Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia Mecânica
Área de Automação e Sistemas

Desenvolvimento de Software para Aquisição de Dados e Controle (liga/desliga) de Dispositivos

Claudio Tokeiama

São Paulo
1997

Claudio Tokeiama

Desenvolvimento de Software para Aquisição de Dados e Controle (liga/desliga) de Dispositivos

Trabalho de Formatura apresentado à
Escola Politécnica da Universidade de São Paulo

Orientador:
Prof. Dr. José Roberto Simões Moreira

São Paulo
1997

AGRADECIMENTOS

Ao Prof. Dr. José Roberto Simões Moreira, pelo incentivo e coordenação ao longo do trabalho, tendo em vista as dificuldades que apareceram, como o problema ocorrido com a placa. E principalmente pela oportunidade de poder participar de alguns de seus projetos ao longo de pouco mais de 2 anos, como estagiário e depois como trabalho de formatura.

Aos professores de PMC 497 - Laboratório de CAD/CAE, que autorizaram a utilização de uma parte de um programa (Unit Edittool), sendo que esta facilitou muito o nosso trabalho.

Ao Depto de Eng. Mecânica, por ceder os recursos necessários para a realização deste trabalho.

E à minha família e amigos pelo apoio ao longo de todos esses anos.

RESUMO

Este trabalho objetiva a implementação de um programa computacional, que utilizado com uma placa conversora A/D e D/A possa fazer a aquisição de dados e controle de alguns dispositivos. Será estudado e construído também dois dispositivos de interfaceamento de potência entre o sinal de comando da placa e seus periféricos.

Sumário

1. Introdução

Os sistemas de supervisão e controle são muito importantes e são largamente utilizados tanto na indústria quanto no meio acadêmico. Dentre as inúmeras possibilidades de utilização, eles são mais utilizados para:

- coletar e armazenar sinais provenientes de sensores para medição de temperatura, pressão, toxinas, nível de acides ou basicidade (PH), tensões, correntes etc.;
- monitorar esses sinais para tomada de decisões durante o processo, para que seja inicializado um próximo passo, ligando e desligando dispositivos em uma determinada condição e muitas outras possibilidades.

Os sistemas utilizados na indústria geralmente são dedicados e, para facilidade e redução de preço, possuem limitações quanto à possibilidade de se controlar um número grande de dispositivos, além de serem fixados e apropriados para certa aplicação, como é o caso de sistemas de injeção eletrônica, sistemas de alarmes etc. Sistemas dedicados, e que requerem um grande número de características são bastante caros, como o caso de sistemas de monitoramento de plataformas petrolíferas e indústrias químicas. Nesses casos, é necessário o monitoramento de um número muito grande de sinais (principalmente pressão, temperatura, posição de válvulas e/ou portas, entre outras). Adicionalmente, é necessário o controle de outros dispositivos, como válvulas e portas, alarmes de emergência, sistemas de segurança etc. Esses sistemas são extremamente complexos, além de ter uma grau de confiabilidade muito alto. Nesses casos os programas computacionais são

para suas aplicações e não possibilitam sua adaptação para outros tipos de sistemas.

No meio acadêmico, a utilização de uma placa como a CAD 12/36 proporciona uma maior versatilidade, pois além de possuir todas as características para realizar as atividades acima citadas, possui uma série de outros recursos, além de ser quase que totalmente configurável, podendo ser adaptada para muitos tipos de experimentos. Isso será detalhado em "*3. Características Gerais da Placa CAD 12/36*". Adicionalmente a implementação de um programa computacional é muito fácil e pode ser feita por qualquer pessoa com conhecimento de linguagem de programação e noções do sistema no qual será aplicado.

2. Objetivos

O objetivo deste trabalho é o desenvolvimento de um programa computacional (em linguagem "Pascal") para aquisição de dados e controle (do tipo liga - desliga) de aparelhos. Esses aparelhos são um flash e um solenóide. Além disso serão estudados e desenvolvidos dispositivos para interfaceamento de potência do sinal proveniente da placa de controle no microcomputador, com o solenóide. Esse dispositivo será utilizado no Laboratório SISEA do Departamento de Engenharia Mecânica da Escola Politécnica da USP, num trabalho de pós-graduação. Será citado esse trabalho para localização do sistema que será desenvolvido, no item **3. Breve Descrição do Laboratório SISEA**.

O trabalho foi dividido em duas partes, especificamente para uma primeira apresentação (no final de maio de 1997), que serve como uma análise do andamento do trabalho, e uma segunda apresentação (realizada em novembro de 1997), com a conclusão do trabalho. Este relatório contém o trabalho final, que já foi apresentado, englobando a primeira e a segunda parte.

3. Breve Descrição do Laboratório Sisea

Este trabalho serve de apoio para uma experiência de pós-graduação que está sendo realizada pelo Marcelo Mendes Vieira, sob a orientação do Prof. Dr. José Roberto Simões Moreira.

A seguir encontra-se uma foto do laboratório onde está sendo feito o estudo.

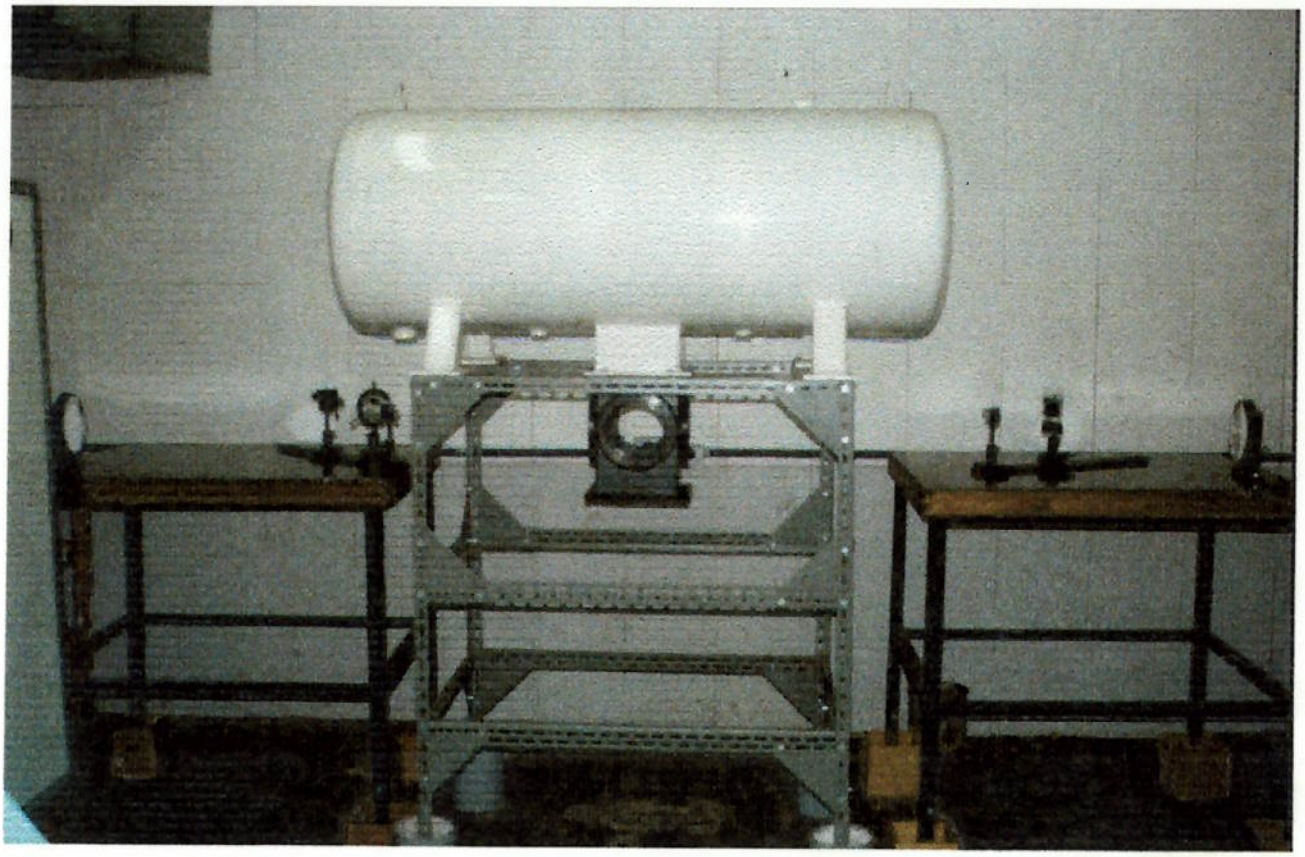


Figura 1 - Foto do Laboratório SISEA

Nesse trabalho, será necessária a coleta de dados de temperatura e pressão, bem como o acionamento de um flash e de um solenóide. O solenóide aciona uma válvula, tipo bico injetor, que libera dodecana a alta pressão, em uma câmara de vácuo. Este cria ondas de choque, que são o principal estudo do trabalho já mencionado. Depois de um certo tempo, será necessário tirar algumas fotos dessas ondas, e nesse ponto será acionado o flash. Uma câmera, com o diafragma previamente aberto será colocado de modo que quando o flash seja acionado, se obtenha a foto desejada. A seguir encontra-se esquematizada o posicionamento do "flash" (fonte de luz) e da câmera (filme).

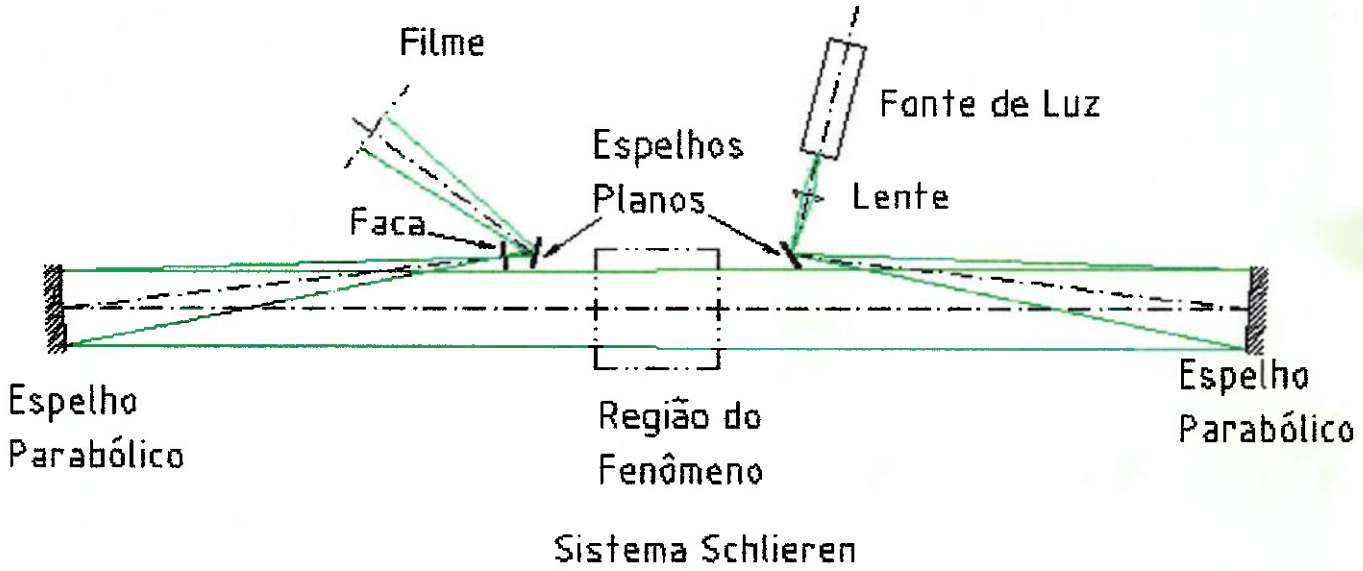


Figura 2 - Esquema de Posicionamento do "Flash" e da Câmera

3.1. Equipamentos utilizados

Nesse trabalho serão utilizados os seguintes equipamentos:

- Placa de aquisição de dados **"CAD12/36 Conversor A/D e D/A para microcomputadores"** da marca Lynx Tecnologia Eletrônica Ltda (maiores informações sobre as características da placa estão no item **4. Características da Placa CAD 12/36**;
- Módulo condicionador de sinais
- Microcomputador IBM-PC modelo 486 DX4, de 100 MHz;
- Além de equipamentos do prórpio laboratório, como geraodr de sinais, osciloscópio, cabos, etc.

4. Características da Placa CAD 12/36

É necessário que um sistema de aquisição de dados utilizado em um laboratório precisa seja flexível o suficiente para que possibilite um número quase que ilimitado de adaptações, e possua algumas facilidades que possibilite sua utilização sem a necessidade de ser utilizado outros dispositivos ou placas adicionais, o que poderia inviabilizar seu uso. Por isso a escolha de uma placa de aquisição de dados como a placa CAD 12/36, que possui uma série de recursos possibilitando uma grande número de aplicações. A seguir serão citados suas principais características:

- conversor A/D de 12 bits de resolução;
- possibilidade de, com o auxílio de circuitos externos, realizar amostragem simultânea de até 16 canais;
- suporte para Interrupções;
- base de tempo interna (2,00 MHz);
- 3 contadores / temporizadores de 16 bits;
- até 4 saídas analógicas ou outras expansões (a placa utilizada está disponível com 2 saídas analógicas, com o preenchimento de 1 expansão);
- 16 entradas analógicas simples ou 8 diferenciais multiplexadas;
- 16 entradas digitais;
- 16 saídas digitais;
- seqüência de leitura e ganhos programáveis através de memória de canais;
- aquisição em "Burst" propiciada por buffer (FIFO) de 16 posições;

- suporte para DMA (Direct Memory Access: Acesso Direto à Memória), permitindo a velocidade máxima de coleta de sinais independentemente da velocidade da UCP (Unidade Central de Processamento) do microcomputador;
- possibilidade se ser implementado um programa computacional com as principais linguagens de programação existentes no mercado, como "BASIC", "Pascal" e "C";

5. Trabalho Realizado

5.1. Programa Computacional "AQSinal"

O programa foi dividido em uma série de módulos para facilitar o entendimento , separar as rotinas por categorias e ajudar na localização de possíveis erros de programação. A interação entre esses módulos é mostrada na figura 3, e os módulos são serão explicados adiante.

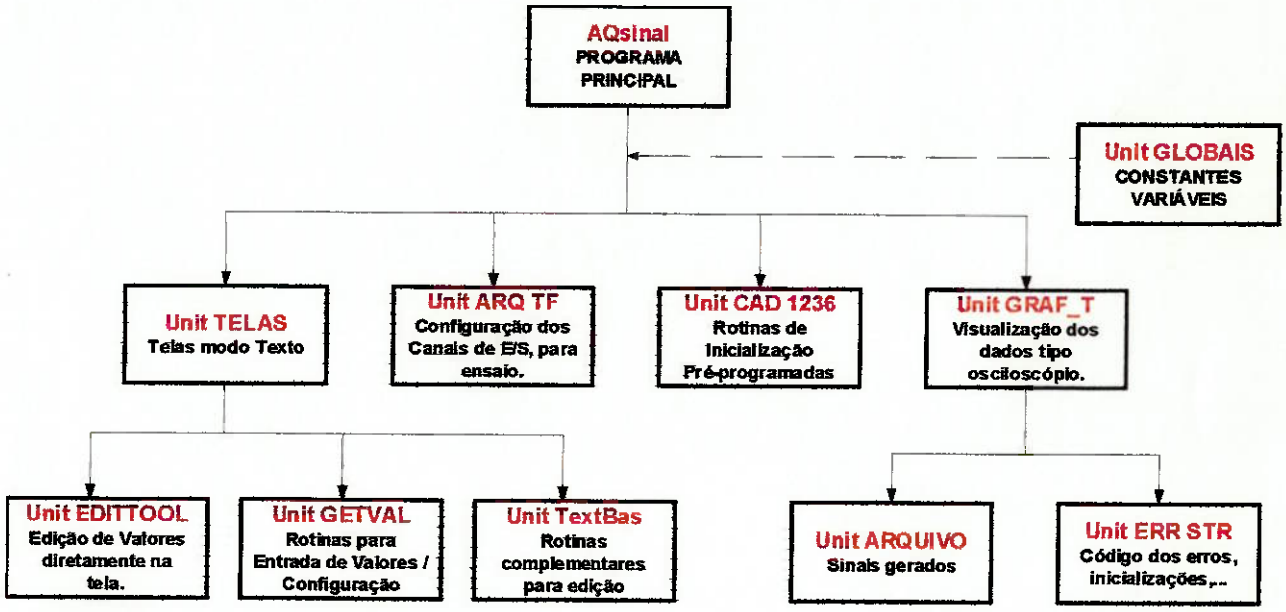


Figura 3 - Diagrama dos Módulos do Programa

Deve-se deixar claro que algumas rotinas não foram implementadas nesse trabalho. Entre elas temos: a Unit EditTool, a Unit TextBas e algumas rotinas da Unit CAD1236.

A seguir são explicados os módulos implementados.

5.1.1. Program AQSinal

Esse módulo contem a principal rotina do programa, pois a partir dela é que serão chamadas as outras rotinas implementadas ao longo do trabalho.

Ela é responsável por chamar as rotinas de apresentação de telas, escolha de opções, habilitação de interrupção, inicialização da placa CAD 12/36, e indiretamente de todas as outras rotinas.

Ë nessa Unit que foi desenvolvido a rotina de tratamento de interrupções.

5.1.2. Unit Globais

Esta unit representa a base da estrutura de dados utilizada para armazenar as variáveis inerentes do programa, bem como as constantes, que são utilizadas globalmente. Como exemplo das variáveis temos os valores de configuração de cada canal de entrada e de saída

As constantes utilizadas globalmente pelo programa são:

```
SecLimite = 0;{End. secundário da CAD12/36 - Reg. de Limite}
SecPonteiro =1; { End. secundário da CAD12/36 - Ponteiro Memória}
SecComAD = 2; {End. secundario da CAD12/36 -Comando Conv. A/D}
SecRM = 3;  {End. secundário da CAD12/36 - Reg. de Modo}
SecMemoria = 4; {End. secundário da CAD12/36 - Escrita Memoria}
SecAutoCal = 6;  {End. secundario da CAD12/36 - Auto Calibrarão}

NMaxConvBuff = 10000; { Numero máximo de pontos no buffer}
NMAxPoint    = 1;      { Numero de ponteiros para buffer de dados (Valor +
      1)}
cGanho : array [0..3] of real = (1,2,5,10);
constTipo  : array [0..4] of string = ('S1','S2','S3','S4','S5');

deslocX = 100;
```

As variáveis utilizadas globalmente são

```
Type
TpValAD =  array [0..NMaxConvBuff ] of single; {Valor lido por canal no.
      Pontos(IndConv), Canal}
pTpValAD = ^TpValAD;

type EntradaRec = record          {Armazena as configurações dos }
valor: real;                      {canais de entrada (A/D)       }
status: boolean;
tipo: integer;
ganho: integer;
id: string;

SaidaRec = record                 {Armazena as configurações dos }
status: boolean;                  {canais de saída (D/A)         }
tempo: longInt;
TempoON: longInt;
valor: single;

DMA: integer;
AddBase: word;            { Endereco base do conversor A/D}
NCanais: integer;         {Numero de canais a serem convertidos }
VMaxLoc: array [0..15] of single;   { Limite superior do A/D (por Canal) }
VMinLoc: array [0..15] of single;   { Limite inferior do A/D (por Canal) }
```

```
DadoOK: boolean;                { existe dado convertido }
Cn: integer;                    { Canal em questao }
CnGraf: array [0..7] of integer;
VADGraf: array [0..17] of single;  {0..15 Canal do A/D ;16=Flash;
    17=Rele}
DadoGravado: boolean;           {informa que houve pelo menos um
    dado gravado }

EndBase: word;          {Endereco base da placa CAD12/36}
CadCtr0: word;          {Reg. Contador 0 da CAD12/36}
CadCtr1: word;          {Reg. Contador 1 da CAD12/36}
CadCtr2: word;          {Reg. Contador 2 da CAD12/36}
CadModo: word;          {Reg. de Modo do Timer}
CadStatus: word;        {Reg. Estado da CAD12/36}
ByteA: word;            {Reg. Byte A do conversor A/D}
ByteB: word;            {Reg. Byte B do conversor A/D}
CadEnd: word;           {Registrador de Endereco}
CadDado: word;          {Reg. dado de escr. memoria}
IndConv: integer;       {Indice para gravar no buffer de dados}
buffer: integer;        {buffer que sera carregado}
BuffCheio: array [0..NMAxPoint] of boolean;  { Informa Buffer cheio}
VetValAD: array [0..NMaxPoint] of pTpValAd;arq: file;
NInt: integer;          {interrupcao usada}
```

```
status: byte;                {Status da CAd1236}

Time: LongInt;          {Tempo final da conversao}
TimeCount: Real;             {contador do tempo de conversao (ms)}
AccCount : LongInt;          {Acumulador de tempo}
Aquisita : boolean;          {indica inicio de aquisicao}
AccTimeFlash : Single;       {acumulador de tempo p/ Flash}
CondFlash : boolean;{Condicao do flash}

AccTimeRele : Single;        {acumulador de tempo p/ Rele}
CondRele : boolean;          {Condicao do Rele}

Atuadores: byte;             {Situacao dos atuadores (bit : 0 = flash; 1 =
    rele)}
AtuadoresOK: boolean;

Freq1                        {Frequencia escolhida pelo usuário}
FreqReal:Integer;            {Frequencia calculada pelo programa}
```

A listagem completa com a estrutura utilizada nessa Unit esta disponível no capítulo 9. ANEXOS.

5.1.3. Unit Telas

Esta Unit contém as as rotinas que apresentam as telas utilizadas pelo programa principal **"Program AQSinal"**, como a tela de apresentação, a tela de configuração dos canais de entrada, a tela de configuração dos canais de saída, as telas de leitura e gravação das configurações da placa, a tela de parâmetros de ensaio e a tela de configuração de hardware.

A figura 4 mostra a tela principal, que aparece com a execução do arquivo AQSinal.exe.

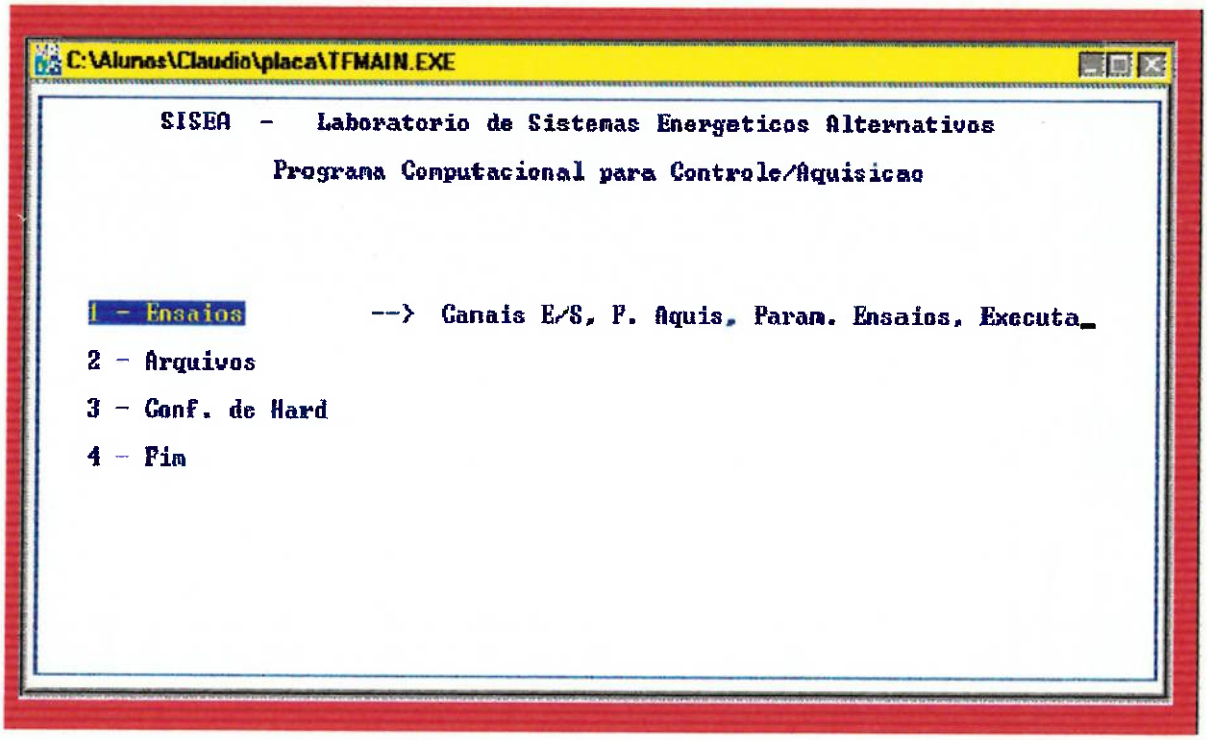


Figura 4. Tela Principal

Suas rotinas são as seguintes:

- Function Centraliza(C1,C2:integer; titulo:string):string: coloca um texto numa posição centralizada na tela;
- Procedure Tela: escreve o nome do Laboratório "SISEA – Laboratório de Sistemas Energéticos Alternativos ;
- Procedure WriteTitulo(titulo:string): escreve um título;
- Procedure TelaSaida: apresenta a tela de saída do programa;
- Procedure TelaConfSaida: apresenta a tela de configuração dos parâmetros de Configuração dos Canais de Saída da Placa, como Status, Tempo X, TempoOn e Valor;
- Procedure LimpaTela: limpa a tela para colocação de outra tela,;
- Procedure TelaConfigEntrada: apresenta a tela de configuração dos parâmetros de Configuração dos Canais de Entrada da Placa, como Status, Ganho, e Tipo;
- Procedure TelaFreq: apresenta a tela de configuração dos "Parâmetros de ensaio" do programa, como a Frequência de Aquisição, e o Tempo de Aquisição;
- Procedure TelaHard; apresenta a tela de configuração dos parâmetros de configuração da placa, como o DMA (Acesso Direto a Memória), o IRQ (Número da Interrupção) e o Endereço de Base;

5.1.4. Unit ArqTF

Esta unit faz o gerenciamento dos arquivos de configuração de entrada e saída da placa. Ele grava e lê a configuração de cada um dos 16 canais de entrada e dos 2 canais de saída da placa.

Suas rotinas são as seguintes:

- Procedure EscreveConfiguracao (arq: string; CanalEntrada: VetorEntrada; CanalSaida: VetorSaida). Essa rotina grava as configurações em um arquivo especificado pelo usuário. As configurações gravadas são: configurações dos canais de entrada e saída, parâmetros de ensaio (tempo e frequência de aquisição) e configuração de hardware (DMA, IRQ e Endereço de Base);
- Procedure LeConfiguracao (arq: string; var CanalEntrada: VetorEntrada; var CanalSaida: VetorSaida;var LeuCerto: boolean).Essa rotina recupera as configurações que foram previamente gravadas, através da opção **Salvar**;

5.1.5. Unit EditTool

Esta unit foi apresentada no Laboratório de CAD (PMC-497) e devido a sua grande funcionalidade, foi utilizada, com as devidas autorizações, para realizar a editoração dos valores de configuração diretamente na tela, de modo rápido e rápido. Portanto é preciso deixar claro que esta Unit não foi implementada neste trabalho., e as rotinas aqui apresentadas não serão explicadas.

Sua rotinas são as seguintes:

- Procedure LeTecla(var tecla:char;var teclafun:boolean);
- Procedure Edita(var palavra:string; var auxreal: real;rs: boolean; col, lin, tam: integer);
- Procedure SetCorSelecionada;
- Procedure SetCorNormal;
- Procedure LeString (Var Campo: string; C, L, Tam: integer; Var FlgIns: boolean; FlgConfirm: boolean;Var CodAbandono: char);
- Procedure LinhaHor(Linha,C1,C2:integer);
- Procedure LinhaVer(Coluna,L1,L2:integer);
- Procedure Box(x1,y1,x2,y2:integer);
- Procedure Edit (l1,c1,l2,c2,nregistros,ncampos:integer; VetorLarg: TipoVetLarg; VetTit:TipoVetTit);

- Procedure Get (vetlinha, vetcoluna, vettamanho: vetinteiro; var vetget: vetstring; vetrealstring: vetboolean; numerogets: integer; FlagWrap, FlagEsc:boolean);
- Function Prompt (var vetlinha, vetcoluna, vetlinhames, vetcolunames: vetinteiro; var vetprompt, vetmensagem: vetstring; numeroprompts: integer; FlagWrap, FlagEsc ,FlagMes: boolean) :char;

5.1.6. Unit GetVal

Esta Unit apresenta as rotinas onde o usuário precisa escolher uma das opções apresentadas pelas telas, ou alterar configurações, todas descritas na Unit Telas.

As rotinas dessa unit são:

- Procedure GetPrincipal(var b:char). Esta rotina apresenta as opções que existem na Tela Principal: **Ensaios**, **Arquivos**, **Conf. de Hard** e **Fim**, e pede que o usuário escolha uma delas;
- Procedure GetEnsaios(var ens:char), Esta rotina apresenta as opções que existem na Tela de Ensaios: **Entrada**, **Saída**, **Parâmetros de Ensaio**, **Executar** e **Fim**, e pede que o usuário escolha uma delas;
- Procedure GetConfigSaida (var CanalSaida: VetorSaida). Esta rotina apresenta as opções existentes na Tela de Configuração dos Canais de Saída, e o usuário pode configurar os dois canais existentes independentemente, sendo os itens configuráveis: Status, Tempo, TempoON, e Valor;
- Procedure GetConfigEntrada (var CanalEntrada: VetorEntrada; var TotalEntrada: integer) Esta rotina apresenta as opções existentes na Tela de Configuração dos Canais de Entrada, e o usuário pode configurar os dois canais existentes independentemente, sendo os itens configuráveis: Status, Ganho e Tipo de Sensor;

- Procedure GetArquivos(var getarq:char). Esta rotina apresenta as opções existentes na Tela de Manipulação dos Arquivos de Configuração: **Gravar**, **Recuperar** e **Sair**, e pede que o usuário escolha uma;
- Procedure GetFreq(var freq:integer;var Time:longint) . Esta rotina apresenta as opções existentes na Tela dos Parâmetros de Ensaio: **Frequência de Aquisição** e **Tempo de Aquisição**, e possibilita ao usuário definir seu ensaio;
- Procedure GetHard(var DMA,Nint:integer;var AddBase:word) . Esta rotina apresenta as opções existentes na Tela de Configuração de Hardware: **DMA**, **IRQ**, e **Endereço de Base** e possibilita ao usuário definir os parâmetros referentes a configuração do computador e a placa CAD 12/36;

5.1.7. Unit CAD1236

Esta Unit contem as rotinas necessárias ao funcionamento da Placa CAD 12/36. Algumas foram implementadas pela própria LINX, sendo que somente as rotinas que manipulam o Relé e o Flash foram implementadas nesse trabalho. Essas rotinas são:

- procedure EscreveRegSecundario (EndReg, Dado: byte). Feita pela LINX;
- procedure Fim_CAd1236. Feita pela LINX;
- procedure Init_Cad1236. Feita pela LINX;
- procedure Init_Cad1236R4. Feita pela LINX;
- procedure ProgramaMemoriaCanais(tipo : integer); (* tipo 0 = R3 e 1= R4 *). Feita pela LINX;
- procedure Flash. Essa rotina manipula os dados de um dos canais do D/A;
- procedure Rele. Essa rotina manipula os dados do outro canal do D/A;

5.1.8. Unit Graf_T

Esta Unit trabalha em modo gráfico para mostrar os dados lidos no A/D como um osciloscópio.

- procedure VerSinal: essa rotina mostra os sinais coletados no A/D e mostra esses sinais como um osciloscópio;
- function CalculaPonto(Venge: single; XY : char; DeslY :integer):integer. Calcula o ponto onde deve ser colocado o valor lido no A/D.
- procedure TracaLinhasBase(color : word): traça as linha que separam os gráficos quando apresentados em modo multi-visualização;
- procedure MarcaCanal(Cor : word): indica o canal que está sendo utilizado, através de uma cor diferenciada;

5.1.9. Unit Arquivo

Esta Unit faz o armazenamento de todos os sinais utilizados em um arquivo para uma possível análise em programas como Microsoft Excel, ou MatLab onde se pode fazer ... As rotinas implementadas nessa Unit são:

- procedure SalvaDadosParcial: essa rotina salva em disco os dados que não completaram o buffer por completo. ;
- procedure SalvaDados: essa rotina salva em disco os dados que completaram o buffer por completo;
- procedure ConvReal_to_text. Essa rotina converte números reais em texto;

5.1.10. Unit ERR_STR

Essa rotina apresenta uma tabela com os principais erros que podem existir quando se utiliza uma placa de Aquisição de Dados. Esses códigos de erros foram apresentados pela Linx e utilizados por completo. Entre Elas temos:

-1: S:= 'Erro interno';

1: S:= 'Arquivo já existe';

2: S:= 'Arquivo não encontrado';

3: S:= 'Diretório não encontrado';

4: S:= 'Muitos arquivos abertos';

5: S:= 'Acesso ao arquivo n„o permitido';

6: S:= 'Erro na configuração de Hardware';

8: S:= 'Memória insuficiente';

12: S:= 'Acesso ao arquivo inválido';

15: S:= 'Drive inválido';

16: S:= 'Nome do arquivo continua o mesmo? (S/N)';

39: S:= 'Correla‡Æo abaixo da esperada';

40: S:= 'Valor abaixo do permitido';

41: S:= 'Valor acima do permitido';

43: S:= 'Valor n„o pode ser igual';

44: S:= 'Numero de pontos deve ser maior que 1';

```
45: S:= 'Imposs¡vel acrescentar mais ganhos';

46: S:= 'Imposs¡vel retirar mais ganhos';

47: S:= 'Imposs¡vel retirar ganhos intermedi rios';

48: S:= 'Todos os ganhos tem o mesmo nome? (S/N)';

51: S:= 'Falta nome de arquivo';

52: S:= 'N„o pode gerar PCX em modo texto';

100: S:= 'Erro na leitura em disco';


101: S:= 'Erro na grava‡„o em disco (disco cheio)';

102: S:= 'Arquivo n„o especificado';

103: S:= 'Arquivo n„o foi aberto';

104: S:= 'Arquivo n„o foi aberto para leitura';

105: S:= 'Arquivo n„o foi aberto para escrita';

150: S:= 'Disco com selo de prote‡„o';

151: S:= 'Unidade desconhecida';

152: S:= 'Dispositivo n„o preparado';

154: S:= 'Erro de CRC';

156: S:= 'Erro na busca em disco';

157: S:= 'Dispositivo desconhecido';

158: S:= 'Setor n„o encontrado';

159: S:= 'Falta papel na impressora';

160: S:= 'Falha na escrita em dispositivo';

161,
```

```
149: S:= 'Falha na leitura em dispositivo';

162: S:= 'Frequencia muito alta / Interrupcao';


200: S:= 'Divis„o por zero';

201: S:= 'Indice ou valor inv lido';

202: S:= 'Pilha insuficiente';

203: S:= 'Mem¢ria insuficiente';

204: S:= 'Ponteiro inv lido';

205: S:= 'Overflow de ponto flutuante';

206: S:= 'Underflow de ponto flutuante';

207: S:= 'Ponto flutuante inv lido';

208: S:= 'Erro no gerenciador de overlay';

209: S:= 'Erro na leitura de overlay';
```

5.1.11. Unit TextBas

Esta rotina foi implementada pelo André (Linx), e funciona basicamente como a Unit EditTool, porém, acrescenta algumas facilidades que julguei serem importantes para o manuseio de opções nas telas de configuração. Sendo assim não será explicado o conteúdo dessa Unit, tampouco será mostrada a listagem dessa Unit.

5.2. Dispositivos de Interfaceamento de Potência

Nessa primeira parte do trabalho foi feita apenas um estudo das possibilidades existentes para o acionamento de dispositivos a partir de um sinal proveniente de um microcomputador. O grande problema existente para um acionamento direto é a baixa potência de um sinal de saída de um micro (5V e corrente máxima de 10 mA). No nosso caso, é necessário acionar um relé de 220V e corrente máxima de 0,5A. Daí a necessidade de se utilizar um dispositivo de interfaceamento de potência.

5.2.1. Acionamento do Solenóide

Para a escolha de um acionamento adequado do solenóide, foi necessário primeiramente determinar as características elétricas desse solenóide. Para a escolha do solenóide, um estudo realizado no trabalho de pós-graduação (já citado), indicou a necessidade de se utilizar um solenóide específico, que possui as seguintes características:

- Tensão de alimentação: 220 V;
- Corrente máxima: 0,5 A;

Essa escolha se baseou principalmente em velocidade de acionamento, força de tração do solenóide, preço e disponibilidade.

Com base nessas características, foi proposto a utilização de um relé de estado sólido, para simplificação do dispositivo de acionamento. O Data Sheet referente a esse relé esta anexo no item **9. ANEXOS**

A seguir encontra-se o esquema de montagem do solenóide, com o relé de estado sólido e o sinal do microcomputador.

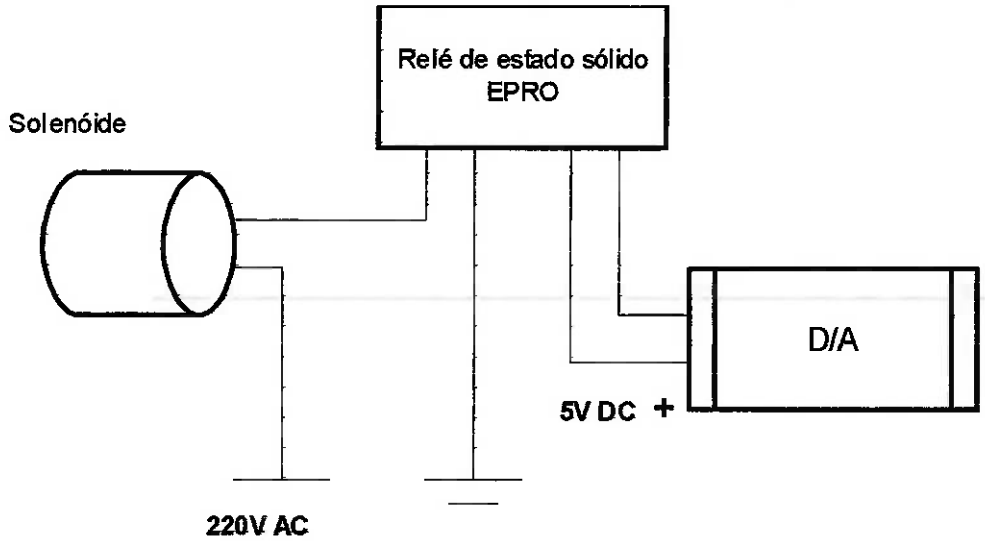


Figura 5. Montagem do Dispositivo de Acionamento do Solenóide

5.2.2. Acionamento do "Flash"

O "Flash"escolhido é da marca **EG&G** da série **1100 Series FlashPacs** e possui as seguintes características:

• Intensidade ultra-radiante;
• Espectro Contínuo UV-VIS-IR;
• Alta taxa de repetição de flash;
• Radiação de cabeça baixa;
• Duração do flash de microsegundos;
• Seleção do tipo de lâmpada de flash;
• Seleção do capacitor de descarga;
• Sem necessidade de aquecimento;
• Compacto;

As características elétricas para acionamento do flash são as seguintes:

Voltagem (VDC)	15 - 18
Corrente DC (A)	<1.2 a 24V
Trigger	TTL

Portanto podemos utilizar o sinal do D/A diretamente para acionar o "Flash", sem a necessidade de se construir um dispositivo de interfaceamento de potência. Entretanto, para melhor utilização do "Flash", foi desenvolvido por

em outro trabalho, um conjunto eletrônico, com uma fonte de alimentação que possibilita o chaveamento do "Flash" através de 3 modos:

- A partir do sinal do D/A;
- Manualmente, com um disparo;
- Intermitente.

É importante ressaltar que esse conjunto eletrônico não foi desenvolvido neste trabalho.

Outras informações sobre o flash podem ser encontradas anexo, no item **9.3. Data Sheet do Flash**.

6. Manual do Usuário

6.1. Tela Principal

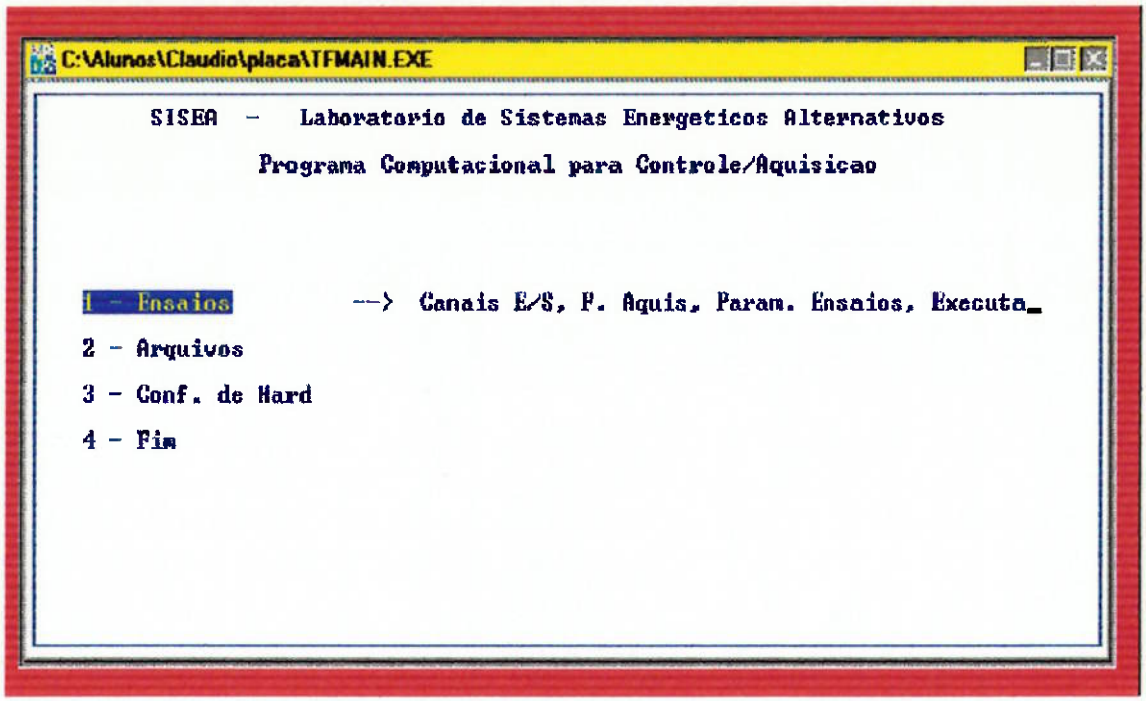


Figura 6. Tela Principal

Nesta tela, tem-se 4 opções:

1. Ensaios: aqui pode-se configurar a disponibilizaçao dos canais de entrada, os parâmetros dos canais de saída, mudar os parâmetros de ensaio, como Frequência e Tempo de Aquisição;
2. Arquivos: nesta opção pode-se escolher entre gravar a configuração ou recuperar uma configuração previamente armazenada;
3. Conf. De Hard: faz-se a configuração dos parâmetros importantes para o funcionamento da placa, como DMA, IRQ e Endereço de Base;

4. Fim: sai do programa, e retorna para o prompt.

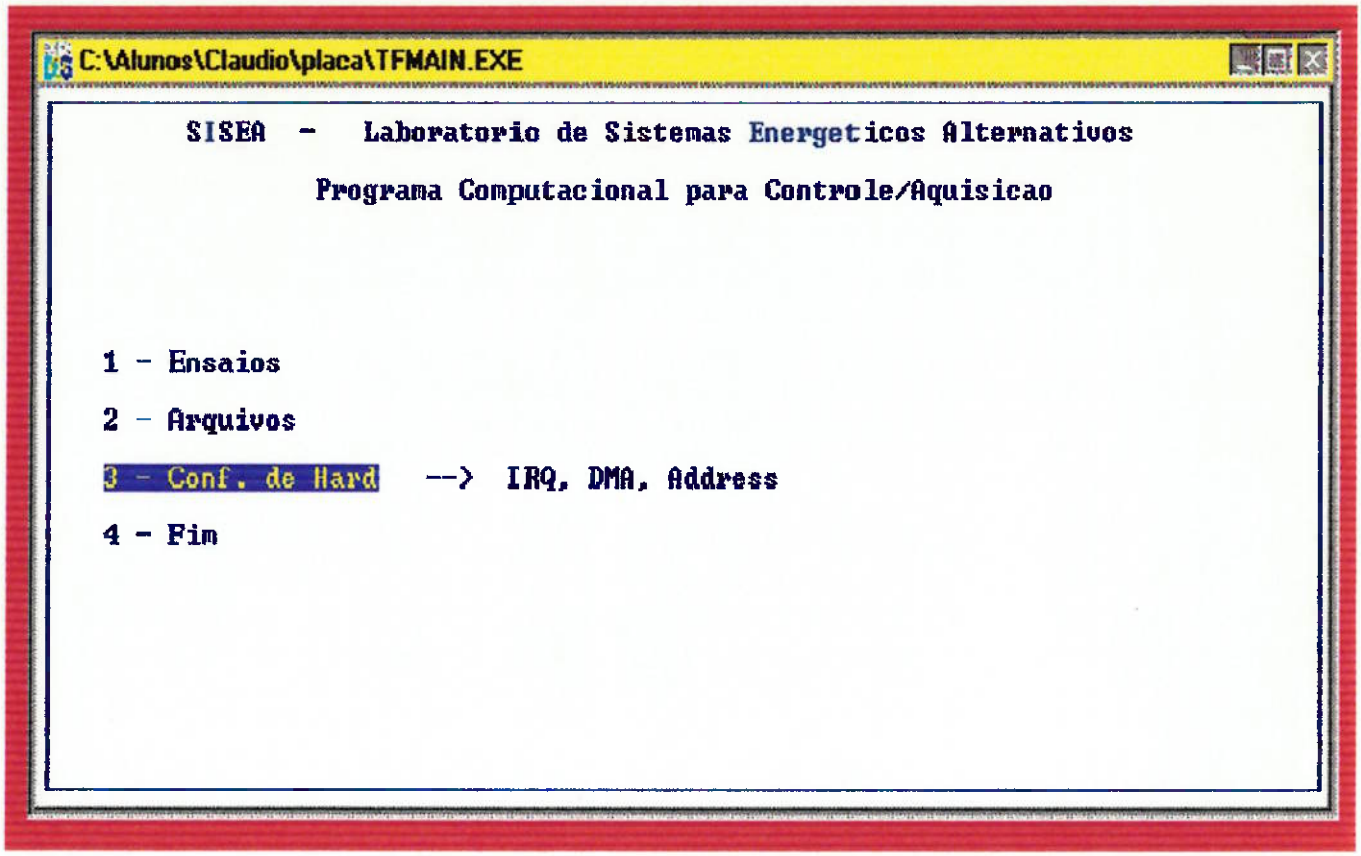


Figura 7. Tela de Conf. de Hard

6.1.1. Ensaios

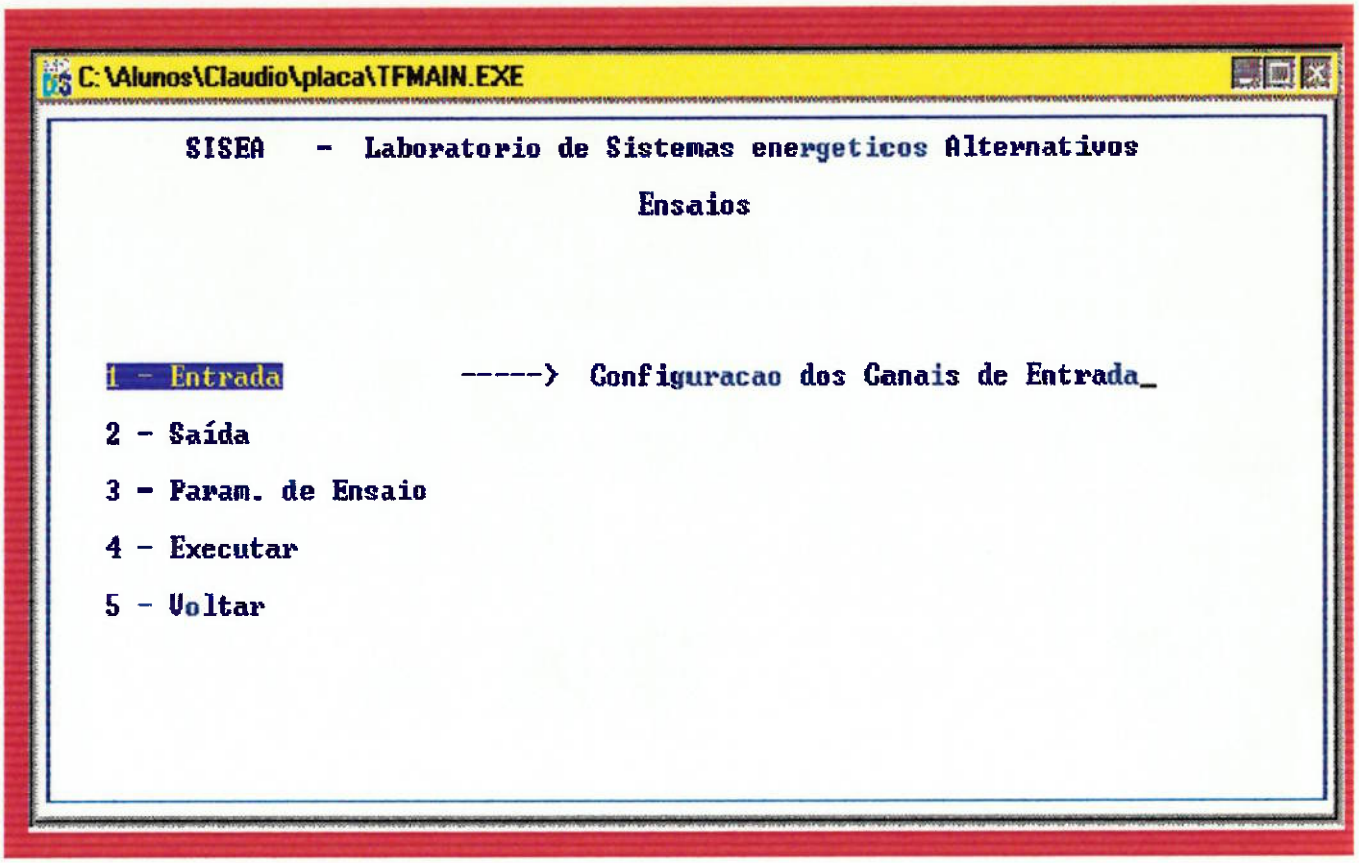


Figura 8. Tela de Ensaio

Nesta tela, tem-se as seguintes opções, como mostrado na figura 8: **Entrada**, **Saída**, **Param. De Ensaio**, **Executa** e **Fim**.

1. **Entrada**: aqui faz-se a configuração dos Canais de Entrada (dezesseis) do conversor A/D que estarão em funcionamento, além de outros parâmetros, como por exemplo, o Ganho e o Tipo de Sensor que será utilizado no respectivo canal. Deve-se proceder do seguinte modo: marca-se com um "S" o Status do Canal que se deseja deixar

habilitado para Aquisição. Feito isso, deve-se selecionar o valor de Ganho que está sendo utilizado no Amplificador Condicionador de Sinais, e então selecionar um tipo de Sensor que será utilizado.

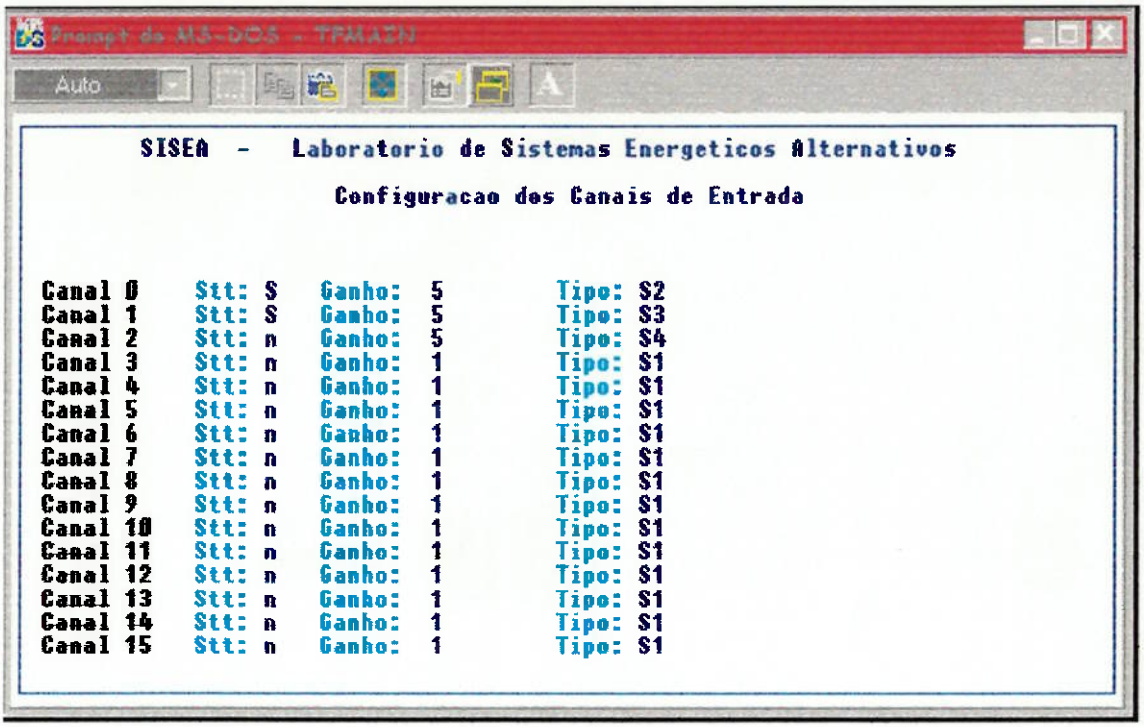


Figura 9. Tela de Configuração dos Canais de Entrada

2. **Saída**: analogamente faz-se a configuração dos Canais de Saída (dois) do conversor D/A que estarão em funcionamento, além de parâmetros como Tempo até o disparo (Tempo0, para o Canal0 e Tempo1 para o Canal1, todos em milisegundos), Tempo em que o sinal vai permanecer ligado (TempoON), e o valor de saída do sinal (VALOR). Deve-se proceder do seguinte modo: Seleciona-se quais dos canais estarão funcionando: o Canal0, o Canal1 ou ambos,

selecionando-se o Status e colocando um "S". Feito isso deve-se colocar os valores de tempo correspondente a cada campo de cada canal. Deve-se notar que os valores de tempo (Tempo0, Tempo1 e TempoON), devem ser múltiplos do Período de Aquisição. Esse período é calculado como sendo o inverso da Frequência de Aquisição. Por exemplo, se a Frequência de Aquisição é 100Hz, o Período de Aquisição é 1/100=0,010 s ou 10 ms. Isso significa que os tempos citados (Tempo0, Tempo1 e TempoON), precisam ser múltiplos de 10 ms.

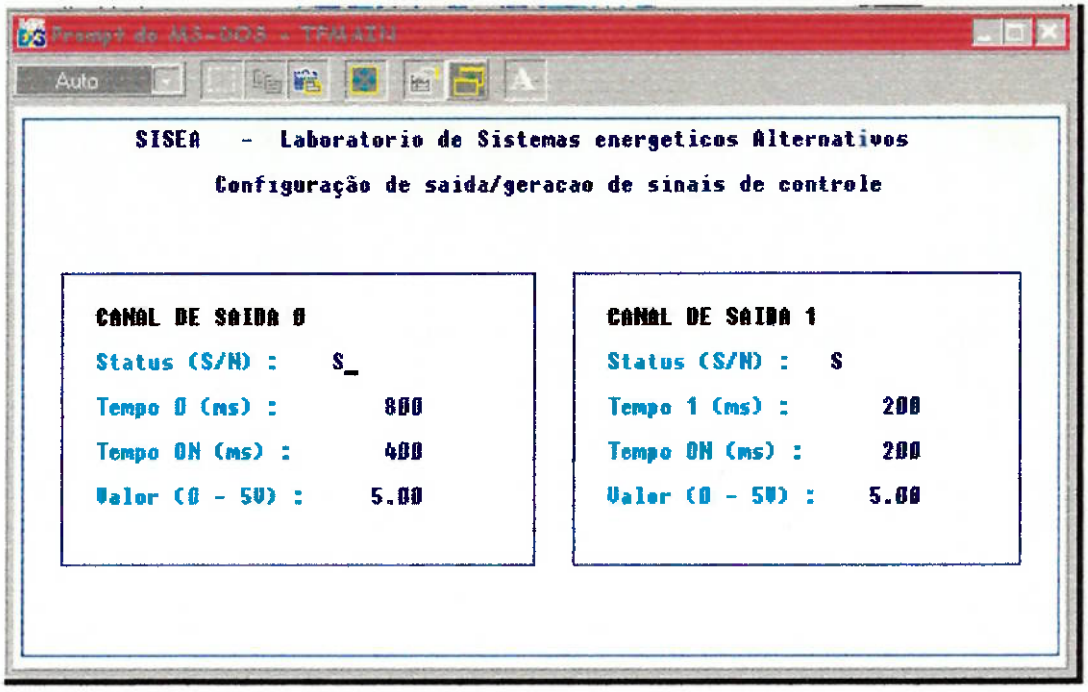


Figura 10. Tela de Configuração dos Canais de Saída

3. **Param. De Ensaio**: configura-se a Freq. De Aquisição e o Tempo de Aquisição (em milisegundos). Deve-se notar que quanto maiores a

Freq. De Aquisição e o Tempo de Aquisição, maior o arquivo que será gerado com os valores coletados pelos canais A/D. Portanto, não se deve aumentar muito esses valores. Outro problema ocorre para valores de Freq. De Aquisição muito grandes, pois isso é diretamente relacionado com a velocidade de processamento da placa CAD 12/36 e o número de Canais Habilitados.

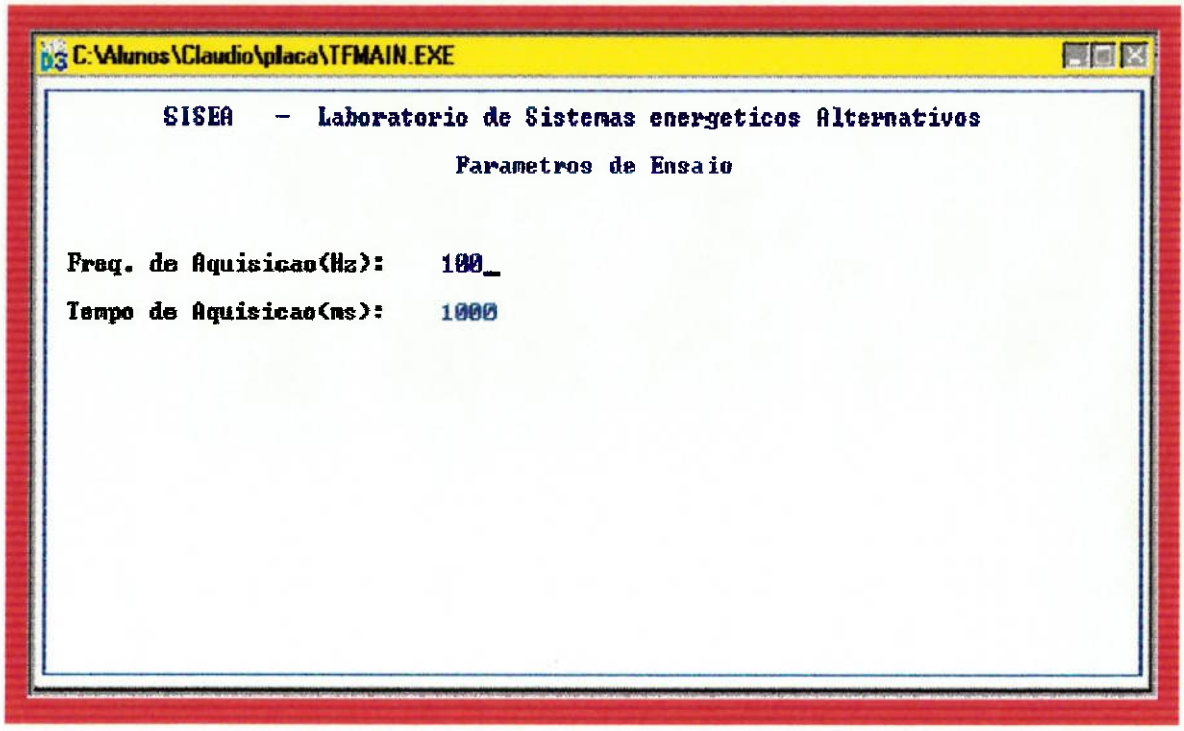


Figura 11. Tela de Parâmetros de Ensaio

4. **Executa**: esta opção entra no modo gráfico para aquisição dos sinais. Assim que se entra na opção, o sistema está no modo de visualização ou modo OSCILOSCÓPIO, onde nenhum sinal de comando é gerado no D/A e nenhum valor é gravado no arquivo texto (este arquivo

armazena os valores dos sinais coletados nos Canais selecionados). A partir de então, temos as seguintes opções:

A. **Enter** ou **F1:** continua o modo OSCILOSCÓPIO, faz GERAÇÃO DE SINAIS NO D/A e a GRAVAÇÃO DOS SINAIS.DO A/D Ao final do Tempo de Aquisição, o sistema volta para a tela de Ensaio;
B. **CTRL + F1:** continua o modo OSCILOSCÓPIO e faz GERAÇÃO DE SINAIS NO D/A, sem terminar esse modo depois de passado o Tempo de Aquisição. Esse modo funciona como um GERADOR DE ONDAS QUADRADAS;
C. **F3:** OffSet: desloca o gráfico para cima ou para baixo. Use essa tecla quando o sinal esta deslocado de seu eixo central e não é possível a visualização de todo o sinal;
D. **Home:** Zera o Offset
E. **F5:** aumenta o número de Telas de Visualização, até o máximo de 8 telas.
F. **F6:** diminui o número de Telas de Visualização, até o mínimo de 1 tela;
G. **SETA P/ DIREITA:** aumenta o número do Canal da tela selecionada;
H. **SETA P/ESQUERDA:** diminui o número do Canal da tela selecionada;
I. **SETA P/ CIMA:** Seleciona a tela acima, caso estejam sendo exibidas mais de uma tela;
J. **SETA P/ BAIXO:** Seleciona a tela abaixo, caso estejam sendo exibidas mais de uma tela;
K. **ESCAPE:** volta para a Tela de Ensaio;

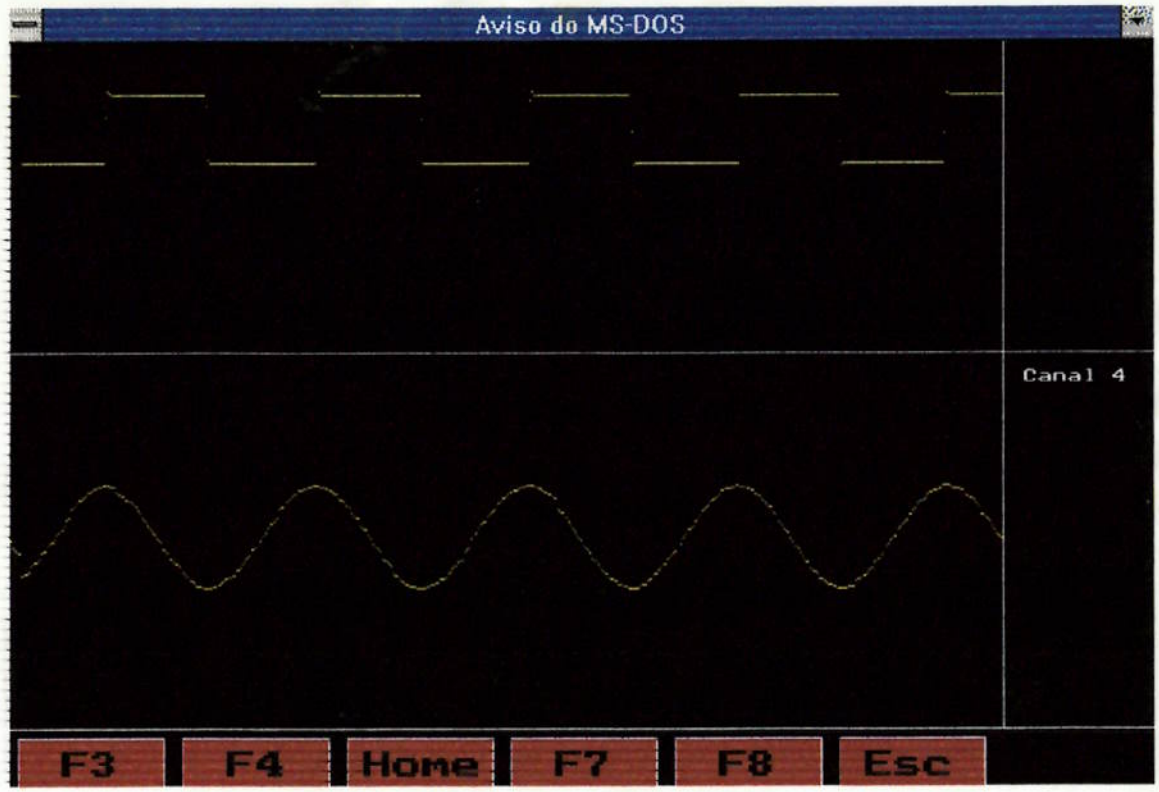


Figura 12. Tela de execução mostrando dois canais

5. **Fim**: volta para a tela principal.

6.1.2. Arquivos

Nesta tela, temos duas opções,

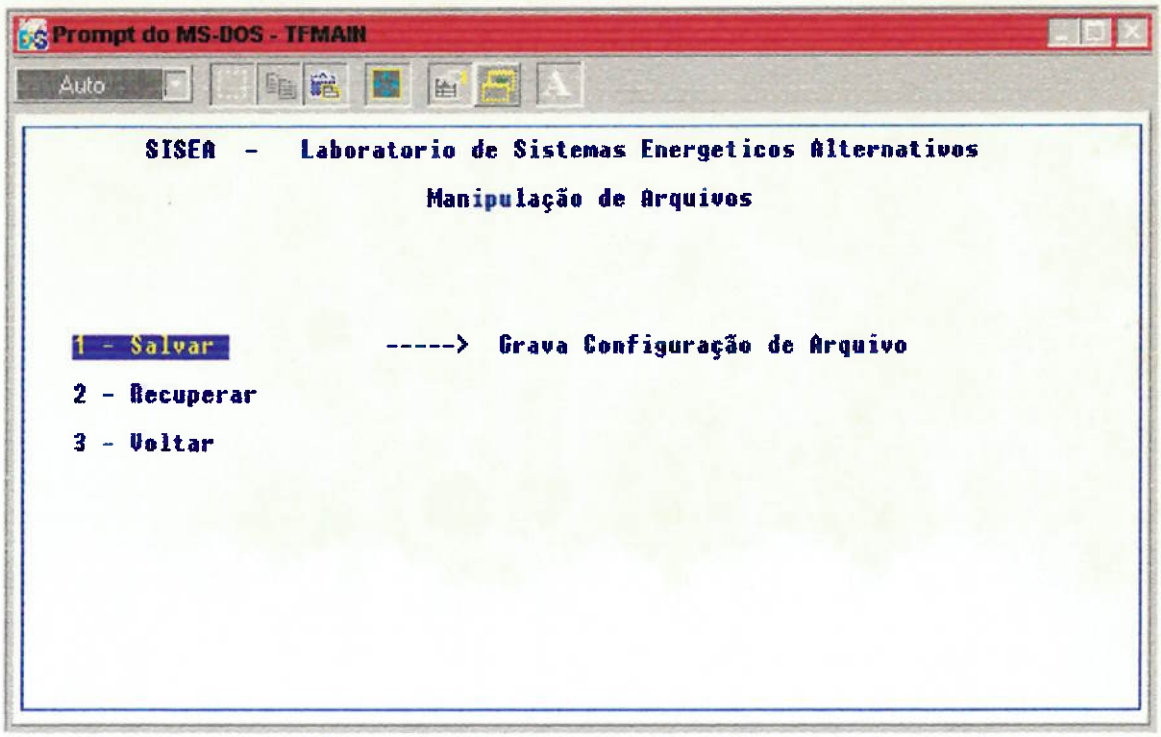


Figura 13. Tela de Arquivos

1. Salvar: esta opção salva as Configurações dos Canais de Entrada e dos Canais de Saída, além dos valores de Configuração de Hardware (DMA, IRQ, Endereço de Base), e dos Parâmetros de Ensaio (Tempo de Aquisição e Frequência de Aquisição). Selecionada esta opção, deve-se escrever o nome do novo arquivo, onde serão gravados todos os dados descreitos anteriormente. Deve-se notar que se for escrito

um nome de arquivo que já exista, este será substituído por um novo, portanto deve-se tomar cuidado para não substituir um arquivo importante.

2. Recuperar: esta opção recupera todos os dados, descritos na opção anterior, que foram gravados. Ao se escolher esta opção, deve-se escrever o nome do arquivo que contém as informações.
3. Sair: voltar para a tela principal

6.1.3. Configuração de Hardware

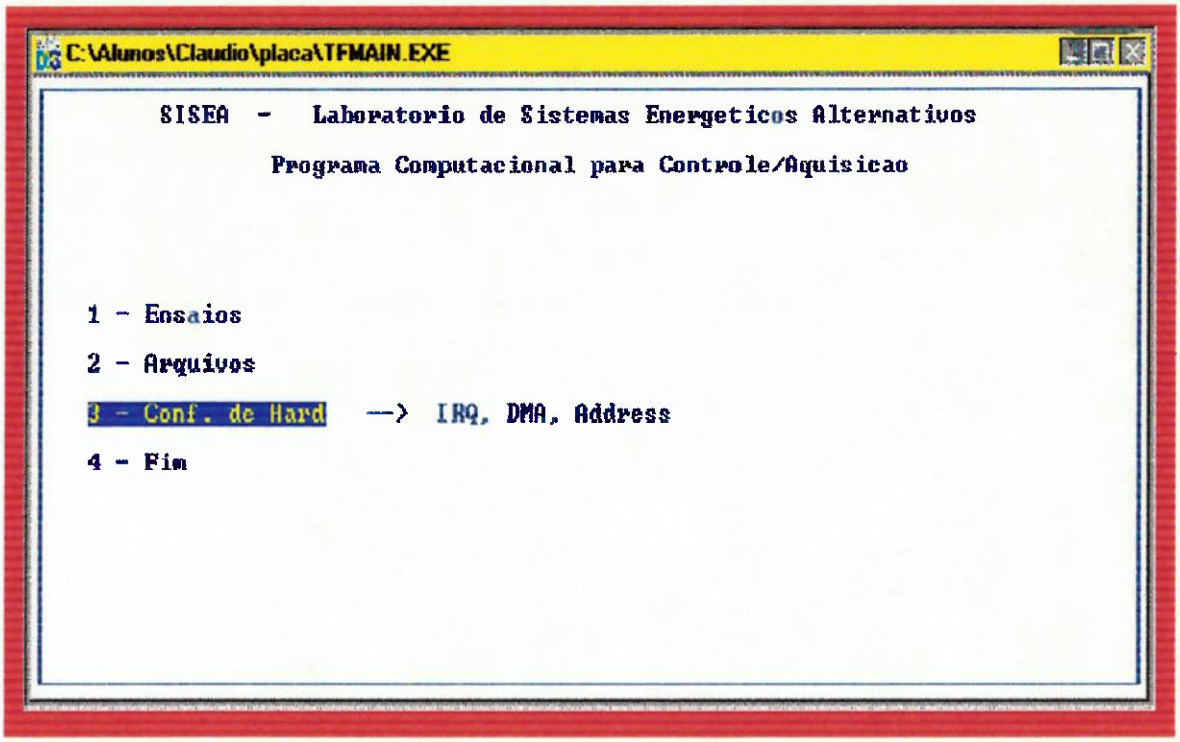


Figura 14. Tela de Configuração de Hardware

Nesta tela, tem-se 3 opções de Configuração de Hardware:

1. DMA: determina o Número do Direct Access Memory
2. IRQ: determina o valor da Interrupção que se'ra utilizada pela placa CAD 12/36. Caso esteja havendo algum problema como conflito de interrupção, deve-se mudar esse número para algum valor de interrupção que não esteja sendo usada. Geralmente, podem ocorrer problemas de conflito de interrupção com placas de rede, ou placas de som (multimídia).

3. Endereço de Base: esse valor indica qual será o início do bloco de endereços que será utilizado pela placa CAD 12/36. Ela ocupa 8 endereços contínuos. A localização desse bloco deve ser selecionada de forma a não coincidir com áreas já ocupadas pelo hardware do microcomputador ou outras placas nele instalada. O endereço inicial da região ocupada pela placa, denominado de Endereço de Base, é, na Configuração Padrão, 896 (ou $0380 na notação hexadecimal). Esse endereço pode ser redefinido através de um conjunto de "junpers" J6 na placa. Deve-se atentar para modificar tanto na configuração do software, quanto na placa, através do "jumper". Mais informações podem ser adquiridas no Manual do usuário e de Referência (LYNX).

A configuração padrão, que é a utilizada no laboratório utiliza DMA=0, IRQ=5 e Endereço de Base=380h.

6.1.4. Fim

Escolha essa opção para sair do programa AQSinal, e voltar para o prompt do DOS

7. Discussão

Apesar de não Ter sido possível a verificação do funcionamento de programa no Laboratório SISEA, pode-se verificar seu funcionamento através de geradores de função e de um osciloscópio. Verificamos a aquisição de dados em todos os canais do A/D através de sinais provenientes do gerador de funções, e verificamos o funcionamento do D/A através da sua leitura no osciloscópio. Observamos também esse funcionamento através da instalação de uma lâmpada e do próprio solenóide (ligada ao relé de estado sólido), e constatamos o funcionamento do sistema.

O mérito desse trabalho não está em sua complexidade, que por sinal, até o presente momento, não mostrou precisar de conhecimentos específicos, e sim na possibilidade de se obter conhecimentos relacionados com eletrônica analógica e digital, juntamente com uma utilização de técnicas de programação que não são vistas nas disciplinas ministradas na Escola Politécnica, como por exemplo programação de canais de entrada e saída, através da utilização de interrupções. Adicionalmente a isso, se deve ao fato de ser parte importante para dar continuidade em um trabalho de pós-graduação, isto é, o resultado de outro trabalho depende deste. Apesar desse trabalho se dar por encerrado, muitas alterações ainda podem ser feitas, para atender a necessidades que irão surgindo no decorrer da utilização do programa.

8. Bibliografia Recomendada

- MOTOROLA; **Optoelectronics Device Data**; Motorola Semiconductor Products, Catálogo;
- HARRIS; **Intelligent Power ICs for Commercial, Industrial and Automotive Applications**; Harris Semiconductor, Catálogo, 1994;
- ANALOG DEVICES; **Amplifier Reference Manual**; Abnalog Devices; Catálogo, 1992;
- MALVINO, ALBERT P.; **Eletrônica**, Volume I; São Paulo; McGRAW HILL, 1987
- MALVINO, ALBERT P.; **Eletrônica**, Volume II; São Paulo; McGRAW HILL, 1987
- LYNX; **CAD12/36 Conversor A/D e D/A para Microcomputadores, Manual do Usuário e de Referência**; LYNX Tecnologia Eletrônica Ltda; 1993
- LYNX; **MSC1000-V2 - Módulo Condicionador de 16 Entradas, Manual do Usuário**; LYNX Tecnologia Eletrônica Ltda; 1996
- PORTER, K. , **Mike. F.; Stretching Turbo Pascal**. Brady, 1987
- RUGG, T., FELDMAN, P., **Turbo Pascal Programmer's Toolkit** , Que, 1993
- MALVINO, A. **Microcomputadores e microprocessadores**, São Paulo, McGraw-Hill, 1985
- UNIVERSIDADE DE SÃO PAULO, **Diretrizes para apresentação de dissertações e teses**, Escola Politécnica - Serviço de Bibliotecas, São Paulo, 1991

9. ANEXOS

9.1. Listagens

9.1.1. Program AQsinal

```
Program AQSinal;

{*************************************************************************}
{                                                                         }
{  Programa Computacional para coleta de dados e geracao de sinal         }
{  para controle (liga e desliga) de dispositivos                         }
{                                                                         }
{*************************************************************************}
{                                                                         }
{  SISEA - Departamento de Engenharia Mecanica                            }
{  Sao Paulo 1997                                                         }
{  Programador: Claudio Tokeiama  NUSP: 2924576                           }
{  Desenvolvido como Trabalho de Formatura                                }
{                                                                         }
{                                                                         }
{*************************************************************************}
uses Crt, Dos, Telas, Edittool, Arqtf, GetVal,TextBas,ErrStr,Globais,
    Cad1236,Graf_T;

var Narq,titulo:string;
   CanalEntrada:VetorEntrada;
   CanalSaida:VetorSaida;

   OldIntVec: pointer;

   a,codchar,ens,getarq:char;
   saidograf,saiprincipal,sai,flins,flagalt:boolean;
   TotalEntrada,gd,gm,i,j: integer;


const
 IntAck     = $20; { Registrador de reconhecimento de interrupcao   }
 IntMask    = $21; { Registrador da mascara de interrupcoes         }

Procedure InicializaDados;
```

```
begin
  for i:=0 to 15 do
    CanalEntrada[i].ganho:=1;
end;

{ ================================================  = }
{ =              Rotina  InibeINT                    = }
{ ================================================  = }

Procedure InibeINT;
  Var Mask : byte;

begin
  Mask          := Port [IntMask];
  Port [IntMask] := Mask or ($01 shl NInt);
end;


{ ================================================  = }
{ =              Rotina  HabilitaINT                 = }
{ ================================================  = }

Procedure HabilitaINT;
  Var Mask : byte;

begin
  Mask          := Port [IntMask];
  Port [IntMask] := Mask and not ($01 shl NInt);
  Port [IntAck]  := $60 + NInt;   { limpa reg. de interrupcoes }
end;

procedure TrataInt; interrupt;
var i : integer;
  bAD : LongInt;
begin
  InLine($fb);
  if Not FlgTrtInt then
  begin

    AccTimeFlash:= AccTimeFlash + 1/Freqreal * 1000;
    if ((AccTimeFlash = CanalSaida[0].tempo) and not CondFlash )
       or ((AccTimeFlash = CanalSaida[0].tempoON) and CondFlash) then
    begin
      AccTimeFlash := 0;
      CondFlash := not CondFlash;
      Flash;
    end;
```

```
AccTimeRele := AccTimeRele + 1/FreqReal * 1000;
if ((AccTimeRele = CanalSaida[1].tempo) and not CondRele)
  or ((AccTimeRele = CanalSaida[1].tempoON) and CondRele) then
begin
 AccTimeRele := 0;
 CondRele := not CondRele;
 Rele;
end;

flgTrtInt    := true;
status := port[cadStatus];
Port [IntAck]  := $60 + NInt;   { limpa reg. de interrupcoes }
if aquisita then
 TimeCount := TimeCount + 1/FreqReal* 1000;
if not (buffCheio[0] and buffCheio[1]) and (TimeCount <= Time) then
begin
 if CondFlash then
  VADGraf[16] := 5
 else
  VADGraf[16] := 0;
 if CondRele then
  VADGraf[17] := 5
 else
  VADGraf[17] := 0;
 for Cn := 0 to NCanais - 1 do
 begin
  bAD:= ((Port [ByteB] shl 8)+ Port [ByteA]); { Byte A deve ser lido primeiro }
  VADGraf[Cn] := Bad * 10 / 32768;
 end;
 for Cn := 0 to NCanais - 1 do
 begin
  VetValAD[buffer]^[indConv] := VADGraf[Cn];
  if aquisita then
   inc(indConv);
  if indConv > NMaxConvBuff then
   begin
    BuffCheio[Buffer] := true;
    indConv :=  0;
    if Buffer = 0 then
     buffer := 1
    else
     buffer := 0;
   end;
 end;
 for Cn := 16 to 17 do
 begin
```

```
        VetValAD[buffer]^[indConv] := VADGraf[Cn];
        if aquisita then
          inc(indConv);
        if indConv > NMaxConvBuff then
          begin
            BuffCheio[Buffer] := true;
            indConv :=  0;
            if Buffer = 0 then
              buffer := 1
            else
              buffer := 0;
          end;
      end;
    end;
    if buffCheio[0] and buffCheio[1] then
      FlgErrInt := true;
    flgTrtInt     := false;
  end
  else
    FlgErrInt := true;
end;


begin
  atuadores := 1;
  DadoGravado := false;
  aquisita := false;
  for indConv := 0 to nMaxConvBuff do
    for buffer := 0 to NMaxPoint do
      VetValAD[buffer]^[round(IndConv)] := 0;
  AccCount := 0;
  buffCheio[0] := false;
  buffCheio[1] := false;
  TimeCount := 0;

  CanalSaida[0].tempo:= 800;       { Intervalo entre os diparos do flash (ms) }
  CanalSaida[0].tempoON:= 400;     { Tempo que o flash fica ativo (ms) }
  AccTimeFlash := 0;         { zera acumulador de tempo p/ Rele }
    CondFlash := false;    { Flash nao disparado }

  CanalSaida[1].tempo:=200;        { Intervalo entre os diparos do Rele (ms) }
  CanalSaida[1].tempoON:=200;      { Tempo que o rele fica ativo (ms) }
  AccTimeRele  := 0;         { zera acumulador de tempo p/ Rele }
     CondRele := false;    { Rele nao disparado }

   AtuadoresOK := False;    { se nao esta aquisitando, os atuadores estao em
repouso }
```

```
  Flash;
  Rele;

  assign (arq,'teste.Dat');
  rewrite(arq,1);
  IndConv      := 0;
  buffer       := 0;
  flgTrtInt    := false;
  flgErrInt    := false;
  flgTrtInt    := false;
  for i := 0 to 15 do
  begin
   VMaxLoc[i] :=  10;
   VMinLoc[i] := -10;
  end;
  NInt := 5;
  ClrScr;
  AddBase := $380;
  Time := 1000; {ms / default}
  NCanais := 8;
  FreqReal := 100;    {default}
  Init_Cad1236;
  status := port[CadStatus];
  EscreveRegSecundario (SecRM,$CC);{ Conversao ao final de burst,inicio em
timer 0   }
  GetIntVec(NInt + 8,OldIntVec);
  SetIntVec(NInt + 8,@TrataInt);
  HabilitaInt;
  CN := 0;
   ClrScr;
   SetCorNormal;
   ClrScr;
   Tela;
   InicializaDados;
   saiprincipal:=false;
   While not saiprincipal do begin
     LimpaTela;
     titulo:='Programa Computacional para Controle/Aquisicao';
     titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
     Writetitulo(titulo);
     GetPrincipal(a); {menu principal}
     case a of
       '1': begin            {ENSAIOS}
            sai:=false;
            while not sai do begin
                 LimpaTela;
```

```
                titulo:='Ensaios';
                titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                Writetitulo(titulo);
                GetEnsaios(ens);
                case ens of
                '1': Begin
                    LimpaTela;        {dados de entrada da placa}
                    titulo:='Configuracao dos Canais de Entrada';
                    titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                    Writetitulo(titulo);
                    TelaConfigEntrada;
                    GetConfigEntrada(CanalEntrada,TotalEntrada);
                end;{case '1'}
                '2': begin            {dados de saida da placa}
                    LimpaTela;
                    titulo:='Configura‡Æo de saida/geracao de sinais de controle';
                    titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                    Writetitulo(titulo);
                    TelaSaida;
                    TelaConfSaida;
                    GetConfigSaida(CanalSaida);
                end;{case '2'}
                '3': begin            {Parâmetros de Ensaio}
                    LimpaTela;
                    titulo:='Parametros de Ensaio';
                    titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                    Writetitulo(titulo);
                    TelaFreq;
                    GetFreq(FreqReal,Time);
                end;
                '4': begin            {Execu‡Æo do Ensaio}
                    AddBase:=$380;
                    LimpaTela;
                    titulo:='Execucao do Ensaio';
                    titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                    Writetitulo(titulo);
                    Init_Cad1236;
                    status := port[CadStatus];
                    EscreveRegSecundario (SecRM,$CC);{ Conversao ao final de
burst,inicio em timer 0   }
                    GetIntVec(NInt + 8,OldIntVec);
                    SetIntVec(NInt + 8,@TrataInt);
                    HabilitaInt;
                    CN := 0;
                    ClrScr;
                    VerSinal;
                    fim_cad1236;
```

```
                SetIntVec(NInt + 8,OldIntVec);
                InibeInt;
                if flgErrInt then
                  erro(192);
                {SalvaDadosParcial;}
                {close(arq);}
                status := port[cadstatus];
                {ConvReal_to_text;}
                EscreveDA(0,0);
                EscreveDA(1,0);

            end;
            '5': sai:=true;
            end;
        end;
    end;
    '2': begin          {Arquivos de configuracao}
        LimpaTela;
        sai:=false;
        while not sai do begin
            titulo:='Manipulacao de Arquivos';
            titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
            Writetitulo(titulo);
            GetArquivos(getarq);
            case getarq of
            '1': Begin
                LimpaTela;      {dados de entrada da placa}
                titulo:='Salvar configuracao';
                titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                Writetitulo(titulo);
                flins:=TRUE;
                gotoxy(5,15);
                write('Nome do arquivo :');
                LeString(Narq,38,15,8,flins,true,codchar);
                if Narq<>'      ' then

EscreveConfig(Narq,CanalEntrada,CanalSaida,FreqReal,Time,
                        DMA,NInt,AddBase);
            end;{case '1'}
            '2': begin          {dados de saida da placa}
                LimpaTela;
                titulo:='Recuperar configuracao';
                titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
                Writetitulo(titulo);
                flins:=true;
                Gotoxy(5,15);
                Write('Nome do arquivo (sem extensao) :');
```

```
                    LeString(Narq,38,15,8,flins,true,codchar);
                    if Narq<>'        ' then
                      LeConfig
(Narq,CanalEntrada,CanalSaida,flins,FreqReal,Time,
                              DMA,NInt,AddBase);
                    if not flins then begin
                      Gotoxy(5,16);
                      Write('Arquivo inexistente');
                      delay(2000); {2 segundos de espera}
                    end;
                  end;{case '2'}
                  '3': sai:=true;
                  end;
              end;
          end;
          '3': begin          {salva configuracao}
            LimpaTela;
            titulo:='Configuracao de Hardware';
            titulo:=Centraliza(COLUNAINIC,COLUNAFIM,titulo);
            Writetitulo(titulo);
            TelaHard;
            GetHard(DMA,Nint,Addbase);
          end;
          '4': saiprincipal:=true;  {encerra o programa}
          end;
    end;

end.
```

9.1.2. Unit Globais

```
Unit Globais;

interface

{ ================================================      = }
{ =          Declaracao de Constantes                   = }
{ ================================================      = }


Const
      SecLimite =0;{ End. secundario da CAD12/36 - Reg. de Limite}
      SecPonteiro = 1;{ End. secundario da CAD12/36 - Ponteiro Memoria}
      SecComAD = 2;{ End. secundario da CAD12/36 - Comando Conv. A/D }
      SecRM = 3;{ End. secundario da CAD12/36 - Reg. de Modo}
      SecMemoria = 4;{ End. secundario da CAD12/36 - Escrita Memoria }
      SecAutoCal= 6;{ End. secundario da CAD12/36 - Auto Calibracao }

      NMaxConvBuff = 10000; { Numero maximo de pontos no buffer}
      NMAxPoint= 1; { Numero de ponteiros para buffer de dados (Valor+1)}

      cGanho : array [0..3] of real = (1,2,5,10);
      constTipo: array [0..4] of string = ('S1','S2','S3','S4','S5');

      deslocX = 100;

{ ================================================ }
{ =Declaracao das Variaveis = }
{ ================================================ }
type
      TpValAD =array [0..NMaxConvBuff ] of single; { Valor lido por canal }
      { no. Pontos(IndConv), Canal }
      pTpValAD = ^TpValAD;

type EntradaRec = record
      valor: real;
      status : boolean;
      tipo : integer;
      ganho: integer;
      id : string;
      end;

SaidaRec = record
      status: boolean;
```

```
    tempo : longInt;
    TempoON : longInt;
    valor : single;
    end;

  VetorEntrada = array [0..15] of EntradaRec;
  VetorSaida = array [0..1] of SaidaRec;


var
  DMA : integer;
  AddBase : word;{ Endereco base do conversor A/D}
  NCanais : integer; { Numero de canais a serem convertidos }
  VMaxLoc : array [0..15] of single; { Limite superior do A/D (por Canal) }
  VMinLoc : array [0..15] of single; { Limite inferior do A/D (por Canal) }
  DadoOK: boolean; { existe dado convertido }
  Cn: integer; { Canal em questao }
  CnGraf: array [0..7] of integer;
  VADGraf : array [0..17] of single; { 0..15 Canal do A/D ; 16 = Flash ; 17 =
  Rele }
  DadoGravado : boolean; { informa que houve pelo menos um dado
  gravado }

  flgTrtInt : boolean;
  flgErrInt : boolean;

  EndBase: word;{ Endereco base da placa CAD12/36 }
  CadCtr0: word;{ Reg. Contador 0 da CAD12/36}
  CadCtr1: word;{ Reg. Contador 1 da CAD12/36}
  CadCtr2: word;{ Reg. Contador 2 da CAD12/36}
  CadModo: word;{ Reg. de Modo do Timer}
  CadStatus: word;{ Reg. Estado da CAD12/36}
  ByteA: word;{ Reg. Byte A do conversor A/D }
  ByteB: word;{ Reg. Byte B do conversor A/D }
  CadEnd : word;{ Registrador de Endereco}
  CadDado: word;{ Reg. dado de escr. memoria }
  IndConv: integer; { Indice para gravar no buffer de dados }
  buffer : integer; { buffer que sera carregado }
  BuffCheio: array [0..NMAxPoint] of boolean;{ Informa Buffer cheio }
  VetValAD : array [0..NMaxPoint] of pTpValAd;
  arq: file;
  NInt : integer; { interrupcao usada }
   status: byte;{ Status da CAd1236 }


  Time : LongInt;{ Tempo final da conversao }
  TimeCount: Real; { contador do tempo de conversao (ms) }
```

```
        AccCount : LongInt;{ Acumulador de tempo }
        Aquisita : boolean;{ indica inicio de aquisicao }

        AccTimeFlash : Single; { acumulador de tempo p/ Flash }
         CondFlash : boolean;{ Condicao do flash }

         AccTimeRele : Single; { acumulador de tempo p/ Rele }
        CondRele : boolean;{ Condicao do Rele }

         Atuadores : byte; { Situacao dos atuadores (bit : 0 = flash; 1 = rele) }
         AtuadoresOK : boolean;{}

        Freq1, {Frequencia escolhida pelo usu rio}
        FreqReal:Integer;{Frequencia calculada pelo programa}


function inttostr(valor : single; csDec: integer) : String;
procedure escreveDA(CnDA: integer; Valor: Single);

implementation

function inttostr(valor : single; csDec: integer) : String;
var S : String;
begin
        Str(Valor:0:csDec, S);
        IntToStr := S;
end;

procedure escreveDA(CnDA: integer; Valor: Single);
var
        iValor : integer;
        portDALo: word;
begin
        iValor := round(32768 * Valor / 10);{ converte valor real em
complemento de dois }
        portDaLo := AddBAse + 8 + (CnDA * 2);{ portDalo = port para bit menos
significativo }
        port[portDaLo] := Lo(iValor);{ Escreve 4 bits menos significativos }
        port[portDaLo + 1] := Hi(iValor);{ Escreve 8 bits mais significativos }
end;

var i : integer;

begin
        for i := 0 to NMaxPoint do
        begin
                New(VetValAD[i]);
```

```
        end;
end.
```

9.1.3. Unit Telas

```
Unit Telas;

INTERFACE

uses crt,EditTool,Globais;

Const LINHAFIM=25;
      LINHAINIC=1;
      COLUNAFIM=80;
      COLUNAINIC=1;
      LINHATRACO=4;
      LINHATRACO2=3;
      COLTITULO=3;
      LINHATIT=4;
      CorFrenteNormal=red;


Function  Centraliza(C1,C2:integer; titulo:string):string;
Procedure Tela;
Procedure WriteTitulo(titulo:string);
Procedure TelaSaida;
Procedure TelaConfSaida;
Procedure LimpaTela;
Procedure TelaConfigEntrada;
Procedure TelaFreq;
Procedure TelaHard;

IMPLEMENTATION

Function Centraliza(C1,C2:integer; titulo:string):string;

var i:integer;

begin
  for i:=1 to trunc((c2-c1-ord(titulo[0]))/2) do
     titulo:=' '+titulo;
  Centraliza:=titulo;
end;


Procedure TelaFreq;
begin
  TextColor(white);
```

```
  Gotoxy(3, 8);
  Write('Freq. de Aquisicao(Hz):');
  Gotoxy(3, 10);
  Write('Tempo de Aquisicao(ms):');
end;

Procedure TelaHard;
begin
  TextColor(white);
  Gotoxy(10, 8);
  Write('DMA:');
  Gotoxy(10, 10);
  Write('IRQ:');
  Gotoxy(10, 12);
  Write('End. de Base:');
end;


Procedure TelaConfigEntrada;
begin
  TextColor(white);
  Gotoxy(3, 8);
  Write('Canal 0');
  Gotoxy(3, 9);
  Write('Canal 1');
  Gotoxy(3, 10);
  Write('Canal 2');
  Gotoxy(3, 11);
  Write('Canal 3');
  Gotoxy(3, 12);
  Write('Canal 4');
  Gotoxy(3, 13);
  Write('Canal 5');
  Gotoxy(3, 14);
  Write('Canal 6');
  Gotoxy(3, 15);
  Write('Canal 7');
  Gotoxy(3, 16);
  Write('Canal 8');
  Gotoxy(3, 17);
  Write('Canal 9');
  Gotoxy(3, 18);
  Write('Canal 10');
  Gotoxy(3, 19);
  Write('Canal 11');
  Gotoxy(3, 20);
```

```
    Write('Canal 12');
    Gotoxy(3, 21);
    Write('Canal 13');
    Gotoxy(3, 22);
    Write('Canal 14');
    Gotoxy(3,23);
    Write('Canal 15');
    TextColor(CorFrenteNormal);
    Gotoxy(14, 8);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 9);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 10);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 11);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 12);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 13);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 14);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 15);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 16);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 17);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 18);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 19);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 20);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 21);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 22);
    Write('Stt:    Ganho:        Tipo:');
    Gotoxy(14, 23);
    Write('Stt:    Ganho:        Tipo:');
end;

Procedure Tela;

begin
  Box(COLUNAINIC,LINHAINIC,COLUNAFIM,LINHAFIM);
  Gotoxy(COLUNAINIC,LINHATRACO2);
```

```
  Write('³');
  Gotoxy(COLUNAFIM,LINHATRACO2);
  Write('³');

  gotoxy(COLUNAINIC+1,LINHAINIC+1);
  write('      SISEA  -  Laboratorio de Sistemas Energeticos Alternativos  ');
end;


Procedure WriteTitulo(titulo:string);

var i:integer;
   s:string;

begin
  s:='';
  for i:=1 to 78 do s:=s+' ';
  Gotoxy(COLUNAINIC+1,LINHATIT);
  Write(s);
  Gotoxy(COLUNAINIC+1,LINHATIT);
  Write(titulo);
end;


Procedure TelaSaida;

begin
  Box(COLUNAINIC,LINHAINIC,COLUNAFIM,LINHAFIM);
  Gotoxy(COLUNAINIC,LINHATRACO2);
  Write('³');
  Gotoxy(COLUNAFIM,LINHATRACO2);
  Write('³');

  gotoxy(COLUNAINIC+1,LINHAINIC+1);
  write('      SISEA  - Laboratorio de Sistemas energeticos Alternativos');
  Box(4,8,40,21);
  Box(43,8,77,21);
end;



Procedure TelaConfSaida;

Const COLUNA1=7;
    COLUNA2=46;
    LINHAI=12;
```

```
 begin
   Textcolor(white);
   Gotoxy(COLUNA1,LINHAI-2);
   Write('CANAL DE SAIDA 0');
   Gotoxy(COLUNA1,LINHAI);
   Textcolor(CorFrenteNormal);
   Write('Status (S/N) :');
   Gotoxy(COLUNA1,LINHAI+2);
   Write('Tempo 0 (ms) :');
   Gotoxy(COLUNA1,LINHAI+4);
   Write('Tempo ON (ms) :');
   Gotoxy(COLUNA1,LINHAI+6);
   Write('Valor (0 - 5V) :');
   Gotoxy(COLUNA2,LINHAI-2);
   Textcolor(white);
   Write('CANAL DE SAIDA 1');
   Gotoxy(COLUNA2,LINHAI);
   Textcolor(CorFrenteNormal);
   Write('Status (S/N) :');
   Gotoxy(COLUNA2,LINHAI+2);
   Write('Tempo 1 (ms) :');
   Gotoxy(COLUNA2,LINHAI+4);
   Write('Tempo ON (ms) :');
   Gotoxy(COLUNA2,LINHAI+6);
   Write('Valor (0 - 5V) :');
   {readkey;}
end;


Procedure LimpaTela;


begin
   Window(COLUNAINIC+1,LINHATRACO+1,COLUNAFIM-1,LINHAFIM-1);
   ClrScr;
   Window(1,1,80,25);
   LinhaHor( LINHAFIM, COLUNAINIC+1, COLUNAFIM-1 );
   LinhaVer( COLUNAFIM, LINHATRACO+1, LINHAFIM-1 );
end;

end.
```

9.1.4. Unit ArqTF

```
Unit Arqtf;

INTERFACE

uses crt,Globais,Telas;

Procedure LeConfig(
   arq: string;
   var CanalEntrada:VetorEntrada;
   var CanalSaida:VetorSaida;
   var LeuCerto: boolean;
   var FreqReal:integer;
   var Time:longint;
   var DMA:integer;
   var Nint:integer;
   var Addbase:word);

Procedure EscreveConfig(
   arq: string;
   CanalEntrada:VetorEntrada;
   CanalSaida:VetorSaida;
   FreqReal:integer;
   Time:longInt;
   DMA,Nint:integer;
   AddBase:word);


IMPLEMENTATION

Procedure LeConfig;

{*
 * Le os dados do eixo e da secao contidos no arquivo arq e coloca
 * estes dados em dados.
 * Se a leitura sucedeu-se corretamente LeuCerto retorna TRUE.
 *}

var i, j: integer;
   t: text;
   a: char;

begin
```

```
    assign(t,arq);
    {$I-}
    reset(t);
    {$I+}
    LeuCerto:=(IoResult=0);
    if LeuCerto then begin
      readln(t,FreqReal);
      readln(t,Time);
      readln(t,DMA);
      readln(t,Nint);
      readln(t,Addbase);
      for i:=0 to 15 do begin
         readln(t,a);
         CanalEntrada[i].status := upcase(a) = 'S';
         readln(t,CanalEntrada[i].ganho);
         readln(t,CanalEntrada[i].tipo);
      end;
      readln(t,a);
      CanalSaida[0].status := upcase(a) = 'S';
      readln(t,CanalSaida[0].tempo);
      readln(t,CanalSaida[0].tempoON);
      readln(t,CanalSaida[0].valor);
      readln(t,a);
      CanalSaida[1].status := upcase(a) = 'S';
      readln(t,CanalSaida[1].tempo);
      readln(t,CanalSaida[1].tempoON);
      readln(t,CanalSaida[1].valor);
    end;
    close(t); {fecha arq}
    LimpaTela;
end;


Procedure EscreveConfig;

{** Escreve dados no arquivo arq. *}

var i, j: integer;
    t:text;
    a:char;

begin
   assign(t,arq);
   rewrite(t);
   a:=' ';
   writeln(t,FreqReal);
   writeln(t,Time);
```

```
    writeln(t,DMA);
    writeln(t,Nint);
    writeln(t,AddBase);
    for i:=0 to 15 do begin
       if (CanalEntrada[i].status) then
         writeln(t, 'S')
       else
          writeln(t, 'N');
       writeln(t,CanalEntrada[i].ganho);
       writeln(t,CanalEntrada[i].tipo);
    end;
    if (CanalSaida[0].status) then
      writeln(t, 'S')
    else
       writeln(t, 'N');
    writeln(t,CanalSaida[0].tempo);
    writeln(t,CanalSaida[0].tempoON);
    writeln(t,CanalSaida[0].valor);
    if (CanalSaida[1].status) then
      writeln(t, 'S')
    else
       writeln(t, 'N');
    writeln(t,CanalSaida[1].tempo);
    writeln(t,CanalSaida[1].tempoON);
    writeln(t,CanalSaida[1].valor);
    close(t);
    LimpaTela;
  end;

  end.
```

9.1.5. Unit EditTool

```
Unit Arqtf;

INTERFACE

uses crt,Globais,Telas;

Procedure LeConfig(
  arq: string;
  var CanalEntrada:VetorEntrada;
  var CanalSaida:VetorSaida;
  var LeuCerto: boolean;
  var FreqReal:integer;
  var Time:longint;
  var DMA:integer;
  var Nint:integer;
  var Addbase:word);

Procedure EscreveConfig(
  arq: string;
  CanalEntrada:VetorEntrada;
  CanalSaida:VetorSaida;
  FreqReal:integer;
  Time:longInt;
  DMA,Nint:integer;
  AddBase:word);


IMPLEMENTATION

Procedure LeConfig;

{*
 * Le os dados do eixo e da secao contidos no arquivo arq e coloca
 * estes dados em dados.
 * Se a leitura sucedeu-se corretamente LeuCerto retorna TRUE.
 *}

var i, j: integer;
   t: text;
   a: char;

begin
```

```
  assign(t,arq);
  {$I-}
  reset(t);
  {$I+}
  LeuCerto:=(IoResult=0);
  if LeuCerto then begin
    readln(t,FreqReal);
    readln(t,Time);
    readln(t,DMA);
    readln(t,Nint);
    readln(t,Addbase);
    for i:=0 to 15 do begin
       readln(t,a);
       CanalEntrada[i].status := upcase(a) = 'S';
       readln(t,CanalEntrada[i].ganho);
       readln(t,CanalEntrada[i].tipo);
    end;
    readln(t,a);
    CanalSaida[0].status := upcase(a) = 'S';
    readln(t,CanalSaida[0].tempo);
    readln(t,CanalSaida[0].tempoON);
    readln(t,CanalSaida[0].valor);
    readln(t,a);
    CanalSaida[1].status := upcase(a) = 'S';
    readln(t,CanalSaida[1].tempo);
    readln(t,CanalSaida[1].tempoON);
    readln(t,CanalSaida[1].valor);
  end;
  close(t); {fecha arq}
  LimpaTela;
end;


Procedure EscreveConfig;

{*
 * Escreve dados no arquivo arq.
 *}

var i, j: integer;
   t:text;
   a:char;

begin
  assign(t,arq);
  rewrite(t);
  a:=' ';
```

```
    writeln(t,FreqReal);
    writeln(t,Time);
    writeln(t,DMA);
    writeln(t,Nint);
    writeln(t,AddBase);
    for i:=0 to 15 do begin
        if (CanalEntrada[i].status) then
          writeln(t, 'S')
        else
           writeln(t, 'N');
        writeln(t,CanalEntrada[i].ganho);
        writeln(t,CanalEntrada[i].tipo);
    end;
    if (CanalSaida[0].status) then
      writeln(t, 'S')
    else
        writeln(t, 'N');
    writeln(t,CanalSaida[0].tempo);
    writeln(t,CanalSaida[0].tempoON);
    writeln(t,CanalSaida[0].valor);
    if (CanalSaida[1].status) then
      writeln(t, 'S')
    else
        writeln(t, 'N');
    writeln(t,CanalSaida[1].tempo);
    writeln(t,CanalSaida[1].tempoON);
    writeln(t,CanalSaida[1].valor);
    close(t);
    LimpaTela;
end;

end.
```

9.1.6. Unit GetVal;

```
Unit GetVal;

Interface

uses crt,EditTool,Globais,textBas;


Procedure GetPrincipal(var b:char);
Procedure GetEnsaios(var ens:char);
Procedure GetConfigSaida (var CanalSaida: VetorSaida);
Procedure GetConfigEntrada (var CanalEntrada: VetorEntrada;
                    var TotalEntrada: integer);
Procedure GetArquivos(var getarq:char);
Procedure GetFreq(var freq:integer;var Time:longint);
Procedure GetHard(var DMA,Nint:integer;var AddBase:word);

IMPLEMENTATION


Procedure GetPrincipal(var b:char);

var vetlinhaP,vetcolunaP,vetlinhames,vetcolunames:vetinteiro;
   vetprompt,vetmensagem:vetstring;
   numeroprompts:integer;
   FlagWrapP,FlagEscP,FlagMesP:boolean;
   achar:char;

const COL=5;
    LIN=10;

begin
  vetlinhaP[1]:=LIN;
  vetlinhaP[2]:=LIN+2;
  vetlinhaP[3]:=LIN+4;
  vetlinhaP[4]:=LIN+6;

  vetlinhames[1]:=LIN;
  vetlinhames[2]:=LIN+2;
  vetlinhames[3]:=LIN+4;
  vetlinhames[4]:=LIN+6;

  vetcolunaP[1]:=COL;
  vetcolunaP[2]:=COL;
```

```
   vetcolunaP[3]:=COL;
   vetcolunaP[4]:=COL;

   vetcolunames[1]:=COL+20;
   vetcolunames[2]:=COL+20;
   vetcolunames[3]:=COL+20;
   vetcolunames[4]:=COL+20;

   vetprompt[1]:='1 - Ensaios';
   vetprompt[2]:='2 - Arquivos';
   vetprompt[3]:='3 - Conf. de Hard';
   vetprompt[4]:='4 - Fim';

   vetmensagem[1]:='--> Canais E/S, F. Aquis, Param. Ensaios, Executa';
   vetmensagem[2]:='--> Salvar/Recuperar Configuracao';
   vetmensagem[3]:='--> IRQ, DMA, Address';
   vetmensagem[4]:='--> Sai do Programa';

   numeroprompts:=4;

   FlagWrapP:=TRUE;
   FlagEscP:=TRUE;
   FlagMesP:=TRUE;

   achar:=Prompt(vetlinhaP,vetcolunaP,vetlinhames,vetcolunames,
            vetprompt,vetmensagem,numeroprompts,FlagWrapP,
            FlagEscP,FlagMesP);
   b:=achar;

end;

Procedure GetArquivos(var getarq:char);
var vetlinhaP,vetcolunaP,vetlinhames,vetcolunames:vetinteiro;
   vetprompt,vetmensagem:vetstring;
   numeroprompts:integer;
   FlagWrapP,FlagEscP,FlagMesP:boolean;
   achar:char;



const COL=5;
    LIN=10;

begin
  vetlinhaP[1]:=LIN;
  vetlinhaP[2]:=LIN+2;
  vetlinhaP[3]:=LIN+4;
```

```
   vetlinhames[1]:=LIN;
   vetlinhames[2]:=LIN+2;
   vetlinhames[3]:=LIN+4;

   vetcolunaP[1]:=COL;
   vetcolunaP[2]:=COL;
   vetcolunaP[3]:=COL;

   vetcolunames[1]:=COL+22;
   vetcolunames[2]:=COL+22;
   vetcolunames[3]:=COL+22;

   vetprompt[1]:='1 - Salvar ';
   vetprompt[2]:='2 - Recuperar ';
   vetprompt[3]:='3 - Voltar';

   vetmensagem[1]:='-----> Grava Configuracao de Arquivo';
   vetmensagem[2]:='-----> Le Configuracao de Arquivo';
   vetmensagem[3]:='-----> Volta para Tela Principal';

   numeroprompts:=3;

   FlagWrapP:=TRUE;
   FlagEscP:=TRUE;
   FlagMesP:=TRUE;

   achar:=Prompt(vetlinhaP,vetcolunaP,vetlinhames,vetcolunames,
            vetprompt,vetmensagem,numeroprompts,FlagWrapP,
            FlagEscP,FlagMesP);
  getarq:=achar;

end;



Procedure GetEnsaios(var ens:char);

var vetlinhaP,vetcolunaP,vetlinhames,vetcolunames:vetinteiro;
   vetprompt,vetmensagem:vetstring;
   numeroprompts:integer;
   FlagWrapP,FlagEscP,FlagMesP:boolean;
   achar:char;



const COL=5;
```

```
    LIN=10;

begin
  vetlinhaP[1]:=LIN;
  vetlinhaP[2]:=LIN+2;
  vetlinhaP[3]:=LIN+4;
  vetlinhaP[4]:=LIN+6;
  vetlinhaP[5]:=LIN+8;

  vetlinhames[1]:=LIN;
  vetlinhames[2]:=LIN+2;
  vetlinhames[3]:=LIN+4;
  vetlinhames[4]:=LIN+6;
  vetlinhames[5]:=LIN+8;

  vetcolunaP[1]:=COL;
  vetcolunaP[2]:=COL;
  vetcolunaP[3]:=COL;
  vetcolunaP[4]:=COL;
  vetcolunaP[5]:=COL;

  vetcolunames[1]:=COL+22;
  vetcolunames[2]:=COL+22;
  vetcolunames[3]:=COL+22;
  vetcolunames[4]:=COL+22;
  vetcolunames[5]:=COL+22;

  vetprompt[1]:='1 - Entrada';
  vetprompt[2]:='2 - Sa¡da';
  vetprompt[3]:='3 - Param. de Ensaio';
  vetprompt[4]:='4 - Executar';
  vetprompt[5]:='5 - Voltar';

  vetmensagem[1]:='-----> Configuracao dos Canais de Entrada';
  vetmensagem[2]:='-----> Configuracao dos Canais de Sa¡da';
  vetmensagem[3]:='-----> Freq. de Aquis. / Tempo de Aquis.';
  vetmensagem[4]:='-----> Aquisicao de Dados ';
  vetmensagem[5]:='-----> Volta para Tela Principal';

  numeroprompts:=5;

  FlagWrapP:=TRUE;
  FlagEscP:=TRUE;
  FlagMesP:=TRUE;

  achar:=Prompt(vetlinhaP,vetcolunaP,vetlinhames,vetcolunames,
          vetprompt,vetmensagem,numeroprompts,FlagWrapP,
```

```
            FlagEscP,FlagMesP);
  ens:=achar;

end;


Procedure GetFreq;

Const COLUNA1=30;
    LINHAI=8;

var vettamanho,vetlinha,vetcoluna:vetinteiro;
   vetget:vetstring;
   vetrealstring:vetboolean;
   code,i,tempcont,numerogets:integer;
   FlagWrap,FlagEsc:boolean;
   titulo:string;
   auxreal:real;
Begin
  vetlinha[1]:=LINHAI;  {freq. de aquisicao}
  vetlinha[2]:=LINHAI + 2;  {tempo de aquisicao}
  vetcoluna[1]:=COLUNA1;
  vetcoluna[2]:=COLUNA1;
  vettamanho[1]:=8;
  vettamanho[2]:=8;
  vetrealstring[1]:=FALSE;
  vetrealstring[2]:=FALSE;
  str(FreqReal:0,vetget[1]);
  str(Time:0,vetget[2]);
  numerogets:=2;
  FlagWrap:=TRUE;
  Flagesc:=TRUE;
  Get(vetlinha,vetcoluna,vettamanho,vetget,vetrealstring,numerogets,
     FlagWrap,FlagEsc);
val(vetget[1],FreqReal,code);
  val(vetget[2],Time,code);

end;

Procedure GetHard;

Const COLUNA1=30;
    LINHAI=8;

var vettamanho,vetlinha,vetcoluna:vetinteiro;
   vetget:vetstring;
   vetrealstring:vetboolean;
```

```
    code,i,tempcont,numerogets:integer;
    FlagWrap,FlagEsc:boolean;
    titulo:string;
    auxreal:real;
Begin
  vetlinha[1]:=LINHAI;  {DMA}
  vetlinha[2]:=LINHAI + 2;  {IRQ}
  vetlinha[3]:=LINHAI + 4;  {Address}
  vetcoluna[1]:=COLUNA1;
  vetcoluna[2]:=COLUNA1;
  vetcoluna[3]:=COLUNA1;
  vettamanho[1]:=8;
  vettamanho[2]:=8;
  vettamanho[3]:=8;
  vetrealstring[1]:=FALSE;
  vetrealstring[2]:=FALSE;
  vetrealstring[3]:=FALSE;
  str(DMA:0,vetget[1]);
  str(NInt:0,vetget[2]);
  str(AddBase:0,vetget[3]);
  numerogets:=3;
  FlagWrap:=TRUE;
  Flagesc:=TRUE;
  Get(vetlinha,vetcoluna,vettamanho,vetget,vetrealstring,numerogets,
     FlagWrap,FlagEsc);
  val(vetget[1],DMA,code);
  val(vetget[2],NInt,code);
  val(vetget[3],AddBase,code);

end;



Procedure GetConfigSaida;

Const COLUNA : array [1..2] of integer = (25,63);
     LINHAI=12;

var vettamanho,vetlinha,vetcoluna:vetinteiro;
   vetget:vetstring;
   vetrealstring:vetboolean;
   code,i,numerogets:integer;
   FlagWrap,FlagEsc:boolean;
   titulo:string;
   auxreal:real;

begin
```

```
Vetlinha[1]:=LINHAI;
Vetlinha[2]:=LINHAI+2;
Vetlinha[3]:=LINHAI+4;
Vetlinha[4]:=LINHAI+6;
Vetlinha[5]:=LINHAI;
Vetlinha[6]:=LINHAI+2;
Vetlinha[7]:=LINHAI+4;
Vetlinha[8]:=LINHAI+6;

Vetcoluna[1]:=COLUNA[1];
Vetcoluna[2]:=COLUNA[1];
Vetcoluna[3]:=COLUNA[1];
Vetcoluna[4]:=COLUNA[1];
Vetcoluna[5]:=COLUNA[2];
Vetcoluna[6]:=COLUNA[2];
Vetcoluna[7]:=COLUNA[2];
Vetcoluna[8]:=COLUNA[2];

for i:=1 to 8 do
begin
  vettamanho[i]:=8;
  vetrealstring[i]:=TRUE;
end;

vettamanho[1]:=1;
vettamanho[5]:=1;

if CanalSaida[0].status then vetget[1] := 'S'
else vetget[1] := 'N';
str(CanalSaida[0].tempo:7,vetget[2]);
str(CanalSaida[0].tempoON:7,vetget[3]);
str(CanalSaida[0].valor:7:2,vetget[4]);
if CanalSaida[1].status then vetget[5] := 'S'
else vetget[5] := 'N';
str(CanalSaida[1].tempo:7,vetget[6]);
str(CanalSaida[1].tempoON:7,vetget[7]);
str(CanalSaida[1].valor:7:2,vetget[8]);

vetrealstring[1] := FALSE;
vetrealstring[5] := FALSE;

numerogets:=8;
FlagWrap:=TRUE;
Flagesc:=TRUE;

Get(vetlinha,vetcoluna,vettamanho,vetget,vetrealstring,numerogets,
```

```
      FlagWrap,FlagEsc);

   CanalSaida[0].status := (upcase(vetget[1,1]) = 'S');
   val(vetget[2],CanalSaida[0].tempo,code);
   val(vetget[3],CanalSaida[0].tempoON,code);
   val(vetget[4],CanalSaida[0].valor,code);
   CanalSaida[1].status := (upcase(vetget[5,1]) = 'S');
   val(vetget[6],CanalSaida[1].tempo,code);
   val(vetget[7],CanalSaida[1].tempoON,code);
   val(vetget[8],CanalSaida[1].valor,code);

end;

Procedure GetConfigEntrada;
Const
     coluna : array [1..4] of integer = (5,14,30,46);
     LINHAI=8;

var
  Canal : integer;
icoluna : integer;
  escr  : string;

procedure WriteColuna(iColuna : integer);
begin
   PoeCursor;
   TextColor(yellow);
   case iColuna of
   2: begin
       if CanalEntrada[Canal].status then
         escr := 'S'
       else
         escr := 'n';
       WrStr2(LINHAI + Canal,COLUNA[2] + 5,1,escr);
     end;
   3: WrSingle(LINHAI +
Canal,COLUNA[3],2,0,cGanho[CanalEntrada[Canal].ganho]);
   4: WrStr2(LINHAI +
Canal,COLUNA[4],2,constTipo[CanalEntrada[Canal].tipo]);
   end;
end;

procedure GetColuna(iColuna : integer ; UpDw : boolean);
begin
   case iColuna of
    2: begin
        CanalEntrada[Canal].status := not CanalEntrada[Canal].status
```

```
        end;
      3: begin
          if UpDw then
          begin
            inc(CanalEntrada[Canal].ganho);
            if CanalEntrada[Canal].ganho > 3 then
              CanalEntrada[Canal].ganho := 0;
          end
          else
          begin
            dec(CanalEntrada[Canal].ganho);
            if CanalEntrada[Canal].ganho < 0 then
              CanalEntrada[Canal].ganho := 3;
          end
        end;
      4: begin
          if UpDw then
          begin
            inc(CanalEntrada[Canal].Tipo);
            if CanalEntrada[Canal].Tipo > 4 then
              CanalEntrada[Canal].Tipo := 0;
          end
          else
          begin
            dec(CanalEntrada[Canal].Tipo);
            if CanalEntrada[Canal].Tipo < 0 then
              CanalEntrada[Canal].Tipo := 4;
          end
        end;
    end;
end;

begin
  TextColor(Yellow);
  for Canal := 0 to 15 do
 begin
   WriteColuna(2);
   WriteColuna(3);
   WriteColuna(4);
  end;
  Canal := 0;
icoluna := 2;
  repeat
    WriteColuna(iColuna);
    WaitKbd;
    case FncKey of
      Rarr : begin
```

```
            inc(icoluna);
            if icoluna > 4 then
              icoluna := 2;
          end;
    Larr : begin
            dec(icoluna);
            if icoluna < 2 then
              icoluna := 4;
          end;
    Uarr : begin
            dec(canal);
            if canal < 0 then
              canal := 15;
          end;
    Darr : begin
            inc(canal);
            if canal > 15 then
              canal := 0;
          end;
     CR : GetColuna(icoluna,true);
    Null : case Upcase(CH) of
            '+' : GetColuna(icoluna,true);
            '-' : GetColuna(icoluna,false);
          end;
  end;
 until FncKey = Escape;
 FncKey := Null;
end;


end.
```

9.1.7. Unit CAD1236

```
Unit CAD1236;

Interface
  procedure EscreveRegSecundario (EndReg, Dado: byte);
  procedure Fim_CAd1236;
  procedure Init_Cad1236;
  procedure Init_Cad1236R4;
  procedure ProgramaMemoriaCanais(tipo : integer); (* tipo 0 = R3 e 1 = R4 *)
  procedure Flash;
  procedure Rele;

procedure Le_Cad(Canal : integer);

Implementation

  Uses Dos,Crt,Globais;



{ ================================================ }
{ =       Rotina  IniciaEnderecoHardware        = }
{ ================================================ }

Procedure IniciaEnderecoHardware;

begin
  CadCtr0   := AddBase;
  CadCtr1   := AddBase + 1;
  CadCtr2   := AddBase + 2;
  CadModo   := AddBase + 3;
  CadStatus := AddBase + 3;
  ByteA     := AddBase + 4;
  ByteB     := AddBase + 5;
  CadEnd    := AddBase + 4;
  CadDado   := AddBase + 5;
end;


{ ================================================ }
{ =       Rotina  EscreveRegSecundario        = }
{ ================================================ }
```

```
Procedure EscreveRegSecundario (EndReg, Dado: byte);

begin
  Port [CadEnd]  := EndReg;
  Port [CadDado] := Dado;
end;


{ ================================================ }
{ =          Rotina  EscreveMemCanais          = }
{ ================================================ }

Procedure EscreveMemCanais (R4R3 : integer ; Posicao, Dado: byte);

begin
  EscreveRegSecundario (SecPonteiro, Posicao);     { Endereco da Memoria de
Canais }
  if R4R3 = 0 then
    EscreveRegSecundario (SecMemoria,  Dado)       { Canal e Ganho
correspondente  }
  else
    EscreveRegSecundario (SecMemoria,  Not Dado)       { Canal e Ganho
correspondente  }
end;


{ ================================================ }
{ =          Rotina  AutoCalibracao          = }
{ ================================================ }

Procedure AutoCalibracao;

begin
  EscreveRegSecundario (SecAutoCal, 0);
  Delay (800);
end;

{ ================================================ }
{ =             Rotina LimpaFIFO             = }
{ ================================================ }

Procedure LimpaFIFO;
  Var i, Dado: integer;

begin
  EscreveRegSecundario (5,0);
  for  i := 1 to 17  do
```

```
  begin
    Dado := Port [ByteA];
    Dado := Port [ByteB];
  end;
end;


{ ================================================ }
{ =          Rotina  ProgramaTimer           = }
{ ================================================ }

Procedure ProgramaTimer (Contador, Modo: byte; Valor: Real);
  Var EndCtr: word;
   ValorWord: word;

begin
  ValorWord := round(2000000 / valor);
  EndCtr := CadCtr0 + Contador;
  Port [CadModo] := Contador * 64 + 48 + Modo * 2;
  Port [EndCtr]  := Lo (ValorWord);
  Port [EndCtr]  := Hi (ValorWord);
end;


{ ================================================ }
{ =          Rotina  ProgramaMemoriaCanais      = }
{ ================================================ }

Procedure ProgramaMemoriaCanais(tipo : integer);

  const
    G1   = 7*16;            { constantes para programar o ganho }
    G2   = 6*16;
    G5   = 5*16;
    G100 = 3*16;
    Bipolar  = $80;
    Unipolar = $00;

  var i    : integer;
      Ganho: array [0..15] of byte;
      Polar: array [0..15] of byte;

begin
  EscreveRegSecundario (SecLimite, $0f);   { Carrega 15 no Reg. Limite }
  for i := 0 to 15 do
  begin
    polar[i] := bipolar;
```

```
   Ganho[i] := G1;
   EscreveMemCanais (Tipo, i, polar[i] + Ganho[i] + i);  { memoria 7 com canal
7, bipolar e ganho 1 }
  end;
  EscreveRegSecundario (SecLimite, NCanais - 1);    { Carrega 15 no Reg.
Limite }
end;


{ ================================================ }
{ =            Inicia CAD12/36                   = }
{ ================================================ }

procedure Init_Cad1236;
begin
  IniciaEnderecoHardware;          { Determina os enderecos de I/O da placa
CAD12/36 }
  EscreveRegSecundario (SecRM,0);    { Conversao ao final de burst,inicio em
timer 0   }
                          { Programa Timers com um modo qualquer,    }
                          { de forma a ficarem em um estado definido }
  ProgramaTimer (0, 2,FreqReal);     { Programa Timer 0 no modo 2 }
  ProgramaTimer (1, 2,1);          { Programa Timer 1 no modo 2 }
  ProgramaTimer (2, 2,4000);        { Programa Timer 2 no modo 2 }

  AutoCalibracao;                 { Comando de Auto-Calibracao do Conversor
A/D    }
  ProgramaMemoriaCanais(0);         { Programa a memoria de canais
}
  LimpaFIFO;                     { Limpa a memoria FIFO da CAD12/36            }

end;

procedure Init_Cad1236R4;
begin
  IniciaEnderecoHardware;        { Determina os enderecos de I/O da placa
CAD12/36 }
  EscreveRegSecundario (SecRM,$0);{ Conversao ao final de burst,inicio em
timer 0   }

                         { Programa Timers com um modo qualquer,    }
                         { de forma a ficarem em um estado definido }
  ProgramaTimer (0, 1,4000);     { Programa Timer 0 no modo 1 }
  ProgramaTimer (1, 1,4000);     { Programa Timer 1 no modo 1 }
  ProgramaTimer (2, 1,4000);     { Programa Timer 2 no modo 1 }
```

```
  AutoCalibracao;              { Comando de Auto-Calibracao do Conversor A/D
}
  ProgramaMemoriaCanais(1);    { Programa a memoria de canais
}
  LimpaFIFO;                   { Limpa a memoria FIFO da CAD12/36          }

end;

procedure Le_Cad(Canal : integer);
begin
end;

procedure Fim_CAd1236;
begin
  EscreveRegSecundario (SecRM,0);
end;

procedure Flash;
begin
  if Aquisita or AtuadoresOK then
  begin
    if CondFlash then
      atuadores := atuadores or 1
    else
      atuadores := atuadores and $FE;
  end
  else
    atuadores := atuadores and $FE;
  port[AddBase + 7] := atuadores;
  if atuadores and 1 = 1 then
    EscreveDA(0,5)
  else
    EscreveDA(0,0)
end;

procedure Rele;
begin
  if Aquisita or AtuadoresOK then
  begin
    if CondRele then
      atuadores := atuadores or 2
    else
      atuadores := atuadores and $FD;
  end
  else
    atuadores := atuadores and $FD;
  port[AddBase + 7] := atuadores;
```

```
  if atuadores and 2 = 2 then
    EscreveDA(1,5)
  else
    EscreveDA(1,0)
end;


end.
```

9.1.8. Unit Graf_T;

```
Unit Graf_T;

interface
procedure VerSinal;

Implementation

Uses crt,Dos,Graph,Globais,Cad1236,TextBas,Arquivo;

var desloc        : integer;
   VMaxLocGraf    : array [0..7] of single;
   VMinLocGraf    : array [0..7] of single;
   PosXReal       : real;
   PosX           : integer;      { Posi‡„o X Atual }
   NGrafTotal     : integer;
   NGraf          : integer;
   GrafMarc       : integer;
   escr           : string;

const
    posEscrX = 550;   { posicao X das strings }

function CalculaPonto(Venge: single; XY : char; DesIY : integer):integer;
var val : integer;
begin
  case XY of
    'X' : begin
          Val := round(((Venge - (-VMinLocGraf[NGraf])) / (VMaxLocGraf[NGraf] -
(VMinLocGraf[NGraf]))) * GetMaxX);
          if Val > GetMaxX then
            Val := GetMaxX;
        end;
    'Y' : begin
            Val := round(((Venge - VMinLocGraf[NGraf]) / (VMaxLocGraf[NGraf] -
VMinLocGraf[NGraf])
                    * - ((GetMaxY - DesIY)/ NGrafTotal) + (GetMaxY - DesIY)/
NGrafTotal)
                    + (GetMaxY - DesIY)/ NGrafTotal * NGraf);
           if Val > GetMaxY - desIY then
             Val := round((GetMaxY - desIY) / NGrafTotal * NGraf);
          end;
  end;
 if Val < 0 then
```

```
    Val := 0;
  CalculaPonto := Val;
end;

procedure TracaLinhasBase(color : word);
var i : integer;
begin
  for NGraf := 0 to NgrafTotal - 1 do
  Repeat
    PutPixel(i,calculaPonto(0,'Y',desloc),color);
    PutPixel(i,calculaPonto(VMaxLoc[CnGraf[NGraf]] / 2,'Y',desloc),color);
    PutPixel(i,calculaPonto(VMinLoc[CnGraf[NGraf]] / 2,'Y',desloc),color);
    inc(i,5)
  until i >= (GetMaxX - deslocX);
end;

procedure MarcaCanal(Cor : word);
begin
  SetColor(cor);
  escr := 'Canal ' + inttostr(CnGraf[Ngraf],0);
  OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
end;

procedure VerSinal;
var
  LocalPixelAntY : array [0..639,0..7] of integer;   (* Posi‡„o Y do pixel anterior
*)
  Cor          : array [0..639,0..7] of word;
  PosY         : integer;                  (* Posicao Y Atual *)
  VGraf        : single;                   (* Valor do A/D *)
  i,j          : integer;
  GD,GM        : integer;
  VMax         : single;
  VMin         : single;
  Fim          : boolean;
  maxBloc      : integer;
  index        : single;
  GCond        : string;
  CnGra        : string;
  VOffSetGraf  : array [0..7] of single;
  divisor      : integer;
  Varredura    : single;
  posxAnt      : integer;
  CnTemp       : integer;

begin
  NGrafTotal := 1;
```

```
  for NGraf := 0 to 7 do
  begin
    VMaxLocGraf[NGraf]    := VMaxLoc[CnGraf[NGraf]];
    VMinLocGraf[Ngraf]    := VMinLoc[CnGraf[Ngraf]];
  end;
  divisor       := 1;
  Varredura     := 0.4;
  for NGraf := 0 to 7 do
    VOffSetGraf[NGraf] := 0;
  desloc := 70;
  fim := false;
  DetectGraph(Gd,GM);
  InitGraph(Gd,GM,'');
  if GraphResult <> 0 then
  begin
    erro(506);
    exit;
  end;
  posXReal := 1;
  i := 0;
  Rectangle(0,0,GetMaxX,GetMaxY);
  Line(GetMaxX - deslocX,0,GetMaxX - deslocX,GetMaxY - desloc);
  Line(0,GetMaxY - desloc,GetMaxX - deslocX,GetMaxY - desloc);
  maxBloc := 6;
  for i := 1 to maxBloc do
  begin
{   Line ((GetMaxX - deslocX) div maxBloc * i,GetMaxY - desloc,
        (GetMaxX - deslocX) div maxBloc * i,GetMaxY);}
    rectangle (5 + ((GetMaxX - deslocX) div maxBloc * i) - (GetMaxX - deslocX)
div maxBloc, GetMaxY - (desloc - 5),
            (GetMaxX - deslocX) div maxBloc * i - 5, GetMaxY - (desloc div 2 +
2));
    SetFillStyle (1,red);
    FloodFill ((GetMaxX - deslocX) div maxBloc * i - 6, GetMaxY - (desloc div 2 +
3),White);
    rectangle (5 + ((GetMaxX - deslocX) div maxBloc * i) - (GetMaxX - deslocX)
div maxBloc, GetMaxY - (desloc div 2 - 2),
            (GetMaxX - deslocX) div maxBloc * i - 5, GetMaxY - 5);
    SetFillStyle (1,White);
    FloodFill ((GetMaxX - deslocX) div maxBloc * i - 6, GetMaxY - 6,White);
  end;
{  TracaLinhasBase(Lightblue);}
  SetTextStyle(0,HorizDir,2);

  SetColor(black);
  escr := 'F3';
```

```
  OutTextXY((GetMaxX - deslocX) div maxBloc * 1 - 60, GetMaxY - (desloc div
2 + 23),escr);
  escr := 'F4';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 2 - 60, GetMaxY - (desloc div
2 + 23),escr);
  escr := 'Home';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 3 - 75, GetMaxY - (desloc div
2 + 23),escr);
  escr := 'F7';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 4 - 60, GetMaxY - (desloc div
2 + 23),escr);
  escr := 'F8';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 5 - 60, GetMaxY - (desloc div
2 + 23),escr);
  escr := 'Esc';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 6 - 70, GetMaxY - (desloc div
2 + 23),escr);
  Wrstr2(22,10,10,escr);
  SetTextStyle(0,HorizDir,1);

  SetColor(black);
  {SetTextColor(White);}
  escr := 'OffSet';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 1 - 68, GetMaxY - 5 - 18,escr);
  escr := 'Congela';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 2 - 73, GetMaxY - 5 - 18,escr);
  escr := 'Zera';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 3 - 70, GetMaxY - 5 - 22,escr);
  escr := 'OffSet';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 3 - 70, GetMaxY - 5 - 14,escr);
  escr := '+ Ganho';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 4 - 73, GetMaxY - 5 - 18,escr);
  escr := '- Ganho';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 5 - 73, GetMaxY - 5 - 18,escr);
  escr := 'Sai';
  OutTextXY((GetMaxX - deslocX) div maxBloc * 6 - 62, GetMaxY - 5 - 18,escr);

  SetColor(White);
  for i := 0 to 639 do
   for j := 0 to 7 do
   begin
     LocalPixelAntY[i,j] := 0;
     Cor [i,j]          := White;
   end;
  SetTextStyle(DefaultFont, HorizDir, 1);
  SetColor(White);
  for NGraf := 0 to NGrafTotal - 1 do
```

```
begin
 Line(GetMaxX - deslocX + 1,CalculaPonto (-10,'Y',desloc),
     GetMaxX  - 1,CalculaPonto (-10,'Y',desloc));
  Line(1,CalculaPonto (-10,'Y',desloc),
      GetMaxX - deslocX - 1,CalculaPonto (-10,'Y',desloc));
  escr := 'Canal ' + inttostr(CnGraf[Ngraf],0);
  OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
end;
for NGraf := 0 to 7 do
  CnGraf[NGraf] := NGraf;
NGraf := 0;
MarcaCanal(LightBlue);
repeat
  begin
    SalvaDados;
    NGraf := 0;
    repeat
      SetColor(White);
      PutPixel(posX,LocalPixelAntY[posX,NGraf], Cor [PosX,NGraf]);
      PosY:= CalculaPonto (VADGraf[CnGraf[Ngraf]],'Y',desloc);
      LocalPixelAntY[posX,Ngraf] := PosY;
      Cor [PosX,NGraf] := GetPixel (posX,posY);
      PutPixel (PosX, PosY, Yellow);
      inc(Ngraf);
    until NGraf = NGrafTotal;
    posXReal := posXReal + FreqReal / 1000 * Varredura;
    if posXReal > GetMaxX - deslocX then
    PosXReal := 1;
    posX    := round(posXReal);
    if (GetKbd (Ch) and (FncKey in [F1,CtlF1,F3,F5,F6,F7,F8,F9,F10,Home,
                    CR,Escape,Larr,Rarr,Darr,Uarr])) then
    case FncKey of
      F1,CR: begin
             AccTimeFlash := 0;       { zera acumulador de tempo p/ Rele }
               CondFlash := false;    { Flash nao disparado }

             AccTimeRele  := 0;       { zera acumulador de tempo p/ Rele }
               CondRele := false;     { Rele nao disparado }

             Flash;
             Rele;

             Aquisita := true;
           end;
     CTlF1: AtuadoresOK := not AtuadoresOK;   { Mesmo sem aquisitar dados,
os
                                    atuadores estao trabalhando }
```

```
Darr : begin  { incrementa o grafico marcado }
        CnTemp := NGraf;
        NGraf := GrafMarc;
        MarcaCanal(White);
        inc(GrafMarc);
        if GrafMarc > NGrafTotal - 1 then
          GrafMarc := 0;
        NGraf := GrafMarc;
        MarcaCanal(LightBlue);
        NGraf := CnTemp;
      end;
Uarr : begin  { decrementa o grafico marcado }
        CnTemp := NGraf;
        NGraf := GrafMarc;
        MarcaCanal(White);
        dec(GrafMarc);
        if GrafMarc < 0  then
          GrafMarc := NGrafTotal - 1;
        NGraf := GrafMarc;
        MarcaCanal(LightBlue);
        NGraf := CnTemp;
      end;
Larr : begin  { decrementa o canal do grafico marcado }
        CnTemp := NGraf;
        NGraf := GrafMarc;
        SetColor(black);
        escr := 'Canal ' + inttostr(CnGraf[GrafMarc],0);
        OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
        dec(cnGraf[GrafMarc]);
        If CnGraf[GrafMarc] < 0
          then CnGraf[GrafMarc] := NCanais - 1;
        VMaxLocGraf[GrafMarc] := VMaxLoc[CnGraf[GrafMarc]];
        VMinLocGraf[GrafMarc] := VMinLoc[CnGraf[GrafMarc]];
        SetColor(LightBlue);
        escr := 'Canal ' + inttostr(CnGraf[GrafMarc],0);
        OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
        NGraf := CnTemp;
      end;
Rarr : begin  { incrementa o canal do grafico marcado }
        CnTemp := NGraf;
        NGraf := GrafMarc;
        SetColor(black);
        escr := 'Canal ' + inttostr(CnGraf[GrafMarc],0);
        OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
        inc(cnGraf[GrafMarc]);
        If CnGraf[GrafMarc] > NCanais - 1
          then CnGraf[GrafMarc] := 0;
```

```
          VMaxLocGraf[GrafMarc] := VMaxLoc[CnGraf[GrafMarc]];
          VMinLocGraf[GrafMarc] := VMinLoc[CnGraf[GrafMarc]];
          SetColor(LightBlue);
          escr := 'Canal ' + inttostr(CnGraf[GrafMarc],0);
          OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
          NGraf := CnTemp;
        end;
      F5 : begin   { aumenta o numero de graficos em tela (Maximo 8)}
          if NGrafTotal < 8 then
          begin
           CnTemp := NGraf;
           NGraf := GrafMarc;
           MarcaCanal(White);
           NGraf := CnTemp;
{           TracaLinhasBase(Black);}
           SetColor(Black);
           for NGraf := 0 to NGrafTotal - 1 do
           begin
            Line(1,CalculaPonto (-10,'Y',desloc),
                GetMaxX - deslocX - 1,CalculaPonto (-10,'Y',desloc));
            Line(GetMaxX - deslocX + 1,CalculaPonto (-10,'Y',desloc),
                GetMaxX  - 1,CalculaPonto (-10,'Y',desloc));
            escr := 'Canal ' + inttostr(CnGraf[Ngraf],0);
            OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
           end;
           for posX := 0 to GetMaxX - deslocX do
            for NGraf := 0 to NGrafTotal - 1 do
              PutPixel(posX,LocalPixelAntY[posX,NGraf], Cor [PosX,NGraf]);
            inc(NgrafTotal);
           posXReal := 1;
           SetColor(White);
           for NGraf := 0 to NGrafTotal - 1 do
           begin
            Line(1,CalculaPonto (-10,'Y',desloc),
                GetMaxX - deslocX - 1,CalculaPonto (-10,'Y',desloc));
            Line(GetMaxX - deslocX + 1,CalculaPonto (-10,'Y',desloc),
                GetMaxX  - 1,CalculaPonto (-10,'Y',desloc));
            escr := 'Canal ' + inttostr(CnGraf[Ngraf],0);
            OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
           end;
           for i := 0 to 639 do
            for j := 0 to 7 do
            begin
             LocalPixelAntY[i,j] := 0;
             Cor [i,j]          := White;
            end;
            NGraf := 0;
```

```
                GrafMarc := 0;
                MarcaCanal(LightBlue);
{               TracaLinhasBase(Lightblue);}
              end;
            end;
        F6 : begin   {diminui o numero de graficos em tela (Minimo 1)}
              if NGrafTotal > 1 then
              begin
               CnTemp := NGraf;
               NGraf := GrafMarc;
               MarcaCanal(White);
               NGraf := CnTemp;
{               TracaLinhasBase(black);}
               SetColor(black);
               for NGraf := 0 to NGrafTotal - 1 do
               begin
                Line(1,CalculaPonto (-10,'Y',desloc),
                    GetMaxX - deslocX - 1,CalculaPonto (-10,'Y',desloc));
                Line(GetMaxX - deslocX + 1,CalculaPonto (-10,'Y',desloc),
                    GetMaxX  - 1,CalculaPonto (-10,'Y',desloc));
                escr := 'Canal ' + inttostr(CnGraf[Ngraf],0);
                OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
               end;
               for posX := 0 to GetMaxX - deslocX do
                for NGraf := 0 to NGrafTotal - 1 do
                  PutPixel(posX,LocalPixelAntY[posX,NGraf], Cor [PosX,NGraf]);
                dec(NgrafTotal);
               posXReal := 1;
               SetColor(White);
               for NGraf := 0 to NGrafTotal - 1 do
               begin
                Line(1,CalculaPonto (-10,'Y',desloc),
                    GetMaxX - deslocX - 1,CalculaPonto (-10,'Y',desloc));
                Line(GetMaxX - deslocX + 1,CalculaPonto (-10,'Y',desloc),
                    GetMaxX  - 1,CalculaPonto (-10,'Y',desloc));
                escr := 'Canal ' + inttostr(CnGraf[Ngraf],0);
                OutTextXY(posEscrX,CalculaPonto (10,'Y',desloc) + 10,escr);
               end;
               for i := 0 to 639 do
                for j := 0 to 7 do
                begin
                 LocalPixelAntY[i,j] := 0;
                 Cor [i,j]           := White;
                end;
                NGraf := 0;
                GrafMarc := 0;
                MarcaCanal(LightBlue);
```

```
{              TracaLinhasBase(Lightblue);}
             end;
           end;
       F7 : begin   { aumenta o zoom }
             divisor := divisor * 2;
             if divisor > 128 then
               divisor := 128
             else
             begin
               Index := (VMaxLocGraf[GrafMarc] - VMinLocGraf[GrafMarc]) / 4;
               VMaxLocGraf[GrafMarc] := VMaxLocGraf[GrafMarc] - Index;
               VMinLocGraf[GrafMarc] := VMinLocGraf[GrafMarc] + Index;
             end;
           end;

       F8 : begin   { diminui o zoom }
             divisor := divisor div 2;
             if divisor < 1 then
               divisor := 1
             else
             begin
               VMaxLocGraf[GrafMarc] := VMaxLocGraf[GrafMarc] + Index;
               VMinLocGraf[GrafMarc] := VMinLocGraf[GrafMarc] - Index;
               Index := Index * 2;
             end;
           end;

       F9 : begin   { aumenta a varredura }
             if Varredura < 0.9 then
               Varredura := Varredura + 0.1;
           end;
      F10 : begin   { diminui a varredura }
             if Varredura > 0.2 then
               Varredura := Varredura - 0.1;
           end;

       F3 : begin   { insere Offset no grafico marcado }
             VMaxLocGraf[GrafMarc] := (VMaxLocGraf[GrafMarc] -
VOffSetGraf[GrafMarc]);
             VMinLocGraf[GrafMarc] := (VMinLocGraf[GrafMarc] -
VOffSetGraf[GrafMarc]);
             VOffSetGraf[GrafMarc] := VADGraf[CnGraf[GrafMarc]] -
(VMaxLocGraf[GrafMarc] + VMinLocGraf[GrafMarc]) / 2;
             VMaxLocGraf[GrafMarc] := (VMaxLocGraf[GrafMarc] +
VOffSetGraf[GrafMarc]);
             VMinLocGraf[GrafMarc] := (VMinLocGraf[GrafMarc] +
VOffSetGraf[GrafMarc]);
```

```
            end;
        Home : begin   { zera Offset no grafico marcado }
                VMaxLocGraf[GrafMarc] := (VMaxLocGraf[GrafMarc] -
VOffSetGraf[GrafMarc]);
                VMinLocGraf[GrafMarc] := (VMinLocGraf[GrafMarc] -
VOffSetGraf[GrafMarc]);
                VOffSetGraf[GrafMarc] := 0;
               end;
      Escape : fim := true;              { sai do modo grafico }
      end;
    end;
  until Fim or FlgErrInt or (TimeCount >= Time);
  CloseGraph;
  PoeCursor;
  FncKey := Null;
end;

end.
```

9.1.9. Unit Arquivo;

```
Unit Arquivo;

interface
  procedure SalvaDadosParcial;
  procedure SalvaDados;
  procedure ConvReal_to_text;

implementation

Uses Globais;


procedure SalvaDados;
var
  BufferaGravar : integer;
begin
  if buffCheio[0] or buffCheio[1] then
  begin
    DadoGravado := true;
    if Buffer = 0 then
      bufferaGravar := 1
    else
      bufferaGravar := 0;

BlockWrite(arq,VetValAD[BufferaGravar]^,SizeOf(VetValAD[BufferaGravar]^));
    BuffCheio[BufferaGravar] := false;
  end;
end;

procedure SalvaDadosParcial;
var
    i           : integer;
    c           : integer;
    b           : integer;
    t           : single;
begin
  if IndConv <> 0 then
  begin
    DadoGravado := true;
    BlockWrite(arq,VetValAD[Buffer]^,indConv * 4);
  end;
end;
```

```
procedure ConvReal_to_text;
var arqText : text;
    Linha   : integer;
    tempo   : single;
    Valor   : array [0..17] of single;
    Canal   : integer;
  arqSingle : file of single;
  ponto     : integer;
  NLeit     : word;
  AccLeit   : integer;
begin
  if Not DadoGravado then
    exit;
  AccLeit := 0;
  ponto := 0;
  assign(arqText,'Teste.TXT');
  assign (arqSingle,'teste.Dat');
  reset(arqSingle);
  Rewrite(ArqText);
  linha := 0;
  tempo := 1;
  for Canal := 0 to NCanais - 1 + 2 do
    Read(arqSingle,Valor[Canal]);
  repeat
    Write(arqText,Tempo * 1/FreqReal * 1000 :0:2,' ms     ');
    for Canal := 0 to NCanais - 1 + 2 do
    begin
      Read(arqSingle,VetValAD[Buffer]^[ponto]);
      Write(arqText,VetValAD[Buffer]^[ponto]:0:4,'   ');
    end;
    tempo := tempo + 1;
    WriteLn(arqText);
  until eof(arqSingle);
  Close(ArqSingle);
  Close(ArqText);
end;


end.
```

9.1.10. Unit Err_Str;

```
unit ErrStr;

(* ---------------------- Descricao da Interface ------------------- *)

interface

function ErrorString (N: integer): string;


(* ---------------------- Descricao da Implementacao ------------------- *)

implementation

function ErrorString (N: integer): string;
var
  S: string[80];
begin
  Str (N:4, S);
  case N of
     -1: S:= 'Erro interno';
      1: S:= 'Arquivo ja existe';
      2: S:= 'Arquivo n„o encontrado';
      3: S:= 'Diret¢rio n„o encontrado';
      4: S:= 'Muitos arquivos abertos';
      5: S:= 'Acesso ao arquivo n„o permitido';
      6: S:= 'Erro na configuracao de Hardware';
      8: S:= 'Mem¢ria insuficiente';
     12: S:= 'Acesso ao arquivo inv lido';
     15: S:= 'Drive inv lido';
     16: S:= 'Nome do arquivo continua o mesmo? (S/N)';

     39: S:= 'Correla‡Æo abaixo da esperada';
     40: S:= 'Valor abaixo do permitido';
     41: S:= 'Valor acima  do permitido';
     43: S:= 'Valor n„o pode ser igual';
     44: S:= 'Numero de pontos deve ser maior que 1';
     45: S:= 'Impossˇvel acrescentar mais ganhos';
     46: S:= 'Impossˇvel retirar mais ganhos';
     47: S:= 'Impossˇvel retirar ganhos intermedi rios';
     48: S:= 'Todos os ganhos tem o mesmo nome? (S/N)';
     51: S:= 'Falta nome de arquivo';
     52: S:= 'N„o pode gerar PCX em modo texto';
```

```
    100: S:= 'Erro na leitura em disco';
    101: S:= 'Erro na grava‡„o em disco (disco cheio)';
    102: S:= 'Arquivo n„o especificado';
    103: S:= 'Arquivo n„o foi aberto';
    104: S:= 'Arquivo n„o foi aberto para leitura';
    105: S:= 'Arquivo n„o foi aberto para escrita';
    150: S:= 'Disco com selo de prote‡„o';
    151: S:= 'Unidade desconhecida';
    152: S:= 'Dispositivo n„o preparado';
    154: S:= 'Erro de CRC';
    156: S:= 'Erro na busca em disco';
    157: S:= 'Dispositivo desconhecido';
    158: S:= 'Setor n„o encontrado';
    159: S:= 'Falta papel na impressora';
    160: S:= 'Falha na escrita em dispositivo';
    161,
    149: S:= 'Falha na leitura em dispositivo';
    162: S:= 'Frequencia muito alta / Interrupcao';

    200: S:= 'Divis„o por zero';
    201: S:= 'Indice ou valor inv lido';
    202: S:= 'Pilha insuficiente';
    203: S:= 'Mem¢ria insuficiente';
    204: S:= 'Ponteiro inv lido';
    205: S:= 'Overflow de ponto flutuante';
    206: S:= 'Underflow de ponto flutuante';
    207: S:= 'Ponto flutuante inv lido';
    208: S:= 'Erro no gerenciador de overlay';
    209: S:= 'Erro na leitura de overlay';

    400: S:= 'Erro na For‡a de Deslizamento';
    401: S:= 'Erro na vel. de DESARME';
    402: S:= 'Erro na vel. de desarme de CHAVE SOBREVEL.';
    403: S:= 'Erro na vel. de desarme de CHAVE DESACEL.';
    404: S:= 'Erro na vel. de desarme de CHAVE RED. VEL.';
    500: S:= 'N„o h  dados para backup do LOG';
    501: S:= 'Erro no arquivo de LOG';
    503: S:= 'Arquivo de dados n„o ser  gravado';
    504: S:= 'Erro no NIP do condicionador/conversor (nome do arquivo de
dados)';
    505: S:= 'Erro no nome do arquivo de dados';
    506: S:= 'Erro nos arquivos Gr ficos';
    507: S:= 'Ganho deve ser diferente de zero ';
    508: S:= 'Dados inconsistentes';
    509: S:= 'Conversor A/D inexistente';
    510: S:= 'Conversor ou Condicionador devem ser diferentes de ------';
    550: S:= 'Fim da Impressao';
```

```
    551: S:= 'Erro na leitura do multimetro';
    552: S:= 'Conversor A/D n„o reconhecido';
    553: S:= 'Impossivel fazer calibra‡„o no modo atual';
    554: S:= 'Resistor de calibra‡Æo nÆo dispon¡vel para este condicionador';
    555: S:= 'Para este tipo de calibra‡„o deve existir ESD3201 ou PSC002';
    556: S:= 'Fim maior que inicio';
    557: S:= 'ImpressÆo do relatorio deve ser em papel timbrado   (tecle algo
p/ continuar)';
    600..799:
        S:= 'Arquivo de Configuracao/' + ErrorString (N-600);
    800..999:
        S:= 'Arquivo de Modelos/' + ErrorString (N-800);
   1000..1199:
       S:= 'Arquivo de Status/' + ErrorString (N-1000);
   1200..1399:
       S:= 'Arquivo de LOG/' + ErrorString (N-1200);
   1400..1599:
       S:= 'Impressora/' + ErrorString (N-1400);

   else
     S := 'Erro ' + S;
  end;
  ErrorString := S;
end;

begin
end.


function ErrorString (N: integer): string;
var
  S: string[80];
begin
  Str(N:4,S);
  case N of
    40  : ErrorString:= 'Valor menor do que o permitido';
    41  : ErrorString:= 'Valor MAIOR do que o permitido';
    100 : ErrorString:= 'Erro na For‡a de Deslizamento';
    101 : ErrorString:= 'Erro na vel. de DESARME';
    102 : ErrorString:= 'Erro na vel. de desarme de CHAVE SOBREVEL.';
    103 : ErrorString:= 'Erro na vel. de desarme de CHAVE DESACEL.';
    104 : ErrorString:= 'Erro na vel. de desarme de CHAVE RED. VEL.';
    200 : ErrorString:= 'Erro no arquivo de Rastreabilidade.';
    else
      ErrorString:= 'erro '+S;
  end;
end;
```

9.2. Data Sheet do Relé

EPRO

Características Técnicas Gerais

4 pinos (MINI) - Para placa de circuito Impresso.

Parâmetro		Min.	Tip.	Máx.	Unidade
Tensão de Disparo		03	-	32	VDC
Corrente de Comutação	3 à 32 VDC	10,0	35,0	50,0	mA
Tempo de Operação				1	Milisegundo
Obs.: sob cosulta fornecemos corrente mínima de 5ma					
Tensão de comutação (Valor RMS da rede AC)		90		440	V
Corrente de Comutação (25°C a 50°C)		-	-	03 = 30(6)	A
Pico de corrente admissível (16ms)		-	-	05 = 50(10)	A
Resolução da corrente de comutação em altas temperatura. I=I NOM (100-O.K)/100 O=T-50°C para T > 50°C	03A	-	1,3	1,4	%/°C
	05A	-	1,3	1,4	
Corrente RMS de Surto		6 I NOM.	8 I NOM.	10 I NOM.	-
Tensão de Isolação Entrada / Saída 2500V		1.5	2.0	3.0	KV RMS
Configuração		(Normalmente Aberto)			

Características Elétricas a 25°C

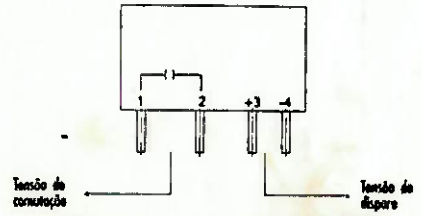
In(A)	Ih(mA)	Vcont(V)	Rcont(W)
3	30	2,2	0,753
5	60	2,1	0,263

Dimensões Físicas Monofásico p/ Placa de CI com 4 pinos

Fig nº 03

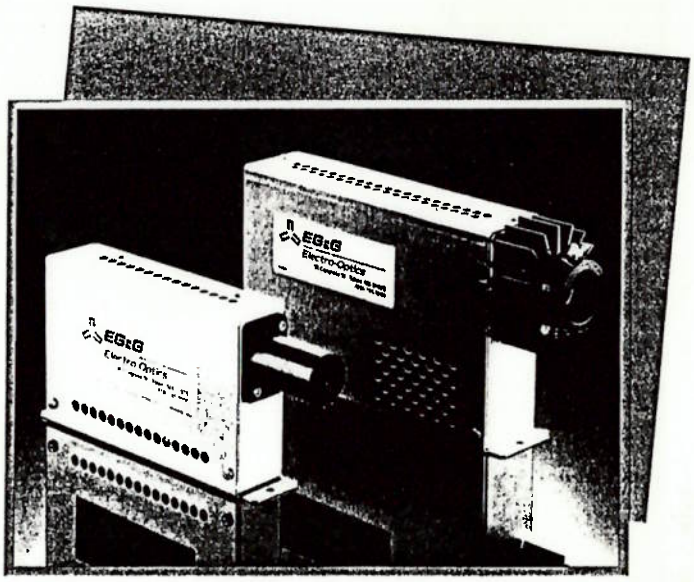
Diagrama de Ligação

4 pinos



9.3. Data Sheet do Flash

1100 Series
FlashPacs



LS1102 and LS 1130 FlashPacs

The 1100 Series FlashPacs have been designed to combine state-of-the-art circuitry and components into a packaged light source which provides microsecond-duration pulses of broadband light with exceptional arc stability. 1100 Series FlashPacs utilize the PS1110 and PS1120 variable output power supplies in conjunction with the appropriate LitePac trigger Module to operate several types of flashlamps.

A fully shielded steel case and EMI suppression circuitry combine to diminish the radiated and conducted noise normally associated with high peak discharge currents.

FlashPacs can be ordered with a wide range of discharge capacitance values and several types of the 1100 Series flashlamps. These systems make ideal sources of pulsed light for absorption analysis, immunoassay systems, fluorescent photometers, spectroradiometry, liquid chromatography, gas chromatography, colorimetry and ultraviolet applications.

Features

Exceptional arc stability

High radiant intensity

Continuous spectrum UV-VIS-IR

Long life

High repetition flash rates

Low heat radiation

Microsecond flash durations

Selection of flashlamp types

Selection of discharge capacitor

No warm up period

High efficiency output in the blue

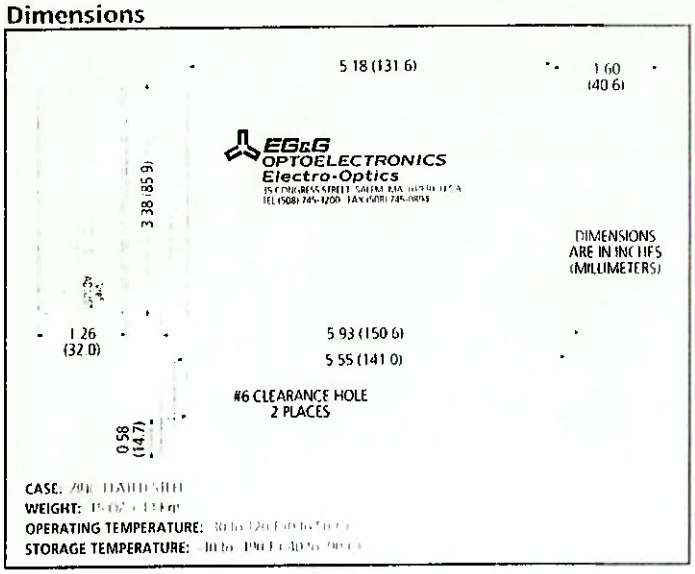
Simple fiber optic coupling

Small size

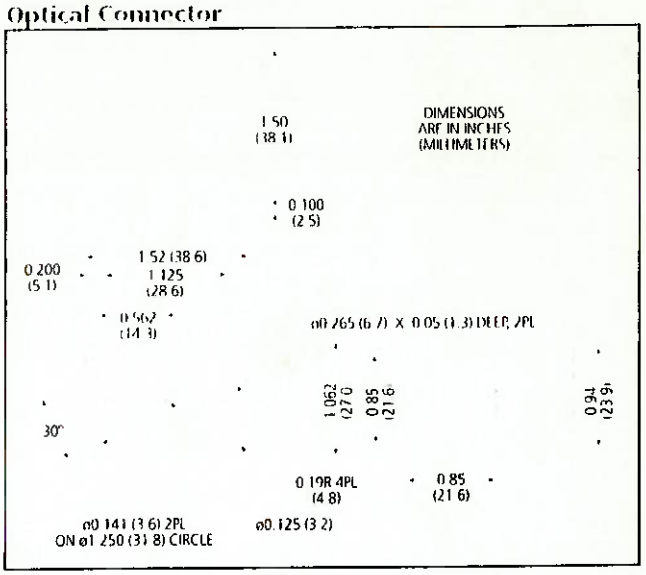
EG&G

LS 1102

Dimensions



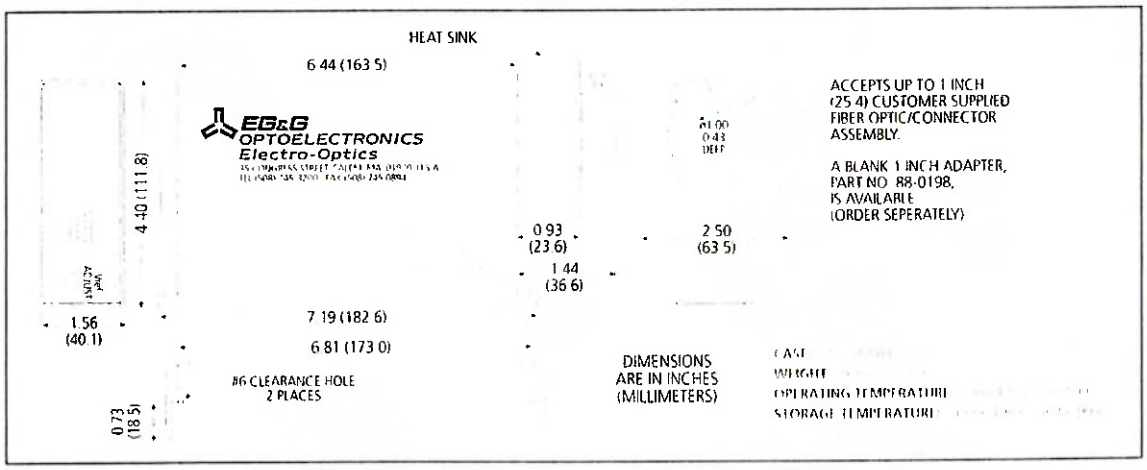
Optical Connector



Discharge Capacitors Available

Model Number	Discharge Capacitor	Maximum Input Energy Per Flash	Maximum Flash Rate @ 600 VDC
LS 1102–1	0.1 µfd	18 mJ	550 Hz
LS 1102–2	0.25 µfd	45 mJ	220 Hz
LS 1102–3	0.5 µfd	90 mJ	110 Hz
LS 1102–4	1.0 µfd	180 mJ	55 Hz
LS 1102–5	2.0 µfd	360 mJ	28 Hz

LS 1130



Discharge Capacitors Available

Model Number	Discharge Capacitor	Maximum Input Energy Per Flash	Maximum Flash Rate @ 1000 VDC
LS 1130-1	0.1 µfd	50 mJ	400 Hz
LS 1130-2	0.25 µfd	125 mJ	160 Hz
LS 1130-3	0.5 µfd	250 mJ	80 Hz
LS 1130-4	1.0 µfd	500 mJ	40 Hz

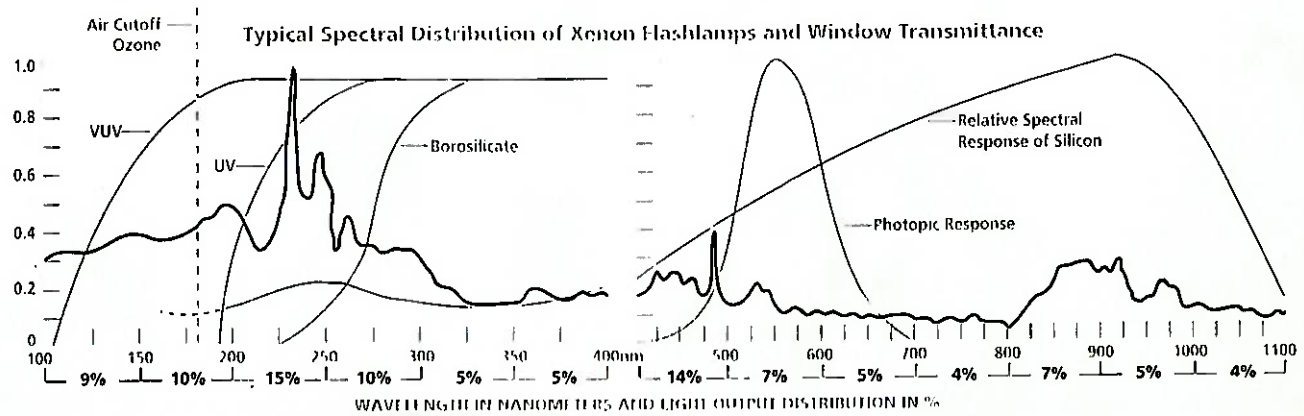
Flashlamps available for LS 1102 FlashPacs (order flashlamps separately)

Type	Arc Length (mm)	Spectral Distribution (nm)	Window Material	Stability (Intensity)	Stability (Spatial)	Jitter	Life
FX 1101	1.5 mm	225 – 1100+	Borosilicate	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1102	1.5 mm	190 – 1100+	UV	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1103	1.5 mm	120 – 1100+	VUV	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1104	3.0 mm	225 – 1100+	Borosilicate	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1105	3.0 mm	190 – 1100+	UV	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1106	3.0 mm	120 – 1100+	VUV	1%	<0.1 mm	<200 ns	>1 x 10^9

Flashlamps available for LS 1130 FlashPacs (order flashlamps separately)

Type	Arc Length (mm)	Spectral Distribution (nm)	Window Material	Stability (Intensity)	Stability (Spatial)	Jitter	Life
FX 1150	1.5 mm	225 – 1100+	Borosilicate	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1151	1.5 mm	190 – 1100+	UV	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1152	1.5 mm	120 – 1100+	VUV	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1153	3.0 mm	225 – 1100+	Borosilicate	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1154	3.0 mm	190 – 1100+	UV	1%	<0.1 mm	<200 ns	>1 x 10^9
FX 1155	3.0 mm	120 – 1100+	VUV	1%	<0.1 mm	<200 ns	>1 x 10^9

Note: The data shown in the above charts is typical for the products listed. Refer to the EG&G 1100 Series Flashlamp Data Sheet and Technical Brief for complete specifications.



Inputs

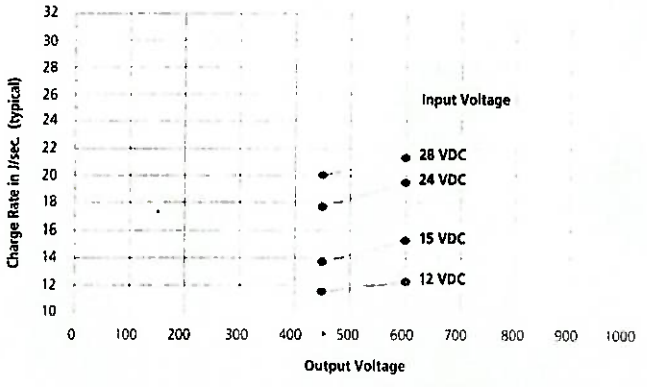
	LS 1102	LS 1130
Voltage (VDC)	11 – 28	15 – 28
DC Current (amps)	1.3 @ 12V	<1.2 @ 24V
Trigger	(1) TTL	(1) TTL
Vref (Vo:Vref = 100:1)	(2) 4.5 – 6.0	(2) 4 – 10
EMI Suppression	(3) YES	(4) YES

Discharge Parameters

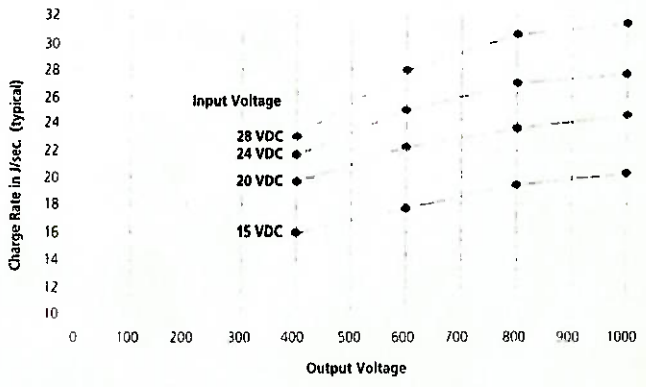
	LS 1102	LS 1130
Voltage (DC) (Vo)	(2) 450 – 600	(2) 400 – 1000
Power (watts)	10 max	20 max
Line Regulation	1%	1%
Ripple	(5) 0.5%	(5) 0.5%
Recharge Delay (µsec)	200	200
Flash Rate (Hz)	(see discharge curve dependent, see next page)	
Discharge Capacitance (µfd)	(see next page for values available)	

Notes:
(1) Opto-isolated, + 5V TTL compatible, 20–50 ma peak input, 10–20 µsec pulse width, leading edge trigger, internal series resistor 150 ohms.
(2) Output voltage level is controlled by a reference voltage (Vref) which may be adjusted internally or applied externally. The internal/external mode is determined by the position of a jumper on the power supply PC board. When set to the internal mode, voltage is set via a potentiometer which is accessible from the rear panel of the unit.
(3) Inductor and filter capacitor for power input (+). All inputs through a shielded 9-PIN "D" connector.
(4) Common-mode inductor and filter capacitor in power input. All inputs through a shielded 9-PIN "D" connector.
(5) Peak-to-peak maximum with 0.1 µfd discharge capacitor at maximum output.

LS 1102
Charge Rate Curve



LS 1130
Charge Rate Curve



Circuit Equations

E	=	$1/2\ CV^2$ • • • • • where:	E	=	Discharge energy (joules)
			C	=	Capacitance (microfarads)
			V	=	Discharge voltage (kilovolts)
P_{AVG}	=	$E\ F$ • • • • • • • • • •	P_{AVG}	=	Average power (watts)
			E	=	Discharge energy (joules)
			F	=	Flash rate (pulses per second)
I_{PK}	=	$V\sqrt{C/L}$ • • • • • • • •	I_{PK}	=	Peak discharge current (keep below 1000 amps)
			L	=	Circuit inductance (use 0.5μH for best approximation)
$t_{1/3}$	=	$\pi\sqrt{LC}$ • • • • • • • •	$t_{1/3}$	=	Pulse width at 1/3 peak.

Note: Peak currents should be kept below 1000 amps. Exceeding this limit could cause envelope fracture, excessive electrode wear and premature darkening.

Equations normally predict peak currents somewhat higher than what will actually occur as because they ignore circuit resistance.

CAUTION

Some glass flashlamps are under high internal pressure, and, if broken, could result in glass particles being blown into the face and hand areas. To prevent injury, wear suitable protective devices such as safety glasses and/or face mask and gloves.

Some types of pulsed lamps generate intense ultraviolet radiation which, if not properly shielded from personnel in the area, will cause burns to any exposed skin and especially to the eyes. Do not expose any skin area or the eyes to the direct or reflected radiation of an operating lamp. If you have to view an operating lamp, always use protective covering for exposed skin area and ultraviolet-attenuating goggles for the eyes.

EG&G
OPTOELECTRONICS
Electro-Optics

35 CONGRESS STREET, SALEM, MA 01970 U.S.A. TEL (508) 745-3200 FAX (508) 745-0894

Because of their modular construction, FlashPacs can be customized into other dimensional configurations. Requests for custom designs are encouraged as they are often a cost effective approach for OEM applications ..

Notes