquisição de dados do SAD. Para ambos os casos, os arquivos são gerados em formato texto (ASCII) e deverão conter as tabelas de dados a serem utilizadas pelo STAG.

O aluno poderá também desejar realizar análises de dados externos ao sistema. Nesse caso, o formato desses arquivos deverá obedecer ao mesmo formato definido para os arquivos do SAD.

B.3.1) Arquivos no Formato de Saída do SPICE

Os arquivos de saída do SPICE conterão inicialmente os dados de descrição do *net-list* do circuito e do programa da simulação, sendo seguidos pelos dados de resultados das simulações. O STAG será capaz de extrair desses arquivos tabelas de dados correspondentes aos comandos ".PRINT" contidos no programa da simulação. O usuário poderá selecionar quais as tabelas de dados a serem interpretadas. Uma restrição imposta aos arquivos em formato SPICE é que eles não poderão conter o comando ".MC" (análise de MonteCarlo, disponível no PSPICE).

Com exceção dos comandos ".PRINT" e das tabelas de resultados das simulações, todo o restante do arquivo de saída do SPICE é desprezado pelo STAG.

B.3.2) Arquivos Gerados pelo SAD

Os arquivos gerados pelo subsistema de aquisição de dados do SAD conterão um cabeçalho, onde serão descritas algumas características referentes aos testes e em seguida as tabelas de resultados das medições efetuadas. Um arquivo de dados poderá conter até dez tabelas de dados.

O arquivo será iniciado por uma ou mais linhas de cabeçalho, que serão seguidas pelas tabelas de dados propriamente ditas.

O formato do arquivo deverá obedecer às seguintes regras:

- As linhas de cabeçalho serão opcionais e, caso existam, deverão estar no início do arquivo de dados e ser iniciadas pelo caracter "$". Linhas não iniciadas pelo caracter "$" serão consideradas como linhas de tabelas de dados.
- A primeira linha da tabela de dados poderá opcionalmente conter os nomes das variáveis da tabela.
- Os nomes de variáveis deverão ser iniciados por uma letra e poderão possuir até oito caracteres alfanuméricos.
- Após a linha com os nomes das variáveis poderá existir uma linha em branco e em seguida se iniciará a tabela de dados propriamente dita.
- Os nomes das variáveis e os valores de uma mesma linha da tabela de dados serão separados por um ou mais caracteres brancos (ASCII 32) e/ou por vírgulas.

Apresentamos a seguir um exemplo de arquivo de dados gerado pelo SAD.

```
$ Exemplo de Arquivo de Dados Gerado pelo SAD
$

X       Y1    Y2    Y3
0       1     10.0  11
1       2     11.0  13
2       3     14.0  17
3       5     19.0  24
4       8     26.0  34
5       13    35.0  48
6       21    46.0  67
7       34    59.0  93
8       55    74.0  129
9       89    91.0  180
```

B.4) Tabelas de Dados Geradas pelo STAG

As tabelas de dados geradas pelo STAG são obtidas através da opção **Gravar Dados**. Como seria de se esperar, os arquivos de tabelas de dados gerados pelo STAG mantém compatibilidade, a nível de formato, com os arquivos gerados pelo subsistema de aquisição de dados do SAD.

C) Bases de Dados para o STEF

C.1) Dados de Descrição do Experimento

Os dados de descrição do experimento consistem essencialmente do programa do experimento. O programa do experimento é definido num arquivo em formato texto, segundo a linguagem para descrição de testes funcionais definida no capítulo III deste texto. A extensão dos nomes dos arquivos de programas de testes funcionais será ".TF".

Os programas do experimento podem, conforme discutimos anteriormente, ser gerados através do uso de um editor de textos, através da conversão automática de programas de simulação para programas de experimento ou ainda através da utilização de ferramentas interativas.

Outros dados relacionados à descrição do experimento são os dados de descrição esquemática do circuito. A versão atual do STEF não dá suporte à edição ou interpretação do diagrama esquemático do circuito. Em versões futuras deverão estar disponíveis, inicialmente, recursos voltados à interpretação de arquivos de descrição esquemática (por exemplo, no formato gerado pelo PESQA) e, posteriormente, funções que permitam a edição do diagrama esquemático dos circuitos.

C.2) Dados de Resultados de Simulações

Os dados de resultados de simulações serão utilizados pelo STEF para a análise comparativa de resultados dos testes funcionais com os resultados da simulação lógica do circuito.

O STEF, inicialmente, deverá ser capaz de ler apenas arquivos de resultados de simulações gerados pelo SIMUL. Caso se mostre necessário, versões futuras implementarão suporte a resultados gerados por outros simuladores lógicos.

Os arquivos de resultados de simulações gerados pelo SIMUL são arquivos em formato texto. Nele estão contidos, além dos resultados propriamente ditos, as descrições lógicas do circuito e dos macrocircuitos por ele utilizados, os estímulos externos definidos no programa para a simulação e uma lista associando cada nó do circuito com os nós para os quais ele é entrada.

Excetuando-se a descrição lógica do circuito e os dados de resultados da simulação, todas as demais informações contidas no arquivo de saída do SIMUL serão desprezados pelo STEF.

A descrição lógica do circuito obedece ao mesmo formato definido para o programa da simulação. O início da descrição do circuito é identificado pelo texto "DESCRICAO DO CIRCUITO". O final da descrição é identificado pela palavra "FIM".

Os resultados da simulação são apresentados na forma de tabela, com os estados lógicos de cada um dos sinais de saída ao longo do tempo. O nível lógico alto é representado pelo caracter "1", o nível baixo representado pelo caracter "." e o nível indefinido representado pelo caracter "?". O início da tabela é identificado por uma linha do arquivo contendo o texto "RESULTADOS DA SIMULACAO". Imediatamente após essa linha, o arquivo trará uma linha com os nomes de cada uma das variáveis (sinais) de saída e em seguida a tabela com os valores do tempo e das variáveis de saída.

Apresentamos a seguir, a título de exemplo, trechos de um arquivo de resultados de simulação.

```
DEFINICAO DA MACRO JK
TIPO        ENTRADAS                                  SAIDAS
JK          J     K     CLK   RST   SET               Q     QB
NAND        CLK   0     0     0     0                 CLB   0
NAND        J     CLB   QB    0     0                 J1    0
NAND        K     CLB   Q     0     0                 K1    0
NAND        SET   J1    K2    0     0                 J2    0
NAND        RST   K1    J2    0     0                 K2    0
NAND        J2    CLK   0     0     0                 J3    0
NAND        K2    CLK   0     0     0                 K3    0
NAND        J3    QB    SET   0     0                 Q     0
NAND        K3    Q     RSET  0     0                 QB    0
FIM
```

```
DESCRICAO DO CIRCUITO
TIPO        ENTRADAS                                  SAIDAS
NAND        MRB1      0     0     0     0             M1A   0
NAND        M1A       0     0     0     0             MRB   0
NAND        MRB1      PLB   0     0     0             S1    0
NAND        S1        0     0     0     0             S2    0
NAND        MRB       S1    0     0     0             S3    0
NAND        S1        S3    0     0     0             S4    0
JK          S2        S2    CP1   S4    S3            Q0    Q0B
NAND        MRB       S1    0     0     0             S5    0
NAND        S1        S5    0     0     0             S6    0
JK          S2        S2    Q0B   S6    S5            Q1    Q1B
NAND        MRB       S1    0     0     0             S7    0
NAND        S1        S7    0     0     0             S8    0
JK          S2        S2    Q1B   S8    S7            Q2    Q2B
NAND        MRB       S1    0     0     0             S9    0
NAND        S1        S9    0     0     0             S10   0
JK          S2        S2    Q2B   S10   S9            Q3    Q3B
FIM
```

.
.
.

```
ESTIMULOS EXTERNOS
A       MRB1      0     0     10    0     0     0     0     0
A       PLB       1     0     0     0     0     0     0     0
R       CP1       0     0     5     5     0     0     0     0
FIM
```

RESULTADOS DA SIMULAÇÃO

TEMPO	MRB1	PLB	CP1	Q3	Q2	Q1	Q0
0	.	1	.	?	?	?	?
5	.	1	1	?	?	?	?
10	1	1	.	.	.	.	.
15	1	1	1	.	.	.	.
20	1	1	.	.	.	.	1
25	1	1	1	.	.	.	1
30	1	1	.	.	.	1	.
35	1	1	1	.	.	1	.
40	1	1	.	.	.	1	1
45	1	1	1	.	.	1	1
50	1	1	.	.	1	.	.
55	1	1	1	.	1	.	.
60	1	1	.	.	1	.	1
65	1	1	1	.	1	.	1
70	1	1	.	.	1	1	.
75	1	1	1	.	1	1	.
80	1	1	.	.	1	1	1
85	1	1	1	.	1	1	1
90	1	1	.	1	.	.	.
95	1	1	1	1	.	.	.
100	1	1	.	1	.	.	1
105	1	1	1	1	.	.	1
110	1	1	.	1	.	1	.
115	1	1	1	1	.	1	.
120	1	1	.	1	.	.	.

C.3) Dados de Resultados de Testes

Os dados de resultados de testes estarão contidos em arquivos em formato texto e conterão, além dos resultados dos testes propriamente ditos, os dados de descrição do experimento (mais propriamente, o programa de testes funcionais).

O arquivo de resultados de testes conterá inicialmente um cópia idêntica do programa do experimento e, em seguida os resultados, na forma de uma tabela relacionando os estados lógicos dos sinais selecionados como saídas ao longo do tempo.

O formato da tabela de resultados é semelhante ao da tabela gerada pelo SIMUL com os resultados da simulação. As únicas diferenças de formato entre essas tabelas é que, para o STEF, não teremos o nível indefinido (representado pelo caracter "?", nos arquivos do SIMUL) e que o nível lógico baixo será representado pelo caracter "0" ao invés do caracter ".".

Anexo II - Ferramentas de Suporte ao Desenvolvimento

1) Introdução

Com o objetivo de facilitar algumas atividades relacionadas ao desenvolvimento e manutenção do SAD, foi implementado um conjunto de ferramentas automatizadas de suporte ao projeto.

Procurou-se implementar essas ferramentas de forma a permitir sua utilização pelos diversos subsistemas que compõem o SAD e também por outros sistemas que venham a ser implementados futuramente.

As ferramentas atualmente disponíveis incluem um programa utilitário voltado à criação e manutenção das bases de dados de *help* - EDHELP - e um utilitário voltado à extração automática de documentação a partir de programas-fonte, o EXTRADOC.

Descreveremos neste anexo as principais características dessas ferramentas, sob o enfoque dos recursos disponíveis e do modo de operação das mesmas.

2) Utilitário para Criação e Manutenção dos Sistemas de *Help*

2.1) Introdução

Os sistemas de auxílio ao usuário (*help*) dos subsistemas do SAD são implementados em hipertexto e de forma a serem sensíveis ao contexto operacional.

A implementação de sistemas de auxílio ao usuário na forma de hipertextos apresenta como principal vantagem em relação aos sistemas convencionais a flexibilidade contextual. Através de ligações (*"hot-links"*) entre diferentes contextos, o usuário pode "navegar" através do texto, explorando diferentes alternativas contextuais. Em outras palavras, o hipertexto permite que as informações de auxílio sejam consultadas na sequência definida pelo usuário e não de uma forma linear, como em textos convencionais.

O sistema de hipertexto implementado para o SAD é restrito apenas a informações textuais, não suportando outros tipos de informação, como figuras gráficas ou som. O hipertexto é orientado por páginas, ou seja, cada contexto é descrito por uma ou mais páginas e cada *hot-link* aponta para um única página.

O sistema de hipertexto consiste de dois módulos básicos:

- Um módulo de execução, implementado numa biblioteca de rotinas de uso comum, o qual está incorporado a cada uma das ferramentas do SAD e é responsável pelo controle da execução do sistema de auxílio ao usuário.
- Um módulo de desenvolvimento, implementado na forma de um programa utilitário (EDHELP), através do qual pode-se criar ou modificar os sistemas de *help*, e que é o objeto de nossa discussão neste anexo.

2.2) Características Gerais do EDHELP

O EDHELP é um programa utilitário executável sob sistema operacional compatível com o MS-DOS (versão 3.1 ou superior) e tem como objetivo dar suporte à criação e manutenção de sistemas de *help* dos diversos sub-sistemas do SAD ou de outros sistemas de *software*.

Além das funções básicas de criação e atualização dos hipertextos dos sistemas de *help*, o EDHELP inclui ainda algumas funções acessórias, discutidas mais adiante.

A versão atual do utilitário permite a criação de hipertextos contendo apenas informações textuais, não sendo possível a inclusão de figuras, som, etc... O conjunto de caracteres suportado restringe-se à tabela ASCII padrão, não sendo possível o uso de acentuações ou do conjunto de caracteres semi-gráficos da tabela ASCII extendida.

As funções do EDHELP podem ser classificadas nos seguintes grupos:

- Edição dos Arquivos de *Help* e Criação dos Hipertextos
- Execução de um Sistema de *Help*
- Listagem dos Arquivos de *Help*
- Consultar Ligações entre Contextos do Hipertexto

A operação de **edição dos arquivos de *help*** é realizada através de um editor de textos simples, com algumas características especiais voltadas à definição das ligações com outros contextos. Durante a edição do texto o usuário poderá, através de teclas especiais, criar ou eliminar ligações de palavras do texto com outros contextos (ou seja, definir *hot-links* entre contextos).

Através da **execução de um sistema de *help*** o usuário pode verificar, a nível de forma e conteúdo, um sistema de *help*. A execução do sistema de *help* é feita de forma idêntica àquela que será realizada quando da integração do *help* ao sistema-alvo.

A operação de **listagem dos arquivos de *help*** permite que o usuário obtenha, em tela ou impressora, uma listagem completa de um sistema de *help* ou parte dele. As palavras definidas como *hot-links* são mostradas em destaque (negrito, se listagem em tela ou sublinhadas, se listagem na impressora). São mostrados também nessa listagem os números dos contextos de *help* associados a cada *hot-link*.

Através da operação de **consulta a ligações entre contextos** é possível obter, em tela ou impressora, uma relação de todas as ligações entre contextos definidas num hipertexto. Para cada contexto serão apresentadas todas as palavras-chave que definem *hot-links* e os contextos associados a cada uma delas.

2.3) Usando o EDHELP

2.3.1) Ativação do Utilitário

Para ativar o EDHELP o usuário deverá, a partir do *"prompt"* do sistema operacional, digitar a seguinte linha de comando:

EDHELP [*nomarq*]

onde

nomarq é um parâmetro opcional que define o nome de arquivo de *help* a ser usado.

Após ser ativado, o EDHELP apresentará ao usuário sua tela inicial, com o menu principal de operação do sistema.

2.3.2) Operações Básicas

As operações básicas do EDHELP são acionadas a partir de menus de opções. Ao ser ativado, o utilitário apresenta ao usuário o menu principal, conforme mostrado na figura A2.1.

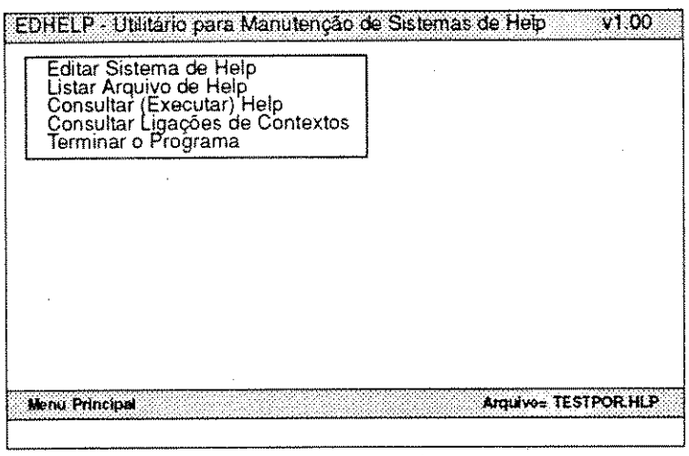


Figura A2.01 - Tela Principal do EDHELP

Discutiremos a seguir cada uma das opções disponíveis no menu principal.

2.3.2.1) Editar Sistema de *Help*

A operação **editar sistema de *help*** permite ao usuário modificar um arquivo de *help* existente ou criar um novo arquivo.

Ao selecionar esta opção, o EDHELP solicitará ao usuário o nome do arquivo a ser editado e, em seguida, entrará no modo edição. O modo de edição é composto basicamente por um editor de textos bastante simples, mas com algumas características especiais voltadas à definição das ligações entre contextos (*hot-links*). A figura A2.2 mostra a tela apresentada ao usuário para a edição de um arquivo de *help*.

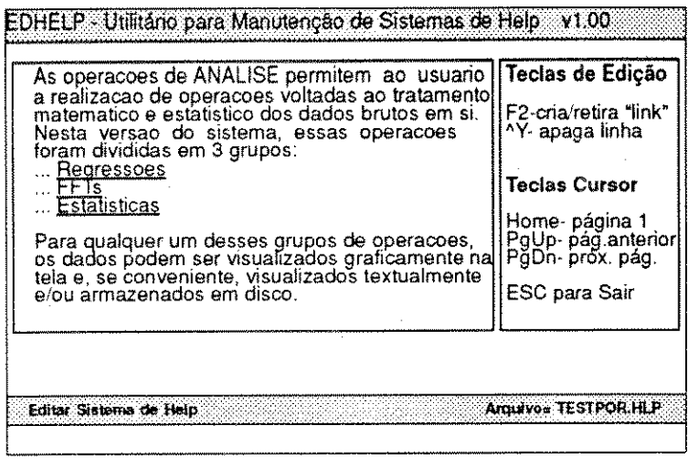


Figura A2.02 - Tela para Operação de Edição de um Hipertexto

Como podemos observar através dessa figura, as funções de edição consistem apenas das operações basicas de movimentação do cursor e da inserção ou deleção de caracteres ou de linhas. Através da tecla <F2> o usuário pode criar ou remover *hot-links* com outros contextos.

Para a criação de um *hot-link*, o usuário deve posicionar o cursor sobre a palavra que ele deseja ligar com outro contexto e em seguida teclar <F2>. O EDHELP mostrará então essa palavra em destaque e, solicitará ao usuário o número do contexto que ele deseja ligar com a palavra marcada, retornando em seguida ao modo normal de edição.

Para eliminar um *hot-link*, a operação consiste em posicionar o cursor sobre a palavra marcada como *hot-link* e simplesmente teclar <F2>. Após teclar-se <F2> a palavra é novamente mostrada com caracteres normais.

Para mover-se para outro contexto de *help*, o usuário deverá utilizar as teclas <PgUp> e <PgDn>. Teclando-se <ESC>, o sistema termina a edição do hipertexto, retornando ao menu principal.

2.3.2.2) Listar Arquivos de Help

A operação de listagem dos arquivos de *help* permite que o usuário obtenha, em tela ou impressora, uma listagem (completa ou parcial) de um determinado hipertexto.

Ao selecionar-se esta opção, será apresentado ao usuário um submenu com as opções para o dispositivo de saída para a listagem (tela ou impressora). Após selecionado o dispositivo de saída, é solicitado ainda o nome do arquivo, emitindo então a listagem.

A seguir é apresentado, como exemplo, um trecho de listagem gerada pelo EDHELP.

Pag: 1

INDICE

Selecione abaixo o topico de seu interesse:
.. Conceitos Basicos
.. Operacoes sobre Arquivos
.. Operacoes para Tratamento Grafico
.. Analises Matematicas e Estatisticas
.. Configuracao do Sistema
.. Sistema de Auxilio ao Usuario
.. Informacoes Adicionais

<Alt-F1> mostra as teclas de funcoes disponiveis

Pag: 2

As operacoes sobre ARQUIVOS agrupam os
recursos para manipulacao de dados armazenados
em disco. Sao elas:
... Ler Dados - Permite leitura de arquivos de
dados em diferentes formatos.
... Gravar - Grava dados carregados em disco
(formato ASCII).
... Anotacoes - Permite edicao do arquivo de
notas de projeto.
... Diretorio - Permite consultas a diretorios
de arquivos em disco.

Pag: 5

A operacao de ANOTACOES permite ao usuario criar e modificar, usando um editor de textos, dados historicos e de observacoes realizadas durante uma sessao de uso do sistema.

Nesta versao do sistema, esta opcao nao se encontra disponivel ainda.

Pag: 7

As operacoes de ANALISE permitem ao usuario a realizacao das operacoes voltadas ao tratamento matematico e estatistico dos dados brutos em si. Nesta versao do sistema, essas operacoes foram divididas em 3 grupos:
... Regressoes
... FFTs
... Estatisticas
Para qualquer um desses grupos de operacoes, os dados podem ser visualizados graficamente na tela e, se conveniente, visualizados textualmente e/ou armazenados em disco.

Pag: 8

Quatro tipos de REGRESSOES estao disponiveis ao usuario nesta versao do sistema:
.. Regressao Linear
.. Regressao Geometrica
.. Regressao Exponencial
.. Regressao Polinomial

Para qualquer tipo de regressao selecionado, o usuario devera' escolher a variavel sobre a qual sera' aplicada a regressao.
Atraves de hot-keys e' possivel visualizar os resultados numericos e armazena-los em disco.

Pag: 23

Os menus de opcoes sao a principal forma de interacao do usuario com o sistema. Os usuarios mais familiarizados com o sistema poderao achar mais eficiente o acesso 'as funcoes atraves das teclas de funcao (hot-keys).
As operacoes foram agrupadas nos menus em cinco diferentes classes:
.. Arquivos
.. Graficos
.. Analises
.. Configuracao
.. Help

Pag: 27

A opcao GRAFICOS agrupa os comandos relativos ao tratamento de visualizacao grafica dos dados gerados. Sao disponiveis as seguintes opcoes:

.. Regua
.. Zoom
.. Var.Grafico
.. Tipo Tracado
.. Imprimir.
.. Grade
.. Tipo Cursor
.. Var.Eixo X
.. Escalas

Para detalhes sobre essas opcoes, selecione a palavra-chave desejada.

Pag: 45

As Hot-Keys disponiveis neste instante sao:

F1 - Help	F9 - Exporta Dados
F2 - Grade	F10 - Imprimir
F3 - Liga Zoom	Alt-F1 - Mostra Hot-Keys
F4 - Desliga Zoom	Alt-F2 - Linha Status
F5 - Var. Grafico	Alt-F3 - Tipo de Cursor
F6 - Var. Eixo X	Alt-F4 - Tipo de Escala
F7 - Tipo Tracado	Alt-F5 - Def. Parametros
F8 - Analises	Alt-F10 - Config. Cores
INS - Liga Regua	DEL-Desliga Regua

(Modo: TRATAMENTO)

2.3.2.3) Executar (Consultar) Sistema de Help

Ao selecionar-se a operação de execução de um sistema de help, o EDHELP solicitará o nome do arquivo a ser lido e em seguida realizará a execução do mesmo, a partir do contexto inicial (que é, em geral o índice do hipertexto). A próxima versão do EDHELP deverá permitir também a execução a partir de um contexto qualquer que não o inicial.

Esta operação é de grande utilidade na verificação de um sistema de *help* já existente. A figura A2.3 mostra o formato geral da tela para esta operação. Podemos verificar nessa figura que as palavras definidas como *hot-links* aparecem sublinhadas na tela. Uma dessas palavras é mostrada em vídeo reverso, identificando a palavra atualmente selecionada. Movendo o cursor o usuário "caminha" pelas palavras selecionadas e teclando <ENTER> ele ativa o contexto de *help* associado à palavra selecionada. Através da tecla <PgUp> pode-se retornar ao contexto anterior. A tecla <ESC> encerra uma sessão de consulta (execução) ao *help.*

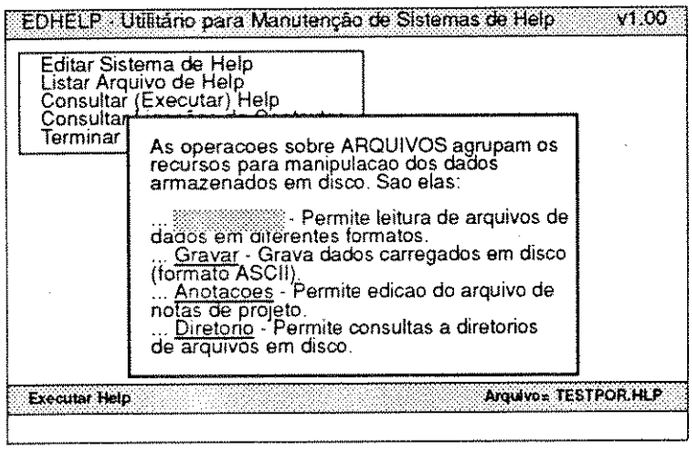


Figura A2.3 - Operação de Execução de um Sistema de *Help*

2.3.2.4) Consultar Ligações entre Contextos

A operação de consulta a ligações entre contextos permite que o usuário obtenha uma listagem de referências cruzadas entre todos os contextos do hipertexto.

Selecionando-se esta opção no menu principal, será apresentado um submenu para a seleção do dispositivo de saída a ser utilizado para a listagem (tela, impressora ou disco). Em seguida o EDHELP solicitará ao usuário o nome do arquivo de entrada a ser utilizado e fará a geração de uma listagem conforme o exemplo mostrado a seguir.

Contexto: 01	Palavra-chave	Contexto Associado
	Conceitos	18
	Definicao do Projeto	64
	Arquivos	2
	Tratamento Grafico	27
	Analises	7
	Configuracao	20
	Auxilio ao Usuario	26
	Informacoes	19

Contexto: 02	Palavra-chave	Contexto Associado
	Ler Dados	3
	Gravar	4
	Anotacoes	5
	notas de projeto	28
	Diretorio	6

Contexto: 03	Palavra-chave	Contexto Associado
	SPICE	29
	ASCII	30

Contexto: 04	Palavra-chave	Contexto Associado
	ASCII	30
	Zoom	32

Contexto: 05 NENHUMA LIGAÇÃO com outros contextos

2.3.2.5) Terminar o programa

Através desta opção o usuário pode terminar uma sessão de uso do sistema. Ao selecionar-se esta opção, será mostrada no centro da tela uma mensagem solicitando a confirmação da opção. Caso o usuário confirme, o programa é terminado.

Uma forma alternativa de abandonar o EDHELP é teclando-se <ESC> a partir do menu principal.

3) Extrator Automático de Documentação

3.1) Introdução

A documentação da implementação de sistemas de *software* é um dos aspectos de maior importância, no que se refere à qualidade e facilidade de manutenção dos mesmos.

Com o objetivo de facilitar a geração e manutenção da documentação da implementação dos sistemas, propusemos um conjunto de regras para documentação em linha dos programas, buscando obter uma documentação mais uniforme, concisa e coerente [Aff89].

A partir desse conjunto de regras, foi criado um programa utilitário dedicado à extração automática da documentação dos programas-fonte, o EXTRADOC. Além da criação dos documentos, o utilitário é capaz também de atualizar a documentação, facilitando o gerenciamento de versões do sistema.

Devemos ressaltar que a confiabilidade e eficiência do uso do utilitário estará estritamente ligada à obediência às regras propostas para a padronizacão da documentação nos programas-fonte.

3.2) Características Gerais do EXTRADOC

O EXTRADOC é um programa utilitário executável sob sistema operacional compatível com MS-DOS versão 3.1 ou superior, tendo como objetivo a extração automática de documentação dos programas-fonte de sistemas de *software*.

A extração automática da documentação abrange a descrição de módulos de implementação de um sistema e a descrição das rotinas de cada módulo. Para que seja possível a extração automática, é necessária a definição de cabeçalhos de módulos e rotinas num formato padronizado.

Deverá ser definida também, num arquivo texto, a relação de todos os arquivos contendo os programas-fonte relacionados ao sistema. Esse arquivo é chamado de **arquivo de identificação do sistema**. O EXTRADOC extrairá os cabeçalhos de módulos e rotinas de todos os arquivos de programa relacionados no arquivo de identificação do sistema.

Os cabeçalhos padronizados são definidos na forma de comentários com delimitadores especiais incluídos nos programas-fonte. Utiliza-se o delimitador "+*+" para indicar o início de um trecho de comentário a ser extraído e o delimitador "-*-" para indicar o término do mesmo. Com vistas à facilidade de inclusão e uma melhor padronização dos comentários a serem extraídos, foram criados arquivos específicos contendo os "esqueletos" (*templates*) das descrições de módulos e rotinas. Apresentamos a seguir o conteúdo desses arquivos, com uma breve descrição de cada um dos campos a serem preenchidos.

Deve-se observar que as regras propostas para a documentação de programas-fonte não se restringem apenas ao uso dos cabeçalhos padronizados. São propostas também outras convenções para a definição de identificadores e nomes de rotinas e variáveis e para a codificação em si.

3.2.1) Convenções para Cabeçalhos de Módulos

Os cabeçalhos de módulos possuem o seguinte formato:

```
(*****************************************************************
+*+
 * Nome_modulo -              (Descricao em uma linha)
 *
 * CRIADO EM:  dd-mes-aa  POR: (nome)
 *
 * ULT.EDICAO: dd-mes-aa  POR: (nome)
 *
 *      Descricao Funcional
 *
 * ROTINAS EXPORTADAS:
 *
 *      nome            Descricao em uma Linha
 *
 * ROTINAS IMPORTADAS:
 *
 *      nome
 *
 * ARQUIVOS INCLUIDOS:
 *
 *      nome
 *
 * PARAMETROS DE GERACAO:
 *
 *      descricao
 *
 * ESTRUTURAS DE DADOS:
 *
 *      nome            Descricao em uma linha
-*-
 * NOTAS:
 *
 * USO INDEVIDO:
 *
 * ATUALIZACOES:
 *
 * Revisao   dd-mes-aa  Nome       Descricao
 *

*****************************************************************)
```

Comentários:

- somente os campos marcados como opcionais poderão ser omitidos. Nesse caso deverá ser omitido também o nome do campo.
- o par de delimitadores "+*+" e "-*-" marcam respectivamente o início e o final de uma região de comentário a ser extraída.
- ROTINAS EXPORTADAS: Este campo deve conter apenas uma descrição sucinta das rotinas exportadas. A descrição detalhada será apresentada no cabeçalho da própria rotina.
- ROTINAS IMPORTADAS: Deverão ser descritos somente os nomes dessas rotinas.
- ARQUIVOS INCLUIDOS: Relação dos nomes dos arquivos incluídos.
- PARAMETROS DE GERAÇÃO: Deve incluir os parâmetros para compilação e nomes de constantes definidas no módulo cujos valores são importantes para o usuário (p.ex., tamanhos de tabelas, etc...).
- ESTRUTURAS DE DADOS: Descrição sucinta das estruturas de dados importantes e que sejam compartilhadas por diversas rotinas.
- NOTAS: Este campo descreverá características especiais de implementação, restrições de uso, *bugs* conhecidos e quaisquer outras notas importantes ao usuário.
- USO INDEVIDO: Devem ser descritos neste campo os modos como as rotinas exportadas não devem ser utilizadas. destacando-se as chamadas indevidas.
- ATUALIZAÇÕES: Este campo deve conter o histórico das modificações realizadas, identificando-se a data, o nome do responsável pelas modificações e os motivos destas.

3.2.2) Convenções para Cabeçalhos de Rotinas

Os cabeçalhos de rotinas possuem o seguinte formato:

```
(*************************************************************************
+*+
 * MMnomerot -    (Descricao em uma linha)
 *
 *            Descricao          (opcional)
 *
 * CHAMADA PADRAO:
 *
 *            ValRet := MMnomerot(Par1, Par2, ..., Parn);
 *
 *            Nome       i/o    tipo          Descricao
 *
 * RETORNA:                     (usado somente para funcoes)
 *
 * SAIDA DE EXCECAO:      (opcional)
 *
 *      Descricao das Saidas Excepcionais
 *
 * OUTRAS ENTRADAS:       (opcional)
 *
 *      Dados Externos ou Globais Utilizados (inclusive arquivos)
 *
 * OUTRAS SAIDAS:         (opcional)
 *
 *      Dados Externos ou globais Alterados
 *
 * ROTINAS CHAMADAS:
 *
 *      nome              descricao
 *
-*-
 * NOTAS:                       (opcional)
 *
*************************************************************************)
```

Comentários:

- NOME DA ROTINA: Este campo contém o nome da rotina seguido de uma breve descrição da mesma. Na definição dos nomes de rotinas, os dois primeiros caracteres devem identificar o módulo do sistema ao qual ela pertence.
- DESCRIÇÃO DA ROTINA: Este campo deve conter uma descrição detalhada da rotina.

- CHAMADA PADRÃO: Deve conter o formato padrão de ativação da rotina. Na descrição dos parâmetros da rotina aparecem os campos **nome**, **i/o**, **tipo** e **descrição**. Esses campos são obrigatórios e conterão as seguinte informações:
  - **nome:** nome do parâmetro formal.
  - **i/o:** tipo de acesso ao parâmetro (i=entrada, o=saída, i/o=entrada e saída).
  - **tipo:** tipo do parâmetro (inteiro, real, string, etc...)
  - **descrição:** descrição sucinta do parâmetro. Se valores especiais forem relevantes para o parâmetro, eles também deverão ser descritos.
- RETORNA: Este campo é usado somente para funções. Se a função retorna um *status*, devem ser descritos os possíveis valores. Se retorna valores, devem ser descritos os valores esperados.
- SAIDA DE EXCEÇÃO: Este campo é usado para descrever condições excepcionais que possam ocorrer, e que provoquem um término prematuro da rotina.
- OUTRAS ENTRADAS: Deverá conter uma relação dos dados globais ou externos utilizados pela rotina.
- OUTRAS SAIDAS: Deve conter a relação de dados externos ou globais (inclusive arquivos e mensagens) que possam ser alterados pelas rotinas.
- ROTINAS CHAMADAS: Deve conter uma relação dos nomes da rotinas chamadas.
- NOTAS: Deve conter informações relevantes sobre a rotina, como restrições, *bugs* conhecidos, etc...

3.3) Usando o EXTRADOC

O EXTRADOC realiza a extração da documentação dos programas fonte com base num arquivo que relaciona os nomes de todos os programas-fonte pertencentes a um sistema, chamado de **arquivo de identificação do sistema**.

Antes de ativar o utilitário, o usuário deve criar, através de um editor de textos, esse arquivo. As três primeiras linhas do arquivo deverão conter respectivamente o nome do sistema, uma descrição sucinta e a versão do sistema. A versão da documentação será definida automaticamente pelo EXTRADOC. A partir da quarta linha, deverão ser descritos os nomes dos arquivos que contém os programas-fonte do sistema. Aconselha-se que na definição dos nomes dos arquivos seja informado o *pathlist* completo dos mesmos. Para o nome do arquivo de identificação do sistema é utilizada a extensão ".IDS" como *default*.

3.3.1) Ativando o Utilitário

Para ativar o EXTRADOC, o usuário deverá, a partir do *prompt* do sistema operacional, digitar:

EXTRADOC [*nomearq*]

onde

nomearq é um parâmetro opcional que define o nome do arquivo de identificação do sistema a ser utilizado para a geração da documentação.

3.3.2) Operações Básicas

Ao ser carregado o sistema, é apresentada ao usuário uma tela contendo o menu principal de opções, conforme mostrado na figura A2.4.

Discutiremos a seguir as operações realizadas pelo utilitário para cada uma das três opções disponíveis no menu principal.

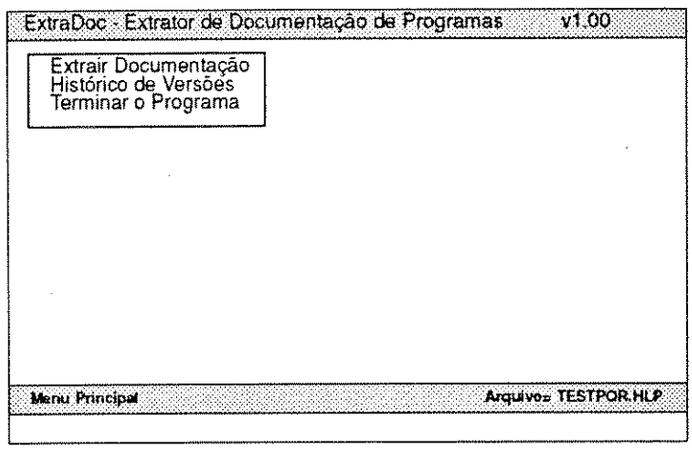


Figura A2.04 - Tela Principal do EXTRADOC

3.3.2.1) Extrair Documentação

Ao ser ativada a opção **extrair documentação** , será solicitado ao usuário o nome do arquivo de identificação do sistema e, a partir desse arquivo, o utilitário realizará a extração e atualização automatica da documentação do sistema.

A documentação extraída é armazenada num arquivo com nome idêntico ao nome do arquivo de identificação do sistema, sendo utilizada a extensão ".DOC". Na versão atual do EXTRADOC, cada vez que é solicitada esta operação, é feita uma atualização completa da documentação. Em versões futuras, pretende-se que o utilitário faça atualizações somente das informações que sofreram modificação, tornando mais rápida a geração do documento.

A documentação gerada é composta por três partes principais:

- Um Indice do Conteúdo do Documento
- A Descrição de Cada um dos Módulos do Sistema
- A Descrição das Rotinas de Cada Módulo

O índice de conteúdo do documento relacionará todos os nomes de módulos e das rotinas de cada um desses módulos e as respectivas páginas no documento. Os tópicos principais serão constituídos pelos nomes de módulos. Os nomes das rotinas serão tratados como sub-tópicos.

As descrições dos módulos estarão relacionadas em ordem alfabética. A descrição de um módulo será seguida imediatamente das descrições das rotinas contidas no módulo.

As descrições das rotinas de cada módulo estarão também organizadas em ordem alfabética, para cada módulo.

Em versões futuras do EXTRADOC pretende-se criar uma sessão adicional, contendo uma tabela de referências cruzadas entre rotinas.

Mostramos a seguir, um exemplo de trechos da documentação gerada através do EXTRADOC.

SISTEMA: Exemplo1
Versão: 1.05

Exemplo de Documentação Gerada pelo EXTRADOC

Data da Geração: 12.out.92

Conteúdo

Módulo BGGRAF.PAS

Biblioteca basica de rotinas graficas para tela.

Criado em: 04.ago.90 por: Mauricio

Ult.Edição: 14.dez.92 por: Mauricio

ROTINAS EXPORTADAS:

BGApita - Emite um apito para avisar alguma coisa ao usuario
BGChk_IO - Checa se ultima operacao de I/O realizada foi bem sucedida
BGCabectela - Escreve cabecalho da tela grafica.
•
•
•
BGMensagemGr - Manda uma mensagem no "footer" da tela grafica
BGMenu - Gerenciamento dos Menus de Opcoes do Sistema
BGMoldura - Faz moldura na tela , na CorMold, preenchida com CorFundo
BGPedeNomArq - Pede nome do arquivo de entrada em modo grafico
BGRestauraJanela - Restaura uma janela em modo grafico
BGReverso - Coloca uma regiao do video em reverso - Coorden. Graficas
BGReversoTxt - Coloca uma regiao do video em reverso - Coorden. Texto
BGSaiTexto - Coloca um texto na tela c/ coordenadas texto
BGSaiTextXY - Mostra string na tela a partir das coord. (graficas) dadas
BGSalvaJanela - Salva uma Janela em modo grafico (coord. sao relativas)
BGTrim - Retira os brancos 'a direita de uma string
BGUpper - Transforma uma String em maiusculas

ROTINAS IMPORTADAS:

As das bibliotecas do compilador (Dos, Graph, Crt).

PARAMETROS DE GERACAO:

Verificar no arquivo BGVARS.pas os tamanhos maximos das tabelas de dados para menus de opcoes.

ESTRUTURAS DE DADOS:

Descritas no módulo BGVARS

BGFinalizaMenu

Libera memoria dinamica alocada para estruturas dos menus de opcoes.

CHAMADA PADRAO:

BGFinalizaMenu;

OUTRAS SAIDAS: (Dados Externos ou globais Alterados)

InicializouMenu

BGInicMenu

Aloca memoria e inicializa as estruturas de dados relacionadas aos menus de opcao do sistema

CHAMADA PADRAO:

BGInicMenu;

OUTRAS SAIDAS: (Dados globais Alterados)

InicializouMenu

BGMenu

Faz o gerenciamento dos menus de opcoes.

CHAMADA PADRAO:

Opcao:= BGMenu(Col, Lin, NOpc, Ind)

Nome	i/o	tipo	Descricao
Col	i	inteiro	Coluna inicial p/ menu (coord. texto)
Lin	i	inteiro	Linha inicial p/ menu (coord. texto)
NOpc	i	inteiro	Numero de opcoes do menu
Ind	i	inteiro	Posicao inicial no vetor Menu

RETORNA:

< 0, caso tenha sido executada uma operacao que e' folha da arvore de menus (Foi_nas_folhas = True).
0, caso o usuario tecle <ESC> (retornar para o nivel anterior do menu)
> 0, caso o usuario tenha selecionado uma opcao valida do menu que nao seja folha da arvore de opcoes. Neste caso o valor retornado corresponde ao item selecionado no menu, ou seja, retorna "1" caso o usuario selecione a primeira opcao do menu, "2" para a segunda opcao, etc...

OUTRAS ENTRADAS: (dados globais utilizados)

CurHlp - Identifica o contexto atual para o sistema de help
Foi_nas_folhas - Indica se chegou as folhas da arvore de menus
Menu - Contem a descricao da arvore dos menus de opcoes

OUTRAS SAIDAS: (dados globais alterados)

CurHlp
Foi_nas_folhas

ROTINAS CHAMADAS:

BGExecMenu	Executa um operacao que e' folha da arvore de menus
BGLeChar	Leitura de uma tecla do teclado.
BGMensagemGr	Escreve mensagem no rodape da tela.
BGMenu	... esta rotina ...
BGMoldura	Desenha a moldura para o menu de opcoes.
BGOpcaoMenu	Permite a escolha de uma opcao dos menus.
BGRestauraJanela	Restaura regiao da tela previamente salva.
BGSaiTexto	Escreve uma String numa determinada posicao da tela.
BGSalvaJanela	Salva conteudo de uma determinada regiao da tela.

3.3.2.2) Histórico de Versões

A opção **histórico de versões** permite que o usuário obtenha, em tela ou na impressora, um listagem histórica de todas as atualizações de versões da documentação geradas através do EXTRADOC.

Essa listagem é gerada a partir de informações armazenadas pelo utilitário num arquivo de registro do histórico de versões. O nome desse arquivo será o mesmo nome usado para os arquivos de identificação do sistema e de documentação, sendo utilizada a extensão ".HIS". Os arquivos de histórico de versões terão formato texto.

A versão atual do EXTRADOC ainda não incorpora a opção de consulta ao histórico de versões.

3.3.2.3) Terminar o programa

Através desta opção o usuário pode terminar uma sessão de uso do sistema. Ao selecionar-se esta opção, será mostrada no centro da tela uma mensagem solicitando a confirmação da opção. Caso o usuário confirme, o programa é terminado.

Alternativamente o usuário pode abandonar o EXTRADOC teclando <ESC> a partir do menu principal.

STAG - Sistema de Tratamento e Análise Gráfica de Dados

Manual do Usuário

Versão 1.00 - Janeiro/1993

Edição Preliminar

Revisão: B

Maurício de Vasconcelos Affonso

Índice

I - Introdução

O Sistema de Tratamento e Análise Gráfica de Dados - STAG tem como objetivo permitir a visualização e manipulação gráfica de dados obtidos através de ferramentas do SAD ou de outros sistemas.

Procurou-se estabelecer um conjunto de recursos que possibilitem ao usuário manipular livremente os resultados dos testes, oferecendo assim maior facilidade para a análise visual dos resultados.

Sob o enfoque da análise dos resultados, são oferecidas operações voltadas ao tratamento matemático e estatístico dos dados brutos, como regressões, transformadas de Fourier e análises estatísticas de tendência central, dispersão e distribuição de frequências. Através das operações de análise, o usuário poderá verificar aspectos do comportamento de desempenho de circuitos ou dispositivos sob estudo e, de maneira indireta, realizar a extração de parâmetros e dados para otimização dos mesmos.

Quaisquer das operações disponíveis no STAG podem ser selecionadas através de menus de opções, e as mais frequentemente utilizadas serão acessíveis também através de teclas especiais(*hot-keys*).

Encontra-se disponível também um mecanismo de auxílio ao usuário sensível ao contexto e na forma de hipertexto, permitindo que a qualquer instante sejam obtidas informações de auxílio referentes ao contexto atual do sistema e a outros tópicos associados a ele.

Descreveremos inicialmente neste manual alguns conceitos e procedimentos básicos à operação do sistema, passando em seguida à descrição geral do sistema e operação do mesmo.

II.1 - Definindo as Características do Ambiente

Para que o STAG faça uso pleno de seus recursos, o seu sistema deverá ter definidas algumas características ambientais essenciais. Essas características são descritas no arquivo CONFIG.SYS e são ativadas durante a carga do sistema operacional (*ie*, no *boot* do sistema).

Antes de utilizar o STAG, você deve certificar-se de que o arquivo CONFIG.SYS contém as seguintes definições:

```
FILES = 20
BUFFERS = 24
```

Os valores descritos acima são os valores mínimos para as variáveis do ambiente. Valores maiores não deverão causar qualquer tipo de problema.

Se você possui um *mouse* e deseja utilizá-lo, é necessária a instalação de um "*driver*" para emulação de teclado. O botão da esquerda deve ser "setado" para a tecla <CR > e o da direita para a tecla <ESC >.

II.2 - Instalando o Sistema

Para instalar o sistema no disco rígido você deverá seguir os seguinte passos:

1) Coloque o disquette contendo o STAG no drive "A:"

2) Digite: **A:INSTSTAG** <ENTER >

3) A partir deste ponto, o programa de instalação fará uma série de perguntas sobre parâmetros de instalação, como a unidade de disco e o diretório onde o STAG deve ser instalado. Ao término da instalação será dada a mensagem **"STAG instalado com sucesso"** ou **"Erro na Instalação"**.

III.1 - Projeto

Na realização de estudos e análises de resultados de testes e simulações é bastante comum que se tenha vários gráficos e tabelas de dados, referentes a diferentes estudos acerca de um mesmo circuito.

A fim de viabilizar ao usuário do sistema o agrupamento dos dados referentes a um mesmo projeto ou experimento, criou-se a conceituação de projeto.

Para o STAG, um projeto tem associado a si um conjunto de arquivos de dados e de resultados de análises referentes a um experimento realizado pelo usuário.

Ao definir um projeto, o usuário deverá definir dois tipos de arquivos:

- **Arquivos de Dados:** São arquivos onde se encontram as tabelas de dados a serem visualizadas e analisadas. Na versão atual do sistema são permitidos até 10 arquivos de dados para um projeto, em formato ASCII (gerado pelo próprio SAD) ou no formato de saída do SPICE. Cada arquivo de dados poderá, por sua vez, conter até dez tabelas de dados.
- **Arquivo de Anotações:** Um projeto tem associado a si um único arquivo de anotações, onde o usuário poderá registrar dados de interesse acerca do projeto. O STAG poderá incluir nesse arquivo dados de resultados de operações de análise e tratamento dos dados de forma automática. O usuário poderá também realizar suas anotações manualmente, através de um editor de textos.

Nos arquivos de dados, as informações de interesse ao STAG consistem basicamente de tabelas de dados, as quais relacionam um conjunto de valores para duas ou mais variáveis correlatas. Essas tabelas poderão descrever uma curva, um conjunto de curvas ou ainda famílias de curvas, obtidas através da parametrização de uma ou mais variáveis contidas na tabela. A primeira coluna da tabela será assumida como a variável de varredura (ou variável independente), correspondendo à variável do eixo X no traçado das curvas. As demais colunas serão tratadas, uma a uma, como funções individuais da variável de varredura.

Para a utilização do STAG, a definição do nome do projeto não é obrigatória. Considerando porém, que a definição do projeto tem como objetivo básico uma melhor organização das atividades do usuário, ela é fortemente recomendada.

III.2 - Modos de Operação

Durante a utilização do STAG, o usuário terá dois modos básicos de operação: o modo de **Tratamento dos Gráficos** e o modo de **Análise dos Dados**. O usuário pode migrar de um modo de operação para outro através de opcão no menu principal ou através de teclas especiais (*hot-keys*).

O conjunto de operações disponíveis num determinado instante depende do modo de operação em que o usuário se encontre. O STAG possui um conjunto de operações comuns aos dois modos de operação, um conjunto de operações disponíveis somente no modo de tratamento dos gráficos e outro conjunto exclusivo ao modo de análise dos dados.

O modo de operação *default* do sistema é o modo **Tratamento**. Para migrar desse modo para o modo **Análise** o usuário deve selecionar a opção correspondente no menu principal do STAG ou ainda ativar a *hot-key* correspondente. Abandonando o modo de análise (através da tecla <ESC>) o usuário retorna automaticamente ao modo **tratamento**.

O modo de operação atual é indicado no início da linha de status, mostrada na região inferior da tela.

IV.1 - Parâmetros de Ativação do Sistema

Diversos parâmetros de ativação podem ser definidos pelo usuário na ativação do STAG, sendo todos eles opcionais. A linha de comando para a ativação do sistema tem o seguinte formato:

STAG [?] [F=<NomArq >] [T=<TipoArq >] [P=<Proj >]
onde:

?	:	Mostra os possíveis parâmetros para ativação
F=<NomArq >	:	Define Nome do arquivo de entrada
T=<TipoArq >	:	Define o tipo do arquivo de dados 1= Formato de saída do SPICE 2= Formato ASCII
P=<Proj >	:	Define o nome do projeto

Observações:

1) Caso o parâmetro "F=<NomArq >" seja definido na ativação do sistema, o tipo de arquivo default é em formato ASCII.

2) Caso seja definido o parâmetro "T=<TipoArq >" e não seja definido o nome do arquivo de entrada, o tipo de arquivo é desconsiderado.

IV.2 - O Mecanismo de Auxílio ao Usuário (*Help*)

O mecanismo de Auxílio ao Usuário foi implementado de forma a ser sensível ao contexto. Desta maneira, as telas de auxílio apresentadas ao usuário se referem sempre ao contexto operacional em que o sistema se encontra.

Além disso, ele á implementado na forma de hipertexto, permitindo que além da simples visualização da tela de auxílio, o usuário possa realizar a seleção de palavras-chave apresentadas nessa tela, caminhando através das telas de auxílio e visualizando assim detalhamentos dos tópicos selecionados a cada tela.

A qualquer momento durante o uso do sistema, a tecla <F1 > mostra a tela de auxílio referente ao contexto operacional nesse instante. Teclando <ALT-F1 > o usuário pode visualizar a tela de auxílio que mostra as teclas especiais (*"hot-keys"*) disponíveis no modo de operação atual.

Na maioria das telas de *help* há algumas palavras-chave mostradas em destaque. O usuário pode consultar diretamente as telas de auxílio referentes a qualquer uma dessas palavras-chave, simplesmente movimentando o cursor até a palavra desejada e teclando <ENTER >.

Estando dentro do sistema de *Help*, o usuário pode a qualquer instante usar a tecla <PgUp > para retornar à tela de auxílio anterior ou a tecla <ESC > para abandonar o *Help*.

IV.3 - Descrição das Regiões de Tela

A tela principal do sistema, através da qual o usuário pode manipular os gráficos é organizada em diversas regiões. Na figura A3.1 são mostradas as principais regiões da tela.

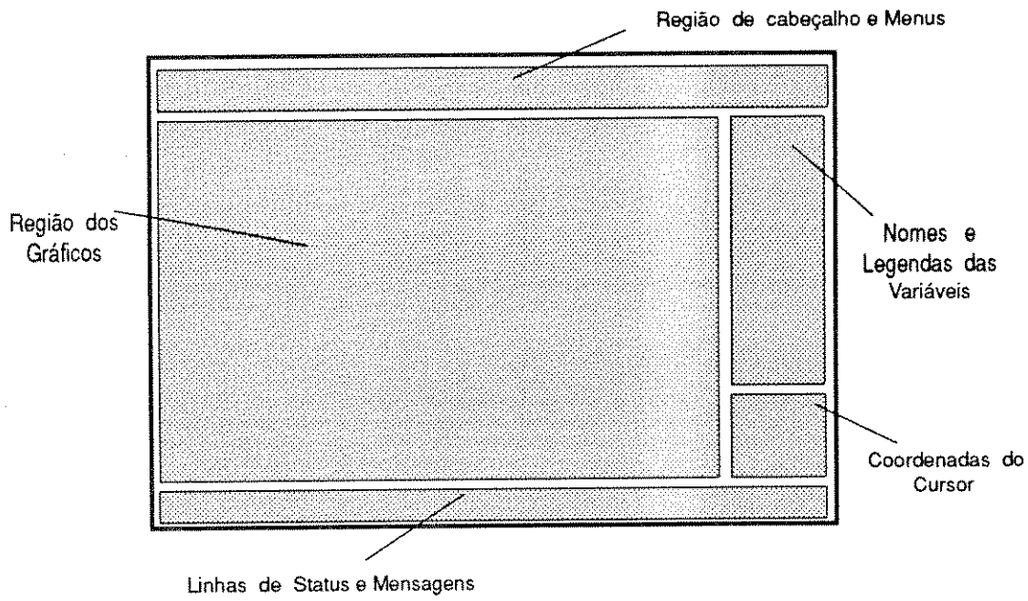


Figura A3.1 - Regiões da Tela para o STAG

Descrevemos a seguir cada uma das regiões da tela principal.

- **Região do Menus:** Nesta região serão mostrados os menus de opções do STAG, através dos quais o usuário poderá escolher quaisquer das operações disponíveis no sistema. Deve-se observar que o menu principal não permanece presente na tela o tempo todo. Para ativar os menus, o usuário deve, estando no modo de tratamento dos dados, teclar <ESC >. Para abandonar os menus, deve-se teclar <ESC > a partir do menu principal do sistema.
- **Região dos Gráficos:** Nesta região serão mostrados os resultados gráficos dos tratamentos e análises aplicados aos dados em estudo. Através de um cursor, o usuário poderá realizar diversas operações de tratamento gráfico dos dados sob estudo. Nesta região serão apresentadas também as informações de auxílio ao usuário (*help*), quando solicitadas.

- **Região das Linhas de Status e Mensagens:** Nesta região o sistema apresenta ao usuário mensagens explicativas e/ou de advertência. É apresentada também uma linha mostrando o estado operacional do sistema, com informações como o modo de operação atual (tratamento ou análises), nível de *zoom*, o nome da projeto, o tipo de escalas, etc...
- **Região de Nomes e Legendas das Variáveis:** Serão mostradas nesta região os nomes das variáveis atualmente em estudo, assim como as suas respectivas legendas. Caso haja mais de uma tabela de dados carregada, as variáveis de diferentes tabelas serão separadas por linhas horizontais. A variável correspondente ao eixo X será sempre a primeira variável da tabela, identificadas pelo texto "[X]" após o nome da variável. Para as variáveis que forem desativadas pelo operador, ao invés da legenda será mostrada a palavra *"OFF"*. As variáveis paramétricas serão identifacadas pela palavra "Param" ao invés da legenda.
- **Região das Coordenadas do Cursor:** Esta região apresentará ao usuário os valores das coordenadas X e Y correspondentes à posição atual do cursor no gráfico.

IV.4 - Limites e Restrições

Descreveremos nesta seção os requisitos de *hardware* e *software* para o STAG, assim como valores-limite estabelecidos para a atual implementação.

IV.4.1 - Requisitos de *Hardware*.

- CPU padrão IBM-PC/XT/AT, com memória mínima de 512 kbytes.
- Monitor de Vídeo padrão CGA, Hércules, EGA, VGA, SVGA , MCGA ou 8514/A.
- Unidade de disco Rígido.
- Teclado.
- Mouse (opcional).
- Impressora (opcional).

IV.4.2 - Requisitos de *Software* Básico.

- Sistema Operacional MS-DOS, versão 3.1 ou superior (ou compatível).
- "*Driver*" de emulador de teclado para *mouse* (caso este esteja instalado).

IV.4.3 - Parâmetros-Limite

- Máximo de pontos por variável numa tabela: 1024
- Máximo de arquivos de dados: 10
- Máximo de tabelas por Arquivo de Dados: 10
- Máximo de Variáveis Simultâneas: 30
- Máximo de niveis de Zoom: 16
- Máximo de Curvas Visualizadas Simultaneamente: 20
- Máximo de Curvas numa família: 20
- Máximo de variáveis paramétricas numa família de curvas: 5

V.1 - Entrando no Sistema

Ao ser carregado, o sistema mostra a sua tela introdutória, que solicita ao usuário o nome do projeto (caso este já não tenha sido definido através da linha de comando). O nome do projeto não é obrigatório. Teclando <ESC > no início do campo, o usuário entra no sistema sem definir nome de projeto.

Após a digitação do nome do projeto, é permitida a modificação da definição dos nomes dos arquivos de dados e de anotações associados ao projeto.

Caso tenha sido definido, na ativação do sistema, um nome de arquivo, o STEF lerá o arquivo e fará o traçado do gráfico.

Em seguida, o sistema apresentará o cursor no centro da tela e passará a aguardar comandos do usuário. Nesse instante o usuário está no modo *Tratamento* e , teclando <ESC > , pode visualizar os menus. Para migrar para o modo *Análises*, o usuário pode utilizar a tecla <F8 > ou selecionar a opção correspondente no menu principal.

A qualquer momento durante o uso do sistema, está disponível o sistema de Auxílio ao Usuário ("*Help*"), que pode ser ativado através das teclas <F1 > e <ALT-F1 >. A tecla <F1 > mostra a tela de *Help* referente ao contexto atual do sistema. A tecla <ALT-F1 > mostra na tela quais as teclas especiais (*Hot-keys*) disponíveis no modo de operação atual (Tratamento ou Análise).

V.2 - Conjunto de Operações do Sistema

Todo o conjunto de operações oferecido pelo STAG encontra-se disponível através dos seus menus de opções e, na sua maioria, também através de teclas especiais (*hot-keys*).

Os menus de opções são a principal forma de interação entre o sistema e o usuário. Usuários mais familiarizados com o sistema poderão achar mais conveniente o acesso às operações através das *hot-keys*, evitando assim a necessidade de "navegar" pelos menus de opções até selecionar a opção desejada.

A figura A3.2 apresenta um exemplo de uma tela de menu de opções do sistema.

Em termos funcionais, as operações foram agrupadas em seis diferentes classes:

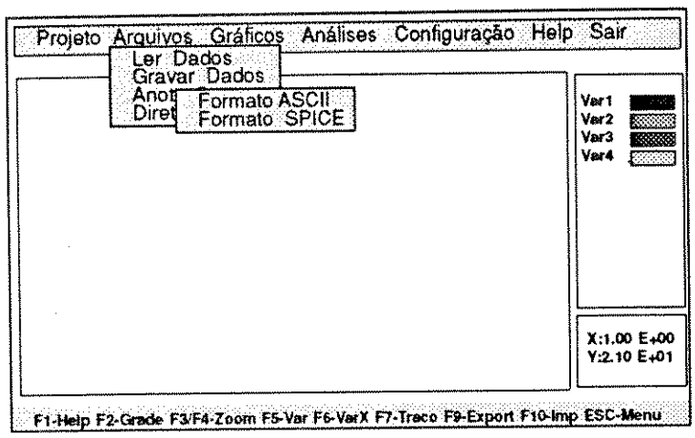


Figura A3.2 - Exemplo de Tela de Menu de Opções do STAG

- Definição do **Projeto**
- Operações sobre **Arquivos**
- Tratamento dos **Gráficos**
- **Análises** dos Dados
- **Configuração** do STAG
- ***Help***

Descreveremos a seguir as operações referentes a cada um desses grupos.

V.2.1 - Definição do Projeto

As operações de definição do projeto têm por objetivo permitir ao usuário do STAG uma melhor organização das suas atividades, agrupando, num mesmo projeto, vários arquivos de dados (em geral, dados referentes a vários estudos de um mesmo circuito ou dispositivo) e um arquivo de anotações.

Ao selecionar-se a opção **Projeto** no menu principal, é apresentado ao usuário um submenu com as seguintes opções:

- Carregar
- Modificar
- Eliminar
- Consultar
- Diretório

Através da opção **Projeto - Carregar**, o STAG permite ao usuário selecionar o projeto a ser utilizado. Ao selecionar-se esta opção, o sistema solicitará o nome do projeto a ser carregado. Caso seja fornecido o nome de um projeto inexistente, será criado um novo arquivo de descrição do projeto. Deve-se ressaltar que o STAG permite um único projeto carregado num determinado instante. Portanto, caso haja um projeto já carregado, ao se carregar um novo projeto, o projeto anterior deixa de estar ativo.

Após digitado o nome do projeto, o usuário tem acesso a uma tela na qual poderá modificar os nomes dos arquivo associados ao projeto.

A opção **Modificar** permite que seja modificada a definição de um projeto. Através desta opção o usuário poderá acrescentar, modificar ou suprimir nomes de arquivos associados ao projeto atualmente em uso.

A opção **Eliminar** permite que o usuário elimine um determinado arquivo de descrição de projeto do disco. Ao selecionar-se esta operação o STAG solicitará ao usuário o nome do projeto a ser eliminado. Após digitado o nome do projeto, será pedida a confirmação da eliminação do projeto e, caso seja confirmada, o arquivo de descrição do projeto é eliminado do disco. Note-se que esta operação elimina apenas o arquivo de definição do projeto. Os arquivos de dados e o arquivo de anotações associados ao projeto permanecem intactos.

A opção **Consultar** permite que o usuário verifique os nome dos arquivos de dados e de anotações associados ao projeto atualmente em uso.

A opção **Diretório** permite ao usuário consultar quais os nomes de projeto atualmente definidos num determinado diretório do disco. Ao selecionar esta opção, o usuário poderá digitar o nome do diretório a ser apresentado e, em seguida, o STAG apresentará os nomes dos projetos definidos naquele diretório.

V.2.2 - Operações sobre Arquivos

O conjunto de operações sobre arquivos reúne as operações voltadas à manipulação dos arquivos de dados de entrada e saída do sistema armazenados em disco. Na versão atual estão disponíveis as seguintes operações:

- Ler Dados
- Gravar
- Anotações
- Diretório

Descreveremos a seguir cada uma dessas operações.

V.2.2.1 - Ler Dados

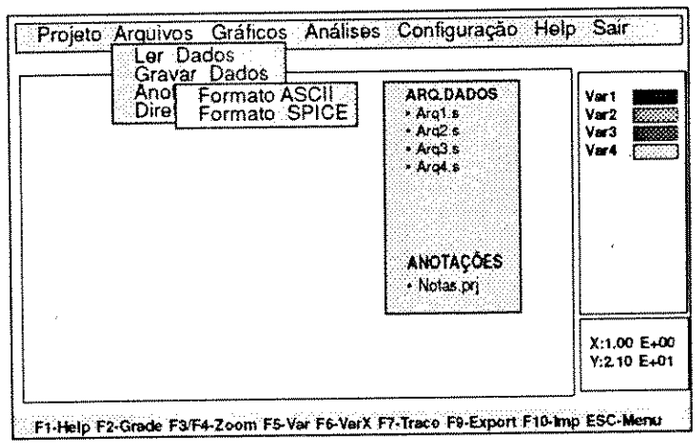


Figura A3.3 - Tela para Seleção do Arquivo para Leitura

Através da operação **Ler Dados** o usuário pode carregar tabelas de dados armazenadas em disco para a visualização, tratamento e análise.

Caso tenha sido definido o nome do projeto, será mostrado um submenu contendo os nomes dos arquivos de dados associados ao projeto, conforme é mostrado na figura A3.3.

O sistema permite a leitura de dados em formato ASCII padrão ou no formato de saida do simulador SPICE. São consideradas válidas tabelas que contenham de 2 até 11 variáveis (colunas).

Os arquivos em formato ASCII devem estar em formato tabular, com cada linha da tabela terminada pelo caracter <CR >.As colunas da tabela deverão ser separadas espaços em branco e/ou vírgula. A primeira linha do arquivo poderá conter os nomes das variáveis tabuladas.

Os arquivos no formato de saída do SPICE devem ser compatíveis com os arquivos de saída gerados pela versão 2G desse simulador.

O STAG suporta um máximo de dez arquivos de dados abertos simultaneamente. Devemos ressaltar porém, que geralmente o sistema esbarrará em outras limitações (por exemplo, exceder o número máximo de variáveis ou mesmo ocorrer falta de memória) antes que seja atingido o número máximo de arquivos abertos simultaneamente.

V.2.2.2 - Gravar Dados

Através da opção de **Gravar** dados, o usuário pode armazenar em um arquivo específico no disco um subconjunto dos dados atualmente "carregados" pelo sistema.

Ao selecionar-se esta operação, o sistema permitirá que seja selecionada uma das tabelas de dados atualmente carregadas e pedirá ao usuário o nome do arquivo a ser gerado. Os dados são armazenados em formato ASCII, permitindo assim o seu uso por outras ferramentas de *software*.

Serão gravados em disco somente os dados referentes à atual visualização em tela. Usando o *Zoom* antes da operação **gravar dados** pode-se selecionar o sub-conjunto de dados a ser gravado em disco.

Esta operação pode ser ativada também pela *hot-key* <F9 >.

V.2.2.3 - Anotações

A operação de **Anotações** permite ao usuário criar e modificar, através de um editor de textos, dados históricos e de observações realizadas durante uma sessão de uso do sistema.

No arquivo de anotações podem ser armazenadas também, a pedido do usuário, informações sobre resultados de análises realizadas.

Ao selecionar-se esta operação o STAG ativa um editor de textos, permitindo que o usuário modifique o arquivo de anotações.

V.2.2.4 - Diretório

A opção **Diretório** permite que o usuário consulte diretórios de arquivos em disco. Ao selecionar esta opção, o sistema solicitará a máscara para busca do diretório. Após isso, serão mostrados na tela os nome dos arquivos que "casam" com a máscara definida pelo usuário.

V.2.3 - Tratamento dos Gráficos

A opção **Gráficos** agrupa os comandos relativos ao tratamento de visualização gráfica dos dados. Estão disponíveis os seguintes grupos de operações:

- Curvas
- Régua
- Grade
- *Zoom*
- Pan
- Cursor
- Traçado
- Escalas
- Imprimir

V.2.3.1 - Curvas

Este grupo de funções permite que o usuário defina características diretamente associadas às curvas, como a variável a ser usada no eixo X de um gráfico, as variáveis cujas curvas devem ou não ser traçadas e variáveis assumidas como paramétricas. Ao selecionar-se a opção **Curvas**, é apresentado ao usuário um submenu com as seguintes opções:

- Variáveis do Gráfico
- Variável para Eixo X
- Variáveis Paramétricas

Discutiremos a seguir cada uma dessas operações.

a) Variáveis do Gráfico

A função **Variáveis do Gráfico** permite que o usuário selecione as variáveis desejadas para a visualização no gráfico. Esta função é útil quando há muitas variáveis no gráfico e o usuário deseja analisar somente algumas delas.

Ao selecionar-se esta operação, a região com os nomes das variáveis do gráfico (no canto direito da tela) funcionará como um menu de opções. As variáveis devem ser selecionadas através das setas. Teclando <ENTER > o usuário ativa/desativa a variável selecionada. Teclando <ESC > o usuário abandona o menu de seleção de variáveis.

Para as variáveis ativas é mostrado, à direita do nome da variável, um ícone ou a cor correspondente a ela no gráfico. Para as variáveis inativas, é mostrada a palavra ***OFF*** à direita do nome da variável.

Caso o usuário desative a variável do eixo X de uma determinada tabela, nenhuma curva referente à tabela é apresentada na tela.

b) Variável para o Eixo X

A função **Variável para o Eixo X** permite que o usuário troque a variável do eixo X com qualquer outra variável ativa no eixo Y.

Caso haja várias tabelas carregadas simultaneamente, o usuário deverá selecionar inicialmente a tabela de dados para a qual ele quer trocar a variável do eixo X.

Após selecionada a tabela de dados, serão mostradas na região de variáveis somente as variáveis da tabela selecionada. O usuário deve selecionar a nova variável usando as setas no menu de variáveis. Teclando <ENTER > a variável selecionada é trocada com a do eixo X.

Observe-se que, caso seja escolhida como nova variável para o eixo X uma variável cuja ordenação não seja monótona crescente ou decrescente, o gráfico visualizado pode ficar "embaralhado". Neste caso, o usuário deve selecionar como tipo de traçado o **traçado por pontos**.

Também deve ser observado que a variável do eixo X só pode ser trocada por outra variável pertencente à mesma tabela de dados.

c) Definição de Variáveis Paramétricas

Para a representação em gráficos bidimensionais de curvas de funções de mais de uma variável, um método frequentemente utilizado consiste da parametrização de (n-1) variáveis da função, definindo assim o que chamamos de famílias de curvas.

Através da opção **parâmetros**, o usuário pode definir variáveis das tabelas de dados como parâmetros, viabilizando assim a análise de famílias de curvas.

Ao selecionar-se esta opção, o STAG deverá marcar a variável a ser usada pelo sistema como paramétrica.

Para excluir uma variável da lista de parâmetros, o usuário deve selecioná-la a partir da lista de nomes de variáveis mostrada no canto direito da tela.

V.2.3.2 - Régua

O comando **Régua** permite que o usuário meça a distância entre dois pontos quaisquer da tela, além da inclinação da reta que os une.

A tecla <INS > é utilizada para ativar a régua, marcando o primeiro ponto. Movimentando o cursor através das setas (ou do *mouse*), o usuário pode ver, em uma janela no canto inferior esquerdo da tela, as distâncias no eixo X e no eixo Y (Dx e Dy respectivamente) entre o ponto inicial e a posição atual do cursor, bem como a inclinação da reta (Dy/Dx) entre os pontos.

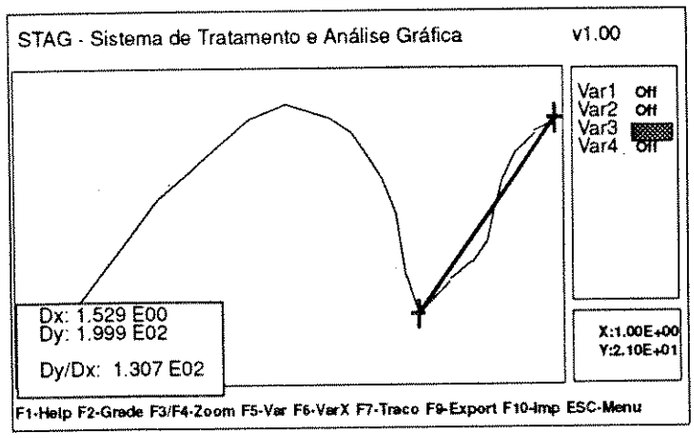


Figura A3.4 - Destaque da Tela com a Função Régua Ativada

Para modificar o ponto inicial marcado para a régua, basta teclar <INS > novamente.

Pressionando-se a tecla <DEL > a régua é desativada.

A figura A3.4 mostra um exemplo da tela do sistema com a régua ativa, destacando-se a janela onde são mostradas as distâncias entre os pontos medidos.

V.2.3.3 - Grade

A grade é um conjunto de linhas pontilhadas paralelas aos eixos do gráfico, usadas para auxiliar visualmente na associação de pontos das curvas aos valores marcados nas escalas dos eixos.

A função **Grade** permite a ativação e desativação da grade. Esta função pode ser selecionada através do menu de opções ou através da tecla <F2 >. Caso a grade esteja ativa, ela é desativada e vice-versa.

V.2.3.4 - *Zoom*

O comando ***Zoom*** permite ao usuário selecionar regiões de interesse específico do gráfico na tela, facilitando o estudo detalhado da região selecionada.

O comando tem duas sub-opções: **Ampliar** e **Reduzir**.

A opção **Ampliar** permite que o usuário marque uma região retangular do gráfico para visualizá-la na tela inteira. A tecla <ENTER > é utilizada para marcar os extremos da região selecionada.

A opção **Reduzir** faz a operação inversa da ampliação, mostrando o gráfico como este se achava antes da última ampliação.

O ***Zoom*** pode ser ativado através dos menus de opções ou através das teclas <F3 > (Ampliar) e <F4 > (Reduzir). No modo de Análises, as operações de ***Zoom*** não são permitidas.

O nível atual de ***Zoom*** é sempre mostrado na linha de status. Na versão atual do sistema são permitidos até 16 níveis de ***Zoom***.

V.2.3.5 - Pan

A opção **Pan**, no submenu das operações de tratamento dos gráficos é uma função complementar das operações de *Zoom*.

Através da função **Pan** o usuário pode definir uma visualização panorâmica dos gráficos, definindo os limites inferiores e superiores para as escalas dos eixos. Pode-se obter também, através desta operação, um efeito semelhante àquele produzido pela operação de *Zoom*.

Ao ser selecionada esta operação, o STAG apresentará uma janela no centro da tela solicitando os limites inferiores e superiores para os eixos do gráfico. Após terem sido digitados esses valores, o gráfico é redesenhado, com os novos limites para os eixos.

V.2.3.6 - Cursor

Ao selecionar-se a opção **Cursor** o sistema apresentará ao usuário duas opções:

- Tipo de Cursor
- Movimentação do Cursor

a) Tipo de Cursor

A função **Tipo de Cursor** permite ao usuário escolher diferentes formatos para o cursor gráfico. Na versão atual são disponiveis três formatos para o cursor:

- Cruz
- Circuferência
- Linhas

O usuário deverá selecionar o tipo de cursor que considerar mais conveniente. O STAG utiliza como *default* o cursor em formato de cruz.

b) Movimentação do Cursor

Através da função **Movimentação do Cursor** o usuário pode escolher o mecanismo utilizado para o deslocamento do cursor. São oferecidas ao usuário duas opções para a movimentação do cursor:

- Normal
- Pontos da Curva

A opção **Normal** permite que o cursor seja deslocado livremente na região dos gráficos.

A opção **Pontos da Curva** restringe o deslocamento do cursor somente a pontos da curva contidos na tabela de pontos do gráfico. As setas para a direita e para a esquerda permitem o deslocamento do cursor sobre uma mesma curva. As setas para cima e para baixo permitem que o cursor se desloque de uma curva para outra. Esta opção é útil quando se deseja conhecer os valores exatos da coordenadas em X e Y para pontos da curva pertencentes à tabela de dados.

V.2.3.7 - Tipo de Traçado

A função **Tipo de Traçado** permite a escolha do tipo de gráfico usado para mostrar os dados em estudo. Estão disponiveis nesta versão, três diferentes tipo de traçado:

- **Por Pontos**: são mostrados somente os pontos definidos pela tabela de dados de entrada (gráfico discreto).
- **Poligonal**: o traçado é realizado interligando-se pontos consecutivos da tabela de entrada através de segmentos de reta.
- **Interpolado**: é calculada uma B-Spline a partir do pontos da tabela de entrada e, a partir dessa função B-Spline, o gráfico é traçado.

O tipo de traçado *default* utilizado pelo sistema é o poligonal. Caso os dados da variável do eixo X não sejam monótonos crescentes ou decrescentes, o sistema assume automaticamente o traçado por pontos.

V.2.3.8 - Tipo de Escalas

A opção **Escalas** permite ao usuário escolher diferentes tipos de escalas para o gráfico. Na implementação atual o sistema permite a seleção de escalas lineares ou logaritmicas para cada um dos eixos independentemente.

Ao selecionar esta opção, o STAG mostrará um submenu com as possíveis combinações de tipos de escala disponíveis. Após selecionado o tipo de escalas desejado, o gráfico é redesenhado.

V.2.3.9 - Imprimir

A função **Imprimir** permite ao usuário listar na impressora uma tabela com os dados correspondentes aos gráficos atualmente visualizados na tela ou ainda que uma imagem do próprio gráfico seja impresso ("*hard-copy*" da tela).

Para imprimir tabelas de dados, o usuário deverá selecionar também qual a tabela cujos dados devem ser impressos.

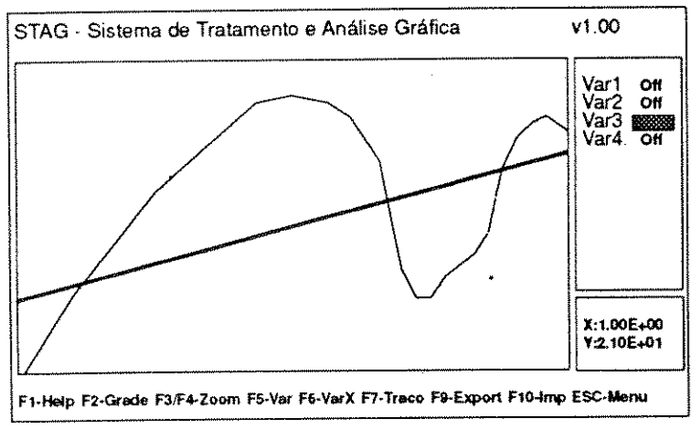


Figura A3.5 - Resultados Gráficos da Operação de Regressão

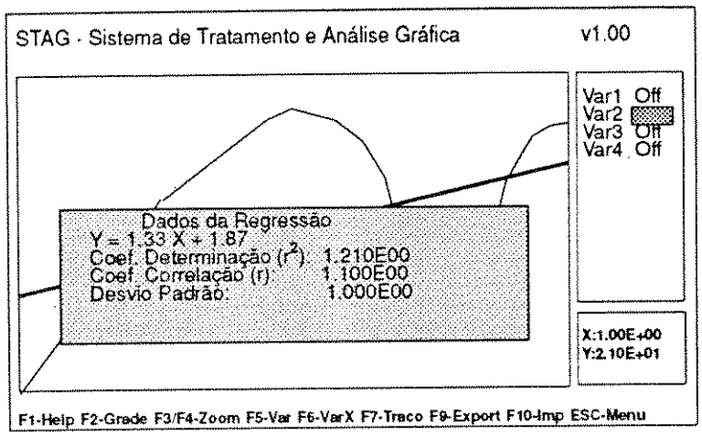


Figura A3.6 - Resultados Numéricos de Operação de Regressão

V.2.4 - Análises dos Dados

As operações de **Análise** permitem ao usuário a aplicação de métodos matemáticos e estatísticos sobre o conjunto de dados de entrada. Esses métodos poderão auxiliar ao usuário, por exemplo, na extração de parâmetros de dispositivos e na determinação de dados para a otimização.

Na versão atual do STAG, estas operações foram divididas em três sub-grupos:

- Regressões
- FFTs
- Estatisticas

Os resultados de qualquer uma das operações de análise podem ser visualizados graficamente na tela ou em formato numérico. Caso seja conveniente, os resultados poderão também ser armazenados em disco (no arquivo de anotações).

V.2.4.1 - Regressões

Ao selecionar a opção de **Regressões**, são oferecidas pelo sistema quatro sub-opções:

- Regressão Linear
- Regressão Geométrica
- Regressão Exponencial
- Regressão Polinomial

Para qualquer uma das opções de regressão, os resultados numéricos podem ser visualizados teclando-se <F3 >. A tecla <F4 > permite que os resultados numéricos sejam armazenados em disco. Para abandonar o modo de **Análise**, o usuário deve teclar <ESC >, retornando assim ao modo **Tratamento**.

Na figura A3.5 é mostrado um exemplo da tela com resultados gráficos de uma regressão. A figura A3.6 apresenta um exemplo da visualização dos resultados numéricos dessa regressão. Telas similares à da figura A3.6 serão também apresentadas para as análises estatísticas.

Descreveremos a seguir cada um dos tipos de regressão.

Regressão Linear

A **Regressão Linear** define uma reta que pode ser descrita pela equação:

$$y = A.x + B$$

O método aplicado para o cálculo dos coeficientes da reta é o dos mínimos quadrados.

Ao selecionar esta opção, o usuário deverá escolher, através do menu no canto direito da tela, a variável à qual será aplicada a regressão. Após isso, o sistema calcula a regressão e traça na região do gráfico a reta correspondente.

Regressão Geométrica

A **Regressão Geométrica** ajusta as coordenadas da variável selecionada pelo usuário através de uma equação do tipo:

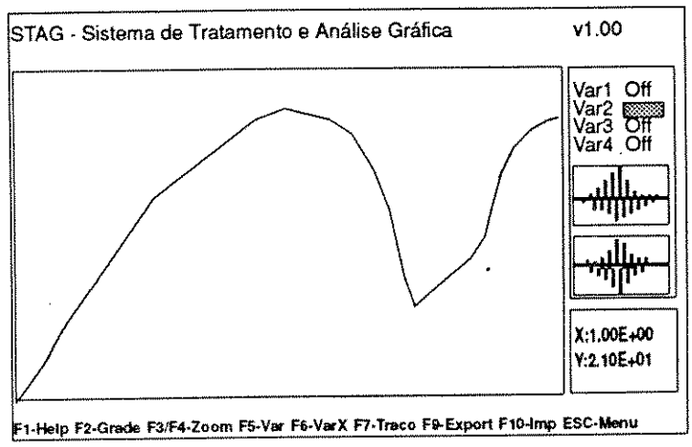


Figura A3.7 - Tela de Resultados do Cálculo de FFT

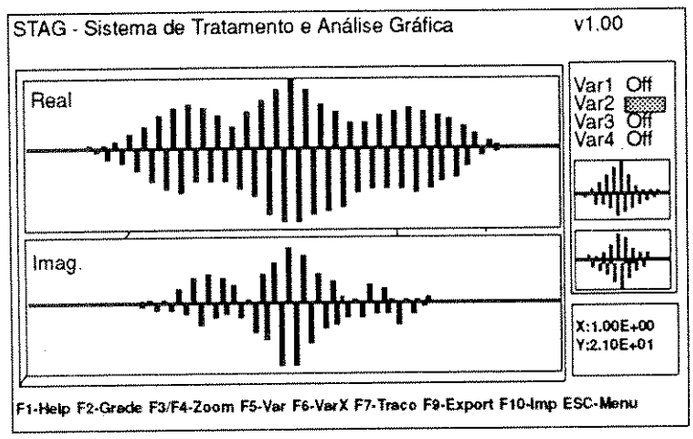


Figura A3.8 - Tela com Resultados da FFT Ampliados

$$y = A.x^B$$

O método utilizado para este cálculo é o dos mínimos quadrados.

A seleção da variável à qual deve ser aplicada a regressão é feita de maneira similar à da regressão linear.

Após calculada a regressão, a curva resultante é mostrada na tela.

Regressão Exponencial

A **Regressão Exponencial** calcula a curva exponencial que mais se aproxima do conjunto de pontos em estudo. Essa curva pode ser definida por uma equação do tipo:

$$Y = A \cdot e^{(B.X)}$$

onde ***e*** é a base neperiana.

Ao selecionar esta opção o usuário deverá escolher, no canto direito da tela, a variável à qual será aplicada a regressão. A regressão é calculada e é traçada na tela a curva por ela definida.

Regressão Polinomial

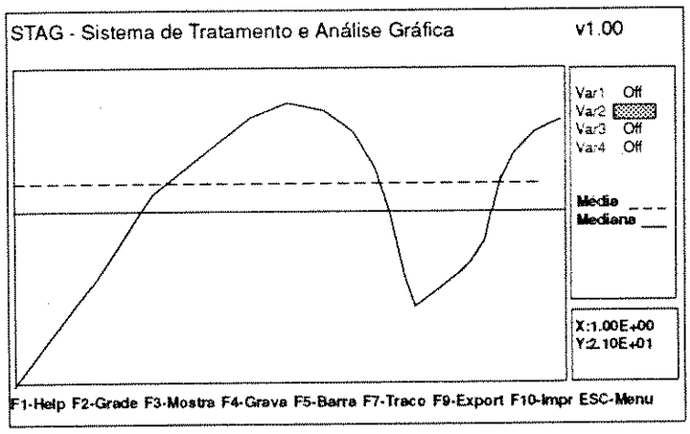


Figura A3.9 - Resultados Gráficos das Estatísticas

A **Regressão Polinomial** faz a aproximação dos pontos em estudo para uma curva definida por um polinômio de grau N.

Ao selecionar esta opção, será pedido o grau do polinômio interpolador a ser calculado e em seguida o usuário deverá escolher a variável à qual será aplicada a regressão. Na versão atual o máximo grau para o polinômio é 10 (dez).

A exemplo dos outros tipos de regressões, a curva correspondente ao polinômio interpolador calculado é mostrada na tela.

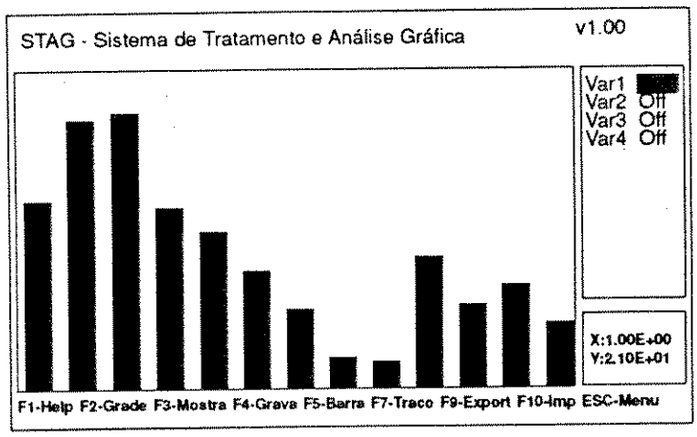


Figura A3.10 - Gráfico de Distribuição de Frequências

V.2.4.2 - FFTs

As **Transformadas Rápidas de Fourier (FFTs)** são um tratamento matemático largamente utilizado em processamento de sinais digitais.

Ao ativar esta opção, serão apresentadas ao usuário duas sub-opções:

- FFT Direta
- FFT Inversa

Após selecionar uma dessas opções, o usuário deverá escolher a variável à qual será aplicada a FFT.

Após calculada a FFT, o resultado é mostrado graficamente em duas pequenas janelas no canto direito da tela, podendo ser selecionada uma visão em tela inteira através da tecla <F3 >. A tecla <F4 > permite ao usuário gravar os coeficientes da FFT no arquivo de anotações. Para abandonar o modo de **Análise**, o usuário deve teclar <ESC >. As figuras A3.7 e A3.8 mostram respectivamente as telas para visualização da FFT no canto da tela (janela reduzida) e na tela inteira (janela ampliada).

V.2.4.3 - Estatísticas

As operações de **Estatísticas** agrupam um conjunto de funções que permitem ao usuário realizar análises de distribuição de frequência, bem como verificar medidas de tendência central e de dispersão das amostras em estudo.

O conjunto de operações estatísticas disponíveis na versão atual engloba apenas estatísticas básicas. O uso do sistema determinará se deve ou não ser ampliado o conjunto de operações estatísticas.

As operações atualmente disponíveis são listadas a seguir:

- Média
- Moda
- Mediana
- Desvio Médio
- Desvio Padrão
- Variância
- Coeficiente de Variação
- Distribuição de Frequências

Após selecionar a opção de estatísticas, o usuário deve selecionar a variável à qual as estatísticas devem ser aplicadas. Em seguida, são realizados os cálculos e são mostrados graficamente os valores da média e da mediana, conforme mostrado na figura A3.9.

O usuário pode ainda obter gráficos de distribuição de frequências da variável selecionada, usando a tecla <F5 >. Teclando-se <F5 >, o sistema solicitará os limites superior e inferior para a variável selecionada, assim como o número de intervalos (nº de barras) do gráfico. Após terem sido fornecidos esses dados, a distribuição é mostrada em um gráfico de barras, conforme a figura A3.10. Para abandonar os gráfico de barras (retornar ao gráfico anterior), o usuário deve teclar <F5> novamente ou <ESC>.

V.2.5 - Configuração

Através da opção de **Configuração** o usuário pode redefinir algumas características ambientais do sistema. Ao ser selecionada esta opção, o STAG apresentará ao usuário um submenu com as seguintes opções:

- Cores
- Idioma
- Impressora
- Plotter
- Editor de Textos

A **Configuração de Cores** permite que o usuário redefina as cores utilizadas em tela para todo o sistema. Ao selecionar-se esta opção será, mostrado um sub-menu com as opções de cores modificáveis. O usuário deverá escolher a sua opção e, em seguida, a cor desejada. As modificações de cores realizadas são válidas durante a sessão de uso do sistema. Ao final da sessão, o sistema perguntará ao usuário se ele deseja ou não gravar em disco a nova configuração de cores.

Ao selecionar-se a opção de **Configuração de Idioma** o STAG apresentará na tela um submenu contendo as opções de idioma disponíveis. Nesta versão do sistema o Português é o único idioma disponível. O objetivo da configuração de idioma é permitir ao usuário selecionar o idioma a ser utilizado para os menus de opções e mensagens de maneira geral. Estão previstos para versões futuras os seguintes idiomas: Português, Inglês, Espanhol e Francês.

A opção de **configuração de impressora** deverá permitir que seja definido o tipo e modelo da impressora instalada, permitindo assim que o STAG faça melhor uso de características particulares de cada tipo de impressora. Na versão atual esta opção na se encontra implementada. Para a impressão de textos o sistema assume que não é necessária nenhuma tabela de tradução de caracteres. Para a impressão de gráficos o sistema assume a impressora definida pelo comando **GRAPHICS** do sistema operacional (definido no arquivo STAG.BAT). Para maiores informações sobre a configuração do tipo de impressora para o comando GRAPHICS, consulte o manual do usuário do MS-DOS.

A opção de **configuração de *plotter***, a exemplo da configuração de impressora e da configuração de idiomas, não se encontra disponível na versão atual do STAG.

Através da operação de **configuração do editor de textos** o usuário poderá definir qual o editor de textos a ser ativado para anotações. Ao selecionar-se esta opção, o STAG apresentará na tela uma janela solicitando a linha de comando para a ativação do editor de textos (*pathlist* completo e nome do arquivo a executar). O STAG assume como *default* o editor de textos incorporado ao SDP. As informações armazenadas automaticamente pelo STAG no arquivo de anotações estão em formato ASCII padrão e, portanto, é aconselhável que o usuário escolha um editor de textos que use esse formato como *default*. Devemos ressaltar também que, dependendo da quantidade de memória requerida para a utilização do editor de textos, pode ocorrer do sistema não ser capaz de ativar o editor. Nesse caso, recomendamos a utilização do editor *default* do sistema.

V.2.6 - *Help*

Como já foi descrito na seção IV.2, o mecanismo de auxílio ao usuário foi implementado na forma de hipertexto, podendo ser ativado através do menu principal ou da tecla <F1 >.

Se ativado através da tecla <F1 >, é mostrada ao usuário a tela de auxílio referente ao contexto atual. Ativando o *help* através do menu principal, a tela de auxílio apresentará os principais tópicos do sistema, devendo o usuário selecionar o tópico de seu interesse.

Teclando-se <ALT-F1 > o sistema mostra na tela o conjunto de *hot-keys* disponíveis naquele instante.

VI - Resumo dos Comandos das Teclas Especiais (Hot-Keys)

Como foi descrito anteriormente, a maior parte das operações oferecidas pelo sistema podem ser ativadas diretamente através de *hot-keys*. Algumas dessas *hot-keys* estão associadas a uma única função, independentemente do contexto operacional do sistema, enquanto outras têm funções distintas de acordo com o contexto atual. Os conjuntos de *hot-keys* disponíveis estão associados diretamente ao **modo de operação** que o sistema se encontra.

Descreveremos a seguir, de forma resumida, as *hot-keys* disponíveis em cada um dos modos de operação: **tratamento** ou **análise**.

Hot-Keys disponíveis no Modo Tratamento

F1 - **"Help"** - Mostra tela de auxílio referente ao contexto atual.
F2 - Liga ou desliga **Grade**.
F3 - Ativar **Zoom**.
F4 - Desativar **Zoom**.
F5 - Seleção das **variáveis para o gráfico**.
F6 - Seleção da **variável para o eixo X**.
F7 - Seleção do **tipo de traçado** para o gráfico.
F8 - Ativa modo **análise** (estatísticas, FFTs, regressões).
F9 - **Exportar dados** (grava os dados da visualização atual em formato ASCII).
F10 - **Imprimir gráfico** (*hard-copy* da tela).
ALT-F1 - Mostra ***Help das hot-keys*** disponíveis.
ALT-F2 - Liga ou desliga **linha de status**.
ALT-F3 - Modifica o **tipo de cursor**.
ALT-F4 - Modifica **tipo de escala** usado para o gráfico.
ALT-F5 - Permite a definição variáveis a serem tratadas como parâmetros.
ALT-F10 - Permite a **configuração de cores**.
INS - Liga a **régua**.
DEL - Desliga a **régua**.
+ - Incrementa o tamanho do passo do cursor.
- - Decrementa o tamanho do passo do cursor.
ESC - Dá acesso aos menus de opções. Teclando <ESC > no menu principal, o usuário volta ao modo Tratamento.

Hot-Keys **disponíveis no Modo Análise**

F1 - **"Help"** - Mostra tela de auxílio referente ao contexto atual.
F2 - Liga ou desliga **Grade**.
F3 - Mostra na tela **resultados numéricos da análise**.
F4 - **Grava resultados da análise** em disco.
F5 - Ativa ou desativa **Gráfico de Barras** (somente para estatísticas). Permite visualização de gráficos de distribuição de frequências.
F6 - Seleção da **variável para o eixo X**.
F7 - Seleção do **tipo de traçado** para o gráfico.
F8 - Sem uso.
F9 - **Exportar dados** (grava os dados da visualização atual em formato ASCII).
F10 - **Imprimir gráfico** (*hard-copy* da tela).
ALT-F1 - Mostra ***Help das hot-keys*** disponíveis.
ALT-F2 - Liga ou desliga **linha de status**.
ALT-F3 - Modifica o **tipo de cursor**.
ALT-F4 - Modifica **tipo de escala** usado para o gráfico.
ALT-F10 - Permite a **configuração de cores**.
INS - Liga a **régua**.
DEL - Desliga a **régua**.
+ - Incrementa o tamanho do passo do cursor.
- - Decrementa o tamanho do passo do cursor.
ESC - Retorna ao modo **Tratamento**. Teclando <ESC> novamente, dá acesso aos menus de opções.

STEF- Sistema para Testes de Funcionais de Circuitos Digitais

Manual do Usuário

Versão 0.50 - Janeiro/1993

Edição Preliminar

Revisão: A

Maurício de Vasconcelos Affonso

Índice

I - Introdução

O Sistema de Testes Funcionais de Circuitos Digitais - STEF tem como objetivo permitir ao usuário a programação, execução e análise de resultados de testes funcionais de circuitos digitais.

A programação do experimento poderá ser realizada através da descrição de um programa para o teste, contendo os vetores de teste, condições de parada e as saídas a serem monitoradas. Poderão também ser utilizados como entrada, dados obtidos de outras ferramentas do SAD ou do SDP, como exemplo o EDTES, o PESQA e o SIMUL. Uma terceira forma de programação do experimento é a programação de maneira interativa, através de operações incorporadas ao próprio STEF.

A partir do programa do experimento, o usuário poderá efetuar os testes do circuito, armazenando os resultados em memória ou em disco. Durante a execução do experimento o usuário poderá visualizar, em tempo real, os resultados dos testes, havendo a possibilidade de interrromper um teste e posteriormente retomá-lo.

O STEF poderá trazer ao usuário ganhos significativos do ponto de vista da análise do comportamento do circuito, visto que, através desta ferramenta, pode-se verificar a influência que modificações no circuito ou nos vetores de teste exercem sobre as saídas do circuito.

O STEF permitirá também a realização de análises comparativas entre resultados de diferentes testes ou com resultados obtidos através da simulação lógica do circuito.

A interface básica com o usuário faz-se através de menus de opções e de telas gráficas, sendo utilizadas mensagens explicativas e telas de auxílio ao usuário sempre que conveniente.

A qualquer instante durante a utilização do sistema está disponível ao operador um sistema de auxílio ao usuário (*help*) sensível ao contexto e em hipertexto, permitindo que sejam obtidas informações de auxílio referentes ao contexto atual e/ou a outros tópicos a ele associados.

Nas seções seguintes descreveremos, inicialmente, o procedimento para a instalação do STEF, passando em seguida a uma descrição geral do sistema e por fim à operação do sistema.

II.1 - Ambiente de Execução

O STEF é executável em microcomputadores da linha PC-xt ou compatíveis, com os seguintes requisitos mínimos:

- 512 kbytes de memória principal
- Controlador de vídeo com resolução mínima de 640x200 pontos
- Disco rígido de 20 Mbytes.
- impressora (opcional)
- *mouse* (opcional)

Em termos de *software* o STEF requer sistema operacional compatível com o MS-DOS 3.1 ou posterior.

Embora o STEF tenha sido implementado para atuar como uma das ferramentas do SAD, ele pode ser também utilizado como um subsistema independente. Neste último caso, o único *software* requerido é o próprio sistema operacional.

II.2 - Definindo as Características do Ambiente

Para que o STEF faça uso pleno de seus recursos, o seu sistema deverá ter definidas algumas características ambientais essenciais. Essas características são descritas no arquivo CONFIG.SYS e são ativadas durante a carga do sistema operacional (*"boot"* do sistema).

Antes de utilizar o STEF, você deve ser certificar de que o arquivo CONFIG.SYS contém as seguintes definições:

```
FILES = 20
BUFFERS = 24
```

Os valores descritos acima são os valores mínimos para as variáveis do ambiente. Valores maiores não deverão causar qualquer tipo de problema.

Se você possui um *mouse* e deseja utilizá-lo, é necessária a instalação de um "*driver*" para emulação de teclado. O botão da esquerda deve ser configurado para a tecla <CR> e o da direita para a tecla <ESC>.

II.3 - Instalando o Sistema

A instalação do STEF compreende duas etapas básicas: a instalação do sistema de *software* e a instalação da interface de testes funcionais. O procedimento para a instalação da interface de aquisição de dados será discutido no manual de instalação da própria interface. Discutiremos nesta seção somente o procedimento para a instalação do sistema de *software.*

Para instalar o sistema de *software* no disco rígido você deverá seguir os seguintes passos:

1) Coloque o disquete contendo o STAG no drive "A:"

2) Digite: **A:INSTSTEF** <ENTER>

3) A partir deste ponto, o programa de instalação lhe fará uma série de perguntas sobre parâmetros para a instalação, como a unidade de disco e o diretório onde o STEF deve ser instalado. Ao término da execução do instalador será dada a mensagem "**STEF instalado com sucesso**" ou "**Erro na Instalação do STEF**".

A operação do STEF é guiada por menus de opções, estando disponível, a qualquer instante, um sistema de auxílio ao usuário sensível ao contexto operacional e na forma de hipertexto.

Uma sessão típica de uso do STEF consiste das seguintes etapas:

- Definição e Programação do Experimento
- Execução do Experimento
- Visualização e Análises dos Resultados

Ao entrar no sistema o usuário deverá definir um **nome de projeto** e, em seguida definir o programa do experimento, através de um editor de textos ou de forma interativa.

O programa do experimento conterá a definição de nomes lógicos para os pinos do soquete de testes, a descrição dos estímulos a serem aplicados ao circuito e dos sinais a serem monitorados. Poderão ser definidas ainda, condições de parada para os testes. Poderão ser definidos, também, estímulos e condições de parada para inicialização do circuito.

Após a definição do experimento, o usuário deverá executar os testes, podendo armazenar os resultados em memória ou num arquivo em disco. Os resultados dos testes poderão ser monitorados em tela durante a execução do experimento e/ou após o término deste. Será também possível ao usuário interromper a execução do experimento, via teclado.

Ao final da execução dos testes, o usuário poderá visualizar e manipular os dados de resultados, comparando-os com resultados de outros testes ou de simulações lógicas do mesmo circuito. Em versões futuras do sistema deverão ser incorporadas funções que permitam a realização de análises mais avançadas, auxiliando o usuário na definição de vetores de testes que facilitem a detecção de causas de falhas, isolando regiões do circuito suspeitas de defeitos, etc...

Após termos discutido uma sessão típica de uso do STEF, passaremos em seguida à apresentação das operações disponíveis e do modo de utilização das mesmas.

III.1 - Ativando o STEF

Para executar do STEF deve-se, a partir do *prompt* do sistema operacional, digitar a seguinte linha de comando.

STEF [?] | [*Proj*]
onde:

?	:	Mostra os possíveis parâmetros para ativação
Proj	:	Define o nome do projeto

Observações:

1) Caso na ativação do STEF seja definido o nome do projeto, o sistema lerá os dados referentes ao projeto em questão, caso contrário criará um novo projeto chamado "SEMNOME".

2) Caso seja definido o nome de um projeto inexistente, o STEF criará um novo projeto com aquele nome. Caso já exista um programa para o SIMUL com o mesmo nome do projeto, será automaticamente realizada a conversão do programa de simulação lógica para o programa de testes funcionais.

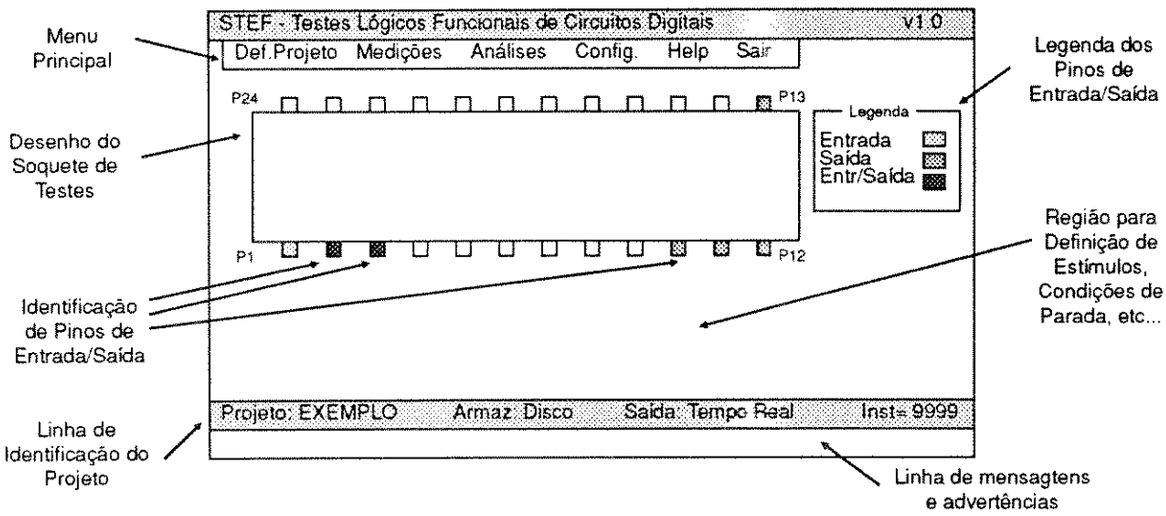


Figura A4.1 - Tela Principal do STEF

Ao ser ativado, o STEF apresentará ao usuário a tela principal de do sistema. A figura A4.1 apresenta essa tela, destacando as principais regiões da mesma. Na parte superior da tela, como podemos observar, é apresentado o menu principal do sistema. Discutiremos a seguir cada uma das opções do menu principal.

III.2 - Operações para Definição do Experimento

As operações para definição das medições permitem que o usuário descreva, de forma interativa ou textual, o programa de testes funcionais. Neste manual concentraremos nossa atenção na descrição interativa do experimento de testes funcionais. A linguagem textual para o programa do experimento será discutida num documento à parte.

Ao selecionar-se a opção de **definição do experimento** no menu principal, o STEF apresentará um submenu contendo as seguintes opções:

- Nome do Projeto
- Descrição do Circuito
- Estímulos para Inicialização
- Estímulos para Testes
- Sinais de Saída
- Editar Programa

Discutiremos a seguir cada uma dessas opções.

III.2.1 - Nome do Projeto

Através da opção **nome do projeto** o usuário pode definir o nome do projeto a ser executado.

Ao selecionar-se esta opção, o STEF solicitará ao usuário o nome do projeto a ser carregado.

Caso seja digitado o nome de um projeto já existente, o programa do experimento é lido, sendo mostrados no desenho do soquete de testes os pinos definidos como entradas e saídas do circuito.

Se for definido um nome de projeto inexistente, o STEF criará um novo projeto.

Durante o processo de carga de um projeto, caso não exista o programa do experimento e exista um programa do SIMUL com o nome definido para o projeto, o STEF fará automaticamente a conversão do programa de simulação para um programa de testes funcionais.

III.2.2 - Descrição do Circuito

A operação de **descrição do circuito** não se encontra disponível na versão atual do sistema.

O objetivo desta função seria permitir ao usuário criar um diagrama esquemático do circuito sob testes, facilitando o processo de definição de estímulos a serem aplicados e saídas a serem monitoradas.

A descrição do circuito seria também utilizada para a realização de funções mais avançadas de análise dos resultados, permitindo o aconselhamento do usuário durante a depuração do circuito sob testes.

III.2.3 - Estímulos para Inicialização

Através da opção **estímulos para inicialização** o usuário pode definir um conjunto de estímulos a serem aplicados para a inicialização do circuito, assim como condições de parada para a inicialização.

Ao ser selecionada esta opção, será ativado um submenu com as seguintes opções:

- Sinais a Aplicar
- Condições de Parada

III.2.3.1 - Sinais a Aplicar

A opção **sinais a aplicar** permite que o usuário defina os estímulos de inicialização do circuito propriamente ditos.

Ao selecionar esta opção o usuário deverá escolher o pino ao qual o sinal será aplicado, através de um cursor que se desloca pelos pinos do desenho do soquete de testes.

Através das setas (ou da movimentação do *mouse*) o usuário pode posicionar o cursor sobre o pino desejado. Teclando <INS> o pino será selecionado para a definição dos estímulos e teclando <DEL> ele eliminará os estímulos de inicialização para aquele pino. Através da tecla <ESC> o usuário pode abandonar esta operação, retornando aos menus de opções.

Ao teclar-se <INS> o sistema solicitará ao usuário o nome lógico do sinal a ser aplicado ao pino e, em seguida apresentará um menu de opções, permitindo que se defina o tipo de sinal a ser aplicado: periódico, aperiódico ou retangular.

A figura A4.2 apresenta exemplos de cada um dos tipos de sinais.

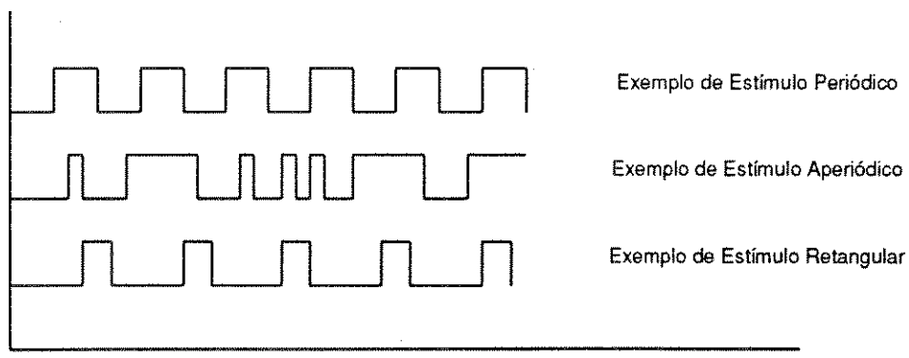


Figura A4.2 - Tipos de Estímulos Externos Disponíveis

Após selecionar o tipo de sinal, na metade inferior da tela será mostrada uma janela na qual o usuário poderá editar graficamente o sinal. Utilizando as setas o usuário deslocará o cursor (que é mostrado como uma barra vertical). Teclando <ENTER> (ou pressionando o botão da esquerda do *mouse*), será incluída (ou excluída, caso já exista) uma transição de sinal para aquele instante.

Para os sinais periódicos ou retangulares, a edição de estímulos apresentará algumas particularidades. O estado inicial do sinal será aquele definido para o instante T=0 (que é o instante inicial para qualquer teste).

Para os sinais periódicos, a primeira transição definida marcará o início da aplicação do sinal. A segunda transição definirá o intervalo entre duas transições consecutivas (metade do período do sinal).

Para sinais retangulares, a primeira transição marcará o instante inicial para a aplicação do sinal, a segunda transição definirá a largura do pulso e a terceira transição definirá o *gap* (tempo de repouso) entre dois pulsos consecutivos.

A figura A4.3 apresenta um exemplo da tela para a operação de edição de estímulos.

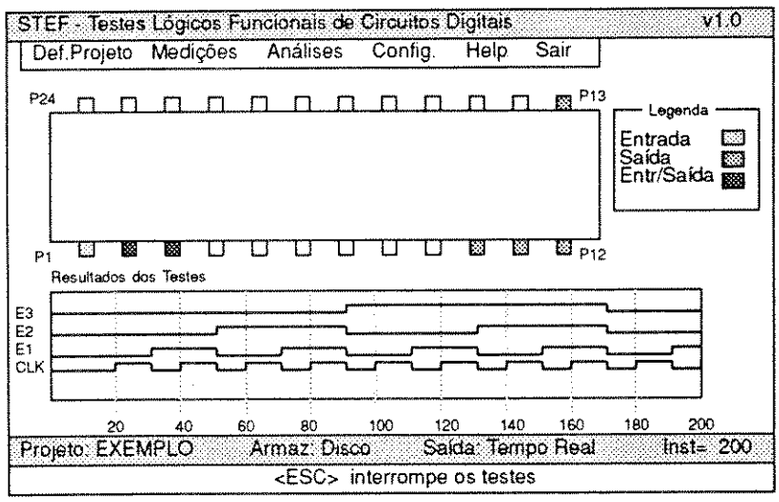


Figura A4.3 - Tela para a Operação de Edição de Estímulos

III.2.3.2 - Condições de Parada

A opção **condições de parada** permitirá ao usuário definir um conjunto de condições lógicas que, quando atingidas, deverão interromper o processo de inicialização do circuito.

Ao selecionar-se esta opção, o STEF apresentará, na região inferior da tela, uma janela, na qual deverá ser digitada a condição de parada para a inicialização do circuito. Essa expressão deve ser definida em termos de valores lógicos nos pinos do soquete de testes e/ou em função do tempo.

Apresentamos na figura A4.4 a descrição da sintaxe para a definição da expressão da condição de parada.

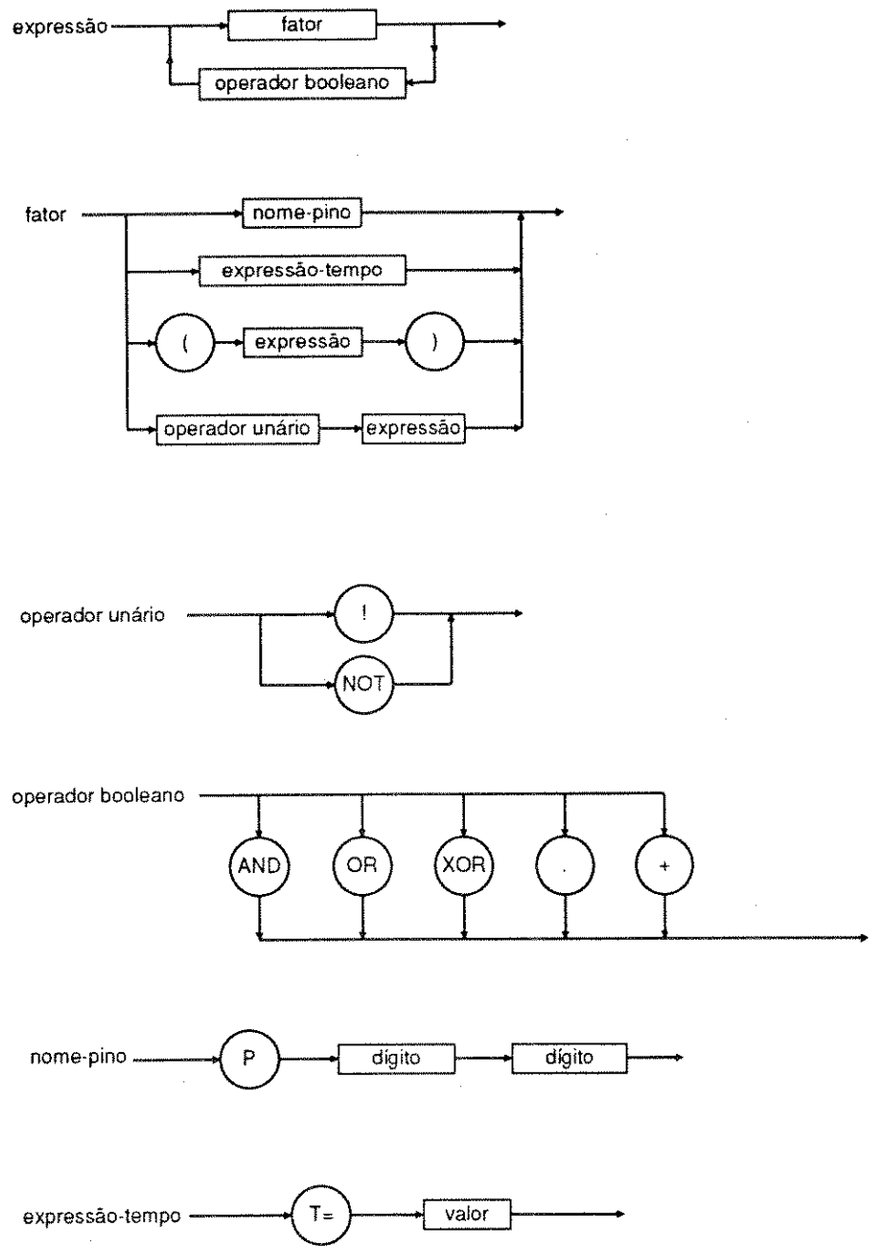


Figura A4.4 - Descrição Sintática das Condições de Parada

Podemos observar, através da descrição sintática da figura A4.4 que os operandos para a expressão poderão ser constituídos por nomes dos pinos ou pela constante "T", que identifica um instante de tempo. Os operadores permitidos são os operadores lógicos booleanos AND, OR, NOT e XOR. Para a subexpressão que define um instante para o término do teste é utilizado o operador de igualdade ("=").

Os nomes dos pinos são formados pelo caracter "P" seguido de um ou dois dígitos que identificam o número do pino. A identificação dos pinos pode ser feita também através de nomes lógicos associados aos pinos do soquete de testes (definidos durante a operação de descrição de estímulos e sinais de saída).

A figura A4.5 apresenta um exemplo da tela apresentada pelo STEF para a operacão de definição das condições de parada.

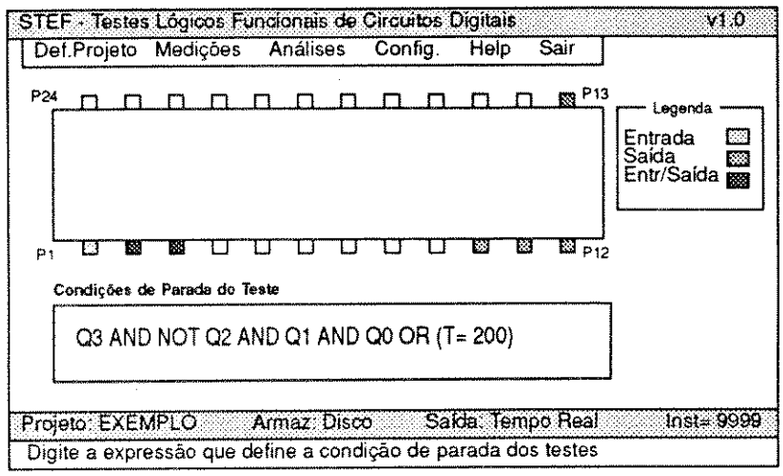


Figura A4.5 - Tela para a Definição de Condições de Parada

III.2.4 - Estímulos para Testes

A opção **estímulos para testes** permite ao usuário definir os vetores de teste a serem aplicados ao circuito.

A exemplo do que ocorre para a definição dos estímulos de inicialização do circuito, ao selecionar-se esta opção o STEF apresentará um submenu contendo as opções de **sinais a aplicar** e de **condições de parada**.

Para ambas as opções, o modo de operação será idêntico àquele descrito para a definição dos estímulos de inicialização do circuito. A única diferença entre os estímulos de inicialização e os estímulos de testes é que o primeiro será aplicado ao circuito visando apenas à obtenção de um conjunto de condições iniciais. Os estímulos para os testes propriamente ditos serão aplicados visando à verificação do comportamento da lógica funcional do circuito, sendo as respostas a esses estímulos devidamente registradas e armazenadas.

III.2.5 - Sinais de Saída

Através da opção **sinais de saída** o STEF permite que sejam definidos, de forma interativa, os sinais a serem monitorados durante a execução dos testes. O usuário poderá selecionar, através de um cursor, os pinos do circuito que deverão ser monitorados.

Através das setas, o usuário movimenta o cursor pelos pinos do desenho do soquete de testes. Teclando-se <INS> são selecionados os pinos a serem monitorados e através da tecla <DEL> é desativada a seleção de um pino anteriormente marcado para monitoração. Teclando <ESC> o usuário termina a operação de seleção dos pinos a serem monitorados, retornando aos menus de opções.

Ao selecionar um pino para saída, o usuário poderá também definir um nome lógico para esse pino.

III.2.6 - Editar Programa

Através da opção **editar programa**, o usuário poderá criar ou modificar um programa de testes funcionais na forma textual.

Ao ser selecionada esta opção, o sistema ativará o editor de textos atualmente configurado para o sistema (para detalhes sobre a configuração do editor de textos, veja a seção III.5), carregando automaticamente o arquivo contendo a descrição textual do programa de testes funcionais. O usuário poderá então editar esse arquivo e, ao abandonar o editor, retornará aos menus de opções do sistema.

III.3 - Operações para Execução das Medições

Através da opção **medições** do menu principal, o usuário poderá realizar a execução do experimento dos testes funcionais, podendo também definir algumas características adicionais relacionadas à execução.

Ao selecionar esta opção, o usuário obterá na tela um submenu com as seguintes opções:

- Armazenar Dados
- Visualizar Resultados
- Executar o Experimento

A opção **armazenar dados** permite ao usuário definir se os dados de resultados dos testes deverão ser armazenados em memória ou diretamente em disco. Caso seja selecionado o armazenamento em memória, será possível, ao final dos testes, armazenar os resultados em disco. Caso seja selecionado o armazenamento em disco, os dados serão gravados durante a execução do experimento. Deve-se ressaltar que, caso seja escolhida a opção de armazenamento dos dados diretamente em disco, a execução do experimento poderá tornar-se mais lenta. O STEF utiliza como *default* o armazenamento dos dados em memória.

A opção **visualizar resultados** permite a definição da estratégia para apresentação dos resultados dos testes. Os resultados poderão ser mostrados durante a execução do experimento (em tempo real) ou somente ao final do mesmo. A apresentação dos resultados em tempo de execução, embora possa tornar a execução mais lenta, permite um melhor acompanhamento da execução dos testes e será, portanto, o *default* para o sistema. A figura A4.6 mostra um exemplo da tela do STEF durante a execução de um experimento com a opção de visualização dos dados em tempo real.

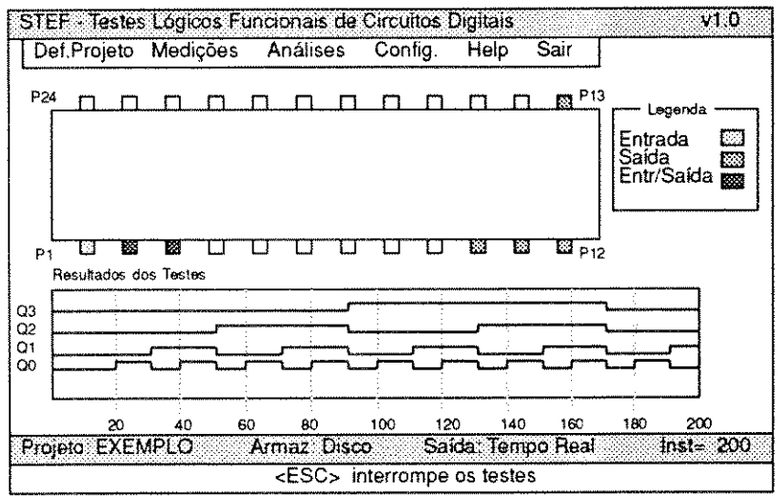


Figura A4.6 - Monitoração de Resultados em Tempo Real

A opção **executar experimento** é responsável pela execução dos testes funcionais propriamente ditos. Ao selecionar-se esta opção o STEF solicitará algumas informações adicionais sobre os testes, como os instantes inicial e final para a monitoração dos resultados, o incremento para a leitura dos dados e o instante final (máximo) para a execução.

A execução dos testes poderá ser interrompida pelo usuário através do teclado, sendo permitida a modificação de informações acerca do experimento e a posterior retomada dos testes. Os testes poderão ser retomados a partir do início ou a partir do instante em que foi interrompido. Para interromper o experimento o usuário deve pressionar simultaneamente as teclas <ALT> e <ESC>.

III.4 - Operações para Análises dos Resultados

As operações de análises, na versão atual do STEF limitam-se apenas à visualização dos resultados dos testes, com alguns recursos voltados à manipulação dos mesmos e comparação com outros resultados.

Em versões futuras deverão ser incluídas funções que implementem técnicas e métodos de análise através das quais o STEF poderá realizar críticas dos resultados e aconselhamento do usuário, auxiliando na localização dos componentes do circuito suspeitos de mau funcionamento, na determinação de vetores de testes capazes de isolar falhas, etc...

Ao selecionar a opção **análises**, o usuário obterá um submenu com as seguintes opções:

- Análises Simples
- Análises Comparativas

III.4.1 - Análises Simples

Selecionando a opção **análises simples** o usuário obterá um tela semelhante à apresentada para monitoração dos resultados em tempo real (figura A4.6), com a diferença que estará disponível também um cursor (apresentado na forma de uma barra vertical) através do qual o usuário poderá se deslocar através dos gráficos. Através da tecla <F10> o usuário poderá imprimir o gráfico de estados atualmente apresentado em tela. O usuário poderá também selecionar outros sinais a serem apresentados, através da opção de definição dos sinais de saída (pela opção **definição do experimento**, no menu principal).

III.4.2 - Análises Comparativas

A opção de **análises comparativas** encontra-se ainda em fase final de integração ao sistema. Selecionando esta opção, o usuário poderá realizar comparações entre os resultados referentes ao projeto atualmente carregado com resultados de outros testes ou resultados de simulações (realizadas através do SIMUL).

Após selecionar esta opção, o usuário deverá digitar o nome do arquivo que contém os resultados a serem comparados. O STEF lerá então esses dados e os apresentará, de forma semelhante àquela apresentada na figura A4.7. As diferenças detectadas entre os dois conjuntos de resultados serão apresentadas em destaque. Estarão também disponíveis os mesmos recursos de manipulação dos dados descritos para a operação de análises simples.

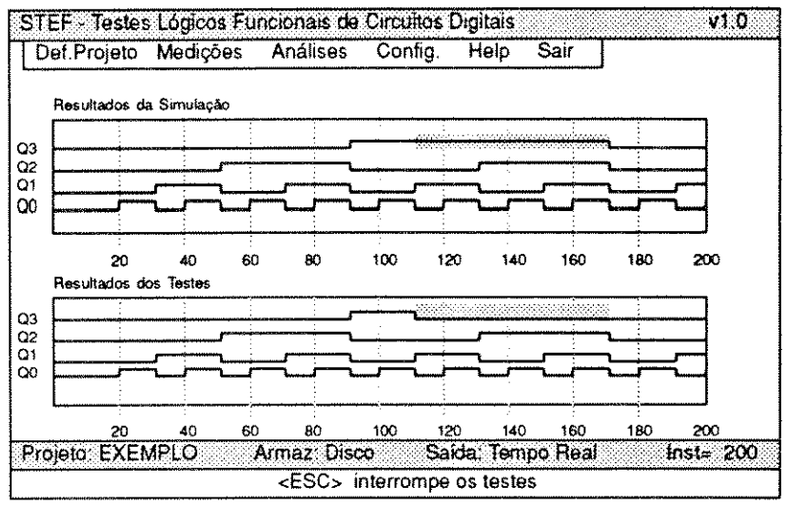


Figura 29 - Análise Comparativa de Resultados no STEF

III.5 - Operações para Configuração do Sistema

Através da opção de **Configuração** o usuário pode definir dados sobre a configuração da interface de testes utilizada pelo STEF e algumas características ambientais do sistema. Ao ser selecionada esta opção, será apresentado ao usuário um submenu com as seguintes opções:

- Interface
- Cores
- Idioma
- Impressora
- Plotter
- Editor de Textos

A opção de **configuração da interface** permitirá ao usuário definir o tipo da interface de testes funcionais utilizada e algumas características da sua configuração. A versão atual do STEF dá suporte a um único tipo de interface de testes [Qua88]. Nessa interface são configuráveis os endereços das portas de entrada e saída (*I/O Ports*). Ao selecionar-se esta opção, o STEF solicitará o endereço base das portas de E/S configuradas na placa. O endereço deve ser digitado em valor hexadecimal (*default = 02F0h*). O usuário deve observar que, nesta opção, o usuário apenas informa ao STEF qual o endereço para o qual a placa está configurada. A modificação do endereço em si deve ser efetuada na própria placa, através de estrapes.

A **Configuração de Cores** permite que o usuário redefina as cores utilizadas em tela para todo o sistema. Ao selecionar esta opção será mostrado um sub-menu com as opções de cores modificáveis. O usuário deverá escolher a sua opção e em seguida a cor desejada. As modificações de cores realizadas são válidas durante a sessão de uso do sistema. Ao final da sessão, o sistema perguntará ao usuário se ele deseja ou não gravar em disco a nova configuração de cores.

Ao selecionar-se a opção de **Configuração de Idioma** o STEF apresentará na tela um submenu contendo as opções de idioma disponíveis. Nesta versão do sistema, o Português é o único idioma disponível. O objetivo da configuração de idioma é permitir ao usuário selecionar o idioma a ser utilizado para os menus de opções e mensagens de maneira geral. Estão previstos, para versões futuras, os seguintes idiomas: Português, Inglês, Espanhol e Francês.

A opção de **configuração de impressora** deverá permitir a definição do tipo e modelo da impressora instalada, possibilitando assim um melhor uso de características particulares de cada tipo de impressora. Na versão atual esta opção na se encontra implementada. Para a impressão de textos o sistema assume que não é necessária nenhuma tabela de tradução de caracteres. Para a impressão de gráficos o sistema assume a impressora definida pelo comando **GRAPHICS** do sistema operacional (definido no arquivo STEF.BAT). Maiores informações sobre a configuração do tipo de impressora para o comando GRAPHICS, podem ser obtidas no manual do usuário do MS-DOS.

A opção de **configuração de *plotter***, a exemplo da configuração de impressora e da configuração de idiomas, não se encontra disponível na versão atual do STEF.

Através da operação de **configuração do editor de textos**, o usuário poderá definir qual o editor de textos a ser ativado para a descrição ou alteração textual do programa de testes funcionais. Ao selecionar-se esta opção, o STEF apresentará na tela uma janela solicitando a linha de comando para a ativação do editor de textos (*pathlist* completo e nome do arquivo a executar). O STEF assume como *default* o editor de textos incorporado ao SDP. As informações armazenadas pelo editor *default* do sistema obedecem ao formato ASCII padrão e, portanto, é aconselhável que o usuário escolha um editor de textos que use esse mesmo formato. Devemos ressaltar também que, dependendo da quantidade de memória requerida para a utilização do editor de textos, pode ocorrer do sistema não ser capaz de ativá-lo. Nesse caso, recomendamos a utilização do editor *default* do sistema.

III.6 - O Sistema de Auxílio ao Usuário (*Help*)

O mecanismo de auxílio ao usuário do STEF, a exemplo das outras ferramentas do SAD, é sensível ao contexto operacional e foi implementado na forma de hipertexto, podendo ser ativado através da opção específica no menu principal ou, a qualquer instante, através da tecla <F1 >.

Se ativado através da tecla <F1 >, é mostrada ao usuário a tela de auxílio referente ao contexto atual. Ativando o *help* através do menu principal, a tela de auxílio apresentará os principais tópicos do sistema, devendo o usuário selecionar o tópico de seu interesse.

Na tela de auxílio, algumas palavras serão apresentadas em destaque (sublinhadas ou numa cor diferenciada). Essas palavras são palavras-chave, que dão acesso a outros contextos do hipertexto, referentes àquele tópico específico. Também nessa tela, uma das palavras será apresentada em vídeo reverso. Essa é a palavra-chave atualmente selecionada. Através das setas, o usuário poderá se deslocar pelas palavras-chave do contexto atual e, teclando <ENTER>, ativará o contexto selecionado. Através da tecla <PgUp> pode-se retornar ao contexto anterior e, com a tecla <ESC>, o usuário abandona o sistema de *help*.

Anexo V - Principais Rotinas e Algoritmos

a) Rotinas de Uso Geral

a.1) Gerenciamento dos Menus de Opções

```
(*************************************************************************
+*+
* BGInicMenu -      Aloca memoria e inicializa as estruturas de dados relacionadas
*                   aos menus de opcao do sistema
*
* CHAMADA PADRAO:
*
*             BGInicMenu;
*
* OUTRAS SAIDAS:    (Dados globais Alterados)
*
*             InicializouMenu
*
-*-
* NOTAS:
*             Esta rotina deve ser ativada logo apos a carga do programa, antes que
* seja feita qualquer outra operacao sobre menus.
*
*************************************************************************)

Procedure BGInicMenu;

Var I: Integer;

begin
InicializouMenu:= True;
Mark(BGInicioHeapMenu);
For i:=0 to MAXOPCMENU do
             begin
             New(Menu[i]);
             FillChar( Menu[i]^, sizeof(Menu[i]^), #0);
             end;
end;


(*************************************************************************
+*+
* BGFinalizaMenu -      Libera memoria dinamica alocada para estruturas dos menus
*                       de opcoes.
*
* CHAMADA PADRAO:
*
*             BGFinalizaMenu;
*
* OUTRAS SAIDAS:      (Dados Externos ou globais Alterados)
```

```
*
*              InicializouMenu
*
-*-
* NOTAS:       (opcional)
*              Esta rotina deve ser ativada somente ao final do programa.
*****************************************************************************)

Procedure  BGFinalizaMenu;
begin
Release(BGInicioHeapMenu);
InicializouMenu:= False;
end;


(*****************************************************************************
+*+
* BGLeDefMenu -  Le o arquivo que define a estrutura de menus de opcao para o
*                sistema.
*
* CHAMADA PADRAO:
*
*              BGLeDefMenu(NomArq);
*
*              Nome       i/o    tipo        Descricao
*              NomArq     i      String      Nome do arquivo de definicao dos menus.
*
* SAIDA DE EXCECAO:
*
*              Se arquivo nao existente, ou algum erro no formato do arquivo, e'
* impossivel prosseguir. Aborta o programa.
*
* ROTINAS CHAMADAS:
*
* BGErro -           Mostra mensagem de erro na tela.
* BGFinalizaMenu-    Libera memoria dinamica alocada para menus de opcoes.
* BGLTrim -          Retira brancos do inicio de uma string
* BGTrim -           Retira brancos do final de uma string
* ErroFormato -      Dá mensagem de "erro no formato do arquivo" e sai... fora.
* TiraPalavra -      Retira a primeira palavra do inicio de uma linha
*
-*-
*
*****************************************************************************)

Procedure BGLeDefMenu(Nomarq:String);

Var
              Arq: Text;
              i: Byte;
              Pos1: Byte;
              Staux: Str128;
              Palavra: Str128;
              Code: Integer;

              {*************************************************************}
              Procedure  ErroFormato;
              begin
              BGFinalizaMenu;
```

```
                BGErro(BG_Mens10+NomArq+BG_Mens11,True);
                end;

                {*****************************************************}

                Function TiraPalavra(Var Linha:Str128):Str128;

                Var Pos1: Byte;

                begin
                Pos1:= Pos(',', Linha);
                if Pos1=0 then
                      begin
                      TiraPalavra:= BGTrim(BGLTrim(Linha));
                      Linha:= '';
                      end
                else
                      begin
                      TiraPalavra:= BGTrim(BGLTrim(Copy(Linha, 1, Pos1-1)));
                      Linha:= Copy(Linha, Pos1+1, Length(Linha));
                      end;
                end; { of TiraPalavra }

begin

if Not InicializouMenu then
                BGErro(BG_Mens12, True);

Assign(Arq,Nomarq);
{$I-} Reset(Arq); {$I+}
if IOResult<>0 then
                BGErro(BG_Mens13+Nomarq+BG_Mens11, True);

i:=0;
While not Eof(Arq) do
                begin
                Readln(arq, StAux);
                StAux:= BGTrim(StAux);
                if Length(StAux) > 0 then
                      begin
                      Palavra:= TiraPalavra(StAux);
                      if Palavra='' then
                            ErroFormato
                      else
                            begin
                            Menu[i]^.Msg := Palavra;          { Texto da Opcao do Menu      }
                            Palavra:= TiraPalavra(StAux);     { Numero de Sub-Opcoes do Menu }
                            {$R-} Val( Palavra, Menu[i]^.Nopc, Code); {$R+}
                            if Code<>0 then
                                  ErroFormato;
                            Palavra:= TiraPalavra(StAux);     { Indice da primeira sub-opcao }
                            {$R-} Val( Palavra, Menu[i]^.Ind, Code); {$R+}
                            if Code<>0 then
                                  ErroFormato;
                            Palavra:= TiraPalavra(StAux);     { Prioridade minima para execucao }
                            {$R-} Val( Palavra, Menu[i]^.Prior, Code); {$R+}
                            if Code<>0 then
                                  ErroFormato;
                            Palavra:= TiraPalavra(StAux);     { Prioridade minima para execucao }
```

```
                        {$R-} Val( Palavra, Menu[i]^.Hlp, Code); {$R+}
                        if Code<>0 then
                              ErroFormato;
                        i:= Succ(i);
                        end;
                  end;
            end;
Close(Arq);
NOP_LIVRE:= 0;
MSG_LIVRE:= i;
end;


(*****************************************************************************
+*+
* BGOpcaoMenu - Le uma opcao de um menu.
*
* CHAMADA PADRAO:
*
*           Opcao := BGOpcaoMenu(Col, Lin, TamMsg, Nopc, Ind);
*
*           Nome      i/o   tipo       Descricao
*           Col       i     inteiro    Coluna inicial p/ opcao (coord. texto)
*           Lin       i     inteiro    Linha inicial p/ opcao (coord. texto)
*           TamMsg    i     inteiro    maximo de carateres das mensagens
*           NOpc      i     inteiro    Numero de opcoes do menu
*           Ind       i     inteiro    Posicao no vetor Menu
*
* RETORNA:
*
*
* OUTRAS SAIDAS:
*
*CurHlp - Contexto atual para o sistema de auxilio ao usuario.
*
* ROTINAS CHAMADAS:
*
* BGLeChar    - Le um caracter digitado pelo usuario.
* BGReverso   - Coloca uma regiao da tela em video reverso.
*
-*-
*NOTAS:
*
*****************************************************************************)


Function  BGOpcaoMenu (col,lin,tammsg,nopc,ind :integer) : integer ;


Var i: integer ;
tecla: char ;
x1,y1,x2,y2 : integer ;
ch: char ;

begin
i:= 1 ;
Repeat
            if (Lin<=2) then
                  begin
```

```
                x1:= (i-1) * (TamMsg+2) * Xch + MeioX ;
                y1:= lin * Ych ;
                x2:= i* (tammsg+2) * Xch ;
                y2:= (lin + 1) * Ych -1 ;
                end
        else
                begin
                x1:= (col * Xch) - MeioX ;
                y1:= (lin + i) * Ych ;
                x2:= (col + tammsg) * Xch + MeioX;
                y2:= (lin + i + 1) * Ych -1 ;
                end;

        CurHlp:=   Menu[ind+i-1]^.Hlp;
        BGReverso (x1,y1,x2,y2) ;
        BGLeChar(Tecla,Ch);
        BGReverso(x1,y1,x2,y2);

        Case ch of
                NORTE,
                OESTE :
                        if i = 1 then
                                i:= nopc
                        else
                                i:= pred (i) ;

                SUL,
                LESTE:
                        if i = nopc then
                                i:= 1
                        else
                                i:= Succ (i) ;
        end; (* of case *)

Until (tecla = Esc) or (tecla = ENTER) ;

if tecla = ESC then
        BGOpcaoMenu := 0
else
        BGOpcaoMenu := i ;
end;


(*************************************************************************
+*+
* BGMenu - Faz o gerenciamento dos menus de opcoes.
*
* CHAMADA PADRAO:
*
*       Opcao:= BGMenu(Col, Lin, NOpc, Ind)
*
*       Nome    i/o   tipo      Descricao
*       Col     i     inteiro   Coluna inicial p/ menu (coord. texto)
*       Lin     i     inteiro   Linha inicial p/ menu (coord. texto)
*       NOpc    i     inteiro   Numero de opcoes do menu
*       Ind     i     inteiro   Posicao inicial no vetor Menu
*       Reply   i     Boolean   Indica que o menu deve ser somente
*                               mostrado na tela, sem permitir selecionar
*                               nenhuma  opcao.
```

```
*
* RETORNA:
*
*              < 0, caso tenha sido executada uma operacao que e' folha da arvore de
*                   menus (Foi_nas_folhas = True).
*              0,   caso o usuario tecle <ESC> (retornar para o nivel anterior do menu)
*              > 0, caso o usuario tenha selecionado uma opcao valida do menu que nao
*                   seja folha da arvore de opcoes. Neste caso o valor retornado
*                   corresponde ao item selecionado no menu, ou seja, retorna "1" caso
*                   o usuario selecione a primeira opcao do menu, "2" para a segunda
*                   opcao, etc...
*
* OUTRAS ENTRADAS: (dados globais utilizados)
*
*   CurHlp - Identifica o contexto atual para o sistema de help
*   Foi_nas_folhas - Indica se chegou as folhas da arvore de menus
*   Menu - Contem a descricao da arvore dos menus de opcoes
*
* OUTRAS SAIDAS:   (dados globais alterados)
*
*   CurHlp
*   Foi_nas_folhas
*
* ROTINAS CHAMADAS:
*
* nome              descricao
* BGExecMenu        Executa um operacao que e' folha da arvore de menus
* BGLeChar          Leitura de uma tecla do teclado.
* BGMensagemGr      Escreve mensagem no rodape da tela.
* BGMenu            ... esta rotina ...
* BGMoldura         Desenha a moldura para o menu de opcoes.
* BGOpcaoMenu       Permite a escolha de uma opcao dos menus.
* BGRestauraJanela  Restaura regiao da tela previamente salva.
* BGSaiTexto        Escreve uma String numa determinada posicao da tela.
* BGSalvaJanela     Salva conteudo de uma determinada regiao da tela.
*
-*-
* NOTAS:
*
*****************************************************************************)


Function BGmenu  (col,lin,nopc,ind:integer; reply: Boolean) : Integer;


Var
              Ch, ChExt: Char;
              i, tammsg : byte ;
              Staux : String[40];
              Size_aux : Word ;
              P_aux : Pointer ;
              V: ViewPortType ;
              x1,x2,y1,y2 : integer ;
              Res : integer ;
              Prox_Menu : integer ;
              ColAux, LinAux: Integer;
              LocalHelp: Integer;

begin
```

```
if nopc <= 0 then                              { Chegou na folha. Executa algo }
            exit ;
if ind=0 then Exit;                            { Voltou do prim. nivel, sai do menu }

tammsg := 0 ;
for i := 0 to nopc-1 do                        { Acha a Maior Mensagem do Menu }
            if length (menu[i+ind]^.msg) > tammsg then
                  TamMsg := length (menu[i+ind]^.msg);

SetLineStyLe (SolidIn ,0, NormWidth);          { Calcula coordenadas tela grafica }
SetColor(Cor_Menu);
if (Lin=1) then
            begin
            x1 := 0 ;
            y1 := (lin) * ych + MeioY ;
            x2 := (1+ NOpc*(TamMsg+2)) * XCh ;
            y2 := (lin+2) * ych + (MeioY div 2);
            end
else
            begin
            x1 := (col-1) * xch ;
            y1 := (lin +1) * ych + MeioY -1 ;
            x2 := (col+3+tammsg) * xch ;
            y2 := (lin+nopc+3) * ych - MeioY + 1;
            end;
GetViewSettings (V) ;                  { Salva ViewPort Inicial }
                                       { Coloca na ViewPort c/ Coord. Absol}

SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
BGSalvaJanela(x1,y1,x2,y2,P_Aux,Size_Aux);     { Salva regiao da janela   }
SetViewPort (x1,y1,x2,y2, ClipOn) ;            { Limpa a janela p/ o Menu }
ClearViewPort ;

BGMoldura(x1,y1,x2,y2, Cor_Fundo, Cor_Menu);   { Faz a moldura do menu }
SetColor(Cor_menu);
if Lin=1 then
            for i:= 0 to (nopc-1) do
                  BGSaiTexto (col+1+i*(TamMsg+2),lin+1,menu[i+ind]^.msg)
else
            for i:= 1 to nopc do BGSaiTexTo (col+1,lin+i+1,menu[i+ind-1]^.msg);

if Reply then
            begin
            LocalHelp:= CurHlp;
            Repeat
                  BGColMenu:= Col;
                  BGLinMenu:= Lin;
                  foi_nas_folhas:= False;
                                                { Le a opcao do Usuario }
                  Res:= BGOpcaoMenu(Col+1,Lin+1,tammsg,nopc,ind);
                  if Res > 0 then               { se uma opção válida  }
                        begin
                        if Lin=1 then           { Calcula coord. p/ prox. menu }
                              begin
                              ColAux:= Col + (Res-1)*(TamMsg+2);
                              LinAux:= Lin + 1;
                              end
                        else
                              begin
```

```
                    ColAux:= Col+2;
                    LinAux:= Lin+1+Res;
                    end;
                                        { Ve posic. no vetor p/ sub-menu }
                Prox_Menu:= Menu[Ind+Res-1]^.Ind;
                                        { e o contexto de Help associado }
                CurHlp:=   Menu[Ind+Res-1]^.Hlp;

                if Prox_Menu>0 then     { Se tem sub-menu, entao    }
                    Begin
                                        { Chama de novo BGMenu (recursao)}
                    With Menu[Ind+Res-1]^ do
                    Res:=BGmenu(ColAux, LinAux, Nopc, Ind, Reply);
                    end
                else                    { Opcao  uma folha da arvore   }
                    begin
                    Foi_nas_Folhas:= True; { Assume que vai sair fora dos menus }
                                    { BGExecMenu pode dizer p/ ficar no menu }
                    BGExecMenu(Prox_Menu);
                    if Foi_nas_Folhas then      { Se é para continuar nos menus }
                        begin
                        BGMenu:= Prox_Menu;{ Restaura último nível de menu }
                        BGRestauraJanela(x1,y1,P_Aux,Size_Aux);
                        CurHlp:= LocalHelp;  { Atualiza contexto de Help  }
                        exit;                { e sai fora ...             }
                        end;
                    end;
                end
            else                        { Teclou ESC , sai da rotina para  }
                begin                   { voltar um nvel nos menus         }
                BGMenu:= 0;
                BGRestauraJanela(x1,y1,P_Aux,Size_Aux);
                CurHlp:= LocalHelp;
                Exit;
                end;
            CurHlp:= LocalHelp;
        Until Foi_nas_folhas ;
        end
else                        {NOT Reply = não quer que leia opção do menu }
        begin
        BGMensagemGr(BG_Mens16);
        Repeat
            BGLeChar(Ch,ChExt);
        Until ch=ESC;
        end;

BGRestauraJanela(x1,y1,P_Aux,Size_Aux);
With V do
        SetViewPort( x1, y1, x2, y2, Clip);
end;


(*************************************************************************
+*+
* BGHotMenu-    Faz o gerenciamento dos menus de opcoes para ativacoes parciais
*               dos menus (normalmente para funcoes ativadas por "hot-keys")
*
* DESCRICAO:
*       Esta rotina e' usada para ativar uma sub-arvore dos menus de
```

```
*               opcoes. Os parametros "Col" e "Lin" determinam as coordenadas (em modo
*               texto) onde deve ser apresentado o menu de opcoes. A rotina busca no
*               vetor "Menu" uma string que coincida com o parametro "texto" e ativa
*               a rotina "BGMenu" a partir do submenu referente a essa posicao do vetor.
*
*
* CHAMADA PADRAO:
*
*               Opcao:= BGHotMenu(Col, Lin, Texto);
*
*               Nome      i/o   tipo       Descricao
*               Col       i     inteiro    Coluna inicial p/ menu (coord. texto)
*               Lin       i     inteiro    Linha inicial p/ menu (coord. texto)
*               Texto     i     string     Texto da opcao para a qual deve ser
*                                          ativado o menu.
*
* RETORNA:
*
*               < 0, caso tenha sido executada uma operacao que e' folha da arvore de
*                    menus (Foi_nas_folhas = True).
*               0,   caso o usuario tecle <ESC> (retornar para o nivel anterior do menu)
*               > 0, caso o usuario tenha selecionado uma opcao valida do menu que nao
*                    seja folha da arvore de opcoes. Neste caso o valor retornado
*                    corresponde ao item selecionado no menu, ou seja, retorna "1" caso
*                    o usuario selecione a primeira opcao do menu, "2" para a segunda
*                    opcao, etc...
*
* SAIDAS DE EXCECAO:
*
*               Aborta o programa caso o valor do parametro "texto" nao seja encontrado
*               no vetor "Menu".
*
* OUTRAS ENTRADAS: (dados globais utilizados)
*
*               CurHlp - Identifica o contexto atual para o sistema de help
*               Menu - Contem a descricao da arvore dos menus de opcoes
*
* OUTRAS SAIDAS:  (dados globais alterados)
*
*               Foi_nas_folhas - Indica se chegou as folhas da arvore de menus
*
* ROTINAS CHAMADAS:
*
*               nome              descricao
*               BGErro            Mostra mensagem de erro e opcionalmente aborta o programa
*               BGMenu            Gerenciamento e controle dos menus de opcao
*               BGRestauraJanela  Restaura regiao da tela previamente salva.
*               BGSaiTextXY       Escreve uma String numa determinada posicao da tela.
*               BGSalvaJanela     Salva conteudo de uma determinada regiao da tela.
*
-*-
* NOTAS:
*
*****************************************************************************)


Function BGHotMenu(col,lin: Integer; Texto: String) : Integer;
```

```
Var
            i: Integer;
            V: ViewPortType;
            x1,y1,x2,y2: Integer;
            P_Aux: Pointer;
            Size_Aux: Word;
            HelpAnter: Integer;
begin
i:=1;
while (i<=MAXOPCMENU) and (Menu[i]^.Msg<>Texto) do
            Inc(i);
if i>MAXOPCMENU then
            BGErro(BG_Mens17, True);

GetViewSettings(V);
SetViewPort( 0, 0, GetMaxX, GetMaxY, ClipOn);

x1 := (col-1) * xch ;                          { Calcula Coordenadas Graficas }
y1 := (lin+1) * ych ;
x2 := (col+3+Length(Texto)) * xch ;
y2 := (lin+2) * ych + MeioY ;
BGSalvaJanela(x1,y1,x2,y2,P_Aux,Size_Aux); { Salva regiao da janela   }
SetViewPort (x1,y1,x2,y2, ClipOn) ;       { Limpa a janela p/ o Menu }
ClearViewPort ;
if DISPLAYMONO then
            SetColor(Blue)
else
            SetColor(Cor_Menu);

BGSaiTextXY(x1, y1+(MeioY div 2), ' '+Texto+' ', True);

HelpAnter:= CurHlp;
With Menu[i]^ do
            begin
            CurHlp:=   Hlp;
            BGHotMenu:= BGMenu(Col, Lin+1, NOpc, Ind, Reply);
            CurHlp:= HelpAnter;
            end;
BGRestauraJanela(x1,y1,P_Aux,Size_Aux);
With V do
            SetViewPort(x1,y1,x2,y2,Clip);
end;
```

a.2) Gerenciamento do Sistema de Auxlio ao Usurio

```
Const
            Header : String[50]= ' Tratamento Grafico - Auxilio ao Usuario ';
            MaxLinHlp = 12;
            MaxColHlp = 50;

Type
            RegHelp = Record              {Estrutura principal do Arquivo de hipertexto }
                        LinHelp : Array[1..MaxLinHlp] of String[MaxColHlp+30];
                        end;
```

```
Var
            HelpRec :  RegHelp;
            ArqHelp : File of RegHelp;
            CorHelp : Array[False..true] of Byte;
            CorNormal : Byte;
            CorNegrito : Byte;

(*************************************************************************
+*+
* BGAtivaHelp - Ativa e gerencia o acesso ao sistema de help.
*
* CHAMADA PADRAO:
*
*           BGAtivaHelp(NomeArq. PagAtual,PagInic);
*
*           Nome      i/o   tipo      Descricao
*           NomeArq   i     String    Nome do arquivo de help a ser usado.
*           PagAtual  i     String    Contexto (pagina) de Help atual.
*           PagInic   i     String    Contexto (pagina) de Help Inicial.
*
* SAIDA DE EXCECAO:
*
*           Caso o arquivo especificado nao exista ou possua formato invalido,
* e' emitida a mesnagem de erro correspondente.
*           Caso o contexto especificado nao exista, e' mostrada uma tela em
*  branco.
*
* ROTINAS CHAMADAS:
*
* Nome                  Descricao
* Abre_Arq_Help         Abre o arquivo de help.
* Acha_Link_Anter       Acha hot-link anterior para o contexto atual.
* Acha_Prox_Link        Acha proximo hot-link definido p/ o contexto atual.
* BGErro                Mostra mensagem de erro no rodape' da tela.
* BGRestauraJanela      Restaura regiao da tela previamente salva por BGSalvaJanela
* BGSalvaJanela         Salva conteudo de uma regiao da tela.
* Le_Reg_Help           Le um registro (Contexto) do arquivo de help.
* Mostra_Pag_Help       Mostra o contexto atual de help.
* MostraLinHelp         Mostra uma linha do contexto atual.
* PoePilha              Empilha um contexto de help.
* TiraPilha             Desempilha um contexto de Help.
*
-*-
*
*************************************************************************)

Procedure BGAtivaHelp(NomeArq: String; PagAtual,PagInic: Word);

Const
            Color : Byte = Black*16+White;
            MaxStackSize = 25;

Type
            StackRec = Record
                  Page : Byte;
                  Lin,
                  Col  : Integer;
            end;
```

```
Var
            Result  : Integer;
            Stack : Array[0..MaxStackSize] of StackRec;
            AHelpRec: RegHelp;
            Ch : CHar;
            StackLvl: Byte;
            StartCol: Integer;
            Linked,
            Load : Boolean;
            V : ViewPortType;
            PAux : Pointer;
            SizAux  : Word;
            xi, yi,
            xf, yf  : Integer;
            XPos,
            YPos    : Integer;

            {-------------------------------------------------------------}

            FUNCTION TiraPilha: Byte;       { Desempilha contexto anterior de help }
            Begin
            If StackLvl>1 then
                  Begin
                  Dec(StackLvl);
                  Load:=True;
                  end;
            TiraPilha:=StackLvl;
            end;  {TiraPilha}

            {-------------------------------------------------------------}

            FUNCTION PoePilha(pageNum: Byte): Byte;  { Empilha 1 contexto de help }
            Begin
            Inc(StackLvl);
            Stack[StackLvl].Page:=PageNum;
            Stack[StackLvl].Col:=1;
            Stack[StackLvl].Lin:=1;
            PoePilha:=StackLvl;
            end;  {push stack}

Begin               { BGAtivaHelp }

XPos:= 10;
YPos:= (GetMaxY+1) div (YCh*5);
GetViewSettings(V);
With V do SetViewPort( 0, 0, GetMaxX, GetMaxY, ClipOn);

xi:= XPos*XCh;
yi:= (YPos-2)*YCh;
xf:= (XPos+MaxColHlp+3)*XCh;
yf:= (YPos+2+MaxLinHlp)*YCh;
BGSalvaJanela( xi, yi, xf, yf, Paux, SizAux);

CorHelp[False]:= Cor_Menu;                  { Cor Texto Normal    }
CorHelp[True] := Cor_Msg;                   { Cor Texto Hot-Link  }
CorNormal:= Cor_Menu;                       { Cor Normal          }
CorNegrito:= Cor_Menu xor $F;               { Cor txt selecion.   }
```

```
If PagAtual=0 then PagAtual:=1;                    { Certificar que e' pagina valida }
Result:=Abre_ArqHelp (NomeArq);
If Result=0 then
            Begin
            Load:=true;
            FillChar(Stack,SizeOf(Stack),0);
            StackLvl := 0;
            If PagInic in [1..255] then
                  StackLvl:=PoePilha(PagInic);
            If (PagAtual in [1..255]) and (PagAtual<>PagInic) then
                  StackLvl:=PoePilha(PagAtual);
            Repeat
                  With Stack[StackLvl] do
                        Begin
                        If Load then                {Preciso ler dados de uma nova pag.}
                              Begin
                              Result:=Le_RegHelp(AHelpRec,Page);
                              Mostra_Pag_Help(XPos,YPos,AHelpRec);
                              SetColor(Cor_Menu);
                              Moveto((XPos+MaxColHlp-6)*XCh, (YPos+MaxLinHlp+1)*YCh);
                              If StackLvl>1 then
                                    OutText('ESC,PgUp')
                              else
                                    OutText('ESC=Sair');
                              Linked:=Acha_Prox_Link(Col,Lin,80,MaxLinHlp,AHelpRec);
                              Load:=False;
                              end;
                        If Linked then              {Tem um hotlink. Vai para lá.}
                              Begin
                              StartCol := Col;
                              While Ord(AHelprec.LinHelp[Lin,StartCol-1])>127 do
                                    Dec(StartCol);
                              MostraLinHelp(XPos,YPos+Lin,StartCol,Pred(Col),
                              AHelpRec.LinHelp[Lin]);
                              end;
                        Ch:= Readkey;
                        if Ch=#0 then Ch:= Readkey;

                        MostraLinHelp(XPos,YPos+Lin,0,0,AHelpRec.LinHelp[Lin]);
                        Case Ch of              { Trata da navegacao atraves do sistema de help }
                              LESTE,
                              TAB   :
                                    Begin
                                    Inc(Col);
                                    Linked:=Acha_Prox_Link(Col,Lin,80,MaxLinHlp,AHelpRec);
                                    end;
                              ENTER :
                                    If Linked then
                                          Begin
                                          Load:=true;
                                          If (StackLvl>1) and (Stack[StackLvl-1].Page=
                                                Ord(AHelpRec.LinHelp[Lin,Col+1])) then
                                                StackLvl:=TiraPilha
                                          else
                                                StackLvl:=PoePilha(Ord(
                                                AHelpRec.LinHelp[Lin,Col+1]));
                                          end;
                              OESTE:
                                    Begin
```

```
                              Dec(Col);
                              Linked:=Acha_Link_Anter(Col,Lin,1,1,AHelpRec);
                              end;
                         SUL:
                              Begin
                              Col:=1;
                              If Lin<MaxLinHlp then Inc(Lin) else Lin:=1;
                              Linked:=Acha_Prox_Link(Col,Lin,80,MaxLinHlp,AHelprec);
                              end;
                         NORTE:
                              Begin
                              If Lin>1 then Dec(Lin) else Lin:=MaxLinHlp;
                              If Col<Length(AHelpRec.LinHelp[Lin]) then
                              Col:= Length(AHelpRec.LinHelp[Lin]);
                              Linked:=Acha_Link_Anter(Col,Lin,1,1,AHelprec);
                              end;
                         PGUP  :
                              StackLvl:=TiraPilha;
                         PGDN  :
                              Begin { Pode ser usada futuramente ... } end;
                         end; {of Case }
                    end;
               Until Ch=ESC;
               end
          else
               BGErro('Arquivo de HELP nao Disponivel!',False);

          {$I-} Close(ArqHelp ); result:=IOResult; {$I+}

          BGRestauraJanela(xi, yi, Paux, SizAux);
          With V do
               SetViewPort( x1, y1, x2, y2, Clip);

          end; { BGAtivaHelp }
```

b) Rotinas do STAG

b.1) Cálculos de Regressões

```
(*************************************************************************
+*+
* GSRegrLin - Faz calculo da regressao linear de uma tabela de pontos X,Y
*
*          Esta rotina calcula a regressao linear de um conjunto de pontos atraves
* do metodo dos minimos quadrados.
*          A reta calculada pode ser descrita pela equacao:
*               y = a + b.x
*          onde
*               a = Ponto de Intersecao com o eixo y.
*               b = Inclinacao da Reta.
*
*          Sao calculados tambem os valores do coeficiente de determinacao, de
*correlacao e o desvio padrao.
*
```

```
* CHAMADA PADRAO:
*
*              GSRegrLin( Sx,Sy,Pi,Pf,LResRegr);
*
*              Nome       i/o   tipo        Descricao
*              Sx         i     inteiro     Variavel do vetor PtGraf para eixo X
*              Sy         i     inteiro     Variavel do Vetor PtGraf para eixo Y
*              Pi         i     inteiro     Ponto Inicial do vetor
*              Pf         i     inteiro     Ponto final do vetor
*              LResRegr   i/o   ParmRegr    Resultados da regressao
*
* OUTRAS ENTRADAS:  (Dados Globais Utilizados)
*
*              PtGraf
*
* ROTINAS CHAMADAS:
*
* nome              descricao
* GSDesloc          Desloca todos os pontos para o primeiro quadrante.
-*-
* NOTAS:
*              Se necessario, e' feito um deslocamento dos eixos x e/ou y para que
* os pontos da tabela p/ interpolacao fiquem todos no 1o. quadrante
*
*************************************************************************)

Procedure GSRegrLin( Sx,Sy,Pi,Pf: Integer; Var LResRegr: ParmRegr);

Var
              i, n: Integer;
              j,k,l,m: Real;
              x,y: Real;
begin

GSDesloc(Sx,Sy,Pi,Pf,LResRegr);       { Coloca todos ptos. no 1.o quadrante }
With LResRegr do
              begin
              SSx:= Sx;
              SSy:= Sy;
              r2:=0;
              j:= 0; k:=0 ; l:=0; m:=0;
              n:= Pf - Pi + 1;
              for i:= Pi to Pf do
                    begin
                    x:= PtGraf[Sx,i] -DeslX;
                    y:= PtGraf[Sy,i] -DeslY;
                    j:= j+ x;
                    k:= k+ y;
                    l:= l+ x*x;
                    m:= m+ y*y;
                    r2:= r2+ x*y;
                    end;
              b:= ( n * r2 - k * j) / ( n * l - j*j);
              a:= ( k - b * j ) / n ;
              j:= b* (r2 - j * k / n);
              m:= m - k*k / n;
              k:=  Abs(m -j);
              r2:= Abs(j /m);
              r:= Sqrt( r2 );
```

```
            Desv:= Sqrt( k / (n-2));
            End;
end;

(*****************************************************************************
+*+
* GSRegrGeo - Calcula a regressao geometrica para uma tabela de pontos (x,y)
*
*           A rotina calcula a regressao pelo metodo dos minimos quadrados.
*           O metodo consiste da linearizacao de uma funcao do tipo ( y = A * x^b )
* para ln(y) = ln(a) + b* ln(x)
*           Apos isso e' aplicado o mesmo metodo utilizado para o calculo de
* regressoes lineares.
*
* CHAMADA PADRAO:
*
*           GSRegrGeo( Sx,Sy,Pi,Pf,LResRegr);
*
*           Nome      i/o   tipo       Descricao
*           Sx        i     inteiro    Variavel do vetor PtGraf para eixo X
*           Sy        i     inteiro    Variavel do Vetor PtGraf para eixo Y
*           Pi        i     inteiro    Ponto Inicial do vetor
*           Pf        i     inteiro    Ponto final do vetor
*           LResRegr  i/o   ParmRegr   Resultados da regressao
*
* SAIDA DE EXCECAO:
*
*           A rotina e' abortada caso os dados nao estejam todos no primeiro
* quadrante ou todos no quarto quadrante.
*
* ROTINAS CHAMADAS:
*
*nome       descricao
*BGErro     Emite mensagem de erro na tela.
*GSDesloc   Desloca todos os pontos para o primeiro quadrante.
-*-
* NOTAS:
*
* Para a aplicacao do calculo da regressao geometrica e' necessario que os
* dados estejam todos exclusivamente no primeiro ou no quadrante.
* Se necessario e' feito um deslocamento da origem dos eixos.
*
*****************************************************************************)

Procedure GSRegrGeo( Sx,Sy,Pi,Pf: Integer; Var LResRegr: ParmRegr);

Var
            i,n: Integer;
            j,k,l,m: Real;
            X,Y: Real;
begin
GSDesloc(Sx,Sy,Pi,Pf,LResRegr);
if (LResRegr.DeslX<0) then
            begin
            BGErro('Regressao Geometrica Exige Valores Positivos em "X"',False);
            TipoRegr:=0;
            FazRegressao:= False;
```

```
                Exit;
                end
else
                if LResRegr.DeslY<0 then    { Posso aplicar regressao no Quarto Quadrante... }
                     begin                  { desde que y sempre menor ou igual a 0        }
                     For i:= Pi to Pf do
                          if PtGraf[Sy,i]>0 then
                               begin
                               BGErro('Pontos em "Y" Invalidos p/ Regressao Geometrica',False);
                               TipoRegr:=0;
                               FazRegressao:= False;
                               Exit;
                               end;
                     end;

With LResRegr do
                begin
                SSx:= Sx;
                SSy:= Sy;
                N:= Pf - Pi + 1;
                j:=0 ; k:=0 ; l:=0; m:=0;
                r2:=0;
                for i:= Pi to Pf do
                     begin
                     if PtGraf[Sx,i]>0 then
                          x:= Ln(PtGraf[Sx,i])
                     else
                          x:= -QInfinito;
                     if Abs(PtGraf[Sy,i])>0 then
                          y:= Ln(Abs(PtGraf[Sy,i]))
                     else
                          y:= -QInfinito;
                     j:= j+ x;
                     k:= k+ y;
                     l:= l+ x*x;
                     m:= m+ y*y;
                     r2:= r2+ x*y;
                     end; {For}
                b:= ( n * r2 - k * j ) / ( n * l - j*j);
                a:= ( k - b * j ) / n ;
                j:= b * ( r2 - j * k / n );
                m:= m - k * k / n;
                k:= Abs(m - j);
                r2:= Abs(j / m);
                r:= Sqrt(r2);
                Desv:= Sqrt( k / (n - 2));
                end;
end;

(*****************************************************************************
+*+
* GSRegrExp - Calcula a regressao exponencial de um conjunto de pontos (x,y)
*
* O calculo da regressao exponencial consiste da linearizacao de uma curva
* do tipo ( y = A e^(Bx) ) pela formula ln(y) = ln(a) + b*x, aplicando-se em
* seguida o calculo de uma regressao linear, pelo metodo dos minimos
* quadrados.
*
* CHAMADA PADRAO:
```

```
*
*            GSRegrExp( Sx,Sy,Pi,Pf,LResRegr);
*
*            Nome      i/o   tipo       Descricao
*            Sx        i     inteiro    Variavel do vetor PtGraf para eixo X
*            Sy        i     inteiro    Variavel do Vetor PtGraf para eixo Y
*            Pi        i     inteiro    Ponto Inicial do vetor
*            Pf        i     inteiro    Ponto final do vetor
*            LResRegr  i/o   ParmRegr   Resultados da regressao
*
* SAIDA DE EXCECAO:
*
*            Caso os pontos da tabela nao estejam todos no primeiro ou no terceiro
* quadrante o calculo nao e' realizado e e' dada uma mensagem de erro.
*
* ROTINAS CHAMADAS:
*
*            BGErro    Emite mensagem de erro na tela.
*            GSDesloc  Desloca todos os pontos para o primeiro quadrante.
*
-*-
* NOTAS:
*
*  Para a aplicacao da regressao exponencial e' necessario que todos os
*  pontos da tabela estejam exclusivamente no primeiro quadrante ou no
*  terceiro quadrante.
*
**************************************************************************)

Procedure GSRegrExp( Sx,Sy,Pi,Pf: Integer; Var LResRegr: ParmRegr);

Var
            i,n: Integer;
            x,y: Real;
            j,k,l,m: Real;
begin
GSDesloc(Sx,Sy,Pi,Pf,LResRegr);
With LResRegr do
            begin
            if DeslY<0 then
                  begin
                  DeslX:=0;
                  For i:=Pi to Pf do
                        if PtGraf[Sy,i]>0 then
                              begin
                              BGErro('Pontos em "Y" Invalidos p/ Regressao Exponencial',False
                              TipoRegr:=0;
                              FazRegressao:= False;
                              Exit;
                              end;
                  end;
            SSx:= Sx;
            SSy:= Sy;
            n:= Pf-Pi+1;
            j:=0 ; k:=0 ; l:=0; m:=0;
            r2:=0;
            for i:= Pi to Pf do
                  begin
                  x:= PtGraf[Sx,i];
```

```
                    y:= Ln(Abs(PtGraf[Sy,i]));
                    j:= j+ x;
                    k:= k+ y;
                    l:= l+ x*x;
                    m:= m+ y*y;
                    r2:= r2+ x*y;
                    end;
                b:= ( n * r2 - k * j ) / ( n * l - j*j);
                a:= Exp ( ( k - b * j ) / n );
                j:= b * ( r2 - j * k / n );
                m:= m - k * k / n;
                k:= Abs(m - j);
                r2:= Abs(j / m);
                r:= Sqrt(r2);
                Desv:= Sqrt( k / (n - 2));
                end;
end;

(*************************************************************************
+*+
* GSRegrPol - Calcula regressao polinomial de um conjunto de pontos (x,y)
*
*           Esta rotina faz a regressao polinomial pelo metodo dos minimos
* quadrados. A partir da tabela de dados e' calculado um polinomio de grau N,
* (no maximo 10).
*
* CHAMADA PADRAO:
*
*           GSRegrExp( Sx,Sy,Pi,Pf,LResRegr);
*
*           Nome      i/o   tipo          Descricao
*           Sx        i     inteiro       Variavel do vetor PtGraf para eixo X
*           Sy        i     inteiro       Variavel do Vetor PtGraf para eixo Y
*           Pi        i     inteiro       Ponto Inicial do vetor
*           Pf        i     inteiro       Ponto final do vetor
*           LResRegr  i/o   ParmRegrPol   Resultados da regressao
*
* SAIDA DE EXCECAO:
*
*           Se a solucao da regressao nao e' unica, e' dada uma mensagem ao usuario.
*
* ROTINAS CHAMADAS:
*
*           BGErro -   Emite mensagem de erro na tela.
*
-*-
* NOTAS:
*
*************************************************************************)


Procedure GSRegrPol(Sx,Sy,Pi,Pf:Integer; Var LResRegr: ParmRegrPol);

Var
            i,j,k,N:Integer;
            x,y: Real;
            Achou: Boolean;
            Aux : Real;
            R: TVetR;
```

```
            T: TVetT;
            P,Q,Z: Real;
begin
With LResRegr do
            begin
            SSx:= Sx;
            SSy:= Sy;
            N:= Pf - Pi + 1;              { Numero de Pontos para a regressao    }
            P:=0;                         { Inicializa vetores                   }
            Q:=0;
            Z:=0;
            for i:=1 to Grau+2 do
                  begin
                  For j:=1 to Grau+1 do
                        R[j,i]:= 0;
                  T[i]:=0;
                  end;
            for i:=1 to 2*Grau+1 do
                  A[i]:=0;
            A[1]:= n;
            for i:=Pi to Pf do
                  begin
                  x:= PtGraf[Sx, i];                  { Pega coordenadas da tabela }
                  y:= PtGraf[Sy, i];
                  for j:=2 to 2*Grau + 1 do
                        A[j]:= A[j] + ElevInt( X, j-1);     { Monta vetores para a regressao }
                  for k:=1 to Grau+1 do
                        begin
                        R[k, Grau+2]:= T[k] + y* ElevInt(X, K-1);
                        T[k] := T[k] + y * ElevInt(X, K-1);
                        end;
                  T[Grau+2]:= T[Grau+2] + ElevInt(y,2);
                  end;

            For j:=1 to Grau+1 do                     { e a matriz ....      }
                  For k:=1 to Grau+1 do
                        R[j,k] := A[j+k-1];
            For j:=1 to Grau+1 do
                  begin
                  Achou:= False;
                  For k:=J to Grau+1 do
                        if Abs(R[k,j]) > QZero then Achou:= True;
                  if Not Achou then
                        begin
                        BGErro('Solucao da Regressao nao e'' Unica',False);
                        TipoRegr:=0;
                        FazRegressao:=False;
                        Exit;
                        end;
                  For i:=1 to Grau+2 do
                        begin
                        Aux:= R[j,i];
                        R[j,i]:= R[k,i];
                        R[k,i]:= Aux;
                        end;

                  Z:= 1 / R[j,j];
                  For i:=1 to Grau+2 do
                        R[j,i]:= Z * R[j,i];
```

```
                For k:=1 to Grau+1 do
                    if k<>j then
                    begin
                    Z:= - R[k,j];
                    For i:=1 to Grau+2 do
                        R[k,i]:= R[k,i] + Z * R[j,i];
                    end;
                end;
            B:= R[1,Grau+2];
            P:=0;
            For j:=2 to Grau+1 do
                P:= P + R[j,Grau+2] * (T[j] - A[j] * T[1] / N);
            Q:= T[Grau+2] - ElevInt(T[1], 2) / N;

            For j:=1 to Grau do
                A[j]:= R[j+1,Grau+2];

            Z:= Q- P;
            I:= N - Grau -1;
            R2:= Abs(P / Q);
            R1:= Sqrt(R2);
            Desv:= Sqrt(Abs(Z / I));
            end;
end;
```

b.2) Cálculo das Estatísticas Básicas

```
(*************************************************************************
+*+
* GSEstatBas - Faz o calculo das estatisticas basicas
*
* CHAMADA PADRAO:
*
*           GSEstatBas(NVar);
*
*           Nome      i/o   tipo      Descricao
*           NVar      i     inteiro   Posica da variavel no vetor PtGraf
*
* ROTINAS CHAMADAS:
*
* nome      descricao
* GSSort    Ordena  o vetor de valores
-*-
* NOTAS:
*
*************************************************************************)

Procedure GSEstatBas( NVar: Integer);

Var
            i,k: Integer;
            QSoma, Soma: Real;
            VetAux: PVetSinal;
            NPtos:  Integer;
            VetModa: Array[0..100] of Integer;
            AuxModa:  Real;
```

```
begin
GSModa:=0;
With StZoom[NivZoom] do
                begin
                Soma:=0;
                NPtos:= PFim-PInic+1;
                New(VetAux);
                For i:= PInic to PFim do
                      begin
                      VetAux^[i]:= PtGraf[NVar,i];
                      Soma:= Soma+ PtGraf[NVar,i];     { Somatoria                  }
                      end;
                GSMedia:= Soma / (PFim-PInic+1);       { Calculo da Media           }

                GSSort(PInic,PFim,VetAux);             { Calculo da Mediana         }
                i:= PInic+(NPtos div 2);
                if (NPtos mod 2) = 1 then
                      GSMediana:= VetAux^[i]           { Numero de Pontos e' Impar }
                else
                      GSMediana:= (VetAux^[i-1]+VetAux^[i])/2; { Numero de Pontos e' Par   }

                                                       { Calcular Moda aproximada }
                                                       { Pego intervalos de 1% da   }
                                                       { amplitude total            }
                FillChar(VetModa, Sizeof(VetModa), #0);
                For i:= PInic to PFim do
                      begin
                      AuxModa:= (VetAux^[i]-VetAux^[PInic])/(VetAux^[PFim]-VetAux^[PInic])*1C
                      Inc(VetModa[Trunc(AuxModa)]);
                      end;
                AuxModa:= VetModa[0];
                k:= 0;
                For i:= 1 to 100 do
                      if VetModa[i]>AuxModa then
                            begin
                            k:= i;
                            AuxModa:= VetModa[i];
                            end;
                GSModa:= (VetAux^[PFim]-VetAux^[PInic])*k/100 + VetAux^[PInic];
                i:= PInic;
                AuxModa:= GSModa-VetAux^[PInic];
                While (i<=PFim) do
                      begin
                      if Abs(GSModa-VetAux^[i]) < Abs(AuxModa) then
                            AuxModa:= GSModa-VetAux^[i];
                      Inc(i);
                      end;
                GSModa:= GSModa-AuxModa;

                Soma:= 0;
                QSoma:=0;
                for i:= PInic to PFim do
                      begin
                                                       { P/ calc. Desvio Medio  }
                      Soma:= Soma+ Abs(PtGraf[NVar,i]-GSMedia);
                                                       { P/ calc. Desvio Padrao }
                      QSoma:= QSoma+ElevInt(PtGraf[NVar,i]-GSMedia,2);
                      end;
                GSDesvMed  := Soma/NPTos;
```

```
            GSVarianca := QSoma/NPtos;
            GSDesvPadr := Sqrt(GSVarianca);
            GSCoefVar  := GSDesvPadr / GSMedia;

            Dispose(VetAux);
            end;
end;
```

b.3) Traçado de Curvas Interpoladas (B-Splines)

```
(*************************************************************************
+*+
* Spline - Interpola um conjunto de pontos (x,y) pelo metodo das B-Splines
*
* Esta rotina calcula a funcao B-SPline que interpola o conjunto de pontos
* (x,y) recebido como entrada. Em seguida ela faz o calculo de um conjunto
* de pontos (x,y) definidos pela funcao B-Spline para o mesmo intervalo de
*  entrada.
*
* CHAMADA PADRAO:
*
*           Res:= Spline( A1, A2, N, Pi, Pf, B1, B2, M);
*
*           Nome     i/o   tipo      Descricao
*           A1       i/o   Real      Valores de entrada p/ o eixo X
*           A2       i/o   Real      Valores de entrada p/ o eixo Y
*           N        i     Inteiro   Numero de pontos de entrada.
*           Pi       i     Inteiro   Posic. do ponto inicial no vetor PtGraf
*           Pf       i     Inteiro   Posic. do ponto final no vetor PtGraf
*           B1       i/o   Real      Valores de saida para o eixo X
*           B2       i/o   Real      Valores de saida para o eixo Y
*           M        i     Inteiro   Numero de pontos de saida.
*
* RETORNA:
*
*           TRUE, se o calculo foi bem sucedido.
*           FALSE, caso contrario.
*
* SAIDA DE EXCECAO:
*
* Se numero de pontos para a interpolacao for menor ou igual a dois ou se
* algum ponto calculado cair fora do intervalo de entrada, e' emitida a
* mensagem de erro correspondente.
*
-*-
* NOTAS:
*
*************************************************************************)

Function Spline(    var A1,A2: VetSinal; N: integer; Pi,Pf: Integer;
                    var B1,B2: VetSinal; M : integer): Boolean;

type
            Vector = VetSinal;
```

```
var
        I, K    : integer;
        Dx, T   : Real;
        B, C, D : Vector;
        X1,Xm   : Real;

        (*-------------------------------------------------------------------------*)
        function SplineEval( T : Real; var I : integer) : Real;
        var
            J, K : integer;
            Dx: Real;
        begin
        if I >= N then
            I := 0;
        if (T < A1[Pi+I]) or (T > A1[Pi+I+1]) then
            begin
            I := 0;
            J := N + 1;
            repeat
                K := (I + J) div 2;
                if T < A1[Pi+K] then
                    J := K;
                if T >= A1[Pi+K] then
                    I := K;
            until J <= (I + 1);
            end;
        Dx := T - A1[Pi+I];
        SplineEval := A2[Pi+I] + Dx * (B[I] + Dx * (C[I] + Dx * D[I]));
        end; { SplineEval }

(*-------------------------------------------------------------------------*)
begin { Spline }
X1:= A1[Pi];
Xm:= A1[Pf];
if N >= 3 then
        begin
        D[0] := A1[Pi+1] - A1[Pi+0];
        C[1] := (A2[Pi+1] - A2[Pi+0]) / D[0];
        for I := 1 to N-1 do
            begin
            D[I] := A1[Pi+I+1] - A1[Pi+I];
            B[I] := 2.0 * (D[I-1] + D[I]);
            C[I+1] := (A2[Pi+I+1] - A2[Pi+I]) / D[I];
            C[I] := C[I+1] - C[I];
            end;
        B[0] := -D[0];
        B[N] := -D[N-1];
        C[0] := 0.0;
        C[N] := 0.0;
        if N > 3 then
            begin
            C[0] := C[2] / (A1[Pi+3] - A1[Pi+1]) - C[1] / (A1[Pi+2] - A1[Pi+0]);
            C[N] := C[N-1]/(A1[Pi+N]-A1[Pi+N-2])-C[N-2]/(A1[Pi+N-1] - A1[Pi+N-3]);
            C[0] := C[0] * Sqr(D[0]) / (A1[Pi+3] - A1[Pi+0]);
            C[N] := -C[N] * Sqr(D[N-1]) / (A1[Pi+N] - A1[Pi+N-3]);
            end;
        for I := 1 to N do
            begin
            T := D[I-1] / B[I-1];
```

```
                B[I] := B[I] - T * D[I-1];
                C[I] := C[I] - T * C[I-1];
                end;
            C[N] := C[N] / B[N];
            for I := N-1 downto 0 do
                C[I] := (C[I] - D[I] * C[I+1]) / B[I];
            B[N] := (A2[Pi+N]-A2[Pi+N-1])/D[N-1]+D[N-1] * (C[N-1] + 2.0 * C[N]);
            for I := 0 to N-1 do
                begin
                B[I] := (A2[Pi+I+1]-A2[Pi+I])/D[I]-D[I]*(C[I+1] + 2.0 * C[I]);
                D[I] := (C[I+1] - C[I]) / D[I];
                C[I] := 3.0 * C[I];
                end;
            C[N] := 3.0 * C[N];
            D[N] := D[N-1];
            end
else
            if N = 2 then
                begin
                B[0] := (A2[Pi+1] - A2[Pi+0]) / (A1[Pi+1] - A1[Pi+0]);
                C[0] := 0.0;
                D[0] := 0.0;
                B[1] := B[0];
                C[1] := 0.0;
                D[1] := 0.0;
                end;
if (N >= 2) and (M >= 2) then
            if (X1 >= A1[Pi+0])  and (Xm <= A1[Pi+N]) then
                begin
                Dx := (Xm - X1) / M;
                K := 0;
                for I := 0 to M do
                    begin
                    B1[I] := X1 + I * Dx;
                    B2[I] := SplineEval(B1[I], K);
                    end;
                Spline:= True;
                end
            else
                begin
                BGErro('Erro no Calculo da Interpolacao - Valores de Contorno',False);
                Spline:= False;
                end
else
            begin
            BGErro('Erro no Calculo da Interpolacao - Numero de Pontos',False);
            Spline:= False;
            end;
end; { Spline }
```

c) Rotinas do STEF

c.1) Interpretação e Avaliação de Condições de Parada

```
(*************************************************************************
+*+
* TDAvaliaCond - Avalia se uma condicao de parada foi ou nao satisfeita
*
* CHAMADA PADRAO:
*
*            ValRet := TDAvaliaCond(NCond: Integer);
*
*            Nome      i/o   tipo       Descricao
*            NCond     i     inteiro    Numero da posicao do vetor de condicao
*                                       de parada a ser avaliado.
*
* RETORNA:
*            TRUE se a condicao de parada foi atingida.
*            FALSE, caso contrario.
*
* ROTINAS CHAMADAS:
*
*  TDCalcula_no - Calcula uma sub-expressao da condicao de parada.
*
-*-
* NOTAS:
*
*************************************************************************)
Function TDAvaliaCond(NCond: Integer): Boolean;

            (*************************************************************
            +*+
            * TDCalcula_No -     Avalia uma sub-arvore da expressao que define a
                                 condicao de parada.
            *
            * CHAMADA PADRAO:
            *
            *     ValRet := TDCalcula_No(PCond: Integer):
            *
            *     Nome      i/o   tipo       Descricao
            *     PCond     i     RegOpPtr   Ponteiro para um determinado no da
            *                                da arvore que define a expressao para
            *                                a condicao de parada dos testes.
            *
            * RETORNA:
            *     TRUE se a condicao de parada da sub-arvore foi atingida.
            *     FALSE, caso contrario.
            *
            * ROTINAS CHAMADAS:
            *
            *     TDCalcula_No -
            -*-
            * NOTAS:
            *
```

```
            ************************************************************)

            Function TDCalcula_No(P: RegOpPtr): Boolean;

            Var
                  Expr1,Expr2: Boolean;
            begin

            With P^ do
                  begin
                  if Operando1 > 0 then              { Operando1 em nivel logico alto }
                        Expr1:= (Sinal[Operando1]= 1)
                  else
                        Expr1:= TDCalcula_No(PEsq);
                  if Not (Operador in Un_Oper) then
                        if Operando2 > 0 then        { Operando2 em nivel logico alto }
                              Expr2:= (Sinal[Operando2] = 1)
                        else
                              Expr2:= TDCalcula_No(PDir);

                  Case Operador of
                        '!':  TDCalcula_No:= not Expr1;
                        '+':  TDCalcula_no:= Expr1 or Expr2;
                        '.':  TDCalcula_no:= Expr1 and Expr2;
                        '=':  TDCalcula_no:= Expr1 =  Expr2;
                        '#':  TDCalcula_no:= Expr1 <> Expr2;
                        end; { of case }
                  end;
            end;

begin  { TDAvaliaCond }
TDAvaliaCond:= TDCalcula_No(RgEst[NCond].PCond);
end;  { of TDAvaliaCond }


(*************************************************************************
+*+
* TDParseCond -    Interpreta condicao de parada, criando a arvore que armazena
*                  a expressao da condicao de parada.
*
* CHAMADA PADRAO:
*
*             TDParseCond(Linha, Operac, Operando);
*
*             Nome      i/o   tipo       Descricao
*             Linha     i     String     Linha c/ texto da expressao a interpretar
*             Operac    i     RegOpPtr   Ponteiro para um no da expressao
*             Operando  i/o   Integer    Numero do pino do soquete de testes
*                                        definido como operador.
*
* SAIDA DE EXCECAO:
*
*             Se erro no formato da expressao, o sistema apresenta mensagem de erro.
*
* ROTINAS CHAMADAS:
*
* nome              descricao
* Atomico           Verifica se uma sub-expressao é atomica ou n<176>o
* BGErro            Mostra mensagem de erro na tela.
```

```
* Novo_no          Cria um novo nó para a árvore da condicao de parada.
* PegaOperador     Encontra o primeiro operador da linha de comando.
* TDParseCond      ... esta rotina ...
*
-*-
* NOTAS:
*
*************************************************************************)

Procedure TDParseCond(Linha:String; Operac:RegOpPtr; Var Operando:Integer);

var
            Cod: Integer;
            St:  String;
            Pos1, NivParent: Integer;
            Val1, CodErr: Integer;
            Operac2: RegOpPtr;
            Conj: Byte;
begin
if Operando<0 then                    { Se Ocorreu algum erro, vai voltando }
            Exit
else
            Operando:= 0;             { Inicializa Operando }

Conj:=0;
repeat
            Conj:= Succ(Conj);
            Case Conj of
                  1: Operadores:= Rel_Oper;
                  2: Operadores:= Bool_Oper;
                  3: Operadores:= Un_Oper;
                  End; { of case }

            Pos1:= PegaOperador(Linha);
            if Pos1>0 then              { Achou Operador deste tipo      }
                  begin
                  Operac^.Operador:= Linha[Pos1];
                  if Pos1>1 then        { Analisa parte esquerda da expr. }
                        begin
                        Novo_No(Operac2);
                        Operac^.PEsq:= Operac2;
                        TDParseCond(Copy(Linha,1,Pos1-1), Operac2, Operac^.Operando1);
                        if Operac^.Operando1>0 then
                              Operac^.PEsq:= Nil;
                        if Pos1<Length(Linha) then  { Analisa parte direita da linha  }
                              begin
                              Novo_No(Operac2);
                              Operac^.PDir:= Operac2;
                              TDParseCond(Copy(Linha,Pos1+1,Length(Linha)),
                                                Operac2,Operac^.Operando2);
                              if Operac^.Operando2>0 then
                                    Operac^.PDir:= Nil;
                              end;
                        Exit;
                        end
                  else                  { e' operador unario ou e' um erro }
                        begin
                        Operac^.Operador:= Linha[Pos1];
                        if Operac^.Operador in Un_Oper then    { Operador Unario }
```

```
                        begin
                        TDParseCond(Copy(Linha,2,Length(Linha)), Operac,
                                                    Operac^.Operando1);
                        if Operac^.Operando1 > 0 then
                              Operac^.PEsq:= Nil;
                        Exit;
                        end
                  else
                        BGErro('Erro no Formato da Condicao de Inicializacao', False);
                  end;
            end
      else        { Nenhum operador deste conjunto ==> e' atomo ou tem parenteses }
            begin
            if (Linha[1]='(')and(Linha[Length(Linha)]=')')and(Operadores=Un_Oper) then
                  begin
                  Delete(Linha, 1, 1);
                  Delete(Linha, Length(Linha), 1 );
                  TDParseCond(Linha, Operac, Operando);   { Refaz parsing da linha }
                  Exit;
                  end
            else
                  begin
                  if Atomico(Linha) then                  { e' um operando atomico }
                        begin
                        Val(Linha, Operando, Cod);
                        Exit;
                        end;
                  end;
            end;

Until Conj>=3;                { Ate' esgotar todos os conjuntos de possiveis operadores }

Operando:= -1;                { Se chegou aqui e' porque houve algum erro na expressao }
BGErro('Erro no Formato da Condicao de Parada!', False);
end; { of TDParseCond }
```

c.2) Acesso à Interface de Testes Funcionais

```
(*************************************************************************
+*+
* TDPreparaEstimulos - Prepara para a aplicacao de Estimulos ao Circuito
*
* CHAMADA PADRAO:
*
*           TDPreparaEstimulos(NCond);
*
*           Nome      i/o   tipo       Descricao
*           NCond     i     inteiro    Identifica se estimulos de inicializacao
*                                      ou se sao os vetores de teste.
*                                      (0= Inicializacao; 1= Vetor de Testes)
*
-*-
* NOTAS:
*           Esta rotina deve sofrer modificacoes de acordo com o tipo de
*           interface de testes funcionais utilizada. Na versao atual está
*           disponivel somente um tipo de interface.
```

```
*************************************************************************)
Procedure TDPreparaEstimulos(NCond: Integer);

var
        i: byte;

        {*************************************************************}
        Procedure SETA_PINO(npino,modo:byte);

        var
            numbyte, numbit: byte;
        begin
        npino:= pred(npino);
        numbyte:= npino div 8;
        numbit:=  npino mod 8;
        SET_BIT(S_Contr[numbyte],numbit,modo);
        end;

begin { TDPreparaEstimulos }
Case CFTipoInterface of
        QUATORZE:
            begin
            for i:=1 to MAXEST do
                with RgEst[NCond].Estim[i] do
                    if (pino>0) and (estado_atual<>INDEFINIDO) then
                        SETA_PINO(pino,1);
            port[Porta[3]]:= S_Contr[0];
            port[Porta[4]]:= S_Contr[1];
            port[Porta[5]]:= S_Contr[2];
            end;
        else { case }
            (* Ainda nao tem outro tipo de interface *) ;
        end; { of case }
end;

(*************************************************************************
+*+
* TDMandaEstimulos - Aplica um conjunto de estimulos aos pinos da interface.
*
* CHAMADA PADRAO:
*
*       TDMandaEstimulos(NCond);
*
*       Nome    i/o   tipo      Descricao
*       NCond   i     Inteiro   Identifica se estimulos de inicializacao
*                               ou se sao os vetores de teste.
*                               (0= Inicializacao; 1= Vetor de Testes)
*
-*-
* NOTAS:
*
*************************************************************************)
Procedure TDMandaEstimulos(NCond: Integer);

var
        i:byte;

        {*************************************************************}
```

```
            Procedure SETA_SINAL( npino, modo: byte);

            var
                  numbyte, numbit: byte;
            begin
            if (NPino <=0) or (NPino>CFNumPinos) then
                  exit;
            NPino := pred(npino);
            numbyte:= npino div 8;
            numbit:=  npino mod 8;
            SET_BIT(S_Estim[numbyte],numbit,modo);
            end;

begin { TDMandaEstimulos }
Case CFTipoInterface of
            QUATORZE:
                  begin
                  for i:=1 to MAXEST do
                        with RgEst[NCond].Estim[i] do
                              SETA_SINAL(pino,estado_atual);
                  port[Porta[0]]:= S_Estim[0];
                  port[Porta[1]]:= S_Estim[1];
                  port[Porta[2]]:= S_Estim[2];
                  end;
            else
                  (* Nao faz nada. Nao tem outro tipo de interface *) ;
            end;
end;


(*************************************************************************
+*+
* TDLeSinais - Le os estados logicos dos sinais nos pinos da interface.
*
* CHAMADA PADRAO:
*
*           TDLeSinais(VSinal);
*
*           Nome      i/o   tipo        Descricao
*           VSinal    i/o   VetSinais   Vetor contendo os estados logicos de
*                                       cada pino do soquete de testes.
-*-
* NOTAS:
*
*************************************************************************)
Procedure TDLeSinais(Var VSinal: VetSinais);
var
            i:byte;
begin
Case CFTipoInterface of
            QUATORZE:
                  begin
                  port[Porta[6]]:= 0;
                  S_Entr[0]:= port[Porta[0]];
                  S_Entr[1]:= port[Porta[1]];
                  Vsinal[1] := S_Entr[0] and $01;
                  Vsinal[2] := (S_Entr[0] and $02) shr 1;
                  Vsinal[3] := (S_Entr[0] and $04) shr 2;
                  Vsinal[4] := (S_Entr[0] and $08) shr 3;
```

```
                    Vsinal[9] := (S_Entr[0] and $10) shr 4;
                    Vsinal[10]:= (S_Entr[0] and $20) shr 5;
                    Vsinal[11]:= (S_Entr[0] and $40) shr 6;
                    Vsinal[12]:= (S_Entr[0] and $80) shr 7;
                    Vsinal[17]:= (S_Entr[1] and $01) ;
                    Vsinal[18]:= (S_Entr[1] and $02) shr 1;
                    Vsinal[19]:= (S_Entr[1] and $04) shr 2;
                    Vsinal[20]:= (S_Entr[1] and $08) shr 3;

                    port[Porta[6]]:= 1;
                    S_Entr[2]:= port[Porta[0]];
                    S_Entr[3]:= port[Porta[1]];
                    Vsinal[5] := S_Entr[2] and $01;
                    Vsinal[6] := (S_Entr[2] and $02) Shr 1;
                    Vsinal[7] := (S_Entr[2] and $04) Shr 2;
                    Vsinal[8] := (S_Entr[2] and $08) Shr 3;
                    Vsinal[13]:= (S_Entr[2] and $10) Shr 4;
                    Vsinal[14]:= (S_Entr[2] and $20) Shr 5;
                    Vsinal[15]:= (S_Entr[2] and $40) Shr 6;
                    Vsinal[16]:= (S_Entr[2] and $80) Shr 7;
                    Vsinal[21]:= (S_Entr[3] and $01) ;
                    Vsinal[22]:= (S_Entr[3] and $02) Shr 1;
                    Vsinal[23]:= (S_Entr[3] and $04) Shr 2;
                    Vsinal[24]:= (S_Entr[3] and $08) Shr 3;
                    end;
               else
                    (* Nao faz nada porque nao existe outro tipo de interface *) ;
               end; { of Case }
end;

(*****************************************************************************
+*+
* TDTestaCircuito - Controla a execucao dos testes do circuito.
*
* CHAMADA PADRAO:
*
*            TDTestaCircuito;
*
* OUTRAS ENTRADAS:        (Dados Globais utilizados)
*
* GravaResult-      Indica se resultados devem ou nao ser gravados em disco.
* NomeProj -        Nome do projeto.
* T_Inic -          Instante inicial da aplicacao dos estimulos.
* TempoReal -       Indica se resultados devem ou nao ser mostrados em tempo real.
*
* OUTRAS SAIDAS:          (Dados globais Alterados)
*
* Anter -           Contem os estados dos sinais no instante anterior ao atual.
* ArqSai-           Arquivo contendo os resultados dos testes.
* Instante -        Instante Atual
* Sinal -           Contem os estados logicos de cada pino no instante atual.
* VetSai-           Contem todos os dados de resultados de teste.
*
* ROTINAS CHAMADAS:
*
* nome                  descricao
* BGCabecTela           Mostra o cabecalho da tela.
* BGErro                Mostra mensagem de erro no rodape da tela.
* TDAvaliaCond          Avalia uma expressao de condicao de parada.
```

```
* TDInicTelaSaida       Faz a inicializacao da tela para saida dos resultados.
* TDLeSinais            Le estado logico de cada um dos pinos do soquete de teste.
* TDMandaEstimulos      Faz a aplicacao de um conjunto de estimulos ao circuito.
* TDMostraSinais        Mostra os estados logicos dos sinais no instante atual
* TDMostraTempo         Mostra janela na tela com o instante atual.
* TDPreparaEstimulos    Prepara a interface para aplicacao de estimulos ao ccto.
* TDVeQuemMuda          Verifca quais estimulos que mudam no instante atual
*
-*-
* NOTAS:
*
*****************************************************************************)

Procedure TDtestacircuito;

Var
            Parar : Boolean;
            k     : LongInt;
            CondParada: Boolean;
            PosVet: LongInt;
            ArqSai: File of Byte;
            NomArqSai: Str20;
            OffSetX, OffSetY: Integer;

begin
Instante:= T_Inic;
ClearDevice;                                    { Limpa a tela para mostrar result. }
BGCabecTela(MensCab);

if GravaResult then                             { Se vai gravar resultado em disco }
            begin                               { cria o arquivo de saida          }
            NomArqSai:= NomeProj+'.STF';
            Assign(ArqSai, NomArqSai);
            {$I-}
            Rewrite(ArqSai);
            {$I+}
            if IOResult<>0 then                 { Se erro na criacao do arquivo,   }
                  begin                         { desabilita a saida em disco      }
                  BGErro('Erro na Criacao do Arquivo! ', False);
                  GravaResult:= False;
                  end
            else
                  begin
                  Write(ArqSai, T_Inic);        { Gravar Instante inicial e increm. }
                  Write(ArqSai, Increm);        { no cabecalho do arquivo.          }
                  end;
            end;

OffSetX:= XMinJan;
OffSetY:= (YMaxJan-YMinJan) Div 2 + YMinJan;
if TempoReal then
            TDInicTelaSaida(OffSetX,OffSetY);

For k:=1 to MAXPINOS do
            begin
            FillChar( VetSai[k]^, SizeOf(VetSai[k]^), #0);
            anter[k]:=0;
            sinal[k]:=0;
            end;
```

```
Parar:= (Instante > T_Final);
CondParada:= False;

While Not Parar do
              begin
              if (Instante Mod 10) = 0 then
                    TDMostraTempo(Instante);
              TDVeQuemMuda(0);              { Ve estimulos que mudam neste inst.   }
              TDPreparaEstimulos(0);        { Prepara para mandar os estimulos     }
              TDMandaEstimulos(0);          { Aplica os estimulos ao circuito      }
              if ((instante-t_inic) mod increm) = 0 then
                    begin
                    if TempoReal then
                          begin
                          For K:=1 to MAXPINOS do
                                Anter[k]:= Sinal[k];
                          TDLeSinais(Sinal);
                          TDmostraSinais(OffSetX,OffSetY,instante);
                          end
                    else
                          TDLeSinais(Sinal);
                    PosVet:= (Instante - T_Inic)  div Increm div 8;
                    For k:=1 to MAXPINOS do
                          VetSai[k]^[Posvet]:= (VetSai[k]^[Posvet] shl 1) + Sinal[k];

                    if GravaResult and (((((Instante- T_Inic) Mod Increm)+1) Mod 8)=0) then
                          Write(ArqSai, VetSai)*);
                    end;
              Inc(instante);
              CondParada:= TDAvaliaCond(0);
              Parar:= (Instante > T_Final) or CondParada;
              end;

TDMostraTempo(Instante);
if Not TempoReal then
              begin
              TDInicTelaSaida(OffSetX, OffSetY);
              For k:= T_Inic to Instante do
                    TDMostraSinais(OffSetX, OffSetY, k);
              end;

if GravaResult then
              Close(ArqSai);

end ; { of TDtestacircuito}
```

Anexo VI - Diagrama Esquemático da Interface para Testes Funcionais