ISaGRAF

Versão 3.4

GUIA DO USUÁRIO

CJ INTERNATIONAL

Impresso na França por CJ International.
3 Rue Hector Berlioz
F-38600 FONTAINE
Fone: 33 (0)4 76 26 87 30
Fax: 33 (0)4 76 26 87 39

Conteúdo

A. Guia do Usuário

A.1 Introdução

Este capítulo apresenta a instalação do Ambiente de Trabalho ISaGRAF. Também inclui um exemplo de um aplicativo ISaGRAF, dando ao usuário uma breve visão das suas principais características e permitindo a imediata utilização do ISaGRAF.

A.1.1 Instalação do ISaGRAF

Este capítulo apresenta a instalação do Ambiente de Trabalho ISaGRAF e como configurar o computador para a execução do aplicativo.

Hardware e software exigidos

O Ambiente de Trabalho ISaGRAF pode ser instalado em qualquer computador com as especificações mínimas para Windows Versão 3.1. Entretanto, o seguinte hardware é recomendado para a execução do aplicativo:

- Um computador pessoal utilizando um microprocessador 80486 ou superior (recomendado um processador Pentium)
- 8 megabytes de memória convencional e estendida (recomendado 16 megabytes)
- Uma unidade de disco 3.5" (1,44 megabyte)
- Um disco rígido com pelo menos 20 megabytes de espaço disponível
- Um adaptador gráfico VGA o SVGA e um monitor compatível
- Um mouse (necessário para execução de ferramentas gráficas)
- Uma porta paralela LPT1 (necessário para chave de proteção)

Antes de instalar o Ambiente de Trabalho ISaGRAF, o seguinte software já deve estar incluído no sistema:

- Windows Versão 3.1 executando no modo 386 expandido
- Windows 95
- Windows NT Versão 3.51 ou 4.00

Utilizando o programa de instalação

O Ambiente de Trabalho ISaGRAF é instalado através do INSTALL, o programa de instalação do ISaGRAF. Este programa copia o software ISaGRAF a partir do CD-ROM ISaGRAF, ou dos discos flexíveis, para o disco rígido do usuário. O INSTALL também adiciona o grupo "ISaGRAF" à janela Gerenciamento de Programas e cria um arquivo de inicialização denominado "ISA.ini" no subdiretório **EXE** instalado.

O INSTALL é um programa em ambiente Windows que deve ser executado a partir do Gerenciamento de Programas do Windows 3.1 ou do programa Executar do menu Iniciar do Windows 95. Para instalar o ISaGRAF, os seguintes passos devem ser seguidos:

- Insira o CD-ROM ISaGRAF ou o disco flexível nº 1 no drive correspondente

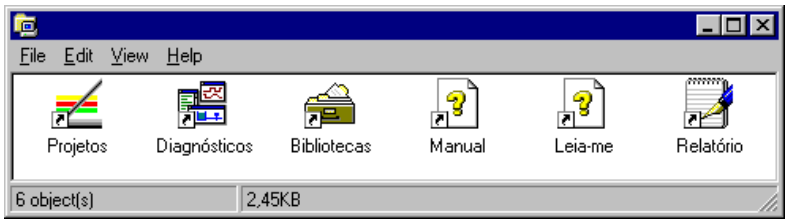
• A partir do Gerenciamento de Programas ou do menu Iniciar, execute “SETUP.EXE” no diretório raiz do CD-ROM ou “A:\INSTALL.EXE” no caso de disco flexível.
• Siga as instruções online para completar a instalação. É recomendado que o Ambiente de Trabalho ISaGRAF seja instalado em um novo diretório para evitar confusões entre os arquivos de diferentes versões do ISaGRAF.

O INSTALL perguntará se os seguintes componentes são necessários:

• Programas executáveis ISaGRAF
• Arquivos de informação online e de ajuda
• Bibliotecas padrão ISaGRAF
• Exemplo de um aplicativo ISaGRAF

É recomendado que, durante a instalação do ISaGRAF pela primeira vez, todos os componentes sejam incluídos. Entretanto, mais componentes podem ser adicionados posteriormente durante a reinstalação do Ambiente de Trabalho ISaGRAF.

O nome sugerido para o diretório principal do ISaGRAF é "\ISAWIN". Isto permite que o ISaGRAF para Windows seja parcialmente instalado no mesmo disco como uma versão do ISaGRAF para MS-DOS. Refira-se à seção "diretórios ISaGRAF " no capítulo "Técnicas avançadas " para maiores informações sobre a arquitetura dos diretórios ISaGRAF no disco. Uma vez copiados os arquivos ISaGRAF, o seguinte grupo é adicionado à sua Janela de Gerenciamento de Programas:



Os principais ícones do ISaGRAF são :

Projetos:.....Gerenciamento de Projetos
Bibliotecas: .....Gerenciamento da biblioteca
Manual:.....Informação online sobre o ISaGRAF
Diagnósticos: .....Ferramenta de diagnóstico para o usuário final
Leia Me:.....Informação sobre a nova versão do ISaGRAF
Relatório:.....Formato padrão de relatório de incidentes

Caso encontre algum problema, utilize o formato padrão de relatório de incidentes. Abra-o, preencha os itens solicitados e utilize o menu comando “Arquivo/Salvar como” para salvá-lo com um novo nome. Em seguida envie este arquivo para CJ International, utilizando Fax ou e-mail.

Atualização dos arquivos de sistema

Com a instalação finalizada, o arquivo CONFIG.SYS precisa ser atualizado antes de reinicializar o computador. O nome do caminho do diretório onde foi instalado o ISaGRAF não precisa ser inserido na variável de ambiente PATH.
O ISaGRAF não utiliza nenhuma variável de ambiente do MS-DOS. Entretanto, as seguintes instruções podem ser acrescentadas no arquivo CONFIG.SYS:

```
files=20
buffers=20
```

O Ambiente de Trabalho ISaGRAF utiliza uma porta serial para comunicar-se com o software ISaGRAF. A porta serial padrão para o ISaGRAF é a COM1. Se o mouse também utilizar uma porta serial, selecione a COM2 para o mouse, de modo que a especificação padrão COM1 será válida para quaisquer novos aplicativos ISaGRAF.

Após a atualização do arquivo CONFIG.SYS, é necessário reinicializar o computador para que as modificações sejam efetivadas.

⇒ **Importante para o usuário do Windows NT:**
Quando o Ambiente de Trabalho é utilizado em ambiente Windows NT 3.51 ou 4.00, a seguinte linha de instrução tem que ser inserida na seção [WS001] do arquivo ISA.ini no diretório \ISAWIN\EXE:

```
[WS001]
NT=1
Isa=C:\ISAWIN
IsaExe=C:\ISAWIN\EXE
IsaApl=C:\ISAWIN\APL1
IsaTmp=C:\ISAWIN\TMP
```

É absolutamente necessário para a comunicação RS.

A chave de proteção

Uma chave por hardware protege o software ISaGRAF contra cópias ilegais. Entretanto, a maioria das funções do Ambiente de Trabalho ISaGRAF estão disponíveis quando a chave ainda não está instalada. A chave de proteção também define a opção do Ambiente de Trabalho ISaGRAF, e define o tamanho máximo dos aplicativos de desenvolvimento. Quando a chave não estiver instalada ou instalada inadequadamente, algumas das funções do Ambiente de Trabalho ISaGRAF não serão executadas. Este é o comportamento NORMAL. Para assegurar que a chave esteja adequadamente instalada, selecione a opção "**Sobre...**" do menu "**Ajuda**" em qualquer janela ISaGRAF. A opção disponível do Ambiente de Trabalho ISaGRAF é apresentada.

A chave pode ser instalada em qualquer porta paralela do computador. Se este tiver mais de uma porta paralela, é preferível instalar a chave e a impressora em portas diferentes. Para algumas configurações PC/impressora, a chave pode não ser reconhecida quando sua saída estiver conectada a uma impressora “offline”. Neste caso, desconecte a impressora, ou inicialize-a no estado “online”, e reinicialize o Ambiente de Trabalho ISaGRAF.

Note que nenhuma chave é necessária para o Ambiente de Trabalho **ISaGRAF-32**.

⇒ **Importante para o usuário do Windows NT:**

Em sistemas Windows NT , o Driver Sentinel/Rainbow™ tem que ser instalado a fim de que a chave seja reconhecida. Um disquete separado é fornecido.

A.1.2 Utilização da informação online

A informação online é instalada com o Ambiente de Trabalho ISaGRAF, para os seguintes tópicos:

- referência de linguagens ISaGRAF
- Guia de usuário completo (para qualquer ferramenta ISaGRAF)
- Observação técnica para os elementos nas bibliotecas

A partir de qualquer janela ISaGRAF, selecione as opções do menu "**Ajuda**" para apresentar a informação online.

A.1.3 Um exemplo de aplicativo

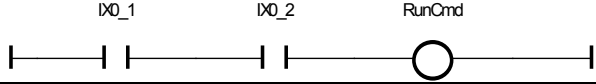
Este capítulo explica, passo a passo, todas as operações básicas necessárias para fazer, projetar, gerar e testar um breve mas completo aplicativo em multi-linguagem.

A seguir são apresentadas as especificações completas deste aplicativo, que utiliza as representações em linguagem LD e SFC:

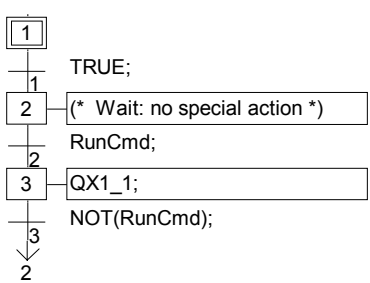
Variáveis booleanas:

IX0_1, IX0_2:	variáveis de entrada para comando de processo
RunCmd:	comando interno "executar/parar"
QX1_1:	variável de saída: estado do processo

Comando de Programa: Execução cíclica – linguagem LD
Avalia o comando interno "executar/parar"



Programa RunStop: Execução seqüencial – linguagem SFC
Controla o processo



Para criar o Ambiente de Trabalho ISaGRAF, execute o comando "**Projetos**", no grupo "ISaGRAF", a partir do menu Iniciar do Windows.

Criando o projeto

Crie o projeto (denominado "RunStop") utilizando o comando "**Novo**" do menu "Arquivo" ou o botão Novo. Na caixa de diálogo aberta:

Entre com o nome do projeto: "**RunStop**"
Selecione a configuração E/S: "**Sim_Boo**"
Pressione o botão "**OK**"..
O projeto já está criado.

Abrindo o projeto

Os programas do projeto são definidos pela abertura da janela de gerenciamento do programa ISaGRAF. Utilize o comando "**Abrir**" da janela Gerenciamento de Projetos ou clique duas vezes sobre o nome do projeto ou utilize o botão Editar.

Criando os programas

A janela Gerenciamento de Programas está aberta e vazia (não há programas definidos). O primeiro programa é criado utilizando o comando "Novo" no menu "**Arquivo**" ou o botão Novo. Na caixa de diálogo aberta:

Entre com o nome do programa: "**Command**".
Selecione a linguagem "**Quick LD**".
Selecione a seção "**Início**" .
Pressione o botão "**OK**" para criar o programa.

A mesma operação deve ser repetida para o segundo programa:
Utilize o comando "Novo" do menu "Arquivo", ou o botão Novo. Na caixa de diálogo aberta:

Entre com o nome do programa: "**RunStop**".
Selecione a linguagem "**SFC**".
Selecione a seção "**Sequencial**".
Pressione o botão "**OK**" para criar o programa.

Os programas já estão criados. Eles são apresentados na janela Gerenciamento de Programas.

Declarando as variáveis

Antes de entrar com os programas, a variável interna a ser utilizada na programação deve ser declarada. Isto é feito utilizando o comando "**Dicionário**" do menu "Arquivo" ou o botão Dicionário. As variáveis E/S são automaticamente declaradas quando o projeto é criado.

A janela dicionário agora está aberta. A partir do menu "**Arquivo**", do submenu "**Outros**", do submenu "**Variáveis Globais**" seguido do comando "**Booleanas**", selecione o dicionário Booleana Global. Os botões Objetos globais e Booleana podem ser utilizados produzindo o mesmo efeito.

O comando "**Novo**" do menu "**Editar**" é utilizado para criar novas variáveis booleanas. Você também pode utilizar o botão "**Inserir objetos**". Na caixa de diálogo aberta, entre com a descrição da variável interna:

nome: **RunCmd**

comentário: **Run/Stop command: internal**
atributo: Selecione o atributo "**Interna**"
Pressione o botão "**Armazenar**" : a variável é criada.
Pressione o botão "**Cancelar**" para sair da caixa de diálogo.

Finalmente, saia do editor de dicionário e salve as modificações introduzidas: Menu "**Arquivo**" - Comando "**Sair**". Clique sobre "**SIM**" para salvar as modificações.

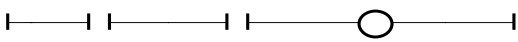
Editando o programa Quick LD

Para iniciar a edição do programa LD "Command", clique duas vezes sobre o seu nome na janela Gerenciamento de Programas ou utilize o botão Editar.

A janela ISaGRAF Quick LD Editor agora está aberta. Para aumentar a área de trabalho, redimensione a janela para utilizar todo o tamanho da tela.

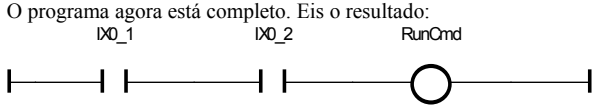
F2 F3 Pressione as teclas F2 e F3:

(* *)



Associe as variáveis aos símbolos LD: mova o cursor utilizando as teclas de seta. Posicione o cursor sobre cada símbolo e pressione a tecla Enter. A caixa de diálogo seleção de variáveis está aberta.
Para o primeiro contato, escreva na caixa seleção de variável: IX0_1 em seguida pressione Enter.
Para o segundo contato, escreva na caixa seleção de variável: IX0_2 em seguida pressione Enter.
Para a bobina, escreva na caixa de seleção de variável: RunCmd em seguida pressione Enter.

O programa agora está completo. Eis o resultado:



Saia do editor e salve as modificações introduzidas: Menu "**Arquivo**" - Comando "**Sair**". Clique sobre "**SIM**" para salvar as modificações.

Editando o programa SFC

Para iniciar a edição do programa "RunStop" SFC, clique duas vezes sobre o seu nome na janela Gerenciamento de Programas ou utilize o botão Editar.

A janela SFC Editor agora está aberta. Para aumentar a área de trabalho, redimensione a janela para utilizar todo o tamanho da tela:

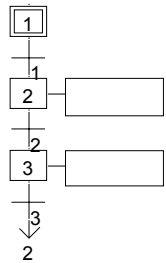
A etapa inicial já existe e está selecionada. Pressione a tecla seta “Para baixo” para selecionar a célula vazia após a etapa inicial (0,1)

F4 F3 Pressione F4 seguido de F3 para inserir uma etapa e uma transição.

F4 F3 Pressione F4 seguido de F3 para inserir mais uma etapa e mais uma transição.

F5 Pressione F5 para inserir um desvio para uma etapa, em seguida selecione GS2 como destino do desvio.

O gráfico agora está completo. Pressione o botão "Zoom" na barra de ferramentas para aumentar o tamanho das células e dar espaço para a apresentação das instruções de nível 2. Eis o gráfico:



Para entrar com a programação de transição "2", selecione-a utilizando as teclas de setas e pressione a tecla "Enter". A janela de programação de Nível 2 está aberta. Entre com a programação de nível 2 para a transição 2:

RunCmd;

^TAB Pressione as teclas "Ctrl + Tab" para retornar ao gráfico SFC, vá para a etapa 3 e pressione a tecla "Enter" para editar seu texto de nível 2:

QX1_1;

E faça o mesmo para inserir o texto da transição 3:

Not (RunCmd);

^F4 Pressione as teclas "Ctrl + F4" para fechar a janela de nível 2.

O programa SFC agora está completo. Saia do editor através do Menu "**Arquivo**" e do Comando "**Sair**", e salve as modificações inseridas clicando sobre "**SIM**".

Construindo o código do aplicativo

Para construir o código do aplicativo utilize o comando "**Compilar aplicação**" do menu "**Compilar**" ou o botão correspondente na Barra de Ferramentas a partir da janela Gerenciamento de Programas.

Quando a geração do código estiver terminada, uma caixa de diálogo será apresentada, propondo o fechamento da janela de compilação ou a continuação do trabalho com ela: Pressione a tecla "**Sair**".

Simulação

Utilize o botão correspondente na Barra de ferramentas ou o menu "**Depurar**" e o comando "**Simular**" a partir da janela Gerenciamento de Programas para executar o simulador ISaGRAF.

Quando a janela do simulador for apresentada, o aplicativo pode ser testado. Neste exemplo, ambas as entradas 1 e 2 (botões verdes) devem ser pressionados para executar o processo (LED vermelho de saída acende).

Feche a janela **Depurador** para sair da simulação: Menu "**Arquivo**" - Comando "**Sair**".

A.2 Gerenciamento de Projetos

Para executar a ferramenta de gerenciamento de projeto ISaGRAF, clique duas vezes sobre ícone ”Projetos", no grupo ISaGRAF. A janela "Gerenciamento de projetos" é então aberta. Um projeto corresponde a um loop PLC executado em um PLC destino. A janela superior contém a lista dos projetos existentes. O texto de descrição do projeto selecionado é exibido na janela mais abaixo.

Redimensionando janelas

Clique sobre o separador (splitter) entre a lista e o texto de descrição para redimensionar as janelas correspondentes. A janela de descrição não pode ser reduzida completamente. Sempre contém, pelo menos, uma linha de texto.

Inserindo separadores

Uma linha de separação pode ser inserida antes de qualquer nome de projeto. Isto permite se agrupar alguns projetos atrelados ao mesmo aplicativo no layout de lista. Utilize o comando "**Editar / Ligar/desligar separador**" para inserir ou apagar um separador antes do projeto selecionado.

Movendo projetos na lista

Para mover um projeto na lista, você primeiro tem que selecioná-lo (destacar) . Então clique sobre seu nome e arraste para outra posição na lista. Ao arrastar o projeto, uma seta pequena na margem esquerda indica onde será colocado. Você também pode utilizar o comando "**Mover para cima na lista”** ou **“Mover para baixo na lista**" do menu "**Editar**" para mover o projeto selecionado linha por linha. Note que se um separador é colocado antes do projeto selecionado, é movido com o projeto.

A.2.1 Criação e trabalhando com projetos

Os comandos do menu de gerenciamento de projeto são utilizados para criar novos projetos, editá-los (abrir) ou gerenciar projetos existentes.

Criando um novo projeto

Para criar um novo projeto, primeiro entre com o seu nome. Um projeto vazio é então criado. Uma configuração de E/S pode ser igualmente selecionada na criação do projeto. Esta configuração de E/S deve ser definida em biblioteca. Se uma configuração é escolhida, o ISaGRAF automaticamente cria as variáveis correspondentes e a conexão de E/S dentro do novo projeto. Quando você cria ou renomeia um projeto, você deve respeitar as seguintes regras:

- O nome não pode exceder **8** caracteres
- O primeiro caractere deve ser uma **letra**
- Os caracteres seguintes podem ser **letras**, **números** ou o caractere de sublinhar
- As letras maiúsculas e minúsculas não são diferenciadas

Quando um projeto é criado, utilize o comando "**Editar / Alterar comentário**" para entrar com o texto a ser apresentado com o nome do projeto na lista.

Editando a descrição do projeto

Utilize o comando "**Projeto / Descrição do projeto**" para entrar com o texto de descrição do projeto selecionado. Este documento identifica completamente o projeto dos outros na lista de projetos. Também pode ser utilizado para gravar qualquer observação ou informação complementar durante a vida útil do projeto.

Editando um projeto

O comando "**Arquivo / Abrir**" abre a janela Gerenciamento de Programa para o projeto selecionado. A partir desta janela, todos os utilitários de edição, de compilação e teste serão chamados. Também é possível clicar duas vezes sobre o nome do projeto para editá-lo.

O histórico de modificações

"O sistema ISaGRAF armazena todas as modificações relativas a um componente de um projeto em um arquivo de histórico. Cada modificação é identificada no histórico por um título, uma data e uma hora. O arquivo de histórico contém as últimas **500** modificações. Há um arquivo de histórico para cada projeto. O histórico de modificações para o projeto é o complemento dos arquivos de diário atrelados aos programas do projeto. O comando "**Projeto / Histórico**" permite que o usuário veja ou imprima o conteúdo do histórico de modificações para o projeto selecionado. O usuário pode selecionar um ou mais itens na lista principal e pressionar os seguintes botões:
OK .....fecha esta janela
Imprimir.....imprime o conteúdo da lista
[Apagar] Selecionados remove (apaga) as linhas selecionadas da lista
[Apagar] Todos.....remove a lista completa
Localizar .....procura um texto dentro da lista

A caixa de entrada acima do botão "**Localizar**" é utilizada para entrar com a seqüência de caracteres para a busca. Esta função não distingue entre letras maiúsculas e minúsculas. Quando a busca chega ao fim da lista, ela continua do início da lista até a posição de início.

Imprimindo um documento completo

O comando "**Projeto / Imprimir**" permite que o usuário elabore e imprima um documento completo sobre o projeto selecionado. Este documento pode agrupar todos os componentes (programa, variável, parâmetros ...) do projeto selecionado. Para elaborar um documento específico (não completo), o usuário só tem que definir o seu conteúdo.

Senha de proteção

O comando "**Projeto / Determinar senha**" permite que o usuário defina senhas de acesso às ferramentas e aos dados do projeto selecionado. Consulte a seção "**Senha de proteção**" no fim da primeira parte deste documento para maiores detalhes sobre o sistema de proteção os níveis de acesso. As senhas são relacionadas ao projeto selecionado e não têm nenhuma influência sobre os outros projetos e as bibliotecas ISaGRAF.

A.2.2 Trabalhando com diversos grupos de projetos

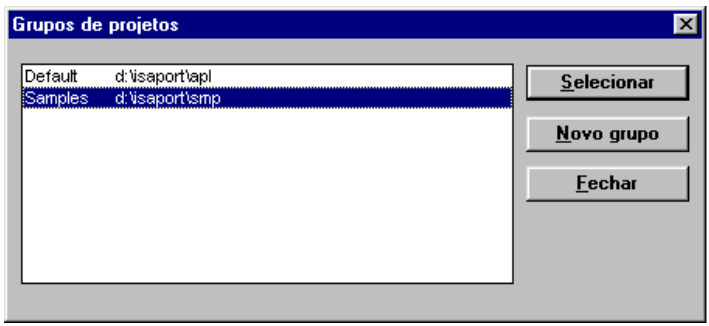
Um projeto ISaGRAF corresponde a um diretório no disco onde todos os arquivos do projeto são arquivados. Um "Grupo de Projeto" corresponde a uma lista de diretórios de projeto

agrupados sob o mesmo diretório raiz. Um grupo de projetos é identificado por um nome. Por padrão (“default”), o ISaGRAF cria os seguintes grupos:

"**Default**".....em "\ISAWIN\APL": sua área de trabalho
"**Samples**".....em "\ISAWIN\SMP": exemplos de aplicativos fornecidos com o ambiente de trabalho ISaGRAF

O nome do grupo de projeto atualmente selecionado é escrito na barra de ferramentas, perto do botão utilizado para selecionar um grupo de projeto:

Você também pode executar "Arquivo / Selecionar grupo de projetos" para selecionar um grupo existente ou criar um novo. A seguinte caixa de diálogo é aberta:



Selecione um grupo na lista e pressione "**Selecionar**" para ativá-lo na lista de gerenciamento de projeto. Você também pode clicar duas vezes sobre seu nome para selecioná-lo. Utilize o comando "**Novo grupo**" para criar um novo grupo. Este comando pode ser utilizado ou para determinar um nome de grupo para um diretório existente ou para criar um novo diretório.

Nota: Nenhum grupo pode ser selecionado ou criado se uma outra janela ISaGRAF (gerenciamento de programas, editores ...) estiver aberta.

A.2.3 Opções

Os comandos do menu "**Opções**" são utilizados para mostrar ou esconder a barra de ferramentas, selecionar a fonte do caractere utilizado no texto pelos editores de texto do ISaGRAF e configurar o modo de fechamento automático da janela de gerenciamento de Projetos.

Quando a opção "**Manter gerenciador de projetos aberto**" não está validada, a janela é automaticamente fechada quando um projeto é aberto.

A.2.4 Ferramentas

Os comandos do menu "**Ferramentas**" são utilizados para executar outros aplicativos ISaGRAF. O comando "**Ferramentas / Diretório**" executa o gerenciamento de arquivos para salvar ou restaurar projetos. O comando "**Ferramentas / Diretório / Dados comuns**" é utilizado para salvar ou restaurar arquivos comuns a todos os projetos (tais como palavras definidas comuns).

O comando "**Ferramentas / Bibliotecas**" executa o gerenciamento de biblioteca ISaGRAF em uma janela separada.

O comando "**Ferramentas / Importar programa IL**" pode ser utilizado para importar um projeto descrito como um único programa IL em um arquivo texto, respeitando o formato de troca definido por PLC Open.

A.3 Gerenciamento de programa

A janela “Programas” mostra os programas (também chamados de módulos ou unidades de programação) do aplicativo e agrupa dentro de seus menus os comandos disponíveis, para criar a arquitetura do projeto, executar os editores, compilar e depurar. Esta janela é o núcleo do ambiente de trabalho durante o desenvolvimento de um projeto. A janela Programas abre durante a execução do comando "Abrir" na janela Gerenciamento de projetos.

A.3.1 Os componentes de um projeto

Os componentes de um projeto são chamados **programas**. Um programa é uma entidade lógica que descreve uma parte da execução de controle . Variáveis globais (tais como variáveis de E/S) podem ser utilizadas por qualquer programa no aplicativo. Variáveis locais podem ser utilizadas por um único programa. Os programas são organizados dentro de uma **árvore hierárquica**, dividida em diferentes **seções lógicas**. A janela mostra os programas e os links entre eles. Os programas "**Top level**" (Nível mais alto) aparecem no lado esquerdo da árvore hierárquica.

- ***Programas de nível mais alto (“Top level”)***

Os programas de nível mais alto aparecem no lado esquerdo da árvore hierárquica. Os programas de nível mais alto das três primeiras seções estão sempre ativos e são executados na seguinte ordem, durante o ciclo de tempo de execução (varredura):

- (Leitura das entradas)
- Execução dos programas de nível mais alto da seção **INÍCIO**
- Execução dos programas de nível mais alto da seção **SEQUENCIAL**
- Execução dos programas de nível mais alto da seção **FIM**
- (Atualização das saídas)

Os programas das seções "**Início**" ou "**Fim**" descrevem as operações cíclicas. Não são dependentes do Tempo. Os programas da seção "**Sequencial**" descrevem operações seqüenciais nas quais a variável Tempo aparece explicitamente para distinguir operações básicas. Os programas principais da seção "**Início**" são sistematicamente executados no início de cada ciclo de tempo de execução. Os programas principais da seção "**Fim**" são sistematicamente executados no fim de cada ciclo de tempo de execução. Os programas principais da seção "**Sequencial**" são executados baseados nas regras **SFC** ou **FC** e devem ser escritos na linguagem **SFC** ou **FC**. Os programas das seções cíclicas não podem ser descritos em linguagem **SFC** ou **FC**. Qualquer programa de qualquer seção pode ter um ou mais **subprogramas**.

- ***Funções e blocos de funções***

Os programas da seção "**Função**" podem ser chamados por qualquer programa de qualquer seção no projeto. Uma função é um algorítmo que calcula um valor de saída em função de diversos valores de entrada. Uma função trabalha somente com variáveis voláteis, apagadas de uma chamada para outra. Isto implica que uma função não deve jamais chamar um bloco de função. Um programa da seção "**Função**" não pode ser descrito na linguagem **SFC** ou **FC**.

Ao contrário das funções, os "**Bloco de função**" associam um algorítmo e os dados estatísticos ocultos que são copiados (exemplo) pelo sistema a cada utilização diferente do bloco de função. Os programas da seção "**Bloco de função**" podem ser chamados por qualquer programa de qualquer seção no projeto. Não podem ser programados em linguagem **SFC** ou **FC**.

Subprogramas

Subprogramas são funções dedicadas a um programa principal (SFC, FC ou outro). Um subprograma só pode ser executado (chamado) por seu programa principal. Cada programa de cada seção pode ter um ou mais subprogramas. Qualquer linguagem, com exceção de **SFC** e **FC**, pode ser utilizada para descrever um subprograma.

SFC Secundário e subprogramas FC

Um **programa secundário SFC** é um programa paralelo que pode ser iniciado ou interrompido pelo seu programa principal. O programa principal e o programa secundário, ambos devem ser descritos na linguagem **SFC**.
Quando um programa principal inicia um programa secundário **SFC**, coloca uma **marca SFC** em cada etapa inicial do programa secundário. Quando um programa principal interrompe um programa secundário **SFC**, apaga todas as marcas existentes nas etapas do programa secundário.
Todo programa **FC** da seção seqüencial pode controlar outros subprogramas **FC**. A execução de um programa principal **FC** é bloqueada (espera) durante a execução de um subprograma FC, quer dizer, até que o fluxo no subprograma encontre o símbolo FIM. Não é possível que operações simultâneas sejam feitas no programa FC principal e um de seus subprogramas FC.

Links entre programas e subprogramas:

Os subprogramas e os programas secundários estão vinculados (link) ao programa principal por uma linha na árvore hierárquica. Um link entre um programa **SFC** e um programa secundário **SFC** é terminado por uma seta. Note que tal link representa operações **paralelas**.

Linguagens de programação

Cada programa é descrito em uma única **linguagem**. Esta linguagem, selecionada quando o programa é criado, não pode ser modificada. Entretanto, diagramas **FBD** podem incluir partes em **LD**, e diagramas **LD** podem incluir chamadas de blocos de funções. As linguagens gráficas disponíveis são: **SFC** (Sequential Function Chart, Fluxograma sequencial de funções ou GRAFCET), **FC** (Flow Chart ou fluxograma), **FBD** (Functional Block Diagram ou diagrama em blocos) e **LD** (Ladder Diagram ou diagrama de contatos). As linguagens de texto (literais) são: **ST** (Structured Text, Texto Estruturado) e **IL** (Instruction List, Lista de Instruções, Lista de Instruções). As linguagens **SFC** e **FC** são reservadas para os programas principal e secundário da seção seqüencial. A linguagem para cada programa é mostrada como um ícone ao lado do nome do programa na janela de Gerenciamento de Programa. A seguir, os ícones utilizados para representar as linguagens :

 SFC .....Sequential Function Chart, Fluxograma sequencial de funções (ou GRAFCET)
 FC.....Flow Chart (fluxograma)
 FBD.....Functional Block Diagram (diagrama em blocos)

LD .....Ladder Diagram (diagrama Ladder com o editor Quick LD)
ST.....Structured Text, Texto Estruturado
IL.....Instruction List, Lista de Instruções

A.3.2 Trabalhando com programas

O menu "**Arquivo**" agrupa todos os comandos utilizados para criar, atualizar ou modificar programas bem como os comandos de abertura dos editores de programas.

Criando um novo programa

A função "**Novo**" do menu "**Arquivo**" permite criar programas de nível mais alto, secundário ou subprogramas em cada seção de programa. A primeira informação a ser introduzida é o nome do nome programa de acordo com as seguintes regras :

- o comprimento máximo do nome não pode exceder **8** caracteres
- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes podem ser **letras**, **algarismos** ou o caractere '_' (sublinha)
- as letras maiúsculas e minúsculas não são diferenciadas

A seguir, selecione a linguagem de edição para o novo programa :

SFC.....Sequential Function Chart, Fluxograma sequencial de funções (GRAFCET)
FC .....Flow Chart (fluxograma)
FBD .....Functional Block Diagram (diagrama em blocos - pode incluir partes em LD)
LD.....Ladder Diagram (diagrama de contatos) editado com Quick LD
ST .....Structured Text, Texto Estruturado (linguagem de texto estruturada)
IL .....Instruction List, Lista de Instruções (lista de instruções)

Finalmente, selecione um estilo de execução para o programa :

Início.....programa de nível mais alto - seção "Início"
Sequencial .....programa de nível mais alto - seção "Sequencial"
Fim .....programa de nível mais alto - seção "Fim"
Function.....programa da seção "Função"
Function block.....programa da seção "Bloco de função"
Child of.....programa secundário SFC ou subprograma FC ou subprograma de um programa existente

Selecionando uma das cinco opções, o programa é colocado no nível superior da seção **Início**, **Fim**, **Sequencial**, **Função** ou **Bloco de função**. A seleção da última opção permite a criação de um programa secundário **SFC** ou um subprograma **FC** ou um subprograma. Lembre-se que o programa seqüencial de mais alto nível deve ser descrito com a linguagem **SFC** ou **FC** e que estas linguagens não podem ser utilizadas para programas cíclicos e seus subprogramas.

Inserindo comentários para cada programa

O ISaGRAF permite associar um texto descritivo para cada programa do projeto. Este comentário é mostrado ao lado do nome do programa, com caracteres menores. Utilize o comando "**Arquivo / Comentários do programa**" para entrar ou modificar o texto do comentário para o programa selecionado.

Editando o conteúdo de um programa

Este comando permite a modificação do conteúdo de um programa. A ferramenta ISaGRAF utilizada para a edição do programa depende da linguagem escolhida para este programa. A edição é realizada em janelas individuais. Também é possível editar diversos programas em paralelo em diferentes janelas. Pressionando a tecla **ENTER** é possível editar o programa selecionado. O usuário pode clicar duas vezes, com o mouse, sobre o nome do programa para editá-lo.

Editando o arquivo "Diário"

Um **arquivo de diário** é dedicado a cada programa. É um arquivo texto que contém todas as notas sobre as modificações feitas ao programa durante o seu desenvolvimento. O arquivo de diário pode ser editado, livremente modificado ou impresso a qualquer momento. Quando você deixa a edição da fonte de um programa que foi modificado, uma janela é automaticamente aberta para a entrada das anotações a armazenar no diário. Tais anotações são gravadas com a data e a hora no arquivo de diário.

O dicionário de variáveis

O comando "**Arquivo / Dicionário**" executa o editor dicionário, em que são declaradas as variáveis do projeto. As variáveis podem ser globais (conhecidas por todos os programas) ou local a um programa selecionado. O editor dicionário também pode ser utilizado para declarar **palavras definidas**, nomes alternativos que substituem um nome ou uma expressão nas fontes dos programas.

Parâmetros de uma função, subprograma ou bloco de função

O comando "**Arquivo / Parâmetros**" permite a definição dos parâmetros de chamada e retorno do subprograma, função ou bloco de função selecionado. Este comando não tem efeito se o programa principal da seção "**Início**" ou "**Fim**", ou um programa SFC é selecionado na janela Gerenciamento de Programa.
Subprogramas, funções ou blocos de função podem ter até 32 parâmetros (entrada ou saída). Uma função ou subprograma sempre tem um (e só um) parâmetro de retorno que deve ter o mesmo nome como a função em conformidade com as regras de escrita da linguagem ST .

A lista no lado esquerdo superior da janela mostra os parâmetros do subprograma, na ordem definida pelo modelo de chamada : primeiro os parâmetros de chamada e por último os parâmetros de retorno. A parte inferior da janela mostra a descrição detalhada do parâmetro selecionado na lista. Quaisquer tipos de dados ISaGRAF podem ser utilizados por um parâmetro. Os parâmetros de retorno devem ser os últimos da lista. As seguintes regras devem ser seguidas para a nomenclatura dos parâmetros :

- o comprimento do nome não pode exceder 16 caracteres
- o primeiro caractere deve ser uma letra
- os caracteres seguintes podem ser letras, algarismos ou o caractere de sublinha
- as letras maiúsculas e minúsculas não são diferenciadas

O comando "**Inserir**" é utilizado para inserir um novo parâmetro antes do parâmetro selecionado. O comando "**Apagar**" é utilizado para apagar o parâmetro selecionado. O comando "**Organizar**" automaticamente arruma (seleciona) os parâmetros de forma que os parâmetros de saída são colocados no fim da lista.

Movendo um programa na árvore hierárquica

O comando "**Renomear/Mover**" do menu "**Arquivo**" é utilizado para modificar o nome de um programa ou para movê-lo para uma outra seção da árvore hierárquica. Entretanto, a linguagem de descrição de um programa existente não pode ser modificada. Durante a execução deste comando, a mesma janela utilizada para a criação de programas é aberta e todos os campos são preenchidos com os valores atuais dos atributos do programa selecionado. O nome do programa pode ser modificado. Uma outra seção ou um novo programa principal pode ser selecionado para movimentar o programa na árvore hierárquica.

O comando "**Organizar programas**" do menu "**Arquivo**" é utilizado para dar uma ordem explícita entre os programas de mesmo nível hierárquico e mesmo programa principal. Se o programa selecionado é um de nível mais alto, o comando é utilizado para arrumar os programas de nível mais alto da seção selecionada. Se o programa selecionado está em nível inferior, o comando arruma somente os programas SFC secundários e subprogramas que tenham o mesmo programa principal. Quando a caixa de diálogo "**Organizar programas**" está aberta, selecione o programa e pressione o botão "**Para cima**" ou "**Para baixo**" para movê-lo na lista até a posição desejada.

Copiando programas

Para fazer uma cópia de um programa, selecione o programa fonte da lista de programas e execute o comando "**Arquivo / Copiar**" . Ao executar este comando, a mesma janela utilizada para criar programas é aberta, com todos os campos preenchidos com os valores dos atributos do programa selecionado. Entre com o nome do programa destino e sua localização nas seções da árvore hierárquica. Se o programa destino não existir, ele é criado no local especificado. Se o programa destino já existir, ele é sobrescrito. Todas as declarações locais e palavras definidas são copiadas com o programa. A linguagem de descrição do programa destino deve ser a mesma da utilizada para o programa de fonte. Pressione o botão "**OK**" para copiar o programa.

O comando "**Copiar para outro projeto**" do menu "**Arquivo**" copia o programa selecionado para outro projeto, com o mesmo nome. Os programas secundários SFC e subprogramas do programa selecionado podem ser copiados com ele. Os nomes do programa selecionado e seus programas secundários não devem ser utilizados no projeto destino. Em hipótese nenhuma, este comando pode ser utilizado para sobrescrever um programa existente. Todas as declarações locais e palavras definidas vinculadas são copiadas com os programas.

Apagando programas

Para apagar um programa, primeiro selecione-o na lista de programas e então execute o comando "**Arquivo / Apagar**" . Um programa que tenha programa secundário ou subprogramas não pode ser apagado. Para apagar um programa com programas secundários ou subprogramas, estes devem ser apagados em primeiro lugar. Todas as declarações locais e palavras definidas são apagadas com o programa.

Importando função ou bloco de função da biblioteca

O comando "**Ferramentas / Importar de biblioteca**" é utilizado para copiar uma função ou um bloco de função escrito em linguagem IEC descrito na biblioteca para a seção "**Função**" ou "**Bloco de função**" do projeto aberto. As variáveis locais e as palavras definidas vinculadas à função importada são copiadas com ela. Quando uma função é corretamente importada da biblioteca, esta pode ser colocada em outra seção ou outro local na árvore hierárquica, utilizando o comando "**Arquivo / Renomear/Move**". Para evitar conflitos de nome entre o projeto e a biblioteca, a função ou o bloco de função importado deve ser renomeado quando importado na área de projeto. Não esqueça de também renomear o parâmetro de retorno no caso de uma função.

Exportando função ou bloco de função da biblioteca

O comando "**Ferramentas / Exportar para biblioteca**" copia um programa da seção "**Função**" ou "**Bloco de função**" na biblioteca de funções ou blocos escritos em linguagem IEC. As variáveis locais e as palavras definidas vinculadas à função ou ao bloco exportado são igualmente exportados. A função ou o bloco exportado terá que ser recompilado (verificado) pelo Gerenciamento de Biblioteca ISaGRAF, para assegurar que pode ser utilizado em um ambiente de biblioteca. Funções e blocos de função da biblioteca não podem utilizar variáveis globais.

A.3.3 Execução das ferramentas de geração de código

Os comandos do menu "**Compilar**" são utilizados para executar o compilador e gerador de código e para introduzir as opções e outros parâmetros utilizados durante a geração do código do aplicativo. Refira-se ao capítulo "**Usando o gerador de código**" neste documento para informação adicional sobre estas ferramentas.

Gerar o código do aplicativo

O comando "**Compilar**" inicia o comando de geração do código de projeto. As opções para os destinos escolhidos destino devem ser corretamente fixadas antes da geração do código. Antes de gerar o código destino, todos os programas são verificados e os eventuais erros de sintaxe são detectados. O ISaGRAF inclui um compilador com incremento que não verifica os programas modificados depois de sua última verificação.

Verificar o programa selecionado

O comando "**Verificar**" permite que o usuário verifique a sintaxe do programa selecionado na lista . Quando um programa é verificado, e nenhum erro é detectado, ele não é verificado novamente durante a geração do código até que o seu conteúdo ou os objetos dependentes (variáveis, palavras definidas...) sejam modificados.

Simulando uma modificação

O comando "**Simular Modificação**" simula uma modificação de cada programa de modo que todos eles serão compilados novamente durante a próxima geração de código.

Opções de tempo de execução do aplicativo

Este comando abre uma caixa de diálogo na qual os parâmetros principais de tempo de execução são introduzidos para a execução do aplicativo. Isto inclui a programação da

temporização do ciclo, do gerenciamento de erro de tempo de execução, o modo de início e a implementação de hardware de variáveis retidas. Refira-se ao capítulo "Usando o Gerador de Código" neste documento para informação adicional sobre este comando.

Opções de compilador

Este comando é utilizado para configurar as opções usadas pelo Gerador Código ISaGRAF para produzir e aperfeiçoar o código destino. Refira-se ao capítulo "Usando o Gerador de Código" neste documento para informação adicional sobre este comando.

Definindo recursos

Um "**recurso**" é um conjunto de dados definidos pelo usuário (por exemplo um arquivo) que deve ser combinado com o código destino ISaGRAF e transferido com ele. Refira-se ao capítulo "Usando o Gerador de Código" neste documento para informação adicional sobre este comando.

A.3.4 Outras ferramentas ISaGRAF

O menu "Projeto" agrupa os comandos que executam as ferramentas ISaGRAF para o projeto selecionado. Refira-se aos capítulos correspondentes neste documento para informação adicional sobre estas ferramentas.

Conectando as variáveis de E/S

O comando "**Conexões de E/S**" executa o editor de conexão de variáveis E/S ISaGRAF . Esta ferramenta é utilizada para descrever a correspondência entre as variáveis de E/S declaradas no dicionário de projeto e o hardware de E/S correspondente.

Executando o editor de referências cruzadas

O comando "**Referências cruzadas**" permite que o usuário calcule, visualize ou imprima as referências cruzadas do projeto. As referências cruzadas mostram ao usuário todas as ocorrências de cada variável no código fonte dos programas, no projeto inteiro. Esta função é muito útil para detectar um acesso a uma variável ou qualquer recurso global, ou listar todas as ocorrências de uma variável global no código fonte.

Inserindo o descrição do projeto

O comando "**Descrição do projeto**" é utilizado para editar o texto da descrição do projeto. Este documento contém a identificação detalhada do projeto bem como as observações sobre suas especificações e sua vida útil. A descrição do projeto é aquela exibida na janela de Gerenciamento de Projeto.

Imprimindo um documento completo

O comando "**Imprimir documento do projeto**" permite que o usuário construa e imprima um documento completo sobre o projeto selecionado. Este documento pode agrupar qualquer componente (programa, variável, parâmetros...) do projeto selecionado. Para construir um documento personalizado (incompleto), o usuário só tem que definir seu conteúdo.

Histórico de modificações

Este comando abre uma caixa de diálogo onde é exibido o histórico de modificações para o projeto. Refira-se ao capítulo "Gerenciamento de Projetos" neste documento para informação adicional sobre este comando.

A.3.5 Criação de novos comandos no menu de Ferramentas

O ISaGRAF provê a maneira de inserir outros comandos no menu "**Ferramentas**". Os comandos suplementares devem ser descritos no arquivo texto "**\ISAWIN\COM\ISA.MNU**" . Você pode adicionar até 10 comandos. Comentários podem ser inseridos em qualquer linha, começando com o caractere "**;**" (ponto e vírgula). Cada comando é descrito em duas linhas de texto, de acordo com a seguinte sintaxe :

M=menu_string
C=*command_line*

O string de menu é o texto a ser exibido no menu "**Ferramentas**". A linha de comando descreve a chamada de um programa executável MS-DOS ou Windows executável, e pode ser completado com argumentos. Na linha de comando, você pode utilizar a seqüência "**%A**" para substituir o nome do projeto aberto, e "**%P**" para substituir o nome do programa selecionado. O exemplo a seguir executa a edição do programa selecionado com "Notepad" (Bloco de Notas) - (a ser utilizado unicamente com os programas ST e IL):

M=Edit with Notepad
C=Notepad.exe \isawin\apl\%A\%P.lsf

A.3.6 Simulação e depuração do aplicativo

Os comandos do menu "**Depurar**" são utilizados para executar o depurador gráfico ISaGRAF, para simular o aplicativo ou testá-lo em modo conectado.

Simulação

O comando "Simular" abre o depurador em modo de simulação. Neste modo, outra janela é aberta, chamada de simulador. Este comando é muito útil para testar qualquer aplicativo quando a máquina destino não está disponível. Iniciando o simulador fecha a janela de Gerenciamento de Programa. A janela de Gerenciamento de Programa é aberta novamente no modo depuração depois que ambas as janelas, depurador e simulação, estejam abertas. O simulador não pode ser iniciado se o código destino não tiver sido gerado. O simulador não pode ser iniciado quando janelas secundárias (editores, geração de código, conexão de E/S...) estiverem abertas. Cada uma delas deve ser fechada antes de executar este comando. Este comando também está disponível a partir de menus de editores ISaGRAF.

Depuração real

O comando "**Depurar**" abre a janela principal do depurador e fecha a janela de Gerenciamento de Programa. A janela de Gerenciamento de Programa é novamente aberta no modo depuração assim que comunicação seja estabelecida entre o depurador e o aplicativo destino. O depurador não pode ser iniciado se o código destino não tiver sido gerado. O depurador não pode ser iniciado quando janelas secundárias (editores, geração de código, conexão de E/S...) estiverem abertas. Cada uma delas deve ser fechada antes de

executar este comando. Este comando também está disponível a partir de menus de editores ISaGRAF.

Preparação da área de trabalho para depuração

O comando "**Depurar / Área de trabalho**" permite que você defina uma lista de documentos para a área de trabalho inicial. Tais documentos podem ser programas, gráficos "SpotLight", listas de variáveis. Gráficos e listas de diagramas de tempo de versões anteriores do ISaGRAF também são listadas com documentos de projeto. Documentos definidos na área de trabalho inicial são abertos automaticamente quando a monitorização de simulação ou online é executada.

A caixa de diálogo mostra, à esquerda, os documentos existentes do projeto e, à direita, os documentos selecionados para a área de trabalho inicial. Utilize os botões ">>" e "<<" para mover documentos de uma lista para outra. Cada projeto tem sua própria lista de documentos para a área de trabalho inicial.

Configuração de link

O comando "**Configuração do link**" permite que o usuário defina os parâmetros do link utilizado para a comunicação entre o depurador no PC host e o sistema ISaGRAF destino.
O "**Número da Estação escrava**" identifica o sistema ISaGRAF destino ou a tarefa destino no caso de uma implementação multi-aplicativo sobre a mesma máquina. O número escravo está compreendido entre **1** e **255**. Refira-se ao manual fornecido pelo fabricante do sistema destino para maiores detalhes sobre a implementação do ISaGRAF.
A "**Prota de comunicação**" identifica os meios de comunicação entre o ambiente de trabalho ISaGRAF e o destino. Ou pode ser o nome de uma porta serial, ou "**Ethernet**", para uma comunicação TCP-IP reservada utilizando o "Winsock" Versão 1.1.
O "**Time out**" é o tempo deixado para o sistema destino para suas operações de comunicação, entre o fim de uma pergunta emitida pelo depurador e o início da sua resposta dada pelo destino. Este tempo é expresso em **segundos**. O campo "Retries" é o número de tentativas automáticas que o depurador executa para uma operação de comunicação antes de detectar um erro de comunicação.

- ***Configuração de link serial***

  Quando uma porta de comunicação selecionada é uma porta serial RS232 (COM1.. 4), o botão "**Configurar**" permite o acesso a outros parâmetros de comunicação de links seriais.
  A taxa de transmissão, a paridade e o formato podem ser configurados. Quando a escolha do "**hardware**" é selecionada para o "**Controle de fluxo**", o ambiente de trabalho ISaGRAF controla as linhas CTS e DSR para habilitar o hardware de handshaking (verificação) durante a comutação.

- ***Configuração da conexão Ethernet***

  Quando a porta de comunicação selecionada é "Ethernet", o botão "**Configurar**" permite o acesso aos outros parâmetros da comunicação TCP-IP. Os campos "Endereço Internet" e "Número da porta" são reservados para a comunicação TCP-IP com a interface Winsock versão 1.1.
  O arquivo WINSOCK.DLL deve ser corretamente instalado no seu disco rígido. "**1100**" é o número da porta utilizada para o destino ISaGRAF .

A.4 Utilizando o editor SFC

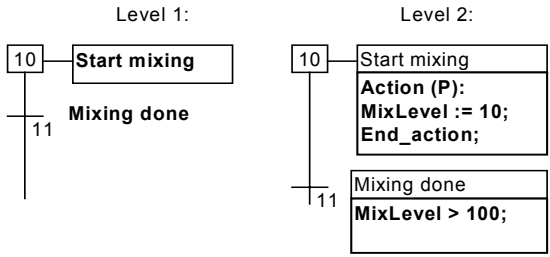
A linguagem SFC é utilizada para descrever as operações de um processo **seqüencial**. Utiliza uma única representação **gráfica** dos diferentes **estados** do processo e as condições que permitem a passagem de um estado para outro. Um programa SFC é introduzido utilizando o editor gráfico SFC do ISaGRAF. O SFC é uma linguagem de alto nível (padrão IEC 1131-3). As outras linguagens são normalmente utilizadas para descrever as **ações** dentro das **etapas** e as condições lógicas vinculadas às **transições**. O editor gráfico SFC do ISaGRAF permite que o usuário entre com programas SFC completos. Combina a capacidade de edição gráfica e de texto permitindo assim a entrada de ambos, o gráfico SFC e as ações e condições correspondentes.

A.4.1 Tópicos principais da linguagem SFC

A linguagem SFC é utilizada para representar processos seqüenciais. Divide o processo em um número de etapas conhecidas (situações estáveis), separadas por transições. Refira-se ao Manual de Referência das Linguagens ISaGRAF para mais informações sobre a linguagem SFC.
Os símbolos de um gráfico SFC são combinados por **linhas orientadas**. A orientação padrão de uma linha é **de cima para baixo**. A seguir, os componentes gráficos utilizados para a construção de um gráfico SFC :

 .....Etapa inicial
 .....Etapa
 .....Transição
 .....Desvio para a etapa
 .....Macro etapa
 .....Etapa de início de uma macro
 .....Etapa de término de uma macro

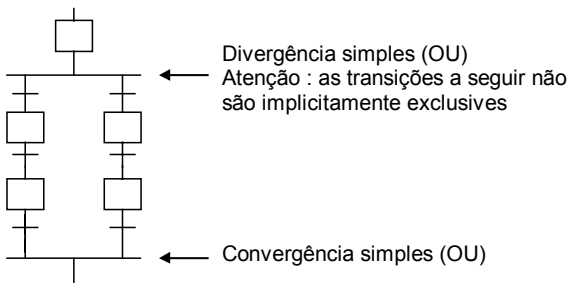
A programação SFC é normalmente decomposta em dois níveis. O **nível 1** mostra o gráfico, os números de referência e os comentários vinculados às etapas e às transições. O **nível 2** é a programação **ST** ou **IL** das ações dentro das etapas e das condições vinculadas às transições. As ações e as condições podem recorrer a **subprogramas** escritos em outras linguagens (**FBD**, **LD**, **ST** ou **IL**). A seguir, um exemplo de programação em nível 1 e nível 2 :



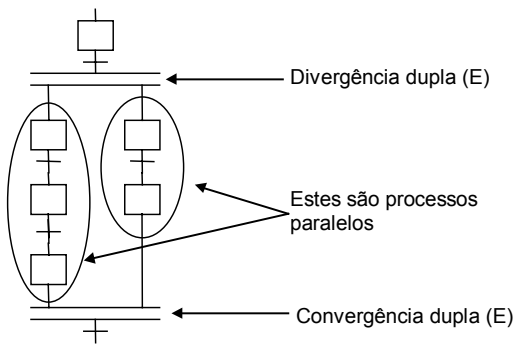
O nível 2 de uma etapa é introduzida através de um editor de texto. Pode incluir blocos de ação programados em ST ou IL. O nível 2 de uma transição pode ser introduzido em linguagem texto IL ou ST, ou com o editor Quick LD (diagrama de contatos).

Divergências e convergências

As divergências e convergências são utilizadas para representar as **múltiplas ligações** entre as etapas e as transições. As divergências e convergências simples representam diversas possibilidades (**inclusive**) na execução do processo.

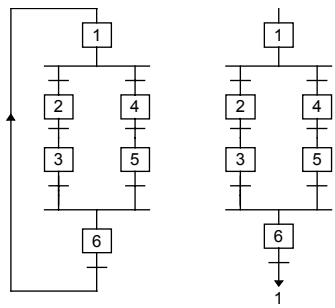


Divergências duplas representam processos **paralelos**.



Desviar para uma etapa

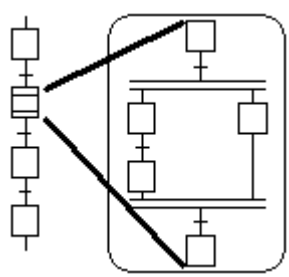
O editor SFC só permite a visualização das ligações traçadas de **cima** para **baixo**. Um **desvio** para uma etapa pode ser utilizado para representar uma ligação com uma parte anterior do gráfico. Por exemplo :



Um desvio para uma transição é proibido e deve ser explicitamente representado como uma convergência dupla (E).

Macro etapas

Uma macro etapa é a representação sob a forma de um símbolo único de um grupo **único** e **conexo** de etapas e transições. Uma macro etapa começa por uma **etapa de início** e finda por uma **etapa de término**.



A representação detalhada de uma macro etapa deve ser descrita no mesmo programa SFC. O símbolo da macro etapa e sua etapa de início devem ter o mesmo **número de referência**. A descrição de uma macro etapa pode conter o símbolo de uma outra macro etapa.

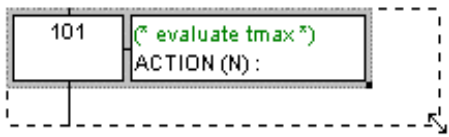
A.4.2 Introduzindo um gráfico SFC

Para desenhar um gráfico SFC, o usuário deve introduzir os componentes significativos do gráfico. Todas as linhas de conexão, horizontais e verticais, são traçadas automaticamente pelo editor SFC. Para inserir um componente SFC no gráfico, o usuário deve mover a seleção para a posição desejada e selecionar o tipo de símbolo na barra de ferramentas do editor. O símbolo é inserido na posição corrente. As seguintes seqüências de teclas também podem ser utilizadas:

F2: .....Insere uma etapa de início
F3: .....Insere uma única etapa
F4: .....Insere uma transição
F5: .....Insere um desvio para uma etapa
F6: F7: .....Insere uma divergência ou convergência OU / Adiciona ramificações
↑F6: ↑F7: .....Insere uma divergência ou convergência E / Adiciona ramificações
F8: .....Insere uma macro etapa
F9: ↑F9: .....Insere uma etapa de início ou de término dentro do corpo de uma macro etapa

(O símbolo "↑" indica uma combinação com a tecla SHIFT)

A grade de edição mostra as **células** da **matriz** de edição gráfica. Uma opção de editor permite que a grade seja mostrada ou não durante a edição do gráfico. A grade é muito útil para a inserção dos componentes gráficos e a seleção das partes do projeto. Utilize o comando "**Opções / Layout**" para mostrar ou não a grade.
O editor gráfico SFC sempre mostra a posição atual na matriz. A célula selecionada está marcada em cinza. Um pequeno retângulo no canto inferior direito pode ser utilizado para redimensionar livremente as células da matriz. Também permite a modificação da proporção X/Y das células.



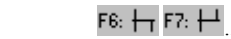
Criando uma divergência ou convergência

As divergências e as convergências são sempre desenhadas **da esquerda para a direita**. Para desenhar uma divergência ou uma convergência, suas ramificações devem ser colocados à **esquerda** da área do gráfico. O tipo de desenho (simples ou duplo) é configurado pela seleção de um dos seguintes botões da barra de ferramentas.

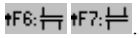
F6: F7: .....Inserir uma divergência ou convergência OU / Adicionar ramificações
↑F6: ↑F7: .....Inserir uma divergência ou convergência E / Adicionar ramificações

Adicionando ramificações às divergências

As posições de **início** e de **término** de cada **ramificação auxiliar** devem ser posicionadas na linha de divergência ou de convergência utilizando estes botões na barra de ferramentas. O canto esquerdo da divergência ou da convergência deve estar presente antes de inserir novas ramificações. Os cantos à direita têm o mesmo estilo (simples ou duplo) do canto principal à esquerda. Os cantos à direita não podem ser posicionados sem que o canto principal à esquerda tenha sido definido.



.....Inserir uma divergência ou convergência OU / Adicionar ramificações



.....Inserir uma divergência ou convergência E / Adicionar ramificações

Inserindo uma macro etapa

Este botão é utilizado para inserir uma macro etapa no gráfico principal. O corpo da macro principal deve ser descrito no mesmo programa SFC.

Corpo de uma macro etapa

As macro etapas devem ser descritas no mesmo programa SFC do gráfico principal. Uma macro etapa deve começar com uma **etapa de início** e terminar com uma **etapa de fim**. O gráfico que descreve a implementação da macro deve ser **conexo**. A etapa inicial da macro deve ter o mesmo **número de referência** que o símbolo da macro etapa no gráfico principal.

A.4.3 Trabalhando sobre um gráfico SFC existente

Você pode utilizar ou o mouse ou as setas do teclado (combinadas com a tecla SHIFT) para selecionar uma área retangular no gráfico. Toda a área selecionada está marcada em cinza. Os seguintes comandos do menu "**Editar**" podem ser utilizados :

Recortar / copiar / apagar / colar

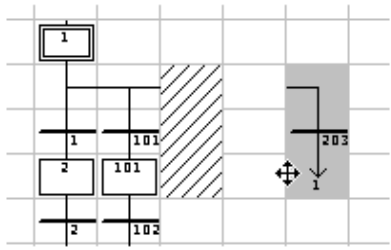
Os seguintes comandos estão disponíveis a partir do menu "**Editar**" quando o botão "**seta**" está selecionado na barra de ferramentas do editor :

Recortar .....Move o retângulo selecionado de uma tela para a área de transferência SFC
Copiar.....Copia o retângulo selecionado da tela para a área de transferência SFC
Apagar .....Apaga (delete) o retângulo selecionado
Colar .....Insere o conteúdo da área de transferência SFC na posição atual

O comando "**Editar / Colar**" copia a área de transferência SFC para a tela. Os comandos Copiar / Colar trabalham em ambos os gráficos SFC e programação em nível 2 de etapa/transição. Também é possível copiar um gráfico em um programa e colá-lo em outro programa SFC. Os elementos são inseridos antes da posição selecionada.

Mover elementos

Quando os elementos SFC são selecionados no gráfico SFC, você pode movê-los para outro lugar do gráfico arrastando a seleção com o mouse (não pode ser feito com o teclado). Enquanto você arrasta a seleção, a posição inicial dos elementos selecionados são hachurados.



A área de destinação dos elementos selecionados deve estar vazia. Nenhuma inserção é permitida durante a movimentação de símbolos SFC.

Renumerando etapas e transições

Cada etapa ou transição é identificada por um número lógico no gráfico SFC. O comando "**Editar / Renumerar**" permite que o usuário configure automaticamente todos os números seqüenciais de referência para qualquer uma das etapas e das transições do programa SFC editado. Quando um número de etapa é modificado, todos os desvios para esta etapa são automaticamente atualizados com o novo número de referência (também se aplica às macro etapas e etapas de início).

Acesso direto a uma etapa ou transição

O comando "**Editar / Ir para**" permite o acesso direto a uma etapa ou transição existente. A posição de rolamento da janela de edição é automaticamente atualizada para que a etapa ou a transição seja visível.

Localizar e substituir textos

O comando "**Editar / Substituir**" pode ser utilizado para localizar ou substituir texto no programa completo (todas as etapas e transições). A caixa de diálogo Localizar/Substituir é utilizada para entrar com o texto a ser localizado e abrir diretamente a seção de programação em nível 2 na qual o texto se encontra.

A.4.4 Inserindo programação de nível 2

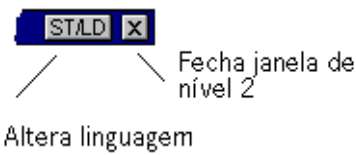
Para entrar com o texto de nível 2, o usuário deve clicar duas vezes sobre o símbolo da etapa ou transição. A programação de nível 2 é mostrada no lado direito da janela SFC. A linha de separação entre as duas áreas de edição SFC e nível 2 pode ser livremente movimentada.

É possível editar dois programas de nível 2 ao mesmo tempo. Os seguintes comandos estão disponíveis a partir do teclado, do mouse ou do menu "Editar" :

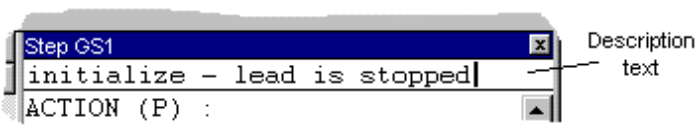
	Teclado	*Mouse*	*menu "Editar"*
Na mesma janela aberta	Enter	Duplo Clique	Editar nível 2
Numa janela separada	Ctrl+Enter	Ctrl + Duplo Clique	Editar nível 2 numa janela separada

Quando dois programas de nível 2 estão visíveis, a linha de separação entre as duas zonas pode ser movimentada livremente. O botão a direita da linha de título de uma zona de nível 2 é utilizado para fechar a janela correspondente.

A linguagem padrão para a programação de nível 2 é **ST** (Structured Text, Texto Estruturado). Para transições, a linguagem de programação para o nível 2 pode ser o editor **Quick LD**. Utilize o botão "**ST/LD**" na barra de título de nível 2 para modificar a linguagem ativa. Este comando só é válido se a janela de programação de nível 2 está vazia.



Uma caixa de edição com uma linha de texto é mostrada na parte superior da janela de nível 2. É utilizada para entrar com textos curtos de descrição para a etapa ou transição. Este texto será mostrado como um comentário IEC na janela principal do SFC. É muito útil já que é utilizado por outros comandos tais como "Ir para..." e também na impressão SFC para a documentação das etapas e transições SFC.



O comando "**Opções / Atualizar**" pode ser utilizado a qualquer momento quando janelas de nível 2 estão abertas para atualizar o gráfico SFC principal com programas de nível 2 modificados.

Inserindo um nome de variável

Durante a programação em linguagem texto, pressione este botão para selecionar uma variável declarada no dicionário do projeto e inserir seu nome na posição corrente do cursor. Durante a programação em Quick LD, pressione este botão para selecionar a variável a ser vinculada ao símbolo selecionado (contato, relé ou parâmetro de bloco).

Inserindo na etapa um bloco de ação de pulso

Durante a programação de nível 2 de uma etapa, pressione este botão para inserir o modelo de um bloco de ação de Pulso na posição corrente do cursor. A seguir, o formato de um bloco de ação de Pulso :

```
Action (P) :
    Declaração ST;
    ...
End_Action;
```

Ações de pulso são instruções que são executadas somente uma vez quando a etapa torna-se ativa. Refira-se ao manual de referência das linguagens ISaGRAF para mais detalhes sobre a programação SFC.

N

Inserindo na etapa um bloco de ação do tipo "Não armazenado"

Durante a programação de nível 2 de uma etapa, pressione este botão para inserir um modelo de um bloco de ação Não armazenado na posição corrente do cursor. A seguir, o formato de um bloco de ação Não armazenado :

```
Action (N) :
    Declaração ST;
    ...
End_Action;
```

Ações do tipo Não armazenado são instruções que são executadas a cada ciclo PLC quando uma etapa torna-se ativa. Refira-se ao manual de referência das linguagens ISaGRAF para mais detalhes sobre a programação SFC.

P0 P1

Novos qualificadores de ação P0 e P1

O ISaGRAF suporta novos qualificadores de ação **P0** e **P1**. Durante a programação do nível 2 de uma etapa, pressione estes botões para inserir o modelo de um bloco de ação P0 ou P1 na posição corrente do cursor. A seguir, o formato de tais blocos :

```
Action (P0) :          Action (P1) :
    Declaração ST                      Declaração ST;
    ...                                ...
End_Action;            End_Action;
```

As ações P1 são instruções que são executadas somente uma vez quando a etapa torna-se ativa (mesmo comportamento das ações de Pulso). As ações P0 são instruções que são executadas somente uma vez quando a etapa torna-se inativa. Refira-se ao manual de referência das linguagens ISaGRAF para mais detalhes sobre a programação SFC.

Ações booleanas

Outros textos semânticos estão disponíveis para atuar diretamente sobre uma variável booleana de acordo com a atividade da etapa. Tais ações estabelecem uma relação entre o **sinal de atividade da etapa** e uma variável booleana interna ou de saída. Eis a sintaxe das ações booleanas :

<variável_booleana> (N);	copia o sinal da atividade da etapa dentro da variável
< variável_booleana >;	mesmo efeito (atributo N é opcional)
/ < variável_booleana >;	copia o inverso (negação) do sinal da atividade da etapa dentro da variável

Outros recursos estão disponíveis para forçar uma variável booleana para o estado FALSE ou TRUE, quando a etapa torna-se ativa. Eis a sintaxe destas ações :

< variável_booleana > (S);	força a variável para TRUE quando o sinal de atividade da etapa torna-se TRUE
< variável_booleana > (R);	força a variável para FALSE quando o sinal de atividade da etapa torna-se TRUE

Ações SFC

Outros textos estão disponíveis para controlar a execução de um programa secundário. Uma ação SFC é uma seqüência SFC secundária, iniciada ou interrompida de acordo com a evolução do sinal de atividade da etapa. Uma ação SFC pode ser descrita com os qualificadores de ação **N** (Não armazenado), **S** (Set), ou **R** (Reset). Eis a sintaxe das ações SFC :

<programa_filho_de>> (N);	inicia a seqüência secundária quando a etapa torna-se ativa e a interrompe quando a etapa torna-se inativa
< programa_filho_de >;	mesmo efeito da anterior (o atributo N é opcional)
< programa_filho_de > (S);	inicia a seqüência secundária quando a etapa torna-se ativa – nada é feito quando a etapa torna-se inativa
< programa_filho_de > (R);	interrompe a seqüência secundária quando a etapa torna-se ativa - nada é feito quando a etapa torna-se inativa

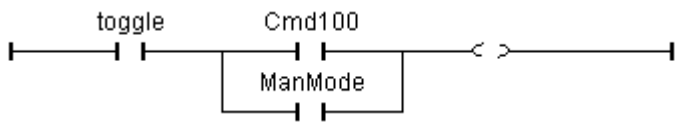
A seqüência SFC especificada na ação deve ser um **programa SFC secundário** do programa em edição, criado com o gerenciador de programa ISaGRAF.

Transições escritas em ST

O nível 2 de uma transição é uma expressão booleana. Para programar em linguagem ST, entre com a condição booleana de acordo com a sintaxe ST. Opcionalmente, a condição pode ser terminada por um ponto-e-vírgula .

Transições escritas em Quick LD

É possível utilizar o editor Quick LD para a programação do nível 2 de uma transição. Neste caso, o diagrama é composto de uma única bobina que representa a transição. O nome da transição não é repetido no diagrama. A seguir, um exemplo de transição programada com o editor Quick LD.



Durante a programação em Quick LD, utilize as setas do teclado para mover a seleção na grade lógica do diagrama e utilize os seguintes atalhos para inserir símbolos :
F2:.....insere um contato após o símbolo selecionado / inicia uma lógica
F3:.....insere um contato antes do símbolo selecionado
F4:.....insere um contato em paralelo com o símbolo selecionado
F6:.....insere um bloco após o símbolo selecionado
F7:.....insere um bloco antes do símbolo selecionado

F8:.....insere um bloco em paralelo com o símbolo selecionado
Você também pode clicar sobre os símbolos correspondentes na barra de teclas de função localizada na parte inferior da janela de edição de nível 2.

Aperte a tecla RETURN para associar um símbolo de variável ao contato selecionado, escolher um tipo de bloco (operação) ou conectar uma variável de entrada ou de saída de um bloco. Você também pode clicar duas vezes sobre um símbolo gráfico para selecionar a variável associada.

Aperte as teclas SPACE+CTRL para modificar o tipo do contato selecionado (direto ou inverso). Refira-se ao capítulo "Usando o Editor Quick LD" deste documento para mais detalhes sobre as possibilidades do editor Quick LD.

A.4.5 Utilizando a galeria SFC

O editor SFC ISaGRAF gerencia uma galeria de objetos SFC: É uma coleção de estruturas gráficas SFC que podem ser inseridas em qualquer gráfico SFC. Os elementos da galeria SFC podem incorporar opcionalmente os programas de nível 2 associados às etapas e às transições. Utilize os seguintes comandos do menu "**Ferramentas**" :

Copiar para galeria SFC	copia os elementos selecionados para a galeria SFC
Colar da galeria SFC	cola um elemento da galeria SFC na posição corrente

Durante a cópia para a galeria SFC (i.e., criando um novo elemento da galeria SFC), você pode solicitar a incorporação dos programas de nível 2 das etapas e transições selecionadas.

A.5 Utilização do editor de Fluxograma

O editor de Fluxograma ISaGRAF permite a inserção dos Fluxogramas e a programação em linguagem ST, IL ou Quick LD das ações e das decisões. O Fluxograma é um diagrama de decisões que também pode ser utilizado para descrever operações seqüenciais já que permite certas operações avançadas tais como desvios inversos não bloqueadores.

A.5.1 Fundamentos da linguagem FC

O **Fluxograma** (**FC**) é uma linguagem gráfica utilizada para descrever operações seqüenciais e um fluxo de decisões. Um diagrama de Fluxograma é composto de **Ações** e de **Decisões**. Entre as ações e as decisões estão as ligações orientadas representando o fluxo de dados. A seguir, os componentes da linguagem FC (Fluxograma):

Início do diagrama FC: Um símbolo "**Início**" deve aparecer no início do Fluxograma. É único e não pode ser omitido. Representa o estado inicial do diagrama quando está ativado.

Fim do diagrama FC: Um símbolo "**Fim**" deve aparecer no fim do Fluxograma. É único e não pode ser omitido. É possível que nenhuma ligação seja feita com o símbolo "Fim" (diagrama em loop infinito), mas o símbolo "Fim" está sempre presente no diagrama (na parte inferior). Representa o estado final do diagrama, quando sua execução é terminada.

Ligações (fluxo): Uma **ligação** (link) é uma linha que representa o fluxo entre dois pontos do diagrama. Uma ligação sempre é terminada por uma seta. Duas ligações não podem ser conectadas ao mesmo ponto de conexão fonte.

Ações: O símbolo de uma **ação** representa as ações a serem executadas. Uma ação é identificada por um número e um nome. Dois objetos diferentes do mesmo gráfico não podem ter o mesmo nome ou número lógico. A linguagem de programação de uma ação pode ser a ST, a LD ou a IL. Uma ação está sempre conectada por ligações, uma de chegada e outra de saída.

Decisão: Uma **decisão** representa uma **condição** booleana. Uma decisão é identificada por um número e um nome. De acordo com a avaliação da expressão vinculada ST, LD ou IL, o fluxo é direcionado para a "SIM" ou "NO". Quando a condição é programada em texto ST, a expressão pode ser, opcionalmente, seguida por um ponto-e-vírgula. Quando programada em LD, a única bobina representa o valor da condição.

Subprograma FC: O sistema permite a descrição de uma estrutura hierárquica de programas FC. Os programas FC estão organizados em uma **árvore hierárquica**. Cada programa FC pode chamar um ou mais **subprogramas FC**. Tais programas são chamados de programas secundários do programa FC que os chamou. Os programas FC que chamam subprogramas FC são chamados de programas principais. Os programas FC estão ligados entre si dentro de uma árvore hierárquica principal, utilizando a relação “principal-secundário”. O símbolo de um **subprograma** em um Fluxograma representa uma chamada

ao subprograma. A execução do programa FC principal é suspensa até que a execução do subprograma esteja terminada.

Ação específica E/S: Um símbolo de **ação específica E/S** representa uma série de ações a executar. Com os outros símbolos de ações, uma ação específica E/S é identificada por um número e um nome. A mesma semântica é utilizada em ações padrões e ações específicas E/S. O propósito das ações específicas E/S é simplesmente tornar a leitura do diagrama mais fácil focalizando as partes não portáteis do diagrama. A utilização das ações específicas E/S é opcional. Os blocos específicos E/S têm o mesmo comportamento das ações padrões.

Conector FC: Os **conectores** são utilizados para representar uma ligação entre dois pontos do diagrama, sem desenhá-la. Um conector é representado por um círculo ligado por uma linha à fonte do fluxo. O desenho do conector está completo, no lado apropriado (dependendo da direção do fluxo de dados), pela identificação do ponto de destino (geralmente o nome do símbolo de destino). Um conector sempre se refere a um elemento definido no mesmo diagrama. O símbolo de destino é identificado pelo seu número lógico.

Comentários FC: Os blocos de **comentário** contêm texto que não influenciam o comportamento do diagrama. Pode ser inserido em qualquer lugar do diagrama e é utilizado para documentar o programa.

A.5.2 Inserindo um Fluxograma

Para inserir um diagrama, você deve posicionar os elementos (ações, decisões, conectores, ...) na área de edição gráfica e desenhar as linhas de fluxo entre eles, os elementos.

Inserindo objetos

Para inserir um objeto no diagrama, selecione o botão correspondente na barra de ferramentas e depois clique no diagrama no local em que se deseja inseri-lo. Você pode colocar o elemento em uma área vazia ou inseri-lo em uma linha de fluxo. A inserção em uma linha é permitida somente nas verticais dirigidas de cima para baixo. Você pode inserir os seguintes elementos básicos:


.....ação programada em ST, IL ou Quick LD


.....ação específica E/S (destaca uma ação particular não portátil)


.....decisão programada em ST, IL ou Quick LD


.....conector


.....chama um subprograma FC


.....comentário (texto descritivo)

O editor de Fluxograma ISaGRAF também sugere uma lista de estruturas clássicas de Fluxograma. Tais estruturas só podem ser inseridas em uma linha de fluxo já existente. Não podem ser colocadas sobre uma área vazia:


.....If / Then / Else – (Se / Então / Senão) - (seleção binária)

 .....Repeat until – (Repetir até) - (espera por uma condição)
 .....While – (Enquanto) - (loop durante uma condição verdadeira)

Selecionando objetos

A seleção de objetos gráficos é necessária para a maioria dos comandos de edição. O editor gráfico ISaGRAF FC permite as seleções simples ou múltiplas. Para selecionar os objetos, o botão "**Selecionar**" (seta) deve ser selecionado na barra de ferramentas. Para selecionar um só objeto, o usuário só tem que clicar sobre o seu símbolo.

Para selecionar um conjunto de objetos, arraste o mouse sobre o diagrama para desenhar uma área retangular. Todos os objetos gráficos dentro do retângulo são selecionados.

Um objeto selecionado é desenhado em azul escuro, com pequenos quadrados pretos ao redor seu símbolo gráfico. Também é possível adicionar ou remover um objeto de uma seleção múltipla, clicando sobre o seu símbolo com a tecla Shift ou Ctrl pressionada.

Uma nova seleção cancela a seleção anterior. Para remover uma seleção existente, simplesmente clique com o mouse sobre uma área vazia, fora do retângulo que limita os objetos selecionados.

Para seleção simples, é possível usar as setas de teclado para mover a seleção de um objeto para o outro no diagrama. Também podem ser selecionadas as linhas de fluxo.

Inserindo comentários

Comentários podem ser inseridos em qualquer lugar do diagrama. Não têm nenhuma influência na execução de programa. Aumentam a legibilidade do diagrama. Para inserir um bloco de comentário, selecione o botão correspondente na barra de ferramenta e clique no diagrama sobre uma área vazia da zona de edição, no local em que deve ser inserido. A seguir, clique duas vezes sobre o símbolo de comentário para entrar com o seu texto. Nenhuma sintaxe ou caractere especial (do tipo "**(***" e "***)**") são necessários durante a inserção do texto no bloco de comentário. Um bloco de comentário pode ser redimensionado arrastando os cantos de sua borda quando é selecionado.

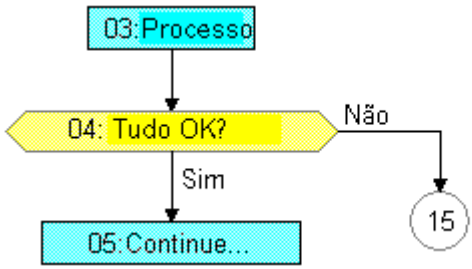
Desenhando fluxo de links

Selecione este botão na barra de ferramentas para desenhar uma linha entre os elementos existentes. Uma linha deve ser sempre desenhada na direção do fluxo. Primeiro clique sobre um ponto de saída ainda não conectado de um elemento FC e arraste o mouse até o ponto de destino para inserir a linha. O ponto de destino pode ser ou o topo (ponto de entrada) de um elemento FC não conectado ou qualquer local de uma outra linha já existente. Os pontos de convergência entre as linhas são marcados com pequenos círculos cinzas no Fluxograma. Os pontos de convergência também podem ser selecionados e movidos para melhorar a organização do diagrama.

Utilizando conectores

O editor de fluxograma ISaGRAF permite a utilização de conectores gráficos, em substituição aos arcos de ligações explícitos. Os conectores podem aumentar a legibilidade do diagrama e facilitar a manutenção de diagramas muitos grandes. Um conector não pode ser utilizado para estabelecer uma ligação com outro programa FC.

Um conector é colocado no diagrama como os outros objetos FC. É representado por um círculo contendo a referência numérica do elemento destino (destino da linha de fluxo). O nome do objeto destino é mostrado à direita do conector.



Movendo objetos

Para mover objetos no diagrama, você tem que selecioná-los e arrastar o mouse até a nova posição escolhida no diagrama. Você pode mover um único elemento ou todos os elementos de uma seleção múltipla. Os elementos não devem ser sobrepostos ao serem movidos. A movimentação de elementos não pode ser utilizada para conectá-los a uma linha existente.

Quando um elemento simples (ação, decisão ...) é movido, o editor de fluxograma ISaGRAF move automaticamente todos os elementos a sob ele conectados. Esta característica não funciona no caso de uma seleção múltipla.

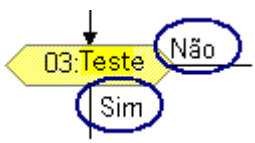
Redimensionando objetos

Qualquer um dos objetos gráficos de um fluxo, com exceção dos símbolos "Início", "Fim" e dos conectores, podem ser livremente redimensionados. Para redimensionar um elemento, você tem que selecioná-lo primeiro. Em seguida, arraste com o mouse os pequenos quadrados desenhados sobre a borda para modificar o seu tamanho.

Quando um elemento está conectado a uma linha de fluxo, o redimensionamento horizontal atua em ambos os lados, esquerdo e direito, de modo que o elemento fique corretamente centralizado em relação à linha, durante o redimensionamento.

Trocando as saídas de um teste

Você pode trocar as localizações das saídas SIM / NO em um teste. Para tal, clique duas vezes sobre uma das marcas, "SIM" ou "No", mostradas perto da símbolo decisão.



A.5.3 Trabalhando com um gráfico existente

Os comandos do menu "**Editar**" são utilizados para modificar ou completar um diagrama já existente. A maioria destes comandos atua sobre os elementos selecionados no diagrama.

Corrigindo um gráfico

Utilize a tecla DEL para remover os elementos selecionados. As linhas vinculadas aos elementos selecionados também são removidas. Utilize o comando "**Editar / Desfazer**" para restaurar os elementos removidos por um comando DEL. O comando DEL também pode ser aplicado a um grupo de elementos selecionados no diagrama. Os comandos "**Recortar**", "**Copiar**", "**Colar**" do menu "**Editar**" são utilizados para mover ou copiar elementos selecionados.

Localizar e substituir

Os comandos "**Editar / Substituir**" podem ser utilizados para localizar ou substituir textos no programa de nível 2 do conjunto de programas (todas as ações e todas as decisões programadas em ST, IL ou Quick LD). A caixa de diálogo Localizar/Substituir é utilizada para entrar com um texto a ser localizado e para abrir diretamente a seção de programa em que o texto é encontrado.

Acesso direto a um elemento

O comando "**Editar / Ir para**" permite o acesso direto a um elemento gráfico existente no diagrama. A posição de rolamento da janela de edição é automaticamente adaptada para que o objeto seja visível. O elemento, quando alcançado, é selecionado.

Renumerando elementos

O comando "**Editar / Renumerar**" é utilizado para renumerar elementos do Fluxograma. Qualquer elemento FC deve ser identificado com um único número de referência. Os números de referência são alocados pelo editor cada vez que novos elementos são inseridos. O comando "**Renumerar**" permite reposicionar todos os números de acordo com suas posições no diagrama. A ordem crescente da numeração é feita de cima para baixo e da esquerda para a direita.

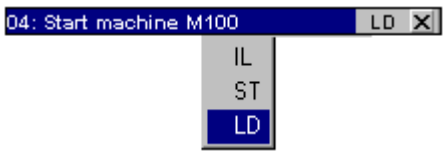
A.5.4 Introduzindo programas de nível 2

Para entrar com um programa de nível 2, o usuário deve clicar duas vezes sobre o símbolo de ação ou decisão. A janela de edição de nível 2 é aberta à direita da janela Fluxograma. A linha de separação entre os níveis 1 e 2 pode ser movimentada livremente de modo a redimensionar as janelas. Você pode abrir até duas janelas de nível 2 ao mesmo tempo. Os seguintes comandos estão disponíveis a partir do teclado, mouse ou menu "Editar" :

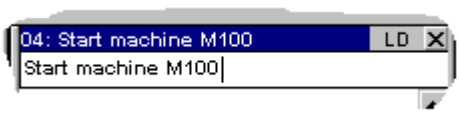
	Teclado	*Mouse*	*Menu "Editar"*
Dentro da mesma janela	Enter	Duplo Clique	Edita nível 2
Dentro de janela separada	Ctrl+Enter	Ctrl + DuploClique	Edita nível 2 em janela separada

Quando as duas janelas de nível 2 estão visíveis, você pode mover a separação entre ambas. O botão à direita da barra de título de uma janela de nível 2 é utilizado para fechá-la.

A linguagem padrão para o programa de nível 2 é **ST** (Structured Text, Texto Estruturado). A linguagem de programação também pode ser **IL** ou **Quick LD**. O nome da linguagem selecionada é mostrado em uma pequena caixa na linha de título. Utilize o comando "**Opções / Determinar linguagem de nível 2**" a partir dos menus ou clique sobre esta caixa para modificar a linguagem de programação ativa. Este comando só é válido se a janela de programa de nível 2 está vazia.



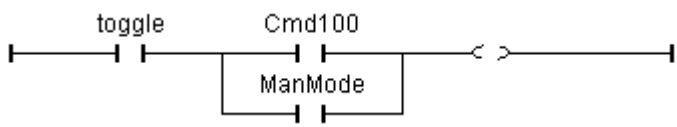
Uma caixa de linha única aparece no topo da janela de nível 2. É utilizada para entrar com o nome (textos de descrição curtos) da ação ou da decisão. Este texto será mostrado dentro dos símbolos FC (como um comentário IEC). É muito útil para a documentação do diagrama e será utilizado por outros comandos tais como "Ir para..." e também na impressão do diagrama para a descrição das ações e das decisões FC.



O comando "**Opções / Atualizar**" pode ser utilizado a qualquer momento quando as janelas de nível 2 estão abertas para atualizar o diagrama FC principal com os programas de nível 2 modificados.

A.5.5 Programação em nível 2 com Quick LD

O editor Quick LD está disponível para a programação de nível 2 do Fluxograma. No caso de uma decisão, o diagrama LD é composto de uma só lógica, com somente uma bobina que representa a decisão. O nome da decisão não é repetido sobre a bobina. A seguir, um exemplo de decisão programada em Quick LD.



Ao programar em Quick LD, utilize as setas do teclado para mover a seleção na grade lógica do diagrama e utilize os seguintes atalhos para inserir símbolos :
F2:.....insere um contato após o símbolo selecionado / inicia uma lógica
F3:.....insere um contato antes do símbolo selecionado
F4:.....insere um contato em paralelo com o símbolo selecionado
F5:.....insere uma bobina em paralelo a selecionada (não para decisões)
F6:.....insere um bloco após o símbolo selecionado

F7:.....insere um bloco antes do símbolo selecionado
F8:.....insere um bloco em paralelo com o símbolo selecionado
F9:.....insere um símbolo de desvio em paralelo com o selecionado (não para decisões)

Um desvio tem como destino uma etiqueta de lógica. O nome de uma lógica pode ser modificado pressionando-se a tecla ENTER quando a seleção estiver sobre o início da lógica. O editor ISaGRAF mantém a memória das etiquetas de lógica já inseridas, se foi especificado para um nome de lógica ou uma operação de desvio. A caixa de diálogo "**Rótulo do Desvio**" dá a possibilidade de entrar com uma nova etiqueta ou selecionar uma já existente. Se você entrar com um novo nome, ele será acrescentado automaticamente à lista. O botão "**Remover**" é utilizado para remover o nome selecionado da lista. Não remove a etiqueta na lógica selecionada no diagrama. Para fazer isto, só pressione a tecla OK quando a caixa de seleção estiver vazia.

Você também pode utilizar a barra de ferramentas LD em substituição às teclas de função.

Pressione ENTER quando a seleção estiver sobre um contato ou sobre um bloco para selecionar uma variável associada, entrar com um valor constante ou escolher um tipo de bloco. Você também pode clicar duas vezes sobre um símbolo com o mesmo efeito.

Pressione as teclas Ctrl+SPAÇO quando a seleção estiver sobre o contato ou uma bobina para modificar o seu tipo (direto, inverso). Refira-se ao capítulo "Usando o Editor Quick LD" neste documento para mais detalhes sobre as possibilidades do Quick LD.

A.5.6 Opções de vídeo

O comando "**Opções / Layout**" abre a caixa de diálogo na qual estão agrupados todos os parâmetros e opções para o espaço de trabalho e de desenho do diagrama. Utilize as caixas de seleção do grupo "Área de trabalho" para mostrar ou esconder a barra de ferramentas e de estado do editor. As opções do grupo "Documento" permite mostrar ou esconder os pontos da grade de edição e apresentar o diagrama em cores ou em preto e branco.

Utilize o botão "Zoom" da barra de ferramentas para modificar a taxa de zoom para a apresentação do diagrama. Este comando está igualmente disponível quando trabalhando em um programa Quick LD vinculado a uma ação ou decisão.

Utilize o botão "Grade" para mostrar ou esconder os pontos da grade de edição. Este comando também está disponível durante a edição em Quick LD das ações e das decisões.

Utilize o comando "**Opções / Fonte…**" para selecionar o nome da fonte do caractere a ser utilizado em todos os documentos ISaGRAF. Quando chamado de um bloco ST ou IL, você pode especificar o tamanho da fonte. Quando selecionando a fonte para uma vista gráfica (FC ou Quick LD), o estilo da fonte e o seu tamanho não são relevantes e não precisam ser especificados. Os editores gráficos ISaGRAF sempre calculam o tamanho da fonte em função da taxa de zoom corrente.

A.6 Utilização do editor Quick LD

A linguagem LD permite a representação gráfica das equações booleanas. Os operadores booleanos AND, OR, NOT estão explicitamente representados pela topologia do diagrama. As variáveis Booleanas de entrada estão vinculadas aos contatos. As variáveis de saída estão vinculadas às bobinas. O editor Quick LD ISaGRAF permite a elaboração rápida e fácil de um diagrama LD utilizando ou o teclado ou o mouse. Os elementos são automaticamente ligados e arrumados pelo editor Quick LD. Nenhuma conexão é desenhada manualmente pelo usuário. O editor Quick LD também arruma os elementos do diagrama de modo a otimizar o espaço gráfico ocupado.

A.6.1 Fundamentos da linguagem LD

Um programa LD é um conjunto de equações chamadas **lógicas** nas quais os contatos e as bobinas estão arrumados. A seguir, os componentes básicos de um diagrama LD:

Início de lógica (barra de energia à esquerda)

Cada lógica começa com uma barra de energia à esquerda que representa o estado inicial "TRUE". O editor Quick LD ISaGRAF cria automaticamente a barra de energia à esquerda quando o primeiro contato da lógica é posicionado pelo usuário. Cada lógica pode ter um nome lógico que pode ser utilizado como uma etiqueta nas instruções de desvio.

Contatos

Um contato modifica os fluxo de dados booleanos de acordo com o estado da variável booleana. O nome da variável é mostrada sobre o símbolo do contato. Os seguintes tipos de contatos são suportados pelo editor Quick LD ISaGRAF:

 .....contato normalmente aberto (NA)

 .....contato normalmente fechado (NF)

 .....contato com detecção de borda de subida

 .....contato com detecção de borda de descida

Bobinas

Uma bobina representa uma ação. A variável associada à bobina é forçada para o estado da lógica (estado da ligação à esquerda da bobina). O nome da variável é mostrado sobre o símbolo da bobina. Os seguintes tipos de bobina são suportados pelo editor Quick LD ISaGRAF :

 .....bobina direta

 .....bobina inversa

 .....bobina liga

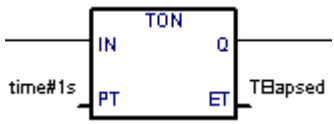
 .....bobina desliga

 .....bobina com detecção de borda de subida

 .....bobina com detecção de borda de descida

Blocos de função

Um bloco em um diagrama LD pode representar um operador, uma função, um subprograma ou um bloco de função. Seus primeiros parâmetros de entrada e de saída estão conectados à lógica. Os outros parâmetros de entrada e de saída são escritos fora do bloco.



Fim de lógica (barra de energia à direita)

Uma lógica termina com uma barra de energia à direita. Utilizando o editor Quick LD, a barra de energia à direita é automaticamente ligada a cada bobina posicionada pelo usuário.

Símbolo de desvio

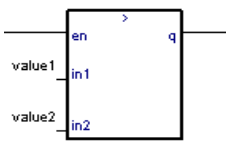
Um símbolo de desvio sempre se refere a uma etiqueta (nome da lógica) definido dentro do mesmo diagrama LD. O símbolo de desvio está posicionado no fim de uma lógica. Quando o estado da lógica é TRUE, a execução do diagrama é desviada diretamente para a etiqueta destino. Note que os desvios para trás são perigosos já que podem bloquear o ciclo automático (loop) em alguns casos.

Símbolo de retorno

Um símbolo "Retorno" é posicionado no fim de uma lógica. Indica que a execução do programa deve ser interrompida se o estado da lógica é TRUE. Um símbolo "Retorno" é equivalente a um desvio após a última lógica do diagrama.

A entrada "EN"

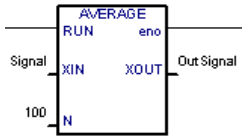
A primeira entrada de certos operadores, blocos de funções ou funções não podem ser do tipo booleano e nem podem ser conectados a uma linha LD booleana. Como a primeira entrada deve sempre estar conectada a uma lógica, outra entrada é automaticamente inserida na primeira posição chamada "**EN**". O bloco é executado somente se a entrada **EN** recebe um TRUE. A seguir, um exemplo do bloco comparador e seu equivalente em ST:




```
IF rung_state THEN
  q := (value1 > value 2);
ELSE
  q := FALSE;
END_IF;
(* continua lógica com estado q *)
```

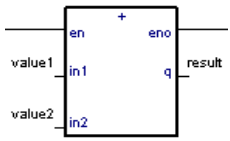
A saída "ENO"

A primeira saída de certos operadores, blocos de funções ou funções não podem ser do tipo booleano e nem podem ser conectadas a uma linha LD booleana. Portanto, para estes blocos, uma saída suplementar booleana é vinculada como primeiro parâmetro de saída. Esta saída tem o nome "**ENO**". A saída **ENO** sempre está no estado da primeira entrada do bloco. A seguir, um exemplo do bloco AVERAGE e seu equivalente em ST:




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(*continua lógica com estado eno *)
```

Em alguns casos, os dois parâmetros **EN** e **ENO** são requeridos. A seguir, um exemplo com um operador aritmético e seu equivalente em ST:




```
IF rung_state THEN
  result := (value1 + value2);
END_IF;
eno := rung_state;
(*continua lógica com estado eno *)
```

Limitações do editor Quick LD

O editor Quick LD ISaGRAF não permite continuar uma lógica (inserção de outros contatos ou bobinas) à direita de uma bobina. Se diversas saídas devem ser atribuídas na mesma lógica, as bobinas correspondentes devem ser colocadas em paralelo.

A.6.2 Inserindo um diagrama LD

Todos os comandos propostos para o editor Quick LD podem ser realizados ou pelo teclado ou com o mouse.

A grade de edição

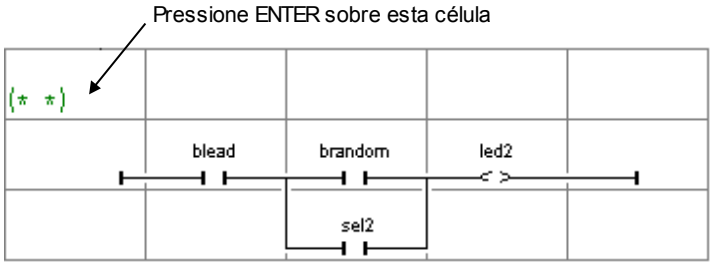
O diagrama LD é inserido em uma matriz lógica. Cada célula da matriz pode conter até um símbolo LD (contato ou bobina). Utilize as setas do teclado, ou clique sobre a célula para mover a seleção atual. A célula selecionada é marcada em vídeo reverso. Para algumas operações (recortar/copiar/colar), é possível selecionar diversas células. Para isto, com o mouse arraste o cursor no diagrama. Com o teclado, utilize as teclas de seta em conjunto com a tecla SHIFT pressionada.

Criando uma nova lógica

Para acrescentar uma nova lógica a um diagrama, mova a seleção após a última lógica existente e insira um contato (pressione F2 ou o botão correspondente na barra de ferramentas LD). Uma nova lógica com um contato e uma bobina é criada.

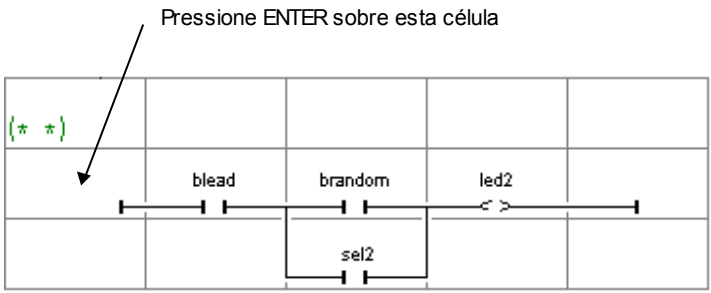
Introduzindo um comentário de lógica

Cada lógica pode ser documentada com até duas linhas de texto. Para introduzir um texto de comentário de lógica, mova a seleção na célula sobre a lógica e pressione a tecla ENTER ou clique duas vezes sobre esta célula com o mouse:



Introduzindo um nome de lógica

Cada lógica pode ser identificada por um nome. Este nome pode ser utilizado como uma etiqueta destino para operações de desvio. Para introduzir ou modificar a etiqueta de um lógica, mova a seleção sobre a início de lógica e pressione a tecla ENTER ou clique duas vezes sobre esta célula com o mouse:



O editor ISaGRAF Quick LD mantém a memória dos nomes de lógica que você já introduziu, quer tenha sido especificado para um nome de lógica ou uma operação de desvio. A caixa de diálogo "**Rótulo do desvio**"' lhe dá a possibilidade para introduzir um novo nome ou selecionar um existente.

Se você introduz um novo nome, este será automaticamente acrescentado à lista. O botão "**Remover**" é utilizado para remover o nome selecionado da lista. Não remove o nome de lógica que você selecionou no diagrama. Para fazer isto, basta pressionar **OK** quando a caixa de edição estiver vazia.

Introduzindo símbolos de lógica

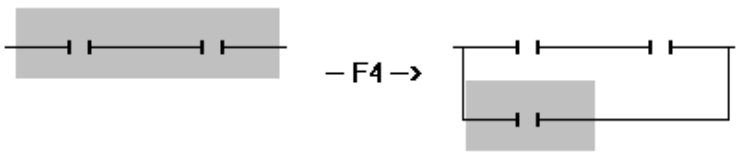
A inserção de símbolos (contatos, bobinas, blocos...) em um lógica existente sempre é feito de acordo com a seleção atual. Você tem que selecionar uma posição de célula válida dentro da lógica e pressionar uma das seguintes teclas de função, para inserir:

F2.....um contato antes do símbolo selecionado (à esquerda)
F3.....um contato depois do símbolo selecionado (à direita)
F4.....um contato em paralelo com o símbolo selecionado
F6.....um bloco antes do símbolo selecionado (à esquerda)
F7.....um bloco depois do símbolo selecionado (à direita)

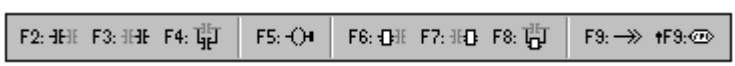
F8.....um bloco em paralelo com o símbolo selecionado

Os comandos a seguir são válidos quando a seleção estiver na saída de lógica (bobina):
F5.....adiciona um bobina em paralelo com a selecionada
F9.....adiciona um símbolo de "Desvio" em paralelo com a selecionada
Shift+F9.....adiciona um símbolo de "Retorno" em paralelo com a selecionada

Para inserção paralela (F4/F8), se diversos contatos de uma lógica são selecionados em conjunto, o símbolo é inserido em paralelo com o grupo de elementos selecionados. A seguir, um exemplo:



Para inserir símbolos no diagrama, você também pode utilizar os comandos do menu "**Inserir**". Com o mouse, você pode clicar na barra de ferramenta LD, sobre o tipo de símbolo que você deseja inserir:



Introduzindo nomes de variáveis

Para associar um símbolo de variável a um contato ou a uma bobina, selecione e pressione a tecla ENTER. Com o mouse, clique duas vezes sobre o contato ou sobre a bobina. Uma caixa de seleção de variável é apresentada. Refira-se ao capítulo "Maiores informações sobre editores de Programas" neste documento para informação adicional sobre como usar esta caixa. Para associar uma função, bloco de função ou operador a um bloco, pressione a tecla ENTER quando a seleção estiver na parte interna do seu retângulo. Para associar um símbolo de variável a um parâmetro de bloco de entrada ou de saída, a seleção deve estar sobre o local correspondente, **do lado de fora** do retângulo do bloco.

Caixas de diálogo incluindo listas de seleção de variável ou de bloco normalmente são utilizadas para a introdução de texto. Se o modo "**Entrada manual pelo teclado**" é verificado no menu "**Opções**", os símbolos de variáveis e os nomes de blocos são introduzidos diretamente em uma única caixa de edição de texto. Entre com o texto novo e pressione a tecla "**Enter**" para validá-lo, ou pressione "**Esc**" para encerrar a modificação e fechar a caixa de edição de texto. A caixa de edição de texto utilizada no modo "manual de entrada via teclado" não pode ser fechada com o mouse.

Modificando o tipo de contatos e bobinas

A opção "**Editar / Alterar tipo de bobina/contato**" modifica o tipo de contato ou de bobina selecionado. Um contato pode ser normalmente aberto (NA), normalmente fechado (NF), com detecção de borda de subida ou com detecção de borda de descida. Um bobina pode ser direta, inversa, liga ou desliga, com detecção de borda de subida ou de descida. Pressionando a barra de espaço (SPACE) obtém-se o mesmo efeito.

Inserindo uma lógica em um diagrama

O comando "**Editar / Inserir lógica**" insere uma nova lógica no diagrama, antes da selecionada. A lógica é iniciada com um contato e um bobina.

A.6.3 Trabalhando em um diagrama existente

Os comandos do menu "**Editar**" são utilizados para modificar ou completar um diagrama existente. A maioria destes comandos age sobre os elementos atualmente selecionados no diagrama.

Corrigindo um diagrama

A tecla DEL pode ser utilizada para remover os elementos selecionados. Não é possível remover bobinas, um desvio ou um símbolo de retorno quando for a única saída de uma lógica. Utilize o comando "**Editar / Desfazer**" para restaurar elementos após um comando DEL. O comando DEL também pode ser aplicado a um grupo de elementos selecionado no diagrama. O comando DEL pode ser utilizado quando seleção estiver sobre o texto de comentário de lógica para reinicializá-la. O comando DEL, utilizado quando a seleção estiver sobre o início de lógica, remove-a por inteiro.

Copiando símbolos

Os comandos "**Recortar**", "**Copiar**", "**Colar**" do menu "**Editar**" são utilizados para mover ou copiar elementos selecionados. Estes comandos não atuam sobre os comentários de lógica. O comando "**Editar / Colar especial**" lhe dá a escolha para inserir os elementos colados:

- antes do elemento selecionado (à esquerda)
- depois do elemento selecionado (à direita)
- em paralelo com o elemento selecionado

Gerenciando lógicas

Todos os comandos de edição (apagar, copiar, recortar...) atuam sobre toda a lógica se a seleção estiver sobre o início de lógica (barra de energia esquerda). Fornece assim um modo fácil de organizar as lógicas no diagrama, só movendo a seleção na primeira coluna. Também é possível estender a seleção verticalmente de forma que isto inclui diversos inícios de lógica. Neste caso, os comandos de edição podem ser aplicados a uma lista inteira de lógicas.

Localizar e substituir

Os comandos do menu "**Editar / Localizar**" e "**Editar / Substituir**" são utilizados para localizar e substituir textos no diagrama. Somente nomes completos podem ser localizados. A localização atua sobre os contatos, as bobinas, os nomes de bloco, os parâmetros de bloco e etiquetas de execução. Não pode ser utilizado para localizar uma “string” em um comentário de lógica. O comando Substituir não pode ser utilizado para mudar o tipo de um bloco. A pesquisa pode ser para cima ou para ou para baixo, começando da posição da seleção atual. Executa um "loop" quando os limites do diagrama são alcançados. Os seguintes atalhos também estão disponíveis para pesquisa rápida de nomes variáveis:

ALT+F2 localiza o próximo elemento com o mesmo nome de variável do elemento atualmente selecionado. Este recurso também pode ser aplicado aos blocos de funções e nomes de lógica.

ALT+F5 localiza a próxima bobina com o mesmo nome de variável do elemento atualmente selecionado. Este recurso é principalmente utilizado no modo depurar para rapidamente localizar as lógicas que forçam uma variável duvidosa.

A.6.4 Opções de vídeo

São utilizados os comandos do menu "**Opções**" para personalizar o desenho do diagrama LD na tela e esconder ou exibir alguns tipos de informação.

- ***Comentários de lógicas***

  Utilize o comando "**Opções / Comentários de lógicas**" para esconder ou exibir os comentários de lógica em todo o diagrama. Pode ser necessário esconder os comentários de lógica para uma visão mais condensada de um diagrama muito grande, já que cada comentário consome uma linha na matriz de edição. Esta opção não afeta os conteúdos dos comentários de lógica existentes e pode ser trocado a qualquer hora.

- ***Nomes e alias***

  Cada variável, quando associada a um contato, uma bobina ou um parâmetro de bloco E/S é identificado por seu nome simbólico. O o editor ISaGRAF Quick LD também introduz a noção ou "**alias**" para cada variável. O alias da variável é o texto de comentário variável, truncado antes do primeiro caractere '**:**', e limitado a 16 caracteres. A seguir, exemplos:

variable comment:	*alias:*
short text	short text
long text with no separator	long text with n
short text: long description	short text

  Alias não tem nenhum efeito na execução do diagrama LD e deve ser considerado como comentário para o ponto de vista sintático. Um alias de variável é extraído automaticamente do comentário de variável quando o nome é selecionado na lista de variável. Não pode ser modificado manualmente. Utilize os comandos "**Opções / Contatos e bobinas**" para selecionar um modo de exibição para identificação de variável. O seguinte modo está disponível:
  - exibir só os nomes de variáveis
  - exibir só os alias de variáveis
  - exibir nomes e alias

  O editor Quick LD não atualiza automaticamente documentos LD quando alias de variáveis são modificados no dicionário. Utilize o comando " **Opções / Contatos e bobinas / Atualizar aliases** " para atualizar todos os alias no diagrama editado. Você também pode fixar a opção "**Sempre atualizar ao abrir**" de "**Opções / Contatos e bobinas** " para solicitar ao ISaGRAF que atualize todos os alias utilizados automaticamente toda vez que um programa Quick LD está aberto. Aviso: Configuração desta opção pode aumentar significativamente o tempo gasto para abrir um programa.

Opções de desenho

O comando "**Opções / Layout**" abre uma caixa de diálogo onde se agrupa todos os parâmetros e opções relativos à área de trabalho de editor e ao desenho do diagrama LD gráfico.
Utilize as caixas de verificação no grupo de caixas "área de trabalho" para exibir ou esconder a barra de ferramentas de edição, a barra de estados e a barra de ferramentas LD. As opções no grupo de caixas "Documento" permite mostrar ou esconder os pontos da grade de edição e habilitar/desabilitar a utilização de cores para o desenho.

As opções no grupo de caixas "Zoom" permites selecionar a taxa principal de zoom. Você também pode utilizar o botão "zoom" na barra de ferramentas de edição para trocar entre os valores padrões de zoom.

Você também pode personalizar a relação entre eixos X/Y das células na grade de edição. Esta última opção pode ser utilizada para reduzir a largura de célula padrão, se pequenos nomes são normalmente utilizados para as variáveis. Você também pode utilizar o botão "largura" na barra de ferramenta de edição para modificar a relação entre eixos X/Y sem entrar na caixa de diálogo Layout.

Utilize o comando "**Opções / Fonte…**" para selecionar o nome da fonte de caractere a ser utilizado em todos os documentos gráficos ISaGRAF. Durante a seleção de fonte, o estilo de fonte e o tamanho não são importantes e não precisam ser especificados. Os editores gráficos ISaGRAF sempre calculam o tamanho da fonte de acordo com a taxa de zoom selecionada.

A.7 Utilizando o editor FBD/LD

O editor gráfico ISaGRAF FBD/LD permite que o usuário entre no programa completo FBD que pode incluir partes em LD. Combina as capacidades de edição de gráfico e de texto, permitindo que diagramas e respectivas entradas e saídas possam ser inseridos. Como este editor é mais dedicado a linguagem FBD, os diagramas LD puros deveriam ser inseridos utilizando o editor ISaGRAF Quick LD.

A.7.1 Fundamentos das linguagens FBD/LD

A linguagem **FBD** é uma representação gráfica de muitos tipos diferentes de equações. **Operadores** são representados através de caixas de função retangulares. As funções de entrada estão conectadas à esquerda da caixa. As funções de saída estão à direita da caixa. O diagrama de entradas e saídas (**variáveis**) está conectado às caixas de função através de **links lógicos**. Uma saída de uma caixa de função pode ser conectada à entrada de outra caixa.

A linguagem LD habilita a representação gráfica de expressões booleanas. Operadores Booleanos **E**, **OU**, **NÃO** são representados explicitamente pelo diagramas topológicos. As variáveis Booleanas de entrada estão atreladas aos **contatos** gráficos. As variáveis Booleanas de saída estão atreladas às **bobinas**. Os **c**ontatos e as bobinas estão conectados entre si e entre as barras de energia através de **linhas horizontais**. Cada segmento de linha tem um estado booleano de **FALSE** ou **TRUE**. O estado booleano é o mesmo para todos os segmentos diretamente ligados. Qualquer linha horizontal conectada na **barra de energia** à esquerda tem o estado TRUE.

Os diagramas LD e FBD são sempre interpretados da esquerda para a direita, e de cima para baixo. Refira-se ao Manual de referência Linguagem ISaGRAF para mais detalhes sobre linguagens LD e FBD. Estes são os componentes gráficos básicos das linguagens LD e FBD, suportadas pelo editor FBD/LD:

Barra de energia à esquerda

As lógicas devem ser conectadas na **barra de energia à esquerda** que representa o "estado inicial" TRUE. O editor ISaGRAF FBD também permite conectar qualquer símbolo booleano a uma barra de energia à esquerda.

Barra de energia à direita

As bobinas podem ser conectadas pela direita à **barra de energia à direita**. Este é um recurso opcional ao utilizar o editor ISaGRAF FBD/LD. Se uma bobina não estiver conectada à direita, inclui uma barra de energia à direita no seu próprio desenho.

Conexão vertical "OU" do LD

A conexão vertical LD aceita diversas conexões à direita e à esquerda. Cada conexão à direita equivale a uma combinação OU de conexões à esquerda.

Contatos

Um contato modifica o fluxo de dados booleanos, de acordo com o estado de uma variável booleana. O nome da variável é exibido no símbolo de contato. Os seguintes tipos de contatos são suportados pelo editor ISaGRAF FBD/LD:

.....contato normalmente aberto (NA)

.....contato normalmente fechado (NF)

.....contato com detecção de borda de subida

.....contato com detecção de borda de descida

Bobinas

Um bobina representa uma ação. Deve estar conectada à esquerda de um símbolo booleano como um contato. O nome da variável é exibido no símbolo de bobina. Os seguintes tipos de bobinas são suportados pelo editor ISaGRAF FBD/LD:


.....bobina direta

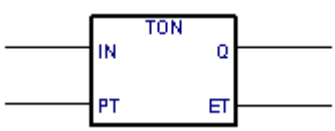

.....bobina inversa


.....bobina liga


.....bobina desliga

Blocos de função

Um bloco em um diagrama FBD pode representar uma função, um bloco de função, um subprograma ou um operador. As entradas e as saídas devem ser conectadas às variáveis, contatos ou bobinas, ou a outro bloco de entrada ou de saída. Os nomes de parâmetros formais são exibidos dentro do retângulo do bloco.



Etiquetas

As etiquetas podem ser colocadas em qualquer lugar do diagrama. As etiquetas são utilizadas como destinos de instruções de desvio, para modificar a ordem de execução no diagrama. As etiquetas não estão conectadas a outros elementos. É muito recomendado colocar as etiquetas à esquerda do diagrama para aumentar a legibilidade do diagrama.

Desvios

Um símbolo de desvio sempre refere-se a uma etiqueta, colocada em qualquer lugar do diagrama. Sua conexão esquerda deve estar unida a um ponto booleano. Quando a conexão esquerda for VERDADE (TRUE), a execução do diagrama vai diretamente (salta) a esta etiqueta designada. Note que desvios para trás são perigosos já que em alguns casos podem conduzir a um bloqueio do loop PLC.

Símbolo de retorno

Um símbolo de retorno é conectado a um ponto booleano. Indica que a execução do programa deve ser interrompida se o estado de lógica for VERDADE.

Variáveis

As variáveis no diagrama são representadas dentro de pequenos retângulos, conectados à esquerda ou à direita com outros elementos do diagrama.

Conexões

Os links de conexão são desenhados entre os elementos postos no diagrama. Os links sempre são desenhados de um ponto de saída para um ponto de entrada (na direção do fluxo de dados).

Conexões booleanas com lógica invertida

Alguns links booleanos são representados com um pequeno círculo nas suas extremidades. Isto representa uma negação booleana da informação transportada pela ligação.

Cantos definidos pelo usuário

Os pontos definidos pelo usuário podem ser definidos em links. Eles permitem que o usuário controle manualmente o encaminhamento de um link. Se nenhum canto é colocado, o editor ISaGRAF FBD/LD utiliza um algoritmo de encaminhamento padrão.

A.7.2 Inserindo um diagrama FBD

Para inserir um diagrama, você tem que colocar elementos (blocos, variáveis, contatos, bobinas...) na área gráfica e fazer as ligações entre eles.

Inserindo objetos

Para inserir um objeto no diagrama, selecione o botão correspondente na barra de ferramentas e clique duas vezes sobre a área gráfica na qual você deseja inseri-lo.

Selecionando objetos

A seleção de objetos gráficos é necessária na maioria dos comandos de edição. O editor gráfico ISaGRAF LD/FBD habilita a seleção de um ou mais objetos existentes na área do diagrama. Para selecionar objetos, o botão "**Selecionar**" (botão com uma seta) deve estar ativado na barra de ferramentas do editor. Para selecionar um objeto, o usuário só tem que clicar sobre o seu símbolo. Para selecionar uma lista de objetos, arraste o mouse no diagrama e selecione uma área de retangular. Todos os objetos gráficos compreendidos nesta área de seleção são marcados como "**Selecionado**". Ao redor de um objeto selecionado são desenhados pequenos quadrados pretos. Ao fazer uma nova seleção, todas os objetos previamente selecionados são desmarcados. Para remover a seleção existente, simplesmente clique com o mouse sobre uma área vazia, fora do retângulo que limita os objetos selecionados.

Inserindo comentários

Comentários podem ser inseridos em qualquer lugar do diagrama. Não têm nenhuma influência na execução de programa. Aumentam a legibilidade do diagrama. Para inserir um bloco de comentário, selecione o botão correspondente na barra de ferramenta e clique no diagrama sobre uma área vazia da zona de edição, no local em que deve ser inserido. A seguir, clique duas vezes sobre o símbolo de comentário para entrar com o seu texto. Nenhuma sintaxe ou caractere especial (do tipo "**(***" e "***)**") são necessários durante a

inserção do texto no bloco de comentário. Um bloco de comentário pode ser redimensionado arrastando os cantos de sua borda quando é selecionado.

Movendo objetos

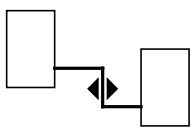
Para mover objetos no diagrama, você tem que selecioná-los e arrastar o mouse até a nova posição escolhida no diagrama. Para mover objetos conectados, o usuário simplesmente tem que mover os símbolos gráficos colocados no diagrama. O editor ISaGRAF LD/FBD redesenhará as linhas de conexão automaticamente entre os objetos que foram movidos, baseado nas novas posições.

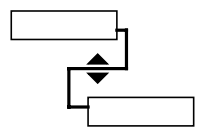
Desenho de conexões

Selecione um destes botões na barra de ferramentas para desenhar uma ligação entre pontos de conexão de elementos existentes. Se você desenhar uma ligação entre um ponto de conexão e um local vazio no diagrama, esta é terminada automaticamente por um canto definido pelo usuário, de forma que você continue desenhando outro segmento.

Modificando o desenho de conexões

Os comandos "**Ferramentas / Mover linha**" são utilizados quando uma ligação é selecionada no diagrama para modificar seu encaminhamento automático. Este comando não tem nenhum efeito quando a ligação é conectada a um canto definido pelo usuário. Quando uma ligação é desenhada como três segmentos, este comando modifica a posição do segundo segmento. A seguir, exemplos:





Invertendo o tipo de uma conexão

Você pode modificar facilmente o tipo de ligação (com ou sem negação booleana) clicando duas vezes com o mouse na sua extremidade direita.

Desenhando lógicas LD

Para desenhar uma nova lógica LD, primeiro insira a barra de energia à esquerda. Posicione uma bobina que será unida automaticamente à barra de energia. Outros contatos e conexões OU vertical podem ser inseridos diretamente na linha de lógica, sem desenhar nenhum novo link de conexão.

Quando um novo contato LD ou bobina é inserido em um espaço vazio da área de edição, a nova linha de lógica horizontal é automaticamente desenhada a partir do novo elemento inserido até as barras de energia existentes à esquerda e à direita. Esta linha não é automaticamente desenhada se o novo contato ou bobina não é colocado entre as barras de energia. O novo contato ou bobina inserido pode então ser movido livremente na lógica desenhada. As linhas horizontais criadas pelo editor durante a inserção de um contato LD ou símbolo de bobina pode ser selecionado e pode ser apagado. Você pode inserir um novo símbolo de contato LD ou de bobina na linha horizontal de uma lógica existente. O editor automaticamente interrompe a lógica e conecta-a aos pontos de conexão à esquerda e à direita do novo contato ou bobina inserido.

Conexões múltiplas

Uma conexão múltipla pode ser criada à direita de qualquer ponto de **saída**. Significa que a informação é **difundida** a vários outros pontos no diagrama. O mesmo estado é propagado à direita em cada extremidade. O número de linhas desenhados à direito de um ponto de conexão de saída não está limitado. Duas linhas de conexão não podem ter as respectivas extremidades à direita conectadas no mesmo ponto de **entrada**, com exceção dos seguintes símbolos LD,:

.....barra de energia à direita

.....conexão múltipla no operador (OU) esquerdo

Estes símbolos LD podem ter um número ilimitado de entradas.

A.7.3 Trabalhando sobre um diagrama existente

Os comandos do menu "**Editar**" são utilizados para modificar ou completar um diagrama existente. A maioria destes comandos age sobre os elementos atualmente selecionados no diagrama.

Corrigindo um diagrama

A tecla DEL pode ser utilizada para remover os elementos selecionados. São apagados os links pendentes com os elementos selecionados. Utilize o comando "**Editar / Desfazer**" para restaurar elementos depois de um comando DEL. O comando DEL também pode ser aplicado a um grupo de elementos selecionado no diagrama. Os comandos "**Recortar**", "**Copiar**", "**Colar**" do menu "**Editar**" são utilizados para mover ou copiar elementos selecionados.

Localizar e substituir

Os comandos de menu "**Editar / Localizar**" e "**Editar / Substituir**" são utilizados para localizar e substituir textos no diagrama. Podem ser localizados somente nomes completos. A pesquisa atua sobre os contatos, bobinas, nomes, variáveis e etiquetas de blocos. Não pode ser utilizado para localizar uma “string” em um texto de comentário. O comando Substituir não pode ser utilizado para modificar o nome de um bloco. A pesquisa pode ser feita para cima ou para baixo, começando na posição de seleção atual. Executa um "loop" quando os limites do diagrama são alcançados.

Apresentando a ordem de execução

Quando um diagrama FBD inclui loops de retorno (realimentação), a ordem de execução não pode seguir o único método da esquerda para a direita/de cima para baixo. Para evitar confusão, utilize o comando "**Ferramentas / Exibir ordem de execução**" ou pressione as teclas **Control+F1** para exibir a ordem de execução que será utilizada no tempo de compilação. São exibidas etiquetas numeradas de 1 a N perto de símbolos que conduzem a uma ação (bobinas, conjunto de variáveis e blocos de função).

Inserindo símbolos e textos

Clique duas vezes com o mouse em um elemento para entrar no símbolo ou texto associado. Isto se aplica a variáveis, contatos e bobinas, textos de comentário e etiquetas. Quando utilizado em um contato ou bobina, isto também permite mudar seu tipo (direto, invertido...).

Caixas de diálogo incluindo listas de seleção de variável ou bloco normalmente são utilizadas para a entrada de texto. Se o modo "**Entrada manual pelo teclado**" está ativo no menu "**Opções**", os símbolos de variáveis e os nomes de blocos são inseridos diretamente em uma única caixa de edição de texto. Entre com o novo texto e pressione a tecla "**Enter**" para validá-lo, ou pressione a tecla "**Escape**" para encerrar a modificação e fechar a caixa de edição de texto. A caixa de edição de texto utilizada no modo " Entrada manual pelo teclado " não pode ser fechada com o mouse.

Se o modo "**Entrada Automática**" está ativo no menu " **Opções** ", o símbolo de variável deve ser inserido imediatamente após a inserção de um novo contato ou bobina. O símbolo sempre deve ser inserido imediatamente quando uma variável ou uma etiqueta é inserida.

Selecionando o tipo de bloco de função

Clique duas vezes com o mouse sobre um bloco para modificar o seu tipo. O tipo de bloco é selecionado da lista de operadores, funções e blocos de função disponíveis. Este comando também permite modificar o número de pontos de entrada no caso de um operador comutativo. (por exemplo AND, OR, ADD, MUL...)

Obtendo espaço livre

Quando você pressiona o botão direito do mouse na área de desenho FBD, um menu instantâneo é exibido. Contém os seguintes comandos que podem ser utilizados para inserir ou remover espaço livre na área do cursor de mouse:

Inserir linhas:.....Este comando insere espaço horizontal livre, constituído de 4 linhas de acordo com o passo da grade, começando na posição do cursor de mouse em que o menu instantâneo é apresentado.

Apagar linhas:.....Este comando remove espaço horizontal não utilizado (linhas) começando na posição do cursor de mouse em que o menu instantâneo é apresentado. Este comando não pode ser utilizado para remover elementos FBD.

Quando menu instantâneo estiver aberto, uma linha cinza na área do desenho FBD indica o local que o espaço vazio será inserido ou será removido.

A.7.4 Opções de vídeo

São utilizados os comandos do menu "**Opções**" para personalizar o desenho do diagrama FBD na tela.

Personalização do layout

O comando "**Opções / Layout**" abre uma caixa de diálogo na qual se agrupa todos os parâmetros e opções relativos à área de trabalho do editor e o desenho do diagrama gráfico. Utilize as caixas de ativação na caixa de grupo "Área de trabalho" para exibir ou esconder as barras de ferramentas e a barra de estado do editor. A opção da caixa de grupo "Documento" permite mostrar ou esconder os pontos da grade de edição.

Opções da caixa de grupo "Zoom" permitem selecionar uma taxa principal de zoom. Você também pode utilizar o botão "zoom" na barra de ferramentas de edição para trocar entre os valores padrões de zoom.

Utilize o comando "**Opções / Fonte…**" para selecionar o nome da fonte de caractere a ser utilizado em todos os documentos gráficos ISaGRAF. Durante a seleção de fonte, o estilo de fonte e o tamanho não são importantes e não precisam ser especificados. Os editores gráficos ISaGRAF sempre calculam o tamanho da fonte de acordo com a taxa de zoom selecionada.

A.7.5 Formatação de texto e marcação das modificações

O editor ISaGRAF LD/FBD permite designar um estilo gráfico a qualquer componente de um diagrama LD/FBD. Um estilo é principalmente definido como um diagrama especial colorido. Mas ISaGRAF também utiliza estilos para habilitar marcação das modificações em diagramas com o propósito de controle de versão.

Note que os estilos não são visíveis durante a simulação ou a depuração online, já que cores (vermelho e azul) são utilizadas neste modo para realçar os estados TRUE / FALSE de variáveis de monitoração.

- ***Formatações predefinidas de textos***

Os seguintes estilos são predefinidos:

Normal.....Desenho padrão (preto). Para marcação de modificação, o estilo "normal" indica que elementos que têm tal estilo são parte do diagrama original. Elementos de estilo "Normal" são esquadrinhados normalmente durante execução.

Modificado .....Elementos marcados como "Modificado" são pintados de rosa. Para a marcação da modificação, o estilo "Modificado" é utilizado para realçar elementos que foram adicionados ou modificados depois da liberação do diagrama original. Elementos de estilo "Modificado" são esquadrinhados normalmente durante execução.

Apagado.....Elementos marcados como "Apagado" são pintados de cinza, com linhas tracejadas. Não são levados em conta tais elementos para a execução do diagrama. Este estilo é utilizado para monitorar os elementos removidos depois da liberação do original quando o controle de versão é requerido.

Personalizado.....Além do estilo predefinido, o editor ISaGRAF LD/FBD permite a seleção de qualquer cor a ser aplicada a uma parte do diagrama. São considerados tais elementos como tendo um estilo "Personalizado". Este estilo não tem nenhum efeito na execução do diagrama no tempo de execução.

Utilize os comandos do submenu "**Estilo**" no menu "**Editar**" para aplicar um estilo manualmente a elementos selecionados.

- ***Marcação de modificações***

O uso de estilos e a disponibilidade do estilo "Apagado" permite a marcação de modificação automática em um diagrama existente. Utilize o comando "**Marcar modificações**" no menu "**Editar/Estilo**" para ativar ou desativar a marcação de modificações.
Quando a opção " **Marcar modificações** " está configurada, todos os elementos modificados ou adicionados ao diagrama são automaticamente configurados como estilo "Modificado". Quando um elemento é apagado, utilizando os comandos "Apagar" ou "Recortar", eles não são visualmente removidos do diagrama, mas simplesmente marcados com o estilo "Apagado". Isto permite que o usuário automaticamente monitore todas as modificações introduzidas no diagrama.
Utilize "**Editar/Estilo/Remover todos os itens apagados**" para remover todos os elementos marcados com o estilo "Apagado" do diagrama LD/FBD. Este comando não considera a seleção atual e sempre aplica ao diagrama inteiro.
Para restaurar um elemento marcado com o estilo "**Apagado**", selecione o elemento desejado e aplique a ele o estilo "**Normal**", o estilo "**Modificado**" ou qualquer estilo "**Personalizado**". Tal operação pode conduzir a conexões inválidas (mais que um link conectado ao mesmo ponto de entrada) que será detectado durante a próxima verificação de programa.

A.8 Utilização do editor de textos

Este capítulo só descreve as características e os comandos do editor de texto ISaGRAF, particularmente quando utilizado para entrar no código fonte de programas ST e IL.

A.8.1 Comandos de edição

Os comandos do menu "**Editar**" são utilizados para trabalhar no texto editado. A maioria destes comandos atua sobre os caractere selecionados no diagrama ou executam uma ação no local atual do sinal de circunflexo (^).

Recortar e colar

A tecla DEL pode ser utilizada para remover o texto selecionado. Utilize o comando "**Editar / Desfazer**" para restaurar elementos após um comando DEL. Os comandos "**Recortar**", "**Copiar**", "**Colar**" do menu "**Editar**" são utilizados para mover ou copiar texto no programa, ou inserir pedaços de textos copiados na área de transferência através de outros aplicativos.

Localizar e substituir

Os comandos de menu "**Editar / Localizar**" e "**Editar / Substituir**" são utilizados para localizar e substituir textos no programa. Qualquer cadeia de caractere pode ser localizada. A pesquisa pode ser executada para frente ou para trás, começando no local atual do sinal de circunflexo (^). Não executa um “loop" quando são alcançados os limites do programa.

Ir para linha

O comando "**Editar / Ir para linha**" é utilizado para mover o sinal de circunflexo a um número de linha específico. Isto pode ser muito útil para o acesso a uma linha com um erro descoberto pelo compilador ISaGRAF em um programa ST ou IL e referenciado por um número de linha.

Inserir símbolo do dicionário

Utilize o comando "**Editar / Inserir variável**" para inserir, na posição atual do sinal de circunflexo, o símbolo de uma variável ou objeto declarado no dicionário de projeto. O símbolo é selecionado pela caixa de seleção de variável comum descrita no capítulo " Outros comandos dos editores de programa " neste documento.

Inserir arquivo

O comando "**Editar / Inserir Arquivo**" insere todo o conteúdo de um arquivo no local atual do sinal de circunflexo (^). Note que somente os arquivos texto ASCII puros podem ser controlados por este comando.

A.8.2 Opções

Os comandos do menu "**Opções**" são utilizados para exibir ou esconder a barra de ferramentas de edição e selecionar a fonte de caractere. A fonte de caractere selecionada será utilizada para qualquer edição de texto em todo o Ambiente de Trabalho ISaGRAF.
Quando utilizado para entrar no código fonte de um programa ST / IL, o comando "**Opções / Exibir palavras-chave**" é utilizado para exibir ou esconder uma barra de ferramentas que agrupa as palavras chaves mais comuns de linguagem ST ou IL. Clique duas vezes sobre um botão na barra de ferramentas para inserir a palavra chave correspondente ou operador no local atual do sinal de circunflexo (^).

A.9 Outros comandos dos editores de programa

Este capítulo contém informações úteis sobre os recursos de edição que são comuns a todos os editores de programa ISaGRAF. Basicamente consiste de associações com outras ferramentas ISaGRAF e caixas de diálogos comuns ISaGRAF.

A.9.1 Chamando outras ferramentas ISaGRAF

Verifica (compila) o programa

O comando "**Arquivo / Verificar**" executa o gerador de código ISaGRAF para verificar a sintaxe de programa atualmente editado. No caso de linguagem SFC, ambos os níveis 1 e 2 são conferidos. Quando a verificação de sintaxe estiver completa, a janela de gerador de código deve ser fechada para continuar o trabalho no programa. Se há só um programa no aplicativo (o editado) o código de aplicativo é gerado se nenhum erro de sintaxe é detectado. O comando "**Opções / Opções do compilador**" é utilizado para configurar a compilação e otimização de parâmetros. Refira-se ao capítulo "Utilização do gerador de código" neste documento para informação adicional sobre compilação e geração de código.

Simular ou depurar o aplicativo

Os comandos "**Arquivo / Simular**" e "**Arquivo / Depurar**" executam o depurador gráfico ISaGRAF ou no modo simulação ou no modo realmente conectado e reabre o programa SFC editado no modo depurar. Utilizado no modo depurar, nenhuma modificação pode ser inserida no programa.

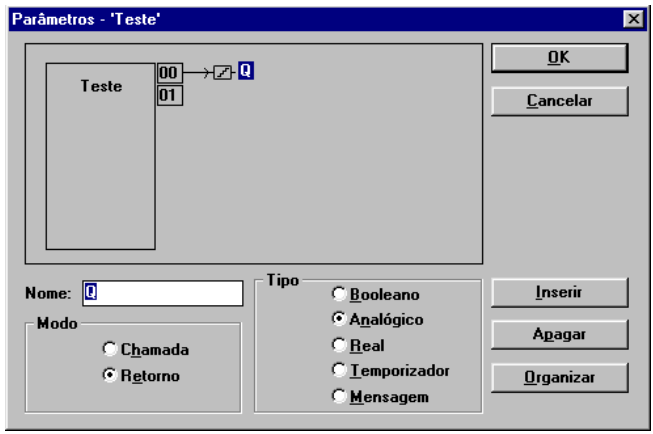
Editando o dicionário de variáveis

O comando "**Arquivo / Dicionário**" é utilizado para editar o dicionário de variáveis para o aplicativo atual e o programa atual. Também contém os pontos de entrada para editar as palavras definidas pelo usuário. As declarações **locais** ou palavras definidas referem-se ao programa atualmente editado.

A.9.2 Parâmetros do programa

Quando o programa editado é uma função, um bloco de função ou um subprograma, o comando "**Arquivo / Parâmetros**" é utilizado para definir os seus parâmetros de chamada e de retorno. Este comando não tem nenhum efeito se o programa editado é um programa SFC ou de mais alto nível da seção **Início** ou **End**.

Subprogramas, funções ou blocos de função podem ter até **32** parâmetros (entrada ou saída). Uma função ou subprograma sempre tem um (e somente um) parâmetro de retorno que tem que ter o mesmo nome da função para conformar com as convenções de escrita de linguagem ST. A seguinte caixa de diálogo é utilizada para descrever os parâmetros do subprograma:



A lista no canto superior esquerdo da janela mostra os parâmetros, na ordem do modelo de chamada: primeiro os parâmetros de chamada, por último os parâmetros de retorno. A parte inferior da janela mostra a descrição detalhada do parâmetro selecionado na lista. Qualquer um dos tipos de dados ISaGRAF pode ser utilizado para um parâmetro. Os parâmetros de retorno devem estar localizados após os parâmetros de chamada na lista. Os parâmetros de nome devem seguir às seguintes regras:

- o comprimento do nome não pode exceder 16 caracteres
- o primeiro caractere deve ser uma letra
- os demais caracteres devem ser letras, algarismos ou caractere de sublinha (_)
- o nome não faz distinção entre maiúscula/minúscula

O comando "**Inserir**" é utilizado para inserir um novo parâmetro antes do parâmetro selecionado. O comando "**Apagar**" é utilizado para apagar o parâmetro selecionado. O comando "**Organizar**" automaticamente rearranja (classifica) os parâmetros, de forma que os parâmetros de retorno sejam colocados no fim da lista.

A.9.3 Outros comandos do menu "Arquivo"

Os seguintes comandos estão disponíveis no menu "**Arquivo**" de todos os editores de programas:

Abrir outro programa

O comando "**Arquivo / Abrir**" permite ao usuário fechar o programa que está sendo editado e iniciar a edição de outro programa do projeto selecionado com a mesma linguagem. Esta função não pode ser utilizada para editar um programa escrito em outra linguagem. O novo programa selecionado substitui o programa apresentado na janela de edição.

Imprimindo o programa

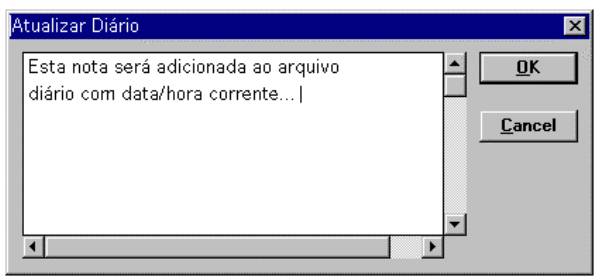
O comando "**Arquivo / Imprimir**" imprime o programa editado na impressora. Este comando automaticamente executa o gerador de documento ISaGRAF para imprimir o programa editado e variáveis locais anexadas.

Para alguns programas gráficos (SFC, FBD and Quick LD) pode-se utilizar o comando "**Editar / Copiar desenho (metafile)**" para copiar para a área de transferência o desenho do gráfico, no formato metafile, para poder ser colado em outros aplicativos tais como processadores de texto. Para programas SFC, somente a informação de nível 1 (gráfico, numeração e comentário de nível 1) aparece no metafile copiado.

A.9.4 Atualização do arquivo de diário do programa

O arquivo de diário atrelado ao programa editado pode ser inserido manualmente utilizando o comando "**Arquivo / Diário**". O arquivo de diário é atualizado automaticamente com mensagens de saída de verificação de sintaxe sempre que o programa é compilado. Saídas compiladas são completadas com a informação de data e hora da compilação.

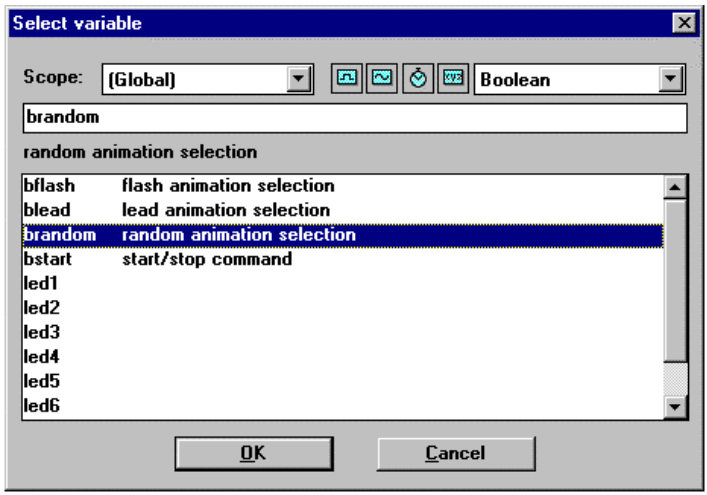
Se o modo "**Atualizar Diário**" é selecionado no menu "**Opções**" do editor de programas, a seguinte caixa de diálogo é aberta sempre que o programa é salvo em disco.



Se o botão OK é pressionado, a nota de texto inserida é então armazenada no fim do arquivo de diário com informação de data e hora atual. Esta característica é muito útil para manutenção de programas completos, bem como fornece ajuda útil sobre o ciclo de vida de programa.

A.9.5 Seleção de uma variável do dicionário

Ao editar um programa texto (ST ou IL) o "**Editar / Inserir variável**" permite a seleção do nome de uma variável declarada a ser inserida na posição atual do sinal de circunflexo (^). Ao editar programa LD ou FBD, a seleção variável é requerida para a descrição de bobinas de contato, parâmetros de bloco E/S ou caixas de variáveis FBD. Em ambos os casos, a seguinte caixa de diálogo é aberta para selecionar uma variável declarada:



A caixa de seleção "**Escopo**" é utilizada para selecionar entre variáveis globais e locais. A caixa de seleção à direita permite a seleção do tipo de dados. Os ícones pequenos ao lado da caixa de seleção de tipo são botões que podem ser utilizados como atalhos para selecionar a maioria dos tipos de dados atuais;

 .....Booleano
 .....Inteiro / Real
 .....Hora
 .....Mensagem

Para selecionar uma variável, clique sobre o seu nome na lista. São exibidos então o seu nome e o comentário no topo da lista. Pressione então botão "**OK**" para confirmar sua seleção. Também é possível inserir um nome de variável diretamente no controle de edição sem utilizar a lista.

A.9.6 A janela de incidentes

Os comandos a seguir estão disponíveis no menu "**Ferramentas**" de todos os editores de linguagem. São utilizados para apresentar as informações em uma pequena lista texto na parte inferior da janela de edição e utilizada para a navegação pelo programa.

"Exibir saída do compilador" apresenta na janela de incidentes as mensagens de erro da última compilação do programa editado.

"Localizar em diagrama" localiza as ocorrências de um texto em todo o programa editado e as apresenta na janela de incidentes. Para as linguagens SFC e FC, este comando procura em todos os programas de nível 2.

"Esconder lista de saída" fecha a janela de incidentes

Quando mensagens de erro ou ocorrências forem apresentadas na janela de incidentes, clique duas vezes sobre uma linha para mover a seleção diretamente para o local correspondente. Para as linguagens SFC e FC, este comando abre a janela de programação de nível 2 do elemento correspondente.

A.10 Utilização do editor de dicionário

O dicionário ISaGRAF é uma ferramenta de edição para a declaração das variáveis internas, variáveis de E/S, instâncias de bloco de função e "palavras definidas" do aplicativo. O dicionário agrupa as variáveis declaradas e as instâncias de bloco de função do aplicativo e as palavras definidas como cadeias constantes.
Variáveis, blocos de função e palavras definidas devem ser declaradas no dicionário antes de utilizá-las no código fonte. Variáveis e palavras definidas podem ser utilizadas com qualquer uma das linguagens de automação: SFC, FBD, LD, ST e IL. Blocos de função utilizados em linguagem FBD não têm que ser declarados porque os editores ISaGRAF FBD e Quick LD automaticamente declaram as instâncias dos blocos utilizados.

Variáveis

As variáveis são classificadas de acordo com o seu **alcance** e o seu **tipo**. Somente variáveis do mesmo tipo e o mesma alcance podem ser inseridas na mesma grade de entrada. Estes são alcances básicos para variáveis:

GLOBAL .....pode ser utilizada por qualquer programa do projeto atual
LOCAL .....pode ser utilizada por somente um programa

Estes são tipos básicos de variáveis:

BOOLEANA..(booleano) valores binários FALSE/TRUE
ANALOG.....(analógico) valores reais ou inteiros
TIMER.....(hora) valores de tempo
MESSAGE.....(mensagem) cadeia de caracteres (“string”)

Uma variável é identificada por um nome, um comentário, um atributo, um endereço de rede e outros campos específicos. Aqui estão os atributos básicos de variáveis:

INTERNA.....memória variável
ENTRADA .....variável vinculada a um dispositivo de entrada
SAÍDA.....variável vinculada a um dispositivo de saída
CONSTANT.....variável interna de leitura (com valor inicial)

Nota: **Temporizadores** são sempre variáveis do tipo **Interna**. Variáveis do tipo **ENTRADA** e **SAÍDA** sempre têm o alcance **GLOBAL**.

Palavras definidas

Uma palavra definida é uma alias que pode ser utilizada em qualquer linguagem para substituir uma cadeia de texto. O texto substituído pode ser um nome variável, uma expressão constante ou uma expressão complexa. As palavras definidas são classificadas de acordo com o seu alcance. Somente palavras definidas do mesmo tipo e de mesmo alcance podem ser inseridas na mesma grade de entrada. Aqui estão alcances básicos:

COMUM .....pode ser utilizada por qualquer programa de qualquer projeto
GLOBAL .....pode ser utilizada por qualquer programa do projeto atual
LOCAL .....pode ser utilizada por somente um programa

Uma palavra definida é identificada por um nome, por um bloco bem definido de equivalência de texto ST e um comentário livre.

Instâncias de blocos funcionais

As instâncias de blocos de função utilizadas nas linguagens ST e IL devem ser declaradas no dicionário. Porque um bloco de função tem dados internos "escondidos", cada cópia de um bloco de função deve ser identificada. O exemplo a seguir mostra o bloco de função "R_TRIG" (detecção de borda de subida) definido na biblioteca, utilizado para a detecção de bordas em diferentes variáveis. Cada cópia do bloco deve ser identificada por um único nome. A nomeação do tipo de bloco e a definição de seus parâmetros é feita utilizando o gerenciador de biblioteca:

Nome do bloco:	**R_TRIG**
***Parâmetros*:**	**Input=CLK**
	Output=Q

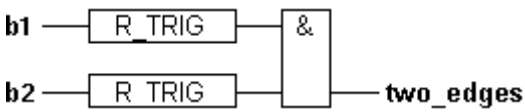
A nomeação das instâncias é feita utilizando o editor de dicionário:

***Nome instância*:**	**TRIG_B1**	***Nome bloco*:**	**R_TRIG**
***Nome instância*:**	**TRIG_B2**	***Nome bloco*:**	**R_TRIG**

As instâncias declaradas podem ser utilizadas em programas ST:

```
TRIG_B1 (b1);
edge_b1 := TRIG_B1.Q;      (* b1 variável detecção de borda *)
TRIG_B2 (b2);
edge_b2 := TRIG_B2.Q;      (* b2 variável detecção de borda *)
```

As instâncias de bloco de função declarado podem ser **GLOBAL** (conhecido por qualquer programa no projeto) ou **LOCAL** a um programa. Blocos de função utilizados em linguagens FBD ou LD não têm que ser declarados porque o editor ISaGRAF FBD declara automaticamente as instâncias dos blocos utilizados.



(* os blocos de função sempre têm o nome do bloco definido na biblioteca. Os editores ISaGRAF FBD e Quick LD declaram um exemplo automaticamente sempre que um bloco é inserido no diagrama *)

Instâncias de bloco de função automaticamente declaradas pelos editores FBD e Quick LD sempre são **LOCAL** para o programa editado.

Endereços de rede

Endereços de rede são **opcionais**. Uma variável com um endereço de rede diferente de zero pode ser monitorada por um sistema externo (por exemplo, um sistema de visualização de processo) no tempo de execução. Mais geralmente, o endereço de rede provê um mecanismo de identificação para cada tempo de execução do sistema de comunicação que não pode controlar nomes simbólicos. Um endereço de rede pode ser inserido para cada variável, durante sua descrição completa, quando a variável é criada ou é modificada.

A.10.1 A janela do dicionário principal

A janela de edição de dicionário mostra uma lista de variáveis do mesmo tipo e alcance. O tipo e o alcance de variáveis editadas sempre são exibidas na barra de título.

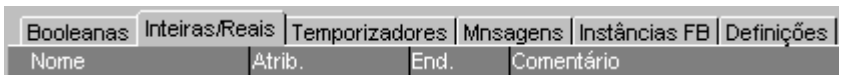
A janela de edição mostra somente os campos principais da descrição de variável: nome, endereço de atributo e de rede e texto de comentário. A descrição completa da variável selecionada sempre é exibida na barra de estado. Utilize os seguintes botões na barra de ferramentas para selecionar o alcance da variável a ser editada:

COMUM.....pode ser utilizada por qualquer programa de qualquer projeto

GLOBAL.....pode ser utilizada por qualquer programa do projeto atual

LOCAL.....pode ser utilizada por somente um programa

Utilize o controle "Tab" exibido com barra de título para selecionar o tipo de objeto a ser editado:



Utilize o campo de entrada de texto à esquerda da barra de ferramenta para localizar um nome do prefixo da variável. Neste caso, a pesquisa é processada na lista inteira, desde o início, baseada na seleção atual. O comando "**Editar / Localizar**" também está disponível para localizar uma cadeia de texto em nomes e comentários de variável, e para mover a seleção para esta variável. A localização não faz distinção entre maiúscula/minúscula.

A.10.2 Variáveis de gerenciamento

Os comandos de menu "**Arquivos**" disponíveis trabalha sobre toda a classe de variáveis selecionada, instâncias de bloco de função ou palavras definidas. Utilize o comando "**Outros**" para selecionar o tipo e o alcance de objetos a serem editados.

Variáveis de impressão

Utilize o comando "**Arquivos / Imprimir**" para imprimir a lista atualmente editada de variáveis ou palavras definidas, em um dispositivo impressora padrão Windows™. A

impressão é feita utilizando o gerador de documento ISaGRAF. A impressão inclui a descrição completa de cada variável ou palavra definida do tipo atualmente editado.

Criando novas variáveis

O comando "**Editar / Novo**" permite que o usuário crie novas variáveis, instâncias de bloco de função ou palavras definidas para o alcance e tipo selecionado. Novas variáveis são inseridas antes da variável atualmente apontada pela barra de seleção. Quando este comando é executado, uma caixa de entrada é aberta para inserir a descrição da variável. Quando a descrição está completa, pressione o botão "**Armazenar**" para colocá-la na lista. A caixa de entrada é reaberta automaticamente, de modo que o usuário pode entrar com outras variáveis com o mesmo comando "**Editar**". Pressionando o botão "**Cancelar**" da caixa de diálogo interrompe o processo de criação de variável.

Modificando variáveis existentes

O comando "**Editar**" do menu "**Editar**" permite que o usuário modifique a descrição da variável atualmente apontada pela barra de seleção. Quando este comando é executado, uma caixa de entrada é aberta para modificar a descrição da variável. Quando a descrição está completa, ao pressionar o botão "**Armazenar**" a modificação está habilitada. O usuário também pode pressionar os botões "**Próximo**" e "**Anterior**" para estender o comando de modificação às variáveis adjacentes. Ao pressionar o botão "**Cancelar**" a caixa de diálogo é fechada sem armazenar qualquer modificação.

Recortar e colar

A ferramenta de edição de dicionário ISaGRAF habilita a **seleção de múltiplas-linha**. Muitos comandos estão disponíveis para trabalhar na lista de variáveis atualmente editada. A seguir, os comandos do menu "**Editar**" que estão disponíveis:

COPIAR .....Copia o grupo selecionado de variáveis para a área de transferência de dicionário
RECORTAR .....Copia o grupo selecionado de variáveis e o remove da lista editada
CLEAR.....Remove o grupo selecionado de variáveis da lista editada
COLAR.....Insere a área de transferência do dicionário antes da variável selecionada

As funções Copiar/Recortar/Colar podem ser utilizadas de uma lista de variáveis para outra. Elas não podem ser utilizadas entre listas de tipos de objetos diferentes.

Variáveis de classificação

O comando "**Ferramentas / Sort**" classifica as variáveis ou palavras definidas da lista atualmente editada. A ordem de escolha é determinada pelos atributos das variáveis:

- primeiro as variáveis internas
- então as variáveis de entrada
- finalmente as variáveis de saída

As variáveis com o mesmo atributo são classificadas em ordem alfabética. As palavras definidas sempre são classificadas em ordem alfabética.

Configurando endereços de rede

Endereços de rede são **opcionais**. Uma variável com um endereço de rede diferente de zero pode ser **monitorado** por um sistema externo (por exemplo um sistema de visualização de

processo) durante o tempo de execução. Um endereço de rede pode ser inserido para cada variável, durante sua descrição completa, quando a variável é criada ou é modificada. O comando "**Ferramentas / Renumerar endereços**" permite que o usuário configure endereços de rede de um grupo inteiro de variáveis. Quando este comando é executado, ele atua sobre o grupo de variáveis selecionado na lista. Entrando em um endereço de **base hexadecimal** (endereço para a primeira variável do grupo) resulta em endereços de rede das variáveis do grupo serem configuradas com **endereços consecutivos**. Entrando em um endereço de base nula reajusta para zero o endereço de rede de todas as variáveis selecionadas.

Importando as palavras correspondentes à "true/false"

Ao editar palavras definidas, o comando "**Ferramentas / Importar definições true/false** " permite que o usuário defina automaticamente como palavras chaves de linguagem as "strings" atreladas a variáveis booleanas para representar os estados TRUE e FALSE. Tais "strings" são normalmente definidas para depuração de formatação. Elas têm que ser especificadas como palavras definidas se elas serão utilizadas em programas. Este comando localiza "strings" booleanas "true/false" nas declarações com o mesmo alcance da atualmente selecionada para a edição das palavras definidas.

A.10.3 Descrição de objetos

Uma descrição completa deve ser inserida para cada variável, instância de bloco de função ou palavra definida. Os campos de descrição são diferentes para cada tipo de objeto. Os seguintes campos são comuns para qualquer tipo de variáveis:

Nome ..... Nome da variável: o primeiro caractere deve ser uma letra, os caracteres seguintes podem ser letras, algarismos ou '_'.
Endereço na Rede ..... Endereço de rede hexadecimal (opcional). Quando este campo é diferente de zero, a variável pode ser monitorada por sistemas externos durante a execução.
Comentário..... Comentário livre para a descrição de variável.
Conservar ..... Esta opção indica que a variável deve ser salva em uma memória de backup.

Estes são outros campos de descrição para uma variável **booleana** :

Attributos ..... Especifica uma variável interna, constante, de entrada ou de saída
Valores "False"..... "string" utilizada como valor FALSE durante a depuração.
Valores "True" ..... "string" utilizada como valor TRUE durante a depuração.
Inicializar como true ..... O valor inicial é TRUE se esta opção está ativa, caso contrário o valor inicial é FALSE.

Estes são outros campos de descrição para uma variável **inteira** ou **real**:

Atributos..... Especifica uma variável interna, constante, de entrada ou de saída.

Formatação ..... Especifica uma variável inteira ou real (ponto flutuante). O formato de apresentação durante a depuração pode ser selecionado.
Unidade..... "string" utilizada para identificar a unidade física durante a depuração.
Conversão..... Nome da tabela de conversão ou função de conversão atrelada à variável (somente para variáveis de entrada ou de saída)
Valor inicial..... Valor inicial da variável (deve ter o mesmo formato da variável). Se não é especificado, o valor inicial é 0.

Estes são outros campos de descrição para uma variável **temporização**:

Atributos..... Especifica uma variável interna ou constante.
Valor inicial..... Valor inicial da variável (valor de tempo). Se não está especificado, o valor inicial é tempo#0s.

Estes são outros campos de descrição para uma variável **mensagem** :

Atributos..... Especifica uma variável interna, constante, de entrada ou de saída.
Tamanho máximo ..... Especifica o número máximo de caracteres que podem ser armazenados na mensagem.
Valor inicial..... Valor inicial da variável (o comprimento não pode exceder a capacidade da mensagem). Se não especificado, o valor inicial é uma "string" vazia.

Estes são os campos de descrição para uma **palavra definida**:

Nome ..... Nome utilizado em arquivos fonte ST: o primeiro caractere deve ser uma letra, os caracteres seguintes podem ser letras, algarismos ou '_'.
Definição..... "string", de acordo com a sintaxe ST, que substitui a palavra definida durante a compilação. Exemplo: Nome = PI - Equivalência = 3,14159
Comentário..... Comentário livre para a equivalência definida.

Estes são os campos de descrição para uma **instância de bloco de função**:

Nome ..... Nome da instância, utilizada nos arquivos fonte ST: o primeiro caractere deve ser uma letra, os caracteres seguintes podem ser letras, algarismos ou '_'.
Tipo ..... Nome do bloco de função correspondente na biblioteca.
Comentário..... Comentário livre para a instância de bloco de função.

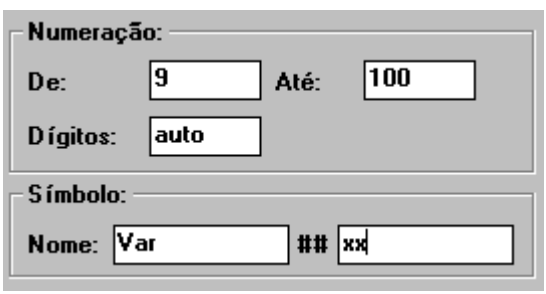
A.10.4 Declaração rápida

O comando "**Ferramentas / Declaração rápida**" permite que você declare diversas variáveis ao mesmo tempo. As variáveis criadas pela declaração rápida são nomeadas utilizando uma convenção de numeração. Para isso, você tem que definir:
- o índice (número) da primeira e da última variáveis,
- o texto a ser adicionado antes e depois do número em símbolos de variáveis
- o número de dígitos utilizado para expressar o número em símbolos de variáveis.

Adicionalmente, você pode especificar atributos básicos de variáveis criadas (interna, de entrada ou de saída...), mais algumas propriedades que dependem do tipo variável (atributo "Conservar", formato inteiro ou real, comprimento máximo de "string" de mensagem).

Você sempre precisa definir um texto a ser inserido antes do número de variável, já que um símbolo de variável não pode começar com um dígito. Quando o " número de dígitos " é fixado em "Auto", o ISaGRAF formata o número de variável no número mínimo necessário de dígitos. Quando o número de dígitos é especificado, o ISaGRAF formata todos os números com o comprimento especificado adicionando caracteres '0' à esquerda. A configuração de um número fixo de dígitos para números de variável pode ser muito útil para prevenir a classificação lexicográfica incorreta. A seguir, um exemplo.

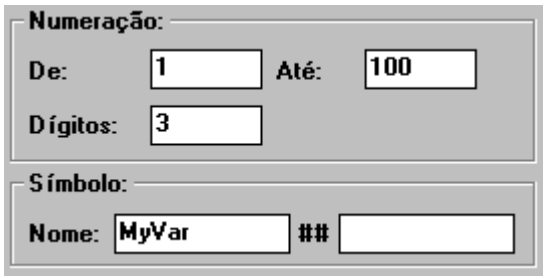
Exemplo: Esta configuração para declaração rápida:



criará as três variáveis a seguir:

Var9xx Var10xx Var11xx

Exemplo: Esta configuração para declaração rápida:



criará 100 variáveis com nomes de MyVar001 até MyVar100

A.10.5 Mapa de endereçamento Modbus SCADA

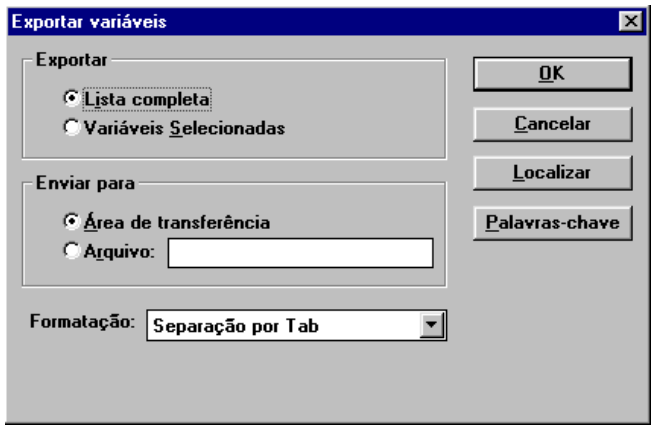
Os "endereços de rede" ISaGRAF são freqüentemente utilizados para estabelecer uma ligação entre o sistema ISaGRAF e um SCADA baseado em comunicação Modbus. Neste caso, o SCADA é um Modbus mestre e o ISaGRAF destino atua como um Modbus escravo. Os endereços de rede são utilizados para criar um mapa Modbus virtual para todas as variáveis ISaGRAF que devem ser controladas a partir do SCADA. O comando "**Ferramentas / Mapa de endereçamento Modbus SCADA**" é poderoso para rapidamente criar um mapa Modbus virtual com as variáveis do aplicativo.
As ferramentas de mapeamento mostram duas listas. A superior é um segmento (4.096 locações) do mapa Modbus, mostrando as variáveis mapeadas (aquelas que têm um endereço de rede). A lista mais baixa mostra variáveis não mapeadas (sem endereço de rede definido). O endereço "0" não pode ser utilizado para mapear uma variável.
Utilize os comandos "**Mapear variável selecionada**" e "**Remover variável do mapa**" do menu "**Editar**" para mover uma variável de uma lista para outra, e assim construa o mapa. As mesmas ações podem ser executadas clicando duas vezes sobre um símbolo de variável em uma lista, para enviá-la para outra lista. A qualquer momento, você pode utilizar a lista suspensa "Segmento" para ver outro segmento do mapa.
Podem ser utilizados os comandos do menu "**Opções**" a qualquer momento para exibir endereços no formato ou decimal ou hexadecimal.
O comando "**Editar / Localizar**" é utilizado para localizar uma variável declarada, se já está mapeada ou não.

A.10.6 Troca de informações com outros aplicativos

A ferramenta de edição de dicionário ISaGRAF oferece funções de importação/exportação de modo a trocar informações com outros aplicativos, como processadores de textos, planilhas eletrônicas, gerenciador de banco de dados ... Estes comandos estão agrupados no menu "**Ferramentas**". O comando "**Exportar texto**" elabora um texto ASCII puro de descrição dos campos descrevendo um conjunto de objetos editados e armazena este texto ou na área de transferência do Windows ou em um arquivo. Tal informação é tipicamente utilizada por outro aplicativo. O comando "**Importar texto**" importa campos de descrição de declaração de variáveis, descritos no formato ASCII texto puro, armazenado ou na área de transferência do Windows ou em um arquivo, e atualiza a lista editada com campos importados. Tal informação é tipicamente produzida por outro aplicativo.

Exportando dados

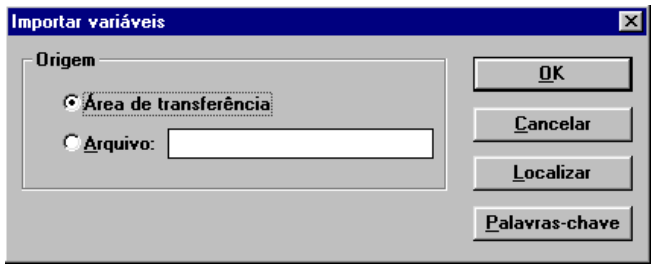
A caixa de diálogo abaixo aparece quando o comando "**Exportar texto**" é executado. Permite que o usuário controle o mecanismo de exportação.



Ativando a escolha "**Lista completa**" indica que a lista editada completa tem que ser exportada. A seleção atual é ignorada neste caso. Ativando a escolha "**Variáveis Selecionadas**" indica que somente as variáveis destacadas serão exportadas.
Se a opção "**Área de transferência**" está ativa, a informação exportada é armazenada, em formato ASCII texto puro, na área de transferência do Windows. O texto está então disponível para os comandos "colar" em outros aplicativos. Se a opção "**Arquivo**" está ativa, o texto exportado é armazenado em um arquivo ASCII. O nome do caminho completo deste arquivo tem que ser informado. O comando "**Localizar**" pode ser utilizado para localizar um nome do caminho existente.
Então o usuário escolhe um formato para o texto exportado. Os formatos disponíveis são descritos em seções adicionais. Pressionando o botão "**OK**" a função de exportação é executada. Pressionando o botão "**Cancelar** " a caixa de diálogo é fechada e sai do comando de exportação.
Todos os campos dos objetos selecionados são armazenados no texto exportado, na ordem de declaração padrão. A primeira linha do texto exportado contém o nome dos campos. Cada objeto é descrito em uma linha de texto. O separador "fim de linha" (EL) é a seqüência MS-DOS padrão "**0d-0a**". Os nomes utilizados para identificar os campos na primeira linha exportada podem ser modificados, pressionando o botão "**Palavras-chave**". Este comando é descrito em seções adicionais.

Importando dados

A caixa de diálogo abaixo aparece quando o comando "**Importar texto**" é executado. Permite que o usuário controle o mecanismo de importação.



Se a opção "**Área de transferência**" está ativa, a informação importada é tirada da área de transferência do Windows, em formato ASCII texto puro. Se a opção "**Arquivo**" está ativa, o texto exportado é lido em um arquivo ASCII. O nome do caminho completo deste arquivo tem que ser informado. O comando "**localizar**" pode ser utilizado para localizar um nome do caminho existente.
A função de importação reconhece automaticamente o formato (separadores) utilizado no texto importado. Os formatos disponíveis são descritos em seções adicionais. Pressionando o botão "**OK**" a função de importação é executada. Pressionando o botão "**Cancelar** " a caixa de diálogo é fechada e sai do comando de importação. Os nomes utilizados para identificar os campos na primeira linha importada podem ser modificados, pressionando o botão "**Palavras-chave**". Este comando é descrito em seções adicionais.
A primeira linha do texto tem que conter o nome dos campos, de acordo com a ordem utilizada nas linhas seguintes. Cada objeto é descrito em uma linha de texto. O separador "fim de linha" (EL) é a seqüência MS-DOS padrão "**0d-0a**". Os campos podem aparecer em qualquer ordem. Se faltam alguns campos, eles são automaticamente preenchidos na descrição de objeto importada com valores padrões. Se um objeto importado já existe na lista editada, o usuário tem que confirmar que será sobreposto. A descrição de objeto é então atualizada com campos importados. Se alguns campos estão faltando, eles não serão atualizados na descrição de objeto.

Formato de texto disponível

A seguir está a lista de formatos disponíveis para o comando de exportação. O comando de importação reconhece estes formatos automaticamente.

- separador tabulação (tab)

Descrição: Os campos são separados através do caractere de tabulação.

Exemplo:

Nome	Atributo	Comentário
level	Interna	nível da água calc. internamente
alrm1	saída	saída de alarme

- separador vírgula

Descrição: Os campos são separados através de vírgulas.

Exemplo:

Nome,Atributo, Comentário
level,Interna, nível da água calc. internamente
alrm1,saída,saída de alarme

- separador ponto-e-vírgula

Descrição: Os campos são separados através de ponto-e-vírgulas.

Exemplo:
Nome;Atributo;Comentário
level;Interna; nível da água calc. internamente
alrm1;saída;saída de alarme

- vírgulas e aspas

Descrição: Os campos são separados através de vírgulas.
Cada campo é escrito entre aspas.

Exemplo:
"Nme","Atributo","Comentário"
"level","Interna"," nível da água calc. internamente "
"alrm1","output","saída de alarme"

Palavras chaves

Os nomes utilizados para identificar os campos na primeira linha importada ou exportada pode ser modificado, apertando o botão "**Palavras-chave**". Este comando abre a seguinte caixa de diálogo:

A janela mostra a lista de campos de objeto, e as palavras chaves associadas. Para modificar uma palavra chave, o usuário tem que selecionar um campo na lista e tem que apertar o botão "**Modificar**". A pressionar o botão "**Padrão**" a lista original de palavras chaves é restabelecida. Os nomes das palavras chaves têm que seguir as seguintes regras:

- o nome não pode exceder **16** caractere
- o primeiro caractere deve ser uma **letra**
- os caractere seguintes podem ser **letras**, **algarismos** ou o caractere sublinha '_'
- o mesmo nome não pode ser utilizado para palavras chaves diferentes

A seguir, as palavras chaves padrões encontradas no ISaGRAF:

Nome de objeto..... **Name**
Texto de comentário ..... **Comment**
Endereço de rede..... **Address**
Atributos (interno, entrada, saída) ..... **Attribute**
“string” booleana 'False'..... **False**

“string” booleana 'True'	**True**
Formato analógico (real ou inteiro)	**Format**
Seqüência unitária analógica	**Unit**
Nome de converasão analógica	**Conversion**
Comprimento máx. mensagem	**MaxLength**
Tipo de biblioteca de bloco de função	**Library**
Equivalência de palavra definida	**Equivalence**
Atributo interno	**Internal**
Atributo de entrada	**Input**
Atributo de saída	**Output**
Atributo constante	**Constant**
Formato analógico real	**Real**
Formato analógico inteiro	**Integer**

A.11 Utilização do editor de conexão de E/S

O objetivo da operação de conexão E/S é estabelecer uma ligação lógica entre as variáveis de E/S do aplicativo e os canais físicos das placas que existem na máquina destino. Para executar esta ligação o usuário tem que identificar e configurar todas as placas da máquina destino e colocar as variáveis de E/S nos canais de E/S correspondentes.
A lista à esquerda mostra o bastidor da máquina destino, com os **slots da placa**. Um slot pode ser livre ou utilizado por uma placa de E/S ou equipamento complexo. Cada slot é identificado por um **número de ordem**. O bastidor pode conter até **255** placas. A lista mostra os parâmetros da placa e as variáveis conectadas na placa selecionada à direita. Uma placa pode ter até **128** canais de E/S. O número total de placas elementares E/S (incluindo equipamentos únicos e placas de equipamentos complexos) não pode exceder **255**.

Ícones

Os ícones exibidos na face dianteira indicam o tipo e os atributos de variáveis que podem ser conectadas aos canais de placa. O sistema ISaGRAF não permite a conexão de variáveis de tipos diferentes na mesma placa. Este é o significado dos ícones utilizados:

.....tipo booleano
.....tipo inteiro/ real (ambos os tipos e variáveis podem ser conectadas)
.....tipo mensagem
.....entrada - nenhum canal conectado
.....saída - nenhum canal conectado
.....entradas - pelo menos um canal conectado
.....saídas - pelo menos um canal conectado

A seguir, os ícones utilizados para mostrar o tipo de dispositivo de E/S instalado em um slot:

.....equipamento E/S complexo
.....placa E/S real
.....placa E/S virtual

A seguir, os ícones utilizados para desenhar um parâmetro ou um canal:

.....parâmetro de placa
.....canal livre
.....canal conectado

Movendo uma placa de uma lista de placas

Utilize estes botões na barra de ferramentas ou os comandos de menu "**Editar / Mover placa acima**" ou "**Editar / Mover placa abaixo**" para mover a placa E/S selecionada uma linha para cima ou para baixo na lista principal. O comando "**Editar / Inserir slot**" insere um slot vazio na posição atual.

A.11.1 Definindo placas E/S

O menu "**Editar**" contém comandos básicos para definir a placa selecionada (configurar seus parâmetros) e conectar variáveis de E/S a seus canais.

Selecionando o tipo de placa E/S

Antes de conectar variáveis de E/S a uma placa, a identificação da placa deve ser informada. Uma biblioteca de placas predefinidas está disponível na área de trabalho ISaGRAF. Esta biblioteca pode ter sido compilada por um ou mais fornecedores de dispositivos de E/S. O comando "**Editar / Deteminar placa/equipamento**" é utilizado para configurar a identificação de placa. Este comando pode ser utilizado para selecionar ou uma única placa ou equipamento de E/S complexo da biblioteca ISaGRAF. Também é possível clicar duas vezes sobre um slot para selecionar a placa correspondente ou equipamento.
Todos os canais de uma única placa são do mesmo tipo (booleano, inteiro/real ou mensagem) e direção (entrada ou saída). As variáveis real e inteira não são diferenciadas durante a conexão E/S. Um equipamento de E/S complexo representa um dispositivo de E/S com canais de diferentes tipos ou direções. Um equipamento de E/S complexo é representado como uma lista de placas E/S únicas. Utiliza só um slot na lista de bastidor.

Removendo uma placa

O comando "**Editar / Limpar slot**" é utilizado para remover a placa atualmente selecionada ou equipamento de E/S. Se as variáveis já estão conectadas aos canais correspondentes, eles serão automaticamente desconectados ao apagar o slot.

Placas reais e placas virtuais

O comando "**Editar / Placa real/virtual**" estabelece a validade da placa selecionada ou equipamento de E/S complexo. Os seguintes ícones são exibidos na lista de bastidor para mostrar a validade de uma placa:

 .....placa E/S real
 .....placa E/S virtual

No modo Real (**Real Mode**), as variáveis de E/S são ligadas diretamente aos dispositivos de E/S correspondentes. As operações de entrada e de saída no programa aplicativo atrelam diretamente à entrada correspondente ou às condições de saída do campo atual de dispositivos de E/S. Em Modo Virtual (**Virtual Mode**), as variáveis de E/S são processadas exatamente como variáveis internas. Elas podem ser lidas ou podem ser atualizadas pelo depurador, de forma que o usuário pode simular o processamento de E/S, mas nenhuma conexão real é feita.

Observações técnicas

O comando "**Ferramentas / Observação Técnica**" mostra o manual de usuário online da placa selecionada ou do equipamento complexo. As observações técnicas da placa são escritas pelo fornecedor do hardware da placa de E/S. Contém toda a informação sobre o gerenciamento da placa E/S. Também descreve o significado de seus parâmetros.

Removendo variáveis conectadas

O comando "**Ferramentas / Liberar canais da placa**" desconecta todas as variáveis de E/S já conectadas na placa selecionada.

Definindo comentários para canais livres

O comando " **Ferramentas / Liberar canais da placa** " desconecta todas as variáveis de E/S já conectadas na placa selecionada.

A.11.2 Configurando os parâmetros da placa

Para configurar o valor de um parâmetro de placa, o usuário tem que clicar duas vezes sobre seu nome na lista à direita. Também é possível selecionar (destacar) e escolher o comando "**Determinar canal/parâmetro**" do menu "**Editar**". Os parâmetros são listados no começo da lista. O seguinte ícone é utilizado para representá-los na lista:

 .....parâmetro de placa

O significado e o formato de entrada do parâmetro são projetados pelo fornecedor da placa ou equipamento de E/S correspondente. Utilize o comando "**Ferramentas / Observação Técnica**" ou refira-se ao manual do seu hardware para mais informações sobre os parâmetros de placa.

A.11.3 Conectando os canais E/S

Para configurar a conexão de um canal, o usuário tem que clicar duas vezes sobre a sua localização na lista à direita. Também é possível selecionar (destacar) e escolher o comando "**Determinar canal/parâmetro**" do menu "**Editar**". Os seguintes ícones são utilizados para representar canais na lista:

 .....canal livre
.....canal conectado

A lista contém todas as variáveis que correspondem ao tipo de placa e direção selecionados. Somente as variáveis ainda não conectadas são listadas aqui. O botão "**Conectar**" conecta a variável selecionada na lista ao canal selecionado. O botão "**Desconectar**" remove (desconecta) a variável do canal selecionado. Os botões "**Próximo**" e "**Anterior**" são utilizados para selecionar outro canal da placa. O local do canal selecionado é sempre exibido no título da caixa de diálogo.

A.11.4 Variáveis diretamente representadas

Canais livres são aqueles que não estão ligados a uma variável de E/S declarada. O ISaGRAF permite a utilização de **variáveis diretamente representadas** na fonte dos programas para representar um canal livre. A identificação de uma variável diretamente representada sempre começa com o caractere "%".

A seguir, as convenções de nome de uma variável diretamente representada para um canal de uma única placa. "***s***" é o número do slot da placa. "***c***" é o número do canal.

%IX*s.c* .....canal livre de uma placa de entrada booleana
%ID*s.c* .....canal livre de uma placa de entrada inteira
%IS*s.c* .....canal livre de uma placa de entrada mensagem
%QX*s.c* .....canal livre de uma placa de saída booleana
%QD*s.c* .....canal livre de uma placa de saída inteira
%QS*s.c* .....canal livre de uma placa de saída mensagem

A seguir, as convenções de nome de uma variável diretamente representada para um canal de um equipamento complexo. "***s***" é o número do slot do equipamento. "***b***" é o índice da placa elementar dentro do equipamento complexo. "***c***" é o número do canal.

%IX*s.b.c* .....canal livre de uma placa de entrada booleana
%ID*s.b.c* .....canal livre de uma placa de entrada inteira
%IS*s.b.c* .....canal livre de uma placa de entrada mensagem
%QX*s.b.c* .....canal livre de uma placa de saída booleana
%QD*s.b.c* .....canal livre de uma placa de saída inteira
%QS*s.b.c* .....canal livre de uma placa de saída mensagem

A seguir, exemplos:

%QX1.6 6º canal da placa #1 (saída booleana)
%ID2.1.7 7º canal da placa #1 no equipamento #2 (entrada inteira)

Uma variável diretamente representada não pode ter dados do tipo "**real**".

A.11.5 Numeração

Utilize o comando "**Opções / Numbering**" para configurar as convenções de numeração. Você pode especificar o número utilizado para o primeiro slot e o número utilizado para o primeiro canal de cada placa na seguinte caixa de diálogo:

Como padrão, a numeração de slot começa com o índice **"0"** e a numeração de canal começa com o índice **"1"**.

Aviso: tenha muito cuidado durante a modificação das convenções de numeração já que tem efeito sobre os símbolos utilizados para as variáveis diretamente representadas e pode conduzir a

erros de compilação se as variáveis diretamente representadas de E/S são utilizadas em programas existentes.

A.11.6 Configurando proteções individuais

O ambiente de trabalho ISaGRAF fornece um sistema de proteção de dados completo baseado em senhas hierarquizadas. A conexão E/S pode ser totalmente protegida por uma senha. Adicionalmente, o ISaGRAF permite que você configure proteção individual para conexão E/S. Isto quer dizer:
- senhas já estão definidas no sistema de definição de senha (utilize o comando "**Projeto / Determinar senha**" da janela de Gerenciamento de Projeto) de modo que os níveis de proteção estão disponíveis para proteção individual.
- você utiliza os níveis de proteção com prioridade mais alta para proteção individual comparada com a proteção de E/S global.

Quando uma conexão E/S tiver proteção individual, um pequeno ícone é desenhado perto de seu nome na janela de conexão E/S:

Utilize os comandos "**Determinar proteção**" e "**Remover proteção**" do menu "**Editar**" para configurar ou remover uma proteção individual para a conexão selecionada. Ambos os comandos lhe pedem que entre com uma senha válida de forma que um nível de proteção possa ser atrelado à conexão. Então, toda vez que você quiser modificar uma conexão com proteção individual, você deve entrar com uma senha com nível de prioridade suficiente.

Aviso: . Se uma conexão é protegida com um nível e a senha correspondente é removida do sistema de proteção, e se nenhuma senha de nível mais alto estiver definida, a conexão não pode ser mais modificada a menos que uma nova senha com nível suficiente seja definida.

A.12 Criando tabelas de conversão

O ambiente de trabalho ISaGRAF permite que o usuário crie tabelas de conversão. Uma tabela de conversão é um conjunto de pontos utilizados para definir uma conversão analógica. Uma tabela de conversão pode ser atrelada a uma variável analógica de entrada ou de saída. Uma tabela cria uma relação proporcional entre valores elétricos (leitura do sensor de entrada ou envio ao dispositivo de saída) e valores físicos (utilizado em aplicativo de programação).
As tabelas de conversão são editadas através da caixa de diálogo executada pelo comando "**Ferramentas / Tabelas de conversão**" na janela dicionário ISaGRAF
Uma tabela de conversão definida pode ser utilizada para filtrar valores de qualquer variável analógica de entrada ou de saída do projeto selecionado. Para atrelar uma tabela de conversão a uma variável utilize os comandos do dicionário ISaGRAF, o editor de declaração de variáveis. Uma variável analógica de entrada ou de saída deve ser então selecionada e seus parâmetros editados. Uma variável não pode ser atrelada a uma tabela de conversão que ainda não está definida.

A.12.1 Comandos principais

A caixa de diálogo "**Tabelas de conversão**" mostra a lista de tabelas de conversão definidas, e contém os botões para os comandos principais, para editar uma tabela existente (definir seus pontos), para criar uma tabela nova e também renomear ou apagar uma tabela. Pressione OK para sair da caixa de diálogo "Tabelas de conversão" e salvá-las em disco.

Criando uma nova tabela

O comando "**Novo**" permite que o usuário crie uma nova tabela de conversão. Até **127** tabelas de conversão podem ser criadas para cada projeto. Somente tabelas utilizadas (aquelas atreladas às variáveis analógicas) são inseridas no código executável de aplicativo. O nome de uma tabela tem que seguir as seguintes regras:

- o nome não pode exceder **16** caracteres
- o primeiro caractere deve ser uma **letra**
- os caractere seguintes podem ser letras, algarismos ou o caractere '_'
- o nome de tabela não faz distinção entre maiúscula / minúscula

Modificando o conteúdo de uma tabela

O comando "**Editar**" é utilizado para inserir os pontos de uma tabela selecionada da lista. Também é possível clicar duas vezes sobre o nome da tabela. O comando "**Editar**" é chamado automaticamente quando uma nova tabela é criada. Pelo menos dois pontos devem ser inseridos em cada tabela.

A.12.2 Pontos de entrada de uma tabela

A caixa de diálogo "**Editar**" permite que o usuário defina os pontos de uma tabela de conversão. A caixa mostra a lista de pontos definidos à esquerda. A caixa no canto inferior direito mostra a tabela definida como uma curva gráfica. Os pontos são inseridos utilizando os comandos de caixa. O usuário tem que obedecer às regras de número para a definição de

pontos, descrito no fim deste capítulo. A caixa à esquerda sempre contém a lista de pontos existentes para a tabela atualmente editada. A coluna à esquerda mostra o valor elétrico (externo) dos pontos. A coluna à direita mostra os valores físicos (internos). O usuário tem que selecionar um ponto na lista para modificar seus valores ou apagá-lo (remover). A última escolha da lista ("**... ...**") é utilizada para definir um novo ponto. A caixa no canto inferior direito mostra a tabela atualmente editada como uma curva gráfica. Nenhum eixo ou coordenadas são mostrados já que é uma representação proporcional da curva. Esta representação é útil como um verificação rápida de que a curva está corretamente definida.

- ***Definindo um novo ponto***

Ao definir um novo ponto, selecione a última entrada ("**... ...**") na lista de pontos. Este também é o modo padrão para começar a definir uma nova tabela de conversão. O usuário tem que inserir os valores elétrico (externo) e físico (interno) de cada ponto. Os valores são armazenados como números de ponto flutuante de precisão simples. Lembre-se que pelo menos **dois pontos** têm que ser inseridos para definir uma curva. Quando ambos os valores são inseridos, ao pressionar o botão "**Armazenar**" um ponto é acrescentado à tabela. Um máximo de **32** pontos pode ser definido para cada tabela de conversão.

- ***Modificando um ponto***

Para modificar os valores de um ponto existente, primeiro selecione-o na lista. Os novos valores do ponto, elétrico (externo) e físico (interno), podem ser então inseridos. Os valores são armazenados como números de ponto flutuante de precisão simples. Quando ambos os valores são inseridos, ao pressionar o botão "**Armazenar**" o ponto na tabela é atualizado.

- ***Apagando um ponto***

Um ponto existente é apagado selecionando-o na lista e pressionando o botão "**Limpar**". Lembre-se que pelo menos **dois pontos** devem ser inseridos para definir uma tabela.

A.12.3 Regras e limites

As regras mostradas abaixo devem ser seguidas durante a definição de uma tabela de conversão. A tabela pode ser utilizada para converter variáveis analógicas de entrada e de saída:

- Dois pontos não podem ser definidos com o mesmo valor elétrico
- A curva deve ser continuamente crescente ou decrescente
- Dois pontos não podem ser definidos com o mesmo valor físico

Os seguintes limites são aplicados na definição das tabelas de conversão para um projeto:

- Não mais que **127** tabelas de conversão podem ser definidas no mesmo projeto
- Não mais que **32** pontos podem ser definidos para a mesma tabela de conversão.

A.13 Utilização do gerador de código

A janela de geração de código é aberta automaticamente pelos comandos "Verificar" e "Compilar" das outras janelas do ambiente de trabalho ISaGRAF. A janela de geração de código não é fechada automaticamente quando a operação de geração de código solicitada termina, de forma que o usuário ainda tem acesso a todos os comandos de geração de código e opções da janela de menu.

A.13.1 Comandos principais

O menu "**Arquivo**" contém os comandos para a verificação da sintaxe de programa e geração de código.

Gerar o código do aplicativo

O comando "**Compilar**" constrói o código inteiro do projeto. Antes de gerar qualquer coisa, este comando confere a sintaxe das declarações e programas. Qualquer erro que não pode ser descoberto durante uma única compilação de programa é descoberto durante a geração de código. Isto se aplica a tabelas de conversão, variáveis de conexões E/S e ligações com as bibliotecas. A geração de código interrompe a compilação de um programa quando erros são detectados. Este programa deve ser corrigido antes de continuar a geração de código. Programas que já tenham sido conferidos (sem erro detectado) e que não tenham sido modificados desde a sua última operação "**Verificar**" não são recompilados. A verificação de declaração de variável e a verificação da coerência do aplicativo sempre são processados. Durante a verificação de programa, a operação "**Compilar**" pode ser abortada pressionando a tecla **ESCAPE** (ESC).

Nota: Se a declaração de uma variável local de um programa foi modificada, este programa é verificado. Se uma variável global foi modificada, todos os programas são verificados.

Verificação da sintaxe do programa

O comando "**Verificar programa**" permite que o usuário verifique somente um programa. O programa selecionado é compilado até mesmo se não foi modificado desde a sua última verificação. O comando "**Verificar Dicionário**" permite que o usuário verifique as declarações de todas as variáveis do projeto.
O comando "**Verificar todos os programas**" verifica a sintaxe de todos os programas do projeto, até mesmo se alguns deles não foi modificado. Este comando **não** é interrompido quando um erro é detectado em um programa. Pode ser utilizado para produzir uma listagem completa de todos os erros que permanecem em programas do projeto. Este comando pode ser abortado pressionando a tecla de **ESCAPE** (ESC).

Simulando uma modificação

O comando "**Simular modificação**" simula uma modificação de todos os programas do projeto, de modo que todos são verificados durante a próxima operação "**Compilar**". O comando "**Abrir**" é utilizado para abrir o último programa verificado. Este comando é muito útil para o acesso direto a um programa no qual foram detectados erros de sintaxe.

A.13.2 Opções de compilador

O comando "**Opções do compilador**" é utilizado para configurar os principais parâmetros utilizados pelo Gerador de Código ISaGRAF para construir e aperfeiçoar o código destino. O objetivo principal deste comando é selecionar o tipo de código que tem que ser gerado, de acordo com os objetivos dos ISaGRAF destinos correspondentes, e para configurar os parâmetros de otimização de acordo com o tempo de compilação esperado e as necessidades de tempo de execução do aplicativo.
O botão "**Carregar**" abre uma segunda caixa de diálogo com outras opções que habilitam a incorporação do código fonte ao código carregado, para habilitar a característica "Carregar" . Refira-se à documentação “Carregar" para explicações adicionais.

- ***Selecionando destinos***

A lista superior mostra a lista de códigos de destinos disponíveis que podem ser produzidos. O sinal ">>" é utilizado para indicar o(s) destino(s) selecionado(s). O Gerador de Código ISaGRAF pode produzir até 3 códigos diferentes na mesma operação de compilação. Utilize os botões "**Selecionar**" e "**Anular seleção**" para determinar a lista de códigos destinos exigidos, de acordo com o seu hardware destino. A seguir, os destinos ISaGRAF padrões:

SIMULATE:.....Este código é dedicado ao Simulador ISaGRAF na Área de Trabalho (“Workbench”). O simulador não pode ser executado se este destino não está selecionado para produzir o código de aplicativo.
ISA86M: .....Este é um código TIC (Target Independent Code) dedicado aos núcleos ISaGRAF instalados em processadores baseados nos processadores Intel. O tipo de processador só considera a ordenação de byte no código gerado.
ISA68M: .....Este é um código TIC (Target Independent Code) dedicado aos núcleos ISaGRAF instalados em um processador baseado em processador Motorola. O tipo de processador só considera a ordenação de byte no código gerado.
SCC:.....A seleção deste destino leva o compilador ISaGRAF a produzir código fonte em linguagem "C" estruturado para ser compilado e ligado às bibliotecas do núcleo destino ISaGRAF para produzir um código executável incorporado
CC86M: .....A seleção deste destino leva o compilador ISaGRAF a produzir código fonte em linguagem "C" não estruturada para ser compilado e ligado às bibliotecas do núcleo destino ISaGRAF para produzir um código executável incorporado. Esta seleção é fornecida para compatibilidade com versões ISaGRAF anteriores a V3.23, quando a geração e a integração de códigos "C" estruturados não eram suportadas.

Refira-se a seu manual de hardware para conhecer o tipo de núcleo destino ISaGRAF instalado em seu PLC. Outros tipos de destinos (código de máquina, código fonte C...) poderão ser suportados em futuras versões de ambiente de trabalho ISaGRAF.

- ***Processamento SFC***

Verifique a caixa "**Processador SFC do usuário embutido**" para habilitar a utilização da máquina SFC ISaGRAF. ***Este modo deve ser preferido já que conduz a desempenhos mais altos de tempo de execução***. Porém, a máquina destino pode estar perdendo em algumas

implementações particulares do ISaGRAF destino, mais comumente nos destinos personalizados baseados no processamento de código postal ISaGRAF. Neste caso você pode ter que remover esta opção e deixar o compilador ISaGRAF traduzir os gráficos SFC com instruções de baixo nível. Refira-se a sua documentação de hardware para mais informação sobre a utilização desta opção.

Opções do otimizador

|A seguir, estão os parâmetros utilizados pelo Gerador de Código ISaGRAF para otimizar o código destino que pode ser configurado a partir da caixa de diálogo "**Opções do compilador**". O botão "**Padrão**" é utilizado para remover todas as opções de otimização para reduzir o tempo de compilação.

- Quando a opção "**Executar dois passos de otimização**" é configurada, o Otimizador de Código ISaGRAF é executado duas vezes. Otimizações feitas durante a segunda passagem geralmente são menos significantes que aquelas feitas na primeira passagem.

- Quando a opção "**Avaliar expressões constantes**" é configurada, expressões constantes são avaliadas pelo compilador. Por exemplo, a expressão numérica " **2 + 3** " é substituída pelo " **5** " no código destino. Quando esta opção não é configurada, as expressões constantes são calculadas durante o tempo de execução.

- Quando a opção "**Suprimir rótulos não utilizados**" é configurada, o Otimizador simplifica o sistema de desvios e etiquetas dos programas, de modo a suprimir etiquetas de destinos não utilizadas ou desvios nulos.

- Quando a opção "**Otimizar cópia de variáveis**" é configurada, a utilização de variáveis temporárias (utilizadas para armazenar resultados intermediários) é otimizada. Esta opção é comumente utilizada com a opção "**Otimizar expressões**". Quando esta opção está configurada, o Otimizador reutiliza o resultado de expressões e subexpressões que são utilizadas mais de uma vez no programa.

- Quando a opção "**Suprimir código não utilizado**" está configurada, o Otimizador suprime o código que não é significante. Por exemplo, se as seguintes declarações estão programadas: "**var := 1; var := X;**", o código gerado correspondente é só : "**var := X;**".

- Quando a opção "**Otimizar operações aritméticas**" está configurada, o Otimizador simplifica as operações aritméticas de acordo com os operandos especiais. Por exemplo, a expressão " **A + 0** " será substituída pelo " **A** ". Quando a opção "**Otimizar operações booleanas**" está configurada, o Otimizador simplifica as operações booleanas de acordo com os operandos especiais. Por exemplo, a expressão booleana " **A & A** " será substituída por " **A** ".

- Quando a opção "**Construir diagramas em decisão binária (BDDs)**" está configurada, o Otimizador substitui as equações booleanas (misturando operadores **AND**, **OR** , **XOR** e **NOT**), por uma lista reduzida de operações de desvios condicionais. A tradução só é operada se o tempo de execução esperado da seqüência de desvios for menor que aquela esperada para a expressão original.

A tabela a seguir resume a otimização esperada e o tempo de compilação solicitado para cada parâmetro:

	ganho (desempenhos)	t. compilação
Executa 2 passos	**xxxx**	(*)
Otimiza expressões constantes	**xxxxxxxx**	**xxxx**
Suprime etiquetas não usadas	**xxxx**	**xxxxxxxx**
Otimiza cópia de variável	**xxxx**	**xxxxxxxx**
Otimiza expressões	**xxxx**	**xxxxxxxx**
Suprime códigos não usados	**xxxx**	**xxxxxxxx**
Otimiza operações aritméticas	**xxxxxxxx**	**xxxx**
Otimiza operações booleanas	**xxxxxxxx**	**xxxx**
Gera diagramas decisão binária	**xxxxxxxxxxx**	**xxxxxxxxxxx**

(*)o tempo (t) de compilação é também multiplicado por 2.

A.13.3 Geração do código fonte em linguagem “C”

O ambiente de trabalho ISaGRAF habilita a saída de código de fonte em "linguagem C". Neste caso, todo o conteúdo do aplicativo, incluindo a descrição de gráfico SFC, definição de banco de dados e seqüências de código são gerados no formato de código fonte em "C". Há duas possibilidades, propostas como dois estilos de código gerado:

CC86M .....(código fonte em C - V3.04) produz código fonte em “C” não estruturado. Este estilo deve ser selecionado se seu software destino está baseado em ISaGRAF versão anterior a 3.23.

SCC.....(código fonte em C estruturado) produz código fonte em “C” estruturado. Este estilo deve ser preferido se seu software destino está baseado em ISaGRAF versão 3.23 ou posterior.

Os dois arquivos a seguir são criados no diretório de projeto:
APPLI.C .....código fonte comum do aplicativo
APPLI.H.....definições de linguagem "C" comuns

No caso de geração de código fonte "C" estruturado, um arquivo fonte "**.C**" e um arquivo de definição "**.H**" são criados para cada programa do aplicativo, além dos arquivos comuns “**APPLI.C**" e "**APPLI.H**". Estes arquivos devem ser compilados e ligados às bibliotecas destino ISaGRAF para produzir o código executável final. Refira-se ao "Guia de usuário do kit de ferramentas de desenvolvimento ISaGRAF E/S" para informação adicional sobre as técnicas de implementação indicadas.

Nota: Algumas características de depuração tais como carga do aplicativo, modificação online e pontos de parada (“breakpoints”) não estão mais disponíveis quando o aplicativo ISaGRAF é compilado em "C".

A.13.4 Visualizando a informação

O menu "**Editar**" contém os comandos para a visualização de diferentes arquivos texto construídos durante as operações de geração do código ou verificação de sintaxe na janela de gerador de código. A janela de geração de código é uma área de texto que contém mensagens durante as operações de geração de código ou verificação de sintaxe. Toda a informação é armazenada no disco de modo que possa ser examinada utilizando os comandos do menu "**Editar**".

- ***Comandos de edição***

O comando "**Apagar**" é utilizado para limpar a área de texto da janela. A janela é limpa automaticamente antes de cada operação de geração de código ou verificação de sintaxe. O comando "**Copiar**" é utilizado para copiar o texto exibido na área de transferência do Windows, podendo assim ser utilizado através por outros aplicativos tais como editores de texto ISaGRAF.

- ***Visualização das mensagens de saída do compilador***

O comando "**Mensagens de execução**" mostra todas as mensagens exibidas durante a última operação "**Compilar**" ou "**Verificar**" na área de texto da janela. Isto se aplica a todas as mensagens de erro.

Outras opções do menu "**Editar**" permitem que o usuário monitore arquivos de texto auxiliares criados durante a verificação de sintaxe e a geração de código. Estes arquivos normalmente não são utilizados para um projeto ISaGRAF comum.

A.13.5 Definindo recursos

O comando "**Recursos**" do menu "**Opções**" permite que o usuário defina recursos. Um recurso são quaisquer dados definidos pelo usuário (configuração de rede, de hardware...) de qualquer formato (arquivo, lista de valores) que tenham que ser combinados com o código gerado e com ele ser carregado no PLC destino. Tais dados não são operados diretamente pelo núcleo ISaGRAF e são normalmente dedicados a outros softwares instalados no PLC destino. Refira-se ao seu manual de hardware para informação adicional sobre recursos disponíveis.

- ***Arquivo de definição de recurso***

Os recursos estão definidos em um "**Arquivo de definição de recurso**" armazenado com outros arquivos do projeto ISaGRAF. Este é um arquivo texto ASCII puro, processado pelo Compilador de Recurso ISaGRAF. Este compilador é executado automaticamente quando o código de aplicativo é construído. Esta seção explica a sintaxe deste arquivo. O arquivo de definição de recurso utiliza regras léxicas da linguagem ST. Comentários, começando com os caracteres"**(***" e terminando com "***)**" podem ser inseridos em qualquer lugar no texto. "strings" são delimitadas através de apóstrofes simples. Refira-se à segunda parte deste manual para mais explicações sobre os formatos léxicos utilizados para inserir valores numéricos.

- ***Linguagem de referência***

A seguir, a lista de palavras chaves e declarações utilizadas em um arquivo de definição de recurso.

ULONGDATA

Significado: Especifica um recurso que é uma lista de valores inteiros. Os valores são armazenados no código destino como inteiros sem sinal de 32 bits. Os valores são armazenados na ordem especificada no arquivo de definição de recurso. Os valores devem ser separados através de vírgulas. O nome do recurso não pode exceder 15 caracteres.

Sintaxe:
ULONGDATA '<nome_recurso>**'**
BEGIN
...seleção_destino...
...lista de valores...
END

Exemplo:
```
ULongData 'MYDATA'
Begin
  ...
  0, -1, 100_000,              (* decimal *)
  16#A0B1, 2#1011_0101         (* hexadecimal, binário *)
End
```

VARLIST

Significado: Especifica um recurso que é uma lista de endereços de variáveis. As variáveis são identificadas por seus nomes no arquivo de definição de recurso. Os endereços de variáveis são armazenados no código destino como inteiros sem sinal de 16 bits. Os endereços são armazenados na ordem especificada no arquivo de definição de recurso. As variáveis devem ser separadas por vírgulas. O nome do recurso não pode exceder 15 caracteres.

Sintaxe:
VARLIST '<nome_recurso>**'**
BEGIN
...seleção_destino...
...lista de nomes de variáveis...
END

Exemplo:
```
VarList 'LIST'
Begin
  ...
  Var100, MyParameter, Command, Alarm
End
```

BINARYFILE

Significado: Especifica um recurso Arquivo Binário. Os dados de fonte são armazenados em um arquivo MS-DOS. A definição do recurso destino é complementada

com um nome do caminho destino. Caracteres de fim de linha (EL) não são convertidos pelo Compilador de Recurso ISaGRAF. O nome do recurso não pode exceder 15 caracteres.

Sintaxe:
```
BINARYFILE '<nome_recurso>'
BEGIN
  ... seleção_destino...
  FROM '<nome_do_caminho_fonte>'
  TO '<nome_do_caminho_destino>'
END
```

Exemplo:
```
BinaryFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.bin'
  To '/dd/user/appl/config.dat'
End
```

TEXTFILE

Significado: Especifica um recurso Arquivo Texto. Os dados de fonte são armazenados num arquivo ASCII. A definição de recurso destino é complementada com um nome do caminho destino. Os caracteres de fim de linha (EL) são convertidos pelo Compilador de Recurso ISaGRAF de acordo com as convenções do sistema "host" destino. O nome do recurso não pode exceder **15** caracteres.

Sintaxe:
```
TEXTFILE '< nome_recurso >'
BEGIN
  ... seleção_destino...
  FROM '< nome_do_caminho_fonte>'
  TO '<nome_do_caminho_destino>'
END
```

Exemplo:
```
TextFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.bin'
  To '/dd/user/appl/config.dat'
End
```

TARGET

Significado: Especifica o nome do código destino que tem que incluir o recurso. Refira-se à seção anterior (opções de compilador) para informação adicional sobre destinos controlados. A declaração "**Target**" pode aparecer mais de uma vez no mesmo bloco de recurso para selecionar vários destinos. Esta declaração não pode ser usada se a declaração "**AnyTarget**" está especificada.

Sintaxe: **TARGET '**<nome_destino>**'**

Exemplo:
```
BinaryFile 'MYFILE'
Begin
  Target 'ISA86M'
  Target 'ISA68M'
  ...
End
```

ANYTARGET

Significado: Especifica que o recurso deve ser combinado com todos os códigos destino criados pelo Gerador de Códigos. O Gerador de Código ISaGRAF pode produzir vários códigos destino durante o mesmo comando "**Compilar**". Esta declaração não pode ser utilizada se uma ou várias declarações "**Target**" são especificadas.

Sintaxe: **ANYTARGET**

Exemplo:
```
ULongData 'MYDATA'
Begin
  AnyTarget
  ...
End
```

FROM

Significado: Especifica o nome do caminho fonte (no PC em que o Ambiente de Trabalho ISaGRAF está instalado) de um recurso **BinaryFile** ou **TextFile**. Os caracteres utilizados para isolar os componentes do nome do caminho (unidade, diretório, prefixo, sufixo) tem que estar conforme as convenções do sistema MS-DOS.

Sintaxe: **FROM '**< nome_do_caminho_destino>**'**

Exemplo:
```
BinaryFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.dat'
  To '/dd/user/appl/config.dat'
End
```

TO

Significado: Especifica o nome do caminho destino (no sistema destino) de um recurso **BinaryFile** ou **TextFile**. Os caracteres utilizados para isolar os componentes do nome do caminho (unidade, diretório, prefixo, sufixo) têm que estar conforme as convenções do sistema host destino.

Sintaxe: **TO '**<nome_do_caminho_destino>**'**

Exemplo:
```
TextFile 'MYFILE'
Begin
  ...
  From 'c:\user\config.dat'
  To '/dd/user/appl/config.dat'
End
```

Exemplo

A seguir, um exemplo completo de um arquivo de definição de recurso:

```
(* arquivo de definição de recursos *)

ULongData 'DATA1'                    (* lista de valores *)
Begin
  Target 'ISA86M'                    (* apenas para este alvo*)
  1, 0, 16#1A2B3C4D, +1, -1          (* valores numéricos *)
End

VarList 'VLIST1'                     (* lista de variáveis *)
Begin
  Target 'ISA86M'                    (* apenas para este alvo*)
  Valve1, StateX, Command, Alrm1     (* nomes das variáveis *)
End

BinaryFile 'FILE1'                   (* arquivo recursos binários *)
Begin
  AnyTarget                          (* dedicado para todos alvos *)
  From 'c:\user\updatef.bin'         (* arquivo fonte no PC *)
  To 'updatef.cfg'                   (* arquivo alvo no PLC *)
End

TextFile 'FILE2'                     (* arquivo texto de recursos *)
Begin
  Target 'ISA68M'
  From 'c:\nw\nwbd.txt'              (* arquivo fonte no PC *)
  To '/nw/dat/nwbd'                  (* arquivo alvo no PLC *)
End
```

Compilar recursos

Se recursos foram inseridos em arquivo de definição de recurso, uma caixa de diálogo é apresentada ao término da geração de código ISaGRAF. Pressione o botão "**Compilar**" para executar o compilador de recurso. Mensagens de saída e erros serão exibidas no controle principal. Pressione "**Sair**" para evitar a compilação de recurso. Neste caso, não serão acrescentados recursos ao código ISaGRAF.

Implementação

O número de recursos, o tamanho das linhas de dados e arquivos não são limitados pelo ISaGRAF. Os recursos são armazenados ao término do código gerado, com um diretório de recurso. A seguir, o formato (utilizando notações da linguagem C) do diretório de recurso:

```
__RESOURCE:
{
  long nbres;                          /*número de recursos definidos */
  {
    char name[16];                     /* nome do recurso */
    long type;                         /* tipo de dado do recurso */
    long size;                         /* tamanho exato do blodo de dados */
    void *data;
    char *path_offset;                 /* aponta para uma string */
  } /* número de gravações */
}
```

A seguir, os possíveis valores do campo "tipo" :

- 1 = arquivo binário
- 2 = arquivo de texto
- 3 = lista de valores (o campo de path_offset não é utilizado neste caso)
- 4 = lista de variáveis (o campo de path_offset não é utilizado neste caso)

Para arquivos texto, os caracteres de fim de linha (EL) são traduzidos pelo compilador de recurso, de acordo com as convenções do sistema destino. Todos os ponteiros são “offsets” de 32 bits do endereço da estrutura correspondente. Todos os nomes de recurso e nomes de caminhos são “strings” terminadas por NULL. Os nomes dos caminhos e os dados seguem o diretório de recurso.

A.14 Referências cruzadas

O ambiente de trabalho ISaGRAF inclui um editor de referências cruzadas que proporciona ao usuário uma visão total das variáveis declaradas nos programas do projeto e em quais são utilizados. O objetivo da referência cruzada é listar todas as **variáveis** declaradas no projeto e **localizar**, na fonte de cada programa, as partes do código fonte em que essas variáveis são utilizadas. As referências cruzadas são muito úteis para uma visão global de um ciclo de vida da variável. Eles ajudam a localizar efeitos colaterais e a reduzir o tempo para entender o projeto durante a manutenção. As referências cruzadas também podem ser utilizadas para uma visão global do dicionário completo de um projeto, de modo que variáveis não utilizadas são facilmente encontradas e a complexidade do projeto medida.
A lista à esquerda mostra os objetos declarados do projeto (programas, variáveis e palavras definidas), e os elementos de biblioteca (funções e blocos de função) referenciadas no projeto. A lista à direita mostra as ocorrências nos programas do objeto atualmente selecionado na primeira lista.
A descrição de um ocorrência inclui o nome do programa, o número da etapa FC ou SFC, transição ou teste, mais o número de linha para as linguagens de texto ou coordenadas para LD ou diagramas FBD. Para diagramas Quick LD, a descrição é complementada com o número da lógica. Se a variável é utilizada como uma saída (em uma bobina) o número da lógica é seguido por um caractere estrela ("*").
Configure a opção "**Exibir variáveis não utilizadas**" do menu "**Opções**" para também exibir na lista principal as variáveis que não estão sendo utilizadas nos programas do aplicativo.

Seleção do tipo de objeto

Porque um projeto pode agrupar um número enorme de objetos declarados, a caixa “combo” na barra de ferramentas do editor é utilizada para selecionar o tipo de objetos que devem ser listados na janela. Isto permite que o usuário tenha acesso à informação selecionada.
Toda vez que as referências cruzadas são recalculadas, a seleção é reajustada para "**Todos**" de modo a apresentar a lista completa.

Recalcular as referências cruzadas

O comando "**Arquivo / Recalcular**" pode ser utilizado a qualquer hora para atualizar as referências cruzadas de acordo com as modificações inseridas em outras janelas de edição ISaGRAF.

Exportar referências cruzadas

O comando "**Ferramentas / Exportar (arquivo texto)**" é utilizado para escrever a listagem completa das referências cruzadas em um arquivo texto ASCII. Este arquivo pode então ser aberto com outros aplicativos tais como Windows “NotePad” (Bloco de Notas) ou processadores de textos.

Dicionário de erros

O comando "**Editar / Erros no dicionário**" mostra em uma caixa de diálogo a lista de erros detectados quando o dicionário de projeto foi carregado.

Estatísticas

O comando "**Ferramentas / Estatísticas**" exibe em uma caixa de diálogo o número de objetos e variáveis declaradas no projeto, de acordo com os tipos de variável e de atributos. Uma aplicativo particular deste comando é saber o número de variáveis de E/S declarado no projeto para assegurar que pode ser compilado, se uma versão limitada do ambiente de trabalho ISaGRAF for utilizada.

Localizar na lista de objeto

O comando "**Editar / Localizar**" permite que o usuário selecione diretamente um objeto na lista de editor. O objeto procurado não pode ser localizado se não está listado de fato (ao usar uma exibição selecionada). É recomendado, antes de procurar um objeto, ativar a opção "**todos**" na barra de ferramentas.

Abrir o programa

A lista à direita contém as ocorrências do objeto selecionado nos arquivos fonte e a conexão E/S do projeto aberto. O comando "**Editar / Abrir programa**" permite que o usuário abra um programa diretamente no lugar em que o objeto aparece. Também é possível clicar duas vezes com o mouse sobre uma ocorrência (na lista de ocorrências) para abrir o programa correspondente.

A.15 Utilizar o depurador gráfico

O ISaGRAF inclui um depurador gráfico e simbólico completo. O comando "Depurar" da janela de gerenciamento de programa executa o depurador para controlar o aplicativo carregado no PLC destino. Neste modo, o depurador comunica-se com o sistema destino através de ligação de hardware. O comando "**Simular**" da janela de gerenciamento de programa simultaneamente executa o depurador e um simulador de destino completo. Isto permite que o usuário teste o seu aplicativo quando o sistema de E/S do destino ainda não está completo. A janela de depurador contém os comandos para controlar todo o aplicativo.
Quando o depurador inicia, e se o aplicativo no PLC destino é igual àquele do ambiente de trabalho, automaticamente abre a **janela de gerenciamento de programa**, no modo depurar. Os comandos desta janela podem ser utilizados para abrir outras janelas ISaGRAF (editores de gráfico e de texto, dicionário, listas de variáveis, conexão de E/S...). Todas as janelas abertas durante uma sessão de depuração operam no "**modo de depuração**", quer dizer, o comando de edição está desabilitado. Os componentes de programa exibidos (etapas, transições, variáveis...) são mostrados com o seu estado ou valor de tempo de execução atual. Clicando duas vezes sobre um objeto o seu estado ou valor é modificado no aplicativo destino.
Ao executar o depurador em **modo de simulação**, a comunicação com o sistema destino ISaGRAF é interrompida. O depurador só se comunica com a janela de simulador. Porque o sistema destino não existe neste modo, os comandos "**Descarregar**", "**Parar**" ou "**Ativar**" não estão disponíveis no menu de depurador.

A.15.1 A janela Depurador

A janela de depurador só contém informação sobre o estado do aplicativo completo. Está ligada a outras janelas ISaGRAF criando um sistema de depuração interativo completo. Os erros de tempo de execução detectados são exibidos na área inferior da janela de depurador. Os comandos do menu "**Opções**" são utilizados para esconder, mostrar ou remover a lista de erros.

O painel de controle (área abaixo do menu de depurador) mostra o estado global do aplicativo destino e a informação sobre o tempo de ciclo de execução. A lista de possíveis estados destino é a seguinte:

Conectando:..... Depurador estabelece comunicação com o sistema destino.
Desconectado:..... Depurador não pode comunicar-se com o sistema destino. Certifique-se que o cabo de conexão e os parâmetros de comunicação são válidos.
Aplicação inexistente:..... A conexão está OK, mas nenhum aplicativo ISaGRAF atualmente existe no sistema destino. Carregue um aplicativo.
Aplicação ativa:..... A conexão está OK e um aplicativo ativo existe no sistema destino. O depurador está agora estabelecendo as comunicações com este aplicativo, se é o mesmo do Ambiente de Trabalho.
EXECUTAR:..... Aplicativo destino está no modo "Tempo Real".
PARAR:..... Aplicativo destino está no modo "Ciclo a Ciclo".
BreakPoint: ..... Aplicativo destino está no modo "Ciclo a Ciclo", porque um ponto de parada (“breakpoint”) foi encontrado.

Fatal Error:..... Aplicativo destino falhou porque um erro sério ocorreu.

Informação sobre o tempo de ciclo de tempo de execução é a seguinte:
Permitido:..... tempo programado.
Atual: ..... tempo exato do último ciclo de execução completo.
Máximo:..... tempo máximo detectado desde o início do aplicativo.
Overflow:..... número de ciclos de execução detectado com um tempo maior que o permitido.
Todos os valores de tempo são determinados em milisegundos (ms). Os valores de tempo não são exibidos quando o depurador for utilizado em modo de simulação.

A.15.2 Controlando o aplicativo

Os menus "**Arquivo**" e "**Controle**" contêm os comandos para a instalação e o controle do aplicativo ISaGRAF atualmente editado no sistema ISaGRAF destino.

Nota: Alguns destes comandos não estão disponíveis durante simulação, porque o aplicativo processado pelo simulador é instalado automaticamente pelo Ambiente de Trabalho ISaGRAF.

Parar o aplicativo destino

O comando "**Arquivo / Parar aplicação**" interrompe a execução do aplicativo atualmente ativo no sistema ISaGRAF destino.

Ativar o aplicativo destino

O comando "**Arquivo / Iniciar aplicação**" executa o aplicativo existente no sistema destino. Quando um aplicativo é carregado, ele é automaticamente iniciado, de forma que o comando "**Iniciar**" não tenha que ser utilizado. O comando "**Iniciar**" é tipicamente utilizado após um comando "**Parar**".

Nota: o aplicativo destino deve ser interrompido (inativo) antes que seja possível carregar um novo aplicativo.

Carregar o aplicativo

O comando "**Arquivo / Descarregar**" é utilizado para carregar o código de aplicativo no sistema destino. Selecione o tipo de código a ser carregado, de acordo com o processador de sistema destino e as opções de aplicativo.

Apresentar o número da versão

O comando "**Arquivo / Obter número de versão**" é utilizado para exibir a identificação completa do Ambiente de Trabalho e aplicativos de destino. O aplicativo de Ambiente de Trabalho é aquele correntemente aberto no ambiente de trabalho ISaGRAF. O aplicativo destino é o executada no ISaGRAF PLC destino. Os seguintes itens são exibidos:
VERSION:..... Este é o número da versão do código do aplicativo. Este número foi calculado pelo gerador de código.
DATE:..... Este item mostra a data e a hora da criação do código.
CRC: ..... Esta é a soma de verificação (checksum) calculada com o conteúdo da tabela de símbolo. Este número foi calculado pelo

gerador de código. Este valor depende do conteúdo do dicionário de variáveis.

Nota: O comando "**Obter número de versão**" também está disponível durante a simulação. No modo depuração real, este comando não pode ser utilizado se o PLC destino não estiver conectado.

Modificação online

O comando "**Arquivo / Realiza atualização**" permite que o usuário utilize a "modificação online" do aplicativo destino em execução. Este comando está detalhado em seções adicionais deste capítulo. Não está disponível quando o depurador for utilizado em modo de simulação.

Modo Tempo Real

O comando "**Controle / Tempo real**" não está disponível quando nenhum aplicativo está ativo. Configura o aplicativo destino no modo "tempo real" normal : modo Normal: os ciclos de execução são sincronizados pelo tempo de ciclo programado.

Modo Ciclo a Ciclo

O comando "**Controle / Ciclo a ciclo**" não está disponível quando nenhum aplicativo está ativo. Configura o aplicativo destino no modo "ciclo a ciclo" normal: Neste modo, os ciclos são executados um por um, de acordo com os comandos "**Executar um ciclo**" feitos pelo usuário a partir do menu depurador.

Executar um ciclo

Quando o destino está no modo ciclo a ciclo, o comando "**Controle / Executar um ciclo**" executa um ciclo.

O tempo de ciclo

O comando "**Controle / Alterar tempo de ciclo**" permite que o usuário modifique o tempo de ciclo programado. Este tempo é entitulado como "**Permitido**" na barra de controle da janela de depurador. O modo "**Ciclo a ciclo**" deve ser configurado antes de modificar o tempo de ciclo. O tempo de ciclo é inserido como um número inteiro em milisegundos (ms).

Remover todos os pontos de parada (breakpoints)

O comando "**Controle / Limpar todos breakpoints**" remove todos os breakpoints instalados atualmente (encontrados ou ainda ativos) em todo o aplicativo. Os breakpoints existentes não são removidos automaticamente quando a janela de depurador está fechada.

Desbloquear variáveis E/S

O comando "**Controle / Destravar todas as variáveis de E/S**" desbloqueia todas as variáveis E/S atualmente bloqueadas no aplicativo. Quando uma variável E/S é fechada, nenhuma modificação de estado de entrada ou de saída é feita ao dispositivo E/S correspondente. As variáveis atreladas ao E/S ainda podem ser escritas pelo aplicativo ou pelo depurador. Variáveis E/S atualmente bloqueadas não são automaticamente desbloqueadas quando a janela de depurador está fechada.

A.15.3 Opções

O menu "**Opções**" contém as opções para controlar a informação exibida na janela de depurador.

Os parâmetros de comunicação

Os parâmetros de tempo de comunicação podem ser ajustados quando o depurador está ativo. Somente os "**timeouts"** de comunicação podem ser configurados aqui. Os outros parâmetros de comunicação (taxa de transmissão de dados, paridade...) devem ser configurados no menu "**Depurar**" da janela de Gerenciamento de Programa.
O "Time out de Comunicação" é o tempo deixado para o sistema destino começar a resposta a um pedido do ambiente de trabalho. A "**Duração da atualização**" é o período de tempo requerido para as solicitações de "leitura" a serem enviadas pelo depurador para atualizar dados nas janelas abertas.
Todos os valores de tempo são exibidos e inseridos como números inteiros em **milisegundos** (ms). Os parâmetros de tempo de comunicação não podem ser configurados quando o depurador é utilizado no modo de simulação.

Opções de visualização

A opção "**Exibir tempo de ciclo**" permite que o usuário esconda ou mostre os valores do tempo de ciclo na barra de controle de depurador. Quando esta opção está ativa, todos os componentes do tempo de ciclo (permitido, corrente, máximo, "overflow") são exibidos e atualizados. Desativando esta opção a carga do depurador de comunicação é reduzida.
Quando a opção "**Exibir erros**" está ativa, os erros de tempo de execução detectados são listados na área inferior da janela de depurador. Quando esta opção está desativada, a lista de erros é fechada. Ao remover esta opção, a carga do depurador de visualização e de comunicação é reduzida. Os comandos "**Opções / Apagar erros**" limpa a lista de erros de tempo de execução atualmente exibida na janela de depurador.
O comando "**Opções / Minimizar janela**" reduz o tamanho da janela de depurador de forma que é mostrada como uma pequena janela, sempre no topo, o painel contendo somente o estado do aplicativo e botões gráficos para os comandos mais comumente utilizados.

A.15.4 Comandos "write" (escreve)

O depurador ISaGRAF simbólico oferece muitos comandos para modificar o **valor** ou o **estado** dos componentes de aplicativo. A seleção do componente a ser modificado é feita clicando duas vezes sobre seu nome ou seu desenho em uma janela de edição, quando a janela de depurador está aberta.

Variáveis

Um estado de variável é modificado clicando duas vezes sobre seu nome em um das seguintes janelas :

- Dicionário
- Listas de variáveis ou diagramas de tempo
- Programas LD ou FBD
- Conexão E/S

Os seguintes comandos são oferecidos na caixa de diálogo de depuração :

- Escreva a variável para um novo valor
- **Lock** (bloqueia) a variável (somente para variáveis E/S)
- **Unlock (Desbloqueie)** a variável (somente para variáveis E/S bloqueadas)
- **Start (Iniciar)** ou **Stop (Parar)** uma variável de temporização (ativar o modo automático de atualização)

Valores simbólicos utilizados para representar valores booleanos **FALSE** e **TRUE** são as "strings" definidas para aquela variável booleana específica no dicionário. O valor analógico especificado para um comando "**Write**" deve ser inserido em um formato inteiro ou real, de acordo com a definição da variável no dicionário. A "string" especificada para um comando "**Write**" para uma mensagem não pode ser maior que a capacidade da mensagem atrelada àquela variável específica no dicionário.

Objetos SFC

Para observar uma operação de controle em um **programa SFC** durante a depuração do aplicativo, os comandos do menu "**Arquivo**" são utilizados na janela de Gerenciamento de Programa. O programa SFC deve ser selecionado na lista de programas. Os seguintes comandos estão disponíveis:

Iniciar programa SFC:..... Habilita o programa selecionado colocando uma marca SFC em cada etapa inicial.
Encerrar programa SFC:. Mata o programa selecionado removendo todas as suas marcas.
Congelar programa SFC:. Remove todas as marcas existentes do programa selecionado e armazena suas localizações.
Reiniciar programa SFC:. Reinicia um programa congelado recolocando as marcas que foram removidas pelo comando "**Freeze**" (Congelar).

Para programas secundários, estes comandos correspondem às funções "**GSTART**", "**GKILL**", "**GFREEZE**" e "**GRST**" na linguagem de programação.

Uma operação de controle pode ser vista em uma **etapa SFC** durante a depuração do aplicativo clicando duas vezes sobre sua representação gráfica na janela de edição SFC. Os seguintes comandos estão disponíveis dentro da caixa de diálogo de depuração:
- Instala um breakpoint na etapa **de ativação**
- Instala um breakpoint na etapa **de desativação**
- **Remove** (**Limpar**) o breakpoint adicionado à etapa

Nota: Breakpoints de ativação e de desativação não podem ser acrescentados à mesma etapa.

Uma operação de controle pode ser vista em uma **transição SFC** durante a depuração do aplicativo clicando duas vezes sobre sua representação gráfica na janela de edição SFC. Os seguintes comandos estão disponíveis dentro da caixa de diálogo de depuração:
- Adiciona um **breakpoint** na remoção de transição
- **Remove** (**Limpar**) um breakpoint acrescentado à transição
- Manualmente **remove** (**Limpar**) a transição (move ou adiciona marcas)

Remoção condicional: um símbolo é criado nas etapas seguintes à transição. As marcas que existem nas etapas precedentes são removidas. **Remoção incondicional**: uma marca é criada

nas etapas seguintes à transição. As marcas que existem nas etapas precedentes não são removidas.

A.15.5 Modificação online

A característica "modificação online" permite que o usuário modifique o aplicativo enquanto o processo estiver em andamento. Às vezes, isto é necessário para processos de substâncias químicas em que qualquer interrupção pode arriscar a produção ou a segurança. Esta função deve ser utilizada **muito cuidadosamente**. O ISaGRAF pode não ser capaz de detectar todos os conflitos possíveis gerados pelas operações definidas pelo usuário como resultado destas mudanças online.

- ***Seqüências de código***

Como ISaGRAF oferece muitas possibilidades para acesso a variáveis, programas ou placas E/S do depurador, a função "modificação online" descrita aqui só se aplica à modificação de seqüências de código. Uma seqüência de código é um conjunto completo de instruções ST, IL, LD ou FBD executadas em uma fila. Em um programa "início de ciclo" ou "fim de ciclo", uma seqüência de código é a lista inteira de instruções escritas no programa. Em um programa SFC, uma seqüência de código é a programação de Nível 2 de uma etapa ou transição. Uma "modificação online" consiste em substituir uma ou mais seqüências de código, sem parar a execução de ciclo PLC. Como o controle das marcas SFC é muito crítico, **não é possível modificar uma estrutura SFC, adicionar, renumerar ou remover uma etapa, uma transição ou um programa SFC**.

- ***Variáveis***

Como o banco de dados variável é uma parte muito crítica do aplicativo, ele pode ser acessado a qualquer hora por outros processos (em PLC multitarefa). Também é possível modificar valores de variáveis do depurador. Entretanto, **o ISaGRAF não permite que o usuário adicione, renomeie ou remova uma variável online**. De qualquer maneira, é possível modificar o modo que uma variável é utilizada no aplicativo. Também é possível reservar variáveis "não utilizadas" interna ou E/S na primeira versão do aplicativo, de forma que modificações futuras possam fazer uso delas.
São diferentes estilos de variáveis no banco de dados ISaGRAF destino. Limitações agem sobre todos elas:

- Variáveis declaradas

São aquelas utilizando o dicionário ISaGRAF. Não podem ser modificadas e não podem ser renomeadas para modificação online. É recomendado que algumas variáveis extras sejam declaradas e inicializadas no aplicativo mesmo que não sejam utilizadas no momento. Tais variáveis extras permitirão modificações futuras para trabalhar sem modificar o “checksum” de dados de aplicativo.

- Instâncias de blocos de função

Cada Instância de bloco de função escrito em "C" ou IEC corresponde a dados armazenados em banco de dados em tempo real ISaGRAF destino. Quando instâncias de bloco de funções são adicionadas ou removidas, a modificação online não é mais possível. Assim é melhor trabalhar em ST com instâncias FB declaradas em dicionário, em lugar de adicionar blocos

(isso corresponderá a novas instâncias automaticamente declaradas) em diagramas Quick LD ou FBD. Também, qualquer modificação na definição de blocos de função disponíveis na biblioteca ISaGRAF conduzirá a uma modificação online impossível.

- Etapas

Cada etapa SFC corresponde a um pedaço de dados nos quais são armazenados atributos dinâmicos SFC (seu tempo de atividade e bandeira). A adição ou a remoção de etapas SFC modifica o banco de dados de aplicativo e é proibido para modificação online.

- Variáveis escondidas alocadas por compiladores

O Compilador ISaGRAF gera variáveis temporárias "escondidas" para resolver expressões complexas. Em algum caso, a modificação de uma expressão pode conduzir a um conjunto diferente de variáveis temporárias não visíveis, e isso conduz a uma modificação online impossível. Para evitar esta situação, você pode adicionar as seguintes entradas no arquivo ISA.INI para forçar que um número mínimo de variáveis temporárias seja alocado para cada programa, mesmo se não utilizadas para a compilação da primeira versão do aplicativo. Os valores aqui apresentados são apenas exemplos:

```
[DEBUG]
MNTVboo=8       ; para booleanas
MNTVana=4       ; para inteiras e reais
MNTVtmr=4       ; para temporizadores
MNTVmsg=2       ; para mensagens
```

Quando tal conjunto é escrito no arquivo ISA.INI, o compilador emite uma mensagem de advertência se uma nova compilação de aplicativo conduz para um número maior de variáveis temporárias alocadas.

Entradas e saídas

Como o sistema E/S ISaGRAF é muito aberto, as modificações exigidas deveriam ser implementadas pelo OEM, utilizando características específicas do hardware correspondente. O sistema ISaGRAF **não permite que o usuário adicione, conecte ou remova uma variável E/S, ou modifique a descrição de uma placa E/S** online. Operações tais como modificar parâmetros de placa e bloquear canais E/S são disponíveis utilizando características de padrão OEM e a função "**OPERATE**".

Tempo de execução das operações

A modificação de um aplicativo em andamento consiste nas seguintes operações:

- modificar o código fonte de aplicativo no ambiente de trabalho
- gerar o novo código de aplicativo
- carregar o novo código de aplicativo utilizando o comando "**Atualizar**" em vez de "**Descarregar**"
- trocar do aplicativo antigo para o novo, entre os ciclos de execução PLC utilizando o comando "**Realizar atualização**".

Este procedimento garante que o PLC destino sempre tem um aplicativo em execução completo e seguro, e permite que o usuário controle a sincronização das operações de

amostragem de uma maneira muito segura e eficiente. Também permite que o usuário modifique o projeto tão freqüentemente quanto possível. Indiferentemente ao processo, a "modificação online" é essencialmente igual ao conjunto de comandos normais "**parar**, **Iniciar** e **Descarregar**". As únicas diferenças são que nenhum estado variável é perdido, e o tempo de comutação é muito curto (normalmente 1 ou 2 ciclos de duração). Durante a comutação, nenhuma variável é modificada, e **todas as variáveis interna, de entrada ou de saída mantêm o mesmo valor** anterior e posterior a modificação de aplicativo. Durante a comutação, nenhuma ação é executada, e **marcas SFC não são movidas**.

Memória necessária

Para suportar a capacidade "modificação online", o PLC destino tem que ter espaço de memória livre para habilitar o armazenamento da versão modificada do código de aplicativo. Ambas as versões do código de aplicativo têm que ser armazenadas em memória PLC durante a operação de comutação.

Limitações

Como descrito anteriormente, somente modificações nas seqüências de código são permitidas. A definição de variável, os parâmetros de aplicativo e conexões E/S não podem ser modificados. Ao carregar uma versão modificada do aplicativo, o ISaGRAF faz uma comparação entre o aplicativo modificado e o em execução para detectar qualquer modificação insegura. Se a comutação parece perigosa ou impossível, um erro de carga é gerado. Uma das proteções executadas pelo ISaGRAF é comparar o "checksum" da tabela de símbolos, de forma que qualquer modificação de nome de variável, programa ou elemento SFC é detectado. Se uma etapa está ativa quando a comutação ocorre, suas ações não armazenadas (N) são perdidas. As ações de ativação de novas etapas não são executadas. Ações executadas durante a desativação da etapa são aquelas trazidas no novo código de aplicativo. Se uma transição é válida quando a comutação ocorre, sua equação de receptividade é atualizada. O novo código de aplicativo carregado não é duplicado no PLC. A cópia de segurança ("backup") é a versão previamente carregada com os comandos de carga padrão.

Operações

Para atualizar o código de um aplicativo em execução, as seguintes operações têm que ser executadas:

- Antes de fazer qualquer modificação em um aplicativo em execução, é muito recomendado fazer uma cópia do projeto atual com outro nome. As modificações podem ser executadas nas cópias.
- Antes de editar qualquer programa, o usuário deve verificar que a opção "**Atualizar Diário**" das ferramentas de edição está ativa, para facilitar a futura manutenção de programa.
- Quando uma ou mais seqüências forem modificadas (sem modificar estruturas SFC e hierarquia de programa), o novo código de aplicativo deve ser gerado no ambiente de trabalho antes de carregar.
- Utilizando o depurador, de dentro o projeto antigo, o usuário tem que conectar o PLC destino e tem que executar qualquer operação que possa fazer o aplicativo atualizar mais rapidamente ou com mais segurança.
- Utilizando o depurador, de dentro do novo projeto, o usuário deve conectar o PLC destino. Se o nome de aplicativo é modificado, o banco de dados destino não pode ser acessado. O usuário tem que executar o comando "**Arquivo / Atualizar**".

• O aplicativo modificado é carregado selecionando a opção "**Atualizar depois**". Isto pode reduzir ligeiramente a velocidade durante a transferência PLC.
• Quando a carga está completa, o usuário pode executar o comando "**Arquivo / Realizar atualização**" para habilitar a comutação no momento mais adequado. A comutação terá uma duração 1 ou 2 ciclos.
• Quando a comutação tiver sido corretamente executada, os programas do aplicativo modificado em execução são mostrados. Se não, o aplicativo em execução existente permanece como é.

A.15.6 Trocas DDE

O depurador ISaGRAF inclui um servidor DDE ("Dynamic Data Exchange"). Um "loop" de aviso pode ser instalado entre o depurador ISaGRAF e outros aplicativos, de modo a exibir dinamicamente o valor atual das variáveis nos aplicativos não ISaGRAF.
Somente as transações "advise" e "poke" são suportadas pelo servidor DDE depurador ISaGRAF. Você só pode utilizar a transação "request" para variáveis já monitoradas dentro de um "loop" de aviso. Outros serviços DDE tais como "execute" não estão disponíveis. Quando um "loop" de aviso está estabelecido em uma variável, o valor desta variável é atualizado no aplicativo de cliente toda vez que ele mudar. Qualquer tipo de variáveis pode ser monitorado. A identificação da ligação dinâmica inclui os seguintes nomes :

Service name: ....."ISaGRAF"
Topic name:.....Nome do projeto ISaGRAF
Item name:.....Nome da variável

Se a variável for local para um programa, o seu nome deve ser seguido pelo nome de seu programa principal, escrito entre parênteses, com seguinte a sintaxe :

nome_da_variável(nome_do_programa)

O servidor DDE depurador ISaGRAF que é dedicado ao aplicativo ISaGRAF atualmente monitorado pelo depurador. Até **256** variáveis podem ser monitoradas pelo servidor ISaGRAF. O servidor DDE pode ser utilizado quando o depurador ISaGRAF é executado ou no modo conectado ou no modo simulação. A duração da atualização é a estabelecida para a comunicação entre o depurador e o sistema ou simulador ISaGRAF destino.

A.16 Listas de variáveis a serem mostradas durante a depuração

O comando "**Spy lists**" no menu "**Ferramentas**" da janela de Depurador permite que o usuário construa listas não contíguas de variáveis que são atualizadas com seus valores atuais. As listas são construídas durante a depuração do aplicativo. As listas podem ser armazenadas no disco e podem ser abertas novamente durante outras sessões de depuração. Uma lista pode conter até **32** variáveis. Podem ser misturadas variáveis de tipos diferentes na mesma lista. Podem ser inseridas variáveis globais e locais em uma lista. Uma lista de variáveis é dedicada a um projeto particular. Listas de variáveis são muito úteis na verificação funcional de um aplicativo. Eles permitem que o usuário assista às modificações de uma parte limitada do processo controlado, independente do código de fonte correspondente nos programas de aplicativo. Listas de variáveis também são úteis durante a depuração de programas texto ST e IL. O usuário pode agrupar facilmente em uma lista o conjunto de variáveis utilizadas em um programa, de modo a controlar ou monitorar a execução das instruções programadas.
Para cada variável da lista, o ISaGRAF exibe seu nome, seu valor atual e seu texto de comentário. As colunas podem ser redimensionadas arrastando as linhas de separação com o mouse na barra de título de lista.

Salvando as listas no disco rígido

Os comandos do menu "**Arquivo**" são utilizados para criar, abrir e Salvar como listas de variáveis. O número de listas para um projeto não é limitado pelo ISaGRAF. Durante a nomeação das listas de variáveis a serem salvas em disco, as regras mostradas a seguir têm que ser seguidas:

- o nome não pode exceder a **8** caracteres
- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes podem ser letras, algarismos ou o caractere sublinha "_"
- o nome das listas não faz distinção entre maiúsculas/minúsculas

O editor de lista não pode exibir mais que uma lista de variáveis de cada vez na mesma janela. Porém, o editor de lista pode ser executado mais que uma vez para monitorar diferentes listas simultaneamente.

Inserindo as variáveis na lista

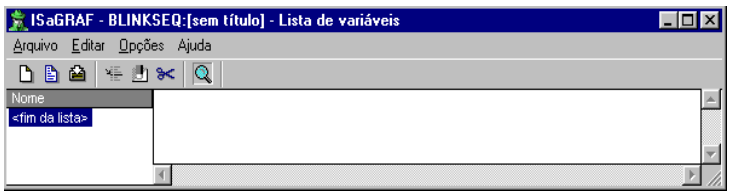
O comando "**Editar / Inserir**" insere outra variável na lista. O nome da variável é selecionado na lista de objetos definida no dicionário de projeto. Deste modo o usuário não tem que entrar manualmente no identificador. A variável é inserida antes da variável atualmente selecionada na lista. A lista não pode conter mais que **32** variáveis. A mesma variável não pode aparecer mais que uma vez na mesma lista.

Modificando a variável selecionada

O comando "**Editar / Modificar**" substitui a variável selecionada por outra variável. Você também pode utilizar o comando "**Recortar**" para remover a variável selecionada da lista.

Modo "Dump" de visualização

A qualquer hora, você pode trocar o modo de visualização entre lista e "Dump". Pressione o botão "zoom" na barra de ferramentas ou utilize o comando "**Opções / Dump**" para trocar o modo de visualização.
No modo "Dump", é exibido só um valor variável. Seu valor é exibido no formato numérico/simbólico no topo da janela, e também é exibido no formato "dump" binário. Este modo permite a monitoração do valor hexadecimal de cada byte no valor da variável.



A exibição "Dump" é muito útil para monitorar e compreender “strings” de mensagem contendo caracteres não imprimíveis.

A.17 Depurando programas ST e IL

Durante simulação ou depuração online de programa ST e IL, nenhuma modificação pode ser inserida no texto de programa.

IL Para programas IL, as instruções são formatadas em uma lista. O valor atual de uma variável utilizada em uma instrução é exibido na mesma linha. Você pode clicar duas vezes em uma instrução para mudar o valor da variável correspondente.

ST Para programas ST, a janela "Spy List" está incorporada à janela do editor. Você pode redimensioná-la arrastando com o mouse a linha de separação entre elas.

Para cada variável da lista, o ISaGRAF exibe seu nome, seu valor atual e seu texto de comentário. Colunas podem ser redimensionadas arrastando as linhas de separação com o mouse na barra de título de lista.

Salvando a lista no disco rígido

O comando "**Arquivo / Salvar** " salva as listas de variáveis no disco, sob o mesmo nome, como o programa editado. Esta lista será automaticamente recarregada toda vez que o programa ST ou IL estiver aberto no modo depurar. Esta lista também pode ser livremente aberta e modificada utilizando a ferramenta Lista de Monitoração ("Spy List") executada pelo comando "**Ferramentas / Spy list**" da janela depurador.

Inserindo variáveis na lista

O comando "**Editar / Inserir** " insere outra variável na lista. O nome da variável é selecionado na lista de objetos definida no dicionário de projeto. Deste modo o usuário não tem que entrar no identificador manualmente. A variável é inserida antes da variável atual selecionada na lista. A lista não pode conter mais de **32** variáveis. A mesma variável não pode aparecer mais de uma vez na mesma lista.

Quando o nome da variável está destacado no texto ST, pressione este botão na barra de ferramentas ou execute o comando "**Editar / Examinar seleção**" para enviar diretamente a variável para a lista de monitoração incorporada.

Modificando a variável selecionada

O comando "**Editar / Alterar variável**" substitui a variável selecionada por outra variável. Você também pode utilizar o comando "**Recortar variável**" para remover a variável selecionada da lista.

A.18 Depurando com “SpotLight”

A ferramenta “SpotLight” ISaGRAF permite que o usuário defina listas de observação que podem ser exibidas como gráfico ou como listas durante a depuração. Itens gráficos devem ser ligados às variáveis do projeto ISaGRAF. O gráfico é tanto definido quanto animado "online".
Para forçar o valor de uma variável, clique duas vezes sobre o item na disposição gráfica ou de lista correspondente, ou pressione ENTER após a seleção.
Você também pode fechar o documento (negar qualquer modificação) utilizando o comando "**Arquivo / Travar**". Quando um documento é bloqueado, você ainda pode forçar variáveis clicando duas vezes no seu símbolo.

A.18.1 Construindo a disposição gráfica

Um gráfico é feito de figuras de fundo (bitmaps ou metafiles), e um conjunto de itens gráficos que serão animados durante a depuração. Para inserir o gráfico, as seguintes operações devem ser executadas: insira as figuras de fundo, insira os itens gráficos, ligue os objetos com as variáveis do projeto.

Figuras de fundo

As figuras de fundo são arquivos "bitmap" (.BMP) ou “metafile” (.WMF). O número de figuras incluídas na disposição gráfica não está limitado. As figuras podem ser movidas ou redimensionadas na disposição gráfica. Eles não aparecem na disposição de lista. As figuras são construídas com outras ferramentas. O “SpotLight” não inclui uma ferramenta de pintura. O comando "**Opções / Cor de fundo**" é utilizado para selecionar uma cor sólida para o espaço vazio na disposição gráfica.

Nota: Bitmaps consomem uma grande quantidade de memória. É muito recomendável que se dimensione corretamente o tamanho da figura, e limite o espaço não utilizado dentro do retângulo de bitmap.

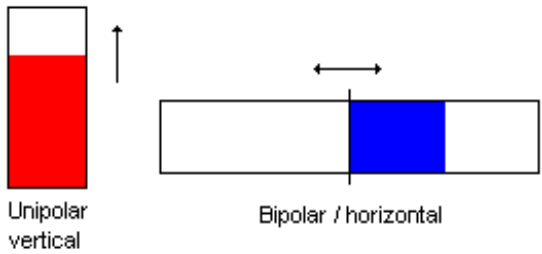
Textos

Um item “único texto” é um texto escrito em um retângulo. O texto exibido é o valor da variável atrelada. Assim, tal item pode ser ligado à variável “string” de mensagem.
O retângulo no qual o texto é exibido pode ser preenchido com uma cor ou ficar transparente. A fonte de caractere utilizada para exibir texto é ajustada para amoldar-se com a altura do retângulo quando o item for redimensionado.

Gráficos de barras unipolar e bipolar

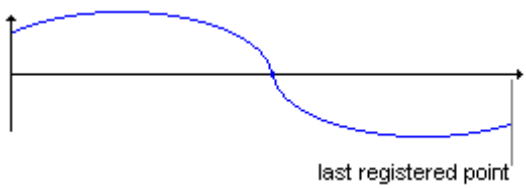
Um gráfico de barra é um retângulo com uma parte colorida que representa o valor numérico da variável atrelada. Opcionalmente, o resto do retângulo pode ser preenchido com uma cor.
Um gráfico de barra pode ser horizontal ou vertical.
Gráficos de barras unipolar podem crescer em qualquer direção: para cima, para baixo, para a esquerda ou para a direita.

Gráficos de barras bipolares na direção positiva ou na negativa, de acordo com o valor de variável atrelada. No caso de um gráfico de barra bipolar, o valor máximo permitido é o mesmo para ambas as escalas, positiva e negativa.



Curvas

É possível inserir uma curva em um documento. Uma curva mostra a história da variável atrelada. Embora não seja uma ferramenta precisa de medida, pode dar informação útil de depuração sobre o sincronismo entre diversas variáveis.
Uma curva armazena os últimos 200 valores de uma variável. O número de amostras não é modificado quando o item de curva é redimensionado na disposição gráfica.

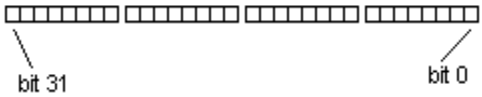


Ícones booleanos

Um item "ícone booleano" é utilizado para exibir um estado binário. Um arquivo ícone (.ICO) é definido para o valor FALSE ou 0. Outro ícone é para todos os valores diferentes de zero. Como o "SpotLight" não inclui um editor de ícone, os arquivos de ícone devem ser preparados com outra ferramenta.

Campos de bit

Um item "campo de bit" mostra em um painel gráfico os 32 bits de um valor inteiro. O bit menos significativo sempre é exibido à direita. Não é recomendado utilizar o campo de bit para outros tipos de dados como valores analógicos reais, já que a informação exibida pode conduzir a confusões.



Selecionar, mover ou redimensionar itens

A seleção de objetos gráficos é necessária para a maioria dos comandos de edição. O "SpotLight" habilita a seleção de um ou mais objetos existentes na área de gráfico. Para selecionar objetos, o botão "**selecione**" (botão com uma seta) deve estar ativo na barra de ferramentas de editor. Para selecionar um objeto, o usuário tem que simplesmente clicar sobre seu símbolo. Para selecionar uma lista de objetos, arraste o mouse na área de desenho para selecionar uma área retangular. Todos os objetos gráficos que estiverem dentro do retângulo de seleção são marcados como "**selecionado**". Um objeto selecionado tem desenhado pequenos quadrados pretos ao redor de seu símbolo gráfico.
Ao fazer uma nova seleção, qualquer objeto previamente selecionado passa a não selecionado. Para remover a(s) selecão(ões) existente(s), simplesmente clique duas vezes com o mouse sobre uma área vazia fora do retângulo que limita os objetos selecionados.
Para mover objetos, você tem que selecioná-los primeiro. Então coloque o cursor do mouse na borda do item selecionado e arraste para outro local.
Para redimensionar um objeto, você tem que selecioná-lo primeiro. Então coloque o cursor do mouse sobre um dos pequenos retângulos exibidos na borda de seleção, e arraste na direção apropriada para redimensionar o objeto. Figuras também podem ser redimensionadas. Neste caso, o bitmap ou metafile correspondente é esticado para amoldar-se ao novo retângulo de item especificado.

Agrupar itens / dissociar grupos

Você pode agrupar itens de forma que eles possam ser gerenciados como um item. Para compor um grupo, selecione os itens na disposição gráfica e execute o comando "**Editar / Agrupar**". O comando "**Editar / Dissociar**" é utilizado para restabelecer itens do grupo selecionado como separados.
Um grupo pode conter uma figura. Um grupo também pode conter outro grupo.
Quando itens são agrupados, os seus estilos não podem mais ser modificados. Itens do grupo são ainda exibidos, mas não podem ser utilizado (com duplo clique) para modificar o valor de variáveis fixas.
Um grupo só aparece em uma linha na disposição de lista.

A.18.2 Apresentação da lista

A qualquer momento você pode trocar da disposição gráfica para lista, e vice-versa, pressionando este botão. Você também pode utilizar o comando "**Opções / Visualização por lista/Gráfica**".
Na disposição de lista, são mostrados itens em uma caixa de lista clássica. A altura de cada item é calculada de acordo com seu estilo de desenho. Figuras (bitmaps e metafile) não são visíveis a partir da disposição de lista. Uma seleção está disponível na disposição de lista e deveria ser utilizado para configurar o estilo de item ou modificar o valor de uma variável. A seleção múltipla e os comandos que a utilizam não estão disponíveis neste modo.

Você pode reordenar os itens na lista que utiliza os comandos "**Editar / Mover para cima na lista**" ou "**Editar / Mover para baixo na lista**". O item a ser movido deve ser selecionado na lista.

A.18.3 Definindo a formatação do item

O estilo gráfico e as configurações de um item existente podem ser modificados, clicando duas vezes sobre o seu símbolo na área gráfica ou executando o comando "**Editar / Determinar estilo do item**" quando o item é selecionado em disposição gráfica ou lista. A caixa de diálogo "Estilo do Item" também é aberta quando um novo item é acrescentado ao documento. Agrupa os seguintes pedaços de informação a ser selecionado pelo usuário:

- ***Estilos e configurações do gráfico:***

  O estilo de exibição (texto único, gráfico de barra, curva…) de um item pode ser modificado dinamicamente. Quando as cores de primeiro plano e de fundo são utilizadas, elas podem ser personalizadas utilizando as caixas correspondentes. Quando estilo é "ícone booleano", o nome do caminho de arquivos .ICO correspondentes tem que ser especificado. Utilize os botões "**…**" próximos a estes controles para varrer os arquivos ícone existentes no disco.

- ***Escala:***

  Este é o valor máximo que pode ser exibido em gráficos de barra e curvas. Para gráficos de barra bipolares e curvas, o mesmo valor absoluto é utilizado para o eixo positivo e negativo.

- ***Nome da variável:***

  Quando o campo "**Nome**" é o campo ativo, pressionando o botão "**…**", botão próximo ao controle de edição, o usuário pode localizar os nomes das variáveis declarados no dicionário de projeto.

- ***Legenda:***

  Uma legenda pode ser exibida perto de um item gráfico em disposição gráfica. Você pode personalizar o local do texto da legenda (topo, em baixo, esquerda ou direita) e seus conteúdos. Legenda pode ser qualquer combinação do nome variável e seu valor formatada como texto. A personalização de legenda não tem nenhum efeito na disposição de lista.

- ***Variável de comando:***

  Se a opção "Variável de comando" está configurada, o usuário pode modificar o valor da variável atrelada durante a depuração clicando duas vezes sobre o item símbolo gráfico.

A.18.4 Comandos do menu "Arquivo"

O menu "**Arquivo**" contém os comandos que permitem o usuário gerenciar o documento completo.

O comando "**Novo**" do menu "**Arquivo**" começa a edição de um novo documento. O número de documentos definido para um projeto não está limitado pelo ISaGRAF. Antes de editar o novo gráfico, o anteriormente aberto é fechado. O "SpotLight" não pode ser

utilizado para editar vários gráficos ao mesmo tampo. Porém, podem ser abertas várias janelas “SpotLight” simultaneamente com cada uma editando um documento diferente.

O comando "**Abrir**" do menu "**Arquivo**" permite que o usuário feche o documento atualmente editado e comece a editar outro documento do projeto atual. O novo documento selecionado substitui o atual na janela de edição. Ao selecionar o novo documento, o botão "**Apagar**" pode ser utilizado para apagar um arquivo existente para limpar o diretório de projeto. Os arquivos ícone e bitmap referenciados em um gráfico não são apagados quando este o é.

O comando "**Salvar**" do menu "**Arquivo**" armazena o documento atualmente editado no disco. Se é um novo documento sem título, o usuário tem que dar um nome antes de salvá-lo. O nome de um documento tem que estar conforme as seguintes regras :

- O comprimento do nome não pode exceder a **8** caracteres
- O primeiro caractere deve ser uma **letra**
- Os seguintes podem ser **letras**, **algarismos** ou o caractere de **sublinha** “_”
- O nome não faz distinção entre maiúscula/minúscula

O comando "**Salvar como**" do menu "**Arquivo**" permite que o usuário armazene o documento atualmente editado com outro nome.

A.18.5 Nota para usuários do ISaGRAF V3.2

O “SpotLight” pode ler gráficos e listas de diagramas de tempo construídos com as ferramentas ISaGRAF V3.0 ou V3.2. Tais arquivos aparecem na caixa de diálogo "**Abrir**", com a descrição de suas origens. Os arquivos podem ser lidos e livremente modificados com o “SpotLight”.

Ao abrir um gráfico ISaGRAF V3.2, o documento é automaticamente marcado como "Travado". Remova a opção "**Travar**" do menu "Arquivo" se você quiser fazer mudanças no gráfico.

Quando um gráfico ISaGRAF 3.2 ou lista de diagrama de tempo está aberta, o “SpotLight” sempre propõe salvá-lo em formato “SpotLight” original. A caixa de diálogo "**Salvar como**" está sistematicamente aberta ao fechar tal documento.

A.19 Aplicativos de carga de projetos

O ISaGRAF suporta a carga de aplicativo armazenado no destino. O procedimento de carga comunica com o destino para carregar o código fonte compactado embutido (EZS) e então restabelece o projeto carregado no ambiente de trabalho.
O projeto em execução no sistema destino conectado pode ser carregado se a versão destino for V3.22 ou posterior, e se o código fonte foi embutido com a aplicativo. O código fonte embutido para a carga é uma característica opcional.

A.19.1 Carregando um projeto

A caixa de diálogo "**Carregar**" é executada a partir do comando "**Arquivos**" do Gerenciador de Projeto ISaGRAF. A carga não se refere a um projeto existente no Ambiente de Trabalho. O projeto atualmente selecionado em lista de gerenciamento de projeto não tem nenhuma relação com mecanismo de carga. Para carregar o aplicativo em execução no destino você deve:
1- assegurar que o destino está corretamente conectado
2- configurar os parâmetros de comunicação de acordo com o link de conexão
3- pressionar o botão " **EXECUTAR**"

O carregamento de uma fonte compactada embutida (EZS) e a descompactação pode levar alguns segundos. As mensagens na caixa de diálogo o informarão quando a carga está completa ou em estado de erro.
O nome utilizado para criar o projeto ISaGRAF é aquele lido no destino através da comunicação. Se este nome já for utilizado para um projeto existente no ambiente de trabalho, será perguntado se quer sobrescrevê-lo ou selecionar um nome novo. Você não pode cancelar a inscrição de fontes carregadas como um projeto quando a carga estiver completa. O projeto carregado está agora pronto e pode ser aberto.

Erros possíveis

Os seguintes erros podem ocorrer durante a carga de um projeto. Você é informado do erro na caixa de diálogo "Carregar".
- a comunicação não pode ser estabelecida com o destino
- destino conectado é um sistema ISaGRAF versão anterior a 3.22
- não há nenhum aplicativo sendo executado no destino
- não há nenhum EZS embutido no destino

A.19.2 Configuração da comunicação

Pressionando o botão "**Configurar**" o usuário pode definir os parâmetros do link utilizado para a comunicação para a carga entre o ambiente de trabalho ISaGRAF e o sistema ISaGRAF destino. Você tem que assegurar-se que os parâmetros configurados combinam com os do destino conectado antes de executar a carga.

A.19.3 Preparando um projeto para carga

Você tem que informar ao Gerador de Código ISaGRAF, que fez a compactação, que o código fonte deve ser incorporado ao código de aplicativo se você quiser habilitar a carga posterior. Para isto, pressione o botão "**Carregar**" na caixa de diálogo "**Opções do compilador**". Uma outra caixa de diálogo permite que você confira, como uma opção, a incorporação de código de fonte compactado. Neste caso, só o mínimo de arquivos fonte requerido será incorporado. Utilize outras caixas de ativação para também embutir arquivos opcionais.

Nota importante: As bibliotecas não são carregadas com código fonte incorporado. Isto inclui funções e blocos de função e placas E/S e equipamentos.

- ***Arquivos opcionais***

Além do código de fonte mínimo exigido, os seguintes arquivos podem também ser embutidos. Eles são opções já que sua seleção conduz a exigência de memória extra no destino.

Descrição do projeto: Se não incorporado, a descrição de projeto após a carga só indicará a data de carga.

Proteção por senha: Função de carga não é protegida por uma senha. Se você quiser que o projeto carregado seja protegido, você tem que incorporar sua senha de proteção de sistema ao código fonte.

Comentários para canais E/S não conectados: O ISaGRAF lhe dá a possibilidade de inserir texto de descrição para os canais E/S não conectados. Não ativando esta opção você só trabalhará com canais E/S conectados.

Histórico de modificações: Este é o histórico global de modificações para o projeto.

Arquivos de diário: Arquivo de diário de cada programa contém observações escritas pelo usuário mais o histórico de mensagens de saída de compilador referentes ao programa. A incorporação de arquivos de diário pode consumir muita memória no destino.

Listas de variáveis e diagramas de tempo: Estes são os arquivos criados durante a depuração contendo as listas de nomes de variáveis para a monitoração de lista ou diagrama de tempo.

Gráficos, ícones e bitmaps: Inclui gráficos ISaGRAF, mais todos os arquivos ícone e bitmap atrelados, se eles estão localizados no diretório de projeto. Atenção: a incorporação de arquivos de diário pode consumir muita memória no destino.

A.19.4 Como uma fonte compactada é armazenada em um destino

A fonte compactada incorporada (EZS) é armazenada em código gerado com recursos. O recurso gerado é chamado "**EZS**". Se a incorporação do código fonte é selecionada, você não pode escolher este nome para outro recurso. A incorporação de código fonte não implica

em qualquer limitação na definição de recurso. O arquivo de definição de recurso escrito pelo usuário não é afetado pela incorporação de fonte.
Refira-se à documentação ISaGRAF sobre o Gerador de Código para detalhes adicionais e informações sobre recursos.

A.19.5 Necessidades de memória no destino

O código fonte compactado incorporado (EZS) exige memória extra para ser armazenado com o código de aplicativo no destino. Uma estimativa geral é que o mínimo de EZS (nenhuma opção extra selecionada para incorporação de fonte) tem uma vez e meia o tamanho do código executável. Isto significa que a incorporação de EZS multiplicará o tamanho de código carregado por 2,.5.
Limitação especial pode aparecer em algum sistema destino baseado em memória segmentada. Como os EZS são armazenados como recursos em código gerado, eles devem ser armazenados no mesmo segmento de dados como código de aplicativo.

A.19.6 Sobre o projeto carregado

O projeto carregado contém todos os arquivos e dados requeridos para recompilar. Dependendo das opções selecionadas durante sua compilação anterior, também pode conter arquivos auxiliares como descrição de projeto e programa arquivo de diários.
Você tem que compilar (fazer) o projeto antes de depurá-lo ou monitorá-lo. Aviso: como o ISaGRAF utiliza a compilação de dados para comparar, você será informado ao abrir o depurador que o ambiente de trabalho e os aplicativos de destino têm códigos de versão diferentes.

Nota importante: As bibliotecas não são carregadas com código de fonte incorporado. Você tem que assegurar-se que as funções de biblioteca apropriadas e os blocos de função estão instalados com seu ambiente de trabalho ISaGRAF antes de recompilar o aplicativo de carga.

A.19.7 Compatibilidade

A carga é suportada pelo ISaGRAF destino e o ambiente de trabalho versão 3.22 ou posterior. Foram feitas extensões ao protocolo de comunicação para suportar carga.
Não há nenhuma restrição na incorporação de código fonte compactado (EZS) em um destino baseado em sistemas ISaGRAF da versão 3.03 a 3.21, já que EZS é armazenado no código de aplicativo como recursos padrões. Mas as informações incorporadas não podem ser carregadas neste caso já que o destino não suporta os serviços de comunicação requeridos.

A.20 Utilização da “Ferramenta de Diagnóstico”

A ferramenta "**Ferramenta de Diagnóstico**" é um subconjunto da ferramenta depurador ISaGRAF. Permite ao usuário final trabalhar sobre um conjunto de variáveis predefinidas, de modo a examinar e controlar o processo. O depurador ISaGRAF é uma ferramenta poderosa que inclui funções de alto nível. A Ferramenta de Diagnóstico é um caminho seguro para o controle do aplicativo alvo para as execuções finais de operações ou manutenção. A Ferramenta de Diagnóstico ISaGRAF é executada diretamente a partir do grupo ISaGRAF no Gerenciamento de Programas clicando duas vezes sobre o seguinte ícone:

A lista dos projetos existentes é mostrado em uma caixa de diálogo. Permite o usuário executar o depurador limitado ISaGRAF sobre um aplicativo ISaGRAF existente e já carregado. Pressionando o botão "**OK**" o depurador limitado é iniciado sobre o projeto selecionado. Pressionando o botão "**Cancelar**" a caixa de diálogo é fechada. O comando "**Configurar**" é utilizado para configurar o enlace de comunicação entre o Ambiente de Trabalho ISaGRAF e o PLC alvo. Veja o capítulo "**Gerenciamento de Programas**" deste manual para maiores informações sobre este comando.

Nota: A Ferramenta de Diagnóstico ISaGRAF (depurador limitado) não pode ser utilizado para carregar, parar ou atualizar a execução do programa no PLC alvo. Nenhuma operação pode ser executada se o projeto selecionado na caixa de diálogo da Ferramenta de Diagnóstico não for o mesmo instalado e executado no PLC.

Quando o depurador limitado ISaGRAF for executado, e corretamente conectado ao aplicativo alvo, os seguintes comandos estarão disponíveis:

- Lista de monitoração de variáveis
- Documentos gráficos de monitoração com “SpotLight”

A.21 Utilizando o simulador ISaGRAF

O simulador ISaGRAF Núcleo é iniciado com o depurador quando o comando "**Simular**" do menu "**Depurar**" da janela de Gerenciamento de Programa é executado. O simulador de núcleo é um sistema completo ISaGRAF destino que suporta características ISaGRAF padrão e todas as funções "C" e os blocos de função da biblioteca padrão fornecida pela CJ International. As placas E/S são simuladas graficamente em uma janela. Qualquer tipo de placa E/S pode ser simulado. As placas também definidas como "Placas Virtuais" durante a conexão E/S também são mostradas na janela de simulação.

A.21.1 Links com o depurador

O simulador de núcleo suporta comunicação total com o depurador ISaGRAF, de forma que quaisquer possibilidades de depuração podem ser utilizadas durante a simulação. O simulador de núcleo sempre trabalha no aplicativo ISaGRAF atual. Durante a simulação, os comandos de depurador "**Iniciar**", "**Parar**", "**Descarregar**" ou "**Atualizar**" não estão mais disponíveis. O simulador não pode ser utilizado se a etiqueta "SIMULATE" destino não tiver sido selecionada nas opções de compilador antes de construir o código destino. O fechamento da janela de simulador implica que a janela de depurador (e qualquer janela ISaGRAF aberta durante a sessão de depuração) também está fechada.

A.21.2 Simulação de E/S

As placas E/S aparecem na janela de simulador, designadas pelos seus nomes e o número do slot. Quaisquer dos tipos ISaGRAF padrão de E/Ss (booleana, analógica ou mensagem) são controladas. Os canais das placas de entrada são exibidos com botões e campos especiais. Os canais das placas de saída são exibidos com luzes gráficas de estado e campos de dados.

Entradas booleanas: Uma entrada booleana é representada por um botão cinza quadrado. O número do canal é exibido com o botão E/S. O valor de entrada é TRUE quando o botão é pressionado. Clicando sobre o botão o valor E/S correspondente é modificado. Utilize o botão direito do mouse para configurar a entrada quando o botão é pressionado.

Saídas booleanas: Uma saída booleana é representada por um pequeno círculo. O número do canal é exibido com o E/S. O valor de saída é TRUE quando o símbolo gráfico está destacado.

Entradas analógicas: Um canal de entrada analógica é um campo numérico simples cujo valor da entrada correspondente pode ser inserido. Clicando sobre a caixa o caractere circunflexo (^) é exibido. Um novo valor para o canal pode ser inserido então. Não é necessário usar a tecla ENTER depois da inserção. Entradas analógicas podem ser inseridas ou na base decimal ou na hexadecimal. Utilize os botões Para cima / Para baixo para aumentar ou diminuir o valor atual.

Saídas analógicas: Um canal de saída analógico é um campo de saída numérico. O valor de saída pode ser exibido como um número decimal ou hexadecimal. Nenhuma ação pode ser executada pelo usuário em um canal de saída.

Mensagens de entrada: Um canal de mensagem de entrada é um campo de texto simples no qual o valor da entrada correspondente é inserido. Clicando sobre a caixa o caractere circunflexo (^) é exibido. Um novo valor para o canal pode ser inserido então. Não é necessário usar a tecla ENTER depois da inserção.

Mensagens de saída: Um canal de mensagem de saída é um campo de saída de texto. Nenhuma ação pode ser executada pelo usuário em um canal de saída.

A.21.3 Componentes da biblioteca

O simulador ISaGRAF suporta completamente as conversões padrões, as funções e os blocos de função, fornecidos pela CJ International. A seguir, a lista de objetos suportados:

- ***Funções de conversão:***

  bcd, scale

- ***Funções:***

  abs, acos, ArCreate, ArRead, ArWrite, ascii, asin, atan, char, cos, delete, expt, find, insert, left, limit, log, max, mid, min, mlen, mod, mux4, mux8, odd, rand, replace, right, rol, ror, sel, shl, shr, sin, sqrt, tan, trunc

- ***Blocos de funções:***

  average, blink, cmp, ctd, ctu, ctud, derivate, f_trig, hyster, integral, lim_alrm, r_trig, rs, sema, sr, stackint, tof, ton, tp

As conversões definidas pelo usuário, funções “C " e blocos de função não estão comumente integradas com o Simulador ISaGRAF. Tipicamente, tais objetos são designados para utilizar os recursos de software e hardware do sistema destino. Tais recursos geralmente não estão disponíveis no sistema Windows. O Simulador ISaGRAF provê o seguinte comportamento padrão para qualquer conversão definida pelo usuário, função ou bloco de função:

- Quando uma nova conversão é processada pelo simulador, é substituída por uma conversão "nula ". Isto significa que o valor físico das variáveis analógicas sempre é igual ao valor elétrico (como inserido ou exibido no painel de Simulador).
- Quando uma nova função “C " ou bloco de função é executado pelo simulador, não processa nenhuma operação. O valor de resultado não é fixado.

A.21.4 Opções

Os comandos do menu "**Opções**" permitem que o usuário controle a exibição de E/S no painel de simulador. O usuário pode fixar ou pode remover estas opções a qualquer momento durante a depuração.

- Quando a opção "**Exibição em cores**" está ativa, os canais E/S são exibidos como bitmaps de cor. Se as cores não podem ser distinguidas em algumas telas LCD, o usuário deve remover esta opção, para obter gráficos de entrada e saída em preto e branco puro para canais E/S.
- Quando a opção "**Nomes de variáveis**" está ativa, uma etiqueta é exibida ao lado de qualquer canal E/S, com o nome da variável E/S conectada. A remoção desta opção permite que o usuário reduza o tamanho do painel de simulador.
- Quando a opção "**Valores em hexadecimal**" está ativa, qualquer canal analógico de entrada ou saída é exibido ou inserido no formato hexadecimal.
- Quando a opção "**Sempre visível**" está ativa, a janela de simulador sempre é visível, até mesmo se o foco de entrada está em outra janela.

A.21.5 Salvando e restaurando os estados de entrada

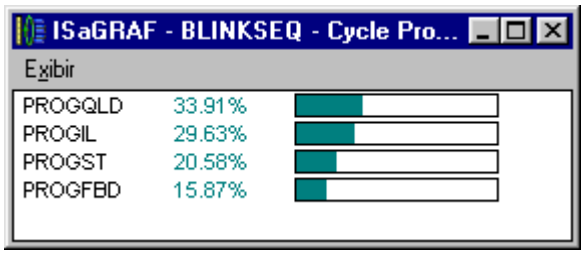
Utilizando o simulador ISaGRAF, os canais de entrada são forçados através de operações manuais, atuando sobre botões de dois estados e controles de edição do painel de simulação. Você pode a qualquer hora utilizar os seguintes comandos do menu "**Ferramentas**" para salvar e restaurar o estado de todos os canais de entrada:

Carregar esquema de entrada	Fixa valores de canais de entrada com valores armazenados em um arquivo criado em disco pelo comando "**Salvar esquema de entrada**" .
Salvar esquema de entrada	Salva o estado dos canais de entrada em um arquivo de forma que eles possam ser restaurados mais tarde utilizando o comando "**Carregar esquema de entrada**". O arquivo é armazenado no diretório de projeto e assim é salvo com outros arquivos de projeto pelo utilitário de arquivo ISaGRAF.

Nota:. Somente canais de entrada nomeados (aqueles tendo uma variável conectada) são salvos em disco.

A.21.6 O analisador de ciclo

O Analisador de Ciclo ISaGRAF é uma ferramenta de diagnóstico poderosa que mostra como o tempo de ciclo é distribuído entre os vários programas, funções e blocos de função de um aplicativo. Esta ferramenta é muito útil para se ter um diagnóstico rápido nos desempenhos do aplicativo, e conduz o programador às partes do código que possam precisar de otimizações.
O Analisador de Ciclo é executado pelo comando " **Ferramentas / Análise do tempo de ciclo** " nos menus da janela de Simulador ISaGRAF. Exibe, para cada programa, função ou bloco de função, a porcentagem do tempo de ciclo gasto para executá-lo:



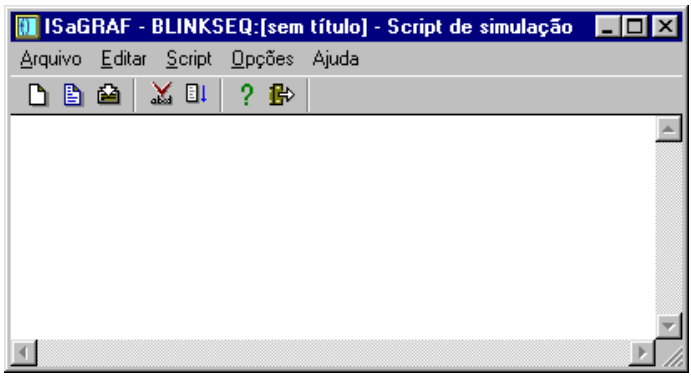
Quando a opção "**Exibir / Média**" está ativa, a informação exibida é uma média das porcentagens calculada desde que o aplicativo foi iniciado, ou desde a última vez que o comando "**Exibir / Inicializar**" foi executado.
Se a opção "**Exibir / Média**" não está ativa, a informação exibida mostra as medidas feitas durante a execução do último ciclo. Você também pode utilizar esta característica quando o aplicativo estiver no modo "**Ciclo a Ciclo**" para ter um conjunto de medidas dependendo do contexto do aplicativo.
Utilize o comando "**Exibir / Copiar**" para copiar nomes de programa e porcentagens para a área de transferência do Windows no formato ASCII. Então, os dados podem ser colados em documentos de texto ou planilhas eletrônicas comuns.

Notas importantes:

Estas não são medidas precisas. O cálculo de porcentagem está baseado na contagem de instruções TIC, levando em conta os vários tempos de execução de instrução. O cálculo não inclui o tempo gasto em funções "C" e blocos de função.
O valor exibido para uma função ou um bloco de função é a soma de todos os "tempos de chamada" dos programas de aplicativo no mesmo ciclo.
O cálculo de tempo está baseado no código TIC e não fornece informação segura se o código de aplicativo atual é gerado em linguagem "C" e construído utilizando um compilador "C".

A.21.7 Scripts de simulação

O simulador ISaGRAF inclui uma ferramenta para a edição e a execução de scripts de simulação. Um script é descrito com a ajuda de uma linguagem simples do tipo ST e permite a automatização de testes e a validação dos aplicativos simulados.
O editor de script é executado pelo comando "**Ferramentas / Scripts de simulação**" da janela “Simulador”. A seguir, a janela de edição de script:



A janela superior é um editor de texto no qual são inseridas as instruções de script. É utilizado como os outros editores de texto ISaGRAF e inclui características de alto nível tais como seleção com o mouse de um símbolo de variável. Você pode utilizar os comandos do menu "**Opções**" para configurar a tabulação e selecionar uma fonte de caractere.
A janela inferior mostra todas as mensagens de saída quando o script é executado. A linha de separação entre as janelas pode ser movido livremente para redimensionar as janelas. A janela de saída pode ser escondida durante a edição de script, mas é automaticamente aberta toda vez que um script é executado.

Editando scripts

Utilize os comandos do menu "Arquivo" para gerenciar arquivos de script:
Novo.....cria um novo script sem nome
Abrir.....carrega um arquivo de script existente
Salvar.....salva o texto de script e o conteúdo da janela de saída no disco, no diretório do projeto
Salvar como.....salva o script com outro nome

São criados dois arquivos no diretório de projeto ISaGRAF para cada script:
<*nome_do_script*>.SCC ..... texto do script (instruções)
<*nome_do_script*>.SCO ..... conteúdo da janela de saída
sendo que <*nome_do_script*> é o nome do script. Ambos os arquivos são arquivos de texto padrão e podem ser abertos utilizando qualquer outro editor de texto.

Durante a edição de um script, você pode utilizar o comando "**Editar / Inserir símbolo**" para selecionar um nome de variável declarado a ser inserido na posição do sinal de circunflexo.

Executando scripts

O script deve ser conferido e compilado antes de ser executado. Se necessário, a verificação de sintaxe é executada automaticamente em um comando "**EXECUTAR**". Utilize os seguintes comandos do menu "**Script**":

VERIFICAR.....verifica a sintaxe e compila o script

EXECUTAR Script ....inicia a execução do script atualmente editado

No caso de um novo script sem nome, ele deve ser salvo (e um nome deve ser dado) antes de sua conferência. No caso de um script nomeado, o script é salvo automaticamente no disco antes de verificação de sintaxe.
Quando o script estiver em andamento, seu conteúdo não pode ser modificado. Uma mensagem é exibida quando o fim do script é alcançado. Você também pode abortar um script em andamento utilizando o seguinte comando do menu "**Script**":

Interrompe Script .....interrompe a execução do script

A execução do script é realizada entre ciclos destinos. No caso de um “loop” infinito programado pelo ciclo, o simulador ISaGRAF assegura que este “loop” é sempre quebrado de forma que ciclos ISaGRAF são ainda executados e outros aplicativos ISaGRAF não são bloqueados. O intérprete de script ISaGRAF decide interromper a execução do script se a mesma "etiqueta" é encontrada mais de uma vez no mesmo ciclo destino. A execução de script também pode ser interrompida normalmente pelas instruções "Cycle" (ciclo) ou "Wait" (aguarda).

Linguagem de descrição do script

.A linguagem de descrição de script é uma linguagem de texto muito simples semelhante ao ST, mas na qual cada instrução é inserida em uma linha de texto separada, e não precisa ser terminada por um ponto-e-vírgula. Utilize o seguinte botão da barra de ferramentas para conhecer a lista de instruções disponíveis e inserir uma palavra chave na posição do sinal circunflexo:

insere instrução (palavra chave e ajuda como comentários)

Há vários tipos de instruções. Primeiro é a assignment (valor forçado) de uma variável:
:= .....assignment

Outras instruções permitem a saída de mensagens pela janela de saída:
Print .....apresenta um texto ou um valor de variável
PrintTime .....apresenta a hora local atual

Outras instruções são utilizadas para sincronizar as instruções de script com o ciclo ISaGRAF:
Cycle .....deixa o simulador ISaGRAF executar um ciclo
Wait .....espera durante tempo determinado

Outras instruções são utilizadas para controlar o fluxo de instrução no script:
Labels.....podem ser colocados em qualquer lugar no script
Goto .....desvio incondicional para uma etiqueta (label)
If goto.....desvio condicional para uma etiqueta (label)
End.....termina o script

A linguagem de script não faz distinção entre maiúscula/minúscula. Comentários podem ser inseridos ao término de qualquer linha de texto. Comentários também podem ser escritos de acordo com as convenções ST (entre caracteres "**(***" e "***)**"), ou prefixado por um caractere " **;**".

":=" Assignment

Significado: Força o valor de uma variável ISaGRAF. Pode ser uma variável interna, um canal de entrada ou um canal de saída.

Sintaxe: <*nome_da_variável*> **:=** <expressão_constante>
<*nome_da_variável*> **=** <expressão_constante>

Argumentos: < *nome_da_variável* > é um símbolo válido de uma variável de aplicativo declarada, ou uma variável E/S diretamente representada que utilizando convenções de escrita "% ".

<*expressão_constante*> é uma expressão constante válida que combina com o tipo da variável especificada. Para booleanos, "0" e "1" podem ser utilizados em vez de "FALSE" e "TRUE". Para temporizadores, o prefixo "T#" ou "TEMPO#" pode ser omitido.

Observação: Variável de entrada forçada por um script não precisa ser bloqueada. O desenho do canal de entrada correspondente é atualizado quando a variável de entrada é forçada por um script.

Aviso: não force uma variável analógica de entrada ou de saída atrelada a uma conversão, já que a execução de script não suporta funções de conversão ou tabelas.

Exemplo:
```
MyBooVar := 1        (* o mesmo que TRUE *)
MyIntVar := 1234
MyRealVar := 1.2345
MyMsgVar := 'Hello'
MyTmrVar := t#12s
```

Print

Significado: Escreve uma “string” ou o valor de uma variável na janela de saída. O texto é escrito como uma nova linha no fim de um texto já escrito na janela de saída.

Sintaxe: **Print '**<*texto*>**'**
Print <*nome_da_variável*>

Argumentos: <*texto*> é qualquer “string” de texto expresso entre aspas simples

<*nome_da_variável*> é um símbolo válido de uma variável de aplicativo declarada, ou uma variável E/S diretamente representada que utilizando convenções de escrita "% ".

Observação: A saída de valores variáveis sempre é formatada de acordo com convenções de IEC.

Exemplo:
```
Print 'Hello'
Print MyBooVar
```

Saída: Hello
MyBoovar = TRUE

PrintTime

Significado: Escreve a hora atual na janela de saída. O texto é escrito como uma nova linha no fim de um texto já escrito na janela de saída.

Sintaxe: **PrintTime**

Observação: A hora atual é formatada de acordo com a configuração atual do Sistema Windows

Exemplo:
```
Print 'A hora agora é:'
PrintTime
```

Saída:
```
A hora agora é:
15:45:22
```

Cycle

Significado: Suspende a execução do script até que o próximo ciclo ISaGRAF seja executado.

Sintaxe: **Cycle**

Observação: São executadas instruções de script no início de um ciclo ISaGRAF. Se o simulador estiver no modo "Ciclo a Ciclo", a instrução "cycle" é imediatamente seguida por um ciclo. As instruções seguintes do script serão executadas no próximo comando "**Executar um ciclo**" do depurador.

Exemplo:
```
(* o programa ISaGRAF copia A para B *)
A := 0
Cycle
Print B
A := 1
Print B (* ciclo não executado / B não foi para 1 *)
Cycle
Print B
```

Saída:
```
B = 0
B = 0
B = 1
```

Wait

Significado: Suspende a execução do script durante um tempo determinado (atraso).

Sintaxe: **Wait** <*atraso*>

Argumentos: <*atraso*> atraso expresso de acordo com convenções IEC para expressões de constante de tempo. O prefixo "T#" ou "TEMPO#" pode ser omitido. O valor de atraso deve estar entre 10 milisegundos e 1 hora.

Observação: A precisão da instrução "Wait" não é grande já que depende do sistema Windows "host". Também, o atraso deve ser considerado com uma precisão de mais ou menos um ciclo ISaGRAF. Quando uma instrução "Wait" é alcançada, ciclos ISaGRAF são executados até que o atraso esteja transcorrido e antes de continuar a execução de script.

Exemplo:
```
PrintTime
Wait 2s
PrintTime
```

Saída:
```
15:45:27
15:45:29
```

Label

Significado: As etiquetas podem ser colocadas em qualquer lugar no script. Elas são utilizadas como um destino através de instruções "Goto" e permitem o controle de fluxo para instruções de script.

Sintaxe: <*nome_do_rótulo*>**:**

Argumentos: <*nome_do_rótulo*> nome único de acordo com as convenções de nome de variável ISaGRAF: limitado a 16 caracteres, começando com uma letra, seguido por letras, algarismos ou caractere sublinha. Quando definido, o nome da etiqueta deve ser seguido por um caractere "**:**".

Observação: Nenhuma instrução deve ser colocada na linha em que uma etiqueta está definida. O nome de etiqueta não deve ser igual a de uma variável de símbolo declarado ISaGRAF

Exemplo:
```
(* exemplo de "script" com "loop" infinito *)
loop:
PrintTime
Wait 1s
Goto loop
```

Goto

Significado: Desvio incondicional para uma etiqueta

Sintaxe: **Goto** <*nome_do_rótulo*>

Argumento: <*nome_do_rótulo*> é o nome de uma etiqueta definida no script.

Observação: São permitidos desvios para trás. No caso de um loop infinito, a execução do script está automaticamente interrompida em cada loop para preservar a execução de ciclos ISaGRAF.

Exemplo:
```
Print 'Before Jump'
Goto MyLabel
Print 'Within Jump' (*nunca executado *)
MyLabel:
Print 'After Jump'
```

Saída:
```
Before Jump
After Jump
```

If Goto

Significado: Desvio condicional para uma etiqueta. A condição é uma comparação entre ou duas variáveis ISaGRAF ou entre uma variável e uma expressão constante.

Sintaxe: **If** <*var1*> ***test*** <*var2*> **Goto** < *nome_do_rótulo* >
If <var1> ***test*** <*expr_cont*> **Goto** < *nome_do_rótulo* >

Os **testes** comparativos disponíveis são:

=	TRUE se ambos os membros têm o mesmo valor
<>	TRUE se os membros têm valores diferentes
<	TRUE se o primeiro membro é menor que o segundo
<=	TRUE se o primeiro membro é menor ou igual ao segundo membro
>	TRUE se o primeiro membro é maior que o segundo
>=	TRUE se o primeiro membro é maior ou igual ao segundo membro

Argumentos: <*var1*> <*var2*> são símbolos válidos de variáveis de aplicativo declaradas, ou variáveis E/S diretamente representadas que utilizam as convenções de escrita "% ".

<*expr_const*> . é uma expressão constante válida que combina com o tipo da variável especificada. Para booleanos, "0" e "1" podem ser utilizados em vez de "FALSE" e "TRUE". Para temporizadores, o prefixo "T#" ou "TEMPO#" pode ser omitido.

<*nome_do_rótulo*> .é o nome de uma etiqueta definida no script.

Observação: São permitidos desvios para trás. No caso de um "loop" infinito, a execução do script está automaticamente interrompida em cada loop para preservar a execução de ciclos ISaGRAF.

Exemplo:
```
(* Este script executa o "loop" até MyVar = TRUE *)
Loop:
If MyVar = TRUE Goto TheEnd
Print MyVar
Goto Loop
TheEnd:
```

End

Significado: Interrompe o script

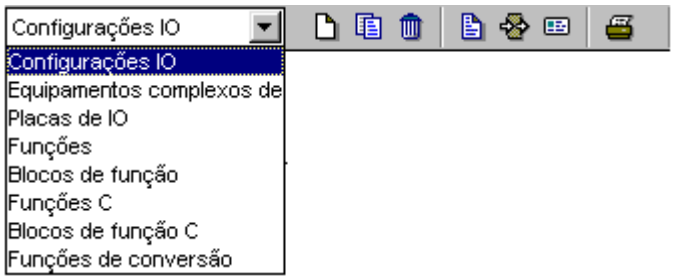
Sintaxe: **End**

Observação: Não é obrigatório colocar a instrução "End" na última linha do script

Exemplo:
```
(*Este script executa o "loop" até MyVar = TRUE *)
Loop:
If MyVar = FALSE Goto Continue
End
Continue:
Print MyVar
Goto Loop
```

A.22 Utilizando o Gerenciador de Biblioteca

As bibliotecas de ISaGRAF constituem a interface entre o desenvolvimento dos aplicativos e os recursos de **software** ou de **hardware** do sistema ISaGRAF destino. Há uma biblioteca para cada tipo de interface. O Gerenciador de Biblioteca ISaGRAF é dedicado ao fabricante do hardware ou ao engenheiro de software. Ele utiliza o gerenciador de biblioteca para a descrição das interfaces dos recursos de programação dos objetos que ele cria.
O Gerenciador de Biblioteca ISaGRAF mostra os elementos de um das bibliotecas ISaGRAF. Na área à esquerda da janela está a **lista dos elementos** da biblioteca selecionada. Na área à direita está a **observação técnica** (o manual do usuário) do elemento atualmente selecionado na lista de elementos. Os menus do gerenciador de biblioteca contêm os comandos para criar, definir ou modificar os elementos da biblioteca ativa. O comando "**Arquivo / Outra**" permite a seleção de um das bibliotecas de ISaGRAF. O caixa combo à esquerda da barra de ferramentas também pode ser utilizada para selecionar uma biblioteca:



A.22.1 Gerenciando elementos de biblioteca

Utilize os comandos do menu "**Arquivo**" para criar elementos e trabalhar sobre os já existentes na biblioteca aberta.

Criando um novo elemento

. O comando "**Novo**" do menu "**Arquivo**" cria um elemento novo na biblioteca selecionada. O nome do novo elemento é inserido, baseado nas seguintes regras:

- o comprimento de máximo de um nome é **8** caracteres
- o primeiro caractere deve ser uma **letra**
- os caractere seguintes podem ser **letras**, **algarismos** ou o caractere sublinha '_'
- o nome de um elemento de biblioteca não faz distinção entre maiúscula/minúscula.

Um texto de comentário está associado a cada elemento de biblioteca. Este comentário é inserido durante a criação do elemento. Quando um novo elemento é criado, o seguinte deve ser especificado:

- sua definição para uma configuração de E/S,
- seus parâmetros para uma placa de E/S,
- sua interface de usuário para uma função ou um bloco de função.

Quando uma conversão "C ", uma função “C " ou um bloco de função "C" é criado, um quadro completo do seu código de fonte é automaticamente gerado.

Trabalhando com elementos existentes

O comando "**Arquivo / Renomear**" permite que o usuário modifique o nome ou o comentário do elemento selecionado da lista de elementos. O comando "**Arquivo / Copiar**" permite que o usuário copie o elemento destacado na biblioteca ativa em outro elemento na mesma biblioteca. Se o elemento destino já existe, todo o seu conteúdo é sobrescrito. Se o elemento destino não existe, é criado automaticamente. O comando "**Arquivo / Apagar**" remove o elemento atualmente selecionado da biblioteca ativa. Os componentes seguintes do elemento são controlados pelos comandos "**Renomear**", "**Copiar**" e "**Apagar**":

- observação técnica
- definição completa para uma configuração de E/S
- parâmetros para uma placa de E/S ou equipamento complexo
- definição de interface para uma função ou bloco de função
- código de fonte para função e bloco de função escritos em linguagem IEC
- código de fonte para uma conversão C, uma função ou um bloco de função

⚠ se o elemento é uma conversão "C ", função “C " ou bloco de função "C", seu nome não é atualizado automaticamente no código fonte atrelado por um comando "**Renomear**" ou "**Copiar**".

⚠ se o elemento é uma função escrita em linguagem IEC, o nome de parâmetro de retorno não é modificado por um comando "**Renomear**" ou "**Copiar**".

Configuração da senha de proteção

O comando "**Arquivo / Determinar senha**" permite que o usuário defina a senha de proteção para o elemento selecionado na biblioteca aberta. Recorra à seção "**Proteção de dados**", mo término da primeira parte neste manual para informação adicional sobre níveis de senha e proteção de dados. As senhas só são relativas ao elemento selecionado. Elas não têm nenhuma influência em outros elementos de bibliotecas ISaGRAF.

Compilando funções e blocos de funções

Quando a biblioteca de funções ou blocos de função escrita em linguagens IEC é selecionada, o comando "**Verificar (compile)**" do menu "**Arquivo**" é utilizado para verificar a sintaxe do elemento selecionado e criar seu código de objeto. As funções e blocos escritos em linguagens IEC têm que ser compilados sem erros antes de poderem ser utilizados em projetos ISaGRAF. Este comando não tem nenhum efeito se outra biblioteca é selecionada.

Observações técnicas

O comando "**Editar / Observação Técnica**" permite que o usuário insira uma observação técnica do elemento selecionado na biblioteca ativa. A observação técnica é inserida com o editor de texto ISaGRAF. A observação técnica de um elemento é seu **guia de usuário**. Será consultado pelo usuário do elemento durante sua integração em um projeto ISaGRAF. A observação técnica em como utilizar o elemento deve conter a descrição de sua função principal, a explicação detalhada de sua interface de programação e parâmetros, e seu contexto e limites.

O comando "**Ferramentas / Formatação de observação padrão**" permite que o usuário defina um formato de texto padrão para todos os elementos da biblioteca atualmente selecionada. Ao editar a observação técnica para um elemento novo, este formato é utilizado como um quadro principal. Isto permite que o usuário otimize a edição da observação técnica.

Parâmetros

Os parâmetros de um elemento descrevem a **interface** entre as suas operações de computador providas pelo elemento e a utilização do elemento em um aplicativo ISaGRAF. Os parâmetros têm um significado diferente para cada tipo de elemento de biblioteca.
Os parâmetros de uma configuração E/S definem o conjunto completo de placas E/S da configuração e os nomes de variáveis padrões utilizados pelos canais E/S. Os parâmetros de uma placa E/S ou equipamento complexo definem a configuração de hardware e de software da placa. Os parâmetros de uma função ou bloco de função definem a interface do elemento, de acordo com as regras semânticas de chamada de função da linguagem ST. Não há nenhum parâmetro para uma função de conversão porque utiliza uma interface padrão predefinida.

Código fonte

O ambiente de trabalho ISaGRAF permite que o programador gere o código fonte das conversões, funções ou blocos de função da biblioteca. O código fonte de uma função ou um bloco escrito em linguagem IEC é um texto ou um diagrama correspondente à linguagem escolhida para a função. O código fonte de um elemento "C" (função “C", blocos de função “C" e funções de conversão) está dividido em dois arquivos separados: um arquivo de **definições** contendo a descrição exata de sua **interface**, de acordo com a definição dos parâmetros do elemento e um arquivo de **código fonte** contendo a implementação da função “C” realizada pelo elemento.
O ambiente de trabalho ISaGRAF gera o arquivo de código fonte quando um novo elemento de biblioteca é criado. Também cria e atualiza o arquivo de definições, cada vez que os parâmetros do elemento são modificados. O programador pode utilizar o texto editor ISaGRAF para completar o arquivo de código fonte.

Arquivando elementos de biblioteca

O comando do menu "**Ferramentas / Diretório**" executa o gerenciador de arquivo ISaGRAF para salvar ou restaurar elementos de biblioteca. Primeiro você precisa selecionar uma biblioteca antes de executar o comando **“Diretório**". O gerenciador de arquivo apresenta a lista de elementos para uma só uma biblioteca de cada vez.

A.22.2 Configuração de E/S

A biblioteca de configurações de E/S ISaGRAF oferece um meio fácil e rápido para inicializar os novos projetos ISaGRAF com uma configuração de E/S predefinida. Uma configuração de E/S define:

- um conjunto de placas de E/S e equipamentos complexos
- os valores padrões (“default”) para os parâmetros das placas
- os nomes padrões (“default”) para os canais E/S

Quando um novo projeto ISaGRAF é criado com uma configuração de E/S, a conexão de E/S correspondente é automaticamente realizada, e as variáveis de E/S associadas aos canais das placas são diretamente declaradas no dicionário do projeto criado.

A definição de uma configuração de E/S é realizada com a ferramenta Conexão de E/S ISaGRAF (a mesma ferramenta utilizada para os projetos) . Veja a seção "**Utilização do editor de conexão de E/S**" neste manual para informação adicional sobre como utilizar esta ferramenta. Durante a inserção de uma nova placa de E/S na configuração, todos os canais da nova placa são declarados com nomes padrões. O nome padrão de um canal de E/S tem o seguinte formato:

<direção><tipo><slot_número>_<canal_número>

O primeiro caractere indica a direção do canal de E/S:
"**I**".....canal de entrada
"**O**".....canal de saída

O segundo caractere indica o tipo do canal de E/S:
"**X**".....booleano
"**D**".....analógico
"**M**".....mensagem

A seguir, exemplos de nomes padrões para os canais de E/S :
IX0_7 .....entrada booleana - placa #0 - canal #7
QD2_4.....saída analógica - placa #2 - canal #4

O comando "Conectando os canais E/S" do Editor de Conexão E/S é utilizado para modificar o nome padrão atrelado a um canal E/S.

A.22.3 Equipamento complexo de E/S

Todos os canais de uma única placa têm o mesmo tipo (booleano, analógico ou mensagem) e direção (entrada ou saída). Um equipamento complexo de E/S representa um dispositivo de E/S com canais de tipos diferentes ou direções. Um equipamento complexo de E/S é representado como uma lista de placas elementares de E/S e só utiliza um slot no bastidor de conexão de E/S.

Para definir um equipamento complexo de E/S, o usuário tem que definir a lista de placas elementares que definem o equipamento de E/S. Ele também tem que entrar com os parâmetros detalhados de cada placa elementar. A lista de placas elementares de E/S é inserida em uma caixa de diálogo.
Pressionando o botão "**Adicionar**" o usuário pode adicionar uma única placa ao término da lista atual. O botão "**Inserir**" é utilizado para inserir uma nova placa elementar antes da atualmente selecionada na lista. O botão "**Apagar**" remove a placa elementar selecionada da lista. Os botões "**Renomear**" e "**Parâmetros**" são utilizados para modificar o nome e os parâmetros da placa elementar selecionada. Recorra à seção seguinte para uma explicação completa sobre os parâmetros da placa elementar. Um equipamento complexo de E/S pode se reunir até **16** placas elementares de E/S. O nome de uma única placa (dentro de um equipamento de E/S) não pode exceder **8** caracteres.

A.22.4 Placa E/S

A biblioteca ISaGRAF das placas E/S define uma interface padrão entre as variáveis do aplicativo e o hardware destino. Durante a descrição do aplicativo, todas as variáveis de E/S são conectadas aos canais das placas de E/S destino. Uma placa ISaGRAF de E/S é definida por um **nome** e um "**código chave OEM**" que identificam seu **provedor**. Outros parâmetros de placa E/S descrevem a topologia da placa E/S (número de canais, direção e tipo de canal) e a sua configuração de hardware ou de software.

Parâmetros da placa E/S

Há dois tipos diferentes de parâmetros para uma placa de E/S: parâmetros comuns que são definidos para qualquer placa da biblioteca ISaGRAF, e parâmetros OEM que são específicos para a implementação da placa, provido pelo fornecedor do hardware. Os parâmetros comuns são inseridos na parte superior da caixa de definição de parâmetros da placa E/S. Estes parâmetros (mais o nome da placa E/S) identifica a interface padrão de placa ISaGRAF E/S.
O "**código chave OEM**" é um número simples que define o fornecedor do hardware. Todas as placas definidas pelo mesmo fornecedor têm que ter o mesmo código chave OEM. O código chave OEM é uma **palavra de valor absoluto de 16 bits**, inserida no formato hexadecimal. O código chave OEM reservado para a **CJ International** é "**1**".
Parâmetros principais definem a topologia da placa de E/S. O **número de canais** define o número de canais disponíveis na placa. O **tipo** da placa é o tipo das variáveis que podem ser conectadas nos canais da placa. A **direção** define se variáveis conectadas na placa são de entrada ou de saída.

Obs: Variáveis E/S de tipos ou direções diferentes não podem ser agrupadas na mesma placa ISaGRAF de E/S. Este recurso requer um equipamento complexo de E/S.

Os parâmetros OEM

Os parâmetros OEM são inseridos na parte mais baixa da caixa de definição dos parâmetros da placa E/S. Estes parâmetros são definidos pelo fornecedor do hardware da placa de E/S e são específicos da placa. No máximo, há **16** parâmetros OEM para uma placa. Uma placa não pode ter nenhum parâmetro OEM. O gerenciador de biblioteca ISaGRAF permite que o fornecedor do hardware defina a identificação e o formato de cada parâmetro e o modo que o programador de automação faz a sua inserção.
A caixa à esquerda contém a lista dos parâmetros OEM. Cada parâmetro é identificado por um **nome** e um **número** lógico, de **0** a **15**. A área à direita contém a descrição detalhada do parâmetro selecionado na lista. Um parâmetro é selecionado na lista para se ter acesso à sua descrição completa. Pressionando o botão "**Limpar**" a descrição do parâmetro é reinicializada e removida da lista de parâmetros. Aviso: este comando não pode ser "desfeito".

O nome de um parâmetro é utilizado para identificar o campo de entrada correspondente durante a conexão da placa de E/S se o campo for definido pelo operador de automação. O nome de um parâmetro tem que estar de acordo com as seguintes regras:

- o comprimento máximo de um nome é **16** caracteres
- o primeiro caractere deve ser uma **letra**
- os caractere seguintes podem ser **letras**, **algarismos** ou o caractere de sublinha '_'

O tipo de um parâmetro define o **formato interno** do parâmetro e seu **formato de entrada** durante a conexão do aplicativo de E/S. A seguir, a lista de formatos internos disponíveis:

word.....palavra de valor absoluto de 16 bits
long .....palavra de valor absoluto de 32 bits
word hexa.....palavra de valor absoluto de 16 bits
long hexa.....palavra de valor absoluto de 32 bits
boolean.....palavra de valor absoluto de 16 bits (só o bit menos significativo é utilizado)
character .....palavra de valor absoluto de 16 bits (só o byte menos significativo é utilizado)
string.....tabela de 16 bytes contendo uma “string” terminada por "0"
float.....número de ponto flutuante de 32 bits, precisão simples

A seguir, os formatos de entrada disponíveis:

word.....palavra de valor absoluto decimal
long .....palavra decimal longa
word hexa.....palavra hexadecimal de valor absoluto
long hexa.....palavra longa hexadecimal
boolean....."TRUE" ou "FALSE"
character .....caractere único
string.....“string” ASCII (máximo de 15 caracteres)
float.....número de ponto flutuante, precisão simples

A caixa "**Acesso**" é utilizada para definir como o parâmetro pode ser acessada pelo usuário final. Se a opção "**Definido pelo Usuário**" está configurada, o parâmetro é mostrado como um campo de entrada durante a conexão da placa de E/S. O valor do parâmetro OEM padrão é utilizado como um padrão para a edição do parâmetro. Se a opção "**Escondido**" está configurada, o parâmetro é uma constante e não aparece na caixa de conexão da placa de E/S. O valor do parâmetro OEM padrão define o valor do parâmetro constante. A opção "**Apenas leitura**" indica que o parâmetro é visível para o usuário, mas não pode ser modificado. Seu valor padrão é utilizado como um valor constante.

A.22.5 Funções e blocos de funções escritos em linguagem IEC

O ISaGRAF controla uma biblioteca de funções e de blocos de funções escrita em linguagens IEC. As linguagens disponíveis para descrever tal função ou bloco de função são **FBD** (Diagrama em Blocos de Função), **LD** (Diagrama **“Ladder”**), **ST** (Texto Estruturado) ou **IL** (Lista de Instrução). Note que as linguagens LD e FBD podem ser misturados no mesmo diagrama. A linguagem de **SFC** (Gráfico de Função Seqüencial) não pode ser utilizada para descrever uma função ou um bloco de função na biblioteca. A linguagem atrelada a um elemento de biblioteca é selecionado quando a função é criada e não pode ser modificado depois.

Compilando

As funções e blocos de funções definidos na biblioteca devem ser compilados (verificados) antes que possam ser utilizados num projeto ISaGRAF. Nada mais tem que ser modificado no lado da Biblioteca relativo a funções e blocos de funções. Os elementos da biblioteca aparecerão automaticamente no menu caixa de seleção durante a utilização do editor gráfico LD/FBD dentro de um projeto.

Uma função definida na biblioteca pode chamar outras funções da biblioteca. Porém, o sistema ISaGRAF **não suporta recursividade** na chamada das funções. Um bloco de função escrito em linguagem IEC não pode chamar outros blocos de função (descritos em "C" ou em linguagem IEC).

Inserindo um código fonte

O código fonte de uma função ou bloco de função de biblioteca é inserido utilizando ferramentas ISaGRAF padrões: editor gráfico para programas LD ou FBD, editor de texto para programas ST ou IL. Veja as seções correspondentes neste manual, para mais informações sobre estas ferramentas. O Gerador de Código ISaGRAF pode ser chamado diretamente da janela de edição de gráfico ou de texto para compilar o código fonte de uma função ou bloco de função da biblioteca.

Dicionário de variáveis locais

Uma função ou bloco de função da biblioteca podem ter variáveis locais e palavras definidas locais. Para ter acesso à declaração variável, o usuário tem que executar os comandos do menu "**Arquivo / Dicionário**", na janela de editor durante a edição do código fonte da função.

Uma função ou bloco de função da biblioteca não pode acessar uma variável global ou uma instância de bloco de função. As variáveis locais de uma função devem ser inicializadas no corpo da função.

As variáveis locais de um bloco de função escrito em linguagem IEC são copiadas toda vez que o bloco é utilizado em um projeto. As variáveis locais de uma instância mantêm os seus valores de uma chamada para outra.

Definindo a interface

As funções ou blocos de função podem ter até **32** parâmetros (entrada ou saída). Uma função sempre tem um (e somente um) parâmetro de retorno que tem que ter o mesmo nome da função, para estar de acordo com as convenções de escrita da linguagem ST.
A lista no lado superior esquerdo da janela mostra os parâmetros, na ordem do modelo de chamada: primeiro os parâmetros de chamada, por último os parâmetros de retorno. A parte inferior da janela mostra a descrição detalhada do parâmetro selecionado na lista. Todos os tipos de dados ISaGRAF podem ser utilizados por um parâmetro. Os parâmetros de retorno devem ser os últimos da lista. A nomenclatura dos parâmetros deve respeitar as seguintes regras:

- o comprimento do nome não pode exceder **16** caracteres
- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes podem ser **letras**, **algarismos** ou o caractere de sublinha
- a nomeação não faz diferença entre maiúscula/minúscula

O comando "**Inserir**" é utilizado para inserir um novo parâmetro antes do parâmetro selecionado. O comando "**Apagar**" é utilizado para apagar o parâmetro selecionado. O comando "**Organizar**" automaticamente rearranja (classifica) os parâmetros, de forma que os parâmetros de retorno sejam postos no fim da lista.

A.22.6 Funções e blocos de funções em "C"

As funções e os blocos de função em "C" são **funções de computador** chamadas do aplicativo de automação, de acordo com as convenções de chamada definidas pelas linguagens ST e FBD.
As funções são processos **síncronos**. O aplicativo ISaGRAF destino está suspenso durante a execução de função. Os blocos de função associam operações e dados estáticos escondidos. Por exemplo, um bloco de função "contador" representa a operação de contagem, bem como o resultado da contagem. As funções e os blocos de função podem ser utilizados para completar as capacidades da linguagem de automação padrão, ou ter acesso aos recursos do sistema.

A caixa de definição de parâmetros é utilizada para definir o nome e o tipo de cada chamada ou parâmetro de retorno da função ou bloco de função. Os comandos do menu "**Editar**" são utilizados para definir os parâmetros da função ou do bloco de função selecionado. Uma função pode ter até **31** parâmetros de chamada, e sempre tem **um** parâmetro de retorno. Um bloco de função pode ter até **32** parâmetros, com qualquer mistura de parâmetros de chamada e de retorno. A seguir, a correspondência entre tipos ISaGRAF e tipos "C":

BOOLEAN	unsigned long	palavra de valor absoluto 32 bits: 1=true / 0=FALSE
ANALOG	long	palavra de valor inteiro de 32 bits, com sinal
REAL	float	valor ponto flutuante, precisão simples
TIMER	unsigned long	palavra de valor absoluto, inteiro de 32 bits (a unidade é 1 ms)
MESSAGE	char *	cadeia de caracteres (“string”).

Quando uma mensagem é passada a uma função ou um bloco de função "C", não pode conter caracteres nulos (código ASCII 00). A “string” passada para o código "C" é terminado em zero.
Veja o Guia do Usuário ISaGRAF Destino para informação adicional de como gerenciar o código fonte "C" de uma função ou um bloco de função e como integrar um elemento novo no sistema ISaGRAF destino.

A.22.7 Funções de conversão

Uma função de conversão é uma função "C" chamada pelo gerenciador ISaGRAF de E/S toda vez que variáveis analógicas que utilizam esta conversão inseridas ou retiradas do projeto.
A função cria a relação entre o **valor elétrico** da variável (lido do sensor de entrada ou enviado ao dispositivo de saída) e seu **valor físico** (utilizado nas expressões do aplicativo). A função é dividida então em duas partes: conversão de entrada e conversão de saída. O gerenciador de biblioteca ISaGRAF permite que o usuário controle o código fonte "C" de uma função de conversão.
Uma conversão pode ser utilizada por uma variável analógica **inteira** ou **real**. Isto implica que a interface de função de conversão sempre está definida para valores de ponto flutuante. A interface é a mesmo para qualquer função de conversão. A definição em "C" desta interface é feita no arquivo de definição "**TACN0DEF.H**".

Veja o Guia do Usuário ISaGRAF Destino para informação adicional de como gerenciar o código fonte "C" de uma função de conversão e como integrar um elemento novo no sistema ISaGRAF destino.

A.23 Utilizando o utilitário Diretório

O utilitário ISaGRAF **Diretório** permite que o usuário salve os projetos e bibliotecas ISaGRAF em disquetes ou diretório de "backup". O gerenciador ISaGRAF **Diretório** é uma caixa de diálogo que pode ser chamada a partir das janelas Gerenciamento de Projeto ou Gerenciamento de Biblioteca ISaGRAF.

⚠ Para criar e manter arquivos seguros, é sugerido que as seguintes diretrizes sejam utilizadas:

- Escreva o nome e descrição do objeto salvo na etiqueta do disco
- Não salve projetos e bibliotecas no mesmo disquete
- Não salve projetos diferentes no mesmo disquete

A.23.1 Chamando o gerenciador de arquivo

A caixa de diálogo "**Ferramentas / Diretório**" pode ser chamada a partir da janela de Gerenciamento de Projeto, para salvar ou restaurar um projeto, ou dados comuns.
A caixa de diálogo "**Diretório**" também pode ser executada a partir do comando "**Ferramentas / Diretório**" do Gerenciador de Biblioteca ISaGRAF, para salvar ou restaurar elementos da biblioteca atualmente selecionados na janela de Gerenciamento de Biblioteca.

Projetos

Um projeto sempre é salvo em sua totalidade. Todos os componentes do projeto (arquivos de programa fonte, código de objeto e código de aplicativo executável) são salvos juntos no mesmo **Diretório**. A seleção da opção "**comprimir**" reduz o tamanho do arquivo de projeto.

Elementos de biblioteca

Os elementos de bibliotecas ISaGRAF podem ser salvos individualmente. Todos os componentes de um elemento de biblioteca (observação técnica, definição, interface, código fonte...) são salvos juntos no mesmo **Diretório**.

Dados comuns

O comando "**Ferramentas / Diretório / Dados comuns**" da janela Gerenciamento de Projeto permite que o usuário faça uma cópia de segurança ("backup") ou restaure os dados "de alcances comuns" existentes no ambiente de trabalho ISaGRAF. Este comando não atua nas bibliotecas ISaGRAF. A seguir, a lista dos arquivos que podem ser copiados com este comando:
common.eqv.....palavras definidas comuns
oem.bat .....arquivo de comando MS-DOS definido pelo usuário

Estes arquivos são salvos um por um no disco de arquivo, nas suas formas originais. Os **Diretório**s correspondentes nunca estão comprimidos.

A.23.2 Opções

O caminho utilizado para arquivos ISaGRAF é mostrado na parte inferior da caixa de diálogo. Pressione o botão "**Localizar**" para passar pelos discos e selecionar outro Arquivo e Diretório.

Quando a opção "**Comprimir**" está configurada, todos os **Diretório**s criados durante um procedimento "**Salvar**" estão comprimidos. Esta opção é muito útil para reduzir o tamanho de um **Diretório** de projeto grande e salva em um só disquete. Geralmente não é necessária a compressão de arquivo por componentes de biblioteca. O Gerenciador de **Diretório** ISaGRAF reconhece o estado de um **Diretório** automaticamente (comprimido ou não) durante a sua restauração. Quer dizer que a opção "**comprimir**" não tem nenhum efeito para um procedimento "**Restaurar**".

A.23.3 Backup e restauração

A lista "**Workbench**" (à esquerda) mostra os objetos existentes no ambiente de trabalho ISaGRAF instalados no disco rígido. A lista "**Diretório**" (à direita) mostra os objetos salvos no Diretório.

- ***Backup***

Selecione o objeto a salvar na lista à esquerda (objetos do ambiente de trabalho ISaGRAF) e pressione o botão "**Salvar**". Diversos objetos na lista podem ser selecionados para a mesma operação de cópia de segurança. O botão "**Salvar**" é inválido quando um elemento é selecionado na lista à direita (modo restauração).

- ***Restaurar***

Selecione o objeto a ser restaurado na lista à **direita** (objetos armazenados no **Diretório**) e pressione o botão "**Restaurar**". Diversos objetos na lista podem ser selecionados para a mesma operação de restauração. O botão "**Restaurar**" é inválido quando um elemento é selecionado da lista à **esquerda** (modo "backup").

A.23.4 Diretório

O gerenciador de diretório ISaGRAF cria um arquivo para cada objeto salvo. O Archive tem o mesmo nome do objeto. Sua extensão (**.xxx**) indica o seu tipo. A seguir, as extensões utilizados:

.pia .....projeto

.bia .....Placa E/S
.iia.....função em linguagem IEC
.aia.....bloco de função em linguagem IEC
.uia .....função C
.fia .....bloco de função C
.cia.....função de conversão C
.ria.....configuração E/S
.xia.....equipamento E/S

A.24 Impressão de um documento completo

O Gerador de Documento ISaGRAF permite que o usuário elabore e imprima um documento completo para o projeto selecionado. Pode ser chamado pelos comandos "**Projeto / Imprimir**" das janelas de Gerenciamento de Projeto ou de Programa para imprimir um documento completo. O Gerador de Documento também é executado pelo comando "**Imprimir**" de todos os outros editores ISaGRAF para a impressão do conteúdo de um único documento ISaGRAF. Entretanto, o Gerador de Documento fornece as mesmas facilidades em ambos os casos.
Os comandos do menu "**Editar**" são utilizados para definir os elementos do projeto que devem ser inseridos no documento. Fazendo isto, o usuário elabora o "índice" para o documento desejado. Qualquer informação sobre o projeto (programas, variáveis, opções, conexão de E/S...) pode ser inserida no documento de projeto. Nenhuma informação de outro projeto ou de bibliotecas ISaGRAF podem aparecer neste documento.

O comando "**Arquivo / Imprimir**" gera e envia o documento à impressora, de acordo com o índice especificado. O trabalho de "Impressão" pode levar alguns minutos para elaborar e formatar o documento. É altamente recomendado esperar até que o "Trabalho de Impressão" esteja terminado na janela Gerador de documento ISaGRAF, antes de executar outros comandos do ambiente de trabalho ISaGRAF. A elaboração de todo o documento pode necessitar de um espaço grande no disco rígido. Uma mensagem de erro será exibida se o disco estiver cheio. Em tal caso, o usuário terá que liberar espaço no disco removendo alguns arquivos ou reduzindo o tamanho do trabalho de impressão. Quando o comando "**Imprimir**" é executado, uma caixa de diálogo é mostrada permitindo que o usuário insira uma nota descrevendo o comando de impressão atual. Estas notas são armazenadas em um arquivo de histórico e serão impressas na primeira página de qualquer futuro documento (inclusive este).

A.24.1 Personalizando o índice

O menu "**Editar**" contém os comandos para a definição do "Índice" do documento. Uma escolha de comandos permite que o usuário utilize um índice padrão (com todos os componentes do projeto), elabore um índice específico (com só alguns componentes) ou mova itens no índice e modifique-o.

A lista padrão

O comando "**Lista padrão**" do menu "**Editar**" define um índice padrão para o documento que inclui todos os componentes do projeto. A seguir, o conteúdo deste índice:
- Descrição de projeto
- Árvore hierárquica (ligações entre programas)
- Código fonte para qualquer programa
- Arquivo de diário para qualquer programa
- Definições comuns
- Definições globais
- Definições locais para qualquer programa
- Variáveis globais
- Variáveis locais para qualquer programa

- Opções de aplicativo
- Conexão de E/S
- Listas de variáveis
- Tabelas de conversão
- Referências cruzadas condensadas
- Referências cruzadas detalhadas
- Resumo de declaração
- Mapa de endereçamento de rede
- Histórico de modificações

O índice pode ser salvo em disco utilizando o comando "**Arquivo / Salvar**". Este comando é destacado em cinza quando o gerador de documento é executado a partir de um editor ISaGRAF para a impressão de um único documento.

Recortar e colar

Utilize os comandos "**Editar / Recortar**" e "**Editar / Colar**" para mover itens na lista para personalizar a ordem da tabela. O Gerador de Documento permite a seleção múltipla de modo que um grupo de itens pode ser recortado e colado.

Limpando o índice

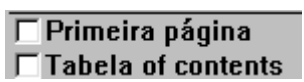
Utilize o comando "**Editar / Limpar**" para reinicializar o índice, de forma que este possa ser totalmente re-elaborado utilizando uma única inserção de item.

Inserindo itens na tabela

Quando o comando "**Editar / Inserir**" é executado, a caixa de diálogo "**Adicionar item**" é mostrada permitindo que o usuário insira itens (componentes do projeto) no índice.
Para um item relativo a um programa, utilize a caixa combo "**Programa**" para selecionar um nome de programa. Pressione o botão "**Adicionar**" para inserir o item selecionado no índice. O mesmo item só pode aparecer uma vez no índice.

A.24.2 Opções

Os comandos "**Opções**" são utilizados para definir e personalizar o formato do documento gerado. Outras opções estão disponíveis diretamente dos botões da janela de Gerador de Documento:



Quando a opção "**Primeira página** " está configurada, um cabeçalho de página é impresso no começo do documento, contendo o título do projeto e o histórico de impressões. Quando esta opção não está configurada, o primeiro item a ser impresso começa na primeira página.
Quando a opção "**Índice**" está configurada, um índice é impresso no fim do documento gerado.
Por padrão, estas duas opções não estão configuradas quando o Gerador de Documento é executado a partir do comando "**Imprimir**" de um editor ISaGRAF (programa, dicionário...)

Gráficos SFC

A opção "**Separar níveis SFC**" permite imprimir, para cada programa SFC, primeiro o nivela 1 do SFC (quadro e comentários), e então a programação do nível 2. Quando esta opção não está selecionada, os níveis 1 e 2 de um programa SFC aparecem juntos na mesma impressão.

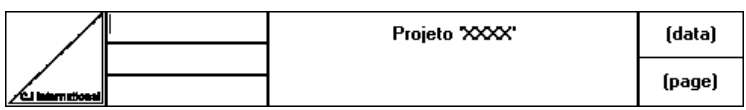
Formato de página

O comando "**Configurar página**" do menu "**Opções**" é utilizado para definir os principais parâmetros operados pelo Gerador de Documento ao formatar uma página. Os seguintes parâmetros podem ser especificados:

- **Margem Esquerda**: (1 ou 2 centímetros ou sem margem)
- **Borda**: quando esta opção é selecionada, uma borda é desenhada em volta da página impressa.

Modelo de página título

O comando "**Título da página**" do menu "**Opções**" é utilizado para definir o conteúdo da caixa de título impresso na parte inferior (rodapé) de qualquer página. O layout padrão desta caixa é o seguinte:



A primeira linha do título principal (com o nome do projeto ISaGRAF), a data atual e o número da página são gerados automaticamente pelo Gerenciador de Documento e não pode ser modificado.
As três linhas de texto no lado esquerdo da caixa (text1, text2, text3) e a segunda linha do título principal são definidas pelo usuário. O usuário também pode modificar o logotipo impresso na caixa à esquerda. Para utilizar outro logotipo, o usuário tem que especificar o nome do caminho de um arquivo de imagem bitmap (.BMP). A imagem pode ter qualquer dimensão. Será ampliado ou será diminuído, de acordo com as dimensões exatas da página impressa. Clicando sobre a área do logotipo, na caixa de diálogo, uma nova imagem especificada é mostrada. O arquivo de imagem deve estar no disco (no diretório especificado e com o nome de arquivo especificado) quando o comando "**Imprimir**" é executado.

Selecionando fontes de caractere

Os comandos "**Fonte do Texto**" e "**Fonte do Título**" do menu "**Opções**" são utilizados para definir as fontes de caracteres utilizadas durante a impressão de títulos e de texto para qualquer item do documento. Também podem ser selecionados o tamanho e o estilo de caractere para texto e títulos. A seleção de uma fonte é feita com a caixa de diálogo padrão definida pelo Windows. Qualquer texto (programas literais, nomes dentro de diagramas...) será impresso com o tamanho selecionado, estilo e fonte de caractere. Só os títulos com a fonte selecionada para os títulos serão impressos.
Se as fontes de caractere não estão definidas, a fonte padrão da impressora é utilizada para qualquer texto, com os seguintes estilos:

- Estilo ‘’Normal" para textos e nomes dentro dos diagramas
- Estilo ‘’Negrito" para títulos

A.25 Proteção por senha

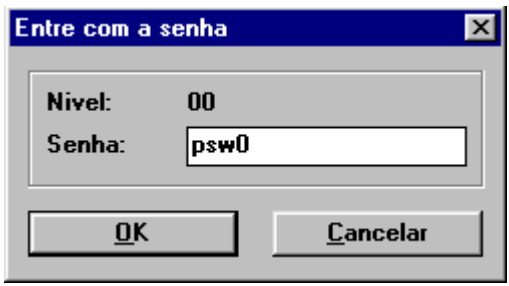
O ambiente de trabalho ISaGRAF inclui um sistema de proteção de dados completo que permite que o usuário proteja os projetos e os elementos de biblioteca com uma senha. Um elemento de biblioteca pode ser uma configuração de E/S, uma placa de E/S ou equipamento complexo, uma função ou bloco de função escritos em linguagens de IEC, uma função "C", bloco de função ou função de conversão. Um sistema de proteção é dedicado a um projeto ou a um elemento de biblioteca, e não pode ser compartilhado entre vários projetos ou elementos.

Níveis de proteção

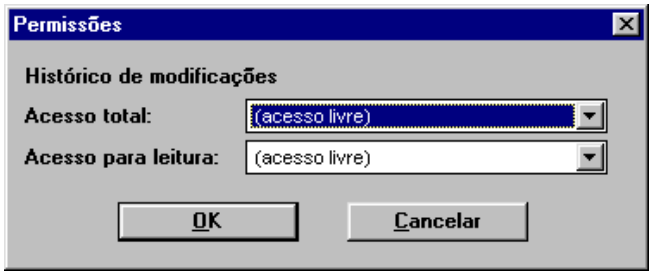
Para um projeto ou um elemento de biblioteca, o usuário pode definir até **16** níveis de acesso, correspondendo a senhas diferentes. Os níveis de acesso são ordenados em um sistema hierárquico. Eles são numerados de **0** a **15**. O nível de acesso mais alto é numerado 0. Quando um usuário do ambiente de trabalho conhece a senha, ele pode ter acesso a todos os dados correspondentes aos níveis inferiores ou iguais aos definidos nesta senha. Cada comando ou dado elementar de um projeto ou elemento de biblioteca pode ser protegido separadamente com um nível de acesso. Por exemplo, o comando "Compilar" dos menus ISaGRAF pode ser protegido separadamente. Dados elementares podem ser um programa, uma lista de opções, a observação técnica de um elemento de biblioteca, etc...

Definindo a senha de proteção

O comando "**Determinar senha**" dos menus ISaGRAF é utilizado para definir as senhas e os níveis de acesso para um projeto ou um elemento de biblioteca. Este comando é chamado dos menus do Gerenciador de Projeto ISaGRAF (para um projeto) ou do Gerenciador de Biblioteca ISaGRAF (para um elemento de biblioteca). Nenhuma senha é solicitada na execução deste comando pela primeira vez. Se senhas já estão definidas, o usuário tem que entrar com a senha de nível mais alto para ter acesso este comando. As senhas e as proteções associadas aos níveis mais altos não podem ser modificadas. O comando "**Determinar senha**" permite que o usuário defina as senhas que correspondem aos diferentes níveis de acesso e proteja comandos ou dados elementares com os níveis definidos.
As senhas (correspondendo aos níveis de proteção) são inseridas clicando duas vezes em uma linha da lista superior. A seguinte caixa é utilizada para inserir uma senha.



A lista na área mais baixa mostra os diferentes itens (dados ou funções) que podem ser protegidos, e o nível de proteção atual atrelado aos acessos "somente leitura" ou "controle total". Designar um nível de proteção de "somente leitura" previne que outros usuários abram ou imprimam um documento.
Clique duas vezes sobre uma linha da lista inferior para modificar as permissões designadas para um elemento. A seguinte caixa é mostrada:



As duas permissões podem ser designadas "livre acesso" ou para um nível de proteção definido por uma senha. O nível escolhido para o controle total não pode estar atrelado a um nível com menor prioridade que o do nível escolhido para somente leitura.
Note que para alguns documentos, naturalmente visível ao utilizar os Ambiente de Trabalho ISaGRAF, como a descrição de projeto, não podem ter um modo "somente leitura" protegido por senha.

Acessando os dados protegidos

Nenhuma senha ou nome de usuário é solicitado quando o ambiente de trabalho ISaGRAF é executado. Toda vez que um usuário deseja ter acesso a dados ou função protegidos ele tem que entrar com a senha solicitada na caixa de diálogo.
Se o usuário entra com a senha solicitada (ou uma senha atrelada a um nível de acesso mais alto), ele pode continuar normalmente. Toda vez que uma senha é inserida pelo usuário, ela é armazenada em memória, de modo que o usuário não terá que entrar novamente depois disto. As senhas armazenadas são retidas toda vez que uma ferramenta ISaGRAF é executada de outra ferramenta ISaGRAF (por exemplo, o Gerenciador de Projeto executa Gerenciador de Programa). As senhas armazenadas são perdidas quando a última janela de ISaGRAF restante for fechada. As senhas inseridas durante a edição de projeto, ou utilizando o Gerenciador de Biblioteca, ou usando o Gerenciador de Arquivo não pode ser compartilhada. Se o usuário inserir uma senha ruim, ele não pode executar a função selecionada.

Links com o gerenciador de arquivo

Durante o salvamento de um objeto (projeto ou elemento de biblioteca) em disquete, a proteção do comando denominado "**Salvar Diretório**" é testada. Isto corresponde ao sistema de proteção de dados atrelados ao objeto no ambiente de trabalho (sobre o disco rígido). Nenhum teste é realizado no sistema de proteção de dados do objeto no arquivo, se ele já existe. O comando "**Salvar**" do Gerenciador de Arquivo ISaGRAF salva a informação de proteção de dados com o objeto.
Ao restaurar um objeto já existente no ambiente de trabalho (sobre o disco rígido), a proteção do comando denominado "**Rescreve arquivo existente**" é solicitada. Isto

corresponde ao sistema de proteção de dados atrelado ao objeto no ambiente de trabalho (disco rígido). Nenhum teste é realizado no sistema de proteção de dados do objeto no arquivo. Se este comando é validado, as informações de proteção de dados restauradas substituirão as existentes no disco rígido.

Proteção individual para as variáveis e os canais de E/S

O ambiente de trabalho ISaGRAF fornece um sistema de proteção de dados completo baseado em senhas hierarquizadas. Declaração variável e conexão de E/S podem ser protegidas globalmente por uma senha. Adicionalmente, o ISaGRAF permite que o usuário configura proteção individual a qualquer variável ou canal de E/S. Isto assume que:
- as senhas já estão definidas no sistema de definição de senha (utilize o comando "**Projeto / Determinar senha**" da janela de Gerenciamento de Projeto) de forma que os níveis de proteção estão disponíveis para proteção individual.
- você utiliza níveis de proteção individual com prioridade mais alta que os níveis utilizados para a proteção global de variável ou E/S.

Quando uma variável ou um canal de E/S tem proteção individual, um pequeno ícone é desenhado perto do seu nome na janela de dicionário ou conexão de E/S.
Utilize os comandos "**Determinar proteção**" e "**Remover proteção**" do menu "**Editar**" nas janelas de dicionário ou conexão de E/S para configurar ou remover uma proteção individual para a variável ou o canal selecionado. Ambos os comandos lhe pedem que entre com uma senha válida de forma que um nível de proteção pode ser atrelado à variável ou canal. Então, toda vez que desejar modificar uma variável ou uma conexão a um canal que tenha proteção individual você deve entrar com uma senha com nível de prioridade suficiente.

Aviso: se uma variável ou canal é protegido com um nível, e a senha correspondente é removida do sistema de proteção, e se nenhuma senha de nível mais alto está definida, a variável ou o canal não podem mais ser modificados a menos que uma nova senha com nível suficiente esteja definida.

A.26 Técnicas de programação avançada

Este capítulo contém mais informação sobre o Ambiente de Trabalho ISaGRAF e o sistema destino. Ele é dedicado ao usuário já familiarizado com as ferramentas e os métodos ISaGRAF.

A.26.1 Mais sobre as ferramentas ISaGRAF

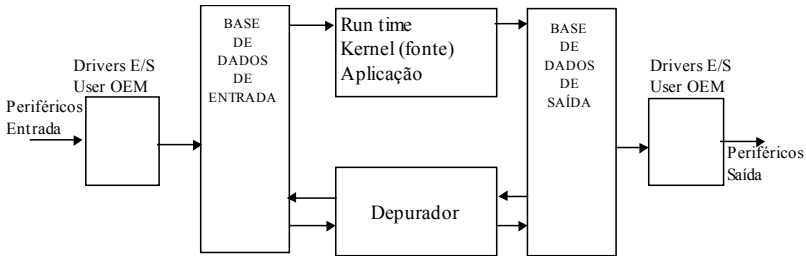
Ao utilizar as ferramentas de edição ISaGRAF, o usuário pode pressionar o botão direito do mouse para abrir um menu instantâneo que contém os principais comandos de edição. O menu é aberto na posição atual do cursor. Isto é muito útil para reduzir as operações de mouse durante os comandos de recortar e colar.
As ferramentas ISaGRAF suportam **execução múltipla**. Embora a mesma ferramenta não possa ser aberta para editar o mesmo documento duas vezes, é possível abrir diferentes janelas com a mesma ferramenta e editar objetos diferentes como operações paralelas.
Outros comandos estão disponíveis para localizar informação sobre os botões de gráfico nas barras de ferramentas. Clique duas vezes sobre uma área vazia de uma barra de ferramenta para exibir o seu conteúdo como um menu instantâneo. Ficar com o cursor do mouse sobre um botão gráfico durante um certo tempo faz apresentar o texto correspondente ao comando associado.

A.26.2 E/S Bloqueado e Virtual

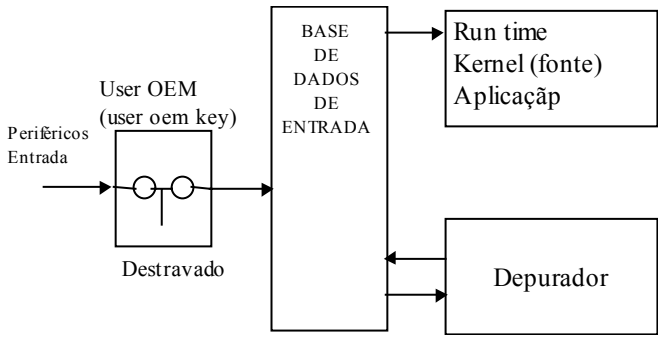
A definição de uma placa de E/S como **virtual** desconecta o processo dos canais de E/S físicos. Quando uma placa está definida como virtual, as operações do ISaGRAF núcleo não são modificadas. A única diferença é que os sensores de entrada não estão prontos e os dispositivos de saída não são atualizados. Neste modo, é possível utilizar o depurador ISaGRAF para modificar os valores de entrada. O atributo **Virtual** refere-se a uma placa completa. É programado durante a definição da placa de E/S, **antes** da geração do código de aplicativo. O atributo **Virtual** é uma característica **estática**, e é armazenado quando a aplicativo é interrompido e reinicializado.
Outra possibilidade é o bloqueio das variáveis de E/S. O bloqueio consiste em desconectar um dispositivo físico e a variável ISaGRAF E/S correspondente. O bloqueio e o desbloqueio de variável é executada pelo depurador e é uma operação **dinâmica**; não é memorizado quando o aplicativo é reinicializado. A operação de **bloqueio** se aplica a uma só variável (um canal de E/S) de cada vez. A seguir, o resumo das funções principais de controle de E/S:

	Atributo Virtual	*Comando bloqueio*
Seleção ferramenta	conexão placa E/S	depurador
definição	estático	dinâmico
modo seleção	placa	variável
aplicativo	validação e testes	manutenção

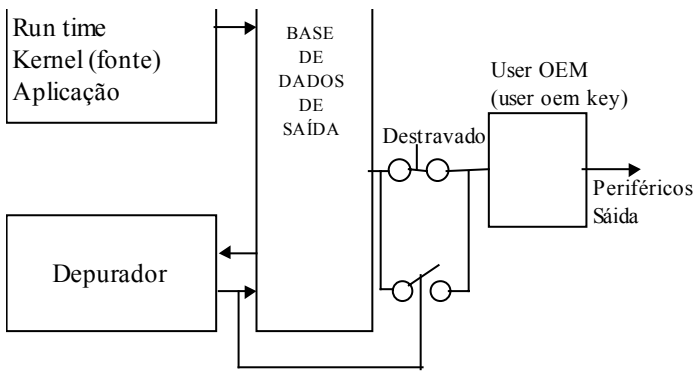
O diagrama a seguir mostra o fluxo de dados de E/S entre os diferentes módulos do ISaGRAF:



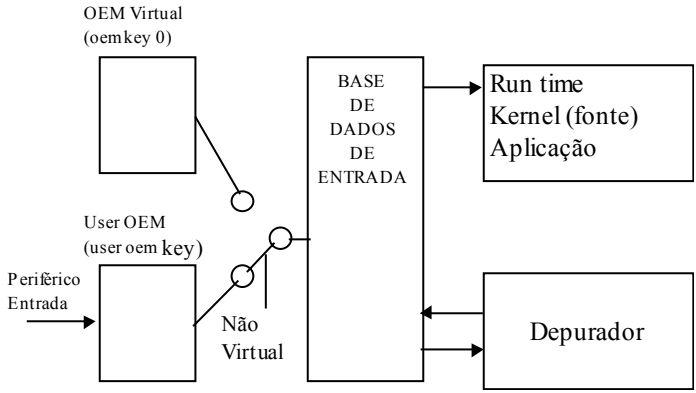
Quando uma variável de entrada é bloqueada, os vários acessos ao banco de dados não são modificados, mas o dispositivo de entrada está desconectado. Os valores de entrada são encaminhados para o depurador e podem ser processados pelo núcleo ISaGRAF:



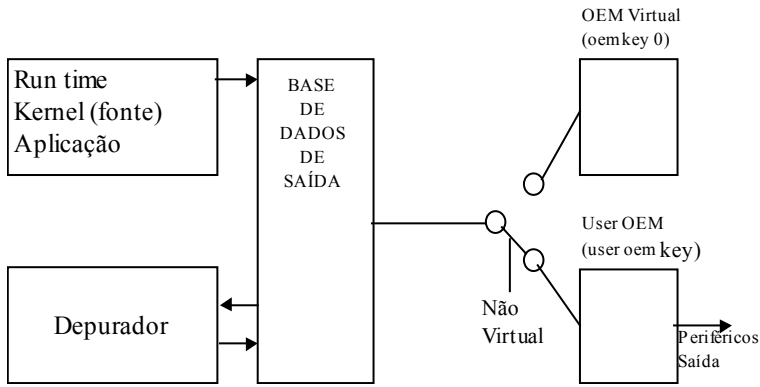
Quando uma variável de saída é bloqueada, o núcleo de execução e o driver de saída estão desconectados. Neste caso, o acesso ao dispositivo de saída ainda é possível, através do driver de saída, com o depurador ISaGRAF,:



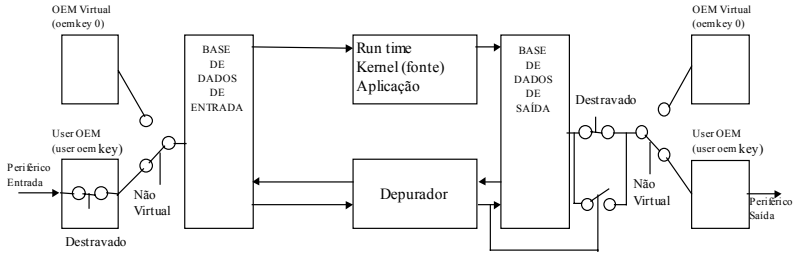
Ao fixar o atributo virtual para uma entrada, o banco de dados de entrada e os dispositivos de entrada associados estão desconectados. Um driver de E/S virtual substitui o real.



A definição do atributo virtual segue as mesmas regras para uma placa de entrada ou uma placa de saída. Para placas de saída, o núcleo ISaGRAF atualiza o banco de dados de saída. Porém, este banco de dados e os dispositivos de saída associados são desconectados. Um driver de E/S virtual substitui o real.



Para resumir todas as possibilidades:



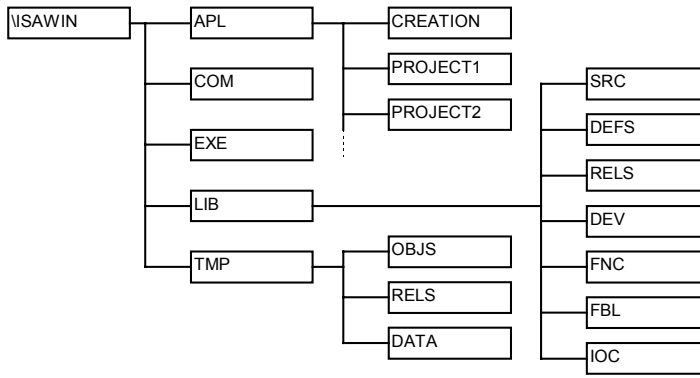
A.26.3 Validação do link PC-PLC

A maioria dos problemas devido a uma má comunicação entre o ambiente de trabalho ISaGRAF e o PLC destino são apresentados pela mensagem "**Desconectado**" na janela do depurador. Antes de qualquer diagnóstico ser executado, a comunicação deve ser validada quando **nenhum aplicativo está ativo** no PLC destino. Deste modo o link de comunicação serial pode ser validado por si só, isolando-o dos efeitos relativos à execução.
A linguagem "C", utilizado para descrever as funções de conversão e funções C, permite o acesso direto ao sistema destino. Um erro de programação em tal componente de software pode gerar erros de sistema ou comportamento incorreto do sistema ISaGRAF. Tais problemas podem acontecer quando os drivers de E/S são desenvolvidos com a ferramenta "Toolkit de E/S" ("**IO Toolkit**") do ISaGRAF. Por exemplo, os erros de sistema podem ser causados se uma placa de E/S está conectada a um endereço de barramento inválido. A tabela a seguir resumo os principais diagnósticos de erros de comunicação entre o ambiente de trabalho ISaGRAF e o ISaGRAF destino:

estado	contexto	Diagnóstico
"**Desconectado**" (antes da carga)		- destino não está funcionando - sem cablagem / cablagem inválida - parâmetros de link inválidos - ISaGRAF destino mal instalado
"**Desconectado**" (após a carga)	Iniciar em ciclo a ciclo	- configuração inválida de E/S - erro de sistema
	Iniciar em Tempo real	- configuração inválida de E/S - erro de sistema (devido a programação "C")
"**Aplicação inexistente**"		- aplicativo não carregado - aplicativo não iniciado (devido à programação "C") - incompatibilidade Intel/Motorola - Versão destino inválida

A.26.4 Diretórios ISaGRAF

O Ambiente de Trabalho ISaGRAF trabalha sobre uma estrutura de diretório dedicada. O diretório raiz desta arquitetura é especificado pelo usuário durante a instalação do ISaGRAF. O nome padrão ("default") para o diretório raiz ISaGRAF é "**ISAWIN**". Estes são os diretórios criados pelo programa de instalação:



Estes são os subdiretórios ISaGRAF padrões

DIRETÓRIO	CONTEÚDO
APL	Diretório de base para os projetos do ISaGRAF : Cada projeto corresponde a um subdiretório que contém todos os dados do projeto. Outros diretórios podem existir para outros grupos de projeto. O programa instalador ISaGRAF cria o diretório "**SMP**" em que são instalados os projetos de exemplo.
COM	Dados de alcance "comum" Dados que podem ser utilizados por qualquer projeto
EXE	Programas ISaGRAF e arquivos de ajuda
LIB	Bibliotecas ISaGRAF: - listas de elementos - interface, parâmetros para cada elemento - observações técnicas
LIB\IOC	Código fonte para configurações de E/S
LIB\FNC	Código fonte de funções em linguagem IEC
LIB\FBL	Código fonte de blocos de função em linguagem IEC
LIB\SRC	Código fonte dos elementos em linguagem C
LIB\DEFS	Arquivos de definições em linguagem C
LIB\RELS	Código de objeto de conversões e funções C
LIB\DEV	Arquivos de comando para o desenvolvimento em linguagem "C" de bibliotecas : Arquivos compilados, arquivo de edição de ligações, etc...
TMP	Arquivos temporários: subdiretórios de TMP são reservados para o Gerador de Código ISaGRAF e não podem ser apagados.

Os subdiretórios podem ser movidos para outros locais do disco. Se uma arquitetura não padrão é utilizada, os caminhos dos subdiretórios devem ser declarados na seção "**WS001**" do arquivo de inicialização "**ISA.ini**", no subdiretório **EXE** do ISaGRAF. Aqui estão as entradas da seção de "**WS001"**:

Isa	Caminho do diretório raiz do ISaGRAF

IsaExe	Caminho do diretório dos executáveis ISaGRAF
IsaApl	Caminho do diretório raiz para projetos ISaGRAF
IsaTmp	Caminho do diretório para arquivos temporários
IsaSrc	Caminho do diretório para arquivos de código fonte C
IsaDefs	Caminho do diretório para arquivos de definições C

Note que se você modificar a entrada IsaTmp, você tem que criar os subdiretórios OBJS, RELS e DATA no novo diretório. O exemplo a seguir mostra as entradas da seção "**WS001**"para uma arquitetura padrão:

;file c:\ISAWIN\EXE\ISA.ini

[WS001]
Isa=c:\isawin
IsaExe=c:\isawin\exe
IsaApl=c:\isawin\apl
IsaTmp=c:\isawin\tmp
IsaSrc=c:\isawin\lib\src
IsaDefs=c:\isawin\lib\defs

Quando você quer juntar as funções ou os blocos de funções "C" ao ISaGRAF destino, o diretório **\ISAWIN\LIB\DEV** pode ser utilizado pelos arquivos de desenvolvimento: arquivos de comando, arquivos compilados, mapas, etc... O diretório de **\ISAWIN\LIB\RELS** pode ser utilizado pelos arquivos de objeto gerados durante a compilação "C", e os arquivos de biblioteca ISaGRAF "C" requeridas para as operações LINK.

A.26.5 Tabelas de símbolos

. Cada objeto de uma aplicativo ISaGRAF é identificado por um nome (inserido durante a declaração das variáveis) e um **endereço virtual** interno, calculado pelo gerador de código. O endereço virtual de uma variável não deve ser confundido com o seu **endereço de rede** inserido durante a declaração de uma variável. Os endereços virtuais são utilizados nas operações de comunicação e aplicativos "C" especiais que utilizam a opção OEM. Quando o gerador de código ISaGRAF é executado, cria um arquivo ASCII com a correspondência lógica entre os nomes e os endereços virtuais para todos os objetos (variável, programas, etapas...) do projeto. Este arquivo pode ser facilmente consultado para informação sobre o banco de dados estático ISaGRAF de qualquer aplicativo de usuário. O arquivo é nomeado "**APPLI.TST**" e fica situado no diretório do projeto de ISaGRAF: "**\ISAWIN\APL\proname**" (proname é o nome do projeto). Esta seção descreve o formato detalhado do arquivo de "**APPLI.TST**". As anotações principais utilizadas para as descrições a seguir, são mostradas abaixo:

VA	endereço virtual
ATTR	atributo de uma variável
USP	função "C"

Possíveis valores para os atributos de uma variável são mostrados abaixo. Tais valores ocorrem nos campos de "**atributos**":

+X	variável interna
+C	variável interna de leitura
+I	variável de entrada
+O	variável de saída

Todos os números, exceto endereços virtuais, são expressos como inteiros decimais. Os endereços virtuais (**VA**) são expressos como números hexadecimais de 4 dígitos, e são precedidos pelo caractere "**!**". Por exemplo:

123	este é um número decimal
!A003	este é um endereço virtual hexadecimal

A estrutura principal do arquivo "**APPLI.TST**" é mostrada a seguir. O arquivo é estruturado como uma lista de **blocos**. Um bloco é uma lista de **registros**. Cada registro é descrito em uma linha de texto. Cada bloco começa com um cabeçalho, colocado em uma linha de texto.

Start block
description blocks
end block

A sintaxe geral de um bloco é mostrada abaixo:

@ <nome_do_bloco> <argumentos>
#record...
#record...
...

A estrutura do primeiro bloco, contendo a informação principal sobre o aplicativo, é mostrado a seguir:

@ISA_SYMBOLS,<appli_crc>
#NAME,<appli_name>,<version>
#DATE,<creation_date>
#SIZE,G=<nbprg>,S=<nbstep>,T=<nbtra>,L=0,P=<nbpro>,V=<nbvar>
#COMMENT,cj international

appli_crc.....checksum (verificação) do arquivo de símbolos
appli_name.....nome do aplicativo
version .....número da versão do ambiente de trabalho ISaGRAF
creation_date.....data de geração do aplicativo
nbprg .....número de programas
nbstep.....número de etapas SFC
nbtra .....número de transições SFC
nbpro .....número de funções "C" utilizadas
nbvar.....número total de variáveis

A estrutura do último bloco que sinaliza o fim do arquivo, é mostrada abaixo:

@END_SYMBOLS

A estrutura do bloco utilizada para descrever os programas da aplicativo, é mostrada a seguir:

@PROGRAMS,<nbprg>
#<va>,<nome>
#...

nbprg .....número de programas definidos neste bloco
va.....endereço virtual do programa
name.....nome do programa

A estrutura do bloco utilizado para descrever as etapas SFC do aplicativo é mostrado a seguir. Note que há uma etapa virtual definida para cada programa não -SFC:

@STEPS,<nbsteps>
#<va>,<name>,<father>
#...

nbsteps .....número de etapas definidas neste bloco
va.....endereço virtual da etapa
name.....nome da etapa
father .....endereço virtual do programa principal

A estrutura do bloco utilizada para descrever as transições SFC do aplicativo é mostrado a seguir:

@TRANSITIONS,<nbtrans>
#<va>,<name>,<father>
#...

nbtrans.....número de transições definidas neste bloco
va.....endereço virtual da transição
name.....nome da transição
father .....endereço virtual do programa principal

A estrutura do bloco utilizada para descrever as variáveis booleanas do aplicativo é mostrado a seguir:

@BOOLEANS,<nb_boo>
#<va>,<name>,<attr>,<program>,<eq_FALSE>,<eq_true>
#...

e se número da variável excede 4095:

X#(1.<varno>),<name>,<attr>,<program>,<eq_FALSE>,<eq_true>

nb_boo .....número de variáveis neste bloco
va.....endereço virtual da variável
varno.....alcance do endereço (dentro do tipo de dados booleanos)
name.....nome da variável

attr .....atributo da variável
program.....endereço virtual do programa parente
.....ou "**!0000**" para uma variável global
eq_FALSE ....."string" utilizada para valor FALSE
eq_true....."string" utilizada para valor TRUE

A estrutura do bloco utilizada para descrever as variáveis analógicas do aplicativo, é mostrado a seguir:

@ANALOGS,<nb_ana>
#<va>,<name>,<attr>,<program>,<format>,<unit>
#...

e se número da variável excede 4095:

X#(2.<varno>),<name>,<attr>,<program>,<eq_FALSE>,<eq_true>

nb_ana .....número de variáveis neste bloco
va.....endereço virtual da variável
varno.....alcance do endereço (dentro do tipo de dados analógicos)
name.....nome da variável
attr .....atributo da variável
program.....endereço virtual do programa parente
.....ou "**!0000**" para uma variável global
format .....= "**I**" para uma variável inteira
.....= "**F**" para uma variável real
unit....."string" unitária

A estrutura do bloco utilizada para descrever as variáveis de temporização do aplicativo, é mostrado a seguir:

@TIMERS,<nb_tmr>
#<va>,<name>,<attr>,<program>
#...

e se número da variável excede 4.095:

X#(3.<varno>),<name>,<attr>,<program>,<eq_FALSE>,<eq_true>

nb_tmr .....número de variáveis neste bloco
va.....endereço virtual da variável
varno.....alcance do endereço (dentro do tipo de dados de temporização)
name.....nome da variável
attr .....atributo da variável (sempre +X: interno)
program.....endereço virtual do programa parente
.....ou "**!0000**" para uma variável global

A estrutura do bloco utilizada para descrever as variáveis de mensagem do aplicativo, é mostrado a seguir:

@MESSAGES,<nb_msg>
#<va>,<name>,<attr>,<program>,< max_len>
#...

e se número da variável excede 4.095:

X#(4.<varno>),<name>,<attr>,<program>,<eq_FALSE>,<eq_true>

nb_msg.....número de variáveis neste bloco
va.....endereço virtual da variável
varno.....alcance do endereço (dentro do tipo de dados de mensagem)
name.....nome da variável
attr .....atributo da variável
program.....endereço virtual do programa parente
.....ou "**!0000**" para uma variável global
max_len.....comprimento máximo (capacidade declarada)

A estrutura do bloco utilizada para descrever as funções "C" utilizadas no aplicativo, é mostrado a seguir:

@USP,<nb_usp>
#<va>,<name>
#...

nb_usp.....número de funções C neste bloco
va.....endereço virtual da função C
name.....nome da função C

A estrutura do bloco utilizada para descrever as instâncias de bloco de funções "C" utilizadas no aplicativo, é mostrado a seguir:

@FBINSTANCES,<nb_fb>
#<va>,<inst_name>,<fb_name>
#...

nb_fb.....número de instâncias de blocos de função C neste bloco
va.....endereço virtual da instância de bloco de função C
inst_name.....nome da instância de bloco de função C
fb_name.....nome do bloco de função C de referência

A.26.6 Limites do Ambiente de Trabalho ISaGRAF "LARGE" (WDL)

Há algumas limitações para os objetos utilizados no Ambiente de Trabalho ISaGRAF. Claro que, muitos outros limites práticos são devidos à configuração do computador utilizado (memória disponível e espaço em disco), e as capacidades do sistema ISaGRAF destino (memória disponível, recursos de hardware e de software disponíveis...). As tabelas a seguir fornecem os limites puramente teóricos que não podem ser excedidos.

- ***Para um projeto:***

Objeto	*Máximo*	*Observações*
Programas	255	conjunto dos programas principais, secundários e subprogramas
Níveis hierárquicos	20	

O número de projetos instalados no ambiente de trabalho só está limitado pelo espaço disponível no disco rígido.

- ***Para nomes:***

Nome para:	*Máximo*	*Observações*
Projeto	8 caracteres	
Programa	8 caracteres	
Variável	16 caracteres	+ 60 caracteres para comentários
Rótulo de palavra definida	16 caracteres	
Equivalência definida	255 caracteres	+ 60 caracteres para comentários
Tabela de conversão	16 caracteres	
Lista de variáveis	16 caracteres	
função / bloco f. (lib)	8 caracteres	aplica-se às funções C, blocos de função C ou funções escritas em linguagem IEC
parâmetro de função (lib)	16 caracteres	aplica-se às funções C, blocos de função C ou funções escritas em linguagem IEC
Placa E/S	8 caracteres	
Configuração E/S	8 caracteres	
Parâmetro de placa OEM	16 caracteres	
Função de conversão	8 caracteres	

- ***Edição (para um programa):***

Objeto	*Máximo*	*Observações*
Linhas SFC	600	
Colunas SFC	20	
Etapas SFC	4.095	para todo o projeto, grupamento de etapas, etapas iniciais, etapas de início e de fim
Transições SFC	4.095	para todo o aplicativo
Edição LD/FBD	200 colunas	
	2.000 linhas	este é o tamanho da área de edição em unidades de células.
Edição Quick LD	sem limites	limites impostos p/ capacidade do PC
Etiquetas IL	251	no mesmo programa IL
Edição de texto	40kBytes	

- ***Para o dicionário (para um projeto):***

Objeto	*Máximo*	*Observações*

Variáveis booleanas	65.535	
Variáveis analógicas	65.535	reunindo variáveis inteiras e reais
Temporizadores	65.535	
Variáveis de mensagem	65.535	
Palavras definidas	4.095	na mesma lista (mesmo alcance)
Palavras definidas	255	utilizado no mesmo programa
Tabelas de conversão	127	utilizado no aplicativo
Ponto em uma tabela	32	definido na mesma tabela de conversão

Os limites dados para número de máximo de variáveis booleana, analógica ou de mensagem incluem as variáveis internas, de entrada e de saída. Também inclui todas as variáveis temporárias alocadas pelos compiladores. O número de variáveis editadas juntas no dicionário (mesmo tipo, mesma extensão) não podem exceder 16.000. Dependendo de configuração de PC, o limite pode ser menor que 16.000. O aplicativo não pode ser executa em um ISaGRAF destino com versão V1.21 ou anterior se o número total de variável para um tipo excede o número 4095. O link "Modbus" padrão utilizando endereços de rede não pode ser utilizado se número de variáveis para um tipo excede 4.095.

- ***Conexões de E/S:***

Objeto	*Máximo*	*Observações*
Placas de E/S	256	definidas para o mesmo aplicativo (placas ou equipamentos complexos)

O número de placas de E/S que incluem placas elementares e artigos de equipamentos complexos não pode exceder 256.

Canais de E/S	128	na mesma placa

- ***Para bibliotecas:***

Objeto	*Máximo*	*Observações*
Funções (ling. IEC)	255	instalado junto na biblioteca
Blocos de funções (ling. IEC)	255	instalado junto na biblioteca
Funções C	255	instalado junto na biblioteca
Blocos de funções C	255	instalado junto na biblioteca
Blocos de funções exemplos	4095	para um mesmo tipo de bloco de função no mesmo aplicativo
Parâmetros de função de entrada	31	aplica-se às funções C e às funções escritas em linguagem IEC
Parâmetros de bloco de função	32	livremente distribuídas entre parâmetros de entrada e de saída. Pelo menos um parâmetro de saída é necessário.
Função de conversão	128	instalado junto na biblioteca
Configurações E/S	255	instalado junto na biblioteca
Placas de E/S	255	instalado junto na biblioteca
Equip. complexo E/S.	255	instalado junto na biblioteca
Parâmetros de placa OEM	16	

B. Linguagem de referência

B.1 Arquitetura do projeto

Um projeto ISaGRAF é dividido em várias unidades chamadas **programas**. Os programas do projeto estão ligados entre si em uma arquitetura semelhante a uma árvore. Programas podem ser descritos utilizando-se qualquer das seguintes linguagens gráficas ou literais: **SFC, FC (Fluxograma)**, **FBD**, **LD, ST** ou **IL**

B.1.1 Programas

Um **programa** é uma unidade lógica de programação, que descreve operações entre **varáveis** do processo. Programas descrevem tanto operações **cíclicas** como **seqüenciais**. Programas cíclicos são executados a cada ciclo do sistema destino. A execução de programas seqüenciais segue as regras dinâmicas da linguagem **SFC** ou **FC** .

Os programas estão ligados entre si em uma árvore hierárquica. Programas posicionados no topo da hierarquia são ativados pelo sistema. Subprogramas (nível mais baixo da hierarquia) são ativados pelo seu programa superior. Um programa pode ser descrito com qualquer das linguagens gráficas ou literais a seguir:

Sequential Function Chart , Fluxograma de funções sequencial (SFC) para programação em alto nível
Flow Chart, Fluxograma (FC) para programação em alto nível
Function Block Diagram, Diagrama de Blocos de função (FBD) para operações cíclicas complexas
Ladder Diagram, Diagrama "Ladder" (LD) apenas para operações booleanas
Structured Text, Texto Estruturado (ST) para quaisquer operações cíclicas
Instruction List, Lista de Instruções (IL) para operações de nível mais baixo

O mesmo programa não pode misturar várias linguagens, exceto LD e FBD, que podem ser combinados em um diagrama.

B.1.2 Operações cíclicas e seqüenciais

. A hierarquia de programas está dividida em quatro **seções** ou grupos principais:

Início	programas executados no início de cada ciclo do destino
Sequencial	programas seguindo regras dinâmicas SFC ou FC
Fim	programas executados no final de cada ciclo destino
Funções	conjunto de subprogramas não dedicados

Programas das seções **'Início'** ou **'Fim'** descrevem operações cíclicas e não são dependentes do tempo. Programas da seção **'Sequencial'** descrevem operações seqüenciais, em que a variável tempo sincroniza explicitamente operações básicas. Programas principais da seção **'Início'** são sistematicamente executados ao início de cada ciclo de execução. Programas principais da seção **'Fim'** são sistematicamente executados ao final de cada ciclo de execução. Programas principais da seção **'Sequencial'** são executados de acordo com regras dinâmicas de **SFC** ou **FC**.

Programas da seção **'Funções'** são subprogramas que podem ser chamados por algum outro programa no projeto. Um programa da seção **'Funções'** pode chamar outro programa desta seção.

Programas principais e secundários da seção seqüencial devem ser descritos com linguagem **SFC** ou **FC**. Programas das seções cíclicas (**Início** e **Fim**) não podem ser descritos com linguagem **SFC** ou **FC**. Qualquer programa de qualquer seção pode possuir um ou mais subprogramas. Qualquer programa da seção seqüencial pode possuir um ou mais programas secundários de linguagem **SFC** ou **FC** (de acordo com sua própria linguagem). Subprogramas não podem ser descritos com linguagem **SFC** ou **FC**.

Programas da seção **Início** geralmente são utilizados para descrever operações preliminares em dispositivos de entrada para construir variáveis de alto nível filtradas. Estas variáveis são freqüentemente utilizadas pelos programas da seção **Seqüencial**. Programas da seção **Fim** geralmente são utilizados para descrever operações de segurança nas variáveis em que se operou na seção **Seqüencial**, antes de se enviar valores aos dispositivos de saída.

B.1.3 Programas SFC secundário e FC

Qualquer programa **SFC** da seção seqüencial pode controlar outros programas **SFC**. Estes programas de nível mais baixo são chamados **programas SFC secundários**. Um **programa SFC secundário** é um programa paralelo que pode ser acionado, terminado, congelado ou reiniciado por seu programa principal. O programa principal e o programa secundário devem ser descritos em linguagem **SFC**. Um programa SFC secundário pode ter variáveis locais e definições.

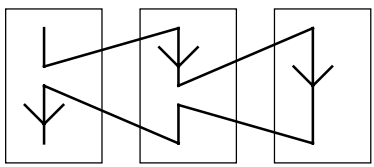
Quando um programa principal aciona um programa **SFC** secundário, ele coloca uma **marca** SFC em cada passo inicial do programa secundário. Este comando é descrito pela declaração **GSTART**. Quando um programa principal termina um programa **SFC** secundário, ele limpa todas as marcas existentes nos **passos** do programa secundário. Um comando assim é descrito pela declaração **GKILL**.

Quando um programa principal congela um programa SFC secundário, ele limpa todas as marcas existentes no programa secundário e mantém suas posições em memória. Um comando assim é descrito pela declaração **GFREEZE**. Quando um programa principal reinicia um programa secundário **SFC** congelado, ele restaura todas as marcas limpas quando o programa secundário foi congelado. Um comando assim é descrito pela declaração **GRST**.

Qualquer programa **FC** da seção seqüencial pode controlar outros programas **FC**. Um programa **FC** principal é bloqueado (espera) durante a execução de um subprograma FC. Não é possível realizar operações simultâneas no programa principal FC e em um de seus subprogramas FC.

B.1.4 Funções e subprogramas

A execução de um subprograma ou uma função é comandada pelo seu programa principal. A execução do programa principal é suspensa até que o subprograma ou a função termine.



principal subprogramas

Qualquer programa de qualquer seção pode ter um ou mais subprogramas. Um subprograma pertence a um único programa principal. Um subprograma pode ter variáveis locais e definições. Qualquer linguagem exceto **SFC** ou **FC** pode ser utilizada para descrever um subprograma. Programas da seção **"Funções"** são subprogramas que podem ser chamados por qualquer outro programa do projeto. Diferente de outros subprogramas, eles não são dedicados a um programa principal. Um programa da seção **"Funções**" pode chamar outro programa desta seção. Uma função pode ser armazenada na Biblioteca.

Advertência: O ISaGRAF não suporta **recursividade** durante chamada de funções. Um erro de execução ocorrerá se um programa da seção **"Funções"** for chamado por si mesmo ou por um de seus subprogramas chamados.

Advertência: Uma função ou subprograma não "armazena" o valor de suas variáveis locais. Uma função ou subprograma não tem instância e portanto não pode chamar blocos de função

A interface de um subprograma deve ser definida explicitamente, com um **tipo** e um **nome único** para cada um dos parâmetros de entrada ou de saída. Para que seja suportada a convenção da linguagem **ST**, o parâmetro de saída deve ter o mesmo nome do subprograma.

A tabela a seguir mostra como determinar o valor do parâmetro de saída no corpo de um subprograma, em diversas linguagens:

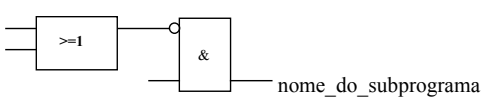
ST: atribua o parâmetro de saída usando o seu nome (o mesmo nome do subprograma):

```
nome_do_subprograma := <expressão>;
```

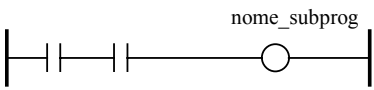
IL: o valor do resultado atual (registro IL) ao final da seqüência é armazenado no parâmetro de saída:

```
LD    10
ADD   20   (*valor do parâmetro de saída = 30 *)
```

FBD: atribua o parâmetro de saída utilizando seu nome:

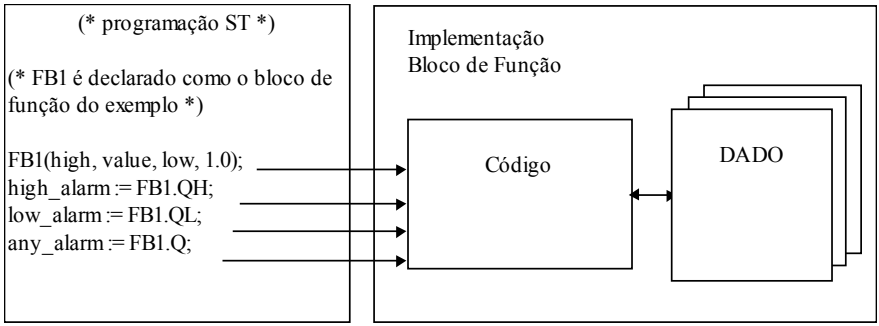


LD: utilize um símbolo de bobina com o nome do parâmetro de saída:



B.1.5 Blocos de funções

Blocos de função podem utilizar as linguagens: LD, FBD, ST ou IL. Blocos de função têm instâncias. Isto significa que variáveis locais de um bloco de função são copiadas para cada instância. Quando se chama um bloco dentro de um programa, na realidade é a instância do bloco que está sendo chamada: o código chamado é sempre o mesmo mas os dados utilizados são os que foram carregados para aquela instância. Os valores das variáveis da instância são armazenados de um ciclo para o outro.



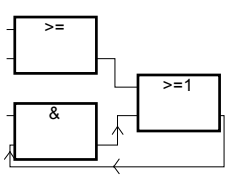
Advertências:
- Um bloco de função escrito em uma das linguagens IEC não pode chamar outros blocos de função: o mecanismo de instanciamento é capaz de acessar somente as variáveis locais do próprio bloco. Esta é a lista de blocos de função padrão que não se pode utilizar dentro de um bloco de função IEC:
SR, RS, R_Trig, F_Trig, SEMA, CTU, CTD, CTUD, TON, TOF, TP, CMP, StackInt, AVERAGE, HYSTER, LIM_ALRM, INTEGRAL, DERIVATE, BLINK, SIG_GEN

- Pela mesmo motivo, não se pode utilizar contatos normalmente abertos ou normalmente fechados, bobinas ou Liga e Desliga.

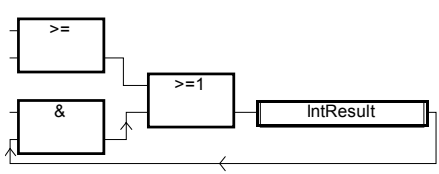
- As funções TSTART e TSTOP para disparar e parar temporizadores não podem ser utilizadas em um bloco de função para destinos 3.0x. Isto é possível para destinos a partir da versão 3.20.

- Quando for necessário um loop no seu bloco de função, você deve utilizar uma variável local antes de fechar o loop. Veja o exemplo a seguir:

Isto não irá funcionar:



Este está correto:



B.1.6 Linguagens

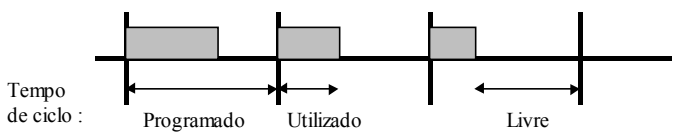
Um programa pode ser descrito em qualquer uma das linguagens literais e gráficas a seguir:

Sequential Function Chart , Fluxograma de funções sequencial (SFC) para programação de alto nível
Flow Chart, Fluxograma (FC) para programação de alto nível
Function Block Diagram, Diagrama de Blocos de função (FBD) para operações cíclicas complexas
Ladder Diagram, Diagrama "Ladder" (LD) apenas para operações booleanas
Structured Text, Texto Estruturado (ST) para quaisquer operações cíclicas
Instruction List, Lista de Instruções (IL) para operações de nível mais baixo

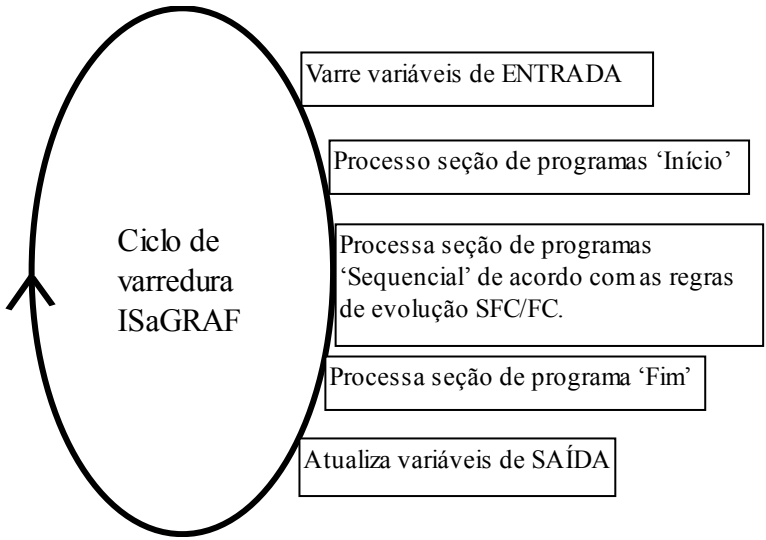
O mesmo programa não pode misturar várias linguagens. A linguagem utilizada para descrever um programa é escolhida quando o programa é criado, e não pode ser alterada mais tarde. A exceção é a possibilidade de combinar FBD e LD em um programa.

B.1.7 Regras de execução

O ISaGRAF é um sistema **síncrono**. Todas as operações são acionadas por um relógio. A duração de um ciclo do relógio é chamada de tempo de ciclo:



As operações básicas processadas durante um ciclo do destino são:



Este sistema permite:

- garantir que uma variável de entrada mantenha o mesmo valor durante um ciclo,
- garantir que um dispositivo de saída não é atualizado mais de uma vez em um ciclo,
- trabalhar com segurança com a mesma variável global a partir de programas diferentes,
- estimar e controlar o tempo de resposta do aplicativo completo.

B.2 Objetos comuns

Serão descritos a seguir as características principais e os **objetos** comuns do banco de dados de programação ISaGRAF. Tais objetos podem ser utilizados em qualquer programa escrito em qualquer uma das linguagens: **SFC**, **FC**, **FBD**, **LD**, **ST** ou **IL**.

B.2.1 Tipos básicos

Qualquer constante, expressão ou variável utilizada em um programa (escrito em qualquer linguagem) deve ser caracterizada por um tipo . Deve-se manter uma coerência de tipos em operações gráficas e declarações literais. Estes são os tipos básicos disponíveis para programação:

BOOLEANO:	valor lógico (verdadeiro ou falso)
ANALÓGICO:	valor contínuo inteiro ou real (ponto flutuante)
TEMPORIZADOR:	valor de tempo
MENSAGEM:	cadeia de caracteres (“string”)

Observação: Os Temporizadores contêm valores menores que um dia e não podem ser utilizados para armazenar datas.

B.2.2 Expressões constantes

Expressões constantes são relativas a um tipo. A mesma notação não pode ser utilizada para representar expressões constantes de tipos diferentes.

B.2.2.1 Expressões constantes booleanas

Existem apenas duas expressões constantes booleanas:

TRUE	é equivalente ao valor inteiro 1
FALSE	é equivalente ao valor inteiro 0

As palavras-chave “True" e "False" não fazem distinção entre maiúsculas e minúsculas.

B.2.2.2 Expressões constantes analógicas inteiras

Expressões constantes inteiras representam valores inteiros longos com sinal (32 bit): de **-2.147.483.647** até **+2.147.483.647**. Constantes analógicas inteiras podem ser expressas em uma das **bases** a seguir, e devem iniciar com um **prefixo** que identifica a base utilizada:

Base	Prefixo	Exemplo
DECIMAL	**(nenhum)**	**-908**
HEXADECIMAL	**"16#"**	**16#1A2B3C4D**
OCTAL	**"8#"**	**8#1756402**
BINÁRIA	**"2#"**	**2#1101_0001_0101_1101**

O caractere sublinhado ('_') pode ser utilizado para separar grupos de algarismos. Ele não tem nenhum significado especial, e é utilizado apenas para facilitar a visualização do valor da expressão.

B.2.2.3 Expressões constantes analógicas reais

Expressões constantes analógicas reais podem ser escritas tanto em notação **decimal** como em notação **científica**. O **ponto decimal** ('**.**') separa as partes inteira e decimal. O ponto decimal deve ser utilizado para diferenciar uma constante real de uma inteira. A notação científica utiliza a letra '**E**' ou '**F**' para separar a **mantissa** do **expoente**. O expoente de uma expressão real em notação científica deve ser um valor inteiro com sinal de **-37** a **+37**. A seguir serão apresentados alguns exemplos de expressões constantes analógicas reais:

3.14159	**-1.0E+12**
+1.0	**1.0F-15**
-789.56	**+1.0E-37**

A expressão "**123**" não representa uma expressão constante real. Sua representação correta é "**123.0**".

B.2.2.4 Expressões constantes do tipo temporizador

Expressões constantes do tipo temporizador representam valores de tempo de **0 segundo** a **23h59m59s999ms**. A menor unidade permitida é o milisegundo. As unidades padrão de tempo utilizadas em expressões constantes são:

Hora	A letra "**h**" deve vir após a quantidade de horas
Minuto	A letra "**m**" deve vir após a quantidade de minutos
Segundo	A letra "**s**" deve vir após a quantidade de segundos
Milisegundo	As letras "**ms**" devem vir após a quantidade de milisegundos

A expressão constante do tipo temporizador deve iniciar com o prefixo "**T#**" ou "**TIME#**". Prefixos e letras que denotam unidades não fazem distinção entre maiúsculas e minúsculas. Podem-se omitir as unidades. Estes são exemplos de expressões constantes do tipo temporizador:

T#1H450MS	1 hora, 450 millisegundos
time#1H3M	1 hora, 3 minutos

A expressão "0" não representa um valor de tempo, e sim uma constante analógica.

B.2.2.5 Expressões constantes do tipo mensagem

Expressões constantes do tipo "string" ou mensagem representam cadeias de caracteres ("strings"). Os caracteres devem ser precedidos de um sinal "**'**" e seguidos por um apóstrofo. Por exemplo:

' ISTO É UMA MENSAGEM '

Advertência: O caractere apóstrofo "**'**" não pode ser utilizado dentro de uma expressão constante de cadeia de caracteres ("string"). Uma cadeia de caracteres ("string") deve estar contida em uma linha do código fonte do programa. Seu tamanho não pode exceder 255 caracteres, incluindo espaços.

"strings" vazias são representadas por dois apóstrofos, sem nenhum espaço ou caractere de tabulação entre eles:

'' (*isto é uma "string" vazia *)

O caractere especial cifrão ('**$**'), seguido por outros caracteres especiais, pode ser utilizado em uma "string" para representar um caractere não impresso:

Seqüência	Significado	ASCII (hexa)	Exemplo
$$	'$' caractere	16#24	'Eu paguei $$5 por isto'
$'	apóstrofo	16#27	'Digite $'S$' fpara SIM'
$L	avanço de linha	16#0ª	'próxima $L linha'
$R	retorno para o início da linha	16#0d	' llo $R He'
$N	nova linha	16#0d0a	'Isto é uma linha$N'
$P	nova página	16#0c	'última linha $P primeira linha'
$T	tabulação	16#09	'nome$Ttamanho$Tdata'
$hh (*)	qualquer caractere	16#hh	'ABCD = $41$42$43$44'

(*) "**hh**" é o valor hexadecimal do código ASCII para o caractere.

B.2.3 Variáveis

As variáveis podem ser **LOCAIS** a um programa, ou **GLOBAIS**. Variáveis locais podem ser utilizadas por um único programa. Variáveis globais podem ser utilizadas em qualquer programa do projeto. Os nomes de variáveis devem seguir as seguintes regras:

O nome não pode exceder **16** caracteres
O primeiro caractere deve ser uma **letra**
Os caracteres seguintes podem ser **letras**, **números** ou o caractere sublinhado

B.2.3.1 Palavras reservadas

Uma lista das palavras reservadas é apresentada a seguir. Tais identificadores não podem ser utilizados para dar nome a um programa, variável, função "C" ou bloco de função:

A	ANA, ABS, ACOS, ADD, ANA, AND, AND_MASK, ANDN, ARRAY, ASIN, AT, ATAN,
B	BCD_TO_BOOL, BCD_TO_INT, BCD_TO_REAL, BCD_TO_STRING, BCD_TO_TIME, BOO, BOOL, BOOL_TO_BCD, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_STRING, BOOL_TO_TIME, BY, BYTE,
C	CAL, CALC, CALCN, CALN, CALNC, CASE, CONCAT, CONSTANT, COS,
D	DATE, DATE_AND_TIME, DELETE, DINT, DIV, DO, DT, DWORD,
E	ELSE, ELSIF, EN, END_CASE, END_FOR, END_FUNCTION, END_IF, END_PROGRAM, END_REPEAT, END_RESSOURCE, END_STRUCT, END_TYPE, END_VAR, END_WHILE, ENO, EQ, EXIT, EXP, EXPT,
F	FALSE, FEDGE, FIND, FOR, FUNCTION,
G	GE, GFREEZE, GKILL, GRST, GSTART, GSTATUS, GT,

I	IF, INSERT, INT, INT_TO_BCD, INT_TO_BOOL, INT_TO_REAL, INT_TO_STRING, INT_TO_TIME,
J	JMP, JMPC, JMPCN, JMPN, JMPNC,
L	LD, LDN, LE, LEFT, LEN, LIMIT, LINT, LN, LOG, LREAL, LT, LWORD,
M	MAX, MID, MIN, MOD, MOVE, MSG, MUL, MUX,
N	NE, NOT,
O	OF, ON, OPERATE, OR, OR_MASK, ORN,
P	PROGRAM
R	R, REDGE, READ_ONLY, READ_WRITE, REAL, REAL_TO_BCD, REAL_TO_BOOL, REAL_TO_INT, REAL_TO_STRING, REAL_TO_TIME, REDGE, REPEAT, REPLACE, RESSOURCE, RET, RETAIN, RETC, RETCN, RETN, RETNC, RETURN, RIGHT, ROL, ROR,
S	S, SEL, SHL, SHR, SIN, SINT, SQRT, ST, STN, STRING, STRING_TO_BCD, STRING_TO_BOOL, STRING_TO_INT, STRING_TO_REAL, STRING_TO_TIME, STRUCT, SUB, SYS_ERR_READ, SYS_ERR_TEST, SYS_INITALL, SYS_INITANA, SYS_INITBOO, SYS_INITTMR, SYS_RESTALL, SYS_RESTANA, SYS_RESTBOO, SYS_RESTTMR, SYS_SAVALL, SYS_SAVANA, SYS_SAVBOO, SYS_SAVTMR, SYS_TALLOWED, SYS_TCURRENT, SYS_TMAXIMUM, SYS_TOVERFLOW, SYS_TRESET, SYS_TWRITE, SYSTEM,
T	TAN, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_BCD, TIME_TO_BOOL, TIME_TO_INT, TIME_TO_REAL, TIME_TO_STRING, TMR, TO, TOD, TRUE, TSTART, TSTOP, TYPE,
U	UDINT, UINT, ULINT, UNTIL, USINT,
V	VAR, VAR_ACCESS, VAR_EXTERNAL, VAR_GLOBAL, VAR_IN_OUT, VAR_INPUT, ,VAR_OUTPUT,
W	WHILE, WITH, WORD,
X	XOR, XOR_MASK, XORN

Todas as palavras-chave iniciadas por um caractere sublinhado ('_') são palavras-chave internas e não podem ser utilizadas em instruções de textos.

B.2.3.2 Variáveis representadas diretamente

O ISaGRAF permite o uso de **variáveis representadas diretamente** no código fonte dos programa para representar um canal livre. Canais livres são aqueles que não estão ligados a uma variável declarada como E/S. O identificador de uma variável representada diretamente sempre começa com o caractere "**%**".

A seguir são apresentadas as convenções de nome de uma variável representada diretamente para um canal ou uma placa. "***s***" é o número do slot da placa. "***c***" é o número do canal.

%IX*s.c* canal livre de uma placa de entrada booleana
%ID*s.c* canal livre de uma placa de entrada inteira
%IS*s.c* canal livre de uma placa de entrada do tipo mensagem
%QX*s.c* canal livre de uma placa de saída booleana
%QD*s.c* canal livre de uma placa de saída inteira
%QS*s.c* canal livre de uma placa de saída do tipo mensagem

A seguir são apresentadas as convenções de nome de um uma variável representada diretamente para um canal de um equipamento complexo. "***s***" é o número do slot do equipamento. "***b***" é o índice da placa dentro do equipamento complexo. "***c***" é o número do canal.

%IX*s.b.c* canal livre de uma placa de entrada booleana
%ID*s.b.c* canal livre de uma placa de entrada inteira
%IS*s.b.c* canal livre de uma placa de entrada do tipo mensagem
%QX*s.b.c* canal livre de uma placa de saída booleana
%QD*s.b.c* canal livre de uma placa de saída inteira
%QS*s.b.c* canal livre de uma placa de saída do tipo mensagem

Exemplos:

%QX1.6 Sexto canal da placa número 1 (saída booleana)
%ID2.1.7 Sétimo canal da placa número 1 no equipamento número 2 (entrada inteira)

Uma variável representada diretamente não pode ser do tipo **real**.

B.2.3.3 Variáveis booleanas

Booleano significa **lógico**. Tais variáveis podem ter um dos dois valores booleanos: **TRUE** ou **FALSE**. Variáveis booleanas geralmente são utilizadas em expressões booleanas. Variáveis booleanas podem ter um dos seguintes **atributos**:

Interna: variável de memória atualizada pelo programa
Constante: variável de somente leitura com valor inicial
Entrada: variável conectada a um dispositivo de entrada (atualizada pelo sistema)
Saída: variável conectada a um dispositivo de saída

Advertência: Quando se declara uma variável booleana, cadeias de caracteres (“strings”) podem ser definidas para substituir os valores 'true' e 'false' durante a depuração. Estas “strings” não podem ser utilizadas em programas a menos que tenham sido inseridas como **‘definições’** para a linguagem.

B.2.3.4 Variáveis analógicas

Analógica significa **contínuo**. Tais variáveis possuem valores inteiros ou reais (flutuantes) com sinal. Os formatos disponíveis para uma variável analógica são:

Inteiro Inteiro de 32 bits com sinal: de **–2.147.483.647** a **+2.147.483.647**
Real valor de ponto flutuante padrão IEEE 32 (precisão simples)
1 bit de sinal + 23 bits de mantissa + 8 bits de expoente

O valor do expoente deve ser um inteiro entre **-37** e **+37**, inclusive. Variáveis analógicas podem ter um dos seguintes **atributos**:

Interna: variável de memória atualizada pelo programa
Constante: variável de somente leitura com valor inicial
Entrada: variável conectada a um dispositivo de entrada (atualizada pelo sistema)
Saída: variável conectada a um dispositivo de saída

Observação: Quando uma variável real é conectada a um dispositivo de E/S, o dispositivo de E/S correspondente opera com o valor inteiro equivalente.

Advertência: Variáveis analógicas ou expressões constantes inteiras e reais não podem ser misturadas na mesma expressão analógica.

B.2.3.5 Variáveis do tipo temporizador

As variáveis do tipo temporizador contêm valores de tempo e são utilizadas em expressões de tempo. Um valor de temporizador não pode exceder **23h59m59s99** e não pode ser negativo. Variáveis do tipo temporizador são armazenadas em palavras de 32 bits. A representação interna é um número positivo de milisegundos.
As variáveis do tipo temporizador podem ter um dos seguintes **atributos**:

Interno	variável de memória gerenciada pelo programa, atualizada pelo sistema ISaGRAF
Constante:	variável de memória ROM com valor inicial

Advertência: Variáveis do tipo temporizador não podem ter atributos de ENTRADA ou SAÍDA.

As variáveis do tipo temporizador podem ser automaticamente atualizadas pelo sistema ISaGRAF. Quando um temporizador está **ativo**, seu valor é automaticamente incrementado de acordo com o relógio de tempo real do sistema destino. As seguintes declarações da linguagem **ST** podem ser utilizadas para controlar um temporizador:

TSTART	inicia a atualização automática de um temporizador
TSTOP	pára a atualização automática de um temporizador

B.2.3.6 Variáveis do tipo mensagem

As variáveis do tipo mensagem contêm cadeias de caracteres (“strings”). O tamanho da “string” pode mudar durante as operações de processamento. O tamanho de uma variável do tipo mensagem não pode exceder a capacidade (tamanho máximo) especificada na sua declaração. A capacidade de uma mensagem é limitada a 255 caracteres. Variáveis do tipo mensagem podem ter um dos seguintes **atributos**:

Interna:	variável de memória atualizada pelo programa
Constante:	variável de memória ROM com valor inicial
Entrada:	variável conectada a um dispositivo de entrada (atualizada pelo sistema)
Saída:	variável conectada a um dispositivo de saída

Variáveis do tipo mensagem podem conter qualquer caractere da tabela padrão ASCII (código ASCII de **0** a **255**). O caractere nulo pode existir em uma cadeia de caracteres (“string”). Algumas funções “C” da biblioteca padrão ISaGRAF não operam corretamente com “strings” que contenham caracteres nulos (**0**).

B.2.4 Comentários

Comentários podem ser inseridos livremente em linguagens literais como **ST** e **IL**. Um comentário deve iniciar com os caracteres especiais "**(***" e terminar com os caracteres "***)**". Comentários podem ser inseridos em qualquer lugar dentro de um programa **ST**, e podem ser escritos em mais de uma linha.

Exemplos de comentários:

```
contador := valor;   (* atribui ao contador principal *)
(* isto é um comentário
em duas linhas *)
c := contador (* você pode colocar comentários em qualquer lugar *) + valor_base + 1;
```

Comentários intercalados não podem ser utilizados. Isto significa que a seqüência de caracteres "**(***" não pode ocorrer dentro de um comentário.

Advertência: A linguagem IL somente aceita comentários como o último componente de uma linha de instrução.

B.2.5 Definições

O ISaGRAF permite a redefinição de expressões constantes, expressões booleanas "true" e "false", palavras-chave ou expressões complexas em **ST**. Para isto, um nome **identificador** deve ser associado a uma expressão correspondente. Por exemplo:

YES	é	**TRUE**
PI	é	**3.14159**
OK	é	**(auto_mode AND NOT (alarm))**

Quando uma equivalência como esta é definida, seu **identificador** pode ser utilizado em qualquer ponto de um programa **ST** para substituir a expressão representada. Este é um exemplo de programação **ST** utilizando definições:

```
If OK Then
  angle := PI / 2.0;
  isdone := YES;
End_if;
```

As definições podem ser **LOCAIS** a um programa, **GLOBAIS**, ou **COMUNS**.
As definições locais podem ser utilizadas por um único programa.
As definições globais podem ser utilizadas por qualquer programa de um projeto.
As definições comuns podem ser utilizadas em qualquer programa de qualquer projeto.
Note que as definições comuns podem ser armazenadas separadamente com o Gerenciador de Diretório.

Advertência: Se o mesmo identificador for definido duas vezes com equivalência diferente em **ST**, a última expressão definida será utilizada. Por exemplo:

Definição: **OPEN** é **FALSE**

OPEN é **TRUE**

significa: **OPEN** é **TRUE**

Os nomes das definições devem seguir as seguintes regras:
- o nome não pode exceder **16** caracteres
- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes podem ser **letras**, **números** ou o caractere sublinhado

Advertência: Uma definição não pode utilizar uma outra definição dentro de seu significado. Por exemplo, não é possível fazer:

PI é **3.14159**
~~**PI2** é **PI*2**~~

escreva a definição completa utilizado constantes ou variáveis e operações:

PI2 é **6.28318**

B.3 Linguagem SFC

Sequential Function Chart , Fluxograma de funções Sequencial (SFC) é uma linguagem **gráfica** utilizada para descrever **operações seqüenciais** . O processo é representado como um conjunto de **etapas** bem definidas, ligadas por **transições**. Uma **condição booleana** é associada a cada transição. **Ações** dentro de cada etapa são detalhadas com o uso de outras linguagens (**ST**, **IL**, **LD** e **FDB**).

B.3.1 Formato principal de um gráfico SFC

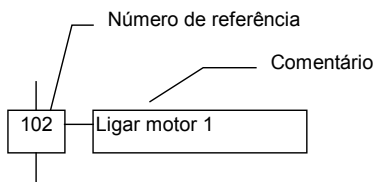
Um programa SFC é um conjunto gráfico de **etapas** e **transições**, conectadas por **ligações orientadas**. Ligações de conexões múltiplas são utilizadas para representar divergências e convergências. Algumas partes do programa completo podem ser separadas e representadas no gráfico principal por um único símbolo, chamado de **macro etapa**. As **regras gráficas** básicas de um SFC são:
- Uma etapa não pode ser seguida por outra etapa
- Uma transição não pode ser seguida por outra transição

B.3.2 Componentes SFC básicos

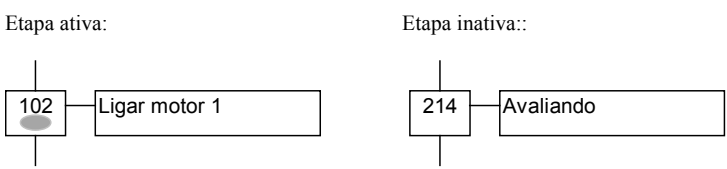
Os componentes básicos (símbolos gráficos) da linguagem SFC são: etapas e etapas iniciais, transições, ligações orientadas e desvios para uma etapa.

B.3.2.1 Etapas e etapas iniciais

Uma etapa é representada por um **quadrado**. Cada etapa é **associada**. a um número, escrito no quadrado símbolo da etapa. Uma descrição geral da etapa é apresentada em um retângulo ligado ao símbolo da etapa. Esta descrição é um **comentário livre** (não faz parte da linguagem de programação). A informação acima é chamada de **Nível 1** da etapa:

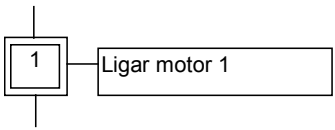


Durante a execução, uma **marca** indica que a etapa está **ativa**:



A **situação inicial** de um programa SFC é descrita pelas **etapas iniciais** . Uma etapa inicial tem um símbolo gráfico com **borda dupla**. Uma marca é automaticamente colocada em cada etapa inicial quando o programa é iniciado.

Etapa inicial:

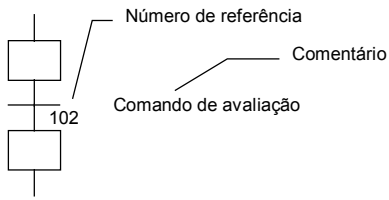


Um programa SFC deve conter **pelo menos uma** etapa inicial.

Estes são os atributos de uma etapa. Tais campos podem ser usados em qualquer outras linguagens:
GSnnn.x .....Estado de atividade de uma etapa (valor booleano)
GSnnn.t .....Duração da atividade da etapa (valor de tempo)
(onde **nnn** é o número de referência da etapa)

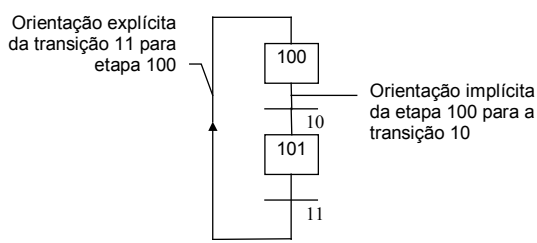
B.3.2.2 Transições

Transições são representadas por uma pequena barra horizontal que cruza a conexão entre duas etapas. Cada transição é **referenciada** por um número que aparece ao lado do símbolo da transição. Uma descrição geral da transição aparece no lado direito do símbolo da transição. Esta descrição é um **comentário livre** (não faz parte da linguagem de programação. A informação acima é chamada de **Nível 1** da transição:



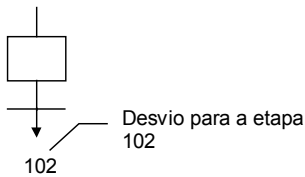
B.3.2.3 Ligações (“links”) orientadas

Linhas simples são utilizadas para ligar etapas e transições. Estas são ligações orientadas. Quando a orientação não for explícita, a ligação é orientada de cima para baixo.

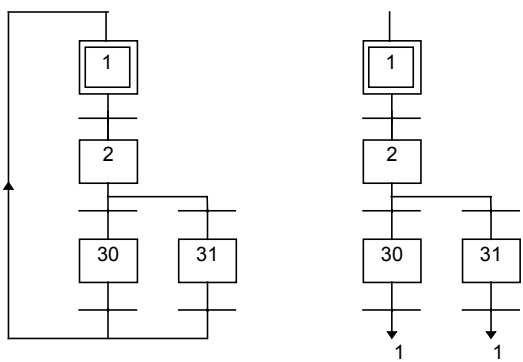


B.3.2.4 Desvio para uma etapa

Símbolos de desvio podem ser utilizados para indicar uma ligação de conexão de uma transição para uma etapa, sem ter que desenhar a linha de conexão. O símbolo de desvio deve ser associado ao número da etapa de destino:



Um símbolo de desvio não pode ser utilizado para representar uma ligação de um passo para uma transição. Exemplo de desvios - os gráficos a seguir são equivalentes:

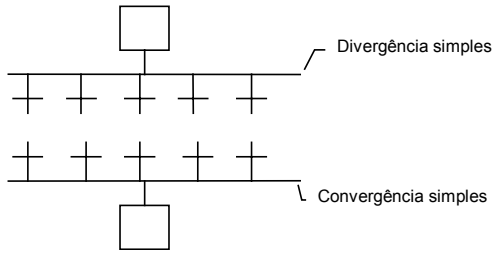


B.3.3 Divergências e convergências

Divergências são **ligações de conexão múltipla** de um símbolo SFC (etapa ou transição) para muitos outros símbolos SFC. Convergências são ligações de conexão múltipla de mais de um símbolo SFC para um outro símbolo individual. Divergências e convergências podem ser simples ou duplas.

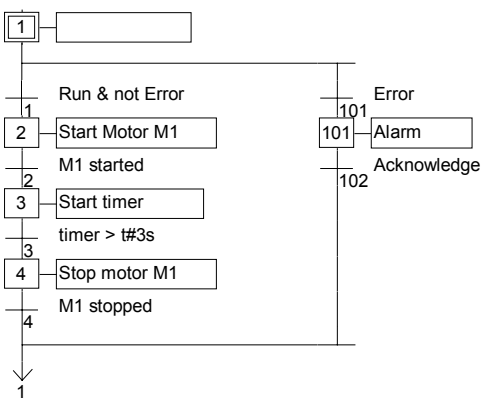
B.3.3.1 Divergências simples

Uma divergência simples é uma ligação múltipla de uma etapa para muitas transições. Ela permite que a marca de atividade passe por uma das ramificações. Uma convergência simples geralmente é utilizada para agrupar ramificações SFC que foram iniciadas em uma divergência simples. Divergências e convergências simples são representadas por linhas horizontais simples.



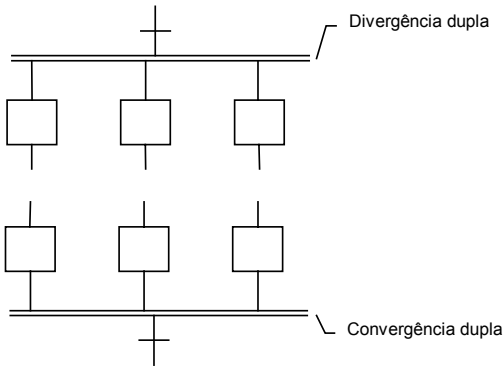
Advertência: As condições associadas a diferentes transições no início de uma divergência simples **não são implicitamente exclusivas**. A exclusividade deve ser detalhada explicitamente nas condições das transições para assegurar que somente uma marca de atividade prossiga em uma ramificação da divergência durante a execução. A seguir é apresentado um exemplo de divergência e convergência simples:

(* Programa SFC com divergência e convergência simples *)



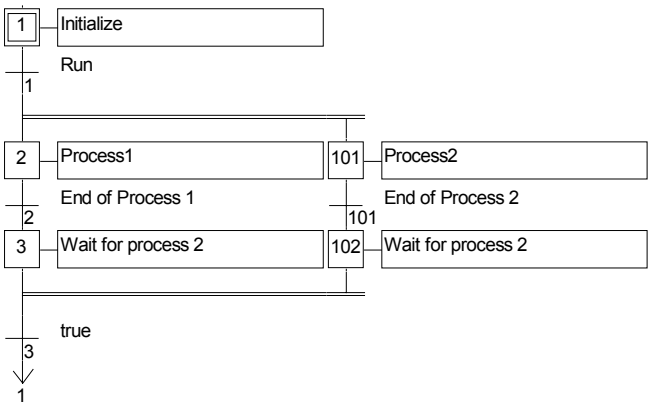
B.3.3.2 Divergências duplas

Uma divergência dupla é uma ligação múltipla de uma transição para muitas etapas. Ela corresponde a operações paralelas do processo. Uma convergência dupla é uma ligação múltipla de muitas etapas para uma mesma transição. Convergências duplas geralmente são utilizadas para agrupar ramificações SFC iniciadas com uma divergência dupla. Divergências e convergências duplas são representadas por linhas horizontais duplas.



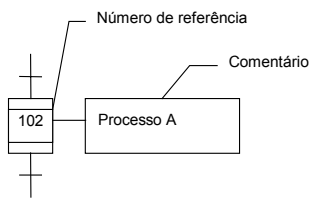
Exemplo de divergência e convergência duplas:

(* Programa SFC com divergência e convergência duplas *)



B.3.4 Macro etapas

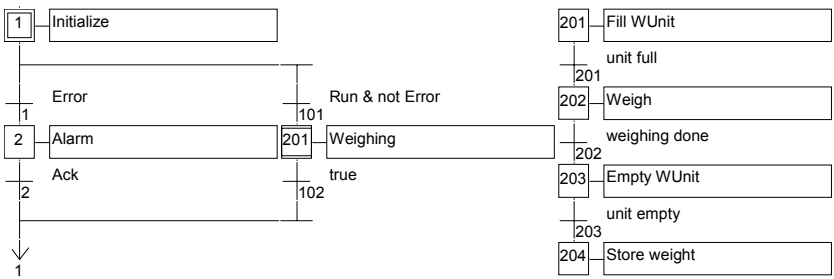
Uma macro etapa é uma representação individual de um grupo definido de etapas e transições. O corpo da macro etapa é descrito separadamente, em algum outro lugar no mesmo programa SFC. Ele aparece como um símbolo individual no gráfico SFC principal. Este é o símbolo utilizado pela macro etapa:



O número de referência apresentado no símbolo da macro etapa corresponde ao número de referência da primeira etapa no corpo da macro etapa. O corpo da macro etapa deve começar com uma **etapa inicial** e terminar com uma **etapa final**. O gráfico deve estar contido em si próprio. Uma etapa inicial pode ter ligação para cima (sem transição para trás). Uma etapa final não pode ter ligação para baixo (sem transição para frente). Um símbolo de macro etapa pode ser inserido no corpo de outra macro etapa.

Advertência: Como uma macro etapa é um conjunto **único** de etapas e transições, a mesma macro etapa não pode ser utilizada mais de uma vez em um programa SFC.

Exemplo de macro etapa:
(* Programa SFC com macro etapa *)
(* Gráfico principal *) **(* Corpo da macro etapa *)**



B.3.5 Ações dentro das etapas

O **nível 2** de uma etapa SFC é a descrição detalhada das **ações** executadas **enquanto a etapa estiver ativa**. Esta descrição é efetuada utilizando-se as **características literais SFC**, e outras linguagens como Texto Estruturado (**ST**). Os tipos básicos de ações são:
- Ações booleanas
- Ações de pulso programadas no ST
- Ações não armazenadas programadas no ST
- Ações SFC

Várias ações (de tipos iguais ou diferentes) podem ser descritas na mesma etapa. As características especiais que permitem o uso de quaisquer das outras linguagens são:
- Chamando subprogramas
- Convenção IL

B.3.5.1 Ações booleanas

Ações booleanas atribuem a uma variável booleana o estado de atividade da etapa. A variável booleana pode ser uma variável de saída ou interna. Ela é atribuída cada vez que a etapa entra ou sai de atividade. Esta é a sintaxe de ações booleanas básicas:

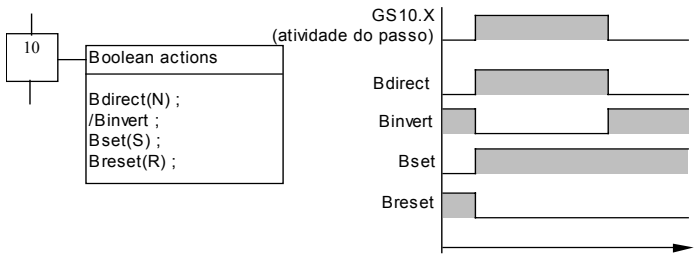
<variável_booleana> (N) ; atribui o estado de atividade da etapa para a variável

<variável_booleana > ;	mesmo efeito (atributo N é opcional)
/ <variável_booleana> ;	atribui a negação do estado de atividade da etapa à variável

Outras características estão disponíveis para ligar ou desligar uma variável booleana, quando a etapa torna-se ativa. Esta é a sintaxe:

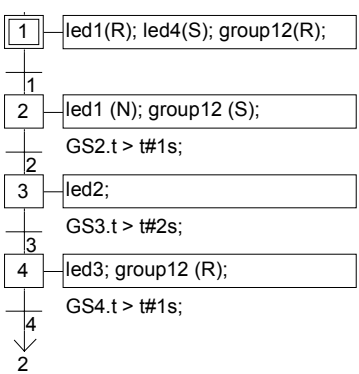
<variável_booleana> (S) ;	Atribui TRUE à variável quando o sinal de atividade da etapa torna-se TRUE.
<variável_booleana> (R) ;	Atribui FALSE à variável quando o sinal de atividade da etapa torna-se FALSE

A variável booleana deve ser de SAÍDA ou INTERNA. O programa SFC a seguir tem este comportamento:



Exemplo de ações booleanas:

(* programa SFC usando ações BOOLEANAS *)

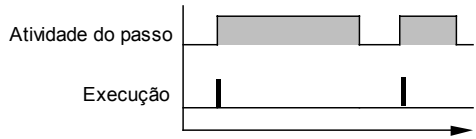


B.3.5.2 Ações de pulso

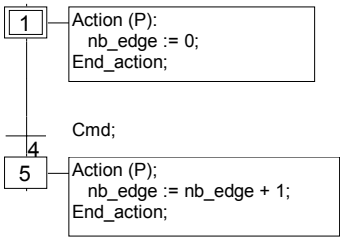
Uma ação de pulso é uma lista de instruções ST ou IL, que são executadas apenas **uma vez** quando da **ativação** da etapa. As instruções são escritas de acordo com a sintaxe SFC a seguir:

ACTION (P) :
(*declarações ST*)
END_ACTION ;

Os resultados de uma ação de pulso são apresentados a seguir:



Exemplo de ação de pulso:

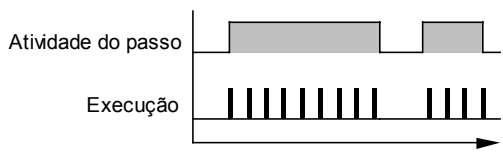


B.3.5.3 Ações não armazenadas

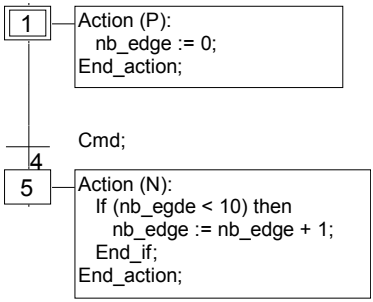
Uma ação não armazenada (normal) é uma lista de instruções ST ou IL executadas **a cada ciclo** durante todo o período em que a etapa estiver **ativa**. As instruções são escritas de acordo com a sintaxe SFC a seguir:

ACTION (N) :
(*declarações ST*)
END_ACTION ;

Os resultados de uma ação não armazenada são apresentados a seguir:



Exemplo de ação não armazenada:



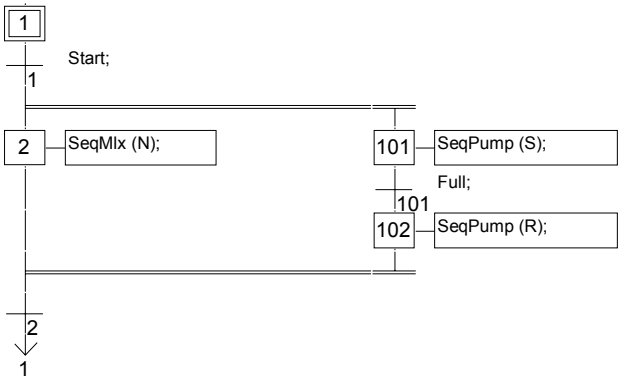
B.3.5.4 Ações SFC

Uma ação SFC é uma seqüência SFC secundária, iniciada ou terminada de acordo com a mudança do estado de ativação da etapa. Uma ação SFC pode ter os qualificadores **N** (não armazenada), **S** (Liga), ou **R** (Desliga). Esta é a sintaxe das ações SFC básicas:

<prog_secund> (N);	inicia a seqüência secundária quando a etapa torna-se ativa, e termina a seqüência secundária quando a etapa torna-se inativa
<prog_secund> ;	mesmo efeito (atributo N é opcional)
<prog_secund> (S);	inicia a seqüência secundária quando a etapa torna-se ativa, nada é feito quando a etapa torna-se inativa
<prog_secund> (R);	termina a seqüência secundária quando a etapa torna-se ativa, nada é feito quando a etapa torna-se inativa

A seqüência SFC especificada como uma ação deve ser um **programa SFC secundário** do programa que está sendo editado no momento. Note que utilizar os qualificadores **S** (Liga) ou **R** (Desliga) para uma ação SFC tem exatamente o mesmo efeito que as declarações **GSTART** e **GKILL** programadas em uma ação de pulso **ST**.
A seguir é apresentado um exemplo de uma ação SFC. O programa principal SFC é chamado de **Father** (principal). Ele tem dois SFC secundários, chamados **SeqMlx** e **SeqPump**. A programação SFC do programa principal é:

(* programa SFC usando ações SFC *)



B.3.5.5 Chamando funções e blocos de função de uma ação

Subprogramas, funções ou blocos de função (escritos em ST, IL, LD ou FBD) ou funções "C" ou blocos de função "C" podem ser chamados diretamente de um bloco de ação SFC, baseando-se na seguinte sintaxe:

Para subprogramas, funções e funções "C":

```
ACTION (P) :
        result := sub_program ( ) ;
END_ACTION;
```

ou

```
ACTION (N) :
        result := sub_program ( ) ;
END_ACTION;
```

Para blocos de função em "C" ou em ST, IL, LD, FBD:

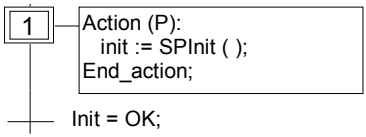
```
ACTION (P) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

ou

```
ACTION (N) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

A sintaxe detalhada pode ser encontrada na seção sobre a linguagem ST.
Exemplo da chamada de um subprograma em blocos de ação:

(* programa SFC com um subprograma chamado por um bloco de ação *)



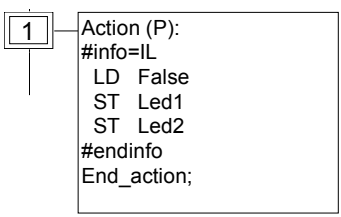
B.3.5.6 Convenção IL

Programação em **Instruction List, Lista de Instruções** (IL) pode ser realizada diretamente em um bloco de ação SFC , baseando-se na seguinte sintaxe:

```
ACTION (P) :            (* ou N *)
#info=IL
   <instrução>
   <instrução>
   ....
#endinfo
END_ACTION;
```

As palavras-chave especiais "#info=IL" e "#endinfo" devem ser inseridas exatamente desta forma, e **fazem distinção entre maiúsculas e minúsculas**. Caracteres de tabulação ou espaços não podem ser inseridos antes, no meio ou depois das palavras-chave. A seguir é apresentado um exemplo de um programa IL em um bloco de ação:

(* programa SFC com uma sequência IL em um bloco de ação*)



B.3.6 Condições associadas a transições

A cada transição, associa-se uma expressão que condiciona a transposição da transição . A condição normalmente é escrita em linguagem ST ou LD (editor Quick LD). Este é o **Nível 2** da transição. Entretanto, outras estruturas podem ser utilizadas:

- Convenção ST
- Convenção LD
- Convenção IL
- Chamando funções de uma transição

Advertência: Quando não houver nenhuma expressão associada à transição, a condição padrão é **TRUE**.

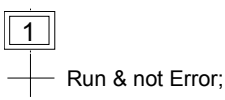
B.3.6.1 Convenção ST

A linguagem **Structured Text, Texto Estruturado** (ST) pode ser utilizada para descrever a **condição** associada a uma transição. A expressão completa deve ser do tipo **booleano** e deve ser terminada por um **ponto e vírgula**, de acordo com a sintaxe a seguir:

< expressão_booleana > ;

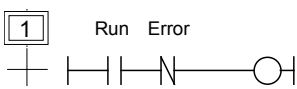
A expressão pode ser uma constante TRUE ou FALSE, uma entrada simples ou uma variável booleana interna, ou uma combinação de variáveis que corresponde a um valor booleano. A seguir é apresentado um exemplo de programação ST para transições:

(* programa SFC com programação ST para transições *)



B.3.6.2 Convenção LD

A linguagem **Ladder Diagram, Diagrama "Ladder"** (LD) pode ser utilizada para descrever uma **condição** associada a uma transição. O diagrama é composto por uma única lógica com uma bobina. O valor da bobina representa o valor da transição. A seguir é apresentado um exemplo de programação LD para transições:



B.3.6.3 Convenção IL

A programação em **Instruction List, Lista de Instruções** (IL) pode ser utilizada diretamente para descrever uma transição SFC , de acordo com a sintaxe a seguir:

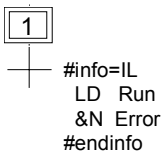
#info=IL
<instrução>
<instrução>
....
#endinfo

O valor do **resultado atual** (registrador IL) no final da seqüência IL é associado à transição pela seguinte regra:

resultado atual = 0 → condição é **FALSE**
resultado atual <> 0 → condição é **TRUE**

As palavras-chave especiais "#info=IL" e "#endinfo" devem ser inseridas exatamente desta forma, e **fazem distinção entre maiúsculas e minúsculas**. Caracteres de tabulação ou espaços não podem ser inseridos antes, no meio ou depois das palavras-chave. A seguir é apresentado um exemplo de programação IL para transições:

(* programa SFC com um programa IL para transições *)



B.3.6.4 Chamando funções de uma transição

Qualquer subprograma ou função (escrita em FBD, LD, ST ou IL) ou função "C" pode ser chamada para avaliar a condição associada a uma transição, de acordo com a seguinte sintaxe:

< subprograma > ( ) ;

O valor retornado pelo subprograma ou pela função deve ser booleano e dele decorre a condição resultante:

valor de retorno = FALSE	➔	condição é **FALSE**
valor de retorno = TRUE	➔	condição é **TRUE**

A seguir é apresentado um exemplo de um subprograma chamado em uma transição:

(* programa SFC com subprograma chamado para transições *)

B.3.7 Regras dinâmicas SFC

As **cinco** regras dinâmicas da linguagem SFC são:

Situação inicial

A situação inicial é caracterizada pelas **etapas iniciais** que estão, por definição, ativas no início da operação. **Pelo menos uma** etapa inicial deve estar presente em cada programa SFC.

Transposição de uma transição

Uma transição pode estar **ativada** ou **desativada** . Ela é dita ativada quando todos as etapas imediatamente precedentes ligadas ao seu símbolo de transição correspondente estiverem

ativas, caso contrário ela é dita desativada. Uma transição não pode ser **transposta** a menos que:
- ela esteja ativada, e
- a condição associada à transição seja verdadeira

Alterando o estado das etapas ativas

A transposição de uma transição leva simultaneamente ao estado ativo as etapas imediatamente seguintes, e ao estado inativo as etapas imediatamente precedentes.

Transposição simultânea de transições

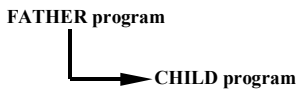
Linhas duplas podem ser utilizadas para indicar transições que devem ser transpostas simultaneamente. Se estas transições forem apresentadas separadamente, o estado de ativação das etapas precedentes (GSnnn.x) pode ser utilizado para expressar suas condições

Ativação e desativação simultânea de uma etapa

Se, durante a operação, uma etapa estiver ao mesmo tempo ativada e desativada, a prioridade é dada para a ativação.

B.3.8 Hierarquia de programas SFC

O ISaGRAF permite a descrição da estrutura vertical de programas SFC. Programas SFC são organizados em uma **árvore hierárquica**. Cada programa SFC pode controlar (iniciar, terminar) outros programas SFC. Tais programas são chamados **secundários** do programa SFC que os controla. Programas SFC são ligados entre si através de uma **árvore hierárquica** principal, utilizando uma relação **principal - secundário**.



As regras básicas impostas pela estrutura hierárquica são:
- Programas SFC que não têm programa principal são chamados programas "**principais**" SFC
- Programas principais SFC são ativados pelo sistema quando o aplicativo inicia
- Um programa pode ter vários programas secundários
- Um programa secundário não pode ter mais de um programa principal
- Um programa secundário pode ser controlado apenas pelo seu programa principal
- Um programa não pode controlar os programas secundários de um de seus programas secundários

As ações básicas que um programa principal SFC pode tomar para controlar seus programas secundários são:

Iniciar	(**GSTART**) Inicia o programa secundário: ativa cada uma das suas etapas iniciais. Secundários deste programa secundário não são iniciados automaticamente.
Terminar	(**GKILL**) Termina o programa secundário, desativando todas as suas etapas ativas. Todos os programas secundários deste programa também são terminados.
Congelar	(**GFREEZE**) Suspende a execução do programa (desativa ações de cada uma das etapas ativas e suspende o cálculo de transições), e memoriza o estado das etapas

	do programa de modo que o programa possa ser reiniciado. Todos os secundários deste programa também são congelados.
Reiniciar	(**GRST**) Reinicia um programa SFC congelado, reativando todas as etapas suspensas. Secundários deste programa não são automaticamente reiniciados.
Obter estado	(**GSTATUS**) Obtém o estado atual (ativo, inativo ou congelado) de um programa secundário.

B.4 Linguagem FC (Fluxograma)

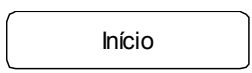
O **Fluxograma** (**FC**) é uma linguagem gráfica utilizada para descrever **operações seqüenciais**. Um Fluxograma é composto de **Ações** e **Testes**. Entre ações e teste existem **ligações orientadas** representando o fluxo de dados. Ligações de conexões múltiplas são utilizadas para representar divergências e convergências. Ações e testes podem ser descritos com as linguagens ST, LD ou IL. Funções e blocos de função de qualquer linguagem (exceto SFC) podem ser chamados a partir de ações e testes. Um Fluxograma pode chamar outro. O Fluxograma chamado é um **subprograma** do programa FC que o está chamado.

B.4.1 Componentes FC

Os componentes gráficos da linguagem de Fluxograma são apresentados a seguir:

- ***Início do Gráfico FC***

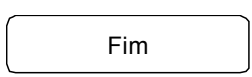
Um símbolo "**Início**" deve aparecer no início de um programa FC. Ele deve ser único e não pode ser omitido. Ele representa o estado inicial do gráfico quando estiver ativo. O desenho de um símbolo "Início" é apresentado a seguir:



O símbolo "Início" sempre tem uma conexão (na parte inferior) para outros objetos do gráfico. Um Fluxograma não é válido se não for efetuada nenhuma conexão do "Início" para outro objeto.

- ***Fim do Gráfico FC***

Um símbolo "**Fim**" deve aparecer no final de um programa FC. Ele deve ser único e não pode ser omitido. É possível que nenhuma conexão seja feita para o símbolo "Fim" (gráfico com laço eterno), mas o símbolo "Fim" ainda é apresentado na parte inferior do gráfico. Ele representa o estado final do gráfico, quando sua execução foi completada. O desenho de um símbolo "Fim" é apresentado a seguir:



O símbolo "Fim" geralmente tem uma conexão (na parte de cima) para outros objetos do gráfico. Um Fluxograma pode não ter conexão com o objeto "Fim" (gráfico com loop sem fim). O objeto "Fim" ainda é visível na parte inferior do gráfico neste caso.

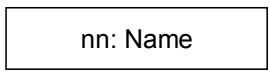
- ***Linhas de fluxo FC***

Uma **linha de fluxo** é uma linha que representa um fluxo entre dois pontos do diagrama. Uma linha é sempre terminada por uma flecha na extremidade. O desenho de uma linha de fluxo é apresentado a seguir:

Duas linhas de fluxo não podem ser conectadas ao mesmo ponto de origem.

Ações FC

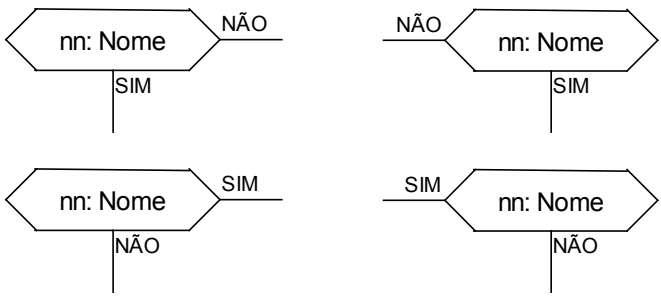
Um símbolo de **ação** representa ações a serem realizadas. Uma ação é identificada por um número e um nome. O desenho de um símbolo de "ação" é apresentado a seguir:



Dois objetos diferentes do mesmo gráfico não podem ter o mesmo nome ou número lógico. Pode-se utilizar a linguagem de programação ST, LD ou IL para uma ação. Uma ação está sempre conectada por linhas de fluxo, uma chegando, e outra partindo dela.

Condições FC

Uma **condição** representa um **teste** booleano. Uma condição é identificada por um número e um nome. De acordo com o resultado da expressão ST, LD ou IL associada, o fluxo é encaminhado para "YES" ou para "NO". Os possíveis aspectos de um símbolo de condição são apresentados a seguir:



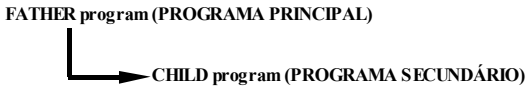
Dois objetos diferentes de um mesmo gráfico não podem ter o mesmo nome ou número lógico. A programação de um teste pode ser
- uma expressão em ST, ou
- uma única lógica em LD, sem nenhum símbolo associado a uma única bobina, ou
- várias instruções em IL. O registrador IL (ou resultado atual) é utilizado para avaliar a condição.

Quando programada em texto ST, a expressão pode opcionalmente ser seguida por um ponto e vírgula. Quando programada em LD, a lógica única representa o valor da condição. Uma condição igual a :
- 0 ou FALSE encaminha o fluxo para NO
- 1 ou TRUE encaminha o fluxo para YES

Um teste é sempre conectado por uma linha de fluxo de chegada, e as duas conexões à frente devem ser definidas.

Subprograma FC

O ISaGRAF permite a descrição da estrutura vertical de programas FC. Os programas FC são organizados em uma **árvore hierárquica**. Cada programa FC pode chamar outros FC. Tais programas são chamados **secundários** do programa FC que os controla. Programas que chamam subprogramas são chamados **programas principais**. Os programas FC são ligados entre si através de uma **árvore hierárquica** principal, utilizando uma relação **principal - secundário**.



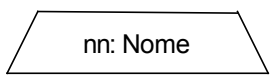
Um símbolo de **subprograma** em um Fluxograma representa uma chamada a um programa secundário FC. A execução do programa que chama é suspensa até que a execução do subprograma seja completada. Um subprograma FC é identificado por um número e um nome, como outros programas, funções e blocos de função. O símbolo de uma "chamada a subprograma" é apresentado a seguir:

Dois objetos diferentes do mesmo gráfico não podem ter o mesmo número lógico. As regras básicas impostas pela estrutura hierárquica FC são:
- Programas FC que não têm programa principal são chamados programas "**principais**" FC
- Programas principais FC são ativados pelo sistema quando o aplicativo inicia
- Um programa pode ter vários programas secundários
- Um programa secundário não pode ter mais de um programa principal
- Um programa secundário pode ser chamado apenas pelo seu programa principal
- Um programa não pode chamar os programas secundários a partir de um de seus programas secundários

O mesmo subprograma pode aparecer várias vezes no Fluxograma do programa principal. Uma chamada a um subprograma FC representa a execução completa do subFluxograma. A execução do programa principal é suspensa durante a execução do subprograma. Os blocos de chamada de subprograma devem seguir as mesmas regras de conexão dos blocos definidos para ações.

Ação específica de E/S FC

Um símbolo de **ação específica de E/S** representa as ações a serem executadas. Como outras ações, uma ação específica de E/S é identificada por um número e um nome. A mesma semântica é utilizada em ações padrão e ações de E/S específicas. A finalidade das ações específicas de E/S é apenas tornar o Fluxograma mais legível e voltar o foco para as partes não portáveis do gráfico. A utilização de ações específicas de E/S é uma característica opcional. O símbolo de uma "ação específica de E/S" é apresentado a seguir:



Blocos específicos de E/S têm exatamente o mesmo comportamento das ações padrão, incluindo as suas propriedades, programação ST, LD ou IL, e as regras de conexão.

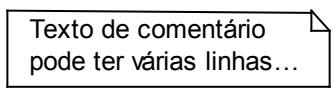
Conectores FC

Conectores são utilizados para representar uma ligação entre dois pontos de um diagrama sem desenhá-la. Um conector é representado por um círculo e é conectado à fonte do Fluxograma. O desenho do conector é finalizado, no lado apropriado (dependendo da direção do fluxo dos dados), com uma identificação do alvo (geralmente o nome do símbolo alvo). Abaixo, apresentamos o desenho padrão de um conector:

Um conector sempre tem como destino um símbolo definido no Fluxograma. O símbolo de destino é identificado pelo seu número lógico.

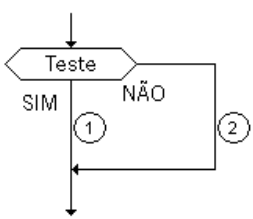
Comentários FC

Um bloco de **comentário** contém um texto que não tem sentido semântico para o Fluxograma. Ele pode ser inserido em qualquer espaço não utilizado da janela do Fluxograma, e serve para documentar o programa. O símbolo de um "comentário" é apresentado a seguir:



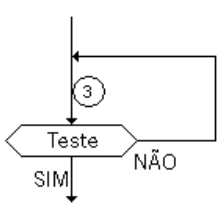
B.4.2 Exemplos de estruturas complexas FC

Esta seção mostra exemplos de **estruturas complexas** que podem ser definidas em um Fluxograma. Tais estruturas são combinações de objetos básicos ligados entre si.



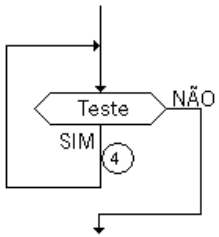
IF / THEN / ELSE (Se / Então / Senão)

(1) lugar para inserção de ações "THEN"
(2) lugar para inserção de ações "ELSE"



REPEAT / UNTIL (Repita / Até)

(3) lugar para inserção de ações repetidas



WHILE / DO (Enquanto / Faça)

(4) lugar para inserção de ações repetidas

B.4.3 Comportamento dinâmico FC

A **execução** de um Fluxograma pode ser explicada conforme a seguir:

- O símbolo "Início" toma um ciclo do destino e inicia a execução de um Fluxograma.
- O símbolo "Fim" toma um ciclo do destino e termina a execução do Fluxograma. Após esse símbolo ser alcançado, nenhuma ação do Fluxograma será executada.
- O fluxo é quebrado cada vez que um item (ação, decisão) for encontrado e que já tenha sido alcançado no mesmo ciclo do destino. Neste caso o fluxo continuará no próximo ciclo.

Observação: Ao contrário do SFC, uma ação não é um estado estável. Não há repetição de instruções enquanto o símbolo da ação estiver destacado.

B.4.4 Verificação do FC

Além da programação ST, LD ou IL associada, algumas outras **regras de sintaxe** aplicam-se ao Fluxograma propriamente dito. A lista das principais regras é apresentada a seguir:
- Todos os pontos de conexão de todos os símbolo devem estar conectados. (a conexão ao símbolo "Fim" pode ser omitida)
- Todos os símbolos devem estar ligados entre si (não deve haver nenhuma parte isolada)
- Todos os conectores devem ter um destino válido

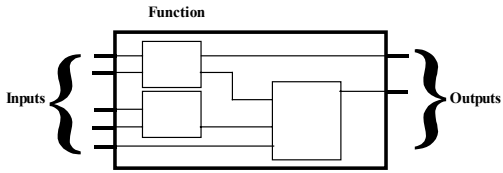
Outros erros menores de sintaxe podem ser reportados:
- Ações vazias (sem programação) serão consideradas como etapas durante a execução
- Testes vazios (sem programação) serão considerados "sempre true"

B.5 Linguagem FBD

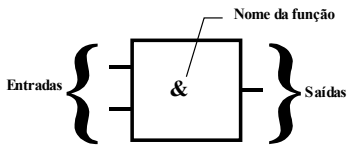
O **Diagrama de Blocos Funcionais** (FBD) é uma linguagem gráfica. Ele permite ao programador construir procedimentos complexos utilizando **funções** existentes na biblioteca do ISaGRAF e **conectando-as** na área do diagrama.

B.5.1 Formato geral do diagrama FBD

Um diagrama FBD descreve uma função entre **variáveis de entrada** e **variáveis de saída**. Uma função é descrita como um conjunto de **blocos de função elementares**. Variáveis de entrada e de saída são conectadas aos blocos por **linhas de conexão**. Uma saída de um bloco de função também pode ser conectada a uma entrada de um outro bloco.



Uma função inteira realizada por um programa FBD é construída com blocos de função **elementar** padrão. Cada bloco de função tem um número fixo de **pontos de conexão de entrada** e um número fixo de **pontos de conexão de saída**. Um bloco de função é representado por um **retângulo** simples. As entradas são conectadas no seu lado **esquerdo**. As saídas são conectadas no seu lado **direito**. Um bloco de função básico realiza uma **função** entre as suas entradas e saídas. O nome da função a ser realizada pelo bloco aparece dentro de seu símbolo retangular. Cada entrada ou saída do bloco tem um **tipo** bem definido.



As variáveis de entrada de um FBD devem ser conectadas a pontos de conexão de entrada de blocos de função. Cada variável deve ser do mesmo tipo esperado para a entrada associada. A Entrada para um diagrama FBD pode ser uma expressão **constante**, qualquer variável **interna**, de **entrada** ou de **saída**.

As variáveis de saída de um programa FBD devem ser conectadas em pontos de conexão de saída dos blocos de função. Cada variável deve ser do mesmo tipo esperado para a saída associada do bloco. Uma Saída para um diagrama FBD pode ser qualquer variável **interna** ou de **saída**, ou o nome de um programa (apenas para **subprogramas**). Quando a saída for o nome do subprograma que está sendo editado no momento, ela representa o valor de retorno do subprograma (ao retornar para o programa que o chamou).

As variáveis de entrada e de saída, entradas e saídas dos blocos de função são ligados entre si através de **linhas de conexão**. As linhas simples podem ser utilizadas para **conectar** dois pontos do diagrama.:

- Uma variável de entrada e uma entrada de um bloco de função
- Uma saída de um bloco de função e uma entrada de outro bloco
- Uma saída de um bloco de função e uma variável de saída

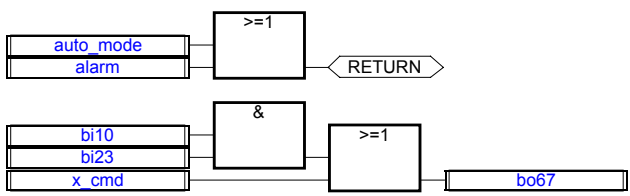
A conexão é **orientada**, de modo que a linha transporta dados associados da extremidade esquerda para a extremidade direita. Extremidades esquerda e direita da linha de conexão devem ser do **mesmo tipo**.

Conexões múltiplas à direita podem ser utilizadas para transmitir uma informação de sua extremidade esquerda para cada uma das suas extremidades direitas. Todas as extremidades da conexão devem ser do mesmo tipo.

B.5.2 Declaração RETURN

A palavra-chave "**<RETURN>**" pode ocorrer como uma saída do diagrama. Ela deve ser conectada a um ponto de conexão booleano de saída de um bloco de função. A declaração RETURN representa um **fim condicional** do programa: se a saída do bloco de função conectada à declaração resultar no valor **TRUE**, o resto do diagrama não será executado.

(* Exemplo de um programa FBD utilizando uma declaração RETURN *)




```
(* Equivalência ST: *)
If auto_mode OR alarm Then
        Return;
End_if;
bo67 := (bi10 AND bi23) OR x_cmd;
```

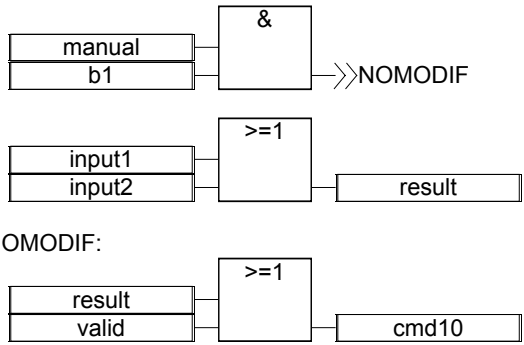
B.5.3 Desvios e rótulos

Desvios e rótulos são utilizados para controlar a execução do diagrama. Nenhum outro objeto pode ser conectado á direita de um símbolo de desvio ou de rótulo. As notações a seguir são utilizadas:

>>LAB.....desviar para um rótulo (nome do rótulo é “ROT”)
LAB:.....definição de um rótulo (nome do rótulo é “ROT”)

Se a linha de conexão à **esquerda** do símbolo de desvio resultar no estado booleano **TRUE**, a execução do programa desvia diretamente para o símbolo do rótulo correspondente.

(* Exemplo de um programa FBD utilizando rótulos e desvios *)

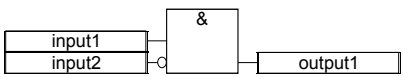



```
(* Equivalência IL: *)
                ld      manual
                and     b1
                jmpc    NOMODIF
                ld      input1
                or      input2
                st      result
NOMODIF:        ld      result
                or      valid
                st      cmd10
```

B.5.4 Inversão booleana

Uma linha de conexão simples com sua extremidade direita conectada à entrada de um bloco de função pode ser terminada por um símbolo de **inversão booleana**. A inversão é representada por um pequeno círculo. Quando uma inversão booleana for utilizada, as extremidades esquerda e direita da linha de conexão devem ser do tipo **BOOLEANO**.

(* Exemplo de um programa FBD utilizando inversão booleana *)

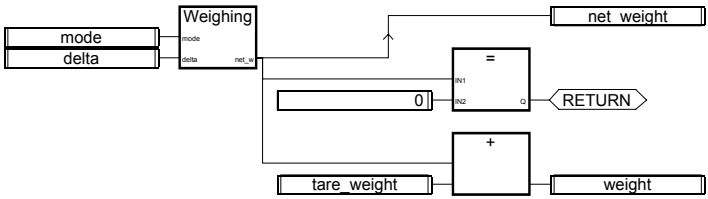



```
(* Equivalência ST: *)
output1 := input1 AND NOT (input2);
```

B.5.5 Chamando funções ou blocos de função do FBD

A linguagem FBD permite a chamada de subprogramas , funções ou blocos de função. Um subprograma, uma função ou um bloco de função são representados por uma caixa de função. O nome que aparece na caixa é o nome do subprograma, da função ou do bloco de função.
No caso de um subprograma ou de uma função, o valor de retorno é a única saída do bloco de função. Um bloco de função pode ter mais de uma saída.

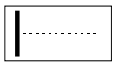
(* Exemplo de um programa FBD utilizando um bloco de SUBPROGRAMA *)

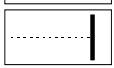


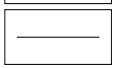
(* Equivalência ST *)
net_weight := Weighing (mode, delta); (* call sub-program *)
If (net_weight = 0) Then Return; End_if;
weight := net_weight + tare_weight;

B.6 Linguagem LD

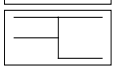
Um Diagrama Ladder (LD) é uma representação gráfica de equações booleanas, combinando **contatos** (argumentos de **entrada**) com **bobinas** (resultados de saída). A linguagem LD permite a descrição de testes e modificações de dados **booleanos** pela disposição de **símbolos gráficos** no gráfico do programa. Diagramas LD são conectados à esquerda e à direita das **barras de energia** verticais.

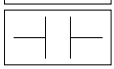

---------- Barra de energia vertical à esquerda

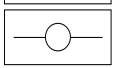

---------- Barra de energia vertical à direita


---------- Linha de conexão horizontal


---------- Linha de conexão vertical

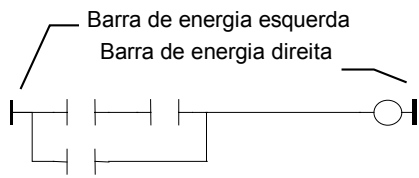

---------- Linhas de conexão múltiplas (todas conectadas entre si)


---------- Contato associado a uma variável

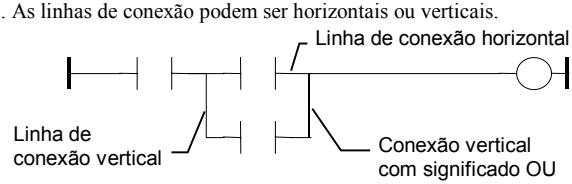

---------- Bobina associada a uma saída ou a uma variável interna

B.6.1 Barras de energia e linhas de conexão

Um diagrama LD é limitado à esquerda e à direita por linhas verticais, chamadas **barra de energia esquerda** e **barra de energia direita**, respectivamente.



Os símbolos gráficos de um diagrama LD são conectados às barras de energia ou a outros símbolos por **linha de conexão**. As linhas de conexão podem ser horizontais ou verticais.

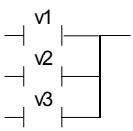


Cada segmento de linha tem um estado booleano **FALSE** ou **TRUE**. O estado booleano é o mesmo para todos os segmentos ligados diretamente. Qualquer linha horizontal conectada à **barra de energia esquerda** resulta no estado **TRUE**.

B.6.2 Conexão múltipla

O estado booleano resultante em uma conexão horizontal simples é o mesmo nas extremidades esquerda e direita da linha. A combinação de linhas horizontais e verticais permite a construção de **conexões múltiplas**. O estado booleano das extremidades de uma conexão múltipla segue as regras lógicas.
Uma **conexão múltipla à esquerda** combina **mais de uma** linha horizontal conectadas ao lado **esquerdo** de uma linha vertical, e **uma** linha conectada ao seu lado **direito**. O estado booleano da extremidade direita é o resultado de um "**OU LÓGICO**" entre todas as extremidades da esquerda.

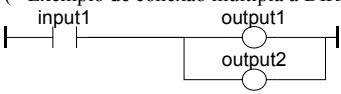
(* Exemplo de uma conexão múltipla à ESQUERDA *)



(* estado da extremidade direita é (v1 OR v2 OR v3) *)

Uma **conexão múltipla à direita** combina **uma** linha horizontal conectada à **esquerda** de uma linha vertical e **mais de uma** linha conectadas à sua **direita**. O estado booleano da extremidade esquerda é propagado a cada uma das extremidades da direita.

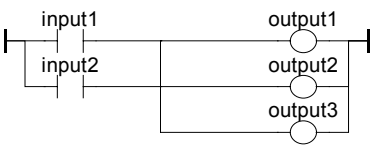
(* Exemplo de conexão múltipla à DIREITA *)




```
(* Equivalência ST: *)
output1 := input1;
output2 := input1;
```

Uma **conexão múltipla à esquerda e à direita** combina **mais de uma** linha horizontal conectadas à **esquerda** de uma linha vertical, e **mais de uma** linha horizontal conectadas à sua **direita**. O estado booleano de cada uma das extremidades direitas é o resultado de um "**OU LÓGICO**" entre todas as extremidades da esquerda.

(* Exemplo de conexão múltipla à ESQUERDA e à DIREITA *)



(* Equivalência ST: *)
output1 := input1 OR input2;
output2 := input1 OR input2;
output3 := input1 OR input2;

B.6.3 Contatos e bobinas LD básicos

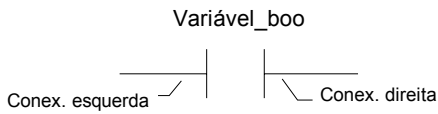
Existem vários símbolos disponíveis para contatos de entrada :
- Contato normalmente aberto
- Contato normalmente fechado
- Contato com detecção de borda

Existem vários símbolos disponíveis para bobinas de saída :
- Bobina simples
- Bobina invertida
- Bobina Liga
- Bobina Desliga
- Bobinas com detecção de borda

O nome da variável aparece acima de qualquer destes símbolos gráficos:

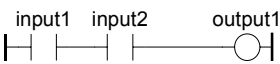
Contato normalmente aberto

Um contato normalmente aberto permite uma **operação booleana** entre o estado de uma **linha de conexão** e uma **variável** booleana.



O estado da linha de conexão à direita do contato é o resultado do **"E LÓGICO"** entre o estado da conexão à esquerda e o valor da variável associada ao contato.

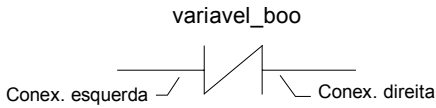
(* Exemplo utilizando contatos normalmente abertos *)



(* Equivalência ST: *)
output1 := input1 AND input2;

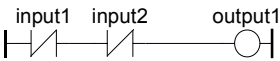
Contato normalmente fechado

Um Contato normalmente fechado permite uma **operação booleana** entre o estado de uma **linha de conexão** e o inverso de uma **variável** booleana



O estado da linha de conexão à direita do contato é o resultado do **"E LÓGICO"** entre o estado da conexão à esquerda e o inverso do valor da variável associada ao contato.

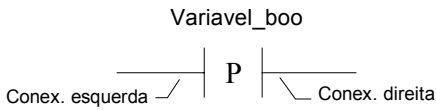
(* Exemplo utilizando contatos normalmente fechados *)




```
(* Equivalência ST: *)
output1 := NOT (input1) AND NOT (input2);
```

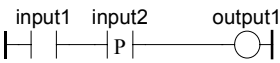
Contato com detecção de borda de subida

Este contato permite uma **operação booleana** entre o estado da **linha de conexão** e a borda de subida de uma **variável** booleana.



O estado da linha de conexão à direita do contato resulta no valor **TRUE** quando o estado da linha de conexão à esquerda do contato é **TRUE**, e o estado da variável associada **sobe** de FALSE para TRUE. Ela resulta no valor FALSE em quaisquer outros casos.

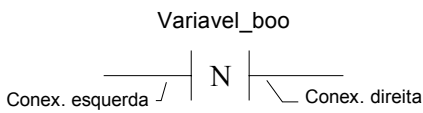
(* Exemplo utilizando contato com detecção de borda de subida *)




```
(* Equivalência ST: *)
output1 := input1 AND (input2 AND NOT (input2prev));
(* input2prev é o valor de input2 no ciclo anterior *)
```

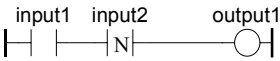
Contato com detecção de borda de descida

Este contato permite uma **operação booleana** entre o estado da **linha de conexão** e a borda de descida de uma **variável** booleana.



O estado da linha de conexão à direita do contato resulta no valor **TRUE** quando o estado da linha de conexão à esquerda do contato é **TRUE**, e o estado da variável associada **desce** de TRUE para FALSE. Ela resulta no valor FALSE em quaisquer outros casos.

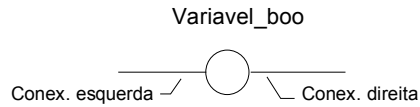
(* Exemplo utilizando contato com detecção de borda de descida *)



(* Equivalência ST: *)
output1 := input1 AND (NOT (input2) AND input2prev);
(*input2prev é o valor de input2 no ciclo anterior *)

Bobina simples

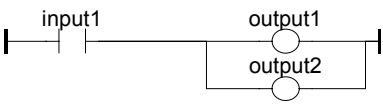
Bobinas simples permitem uma **saída booleana** do estado de uma **linha de conexão**.



A variável associada assume o valor do **estado booleano da conexão à esquerda**. O estado da conexão à esquerda é propagado para a conexão à direita. A conexão à direita pode ser conectada à barra de energia vertical da direita.

A variável booleana associada deve ser de **SAÍDA** ou **INTERNA**.
O nome associado pode ser o nome do programa (apenas para **subprogramas**), que neste caso corresponde ao valor de retorno do subprograma.

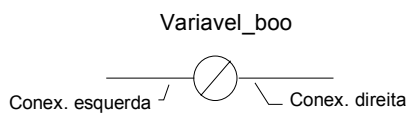
(* Exemplo utilizando bobinas simples *)



(* Equivalência ST: *)
output1 := input1;
output2 := input1;

Bobina invertida

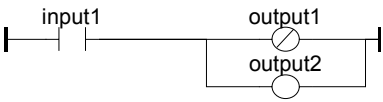
Bobinas invertidas permitem uma **saída booleana** com o valor **inverso** do estado de uma **linha de conexão**.



A variável associada assume um valor que é o resultado do **inverso** do **estado booleano da conexão à esquerda**. O estado da conexão à esquerda é propagado para a conexão à direita. A conexão à direita pode ser conectada à barra de energia vertical da direita.

A variável booleana associada deve ser de **SAÍDA** ou **INTERNA**.
O nome associado pode ser o nome do programa (apenas para **subprogramas**), que neste caso corresponde ao valor de retorno do subprograma.

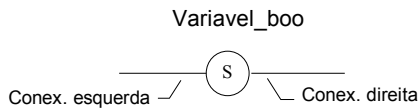
(* Exemplo utilizando bobinas invertidas *)




```
(* Equivalência ST: *)
output1 := NOT (input1);
output2 := input1;
```

Bobina Liga

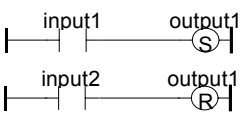
Bobinas "liga" permitem uma **saída booleana** do estado de uma **linha de conexão**.



A variável associada é **ATRIBUÍDA COM TRUE** quando o **estado booleano da conexão à esquerda** torna-se **TRUE**. A variável de saída mantém este valor até que uma ordem inversa seja dada por uma Bobina "Desliga". O estado da conexão à esquerda é propagado para a conexão à direita. A conexão à direita pode ser conectada à barra de energia vertical da direita.

A variável booleana associada deve ser de **SAÍDA** ou **INTERNA**.

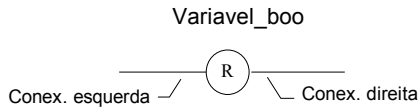
(* Exemplo utilizando bobinas "liga" e "desliga" *)




```
(* Equivalência ST: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

Bobina Desliga

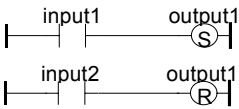
Bobinas “desliga” permitem uma **saída booleana** do estado de uma **linha de conexão**.



A variável associada é **ATRIBUÍDA COM FALSE** quando o **estado booleano da conexão à esquerda** torna-se **TRUE**. A variável de saída mantém este valor até que uma ordem inversa seja dada por uma Bobina “Liga”. O estado da conexão à esquerda é propagado para a conexão à direita. A conexão à direita pode ser conectada à barra de energia vertical da direita.

A variável booleana associada deve ser de **SAÍDA** ou **INTERNA**.

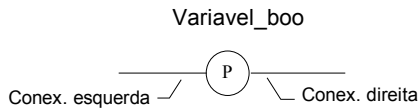
(* Exemplo utilizando bobinas “liga” e “desliga” *)




```
(* Equivalência ST: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

Bobina com detecção de borda de subida

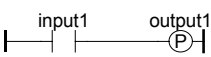
Bobinas com detecção de borda de subida permitem uma **saída booleana** do estado de uma **linha de conexão**. Este tipo de bobina só está disponível no editor Quick Ladder.



A variável associada é **ATRIBUÍDA COM TRUE** quando o **estado booleano da conexão à esquerda sobe** de **FALSE** para **TRUE**. A variável de saída será atribuída com FALSE em quaisquer outros casos. O estado da conexão à esquerda é propagado para a conexão à direita. A conexão à direita pode ser conectada à barra de energia vertical da direita.

A variável booleana associada deve ser de **SAÍDA** ou **INTERNA**.

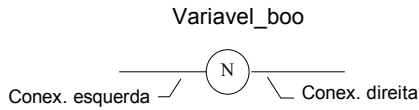
(* Exemplo utilizando uma bobina com detecção de borda de subida *)



```
(* Equivalência ST: *)
IF (input1 and NOT(input1prev)) THEN
  output1 := TRUE;
ELSE
  output1 := FALSE;
END_IF;
(*input1prev é o valor de input1 no ciclo anterior *)
```

Bobina com detecção de borda de descida

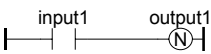
Bobina com detecção de borda de descida permitem uma **saída booleana** do estado de uma **linha de conexão**. Este tipo de bobina só está disponível no editor Quick LD.



A variável associada é **ATRIBUÍDA COM TRUE** quando o **estado booleano da conexão à esquerda desce** de **TRUE** para **FALSE**. A variável de saída será atribuída com FALSE em quaisquer outros casos. O estado da conexão à esquerda é propagado para a conexão à direita. A conexão à direita pode ser conectada à barra de energia vertical da direita.

A variável booleana associada deve ser de **SAÍDA** ou **INTERNA**.

(* Exemplo utilizando uma bobina com detecção de borda de descida *)



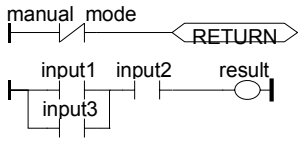

```
(* Equivalência ST: *)
IF (NOT(input1) and input1prev) THEN
  output1 := TRUE;
ELSE
  output1 := FALSE;
END_IF;
(*input1prev é o valor de input1 no ciclo anterior *)
```

B.6.4 Declaração RETURN

O rótulo **RETURN** pode ser utilizado como uma saída para representar um fim condicional do programa. Nenhuma conexão pode ser ligada à direita de um símbolo de RETURN.

Se a saída do bloco de função conectada à declaração tem valor **TRUE**, o programa termina sem executar as equações inseridas nas linhas seguintes do programa.
Observação: Quando o programa LD for um subprograma, seu nome tem que ser associado a uma bobina de saída para atribuição do valor de retorno (retornado ao programa que o chamou).

(* Exemplo utilizando o símbolo RETURN *)



(* Equivalência ST: *)
If Not (manual_mode) Then RETURN; End_if;
result := (input1 OR input3) AND input2;

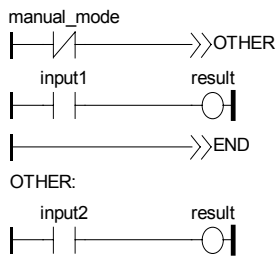
B.6.5 Desvios e rótulos

Rótulos e desvios condicionais e incondicionais podem ser utilizados para controlar a execução do diagrama. Nenhum outro objeto pode ser conectado à direita de um símbolo de desvio ou de rótulo. As seguintes notações são utilizadas:

>>LAB.....desviar para um rótulo (nome do rótulo é "ROT")
LAB:.....definição de um rótulo (nome do rótulo é "ROT")

Se a linha de conexão à **esquerda** do símbolo de desvio resultar no estado booleano **TRUE**, a execução do programa desvia diretamente para o símbolo do rótulo correspondente.

(* Exemplo de um programa LD utilizando rótulos e desvios *)



(* Equivalência IL: *)

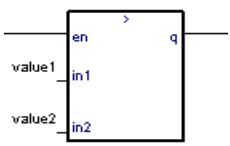
```
                ldn         manual_mode
                jmpc        other
                ld          input1
                st          result
                jmp         END
OTHER:          ld          input2
                st          result
END:            (* end of program *)
```

B.6.6 Blocos em LD

Utilizando o Editor Quick LD, você conecta blocos de função a linha booleanas. Uma função pode na verdade ser um operador, um bloco de função ou uma função. Como nem todos os blocos têm sempre uma entrada booleana e/ou uma saída booleana, a inserção de blocos em um diagrama LD acarreta o acréscimo de novos parâmetros, EN e ENO, à interface do bloco. Os parâmetros EN e ENO não são acrescentados se for utilizado o editor FBD/LD, já que é possível conectar a variável com o tipo necessário.

A entrada "EN"

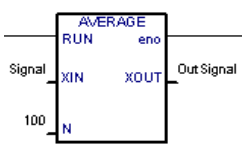
Em alguns operadores, funções ou blocos de função, a primeira entrada não é do tipo booleano. Como a primeira entrada deve estar sempre conectada à lógica, a outra entrada é automaticamente inserida na primeira posição, chamada "**EN**". O bloco é executado somente se a entrada **EN** for TRUE. O exemplo de um operador de comparação é apresentado a seguir, e o código equivalente expresso em ST:




```
IF estado_logica THEN
  q := (value1 > value 2);
ELSE
  q := FALSE;
END_IF;
(*continua logica com estado q *)
```

A saída "ENO"

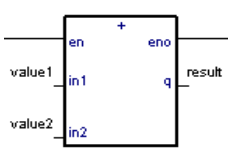
Em alguns operadores, funções ou blocos de função, a primeira saída não é do tipo booleano. Como a primeira saída deve estar sempre conectada à lógica, a outra saída é automaticamente inserida na primeira posição, chamada "**ENO**". A saída **ENO** sempre recebe o mesmo estado da primeira entrada do bloco. Um exemplo com o bloco de função AVERAGE é apresentado a seguir, e o código equivalente expresso em ST:




```
AVERAGE(estado_logica, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := estado_logica;
(*continua logica com estado eno *)
```

Utilizando "EN" e "ENO"

Em alguns casos, tanto **EN** quanto **ENO** são necessários. Um exemplo com um operador aritmético é apresentado a seguir, e o código equivalente expresso em ST:




```
IF estado_logica THEN
  result := (value1 + value2);
END_IF;
eno := estado_logica;
(*continua logica com estado eno*)
```

B.7 Linguagem ST

ST (**Texto Estruturado**) é uma linguagem estruturada de alto nível projetada para processos de automação. Esta linguagem é utilizada principalmente para implementar procedimentos complexos que não podem ser facilmente expressos com linguagens gráficas. ST é a linguagem padrão para a descrição das ações dentro das etapas e condições associadas a transições da linguagem **SFC**.

B.7.1 Sintaxe geral ST

Um programa ST é uma lista de **comandos**. Cada comando termina com um separador ponto-e-vírgula (**";"**). Nomes utilizados no código fonte (identificadores de variáveis, constantes, palavras-chave da linguagem...) são separados por **separadores inativos** (espaço, fim de linha, caracteres de tabulação...) ou por **separadores ativos**, que têm um significado bem definido (por exemplo, o separador **">"** indica uma comparação "maior que"). Comentários podem ser livremente inseridos no texto. Um comentário deve iniciar com **"(*"** e terminar com **"*)"**. Cada comando termina com um separador ponto-e-vírgula (**";"**). Estes são os tipos básicos de comandos ST:

- **atribuição** (variável := expressão;)
- chamada de **subprograma** ou **função**
- chamada de **bloco de função**
- **seleção** (IF, THEN, ELSE, CASE...)
- **iteração** (FOR, WHILE, REPEAT...)
- **controle** (RETURN, EXIT...)
- comandos especiais para ligação com outras linguagens como **SFC**.

Separador inativo podem ser inseridos livremente entre separadores ativos, expressões constantes e identificadores. Os separadores inativos ST são os caracteres **Espaço**, **Tab** e **Fim de linha**. De maneira diferente de outras linguagens formatadas por linha como IL, finais de linha podem ser inseridos em qualquer lugar no programa. As regras a seguir devem ser seguidas quando se usa separadores inativos para melhorar a clareza do programa ST.:

- Não escreva mais de um comando em uma linha
- Use Tabs para indentar comandos complexos
- Insira comentários para aumentar a clareza de linhas ou parágrafos

B.7.2 Expressões e parênteses

Expressões ST combinam **operadores** ST e **operandos** variáveis ou constantes. Para cada uma das expressões (combinando operandos com um operador ST), o **tipo** dos operandos deve ser o mesmo. Esta expressão terá o mesmo tipo dos seus operandos, e pode ser utilizada em uma expressão mais complexa. Por exemplo:

(boo_var1 AND boo_var2)	Tem tipo BOO
not (boo_var1)	Tem tipo BOO
(sin (3.14) + 0.72)	Tem tipo REAL ANALOG
(t#1s23 + 1.78)	é uma expressão inválida

Parênteses são utilizados para isolar sub partes da expressão, e para determinar explicitamente a ordem da operações. Quando não há parênteses em uma expressão complexa, a seqüência de operação é dada implicitamente pela **precedência** padrão entre operadores ST. Por exemplo:

2 + 3 * 6	igual a 2+18=20	operador de multiplicação tem precedência
(2+3) * 6	igual a 5*6=30	precedência é dada pelos parênteses

Advertência: Um número máximo de **8** níveis de parênteses podem ser encadeados dentro de uma expressão.

B.7.3 Chamadas de função ou blocos de função

Chamadas de função padrão ST podem ser utilizadas para cada um dos seguintes objetos:
- Subprogramas
- Funções da biblioteca e blocos de função escritos em linguagens IEC
- Funções "C" e blocos de função
- Funções de conversão de tipo

Chamando subprogramas ou funções

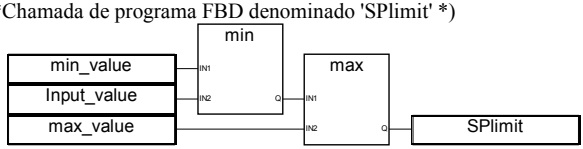
Nome:	nome do subprograma chamado ou função da biblioteca escrita em linguagem IEC ou em "C"
Significado:	chama um subprograma ST, IL, LD or FBD ou função ou um função "C" e obtém seu valor de retorno
Sintaxe:	<**variável**> **:=** <**subprog**> **(**<**par1**>**,** ... <**parN**> **);**
Operandos:	O tipo do valor de retorno e parâmetros da chamada deve seguir a interface definida para o subprograma.
Valor de retorno:	valor retornado pelo subprograma

Chamadas a subprogramas podem ser utilizadas em qualquer expressão. Elas também podem ser utilizadas em transições SFC.

Exemplo1: Chamada de subprograma

```
(* Programa principal ST *)
(* obtém um valor analógico e converte para um valor de tempo limitado *)
ana_timeprog := SPlimit ( tprog_cmd );
appl_timer := tmr (ana_timeprog * 100);
```

(*Chamada de programa FBD denominado 'SPlimit' *)



Exemplo2: Chamada de função

```
(* funções utilizadas em expressões complexas: min, max, right, mlen e left são funções "C" padrão *)
limited_value := min (16, max (0, input_value) );
rol_msg := right (message, mlen (message) - 1) + left (message, 1);
```

Chamando blocos de função

Name:	nome da instância do bloco de função
Significado:	chama um bloco de função da biblioteca ISaGRAF ou da biblioteca do usuário e acessa seus parâmetros de retorno
Sintaxe:	(* chamada do bloco de função *) <**blockname**> **(** <**p1**>**,** <**p2**> ... **);** (* obtém seus parâmetros de retorno *) <**result**> **:=** <**blockname**>**.** <**ret_param1**>**;** ... <**result**> **:=** <**blockname**>**.** <**ret_paramN**>**;**
Operandos:	Parâmetros são expressões de tipos iguais aos tipos dos parâmetros especificados para o bloco de função
Valor de retorno:	Veja Sintaxe para obter os parâmetros de retorno.

Consulte a biblioteca ISaGRAF para obter o significado e o tipo de cada parâmetro do bloco. A instância do bloco de função (nome da cópia) deve ser declarada no dicionário.

Exemplo :

```
(* Programa ST chamando um bloco de função *)

(* declare a instância do bloco no dicionário: *)
(* trigb1 : bloco R_TRIG - detecção de borda de subida*)

(* ativação de bloco de função da linguagem ST *)
trigb1 (b1);
(* acesso aos parâmetros de retorno*)
If (trigb1.Q) Then nb_edge := nb_edge + 1; End_if;
```

B.7.4 Operadores booleanos ST específicos

Os seguintes operadores booleanos são específicos à linguagem ST:

- REDGE	detecção de borda de subida
- FEDGE	detecção de borda de descida

Outros operadores booleanos padrões tais como:

- NOT	inversão booleana
- AND (&)	E lógico
- OR	OU lógico
- XOR	OU exclusivo lógico

podem ser utilizados. Suas descrições podem ser encontradas na seção ‘Operadores padrão, blocos de função e funções’.

Operador "REDGE"

Name: **REDGE**

Significado:	avalia a borda de subida de uma expressão booleana completa
Sintaxe:	**<borda> := REDGE (<expressão_boo >,< variável_memo>);**
Operandos:	primeiro operando é qualquer variável booleana ou expressão complexa segundo operando é uma variável booleana interna utilizada para armazenar o último estado da expressão
Valor de retorno:	TRUE quando a expressão muda de FALSE para TRUE FALSE para todos os outros casos

A borda de subida não pode ser detectada mais do que uma vez durante o mesmo ciclo de execução, utilizando-se o operador REDGE. Este operador pode ser utilizado para descrever a condição associada a uma transição SFC.

Advertência: A variável booleana "memo" utilizada para armazenar o último estado da expressão não pode ser utilizada como acionador de bordas para diferentes expressões.

Quando a expressão é uma variável booleana de nome "**xxx**", uma variável interna única de nome "**EDGE_xxx**" deve ser declarada e usada em expressões REDGE para esta variável. Este método assegura que a variável de memória não será sobrescrita durante outras operações REDGE.

Exemplo:

```
(* Programa ST utilizando operador REDGE *)

(* este programa conta as bordas de subida de uma entrada booleana *)
(* Bi120 é uma variável booleana de entrada *)
(* Edge_Bi120 é a memória do estado da variável Bi120 *)

If REDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Observação: este operador não está na norma IEC1131-3. Você pode preferir o uso do bloco padrão R_TRIG. Este operador foi mantido por razões de compatibilidade.

Operador "FEDGE"

Name:	**FEDGE**
Significado:	avalia a borda de descida de uma expressão booleana completa
Sintaxe:	**<borda> := FEDGE (<expressão_boo >,< variável_memo>);**
Operandos:	primeiro operando é qualquer variável booleana ou expressão complexa segundo operando é uma variável booleana interna utilizada para armazenar o último estado da expressão
Valor de retorno:	TRUE quando a expressão muda de TRUE para FALSE FALSE para todos os outros casos

A borda de descida não pode ser detectada mais do que uma vez durante o mesmo ciclo de execução, utilizando-se o operador FEDGE. Este operador pode ser utilizado para descrever a condição associada a uma transição SFC.
Advertência: A variável booleana "memo" utilizada para armazenar o último estado da expressão não pode ser utilizada como acionador de bordas para diferentes expressões.

Quando a expressão é uma variável booleana de nome "**xxx**", uma variável interna única de nome "**EDGE_xxx**" deve ser declarada e usada em expressões FEDGE para esta variável. Este método assegura que a variável de memória não será sobrescrita durante outras operações FEDGE.

Exemplo:

```
(* Programa ST utilizando operador FEDGE *)

(* este programa conta as bordas de descida de uma entrada booleana *)
(* Bi120 é uma variável booleana de entrada *)
(* Edge_Bi120 é a memória do estado da variável Bi120 *)

If FEDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Observação: este operador não está na norma IEC1131-3. Você pode preferir o uso do bloco padrão F_TRIG. Este operador foi mantido por razões de compatibilidade.

B.7.5 Comandos básicos ST

Os comando básicos da linguagem ST são:
- Atribuição
- Comando RETURN
- Estrutura IF-THEN-ELSIF-ELSE
- Comando CASE
- Comando de iteração WHILE
- Comando de iteração REPEAT
- Comando de iteração FOR
- Comando EXIT

Atribuição

Nome:	**:=**
Significado:	atribui uma variável a uma expressão
Sintaxe:	**<variável> := < expressão> ;**
Operandos:	variável deve ser interna ou de saída variável e expressão devem ter o mesmo tipo

A expressão pode ser uma chamada a um subprograma ou uma função da biblioteca ISaGRAF.

Exemplo:

```
(* Programa ST com atribuições *)

(* variável <<= variável *)
bo23 := bo10;

(* variável <<= expressão *)
bo56 := bx34 OR alrm100 & (level >= over_value);
```

```
result := (100 * input_value) / scale;

(* atribuição de valor de retorno de subprograma *)
rc := PSelect ( );

(* atribuição de chamada de função *)
limited_value := min (16, max (0, input_value) );
```

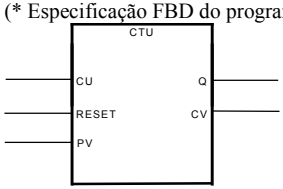
Comando RETURN

Nome: **RETURN**
Significado: termina a execução do programa corrente
Sintaxe: **RETURN ;**
Operandos: (nenhum)

Em um bloco SFC, o comando **return** indica o final da execução apenas daquele bloco.

Exemplo:

```
(* Especificação FBD do programa: contador programável *)
```




```
(* Implementação ST do programa, utilizando o comando RETURN  *)

If not (CU) then
        Q := false;
        CV := 0;
        RETURN; (* termina o programa *)
end_if;

if R then
        CV := 0;
else
        if (CV < PV) then
                CV := CV + 1;
        end_if;
end_if;
Q := (CV >= PV);
```

Comando IF-THEN-ELSIF-ELSE

Nome: **IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF**
Significado: executa uma de duas listas de comandos ST
seleção é feita de acordo com o valor de uma expressão booleana
Sintaxe: **IF <expressão_booleana> THEN**
<comando> ;

```
  < comando > ;
  ...
ELSIF <expressão_booleana> THEN
  < comando > ;
  < comando > ;
  ...
ELSE
  < comando > ;
  < comando > ;
  ...
END_IF;
```

Os comandos ELSE e ELSIF são opcionais. Se o comando ELSE não é utilizado, nenhuma instrução é executada quando a condição é FALSE.

Exemplo:

```
(* Programa ST utilizando comando IF *)

IF manual AND not (alarm) THEN
        level := manual_level;
        bx126 := bi12 OR bi45;
ELSIF over_mode THEN
        level := max_level;
ELSE
        level := (lv16 * 100) / scale;
END_IF;

(* estrutura If sem ELSE *)
If overflow THEN
          alarm_level := true;
END_IF;
```

Comando CASE

Nome: **CASE ... OF ... ELSE ... END_CASE**

Significado: executa uma entre várias listas de comandos ST
a seleção é feita de acordo com uma expressão inteira

Sintaxe:

```
CASE < expressão_inteira> OF
  <valor> : <comandos> ;
  <valor> , <valor> : <comandos> ;
  ...
ELSE
  <comandos> ;
END_CASE;
```

Os valores do Case devem ser expressões constantes inteiras. Diversos valores, separados por vírgulas, podem levar a uma mesma lista de comandos. O comando ELSE é opcional.

Exemplo:

```
(* Programa ST utilizando o comando CASE *)

CASE error_code OF
        255:    err_msg := 'Division by zero';
                fatal_error := TRUE;
        1:      err_msg := 'Overflow';
        2, 3:   err_msg := 'Bad sign';
ELSE
        err_msg := 'Unknown error';
END_CASE;
```

Comando WHILE

Nome: **WHILE ... DO ... END_WHILE**
Significado: estrutura de iteração para um grupo de comandos ST
a condição de "continuação" é avaliada ANTES de cada iteração
Sintaxe:

```
WHILE <expressão_booleana> DO
  <comando> ;
  <comando> ;
  ...
END_WHILE ;
```

Advertência: Como o ISaGRAF é um sistema **síncrono**, variáveis de entrada não são atualizadas durante as iterações WHILE. A mudança de estado de uma variável de entrada não pode ser utilizada para descrever a condição de um comando WHILE.

Exemplo:

```
(* Programa ST utilizando comando WHILE *)

(* este programa utiliza funções específicas "C" para a leitura de caracteres *)
(* em uma porta serial *)

string := ''; (* "string" vazia *)
nbchar := 0;

WHILE ((nbchar < 16) & ComIsReady ( )) DO
        string := string + ComGetChar ( );
        nbchar := nbchar + 1;
END_WHILE;
```

Comando REPEAT

Nome: **REPEAT ... UNTIL ... END_REPEAT**
Significado: estrutura de iteração para um grupo de comandos ST
a condição de "continuação" é avaliada DEPOIS de cada iteração
Sintaxe:

```
REPEAT
  <comando> ;
  <comando> ;
  ...
UNTIL <condição_booleana>
END_REPEAT ;
```

Advertência: Como o ISaGRAF é um sistema **síncrono**, variáveis de entrada não são atualizadas durante as iterações REPEAT. A mudança de estado de uma variável de entrada não pode ser utilizada para descrever a condição de um comando REPEAT.

Exemplo:

```
(* Programa ST utilizando comando REPEAT *)

(* este programa utiliza funções específicas "C" para a leitura de caracteres *)
(* em uma porta serial *)

string := ''; (* "string" vazia *)
nbchar := 0;
IF ComIsReady ( ) THEN
        REPEAT
                string := string + ComGetChar ( );
                nbchar := nbchar + 1;
        UNTIL ( (nbchar >= 16) OR NOT (ComIsReady ( )) )
        END_REPEAT;
END_IF;
```

Comando FOR

Nome:	**FOR ... TO ... BY ... DO ... END_FOR**	
Significado:	executa um número limitado de iterações utilizando uma variável analógica inteira como índice	
Sintaxe:	**FOR <índice> := <mini> TO <maxi> BY <etapa> DO** <comando> ; <comando> ; **END_FOR;**	
Operandos:	**índice:**	variável interna analógica incrementada a cada iteração
	mini:	valor inicial para o índice (antes da primeira iteração)
	maxi:	valor máximo permitido para o índice
	passo:	incremento do índice a cada iteração

A declaração [BY <passo>] é opcional. Se não especificado, o incremento padrão é 1.

Advertência: Como o ISaGRAF é um sistema **síncrono**, variáveis de entrada não são atualizadas durante as iterações FOR.

Este é o equivalente "while" de um comando FOR:

```
indice := mini;
while (indice <= maxi) do
  <comando> ;
  <comando> ;
  indice := indice + passo;
end_while;
```

Exemplo:

```
(* Programa ST utilizando comando FOR *)
(* este programa extrai os caracteres numéricos de uma “string” *)

length := mlen (message);
target := ''; (* “string” vazia *)
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code >= 48) & (code <= 57) THEN
                target := target + char (code);
        END_IF;
END_FOR;
```

Comando EXIT

Nome: **EXIT**
Significado: sai de um comando de iteração (loop) FOR, WHILE ou REPEAT
Sintaxe: **EXIT;**
Operandos: (nenhum)

O comando EXIT é comumente utilizado em um comando IF, dentro de um bloco FOR, WHILE ou REPEAT.

Exemplo:

```
(* Programa ST utilizando comando EXIT *)
(* este programa procura por um caractere em uma “string” *)

length := mlen (message);
found := NO;
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code  = searched_char) THEN
                found := YES;
                EXIT;
        END_IF;
END_FOR;
```

B.7.6 Extensões ST

A seguintes funções são extensões da linguagem ST:
- TSTART - TSTOP: controle de temporização

Os seguintes comandos e funções estão disponíveis para controlar a execução de programas secundários SFC. Eles podem ser utilizados dentro de blocos ACTION(): ... END_ACTION; em etapas SFC.

- GSTART — inicia um programa SFC
- GKILL — termina um programa SFC
- GFREEZE — congela um programa SFC

- GRST reinicia um programa SFC congelado
- GSTATUS obtém o estado corrente de um programa SFC

Advertência: Estas funções não estão na norma IEC 1131-3.
Equivalentes simples podem ser encontrados para GSTART e GKILL utilizando-se a seguinte sintaxe na etapa SFC:

```
child_name(S); (* equivalente a GSTART(child_name); *)
child_name(R); (* equivalente a GKILL(child_name); *)
```

Os seguintes campos podem ser utilizados para acessar o estado de uma etapa SFC:

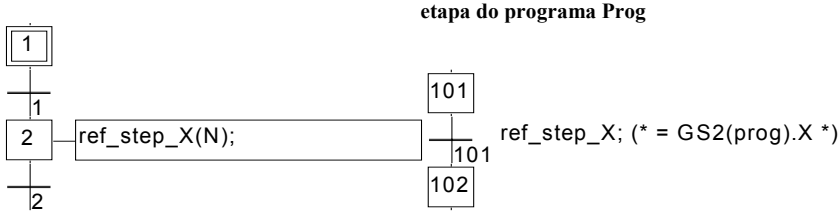
GSnnn.x valor booleano que representa a atividade de uma etapa
GSnnn.t tempo decorrido desde a última ativação da etapa
("**nnn**" é o número de referência da etapa)

Também é possível testar a atividade de uma etapa declarada em outro programa SFC, utilizando-se a seguinte sintaxe:

GSnnn(progname).x

Advertência: a referência a etapas em outro programa através desta sintaxe não faz parte da norma IEC 1131-3. Uma maneira fácil de fazer o mesmo respeitando as regras IEC é declarar uma variável booleana global no dicionário, que irá representar a atividade da etapa que se quer testar (por exemplo ref_step_X). Depois se insere a variável na etapa, com o qualificador N (ref_step_X(N);). Após isso, quando se desejar testar a atividade da etapa, pode-se utilizar esta variável.

Programa Prog **o outro programa que necessita da atividade da etapa do programa Prog**



Comando TSTART

Nome: **TSTART**
Significado: inicia a contagem de uma variável temporizador
o valor do temporizador não é modificado pelo comando TSTART, i.e. a contagem inicia a partir do valor corrente do temporizador.
Sintaxe: **TSTART (<variável_temporizador>);**
Operandos: qualquer variável temporizador inativa
Valor de retorno: (nenhum)

Exemplo:

```
(* Programa SFC utilizando comandos TSTART e TSTOP *)
```

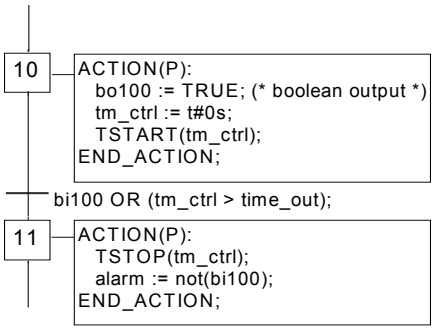
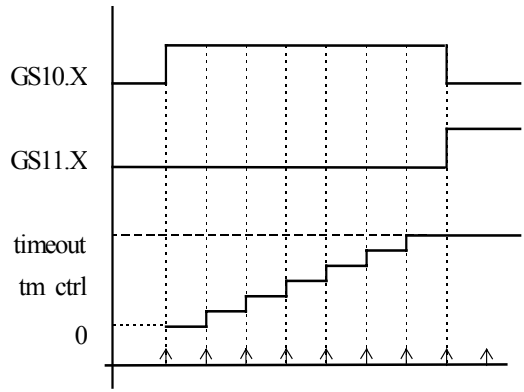


Diagrama de tempo se bi100 for sempre FALSE:



O temporizador mantém o mesmo valor durante um ciclo.

Comando TSTOP

Nome:	**TSTOP**
Significado:	pára de atualizar uma variável temporizador o valor do temporizador não é modificado pelo comando TSTOP
Sintaxe:	**TSTOP (<variável_temporizador>);**
Operandos:	qualquer variável temporizador ativa
Valor de retorno:	(nenhum)

Exemplo: Veja TSTART (a função descrita anteriormente)

Comando GSTART

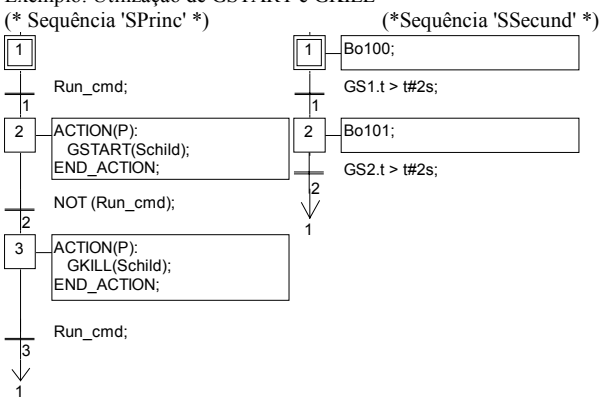
Nome:	**GSTART**
Significado:	inicia um programa SFC secundário colocando marcas de ativação em cada um de seus estados iniciais.
Sintaxe:	**GSTART (<programa_secundário>);**

Operandos: o programa especificado deve ser secundário em relação ao programa no qual o comando está escrito
Valor de retorno: (nenhum)

Programas secundários ao programa secundário sendo ativado não são iniciados automaticamente pelo comando GSTART.
Observação: Como GSTART não faz parte da norma IEC 1131-3, dê preferência à utilização do qualificador S, com a seguinte sintaxe, para iniciar um programa SFC secundário:

Child_name(S);

Exemplo: Utilização de GSTART e GKILL



Comando GKILL

Nome: **GKILL**
Significado: termina um programa SFC secundário, removendo as marcas de ativação existentes em suas etapas no momento
Sintaxe: **GKILL (<programa_secundário>);**
Operandos: o programa especificado deve ser secundário em relação ao programa no qual o comando está escrito
Valor de retorno: (nenhum)

Programas secundários ao programa secundário sendo terminado são automaticamente terminados pelo comando GKILL.
Observação: Como GKILL não faz parte da norma IEC 1131-3, dê preferência à utilização do qualificador R, com a seguinte sintaxe, para terminar um programa SFC secundário:

Child_name(R);

Exemplo: veja GSTART (função descrita anteriormente)

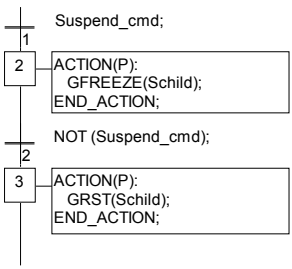
Comando GFREEZE

Nome: **GFREEZE**
Significado: Suspende a execução de um programa SFC secundário.
O programa suspenso pode ser reiniciado pelo comando GRST.
Sintaxe: **GFREEZE (<programa_secundário>);**

Operandos: o programa especificado deve ser secundário em relação ao programa no qual o comando está escrito
Valor de retorno: (nenhum)

Programas secundários ao programa sendo suspenso são automaticamente suspensos, também.
Observação: GFREEZE não faz parte da norma IEC 1131-3.

Exemplo:



Comando GRST

Nome: **GRST**
Significado: reinicia um programa SFC secundário suspenso pelo comando GFREEZE
Sintaxe: **GRST (<programa_secundário>);**
Operandos: o programa especificado deve sminer secundário em relação ao programa no qual o comando está escrito
Valor de retorno: (nenhum)

Programas secundários ao programa sendo suspenso são automaticamente reiniciados, também.
Observação: GRST não faz parte da norma IEC 1131-3.

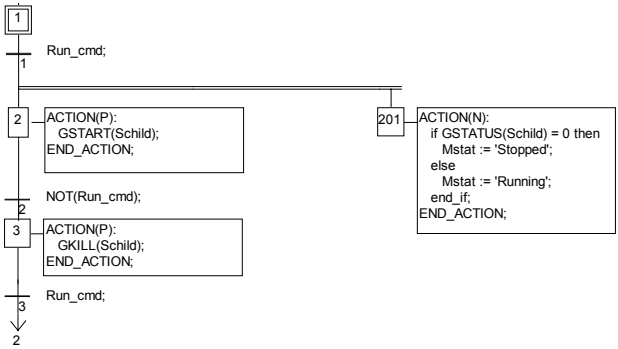
Exemplo: veja GFREEZE (função descrita anteriormente)

Comando GSTATUS

Nome: **GSTATUS**
Significado: retorna o estado de execução corrente de um programa SFC
Sintaxe: **<var_ana> := GSTATUS (<programa_secundário>);**
Operandos: o programa especificado deve ser secundário em relação ao programa no qual o comando está escrito
Valor de retorno: 0 = programa está inativo (terminado)
1 = programa está ativo (iniciado)
2 = programa está suspenso

Observação: GFREEZE não faz parte da norma IEC 1131-3.

Exemplo:


1
Run_cmd;
1
2
ACTION(P):
GSTART(Schild);
END_ACTION;
201
ACTION(N):
if GSTATUS(Schild) = 0 then
Mstat := 'Stopped';
else
Mstat := 'Running';
end_if;
END_ACTION;
NOT(Run_cmd);
2
3
ACTION(P):
GKILL(Schild);
END_ACTION;
Run_cmd;
3
2

B.8 Linguagem IL

A linguagem **IL** (**I**nstruction **L**ist, Lista de Instruções)é uma linguagem texto de baixo nível, particularmente adaptadas aos aplicativos de pequeno porte ou à otimização de partes reduzidas de um aplicativo. As instruções estão sempre relacionadas com o **resultado corrente** (ou **registro IL**). O operador indica o tipo de operação a efetuar entre o resultado corrente e o operando. O resultado da operação é armazenado novamente no resultado corrente.

B.8.1 Sintaxe da linguagem IL

Um programa IL é uma lista de **instruções** . Cada instrução deve começar em uma linha nova e deve conter um **operador**, complementado eventualmente pelos **modificadores** e, se necessário, por uma operação específica, um ou mais **operandos**, separado por vírgulas (**','**). Uma **etiqueta** seguida de dois pontos (**':'**) pode preceder a instrução. Se um **comentário** está atrelado à instrução, deve ser o último elemento da linha. Um comentário sempre começa pelos caracteres **'(*'** e termina por **'*)'**. Linhas vazias podem ser inseridas entre as instruções. Os Um comentário pode ser colocado em linhas vazias. A seguir, exemplos de linhas de instrução:

Etiqueta	*Operador*	*Operando*	*Comentário*
Start:	LD	IX1	(* botão de pressão *)
	ANDN	MX5	(* comando válido *)
	ST	QX2	(* ligar motor *)

Etiquetas

Uma **etiqueta** seguida por um caractere **':'** pode preceder uma instrução. Uma etiqueta pode ser colocada em uma linha vazia. As etiquetas são utilizadas como operandos em algumas instruções como os desvios. A nomenclatura das etiquetas deve respeitar as seguintes regras:
- o nome não pode exceder **16** caracteres
- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes podem ser **letras**, **números** ou o caractere sublinhado '_'

O mesmo nome não pode ser utilizado para mais de uma etiqueta no mesmo programa IL. Uma etiqueta pode ser o mesmo nome de uma variável.

Modificadores de operador

Os modificadores de operador disponíveis são mostrados a seguir. O caractere modificador deve completar o nome do operador sem nenhum caractere de separação (espaço) entre eles:

N inversão booleana do operando
(operação em espera
C operação condicional

O modificador **'N'** indica que o operando deve ser invertido antes de ser utilizado pela instrução. Por exemple, a instrução **ORN IX12** é interpretada como: **resultado := resultado OR NOT (IX12)**.

O modificador **abre parênteses '('** indica que a execução da instrução deve ser adiada até que o operador **fecha parênteses ')'** seja encontrado.

O modificador **'C'** indica que a instrução atrelada deve ser executada somente se o valor corrente tem o valor booleano TRUE (diferente de 0 para valores não booleanos). O modificador **'C'** pode ser combinado com o modificador **'N'** para indicar que a instrução deve ser executada somente se o resultado corrente for FALSE (ou 0 para valores não booleanos).

Operações em espera

Como existe apenas um registrador IL (resultado corrente), pode ser necessário colocar "em espera" algumas operações, de modo que a ordem de execução das instruções possa ser alterada. Parênteses são utilizados para indicar operações em espera:

'('	é um modificador	indica que a operação deve entrar em espera
')'	é um operador	executa a operação em espera

O sinal modificador "abre parênteses" **'('** indica que a execução da instrução deve ser adiada até que o operador "fecha parênteses" **')'** seja encontrado. Por exemplo, a seguinte seqüência:

```
AND(    IX12
OR      IX35
)
```

é interpretada como:

resultado := resultado AND (IX12 OR IX35)

B.8.2 Operadores IL

A seguinte tabela resume os operadores padrão da linguagem IL:

Operador	*Modificadores*	*Operando*	*Descrição*
LD	N	Variável, constante	Carrega operando
ST	N	Variável	Armazena o resultado corrente
S		Variável BOO	Atribui TRUE (set)
R		Variável BOO	Atribui FALSE (reset)
AND	N (	BOO	E booleano
&	N (	BOO	E booleano
OR	N (	BOO	OU booleano
XOR	N (	BOO	OU exclusivo
ADD	(	variável, constante	Adição
SUB	(	variável, constante	Subtração
MUL	(	variável, constante	Multiplicação
DIV	(	variável, constante	Divisão
GT	(	variável, constante	Comparação: >
GE	(	variável, constante	Comparação: >=
EQ	(	variável, constante	Comparação: =
LE	(	variável, constante	Comparação: <=
LT	(	variável, constante	Comparação: <
NE	(	variável, constante	Comparação: <>

CAL JMP RET	C N C N C N	Nome da instância FB Etiqueta	Chama um bloco de função Desvia para a etiqueta Retorna de subprograma
)			Executa operação em espera

Nas seções seguintes, somente os operadores específicos à linguagem IL são descritos. Outros operadores podem ser encontrados na seção "operadores padrão, blocos de função e funções".

Operador LD

Operação carrega um valor no resultado corrente
Modificadores permitidos N
Operando expressão constante
variável interna, de entrada ou de saída

Exemplo:

```
                    (*EXEMPLOS DE OPERAÇÕES LD *)
LDex:       LD      false       (* resultado := constante booleana FALSE *)
            LD      true        (*resultado := constante booleana TRUE *)
            LD      123         (*resultado := constante inteira *)
            LD      123.1       (*resultado := constante real *)
            LD      t#3ms       (*resultado := constante de temporização *)
            LD      boo_var1    (*resultado := variável booleana *)
            LD      ana_var1    (*resultado := variável analógica *)
            LD      tmr_var1    (*resultado := variável de temporização *)
            LDN     boo_var2    (*resultado := NOT ( variável booleana) *)
```

Operador ST

Operação armazena o resultado corrente em uma variável
o resultado corrente não é modificado por esta operação
Modificadores permitidos N
Operando variável interna ou de saída

Exemplo:

```
                    (*EXEMPLOS DE OPERAÇÕES ST *)
STboo:      LD      false
            ST      boo_var1    (* boo_var1 := FALSE *)
            STN     boo_var2    (* boo_var2 := TRUE *)
STana:      LD      123
            ST      ana_var1    (* ana_var1 := 123 *)
STtmr:      LD      t#12s
            ST      tmr_var1    (* tmr_var1 := t#12s *)
```

Operador S

Operação: armazena o valor booleano TRUE em uma variável booleana, se o resultado corrente tem o valor booleano TRUE. Nenhuma operação é processada se o resultado corrente é FALSE. O resultado corrente não é modificado por esta operação

Modificadores permitidos: (nenhum)
Operando: variável booleana de saída ou interna

Exemplo:

```
                    (*EXEMPLOS DE OPERAÇÕES S *)
SETex:      LD      true        (*resultado corrente:= TRUE *)
            S       boo_var1    (* boo_var1 := TRUE *)
                                (*resultado corrente não é modificado *)
            LD      false       (*resultado corrente := FALSE *)
            S       boo_var1    (*nada é feito - boo_var1 inalterada *)
```

Operador R

Operação armazena o valor booleano FALSE em uma variável booleana, se o resultado corrente tem o valor booleano TRUE. Nenhuma operação é processada se o resultado corrente é FALSE. O resultado corrente não é modificado por esta operação
Modificadores permitidos (nenhum)
Operando variável booleana de saída ou interna

Exemplo:

```
                    (*EXEMPLOS DE OPERAÇÕES R *)
RESETex:    LD      true        (* resultado corrente := TRUE *)
            R       boo_var1    (* boo_var1 := FALSE *)
                                (*resultado corrente não é modificado *)
            ST      boo_var2    (* boo_var2 := TRUE *)
            LD      false       (*resultado corrente := FALSE *)
            R       boo_var1    (*nada é feito - boo_var1 inalterada*)
```

Operador JMP

Operação desvia para uma etiqueta especificada
Modificadores permitidos C N
Operando etiqueta definida no mesmo programa IL

Exemplo:

```
(* o seguinte exemplo testa o valor de um seletor analógico (0 ou 1 ou 2)
(* para ligar uma dentre 2 saídas booleanas. A comparação "é igual a zero 0" é feita com
(* o operador JMPC                                                    *)

JMPex:      LD      selector    (* selector = 0 ou 1 ou 2 *)
            BOO                 (* conversão para booleano *)
            JMPC    test1       (* se selector = 0 então *)
            LD      true
            ST      bo0         (* bo0 := true *)
            JMP     JMPend      (* fim de programa *)
test1:      LD      selector
            SUB     1           (* decrementa selector:  agora é 0 ou 1 *)
            BOO                 (* conversão para booleano *)
```

```
            JMPC    test2       (* se selector = 0 então *)
            LD      true
            ST      bo1         (* bo1 := true *)
            JMP     JMPend      (* fim de programa *)
test2:      LD      true        (* última possibilidade *)
            ST      bo2         (* bo2 := true *)
JMPend:                         (* fim do programa IL *)
```

Operador RET

Operação termina a lista de instruções corrente. Se a IL é um subprograma, o resultado corrente é retornado ao programa que o chamou.

Modificadores permitidos C N

Operando (nenhum)

Exemplo:

```
(* o seguinte exemplo testa o valor de um seletor analógico (0 ou 1 ou 2)
(* para ligar uma dentre 2 saídas booleanas. A comparação "é igual a zero 0" é feita com
(* o operador JMPC

JMPex:      LD      selector    (* selector = 0 ou 1 ou 2 *)
            BOO                 (* conversão para booleano *)
            JMPC    test1       (* se selector = 0 então *)
            LD      true
            ST      bo0         (* bo0 := true *)
            RET                 (* fim - retorne 0 *)
                                (* decremente seletor *)
test1:      LD      selector
            SUB     1           (* selector = 0 ou 1 *)
            BOO                 (* conversão para booleano *)
            JMPC    test2       (* se selector = 0 então *)
            LD      true
            ST      bo1         (* bo1 := true *)
            LD      1           (* carrega valor real de selector *)
            RET                 (* fim - retorna 1 *)
                                (* última possibilidade *)
test2:      RETNC               (* retorna se selector tem *)
                                (* um valor inválido *)
            LD      true
            ST      bo2         (* bo2 := true *)
            LD      2           (* carrega valor real de selector *)
                                (* fim - retorna 2 *)
```

Operador ")"

Operação executa uma operação em espera. A operação em espera foi marcada por um '('

Modificadores permitidos (nenhum)

Operando (nenhum)

Exemplo:

```
(* O seguinte programa intercala operações em espera*)
(* res := a1 + (a2 * (a3 - a4) * a5) + a6; *)

Delayed:    LD      a1      (* resultado := a1; *)
            ADD(    a2      (* em espera ADD - resultado := a2; *)
            MUL(    a3      (* em espera MUL - resultado := a3; *)
            SUB     a4      (* resultado := a3 - a4; *)
            )               (* executado MUL em espera - resultado := a2 * (a3-a4); *)
            MUL     a5      (* resultado := a2 * (a3 - a4) * a5; *)
            )               (* executado ADD em espera *)
                            (* resultado := a1 + (a2 * (a3 - a4) * a5); *)
            ADD     a6      (* resultado := a1 + (a2 * (a3 - a4) * a5) + a6; *)
            ST      res     (* carrega resultado corrente na variável res *)
```

Chamando subprogramas ou funções

Um subprograma ou uma função (escrito em linguagem IL, ST, LD, FBD ou "C") é chamado a partir da linguagem IL com o uso de seu nome como operador.

Operação executa um subprograma ou função - o valor retornado pelo subprograma ou função é armazenado no resultado corrente da IL

Modificadores permitidos (nenhum)

Operando O primeiro parâmetro de chamada deve estar armazenado no resultado corrente antes da chamada. Os seguintes são expressos no campo "operando", separados por vírgulas.

Exemplo:

```
(* Programa que chama : converte um valor analógico em um valor de tempo *)

Main:       LD      bi0
            SUBPRO  bi1,bi2 (* chama subprograma para obter valor analógico *)
            ST      result  (* result := valor retornado pelo subprograma *)
            GT      vmax    (* teste de transbordo *)
            RETC            (* retorna se transbordo *)
            LD      result
            MUL     1000    (* converte segundos em milisegundos *)
            TMR             (* converte em temporizador *)
            ST      tmval   (* carrega valor convertido em variável temporizador
*)

(* Subprograma chamado, 'SUBPRO' , processa o valor analógico *)
(* dado com um valor binário em três variáveis de entrada: in0, in1, in2. São três os parâmetros de
entrada booleana do subprograma *)
            LD      in2
            ANA             (* result = ana (in2); *)
            MUL     2       (* result := 2*ana (in2); *)
            ST      temporary (* temporary := result *)
            LD      in1
            ANA
```

```
ADD     temporary   (* result := 2*ana (in2) + ana (in1); *)
MUL     2           (* result := 4*ana (in2) + 2*ana (in1); *)
ST      temporary   (* temporary := result *)
LD      in0
ANA
ADD     temporary   (* result := 4*ana (in2) + 2*ana (in1)+ana (in0); *)
ST      SUBPRO      (* retorna resultado corrente *)
```

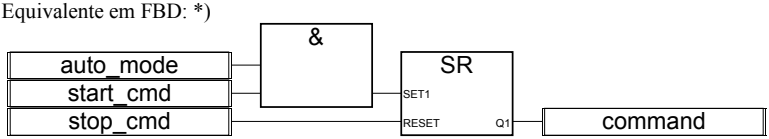
Chamando blocos de função: operador CAL

Operação chama um bloco de função
Modificadores permitidos C N
Operando Nome da instância do bloco de função.
Os parâmetros de entrada dos blocos devem ser atribuídos antes da chamada, com o uso sequências de operações LD/ST.
Parâmetros de saída são conhecidos se utilizados.

Exemplo1:

```
(* Chamando bloco de função SR: SR1 é uma instância de SR *)
LD      auto_mode
AND     start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

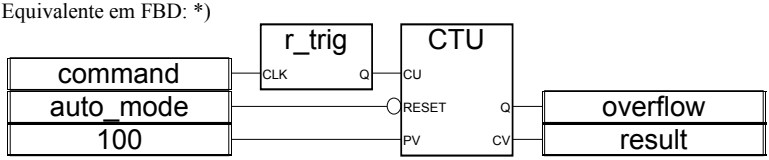
(* Equivalente em FBD: *)



Exemplo 2
(* Supomos que R_TRIG1 seja uma instância do bloco R_TRIG e CTU1 seja uma instância do bloco CTU *)

```
LD      command
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
```

```
LD      CTU1.cv
ST      result

(* Equivalente em FBD: *)
```



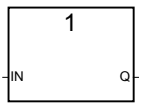
B.9 Operadores padrão, blocos de função e funções

B.9.1 Operadores padrão

A seguir, os operadores padrão da linguagem IEC.

Manipulação de dados	Atribuição, negação analógica
Operações booleanas	E booleano
	OU booleana
	OU exclusivo booleano
Operações aritméticas	Adição
	Subtração
	Multiplicação
	Divisão
Operações lógicas	Máscara E analógica bit a bit
	Máscara OU analógica bit a bit
	Máscara OU exclusive bit a bit
	Negação bit a bit
Testes de comparação	Menor que
	Menor ou igual a
	Maior que
	Maior ou igual a
	Igual a
	Diferente de
Conversão de dados	Conversão para Booleano
	Conversão para Analógico Inteiro
	Conversão para Analógico Real
	Conversão para Temporização
	Conversão para Mensagem
Outros	Concatenação de mensagem
	Acesso aos parâmetros do sistema
	Opera canal de E/S

1 gain



Argumentos:

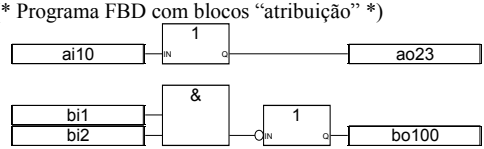
IN	qualquer tipo
Q	qualquer tipo

Descrição:

atribuição de uma variável em outra

Este bloco é muito útil para a ligação direta de uma entrada do diagrama com uma saída do diagrama. Também pode ser utilizada (com uma linha de inversão booleana) para inverter o estado de uma linha conectada a uma saída do diagrama.

(* Programa FBD com blocos "atribuição" *)



(* equivalência ST: *)
ao23 := ai10;
bo100 := NOT (bi1 AND bi2);

(* equivalência IL: *)

LD	ai10
ST	ao23
LD	bi1
AND	bi2
STN	bo100

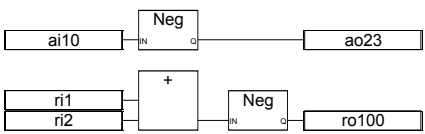
NEG

Argumentos:

IN	INT-REAL	Entrada and saída devem ter o mesmo formato
Q	INT-REAL	

Descrição:
Atribuição da negação de uma variável.

(* Programa FBD com blocos de negação *)



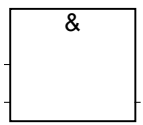
(* equivalência ST: *)
ao23 := - (ai10);
ro100 := - (ri1 + ri2);

(* equivalência IL: *)

LD	ai10

```
MUL     -1
ST      ao23
LD      ri1
ADD     ri2
MUL     -1.0
ST      ro100
```

& AND



Nota: Para este operador, o número de entradas pode ser ampliado para mais de duas.

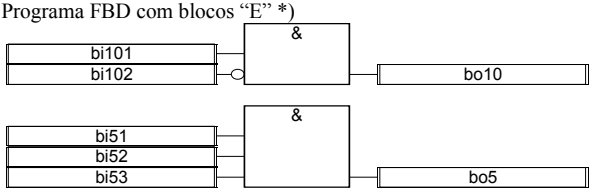
Argumentos:

(entradas)	BOOLEAN	
saída	BOOLEAN	E booleano entre os termos de entrada

Descrição:

E booleano entre dois ou mais termos.

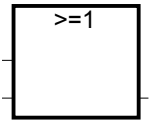
(* Programa FBD com blocos “E” *)




```
(* equivalência ST: *)
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

```
(* equivalência IL *)
        LD      bi101   (* resultado corrente := bi101 *)
        ANDN    bi102   (* resultado corrente := bi101 AND not (bi102) *)
        ST      bo10    (* bo10 := resultado corrente *)
        LD      bi51    (* resultado corrente := bi51;
        &       bi52    (* resultado corrente := bi51 AND bi52 *)
        &       bi53    (* resultado corrente := (bi51 AND bi52) AND bi53 *)
        ST      bo5     (* bo5 := resultado corrente *)
```

>=1 OR



Nota: Para este operador, o número de entradas pode ser ampliado para mais de duas.

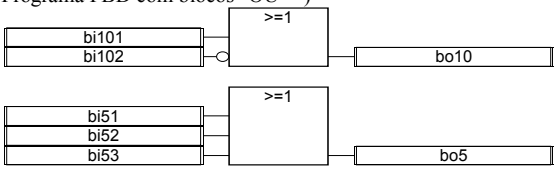
Argumentos:

(entradas)	BOOLEAN	
saída	BOOLEAN	OU booleano entre os termos de entrada

Descrição:

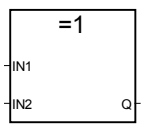
OU booleano entre dois ou mais termos.

(* Programa FBD com blocos "OU" *)




```
(* equivalência ST: *)
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;

(* equivalência IL: *)
LD      bi101
ORN     bi102
ST      bo10
LD      bi51
OR      bi52
OR      bi53
ST      bo5
```

=1 XOR



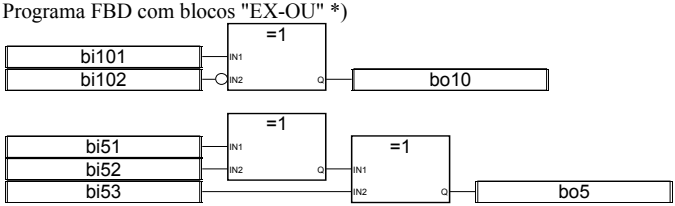
Argumentos:

IN1	BOOLEAN	
IN2	BOOLEAN	
Q	BOOLEAN	OU exclusivo booleano entre 2 termos de entrada

Descrição:

OU exclusivo booleano entre 2 termos.

(* Programa FBD com blocos "EX-OU" *)

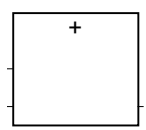


(* equivalência ST: *)
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;

(* equivalência IL: *)

LD	bi101
XORN	bi102
ST	bo10
LD	bi51
XOR	bi52
XOR	bi53
ST	bo5

+



Nota: Para este operador, o número de entradas pode ser ampliado para mais de duas.

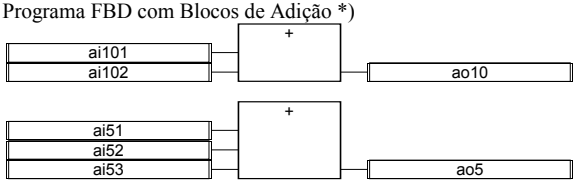
Argumentos:

(entradas)	INT-REAL	pode ser INTEIRO ou REAL (todas as entradas devem ter o mesmo formato)
saída	INT-REAL	adição com sinal dos termos de entrada

Descrição:

Adição de duas ou mais variáveis analógicas.

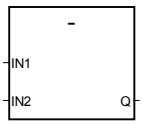
(* Programa FBD com Blocos de Adição *)



(* equivalência ST: *)

```
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;

(* equivalência IL: *)
LD          ai101
ADD         ai102
ST          ao10
LD          ai51
ADD         ai52
ADD         ai53
ST          ao5
```

-



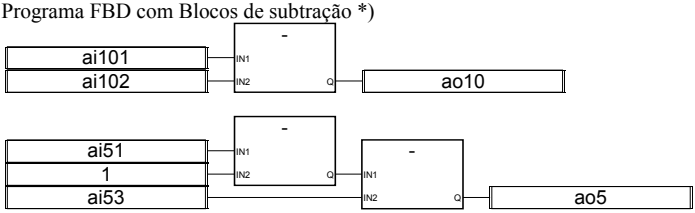
Argumentos:

IN1	INT-REAL	pode ser INTEIRO ou REAL
IN2	INT-REAL	(IN1 e IN2 devem ter o mesmo formato)
Q	INT-REAL	subtração (primeiro - segundo)

Descrição:

Subtração de duas variáveis analógicas (primeiro - segundo).

(* Programa FBD com Blocos de subtração *)




```
 (* equivalência ST: *)
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;

(* equivalência IL: *)
LD          ai101
SUB         ai102
ST          ao10
LD          ai51
SUB         1
SUB         ai53
ST          ao5
```

*

Nota: Para este operador, o número de entradas pode ser ampliado para mais de duas.

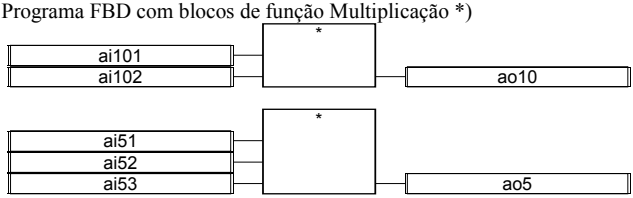
Argumentos:

(entradas)	INT-REAL	pode ser INTEIRO ou REAL (todas as entradas devem ter o mesmo formato)
saída	INT-REAL	multiplicação com sinal dos termos de entrada

Descrição:

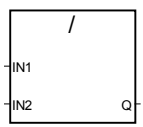
Multiplicação de duas ou mais variáveis analógicas.

(* Programa FBD com blocos de função Multiplicação *)




```
(* equivalência ST *)
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;

(*equivalência IL: *)
LD      ai101
MUL     ai102
ST      ao10
LD      ai51
MUL     ai52
MUL     ai53
ST      ao5
```

/



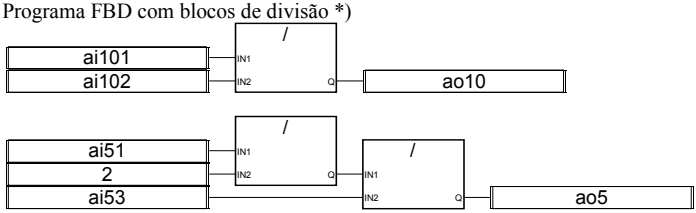
Argumentos:

IN1	INT-REAL	pode ser INTEIRO ou REAL (operando)
IN2	INT-REAL	valor analógico diferente de zero (divisor)

		(IN1 e IN2 devem ter o mesmo formato)
Q	INT-REAL	divisão com sinal do inteiro ou real IN1 por IN2

Descrição:
Divisão de duas variáveis analógicas (a primeira dividida pela segunda).

(* Programa FBD com blocos de divisão *)

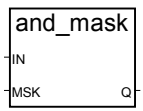


(* equivalência ST: *)
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;

(* equivalência IL: *)

LD	ai101
DIV	ai102
ST	ao10
LD	ai51
DIV	2
DIV	ai53
ST	ao5

AND_MASK

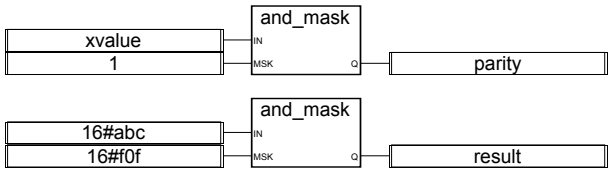


Argumentos:

IN	INT	deve ter o formato inteiro
MSK	INT	deve ter o formato inteiro
Q	INT	E lógico bit a bit entre IN e MSK

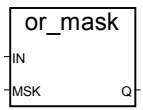
Descrição:
E analógico inteiro máscara bit a bit.

(* Programa FBD com blocos analógicos AND_MASK *)




```
(* equivalência ST: *)
parity := AND_MASK (xvalue, 1); (* 1 se o valor é ímpar *)
result := AND_MASK (16#abc, 16#f0f); (* igual a 16#a0c *)

(* equivalência IL: *)
LD          xvalue
AND_MASK    1
ST          parity
LD          16#abc
AND_MASK    16#f0f
ST          result
```

OR_MASK



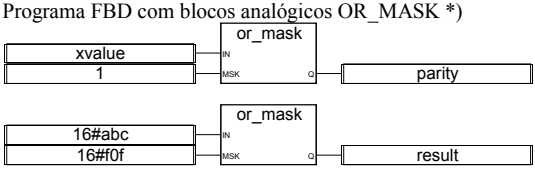
Argumentos:

IN	INT	deve ter o formato inteiro
MSK	INT	deve ter o formato inteiro
Q	INT	OU lógico bit a bit entre IN e MSK

Descrição:

OU analógico inteiro máscara bit a bit.

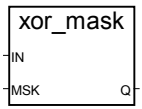
(* Programa FBD com blocos analógicos OR_MASK *)




```
(* equivalência ST: *)
is_odd := OR_MASK (xvalue, 1); (* sempre ímpar *)
result := OR_MASK (16#abc, 16#f0f); (* igual 16#fbf *)

(* equivalência IL: *)
LD          xvalue
OR_MASK     1
ST          is_odd
```

```
LD          16#abc
OR_MASK     16#f0f
ST          result
```

XOR_MASK



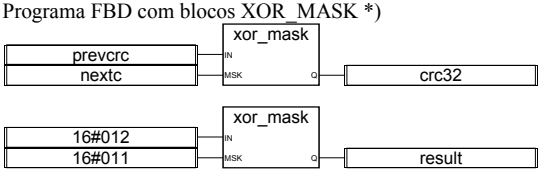
Argumentos:

IN	INT	deve ter o formato inteiro
MSK	INT	deve ter o formato inteiro
Q	INT	EX-OU lógico bit a bit entre IN e MSK

Descrição:

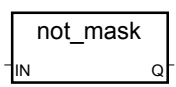
EX-OU analógico inteiro máscara bit a bit.

(* Programa FBD com blocos XOR_MASK *)




```
(* equivalência ST: *)
crc32 := XOR_MASK (prevcrc, nextc);
result := XOR_MASK (16#012, 16#011); (* igual a 16#003 *)

(* equivalência IL: *)
LD          prevcrc
XOR_MASK    nextc
ST          crc32
LD          16#012
XOR_MASK    16#011
ST          result
```

NOT_MASK



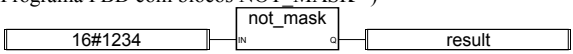
Argumentos :

IN	INT	deve ter o formato inteiro
Q	INT	negação lógica bit a bit sobre os 32 bits de IN

Descrição:

Negação analógica inteira máscara bit a bit.

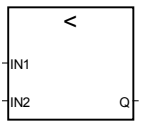
(* Programa FBD com blocos NOT_MASK *)



(equivalência *ST: *)
result := NOT_MASK (16#1234);
(* result = 16#FFFF_EDCB *)

(* equivalência IL: *)
LD 16#1234
NOT_MASK
ST result

<



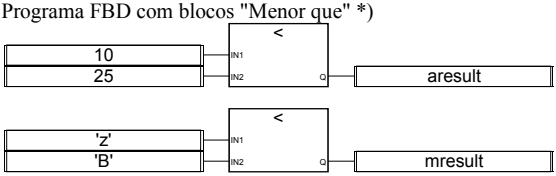
Argumentos:

IN1	INT-REAL-TMR-MSG	
IN2	INT-REAL-TMR-MSG	as duas entradas devem ser do mesmo tipo
Q	BOOLEAN	TRUE se IN1 < IN2

Descrição:

Verifica se um valor é MENOR QUE outro (sobre os analógicos, de temporização ou as mensagens)

(* Programa FBD com blocos "Menor que" *)

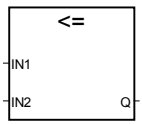


(* equivalência ST: *)
aresult := (10 < 25); (* aresult = TRUE *)
mresult := ('z' < 'B'); (* mresult = FALSE *)

(* equivalência IL: *)
LD 10
LT 25
ST aresult
LD 'z'

```
LT      'B'
ST      mresult
```

<=



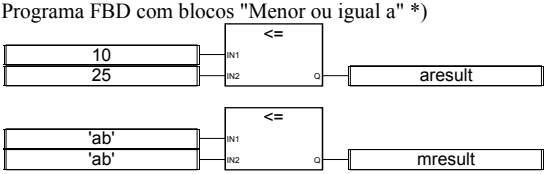
Argumentos:

IN1	INT-REAL-MSG	
IN2	INT-REAL-MSG	as duas entradas devem ser do mesmo tipo
Q	BOOLEAN	TRUE if IN1 <= IN2

Descrição:

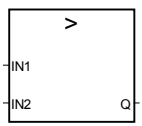
Verifica se um valor é MENOR OU IGUAL A outro (sobre os analógicos ou as mensagens)

(* Programa FBD com blocos "Menor ou igual a" *)




```
(* equivalência ST: *)
aresult := (10 <= 25); (* aresult = TRUE *)
mresult := ('ab' <= 'ab'); (* mresult = TRUE *)

(* equivalência IL: *)
LD      10
LE      25
ST      aresult
LD      'ab'
LE      'ab'
ST      mresult
```

>



Argumentos:

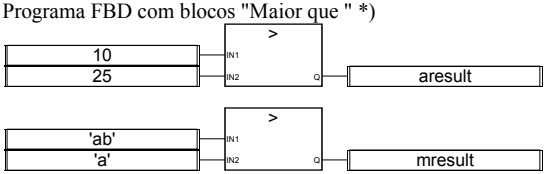
IN1	INT-REAL-TMR-MSG

IN2	INT-REAL-TMR-MSG	as duas entradas devem ser do mesmo tipo
Q	BOOLEAN	TRUE if IN1 > IN2

Descrição:

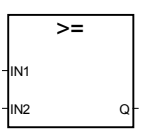
Verifica se um valor é MAIOR QUE outro (sobre os analógicos, de temporização ou de mensagens)

(* Programa FBD com blocos "Maior que " *)




```
(* equivalência ST: *)
aresult := (10 > 25); (* aresult = FALSE *)
mresult := ('ab' > 'a'); (* mresult = TRUE *)

(* equivalência IL: *)
LD      10
GT      25
ST      aresult
LD      'ab'
GT      'a'
ST      mresult
```

>=



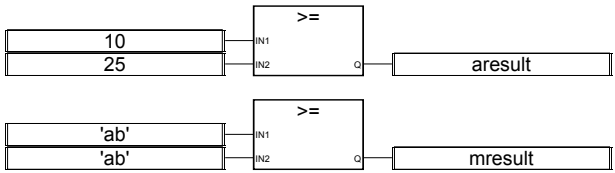
Argumentos:

IN1	INT-REAL-MSG	
IN2	INT-REAL-MSG	as duas entradas devem ser do mesmo tipo
Q	BOOLEAN	TRUE se IN1 >= IN2

Descrição:

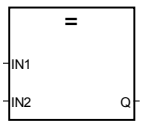
Verifica se um valor é MAIOR OU IGUAL A outro (sobre os analógicos ou as mensagens)

(* Programa FBD com blocos "Maior ou Igual a" *)




```
(* equivalência ST: *)
aresult := (10 >= 25); (* aresult = FALSE *)
mresult := ('ab' >= 'ab'); (* mresult = TRUE *)

(* equivalência IL: *)
LD          10
GE          25
ST          aresult
LD          'ab'
GE          'ab'
ST          mresult
```

=



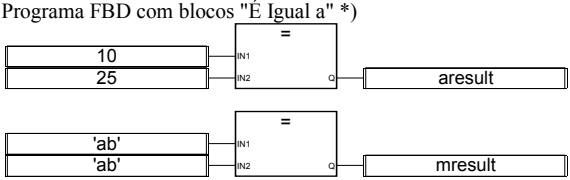
Argumentos:

IN1	INT-REAL-MSG	
IN2	INT-REAL-MSG	as duas entradas devem ser do mesmo tipo
Q	BOOLEAN	TRUE se IN1 = IN2

Descrição:

Verifica se um valor É IGUAL A outro (sobre os analógicos ou as mensagens)

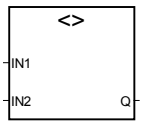
(* Programa FBD com blocos "É Igual a" *)




```
(* equivalência ST: *)
aresult := (10 = 25); (* aresult =s FALSE *)
mresult := ('ab' = 'ab'); (* mresult = TRUE *)

(* equivalência IL: *)
LD          10
EQ          25
```

```
ST      aresult
LD      'ab'
EQ      'ab'
ST      mresult
```

<>



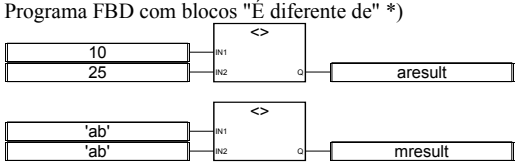
Argumentos:

IN1	INT-REAL-MSG	
IN2	INT-REAL-MSG	as duas entradas devem ser do mesmo tipo
Q	BOOLEAN	TRUE se IN1 <> IN2

Descrição:

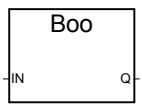
Verifica se um valor É DIFERENTE DE (Not Equal) outro (sobre os analógicos ou as mensagens)

(* Programa FBD com blocos "É diferente de" *)




```
(* equivalência ST: *)
aresult := (10 <> 25); (* aresult = TRUE *)
mresult := ('ab' <> 'ab'); (* mresult = FALSE *)

(* equivalência IL: *)
LD      10
NE      25
ST      aresult
LD      'ab'
NE      'ab'
ST      mresult
```

BOO



Argumentos:

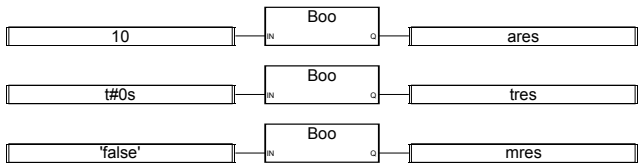
IN	ANY	qualquer valor não booleano

Q	BOO	TRUE para valor numérico diferente de zero FALSE para valor numérico igual a zero TRUE para mensagem 'TRUE' FALSE para mensagem 'FALSE'

Descrição:

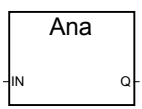
Converte qualquer variável em uma booleana

(* Programa FBD com blocos "Converte em Booleano" *)




```
(* equivalência ST: *)
ares := BOO (10);                    (* ares = TRUE *)
tres := BOO (t#0s);                  (* tres = FALSE *)
mres := BOO ('false');               (* mres = FALSE *)

(* equivalência IL: *)
LD          10
BOO
ST          ares
LD          t#0s
BOO
ST          tres
LD          'false'
BOO
ST          mres
```

ANA



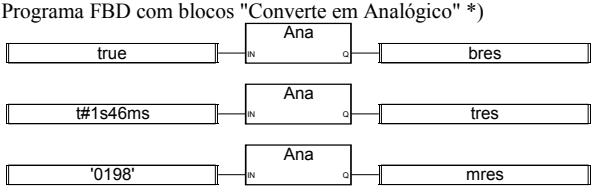
Argumentos:

IN	ANY	qualquer valor analógico não inteiro
Q	INT	0 se IN é FALSE / 1 se IN é TRUE número de milisegundos para um temporizador parte inteira para um analógico real número decimal representado por uma “string”

Descrição:

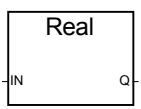
Converte qualquer variável em uma variável inteira

(* Programa FBD com blocos "Converte em Analógico" *)




```
(* equivalência ST: *)
bres := ANA (true);                (* bres = 1 *)
tres := ANA (t#1s46ms);            (* tres = 1046 *)
mres := ANA ('0198');              (* mres = 198 *)

(* equivalência IL: *)
LD      true
ANA
ST      bres
LD      t#1s46ms
ANA
ST      tres
LD      '0198'
ANA
ST      mres
```

REAL



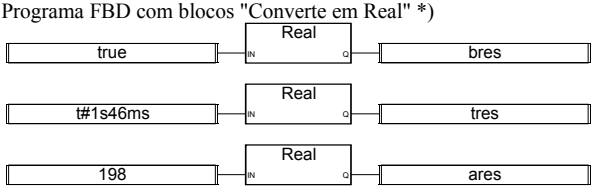
Argumentos:

IN	BOO-INT-TMR	qualquer valor analógico não real (sem mensagem) 0.0 se IN = FALSE / 1.0 se IN = TRUE número de milisegundos para uma temporização
Q	REAL	número equivalente para inteiro analógico

Descrição:

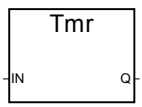
Converte qualquer variável em uma variável real

(* Programa FBD com blocos "Converte em Real" *)



```
(* equivalência ST: *)
bres := REAL (true);                (* bres = 1.0 *)
tres := REAL (t#1s46ms);            (* tres = 1046.0 *)
ares := REAL (198);                 (* ares = 198.0 *)

(* equivalência IL: *)
LD      true
REAL
ST      bres
LD      t#1s46ms
REAL
ST      tres
LD      198
REAL
ST      ares
```

TMR



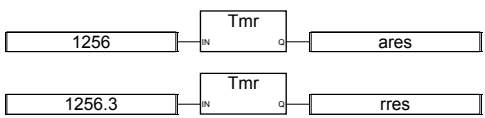
Argumentos:

IN	INT-REAL	qualquer valor diferente de temporização IN (ou a parte inteira de IN se é real) é o número de milisegundos
Q	TIMER	tempo representado por IN

Descrição:

Converte qualquer variável analógica em variável de temporização

(* Programa FBD com blocos "TMR" *)




```
 (* equivalência ST: *)
ares := TMR (1256);                 (* ares := t#1s256ms *)
rres := TMR (1256.3);               (*rres := t#1s256ms *)

(* equivalência IL: *)
LD      1256
TMR
ST      ares
LD      1256.3
TMR
ST      rres
```

MSG

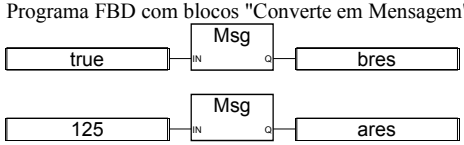
Argumentos:

IN	BOO-	
	INT-REA	qualquer valor diferente de mensagem
Q	MSG	'false' ou 'true' se IN é um booleano representação decimal se IN é um analógico

Descrição:

Converte qualquer variável em uma variável de mensagem

(* Programa FBD com blocos "Converte em Mensagem" *)

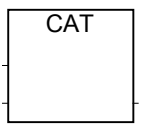


(* equivalência ST: *)
bres := MSG (true); (* bres = 'TRUE' *)
ares := MSG (125); (* ares = '125' *)

(* equivalência IL: *)

```
LD      true
MSG
ST      bres
LD      125
MSG
ST      ares
```

CAT



Nota: Para este operador, o número de suas entradas pode ser ampliado para mais do que duas.

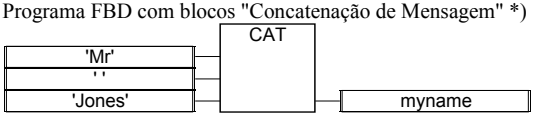
Argumentos:

(entrada)	MSG	(a soma dos comprimentos das mensagens de entrada não deve exceder a capacidade da mensagem de saída)
saída	MSG	concatenação de mensagens de entrada

Descrição:

Concatena diversas mensagens em uma só

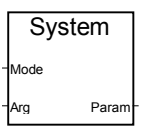
(* Programa FBD com blocos "Concatenação de Mensagem" *)



(* equivalência ST: utilização do operador + *)
myname := ('Mr' + ' ') + 'Jones';
(* significa: myname := 'Mr Jones' *)

(* equivalência IL: *)
LD 'Mr'
ADD ' '
ADD 'Jones'
ST myname

SYSTEM



Argumentos:

Mode	INT	identifica os parâmetros do sistema e o modo de acesso
Arg	INT-TMR	novo valor no caso de um acesso de "escrita"
Param	INT	valor do parâmetro acessado

Descrição:
Acesso aos parâmetros do sistema

A seguir, a lista de comandos disponíveis (palavras-chaves predefinidas para a função SYSTEM:

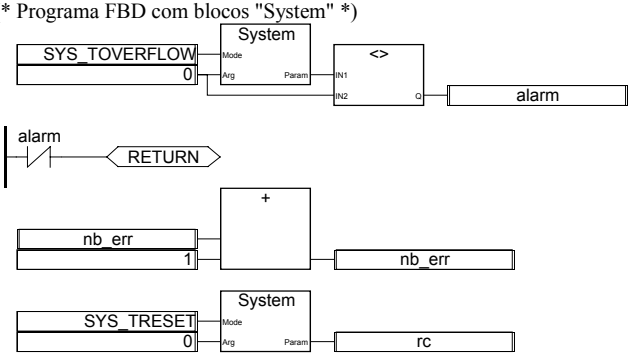
Comando	Significado
SYS_TALLOWED	leitura do tempo de ciclo programado
SYS_TCURRENT	leitura do tempo de ciclo corrente
SYS_TMAXIMUM	leitura do tempo de ciclo máximo
SYS_TOVERFLOW	leitura do transbordo do tempo de ciclo
SYS_TRESET	zera os contadores (reset) de tempo
SYS_TWRITE	escritura do ciclo de tempo
SYS_ERR_TEST	verifica os erros de tempo de execução
SYS_ERR_READ	leitura do último erro de tempo de execução

Estes são os argumentos esperados para as funções predefinidas da função SYSTEM:

Comando	Argumento	Valor de retorno

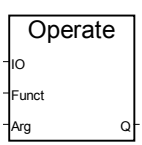
SYS_TALLOWED	0	tempo de ciclo autorizado
SYS_TCURRENT	0	tempo de ciclo corrente
SYS_TMAXIMUM	0	tempo de ciclo máximo detectado
SYS_TOVERFLOW	0	número de transbordos de tempo
SYS_TRESET	0	0
SYS_TWRITE	novo t. de ciclo autorizado	valor escrito
SYS_ERR_TEST	0	0 se nenhum erro detectado
SYS_ERR_READ	0	código do último erro

(* Programa FBD com blocos "System" *)




```
(* equivalência ST: *)
alarm := (SYSTEM (SYS_TOVERFLOW, 0) <> 0);
If (alarm) Then
        nb_err := nb_err + 1;
        rc := SYSTEM (SYS_TRESET, 0);
End_If;
```

OPERATE



Argumentos:

IO	Todo tipo	Variável de entrada ou de saída
Funct	ENT	ação a ser efetuada
Arg	ENT	argumento para a ação de E/S efetuada
Q	ENT	verificação de retorno

Descrição:

Acesso a um canal de E/S

O significado dos argumentos OPERATE depende da implementação da interface de E/S. Veja o manual do seu hardware ou a observação técnica da sua placa de E/S correspondente para saber mais sobre as capacidades do OPERATE.

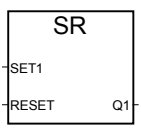
B.9.2 Blocos de função padrão

Estes são os blocos de função padrão suportados pelo sistema ISaGRAF. Tais blocos de função são predefinidos e não têm que ser declarados na biblioteca.

Booleanos	SR	Biestável - dominante TRUE
	RS	Biestável - dominante FALSE
	R_Trig	Detecção de borda de subida
	F_Trig	Detecção de borda de descida
	SEMA	Semáforo
Contadores	CTU	Contador para cima (incremento)
	CTD	Contador para baixo (decremento)
	CTUD	Contador para cima/ para baixo
Temporizadores	TON	Temporizador ligado
	TOF	Temporizador desligado
	TP	Temporizador de pulso
Inteiros analógicos	CMP	Comparador de blocos de função
	StackInt	Pilha de inteiros analógicos
Reais analógicos	AVERAGE	média corrente sobre amostras N
	HYSTER	Histerese sobre a diferença de 2 reais
	LIM_ALRM	Alarme de limite com histerese
	INTEGRAL	Integração em função do tempo
	DERIVATE	Diferenciação em relação ao tempo
Geração de sinal	BLINK	Sinal piscante
	SIG_GEN	Gerador de sinal

Nota: Quando novos blocos de função em "C" são desenvolvidos, eles podem ser chamados a partir da linguagem FBD.

SR



Argumentos:

SET1	BOO	se TRUE, força Q1 para TRUE (prioritário)
RESET	BOO	se TRUE, força Q1 para FALSE
Q1	BOO	estado da memória booleana

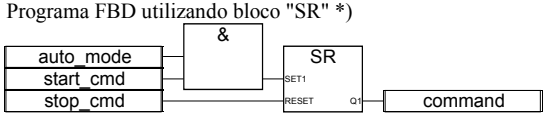
Descrição:

Força o biestável para prioridade (1): Veja a tabela verdade a seguir:

Set1	Reset	Q1	Q1 result.

0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(* Programa FBD utilizando bloco "SR" *)

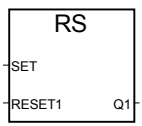


(* equivalência ST: supomos que SR1 é uma instância do bloco SR *)
SR1((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;

(* equivalência IL: *)

LD	auto_mode
AND	start_cmd
ST	SR1.set1
LD	stop_cmd
ST	SR1.reset
CAL	SR1
LD	SR1.Q1
ST	command

RS



Argumentos:

SET	BOO	se TRUE, força Q1 para TRUE
RESET1	BOO	se TRUE, força Q1 para FALSE (prioritário)
Q1	BOO	estado da memória booleana

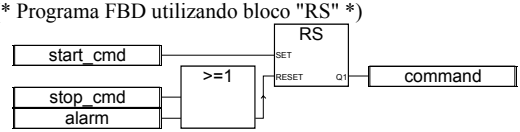
Descrição:

Força o biestável para prioridade (0): Veja a tabela verdade a seguir:

Set	Reset1	Q1	Q1 result.
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

1	1	0	0
1	1	1	0

(* Programa FBD utilizando bloco "RS" *)

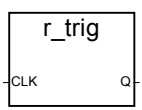


(* equivalência ST: supomos que RS1 é uma instância do bloco RS *)
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;

(* equivalência IL: *)

```
LD      start_cmd
ST      RS1.set
LD      stop_cmd
OR      alarm
ST      RS1.reset1
CAL     RS1
LD      RS1.Q1
ST      command
```

R_TRIG



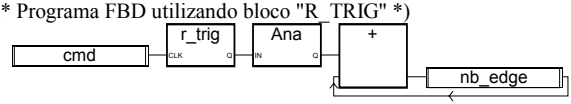
Argumentos:

CLK	BOO	qualquer variável booleana
Q	BOO	TRUE quando CLK vai de FALSE para TRUE FALSE em todos os outros casos

Descrição:

Detecta a borda de subida de uma variável booleana.

(* Programa FBD utilizando bloco "R_TRIG" *)



(* equivalência ST: supomos que R_TRIG1 é uma instância do bloco R_TRIG *)
R_TRIG1(cmd);
nb_edge := ANA(R_TRIG1.Q) + nb_edge;

(* equivalência IL: *)

```
LD      cmd
ST      R_TRIG1.clk
```

```
CAL     R_TRIG1
LD      R_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

F_TRIG

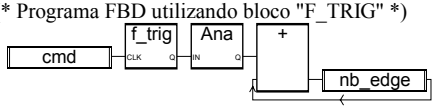
Argumentos:

CLK	BOO	qualquer variável booleana
Q	BOO	TRUE quando CLK vai de TRUE para FALSE FALSE em todos os outros casos

Descrição:

Detecta a borda de descida de uma variável booleana

(* Programa FBD utilizando bloco "F_TRIG" *)




```
(* equivalência ST: supomos que F_TRIG1 é uma instância do bloco F_TRIG *)
F_TRIG1(cmd);
nb_edge := ANA(F_TRIG1.Q) + nb_edge;

(* equivalência IL: *)
LD      cmd
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

SEMA

Argumentos:

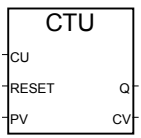
CLAIM	BOOLEAN	comando "test and set"

RELEASE	BOOLEAN	libera o semáforo
BUSY	BOOLEAN	estado do semáforo

Descrição:

```
(* "x" é uma variável booleana inicializada com FALSE *)
busy := x;
If claim Then
        x := True;
Else
        If release Then
                busy := False;
                x := False;
        End_if;
End_if;
```

CTU



Argumentos:

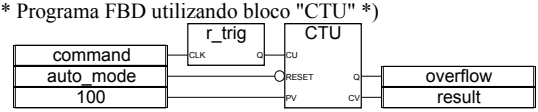
CU	BOO	contador de entrada (contagem se CU = TRUE)
RESET	BOO	comando zerar (reset) (prioritário)
PV	INT	valor máximo programado
Q	BOO	transbordo: TRUE quando CV = PV
CV	INT	resultado corrente do contador

Atenção: O bloco de CTU não detecta a borda de subida ou de descida do contador de entrada (CU). Deve estar associado com um bloco " R_TRIG " ou " F_TRIG " para criar um contador de pulso.

Descrição:

Conta (inteiro) de 0 até um valor determinado, de 1 em 1

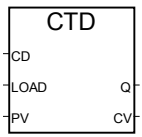
(* Programa FBD utilizando bloco "CTU" *)




```
(* equivalência ST: supomos que R_TRIG1 é uma instância do bloco R_TRIG e CTU1 é uma instância
do bloco CTU *)
CTU1(R_TRIG1(command),NOT(auto_mode),100);
overflow := CTU1.Q;
result := CTU1.CV;

(* equivalência IL: *)
LD              command
```

```
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

CTD



Argumentos:

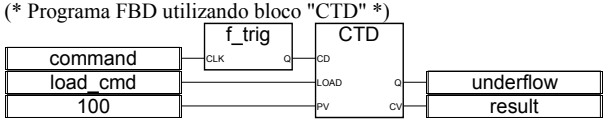
CD	BOO	contador de entrada (contagem se CD = TRUE)
LOAD	BOO	carrega comando (prioritário) (CV = PV quando LOAD = TRUE)
PV	INT	valor inicial programado
Q	BOO	underflow: TRUE quando CV = 0
CV	INT	resultado do contador

Atenção: O bloco CTD não detecta a borda de subida ou de descida do contador de entrada (CD). Deve estar associado com um bloco " R_TRIG " ou " F_TRIG " para criar um contador de pulso.

Descrição:

Conta (decrementa) (inteiro) um valor determinado até zero, de 1 em 1

(* Programa FBD utilizando bloco "CTD" *)



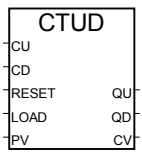
(* equivalência ST: supomos que F_TRIG1 é uma instância do bloco R_TRIG e CTD1 é uma instância do bloco CTD *)

```
CTD1(F_TRIG1(command),load_cmd,100);
underflow := CTD1.Q;
result := CTD1.CV;
```

(* equivalência IL: *)

```
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      result
```

CTUD



Argumentos:

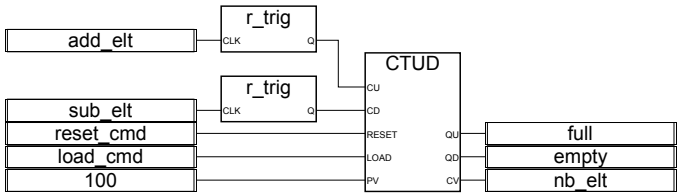
CU	BOO	incrementa (quando CU = TRUE)
CD	BOO	decrementa (quando CD = TRUE)
RESET	BOO	comando zerar (reset) (prioritário) (CV = 0 quando RESET = TRUE)
LOAD	BOO	comando carga (CV = PV quando LOAD = TRUE)
PV	INT	valor máximo programado
QU	BOO	transbordo: TRUE quando CV = PV
QD	BOO	underflow: TRUE quando CV = 0
CV	INT	resultado corrente da contagem

Atenção: O bloco CTUD não detecta as bordas de subida ou de descida do contador de entrada (CU e CD). Deve estar associado com um bloco " R_TRIG " ou " F_TRIG " para criar um contador de pulso.

Descrição:

Conta (inteiro) de 0 até um valor determinado, de 1 em 1
ou decrementa de um valor determinado até zero, de 1 em 1

(* Programa FBD utilizando bloco "CTUD" *)

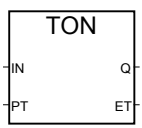


(* equivalência ST: supomos que R_TRIG1 e R_TRIG2 são duas instâncias do bloco R_TRIG e CTUD1 é uma instância do bloco CTUD *)
CTUD1(R_TRIG1(add_elt), R_TRIG2(sub_elt), reset_cmd, load_cmd,100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;

(* equivalência IL: *)

```
LD      add_elt
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTUD1.cu
LD      sub_elt
ST      R_TRIG2.clk
CAL     R_TRIG2
LD      R_TRIG2.Q
ST      CTUD1.cd
LD      reset_cmd
ST      CTUD1.reset
LD      load_cmd
ST      CTUD1.load
LD      100
ST      CTUD1.pv
CAL     CTUD1
LD      CTUD1.QU
ST      full
LD      CTUD1.QD
ST      empty
LD      CTUD1.CV
ST      nb_elt
```

TON



Argumentos: {XE "TON"} {XE "Temporizador ligado"}

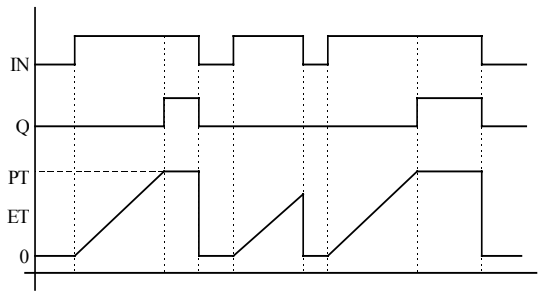
IN	BOO	se borda de subida, inicia o incremento do temporizador interno

		se borda de descida, interrompe e zera o contador interno
PT	TMR	tempo máximo programado
Q	BOO	se TRUE, o tempo programado está esgotado
ET	TMR	tempo decorrido depois do início

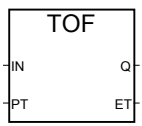
Descrição:

Incrementa um temporizador interno até um valor determinado.

Diagrama de tempo:



TOF



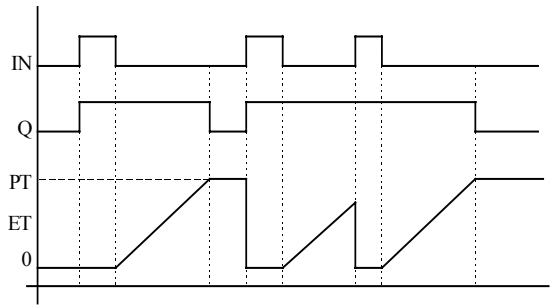
Argumentos:

IN	BOO	se borda de descida, inicia o incremento do temporizador interno
		se borda de subida, interrompe e zera o contador interno
PT	TMR	tempo máximo programado
Q	BOO	se TRUE, o tempo programado não está esgotado
ET	TMR	tempo decorrido depois do início

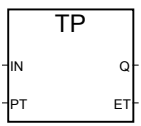
Descrição:

Incrementa um temporizador interno até um valor determinado.

Diagrama de tempo:



TP



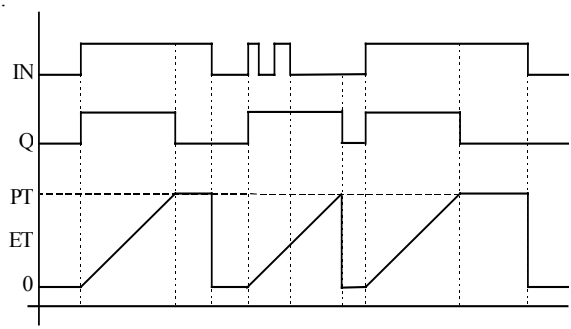
Argumentos:

IN	BOO	se borda de subida, inicia o incremento de um temporizador interno (se ainda não está ativo) se FALSE e somente se o tempo está esgotado, zera o temporizador interno (reset) Qualquer modificação em IN durante a contagem não tem efeito nenhum.
PT	TMR	tempo máximo programado
Q	BOO	se TRUE: a temporização está em andamento
ET	TMR	tempo decorrido corrente

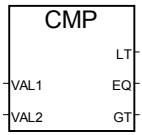
Descrição:

Incrementa um temporizador interno até um determinado valor.

Diagrama de tempo:



CMP



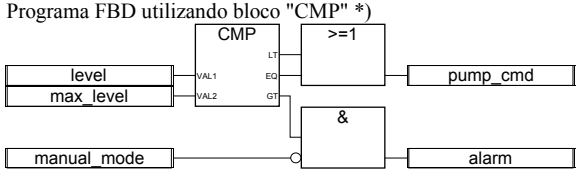
Argumentos:

VAL1	INT	qualquer valor analógico inteiro c/ sinal
VAL2	INT	qualquer valor analógico inteiro c/ sinal
LT	BOO	TRUE se val1 é Menor Que val2
EQ	BOO	TRUE se val1 é Igual A val2
GT	BOO	TRUE se val1 é Maior Que val2

Descrição:

Compara dois valores: indica se são iguais, ou se o primeiro é menor ou maior que o segundo.

(* Programa FBD utilizando bloco "CMP" *)



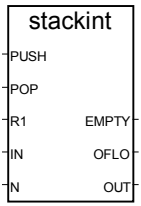
(* equivalência ST: supomos que CMP1 é uma instância do bloco CMP *)

```
CMP1(level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND NOT(manual_mode);
```

(* equivalência IL: *)

```
LD      level
ST      CMP1.val1
LD      max_level
ST      CMP1.val2
CAL     CMP1
LD      CMP1.LT
OR      CMP1.EQ
ST      pump_cmd
LD      CMP1.GT
ANDN    manual_mode
ST      alarm
```

STACKINT



Argumentos:

PUSH	BOO	comando “empilhar” (push) (somente na borda de subida) adiciona o valor IN no topo da pilha
POP	BOO	comando “desempilhar” (pop) (somente na borda de descida) apaga na pilha o último valor empilhado (topo da pilha)
R1	BOO	esvazia a pilha (reset)
IN	INT	valor a empilhar
N	INT	tamanho da pilha definido pelo aplicativo
EMPTY	BOO	TRUE se a pilha está vazia
OFLO	BOO	transbordo: TRUE se a pilha está cheia
OUT	INT	valor no topo da pilha

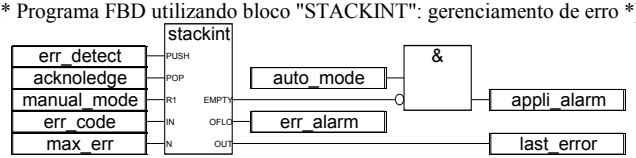
Descrição:

Gerenciamento de uma pilha de valores inteiros.

O bloco de função "STACKINT" inclui uma detecção de borda de subida para os comandos PUSH e POP (empilhar e desempilhar). O tamanho máximo absoluto da pilha é **128**. O tamanho máximo definido pelo aplicativo **N** não pode ser menor que 1 ou maior que 128.

Note que o valor OFLO só é válido após um comando zerar (reset) (R1 foi forçado a TRUE pelo menos uma vez e retornado a FALSE).

(* Programa FBD utilizando bloco "STACKINT": gerenciamento de erro *)




```
(* equivalência ST: supomos que STACKINT1 é uma instância do bloco STACKINT *)
STACKINT1(err_detect, acknoledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;

(* equivalência IL: *)
LD      err_detect
ST      STACKINT1.push
LD      acknoledge
ST      STACKINT1.pop
LD      manual_mode
```

```
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

AVERAGE

Argumentos:

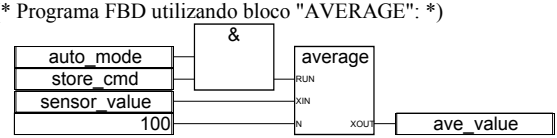
RUN	BOO	TRUE=executa / FALSE=reinicialização
XIN	REAL	qualquer variável analógica real
N	INT	número de amostras definido pelo aplicativo
XOUT	REAL	média corrente do valor de XIN

Descrição:

Armazena um valor a cada ciclo e calcula o valor médio de todos os valores já armazenados. Somente os N últimos valores são armazenados.

O número de amostras **N** não exceder a **128**.
Se o comando "**RUN**" é **FALSE** (reinicialização), o valor de saída é igual ao valor de entrada.
Quando N valores são armazenados, o primeiro valor que foi armazenado é apagado pelo último.

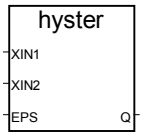
(* Programa FBD utilizando bloco "AVERAGE": *)




```
(* equivalência ST: AVERAGE1 instância do bloco AVERAGE *)
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;

(* equivalência IL: *)
LD      auto_mode
AND     store_cmd
ST      AVERAGE1.run
```

```
LD      sensor_value
ST      AVERAGE1.xin
LD      100
ST      AVERAGE1.N
CAL     AVERAGE1
LD      AVERAGE1.XOUT
ST      ave_value
```

HYSTER



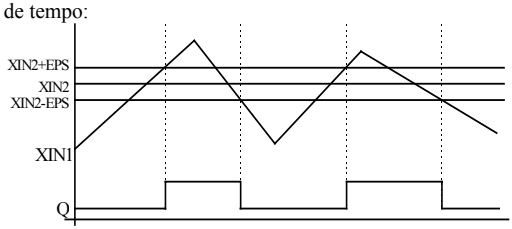
Argumentos:

XIN1	REAL	qualquer valor analógico real
XIN2	REAL	para verificar se XIN1 ultrapassou XIN2+EPS
EPS	REAL	valor de histerese (deve ser maior que zero)
Q	BOO	TRUE se XIN1 ultrapassou XIN2+EPS e ainda é menor que XIN2-EPS

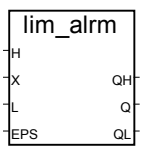
Descrição:

Histerese sobre uma valor real para um limite superior.

Exemplo de diagrama de tempo:



LIM_ALRM



Argumentos:

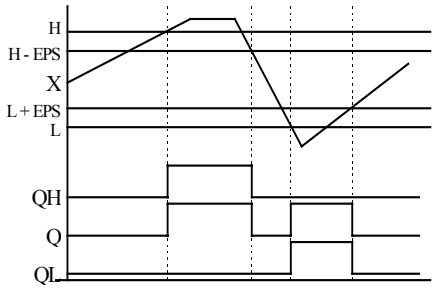
H	REAL	valor limite superior
X	REAL	entrada: qualquer valor analógico real
L	REAL	valor limite inferior
EPS	REAL	valor de histerese (deve ser maior que zero)

QH	BOO	alarme alto ("high"):TRUE se X acima do limite superior H
Q	BOO	alarme: TRUE se X fora dos limites
QL	BOO	alarme baixo ("low"):TRUE se X abaixo do limite inferior L

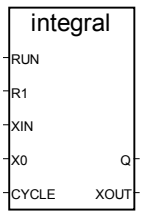
Descrição:

Histerese sobre um valor real para limites superior e inferior.

Uma histerese é aplicada sobre limites superior e inferior. A histerese delta utilizada para o limite superior ou para o limite inferior é uma média do valor do parâmetro EPS. A seguir, um exemplo de diagrama de tempo:



INTEGRAL



Argumentos:

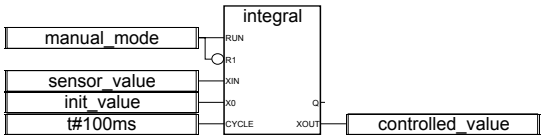
RUN	BOO	modo: TRUE=integração / FALSE=espera
R1	BOO	comando de zerar
XIN	REAL	entrada: qualquer valor analógico real
X0	REAL	valor inicial
CYCLE	TMR	período de amostragem
Q	BOO	Not R1
XOUT	REAL	saída integrada

Descrição:

Integração de um valor real.

Se valor do parâmetro "**CYCLE**" é menor que o tempo de ciclo do aplicativo ISaGRAF, o período de amostragem é o ciclo de tempo do aplicativo.

(* Programa FBD utilizando bloco "INTEGRAL": *)




```
(* equivalência ST: INTEGRAL1 instância do bloco INTEGRAL *)
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value, init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;

(* equivalência IL: *)
LD      manual_mode
ST      INTEGRAL1.run
STN     INTEGRAL1.R1
LD      sensor_value
ST      INTEGRAL1.XIN
LD      init_value
ST      INTEGRAL1.X0
LD      t#100ms
ST      INTEGRAL1.CYCLE
CAL     INTEGRAL1
LD      INTEGRAL1.XOUT
ST      controlled_value
```

DERIVATE

Argumentos:

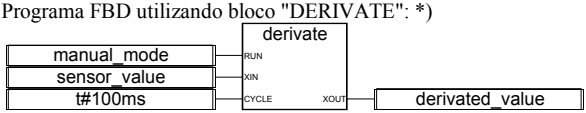
RUN	BOO	modo: TRUE=normal / FALSE=zerar
XIN	REAL	entrada: qualquer valor analógico real
CYCLE	TMR	período de amostragem
XOUT	REAL	saída: derivada do sinal

Descrição:

Diferenciação de um valor real.

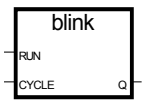
Se o valor do parâmetro "**CYCLE**" é menor que o tempo de ciclo do aplicativo ISaGRAF, o período de amostragem é o tempo de ciclo do aplicativo.

(* Programa FBD utilizando bloco "DERIVATE": *)



```
(* equivalência ST: DERIVATE1 instância do bloco DERIVATE *)
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;

(* equivalência IL: *)
LD      manual_mode
ST      DERIVATE1.run
LD      sensor_value
ST      DERIVATE1.XIN
LD      t#100ms
ST      DERIVATE1.CYCLE
CAL     DERIVATE1
LD      DERIVATE1.XOUT
ST      derivated_value
```

BLINK



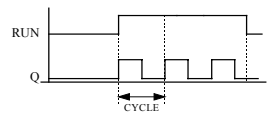
Argumentos:

RUN	BOO	modo: TRUE=piscante / FALSE=força a saída para falso
CYCLE	TMR	período de piscagem
Q	BOO	saída: sinal piscante

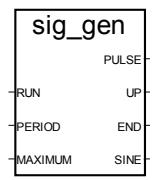
Descrição:

Gera um sinal piscante.

Diagrama de tempo:



SIG_GEN



Argumentos:

RUN	BOO	modo: TRUE=executando / FALSE=zerar
PERIOD	TMR	duração de uma amostra

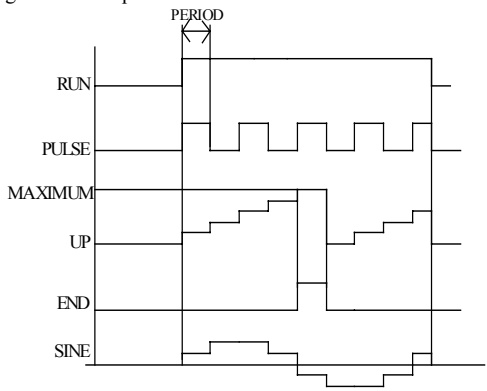
MAXIMUM	INT	valor máximo de contagem
PULSE	BOO	invertido a cada amostragem
UP	INT	contador para cima, incrementa a cada amostragem
END	BOO	TRUE em fim de contagem
SINE	REAL	sinal seno (período = tempo de contagem)

Descrição:

Gera diversos sinais: piscante sobre um booleano, um contador para cima inteiro e uma onda senoidal real.

Quando a contagem alcança o valor máximo, ele começa novamente de 0 (zero). Portando END mantém o valor TRUE somente durante 1 PERIOD.

Diagrama de tempo:



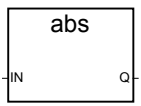
B.9.3 Funções padrão

Estas são funções padrão suportadas pelo sistema ISaGRAF. Tais funções são predefinidas e não têm que ser declaradas na biblioteca.

Matemáticas	ABS	Valor absoluto
	EXPT, POW	Exponenciação, potenciação
	LOG	Logaritmo
	SQRT	Radiciação (raiz quadrada)
	TRUNC	Parte inteira
Trigonométricas	ACOS, ASIN,	Arco coseno, Arco seno,
	ATAN	Arco tangente
	COS, SIN,	Coseno, Seno,
	TAN	Tangente
Controle de registro	ROL, ROR	Rotação à Esquerda, Rotação à Direita
	SHL, SHR	Deslocamento à Esquerda, Deslocamento à Direita
Manipulação de dados	MIN, MAX,	Mínimo, Máximo,
	LIMIT	Limite
	MOD	Módulo

	MUX4, MUX8	Multiplexador (4 ou 8 entradas),
	SEL	Seletor binário
	ODD	Paridade impar
	RAND	Valor aleatório
Conversão de dados	ASCII	Caractere → código ASCII
	CHAR	Código ASCII → Caractere
Cadeia de caracteres (“string”).....	MLEN	Comprimento “string”
	DELETE	Apagamento,
	INSERT	Inserção de caracteres
	FIND,	Localizar,
	REPLACE	Substituir caracteres
	LEFT, MID	Extração da esquerda para o meio
	RIGHT	ou da direita de uma “string”
	DAY_TIME	Data e hora correntes
Gerenciamento de array	ARCREATE	Cria array de valores inteiros
	ARREAD	Leitura dos elementos
ARWRITE	Escrita dos elementos	
Gerenciamento arq. binário.....	F_ROPEN	Abre arq. binário no modo Leitura
	F_WOPEN	Abre arq. binário no modo Escrita
	F_CLOSE	Fecha arquivo binário
	F_EOF	Testa o fim de um arquivo binário
	FA_READ	Lê um valor analógico em um arquivo binário
	FA_WRITE	Escreve um valor analógico em um arq. binário
	FM_READ	Lê uma mensagem (cadeia de caracteres (“string”)) em um arquivo binário
	FM_WRITE	Escreve uma mensagem (cadeia de caracteres (“string”)) em um arquivo binário

ABS



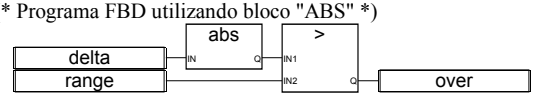
Argumentos:

IN	REAL	qualquer valor analógico c/ sinal
Q	REAL	valor absoluto (sempre positivo)

Descrição:

Dá o valor absoluto (positivo) de um valor real.

(* Programa FBD utilizando bloco "ABS" *)




```
(* equivalência ST: *)
over := (ABS (delta) > range);
```

```
(* equivalência IL: *)
LD      delta
ABS
GT      range
ST      over
```

EXPT

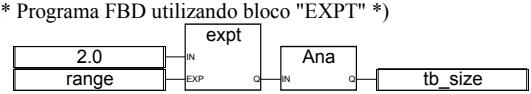
Argumentos:

IN	REAL	qualquer valor analógico real
EXP	INT	expoente analógico inteiro
Q	REAL	(IN ^{EXP})

Descrição:

Dá o resultado real da operação: (base ^{expoente}) 'base' sendo o primeiro argumento e 'expoente' o segundo.

(* Programa FBD utilizando bloco "EXPT" *)




```
(* equivalência ST: *)
tb_size := ANA (EXPT (2.0, range) );

(* equivalência IL: *)
LD      2.0
EXPT    range
ANA
ST      tb_size
```

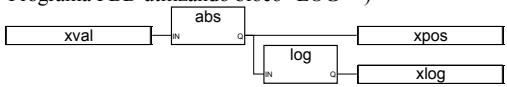
LOG

Argumentos:

IN	REAL	deve ser maior que zero
Q	REAL	logaritmo (base 10) do valor de entrada

Descrição:
Calcula o logaritmo (base 10) de um valor real.

(* Programa FBD utilizando bloco "LOG" *)

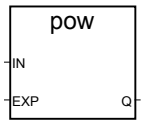


(* equivalência ST: *)
xpos := ABS (xval);
xlog := LOG (xpos);

(* equivalência IL: *)

LD	xval
ABS	
ST	xpos
LOG	
ST	xlog

POW



Argumentos:

IN	REAL	número analógico real a ser elevado
EXP	REAL	potência (expoente))
Q	REAL	(IN^{EXP})
		1.0 se IN não é 0.0 e EXP se 0.0
		0.0 se IN é 0.0 e EXP é negativo
		0.0 se ambos IN e EXP são 0.0
		0.0 se IN é negativo e Y não corresponde um inteiro

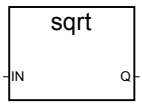
Descrição:
Dá o resultado real de uma operação: (base ^{expoente}) 'base' sendo o primeiro argumento e 'expoente' o segundo. O expoente é um valor real.

(* Programa FBD utilizando bloco "POW" *)

(* equivalência ST: *)
result := POW (xval, power);

```
(* equivalência IL: *)
LD          xval
POW         power
ST          result
```

SQRT



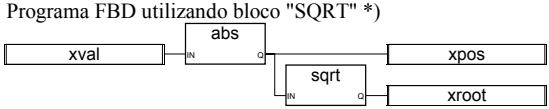
Argumentos:

IN	REAL	deve ser maior ou igual a zero
Q	REAL	raiz quadrada do valor de entrada

Descrição:

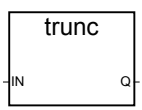
Calcula a raiz quadrada de um valor real.

(* Programa FBD utilizando bloco "SQRT" *)




```
(* equivalência ST: *)
xpos := ABS (xval);
xroot := SQRT (xpos);

(* equivalência IL: *)
LD          xval
ABS
ST          xpos
SQRT
ST          xrout
```

TRUNC



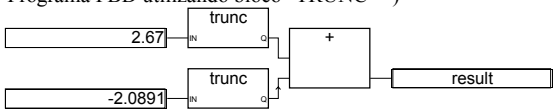
Argumentos:

IN	REAL	qualquer valor analógico REAL
Q	REAL	se IN>0, maior inteiro menor ou igual à entrada (IN) se IN<0, menor inteiro maior ou igual à entrada (IN)

Descrição:

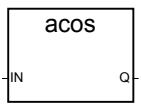
Trunca um valor real para obter sua **parte inteira** .

(* Programa FBD utilizando bloco "TRUNC" *)




```
(* equivalência ST: *)
result := TRUNC (+2.67) + TRUNC (-2.0891);
(* significa: result := 2.0 + (-2.0) := 0.0; *)

(* equivalência IL: *)
LD          2.67
TRUNC
ST          temporary    (* resultado temporário do primeiro TRUNC *)
LD          -2.0891
TRUNC
ADD         temporary
ST          result
```

ACOS



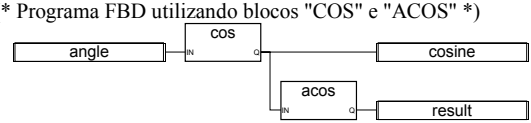
Argumentos:

IN	REAL	deve estar dentro do intervalo [-1.0 .. +1.0]
Q	REAL	arco coseno do valor de entrada no intervalo [0.0 .. PI]) = 0.0 para uma entrada inválida

Descrição:

Calcula o arco coseno de um valor real.

(* Programa FBD utilizando blocos "COS" e "ACOS" *)




```
(* equivalência ST: *)
cosine := COS (angle);
result := ACOS (cosine); (* resultado é igual ao ângulo *)

(* equivalência IL: *)
LD          angle
COS
ST          cosine
```

ACOS
ST result

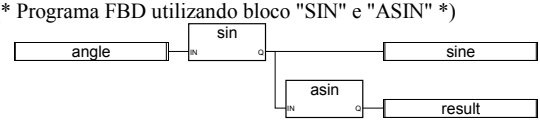
ASIN

Argumentos: {XE "ASIN"} {XE "Arco seno "}

IN	REAL	deve estar no intervalo [-1.0 .. +1.0]
Q	REAL	arco seno do valor de entrada do intervalo [-PI/2 .. +PI/2]) = 0.0 para entrada inválida

Descrição:
Calcula o arco seno de um valor real.

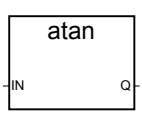
(* Programa FBD utilizando bloco "SIN" e "ASIN" *)



(* equivalência ST: *)
sine := SIN (angle);
result := ASIN (sine); (* resultado é igual ao ângulo *)

(* equivalência IL: *)
LD angle
SIN
ST sine
ASIN
ST result

ATAN



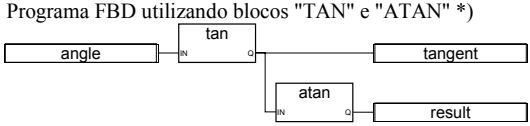
Argumentos:

IN	REAL	qualquer valor analógico real
Q	REAL	arco tangente do valor de entrada dentro do intervalo [-PI/2 .. +PI/2]) = 0.0 para entrada inválida

Descrição:

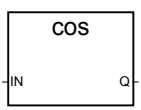
Calcula o acro tangente de um valor real.

(* Programa FBD utilizando blocos "TAN" e "ATAN" *)




```
(* equivalência ST: *)
tangent := TAN (angle);
result := ATAN (tangent); (* resultado é igual ao ângulo *)

(* equivalência IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          result
```

COS



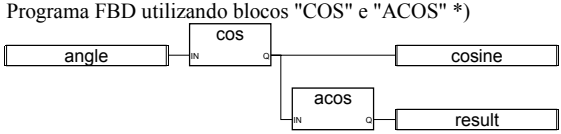
Argumentos:

IN	REAL	qualquer valor analógico REAL
Q	REAL	coseno do valor de entrada no intervalo [-1.0 .. +1.0])

Descrição:

Calcula o coseno de um valor real.

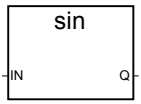
(* Programa FBD utilizando blocos "COS" e "ACOS" *)




```
(* equivalência ST: *)
cosine := COS (angle);
result := ACOS (cosine); (* resultado é igual ao ângulo *)

(* equivalência IL: *)
LD          angle
COS
ST          cosine
ACOS
```

ST result

SIN

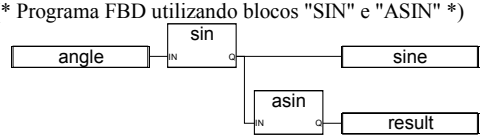


Argumentos:

IN	REAL	qualquer valor analógico REAL
Q	REAL	seno do valor de entrada no intervalo [-1.0 .. +1.0])

Descrição:
Calcula o seno de um valor real.

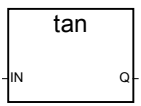
(* Programa FBD utilizando blocos "SIN" e "ASIN" *)



(* equivalência ST: *)
sine := SIN (angle);
result := ASIN (sine); (* resultado igual ao ângulo *)

(* equivalência IL: *)
LD angle
SIN
ST sine
ASIN
ST result

TAN

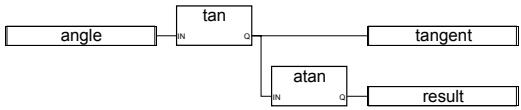


Argumentos:

IN	REAL	não pode ser igual a PI/2 módulo PI
Q	REAL	tangente do valor de entrada = 1E+38 para entrada inválida

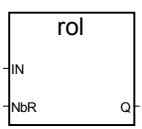
Descrição:
Calcula a tangente de um valor real.

(* Programa FBD utilizando blocos "TAN" e "ATAN" *)




```
(* equivalência ST: *)
tangent := TAN (angle);
result := ATAN (tangent); (* resultado igual ao ângulo *)

(* equivalência IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          result
```

ROL

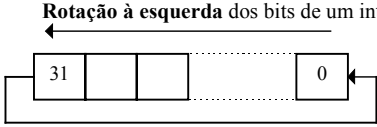


Argumentos:

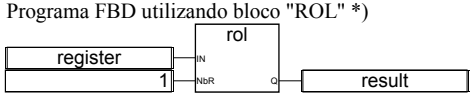
IN	INT	qualquer valor analógico inteiro
NbR	INT	número de rotações de 1 bit no intervalo [1..31])
Q	INT	valor rotacionado à esquerda sem efeito se NbR <= 0

Descrição:

Rotação à esquerda dos bits de um inteiro. A rotação é feita sobre 32 bits:



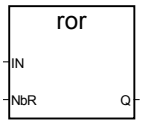
(* Programa FBD utilizando bloco "ROL" *)




```
(* equivalência ST: *)
result := ROL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result   = 2#1001_1010_0110_1010*)

(* equivalência IL: *)
LD          register
ROL         1
```

```
ST          result
```

ROR

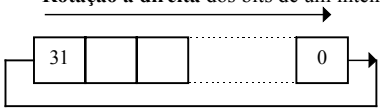


Argumentos:

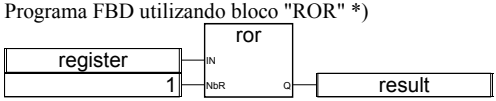
IN	INT	qualquer valor analógico inteiro
NbR	INT	número de rotações de 1 bit no intervalo [1..31])
Q	INT	valor rotacionado à direita sem efeito se NbR <= 0

Descrição:

Rotação à direita dos bits de um inteiro. A rotação é feita sobre 32 bits:

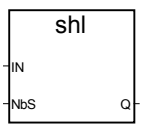


(* Programa FBD utilizando bloco "ROR" *)




```
(* equivalência ST: *)
result := ROR (register, 1);
(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1010_0110_1001_1010 *)

(* equivalência IL: *)
LD          register
ROR         1
ST          result
```

SHL



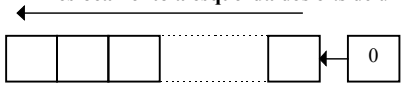
Argumentos:

IN	INT	qualquer valor analógico inteiro
NbS	INT	número de deslocamentos de 1 bit no intervalos [1..31])

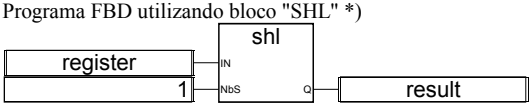
Q	INT	valor de deslocamentos à esquerda sem efeito se NbS <= 0 0 é utilizado para substituir o bit menos significativo

Descrição:

Deslocamento à esquerda dos bits de uma entrada. O deslocamento é efetuado sobre os 32 bits:

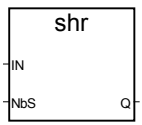


(* Programa FBD utilizando bloco "SHL" *)




```
(* equivalência ST: *)
result := SHL (register,1);
(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1001_1010_0110_1010 *)

(* equivalência IL: *)
LD      register
SHL     1
ST      result
```

SHR

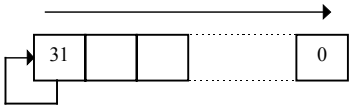


Argumentos:

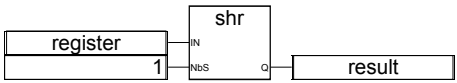
IN	INT	qualquer valor analógico inteiro
NbS	INT	número de deslocamentos de 1 bit no intervalos [1..31]
Q	INT	valor de deslocamentos à direita sem efeito de NbS <= 0 o bit mais significativo é copiado em cada deslocamento

Descrição:

Deslocamento à direita dos bits de uma entrada. O deslocamento é efetuado sobre 32 bits:

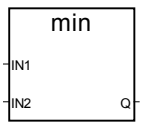


(* Programa FBD utilizando bloco "SHR" *)




```
(* equivalência ST: *)
result := SHR (register,1);
(* register = 2#1100_1101_0011_0101 *)
(* result   = 2#1110_0110_1001_1010 *)

(* equivalência IL: *)
LD          register
SHR         1
ST          result
```

MIN



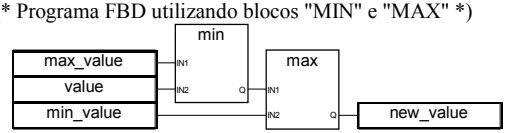
Argumentos:

IN1	INT	qualquer valor analógico inteiro c/ sinal
IN2	INT	(não pode ser REAL)
Q	INT	**mínimo** entre dois valores de entrada

Descrição:

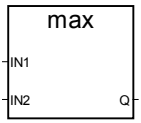
Dá o valor mínimo entre dois valores de inteiros.

(* Programa FBD utilizando blocos "MIN" e "MAX" *)




```
(* equivalência ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* Limita os valores no intervalo [min_value..max_value] *)

(* equivalência IL: *)
LD          max_value
MIN         value
MAX         min_value
ST          new_value
```

MAX



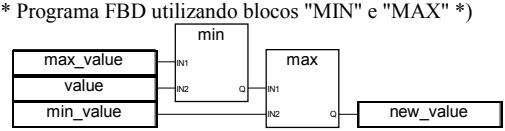
Argumentos:

IN1	INT	qualquer valor analógico inteiro c/ sinal
IN2	INT	(não pode ser REAL)
Q	INT	**máximo** entre dois valores de entrada

Descrição:

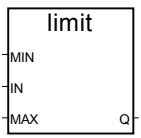
Dá o valor mínimo entre dois valores de inteiros.

(* Programa FBD utilizando blocos "MIN" e "MAX" *)




```
(* equivalência ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* Limita os valores no intervalo [min_value..max_value] *)

(* equivalência IL: *)
LD      max_value
MIN     value
MAX     min_value
ST      new_value
```

LIMIT



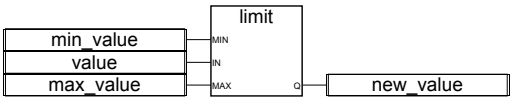
Argumentos:

MIN	INT	valor mínimo permitido
IN	INT	qualquer valor analógico inteiro c/ sinal
MAX	INT	valor máximo permitido
Q	INT	valor limitado ao alcance permitido

Descrição:

Limita um valor inteiro a um determinado intervalo. O valor se mantém se está entre o mínimo e o máximo; é modificado para o máximo se está acima ou para o mínimo se está abaixo.

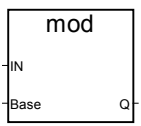
(* Programa FBD utilizando bloco "LIMIT" *)



(* equivalência ST: *)
new_value := LIMIT (min_value, value, max_value);
(* limita o valor dentro do intervalo [min_value..max_value] *)

(* equivalência IL: *)
LD min_value
LIMIT value, max_value
ST new_value

MOD

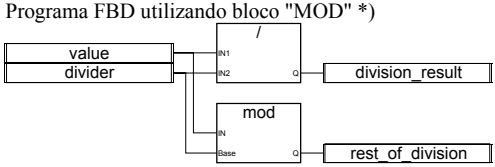


Argumentos:

IN	INT	qualquer valor analógico INTEIRO c/ sinal
Base	INT	deve ser maior que zero
Q	INT	cálculo do módulo (entrada da base MOD) retorna -1 se Base <= 0

Descrição:
Calcula o módulo de um valor inteiro.

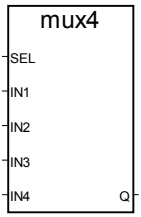
(* Programa FBD utilizando bloco "MOD" *)



(* equivalência ST: *)
division_result := (value / divider); (* divisão inteira *)
rest_of_division := MOD (value, divider); (* resto da divisão *)

(* equivalência IL: *)
LD value
DIV divider
ST division_result
LD value
MOD divider
ST rest_of_division

MUX4



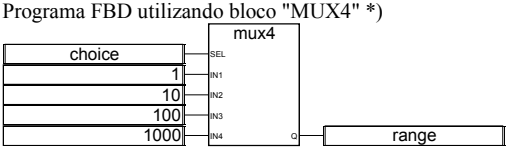
Argumentos:

SEL	INT	valor inteiro da seleção no intervalo [0..3]
IN1..IN4	INT	quaisquer valores analógicos inteiros
Q	INT	= valor1 se SEL = 0 = valor2 se SEL = 1 = valor3 se SEL = 2 = valor4 se SEL = 3 = 0 por todos os outros valores do seletor

Descrição:

Multiplexador com 4 entradas: seleciona um valor entre 4 valores inteiros.

(* Programa FBD utilizando bloco "MUX4" *)

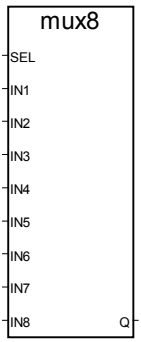


(* equivalência ST: *)
range := MUX4 (choice, 1, 10, 100, 1000);
(* seleciona um valor entre 4 alcances predefinidos, por exemplo, se a escolha é 1, o alcance é igual a 10 *)

(* equivalência IL: *)

```
LD      choice
MUX4    1,10,100,1000
ST      range
```

MUX8



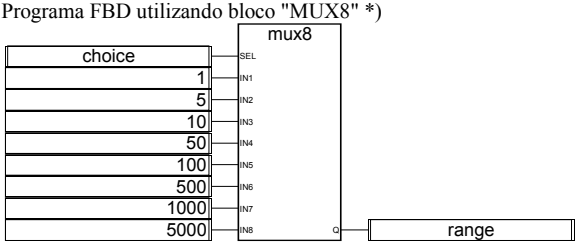
Argumentos:

SEL	INT	valor inteiro da seleção no intervalo [0..7]
IN1..IN8	INT	quaisquer valores analógicos inteiros
Q	INT	= IN1 se SEL = 0 = IN2 se SEL = 1 ... = IN8 se SEL = 7 = 0 para todos os outros valores do seletor

Descrição:

Multiplexador com 8 entradas: seleciona um valor entre 8 valores inteiros.

(* Programa FBD utilizando bloco "MUX8" *)

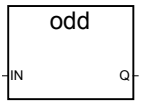


(* equivalência ST: *)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(* seleciona uma valor entre 8 alcances predefinidos, por exemplo, se a escolha é 3, o alcance é igual a 50 *)

(* equivalência IL: *)

```
LD          choice
MUX8        1,5,10,50,100,500,1000,5000
ST          range
```

ODD



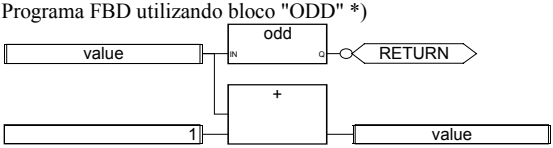
Argumentos:

IN	INT	qualquer valor analógico inteiro c/ sinal
Q	BOO	TRUE se o valor de entrada é ímpar FALSE se o valor de entrada é par

Descrição:

Verifica a **paridade** de um inteiro: o resultado é ímpar ou par.

(* Programa FBD utilizando bloco "ODD" *)

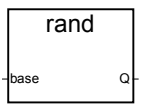


(* equivalência ST: *)
If Not (ODD (value)) Then Return; End_if;
value := value + 1;
(* faz o valor se sempre par *)

(* equivalência IL: *)

LD	value
ODD	
RETNC	
LD	value
ADD	1
ST	value

RAND



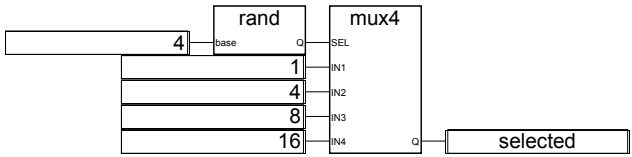
Argumentos:

base	INT	define o intervalo de valores autorizados
Q	INT	valor aleatório no intervalo [0..base-1]

Descrição:

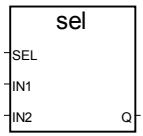
Retorna um **número aleatório** (valor inteiro) num intervalo dado.

(* Programa FBD utilizando bloco "RAND" *)




```
(* equivalência ST: *)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
seleção aleatória de 1 de 4 valores predefinidos.
O valor de saída da chamada de RAND está no intervalo [0..3],
de modo que a saída "selected" de MUX4, terá 'aleatoriamente' o valor
1 se  0 é a saída de RAND,
ou 4 se 1 é a saída de RAND,
ou 8 se 2 é a saída de RAND,
ou 16 se 3 é a saída de RAND,
*)

(* equivalência IL: *)
LD          4
RAND
MUX4        1,4,8,16
ST          selected
```

SEL



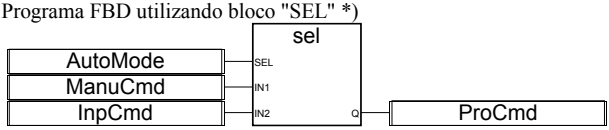
Argumentos:

SEL	BOO	indica o valor escolhido
IN1, IN2	INT	quaisquer valores analógicos inteiros
Q	INT	= IN1 se SEL = FALSE = IN2 se SEL = TRUE

Descrição:

Seletor binário: seleciona um valor entre 2 valores inteiros.

```
(* Programa FBD utilizando bloco "SEL" *)
```

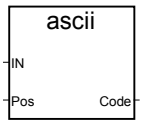



```
(* equivalência ST: *)
ProCmd := SEL (AutoMode, ManuCmd, InpCmd);
```

(* seleção de comando para o processo *)

(* equivalência IL: *)
LD AutoMode
SEL ManuCmd,InpCmd
ST ProCmd

ASCII

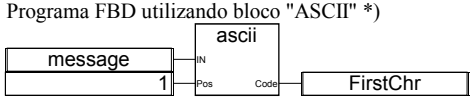


Argumentos:

IN	MSG	qualquer cadeia não vazia (“string”)
Pos	INT	posição do caractere selecionado no intervalo [1.. len] (len é o comprimento da mensagem IN)
Code	INT	código do caractere selecionado (no intervalo [0 .. 255]) retorna a 0 se Pos não está nesta “string”

Descrição:
Dá o código ASCII de um caractere em uma cadeia de caracteres (“string”) (mensagem).

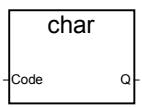
(* Programa FBD utilizando bloco "ASCII" *)



(* equivalência ST: *)
FirstChr := ASCII (message, 1);
(* FirstChr é o código Ascii do primeiro caractere da “string” *)

(* equivalência IL: *)
LD message
ASCII 1
ST FirstChr

CHAR



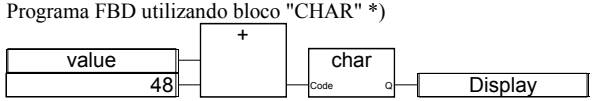
Argumentos:

Code	INT	código ASCII no intervalo [0 .. 255]
Q	MSG	cadeia de um caractere o caractere tem o código ASCII especificado em Code

(código ASCII é utilizado pelo módulo 256)

Descrição:
Dá uma mensagem de um caractere a partir do seu código ASCII.

(* Programa FBD utilizando bloco "CHAR" *)

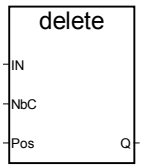


(* equivalência ST: *)
Display := CHAR (value + 48);
(* valor no intervalo [0..9] *)
(* 48 é o código ASCII de '0' *)
(* o resultado é uma cadeia de um caractere de '0' até '9' *)

(* equivalência IL: *)

```
LD      value
ADD     48
CHAR
ST      Display
```

DELETE

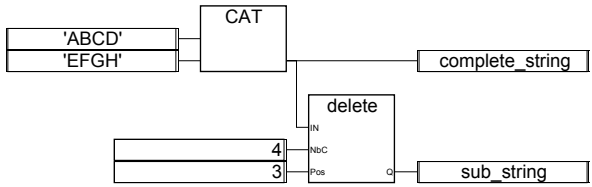


Argumentos:

IN	MSG	cadeia de caracteres (“string”) não vazia (“string”)
NbC	INT	número de caracteres a apagar
Pos	INT	posição do primeiro caractere apagado (primeiro caractere da “string” tem a posição 1)
Q	MSG	“string” modificada “string” vazia se Pos < 1 “string” inicial se Pos > comprimento da “string” IN “string” inicial se NbC <= 0

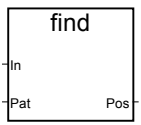
Descrição:
Apaga uma parte (**“sub-string”**) de uma “string” de caracteres (“string”).

(* Programa FBD utilizando bloco "DELETE" *)




```
(* equivalência ST: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string = 'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string = 'ABGH' *)

(* equivalência IL: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
DELETE      4,3
ST          sub_string
```

FIND



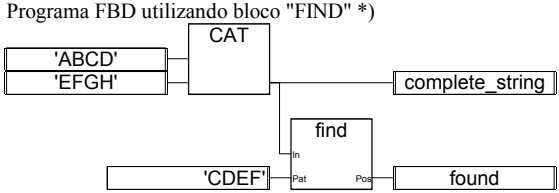
Argumentos:

In	MSG	cadeia de caracteres (“string”)
Pat	MSG	toda “string” não vazia (Pattern)
Pos	INT	= 0 se “sub-string” Pat não localizada = posição do primeiro caractere da primeira ocorrência da “sub-string” Pat (primeira posição válida é 1) esta função **faz distinção entre maiúscula/minúscula**

Descrição:

Localiza uma “sub-string” numa mensagem (“string”). Dá a posição da “sub-string” na mensagem (“string”).

(* Programa FBD utilizando bloco "FIND" *)

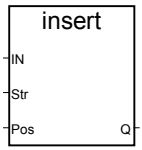



```
(* equivalência ST: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string = 'ABCDEFGH' *)
```

found := FIND (complete_string, 'CDEF'); (* localizada = 3 *)

(* equivalência IL: *)
LD 'ABCD'
ADD 'EFGH'
ST complete_string
FIND 'CDEF'
ST found

INSERT

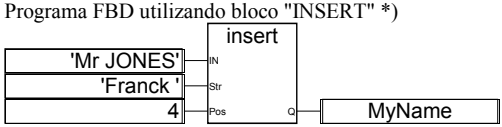


Argumentos:

IN	MSG	"string" inicial
Str	MSG	"string" a inserir
Pos	INT	posição da inserção a inserção é feita antes da posição (a primeira posição válida é 1)
Q	MSG	"string" modificada "string" vazia se Pos <= 0 concatenação de ambas as "strings" se Pos é maior que o comprimento da "string" IN

Descrição:
Insere uma "sub-string" em uma mensagem ("string") em uma determinada posição.

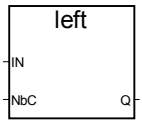
(* Programa FBD utilizando bloco "INSERT" *)



(* equivalência ST: *)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* MyName is 'Mr Frank JONES' *)

(* equivalência IL: *)
LD 'Mr JONES'
INSERT 'Frank ',4
ST MyName

LEFT



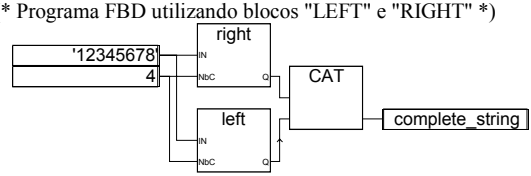
Argumentos:

IN	MSG	qualquer “string” não vazia
NbC	INT	Número de caracteres a extrair não pode ser maior que o comprimento da “string” IN
Q	MSG	parte esquerda da “string” IN (seu comprimento = NbC) “string” vazia se NbC <= 0 “string” completa IN se NbC >= comprimento da “string” IN

Descrição:

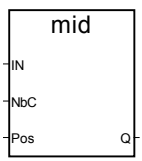
Extração da parte esquerda de uma mensagem (“string”). O número de caracteres a extrair é dado em um parâmetro.

(* Programa FBD utilizando blocos "LEFT" e "RIGHT" *)




```
(* equivalência ST: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string = '56781234'
o valor resultante da chamada RIGHT = '5678'
o valor resultante da chamada LEFT = '1234'
*)

(* equivalência IL: Primeiro feito é chamada para LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string    (* resultado intermediário *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

MID



Argumentos:

IN	MSG	qualquer “string” não vazia (“string”)
NbC	INT	número de caracteres a extrair não pode ser maior que o comprimento da “string” IN
Pos	INT	posição da “sub-string” o primeiro caractere da “sub-string” será o apontado por Pos (primeira posição válida é 1)
Q	MSG	parte central (meio) da “string” (seu comprimento = NbC) “string” vazia se os parâmetros não são válidos

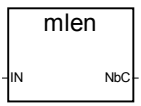
Descrição:

Extração de uma “sub-string” do meio de uma mensagem (“string”). O número de caracteres a extrair e a posição do primeiro caractere são dados em um parâmetro.

(* Programa FBD utilizando bloco "MID" *)


```
(* equivalência ST: *)
sub_string := MID ('abcdefgh', 2, 4);
(* sub_string = 'de' *)

(* equivalência IL: *)
LD      'abcdefgh'
MID     2,4
ST      sub_string
```

MLEN



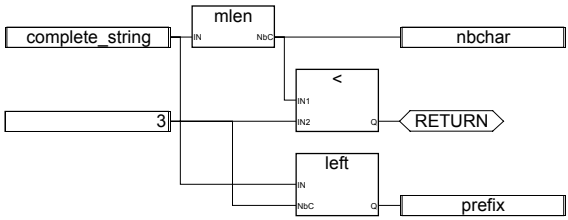
Argumentos:

IN	MSG	qualquer mensagem (“string”)
NbC	INT	número de caracteres na “string” IN

Descrição:

Calcula o comprimento de uma mensagem (“string”).

(* Programa FBD utilizando bloco "MLEN" *)

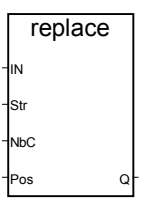


(* equivalência ST: *)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(* este programa extrai os 3 caracteres à esquerda da "string" e coloca o resultado em uma variável 'prefix'.
Nada é feito se o comprimento da "string" é menor que 3 caracteres *)

(* equivalência IL: *)

LD	complete_string
MLEN	
ST	nbchar
LT	3
RETC	
LD	complete_string
LEFT	3
ST	prefix

REPLACE



Argumentos:

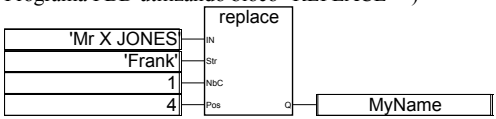
IN	MSG	qualquer "string"
Str	MSG	"string" a inserir (para substituir caracteres NbC)
NbC	INT	número de caracteres a apagar
Pos	INT	posição do primeiro caractere modificado (primeira posição válida é is 1)
Q	MSG	"string" modificada: - caracteres NbC são apagados no posição Pos - em seguida a "sub-string" Str é inserida nesta posição retorna uma "string" vazia se Pos <= 0 retorna concatenação de "strings" (IN+Str) se Pos é maior que o comprimento da "string" IN

retorna “string” inicial IN se NbC <= 0

Descrição:

Substituição de uma parte da mensagem (“string”) por um novo conjunto de caracteres.

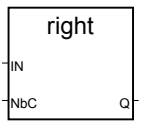
(* Programa FBD utilizando bloco "REPLACE" *)



(* equivalência ST: *)
MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);
(* MyName is 'Mr Frank JONES' *)

(* equivalência IL: *)
LD 'Mr X JONES'
REPLACE 'Frank',1,4
ST MyName

RIGHT



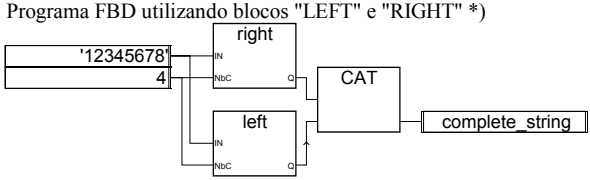
Argumentos:

IN	MSG	qualquer “string” não vazia
NbC	INT	não pode ser maior que o comprimento da “string” IN
Q	MSG	a parte à direita da “string” (comprimento = NbC) “string” vazia se NbC <= 0 “string” completa se NbC >= comprimento da “string”

Descrição:

Extração da parte à direita de uma “string”. O número de caracteres a extrair é dado em um parâmetro.

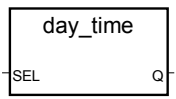
(* Programa FBD utilizando blocos "LEFT" e "RIGHT" *)



(* equivalência ST: *)

```
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string = '56781234'
o valor resultante da chamada RIGHT é '5678'
o valor resultante da chamada LEFT é '1234'
*)

(* equivalência IL: primeiro feito é chamada LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string    (* resultado intermediário *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

DAY_TIME



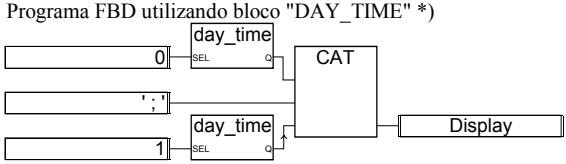
Argumentos:

SEL	INT	seleção da saída 0= data atual 1= hora atual 2= dia da semana
Q	MSG	data/hora sob a forma de “string” de caracteres 'YYYY/MM/DD' se SEL = 0 'HH:MM:SS' se SEL = 1 nome do dia se SEL = 2 (ex: 'Monday') (os nomes dos dias da semana são sempre em inglês)

Descrição:

Dá a data e a hora do dia sob a forma de uma mensagem (“string”).

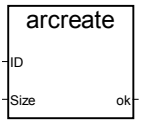
(* Programa FBD utilizando bloco "DAY_TIME" *)




```
(* equivalência ST: *)
Display := Day_Time (0) + ' ; ' + Day_Time (1);
(* O formato do texto mostrado é: 'YYYY/MM/DD ; HH:MM:SS' *)

(* equivalência IL: Primeiro feito é chamada ao day_time(1) *)
LD          1
```

```
DAY_TIME
ST          hour_str    (* resultado intermediário *)
LD          0
DAY_TIME
ADD         ' ; '
ADD         hour_str
ST          Display
```

ARCREATE



Argumentos:

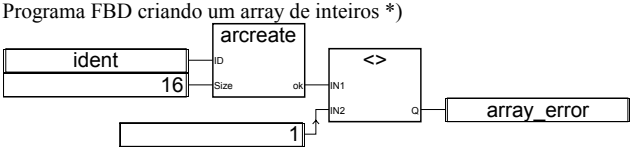
ID	INT	identificador do array (deve estar no intervalo [0..15])
Size	INT	número de elementos no array
ok	INT	estado da execução : **1** = se o array foi criado com sucesso **2** = identificador de array inválido ou array já criado **3** = tamanho inválido **4** = memória insuficiente

Descrição:

Criação de um array de inteiros.

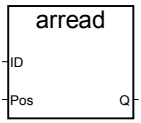
Atenção: Não pode haver mais que **16** arrays em um aplicativo. Os arrays contêm valores **analógicos inteiros**. A memória sendo alocada dinamicamente, este procedimento pode causar erro de sistema se o tamanho do array está muito próximo do tamanho da memória disponível.

(* Programa FBD criando um array de inteiros *)




```
(* equivalência ST: *)
array_error := (ARCREATE (ident, 16) <> 1));

(* equivalência IL: *)
LD          ident
ARCREATE    16
NE          1
ST          array_error
```

ARREAD



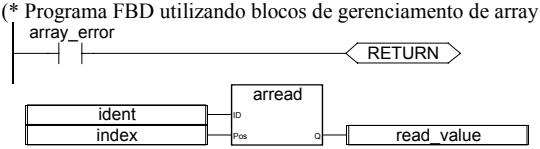
Argumentos:

ID	INT	identificador do array (deve estar no intervalo [0..15])
Pos	INT	posição do elemento no array deve estar no intervalo [0 .. size-1]
value	INT	valor do elemento lido 0 se os argumento não são válidos

Descrição:

Ler um elemento em um array de inteiros.

(* Programa FBD utilizando blocos de gerenciamento de array *)




```
(* equivalência ST: *)
If (array_error) Then Return; End_if;
read_value := ARREAD (ident, index);
(* array_error vem de uma chamada ARCREATE *)

(* equivalência IL: *)
LD          array_error
RETC
LD          ident
ARREAD      index
ST          read_value
```

ARWRITE

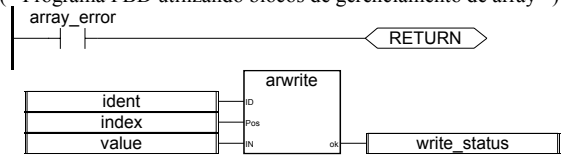
Argumentos:

ID	INT	identificador do array (deve estar no intervalo [0..15])
Pos	INT	posição do elemento no array deve estar no intervalo [0 .. size-1]
IN	INT	novo valor para o elemento
ok	INT	estado da execução

1 = escrita com sucesso
2 = identificador de array inválido
3 = índice inválido

Descrição:
Armazena (**escreve**) um valor em um **array** de inteiros.

(* Programa FBD utilizando blocos de gerenciamento de array *)



(* equivalência ST: *)
If (array_error) Then Return; End_if;
write_status := ARWRITE (Ident, Index, value);
(* array_error vem de uma chamada ARCREATE *)

(* equivalência IL: *)

LD	array_error
RETC	
LD	ident
ARWRITE	index,value
ST	write_status

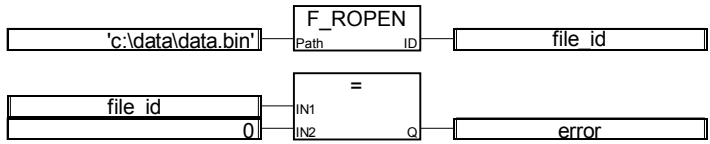
F_ROPEN

Argumentos:

Path	MSG	nome do arquivo Pode incluir o caminho de acesso ao arquivo utilizando o símbolo \ ou / para especificar um diretório. Para facilitar a mobilidade do aplicativo, / ou \ são equivalentes.
ID	INT	número do arquivo 0 se um erro ocorre: arquivo não existe.

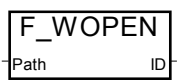
Descrição:
Abre um arquivo binário no modo **leitura**. Deve ser utilizado com FX_READ e F_CLOSE.
Esta função não está incluída no simulador ISaGRAF.

(* Programa FBD utilizando blocos de gerenciamento de arquivo *)




```
(* equivalência ST: *)
file_id := F_ROPEN('c:\data \data.bin');
error := (file_id=0);

(* equivalência IL: *)
LD          'c:\data\data.bin'
F_ROPEN
ST          file_id
EQ          0
ST          error
```

F_WOPEN



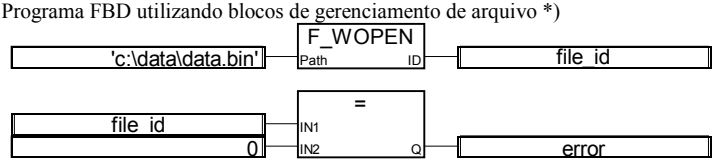
Argumentos:

Path	MSG	nome do arquivo Pode incluir o caminho de acesso ao arquivo utilizando o símbolo \ ou / para especificar um diretório. Para facilitar a mobilidade do aplicativo, / ou \ são equivalentes.
ID	INT	número do arquivo 0 se um erro ocorre. se o arquivo já existe, é sobrescrito.

Descrição:

Abre um arquivo binário no modo **escrita**. Deve ser utilizado com FX_WRITE e F_CLOSE. Esta função não está incluída no simulador ISaGRAF.

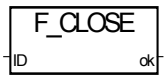
(* Programa FBD utilizando blocos de gerenciamento de arquivo *)




```
(* equivalência ST: *)
file_id := F_WOPEN('c:\data \data.bin');
error := (file_id=0);

(* equivalência IL: *)
LD          'c:\data\data.bin'
F_WOPEN
```

```
ST          file_id
EQ          0
ST          error
```

F_CLOSE



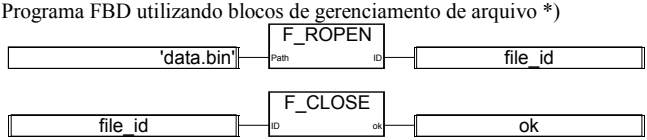
Argumentos:

ID	INT	número de arquivo: retornado por F_ROPEN ou F_WOPEN.
ok	BOO	estado de retorno TRUE se arquivo fechado está OK FALSE se ocorreu um erro no fechamento

Descrição:

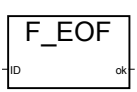
Fecha um arquivo binário aberto com a função F_ROPEN ou F_WOPEN.
esta função não está incluída no simulador ISaGRAF.

(* Programa FBD utilizando blocos de gerenciamento de arquivo *)




```
(* equivalência ST: *)
file_id := F_ROPEN('data.bin');
ok := F_CLOSE(file_id);

(* equivalência IL: *)
LD          'data.bin'
F_ROPEN
ST          file_id
F_CLOSE                 (* file_id já existe no resultado corrente IL *)
ST          ok
```

F_EOF



Argumentos:

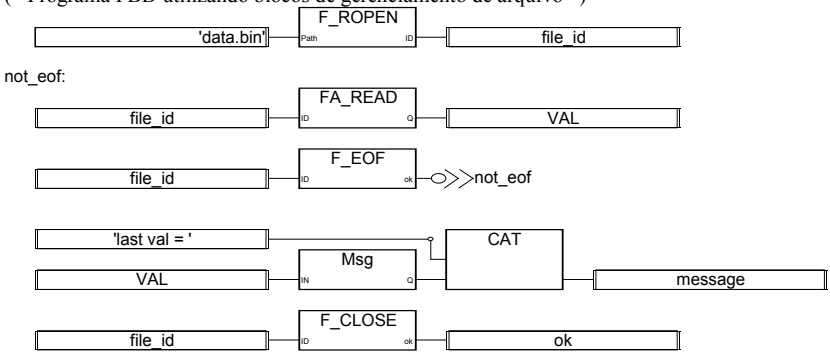
ID	INT	número de arquivo: retornado por F_ROPEN ou F_WOPEN.
ok	BOO	indicador de fim de arquivo

TRUE se o fim de arquivo foi alcançado no momento da última chamada de procedimentos de leitura / escrita.
Com FM_READ, a última mensagem lida de um arquivo pode não estar correta, se o último caractere não é um caractere de fim de "string".

Descrição:

Testa se o **fim de arquivo** foi alcançado.
esta função não está incluída no simulador ISaGRAF.

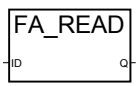
(* Programa FBD utilizando blocos de gerenciamento de arquivo *)




```
(* equivalência ST: *)
file_id := F_ROPEN('data.bin');
WHILE not(F_EOF(file_id))
        VAL := FA_READ(file_id);
END_WHILE;
MESSAGE := 'last val = ' + msg(VAL);
ok := F_CLOSE(file_id);

(* equivalência IL: *)
                LD          'data.bin'
                F_ROPEN
                ST          file_id
                LD          file_id
                F_EOF
                JMPC        END_OF_FILE
NOT_EOF:        LD          file_id
                FA_READ
                ST          VAL
                LD          file_id
                F_EOF
                JMPNC       NOT_EOF   (* se não há eof (fim de arquivo) continue a leitura *)
END_OF_FILE:    LD          VAL
                MSG
                ST          val_msg     (* conversão de VAL em uma mensagem *)
```

```
LD          'last val = '
ADD         val_msg
ST          MESSAGE
LD          file_id
F_CLOSE
ST          ok
```

FA_READ



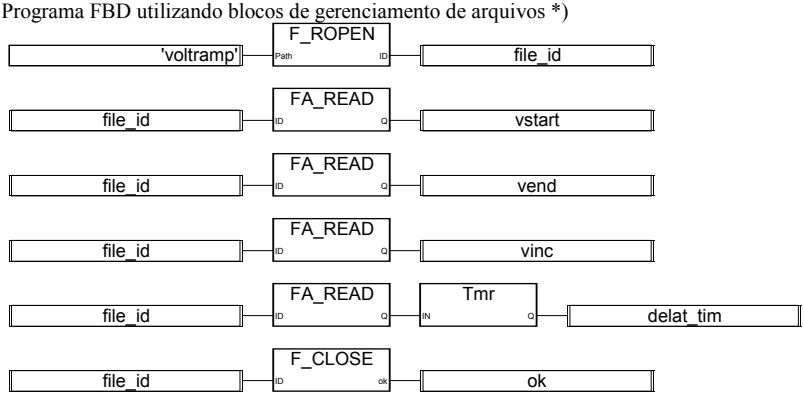
Argumentos:

ID	INT	número de arquivo: retornado por F_ROPEN.
Q	INT	valor analógico inteiro lido do arquivo

Descrição:

Leitura de variáveis analógicas de um arquivo binário. Deve ser utilizado com F_ROPEN e F_CLOSE.
Este procedimento faz o acesso seqüencial ao arquivo, a partir da última posição.
a primeira chamada após F_ROPEN lê os primeiros 4 bytes do arquivo,
cada chamada empurra o ponteiro de leitura.
Para verificar se o fim do arquivo foi alcançado, utilize F_EOF.
Esta função não está incluída no simulador ISaGRAF.

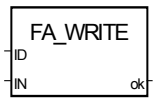
(* Programa FBD utilizando blocos de gerenciamento de arquivos *)




```
(* equivalência ST: *)
file_id := F_ROPEN('voltramp.bin');
vstart := FA_READ(file_id);
vend := FA_READ(file_id);
vinc := FA_READ(file_id);
delta_tim := tmr(FA_READ(file_id));
```

```
ok := F_CLOSE(file_id);

(* equivalência IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FA_READ             (* lê vstart *)
        ST      vstart
        LD      file_id
        FA_READ             (* lê vend *)
        ST      vend
        LD      file_id
        FA_READ             (* lê vinc *)
        ST      vinc
        LD      file_id
        FA_READ             (* lê delta_tim *)
        TMR                 (* conversão em temporizador *)
        ST      delta_tim
        LD      file_id
        F_CLOSE
        ST      ok
```

FA_WRITE



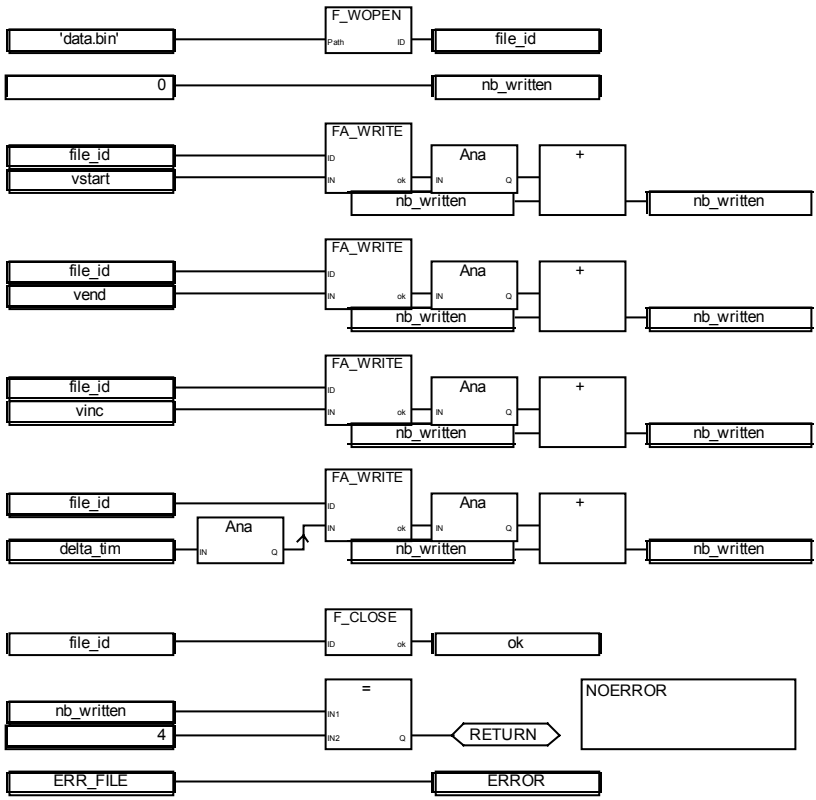
Argumentos:

ID	INT	número de arquivo: retornado por F_WOPEN.
IN	INT	valor analógico inteiro a escrever no arquivo
OK	BOO	estado da execução: TRUE se ok

Descrição:

Escrita dos valores analógicos num arquivo binário.
Este procedimento faz um acesso seqüencial a este arquivo, a partir da última posição.
A primeira chamada F_WOPEN escreve os primeiros 4 bytes do arquivo,
cada chamada empurra o ponteiro de escrita.
Esta função não está incluída no simulador ISaGRAF.

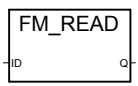
(* Programa FBD *)




```
(* equivalência ST: *)
file_id := F_WOPEN('voltramp.bin');
nb_written  := 0;
nb_written := nb_written + ana(FA_WRITE(file_id,vstart));
nb_written := nb_written + ana(FA_WRITE(file_id,vend));
nb_written := nb_written + ana(FA_WRITE(file_id,vinc));
nb_written := nb_written + ana(FA_WRITE(file_id,ana(delta_tim)));
ok := F_CLOSE(file_id);
IF ( nb_written <> 4) THEN
        ERROR := ERR_FILE;
END_IF;

(* equivalência IL: *)
                LD          'voltramp.bin'
                F_ROPEN
                ST          file_id
```

```
LD          0
ST          nb_written
LD          file_id       (* escreve vstart *)
FA_WRITE    vstart
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (* escreve vend *)
FA_WRITE    vend
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (* escreve vinc *)
FA_WRITE    vinc
ANA
ADD         nb_written
LD                        (* escreve delta_tim *)
ANA                       (* converte em um inteiro *)
ST          ana_delta_tim
LD          file_id
FA_WRITE    ana_delta_tim
ANA
ADD         nb_written
ST          nb_written
F_CLOSE
ST          ok
LD          nb_written
EQ          4
RETC                      (* retorna se igual a 4 *)
LD          ERR_FILE      (* caso contrário = erro *)
ST          ERROR
```

FM_READ



Argumentos:

ID	INT	número de arquivo: retornado por F_ROPEN.
Q	MSG	valor de mensagem lido do arquivo

Descrição:

Leitura das variáveis de mensagem num arquivo binário.
Para ser utilizado com F_ROPEN e F_CLOSE.
este procedimento faz um acesso seqüencial ao arquivo, a partir da última posição.
A primeira chamada após F_ROPEN lê a primeira “string” do arquivo,
cada chamada empurra o ponteiro de leitura.
Uma “string” é terminada por nulo (0), fim de linha ('\n') ou retorno ('\r');
Para verificar se o fim de arquivo foi alcançado, utilize F_EOF.

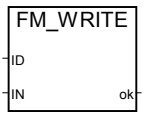
Esta função não está incluída no simulador ISaGRAF.

(* Programa FBD utilizando blocos de gerenciamento de arquivo *)


```
(* equivalência ST: *)
file_id := F_ROPEN('voltramp.bin');
status1 := FM_READ(file_id);
status2 := FM_READ(file_id);
IF (F_EOF(file_id)) THEN
        ERROR := ERR_FILE;
        unused_eof_mes := FM_READ(file_id);
END_IF;
ok := F_CLOSE(file_id);

(* equivalência IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FM_READ         (* lê status1 *)
        ST      status1
        LD      file_id
        FM_READ         (* lê status2 *)
        ST      status2
        LD      file_id
        F_EOF
        JMPNC   CLOSE_FILE (* se fim de arquivo, desvio não efetuado *)
        LD      ERR_FILE
        ST      ERROR
        LD      file_id
        FM_READ         (* lê unused_eof_mes *)
```

```
            ST          unused_eof_mes
CLOSE_FILE  LD          file_id
            F_CLOSE
            ST          ok
```

FM_WRITE



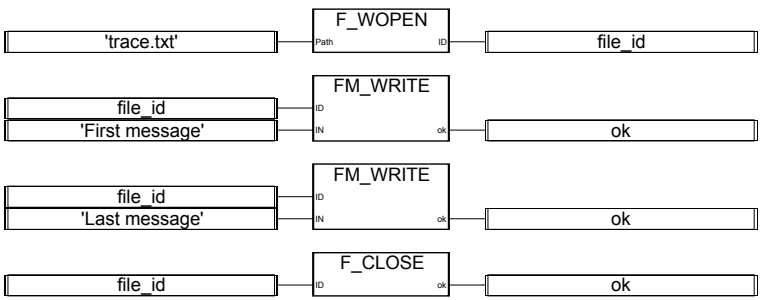
Argumentos:

ID	INT	número de arquivo: retornado por F_WOPEN.
IN	MSG	valor da mensagem a escrever no arquivo
ok	BOO	estado da execução TRUE em caso de sucesso

Descrição:

Escrita das variáveis de mensagem em um arquivo binário.
Deve ser utilizado com F_WOPEN e F_CLOSE.
Uma mensagem é escrita no arquivo como uma "string" terminada pelo caractere nulo.
Este procedimento faz um acesso seqüencial ao arquivo, a partir da última posição.
A primeira chamada após F_WOPEN escreve a primeira "string" no arquivo,
cada chamada empurra o ponteiro de escrita.
Esta função não está incluída no simulador ISaGRAF.

(* Programa FBD utilizando blocos de gerenciamento de arquivos *)



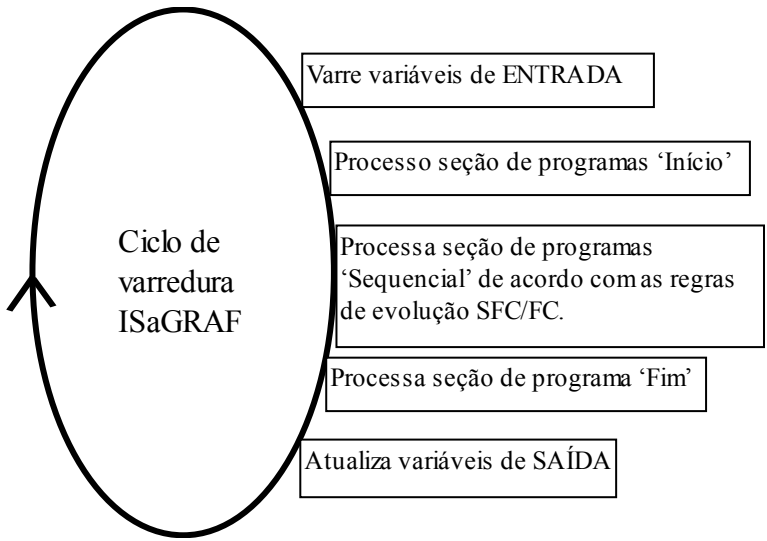

```
(* equivalência ST: *)
file_id := F_WOPEN('trace.txt');
ok := FM_WRITE(file_id,'First message');
ok := FM_WRITE(file_id,'Last message');
ok := F_CLOSE(file_id);

(* equivalência IL: *)
```

```
LD          'trace.txt'
F_WOPEN
ST          file_id
FM_WRITE 'First message'        (* escreve primeira mensagem *)
ST          ok
LD          file_id
FM_WRITE 'Last message'         (* escreve segunda mensagem *)
ST          ok
LD          file_id
F_CLOSE
ST          ok
```

C. Manual do Usuário Destino

C.1 Introdução

O ISaGRAF destino é um software em tempo real executando um aplicativo ISaGRAF em seu sistema ou placa de computador industrial de acordo com o seguinte esquema bem conhecido:



O ciclo destino consiste na varredura das entradas físicas do processo a controlar, processando os dados do aplicativo de acordo com os programas aplicativos do ambiente de trabalho ISaGRAF[1] e então atualizando as saídas físicas.

- A primeira parte desta seção explica como iniciar os sistemas destinos específicos (DOS, OS-9, VxWorks e NT). Para cada um destes sistemas, esta parte explicará como executar o ISaGRAF destino. Em seguida, informações são dadas sobre as características específicas tais como: inicialização do destino no momento da alimentação (ligar), gerenciamento de erro, comportamento geral,...
- A segunda parte descreve o método de implementação das funções C, dos blocos de função e das funções de conversão do usuário para melhorar o desempenho do ISaGRAF destino.
- A terceira parte contém as informações sobre o Modbus e a implementação do ISaGRAF. Descreve os formatos padrões dos diferentes códigos de função.
- A quarta parte descreve as ferramentas para o gerenciamento das falhas de alimentação e reinicialização do destino.

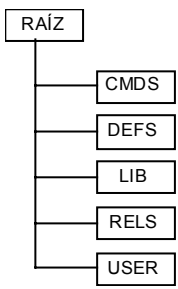
[1] Este manual assume que o usuário está familiarizado com a área de trabalho do ISaGRAF.

C.2 Instalação

A instalação requer aproximadamente 1 Mbyte de espaço livre em seu disco rígido. A instalação é realizada com o programa **install.bat** entregue com o disco de instalação. Este programa instala todos os arquivos necessários para uma plataforma específica em seu PC.

Exemplo: a:\install a: c:*path*
Instala os arquivos a partir da unidade de disquete a: no c: no diretório *path*.

A seguinte arquitetura é utilizada:



O diretório RAÍZ contém algumas ferramentas e os arquivos Readme;
O diretório CMDS contém os arquivos executáveis;
O diretório DEFS contém os arquivos de definição;
O diretório LIB contém as bibliotecas;
O diretório RELS contém os arquivos realocáveis;
O diretório USER contém os procedimentos "C" para as funções, bloco de funções e funções de conversão em "C" (arquivos fonte e definição)

Por enquanto, vamos iniciar com a plataforma instalada.

C.3 Iniciando com o ISaGRAF DOS destino

C.3.1 Executando o ISaGRAF: ISA.EXE

Na implementação MS-DOS, o destino é executado como um programa simples: ISA.EXE. Para iniciar basta executar o comando de ajuda **isa -?** No diretório CMDS.
Em tal sistema, as operações podem ser críticas. Por exemplo é recomendado sobrecarregar o link de comunicação para garantir o bom desempenho. O programa destino não impede a execução das rotinas de controle de interrupção.

Link de comunicação e configuração: opção -t

O ISaGRAF destino utiliza uma link serial para a comunicação com o depurador. O nome da porta é especificado com a opção **-t** . Como a interface de comunicação é desenvolvida para ser compatível com qualquer tipo de máquina, as portas COM1, COM2 ou COM3 podem ser utilizadas, dependendo da versão da BIOS.

Sem valor padrão: Se esta opção não é utilizada, a comunicação com o destino não é possível. Neste caso, o erro número 7 pode ser mostrado.

Com o ISaGRAF DOS destino a comunicação utilizando um link Ethernet não está disponível. Solicite ao seu fornecedor uma implementação especial.

Os parâmetros de comunicação tem que ser definidos antes de iniciar o ISaGRAF, de modo que o usuário fique totalmente livre para utilizar os parâmetros necessários. Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Exemplo:
MODE COM1:9600,N,8,1
Configura os parâmetros de comunicação com os seguintes valores:
- Taxa de transmissão de dados é 9.600 bauds
- Sem paridade
- 8 bits de dados
- 1 stop bit

Note que o valor de 19.200 bauds, fornecido como padrão no ambiente de trabalho, não está autorizado para certas versões da BIOS.
CJ fornece o utilitário ISAMOD.EXE para configurar os parâmetros do ambiente de trabalho:
ISAMOD COM1
É equivalente à MODE COM1:19200,N,8,1

Número escravo: opção -s

Esta opção especifica o número escravo do destino. Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Este número escravo é utilizado pelo protocolo de ligação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados. Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente

de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Valor padrão: O número escravo padrão é 1 (o mesmo do ambiente de trabalho)

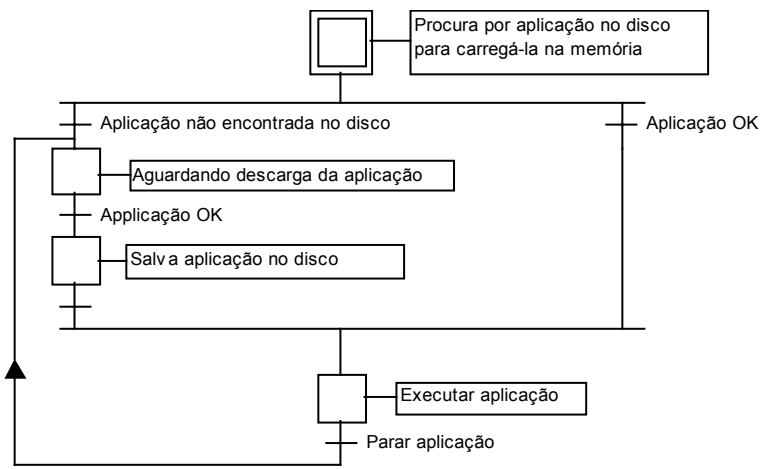
Exemplos:

isamod COM1	Configura a COM1 para 19.200 bauds, sem paridade, 8 bits de dados, 1 stop bit.
isa -t=COM1	Inicia o ISaGRAF destino com o número escravo padrão (1) e com a COM1 como porta de comunicação.
isa -s=3 -t=COM1	Inicia o ISaGRAF destino com o número escravo 3 e com a COM1 como porta de comunicação.

C.3.2 Recursos específicos

Iniciando o ISaGRAF

Quando o destino é iniciado, o seguinte algorítmo é executado.



- **Definições**

O código do aplicativo é a base dos dados binários gerados e carregados pelo ambiente de trabalho e em seguida executado pelo destino. Pode ser completado pela tabela de símbolos.
A tabela de símbolos de um aplicativo é uma base de dados ASCII gerado e carregado pelo ambiente de trabalho. Esta tabela faz a ligação entre os objetos simbólicos e os objetos internos do destino. Não é necessário no destino exceto para o gerenciamento de símbolos específicos do usuário. Para maiores informações sobre a tabela de símbolos veja o manual do usuário:Técnicas de programação avançada.

- **Cópia de segurança do aplicativo**

Quando um novo aplicativo é carregado no destino, o código do aplicativo é gravado no diretório corrente do destino com o nome do arquivo:

ISAx1 arquivo cópia de segurança do código do aplicativo ISaGRAF (x é o número escravo)

Entretanto, se a tabela de símbolos do aplicativo tiver sido carregada anteriormente, também é gravada no diretório corrente do destino com o nome:

ISAx6 arquivo de cópia de segurança dos símbolos do aplicativo ISaGRAF (x é o número escravo)

Quando o destino ISaGRAF é iniciado, estes arquivos (código e símbolos do aplicativo) são localizados no diretório corrente e carregados na memória.
Se nenhum arquivo de símbolos está disponível, o destino inicia a execução do código de aplicativo sem os símbolos.
Se nenhum código do aplicativo está disponível, o destino aguarda o carregamento de um aplicativo.

Para iniciar o destino com um aplicativo específico no momento da alimentação, sem utilizar o depurador, pode-se copiar estes arquivos a partir do disco rígido (se o ambiente de trabalho está no mesmo PC) ou com a ajuda de um disquete diretamente no diretório corrente do destino. Se não há unidade de disco no destino, você pode utilizar um disco virtual.

Se o ambiente de trabalho ISaGRAF está instalado no diretório padrão \ISAWIN, o nome do arquivo de código do aplicativo do projeto MYPROJ é:

\ISAWIN\APL\MYPROJ\appli.x8m

e o nome do arquivo de símbolos do aplicativo do projeto MYPROJ é:

\ISAWIN\APL\MYPROJ\appli.tst

Exemplo:
A partir do diretório no qual o arquivo **isa.exe** está instalado, se o seguinte comando é inserido:

copy \ISAWIN\APL\MYPROJ\appli.x8m isa11

então o arquivo isa.exe localizará e executará o aplicativo 'myproj'.

Todos estes comandos podem ser agrupados num arquivo Batch, por exemplo, e então iniciado a partir do menu de utilidades do ambiente de trabalho (veja o manual do usuário: Gerenciamento de programas).

Gerenciamento de erros e mensagens de saída

O software do ISaGRAF destino permite o gerenciamento da detecção de erros. Você encontrará a lista de mensagens de erro com suas descrições no anexo.

A detecção de erros é tratada da seguinte maneira:

- Um erro é composto de um erro e de um número de argumento que são enviados à rotina de erros do ISaGRAF
- Se a detecção dos erros for validada nos parâmetros de execução do ambiente de trabalho, o erro é processado. Se não, a informação é perdida e o gerenciamento de erros interrompido.

Quando o erro é processado:

- Número do erro (valor decimal) e do argumento (valor hexadecimal) são mostrados na saída padrão stdout .
- Número do erro e do argumento são enviados para um "buffer" de erro do tipo FIFO de modo a serem recuperados mais tarde. O tamanho do "buffer" é definido nas opções "parâmetros de

execução" do ambiente de trabalho. Quando o "buffer" está cheio, a cada novo erro que chega, o mais antigo é perdido.

- Erros podem ser chamados ou pelo depurador ou pelo aplicativo em execução utilizando a chamada SYSTEM (veja o manual do usuário).

Quando o depurador detecta um erro, uma mensagem descrevendo o erro é mostrada na janela de erro. Dependendo do contexto do aplicativo (em execução ou não) o depurador pode mostrar o nome do objeto (variável ou programa) do qual o erro vem, ou o erro do argumento (valor decimal) entre parênteses [x] que pode ter um significado diferente para cada erro.

Uma mensagem de saudação e os valores dos erros são mostrados na saída stdout quando o destino inicia e quando um erro é detectado. Se esta tela não é desejada no canal de saída padrão, um comando de redirecionamento pode ser utilizado, tal como:

isa -t=COM1 -s=1 **>NUL**

Sistema de relógio

Como o ISaGRAF destino é projetado para ser executado em qualquer sistema, a referência de tempo utilizada para a sincronização do ciclo bem como para a atualização das variáveis de temporização, é o tic padrão de 55 milisegundos.

Assim, não é possível obter uma precisão maior que 55ms para as variáveis de temporização. Por esta razão, um tempo de ciclo com duração menor ou igual a 55 ms e diferente de zero gera um erro de transbordo de tempo de ciclo (erro 62) e dos ciclos ativados.

É desaconselhável modificar o tic do sistema para evitar que os aplicativos residentes, ou as funções e blocos de função em "C" integrados no aplicativo, sejam perturbados pela execução do ISaGRAF.

Solicite ao seu fornecedor uma implementação específica se seu aplicativo necessita de maior precisão.

Sair do ISaGRAF

Durante o teste de um aplicativo sob condições não industriais em um PC, o usuário pode interromper o ISaGRAF através de uma combinação de teclas, evitando as paradas não desejadas. A seqüência de teclas é a seguinte:

shift + ctrl + alt

Naturalmente, se o aplicativo industrial não deve ser interrompido quando uma tecla é pressionada, deve-se desativar esta combinação.

Um efeito secundário perigoso devido a estas saídas rápidas é o seguinte: a interface das placas de E/S não é fechada. Por esta razão, a maneira correta de interromper o ISaGRAF destino é a seguinte:
- interromper o aplicativo à partir do depurador (isto fechará as placas de E/S)
- interromper o ISaGRAF destino à partir do teclado

Tamanho do aplicativo

Como o ISaGRAF MS-DOS destino é projetado para Intel modo real, o tamanho máximo de uma estrutura de dados é 64k. Assim, o código do aplicativo carregado pelo ambiente de trabalho não deve exceder este limite. Em alguns casos raros, a estrutura interna alocada pelo ISaGRAF também pode

exceder este limite e travar o aplicativo após a carga. A memória total disponível está limitada a 640k de memória convencional.

Solicite ao seu fornecedor uma implementação especial se o seu aplicativo necessita de maior capacidade de memória.

C.4 Iniciando com o ISaGRAF OS9 destino

Antes de mais nada, você deve transferir os arquivos (pelo menos os executáveis do diretório CMDS) para o seu OS-9 destino utilizando qualquer ferramenta de transferência de arquivo.
Então para começar você pode simplesmente executar os comandos de ajuda do diretório CMDS do seu sistema OS-9.

isa -?
isaker -?
isatst -?
Isanet -?

C.4.1 Executando a mono tarefa ISaGRAF: isa

. O ISaGRAF destino pode ser executado como uma única tarefa. Mas as operações em tal configuração podem ser críticas. Por exemplo, é recomendado não sobrecarregar o link de comunicação para garantir o bom desempenho. No sistema multitarefa OS-9, diferentes ISaGRAF destino mono-tarefas podem ser executados na mesma CPU desde que possuam diferentes números escravos e portas de comunicação.
Esta implementação mono-tarefa foi projetada principalmente para plataformas de hardware simples como placas de baixo custo ou PCs MS-DOS ou para criar um primeiro protótipo para a abordagem de uma nova plataforma. Por esta razão, a implementação ISaGRAF destino multitarefa seja a preferida.
O ISaGRAF destino mono-tarefa não impede a execução de processos de fundo ou de rotinas controladas por interrupção.

Link de comunicação e configuração: opção -t

O ISaGRAF destino mono-tarefa utiliza uma link serial para a comunicação com o depurador. O nome da descrição é definido com a ajuda da opção **-t** .

Sem valor padrão: Se esta opção não é utilizada, a comunicação com o destino não é possível. Neste caso, o erro número 7 pode ser mostrado.

Com a implementação mono-tarefa a comunicação utilizando um link Ethernet não está disponível.

O dispositivo de link serial é aberto no modo transferência de dados binários (sem caracteres de controle, sem XON/XOFF). Antes de iniciar o ISaGRAF, é necessário configurar os parâmetros de comunicação de forma que o usuário possa utilizar os parâmetros necessários com toda a liberdade. Ao utilizar o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho (veja o guia de usuário: Gerenciamento de programas) correspondam aos do destino.
Exemplo:
xmode /t0 baud=19200
Configura a taxa de transmissão de dados para 19.200 bauds no dispositivo /t0

Número escravo: opção -s

Esta opção define o número escravo do destino . Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Este número escravo é utilizado pelo protocolo de ligação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados. Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente

de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Valor padrão: O valor padrão do número escravo é 1 (como o do ambiente de trabalho)

Exemplos:

isa -t=/t0	Inicia um ISaGRAF destino mono-tarefa com o valor padrão do número escravo e com /t0 como porta de comunicação.
isa -s=3 -t=/t1	Inicia um ISaGRAF destino mono-tarefa com o número escravo 3 e com /t1 como porta de comunicação.
isa -t=/t0 & **isa -s=3 -t=/t1**	Inicia dois ISaGRAF destino mono-tarefa. Um com o número escravo padrão (1) e com /t0 como porta de comunicação. O outro com o número escravo 3 e com /t1 como porta de comunicação.

C.4.2 Executando o ISaGRAF multitarefas: isaker, isatst, isanet

Para melhorar o tempo de resposta do núcleo do ISaGRAF destino e do link de comunicação, o destino é dividido em duas tarefas separando o trabalho de comunicação (tarefa de comunicação de isatst ou isanet) e execução do aplicativo (tarefa de núcleo isaker).
Tal arquitetura é mais flexível. Permite que o usuário execute mais que uma tarefa de comunicação ligada a uma só tarefa núcleo ou executar até 4 núcleos com a mesma tarefa de comunicação. Isto facilita certas integrações, como uma ligação de visualização do processo e o depurador do ambiente de trabalho sobre o mesmo aplicativo ou uma ligação simples de até 4 aplicativos diferentes através da mesma porta física.

As tarefas núcleo e comunicação são independentes e podem ser iniciadas separadamente. Só é necessário iniciar a(s) tarefa(s) núcleo antes para que inicie o seu sistema ambiente e a(s) tarefa(s) de comunicação possam se conectar.

O ISaGRAF destino multitarefa não impede a execução dos processos de fundo ou as rotinas controladas por interrupção.

C.4.2.1 Executando a tarefa núcleo: isaker

Número escravo: opção -s

Esta opção define o número escravo do destino. Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Este número escravo é utilizado pelo protocolo de ligação de comunicação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados.

Valor padrão: O número escravo padrão é 1 (o mesmo do ambiente de trabalho)

C.4.2.2 Executando a tarefa de comunicação serial: isatst

Link de comunicação e configuração: opção -t

A tarefa de comunicação isatst utiliza um link serial para a comunicação com o depurador. O nome da descrição é definido com a opção –t .

Sem valor padrão: Se esta opção não é utilizada, a comunicação com o destino não é possível. Neste caso, o erro número 7 pode ser mostrado.

Com a implementação da tarefa isatst, a comunicação utilizando um link Ethernet não está disponível.

O dispositivo de link serial é aberto no modo transferência de dados binários (sem caracteres de controle, sem XON/XOFF). Antes de iniciar o ISaGRAF, é necessário configurar os parâmetros de comunicação de forma que o usuário possa utilizar os parâmetros necessários com toda a liberdade. Ao utilizar o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho (veja o guia de usuário: Gerenciamento de programas) correspondam aos do destino.
Exemplo:
xmode /t0 baud=19200
Configura a taxa de transmissão de dados em 19.200 bauds no dispositivo /t0.

Número escravo: opção -s

Esta opção define o(s) número(s) escravo(s) do destino. Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Esta opção pode ser repetida até 4 vezes para estabelecer a ligação com até 4 diferentes núcleos. Este número escravo é utilizado pelo protocolo de ligação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados. Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Valor padrão: O número escravo padrão é 1 (o mesmo do ambiente de trabalho)

Número lógico das tarefas de comunicação: opção -c

Esta opção especifica o número lógico da tarefa de comunicação. É utilizado para o gerenciamento simultâneo de diversas tarefas de comunicação. Está compreendido de 1 até 255 e deve ser diferente para cada tarefa de comunicação.

Valor padrão: A última opção -s especificada é utilizada. O valor padrão assegura a compatibilidade com as versões ISaGRAF anteriores (3.0).

C.4.2.3 Executando a tarefa de comunicação Ethernet: isanet

Link de comunicação e configuração: opção -t

A tarefa de comunicação destino isanet utiliza o link padrão Ethernet para a comunicação com o depurador. O número de porta é especificado com a opção -t.

Sem valor padrão: Se esta opção não é utilizada, a comunicação com o destino não é possível. Neste caso, o erro número 7 pode ser mostrado.

Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Para o ISaGRAF, o destino OS-9 é o servidor e o depurador é o cliente que se conecta ao número de porta especificado.
Antes de iniciar a sua primeira sessão de depuração na Ethernet, certifique-se que seu dispositivo Ethernet OS-9 está corretamente configurado. Você pode, por exemplo, enviar um "ping" para o sistema OS-9.

Número escravo: opção -s

Esta opção define o(s) número(s) escravo(s) do destino ao qual a tarefa de comunicação está ligada. Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Esta opção pode ser repetida até 4 vezes para estabelecer a ligação com até 4 diferentes núcleos. Este número escravo é utilizado pelo protocolo de ligação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados. Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino existente (tarefas núcleo e de comunicação).

Valor padrão: O número escravo padrão é 1 (o mesmo do ambiente de trabalho)

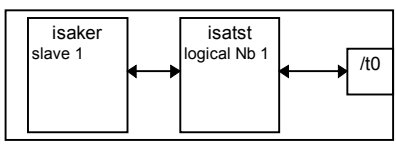
Número lógico das tarefas de comunicação: opção -c

Esta opção especifica o número lógico da tarefa de comunicação. É utilizado para o gerenciamento simultâneo de diversas tarefas de comunicação. Está compreendido de 1 até 255 e deve ser diferente para cada tarefa de comunicação.

Valor padrão: A última opção -s especificada é utilizada. O valor padrão assegura a compatibilidade com as versões ISaGRAF anteriores (3.0).

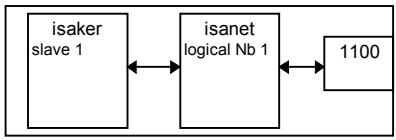
C.4.2.4 Exemplos:

isaker &
isatst -t=/t0



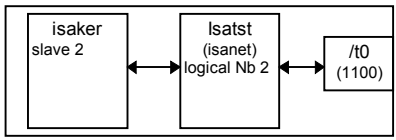
Início:
Uma tarefa núcleo ISaGRAF com o número escravo padrão (1).
Uma tarefa de comunicação serial ISaGRAF, na porta /t0, ligado ao número escravo padrão (1), e com o número lógico padrão (último número escravo especificado = padrão = 1).

isaker &
isanet -t=1100



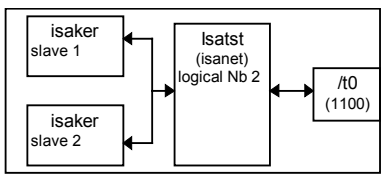
Início:
Uma tarefa núcleo ISaGRAF com o número escravo padrão (1).
Uma tarefa de comunicação Ethernet ISaGRAF, na porta de número 1100, ligada ao número escravo padrão (1), e com o número lógico padrão (último número escravo especificado = padrão = 1).

isaker -s=2 &
isatst -t=/t0 -s=2 (respectivamente isanet -t=1100 -s=2)



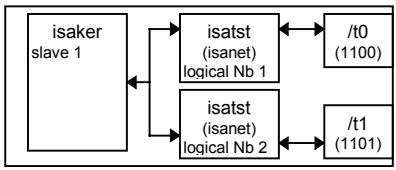
Início:
Uma tarefa núcleo ISaGRAF com número escravo 2.
Uma tarefa de comunicação serial ISaGRAF (Ethernet), na porta /t0 (porta número 1100), ligada ao número escravo 2, e com o número lógico padrão (último número escravo especificado = 2).

Isaker -s=1 &
isaker -s=2 &
isatst -t=/t0 -s=1 -s=2 (respectivamente isanet -t=1100 -s=1 -s=2)



Início:
Uma tarefa núcleo ISaGRAF com número escravo 1.
Uma tarefa núcleo ISaGRAF com número escravo 2.
Uma tarefa de comunicação serial ISaGRAF (Ethernet), na porta /t0 (número da porta 1100), ligado aos números escravos 1 e 2, e com o número lógico padrão (último número escravo especificado = 2).

Isaker -s=1 &
isatst -t=/t0 -s=1 -c=1 & (respectivamente isanet -t=1100 -s=1 -c=1 &)
isatst -t=/t1 -s=1 -c=2 (respectivamente isanet -t=1101 -s=1 -c=2)



Início:
Uma tarefa núcleo ISaGRAF com número escravo 1.
Uma tarefa de comunicação serial ISaGRAF (Ethernet), na porta /t0 (número da porta 1100), ligada ao número escravo 1, e com o número lógico 1.
Uma tarefa de comunicação serial ISaGRAF (Ethernet), na porta /t1 (número da porta 1101), ligada ao número escravo 1, e com o número lógico 2.
Observação:
As tarefas Serial e Ethernet podem ser misturadas.

C.4.3 Recursos específicos

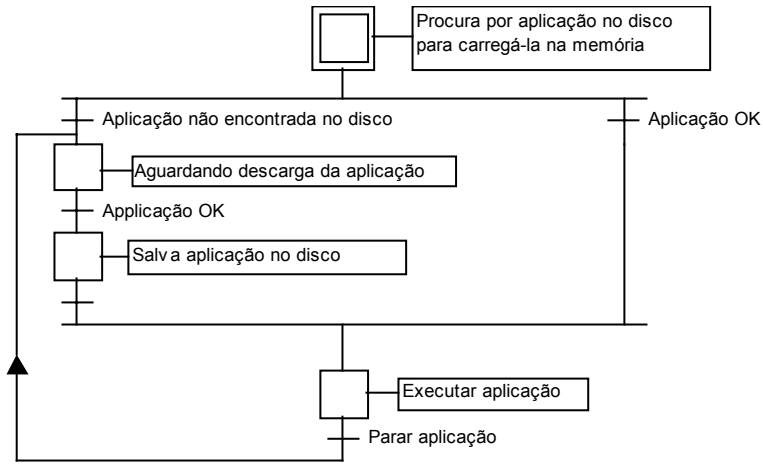
- ***Link de comunicação***

O Gerenciamento Serial de Caracteres OS-9 é muito flexível. A maioria dos dispositivos físicos bidirecionais suportados pelo Microware podem ser utilizados:

Exemplo:
A ligação serial pode ser um caminho de rede para uma porta física localizada em outra CPU.
Sendo assim, a opção -t pode por exemplo ser utilizada da seguinte forma: -t=/nr/MASTER/t0
Aqui, o link de comunicação é deportado sobre uma CPU chamada MASTER em uma rede ramnet. A porta física utilizada é /t0.

- ***Iniciando o ISaGRAF***

Quando o destino é iniciado, o seguinte algorítmo é executado.



- **Definições**

O código do aplicativo é a base dos dados binários gerados e carregados pelo ambiente de trabalho e em seguida executado pelo destino. Pode ser completado pela tabela de símbolos.
A tabela de símbolos de um aplicativo é uma base de dados ASCII gerado e carregado pelo ambiente de trabalho. Esta tabela faz a ligação entre os objetos simbólicos e os objetos internos do destino. Não é necessário no destino exceto para o gerenciamento de símbolos específicos do usuário. Para maiores informações sobre a tabela de símbolos veja o manual do usuário:Técnicas de programação avançada.

- **Objetos ISaGRAF OS-9 e Multi-aplicativo**

Todos os nomes públicos ISaGRAF começam com '**ISAxn**' em que **x** é o número escravo núcleo e **n** é um número de espaço tendo um significado específico, exceto **ISAy3** em que **y** é o número lógico de uma tarefa de comunicação em uma implementação multitarefa.
Diferentes aplicativos(tarefas núcleo e comunicação) podem ser executados simultaneamente por uma CPU, desde que possuam diferentes números escravos e números lógicos de tarefa de comunicação. Quando diferentes aplicativos são executados, o usuário deve tomar cuidado com o compartilhamento de acesso a alguns objetos do aplicativo, do tipo placas de E/S. É possível que diferentes aplicativos (núcleos) utilizem placas físicas distintas a menos que um servidor E/S ou sinal seja implementado através da unidade E/S.

Nomes dos objetos OS-9:
Arquivos de disco:

ISAx1	cópia de segurança de arquivo do código do aplicativo ISaGRAF
ISAx6	cópia de segurança de arquivo dos símbolos do aplicativo ISaGRAF

Módulos de Memória:

ISAx0	dados do sistema do núcleo ISaGRAF
ISAx1	código do aplicativo ISaGRAF
ISAx2	base de dados em tempo real do núcleo ISaGRAF
ISAy3	"buffer" de comutação de dados de comunicação ISaGRAF
ISAx4	modificações online 1 do código do aplicativo ISaGRAF
ISAx5	modificações online 2 do código do aplicativo ISaGRAF
ISAx6	símbolos do aplicativo ISaGRAF

Portanto, o usuário deve tomar cuidado para não utilizar os mesmos nomes dos objetos.

- **Cópia de segurança do aplicativo**

Quando um novo aplicativo é carregado no destino, o código do aplicativo é gravado no diretório corrente do destino com o nome do arquivo:

ISAx1 arquivo cópia de segurança do código do aplicativo ISaGRAF (x é o número escravo)

Se a tabela de símbolos do aplicativo é carregada antes, também é salva no diretório corrente do destino com o nome:

ISAx6 arquivo de cópia de segurança dos símbolos do aplicativo ISaGRAF (x é o número escravo)

Quando o ISaGRAF destino é iniciado, estes arquivos (código e símbolos do aplicativo) são localizados no diretório corrente e carregados na memória como módulos de dados.

Se nenhum arquivo de símbolos está disponível, o destino inicia a execução do código de aplicativo sem os símbolos.

Se nenhum código do aplicativo está disponível na memória, o destino aguarda o carregamento de um aplicativo.

Para iniciar o destino com um aplicativo específico no momento da alimentação, sem utilizar o depurador, existem duas maneiras:

- Copiar os arquivos a partir do PC Host, no qual o ambiente de trabalho está instalado, com a ajuda de uma ferramenta de transferência de dados diretamente no diretório corrente do destino. Você também pode utilizar a barra de ferramentas do ambiente de trabalho (veja o manual do usuário: Gerenciamento de programas) para facilitar estas manipulações.

- Armazenar o código do aplicativo (e se necessário, a tabela de símbolos) em uma memória não volátil (do tipo PROM ou EPROM), à partir dos arquivos no PC host no qual o ambiente de trabalho está instalado, utilizando suas próprias ferramentas.

  Se você desejar ter um acesso rápido ou um gerenciamento dos breakpoints, por exemplo, no momento da alimentação do sistema você pode carregar o código do aplicativo (e eventualmente a tabela de símbolos) como módulo de memória **ISAx1** (e se necessário **ISAx6**) da PROM para a RAM utilizando suas próprias ferramentas.

  ATENÇÃO:
  O gerenciamento do breakpoint do depurador ISaGRAF não pode ser executado corretamente se o módulo do código do aplicativo não está acessível para escrita. Isto não é um problema se o seu aplicativo tiver sido completamente testado anteriormente.

No PC Host, se o ambiente de trabalho ISaGRAF está instalado no diretório padrão \ISAWIN, o nome do arquivo de código do aplicativo do projeto MYPROJ é:

\ISAWIN\APL\MYPROJ\appli.x6m (correspondendo a isax1 no destino).

e o nome do arquivo de símbolos do aplicativo do projeto MYPROJ é:

\ISAWIN\APL\MYPROJ\appli.tst (correspondendo a isax6 no destino).

Gerenciamento de erros e mensagens de saída

O software do ISaGRAF destino permite o gerenciamento da detecção de erros. Você encontrará a lista de mensagens de erro com suas descrições no anexo.

A detecção de erros é tratada da seguinte maneira:
- Um erro é composto de um erro e de um número de argumento que são enviados à rotina de erros do ISaGRAF
- Se a detecção dos erros for validada nos parâmetros de execução do ambiente de trabalho, o erro é processado. Se não, a informação é perdida e o gerenciamento de erros interrompido.

Quando o erro é processado:
- Número do erro (valor decimal) e do argumento (valor hexadecimal) são mostrados na saída padrão stdout .
- Número do erro e do argumento são enviados para um "buffer" de erro do tipo FIFO de modo a serem recuperados mais tarde. O tamanho do "buffer" é definido nas opções "parâmetros de execução" do ambiente de trabalho. Quando o "buffer" está cheio, a cada novo erro que chega, o mais antigo é perdido.
- Erros podem ser chamados ou pelo depurador ou pelo aplicativo em execução utilizando a chamada SYSTEM (veja o manual do usuário).

Quando o depurador detecta um erro, uma mensagem descrevendo o erro é mostrada na janela de erro. Dependendo do contexto do aplicativo (em execução ou não) o depurador pode mostrar o nome do objeto (variável ou programa) do qual o erro vem, ou o erro do argumento (valor decimal) entre parênteses [x] que pode ter um significado diferente para cada erro.

Uma mensagem de saudação e os valores dos erros são mostrados na saída stdout quando o destino inicia e quando um erro é detectado. Se esta tela não é desejada no canal de saída padrão, um comando de redirecionamento pode ser utilizado, tal como:

prog_name [options] **>>>/nil**

Duração do ciclo, comportamento e prioridade das tarefas

- No fim de um ciclo ISaGRAF, justo antes de começar um novo ciclo, o seguinte algorítmo é executado:

  Se um tempo de ciclo é especificado (à partir do ambiente de trabalho, veja o manual do usuário: Gerenciamento de programas) então a CPU é liberada para o período de tempo restante (tempo de ciclo especificado – tempo de ciclo do aplicativo corrente). Se o período de tempo é negativo, um transbordo é gerado e a CPU é liberada por um tic para forçar o scheduling.

  Se o tempo de ciclo não é especificado, ou o tempo restante é menor ou igual a 1 tic ou igual a zero, então a CPU é liberada por 1 tic para forçar o scheduling.

  A precisão de tempo do destino corresponde a um tic padrão do sistema OS-9.

  Um tempo de ciclo especificado é normalmente utilizado para sincronizar ciclos ou para colocar a CPU à disposição das outras tarefas executadas no sistema OS-9.

- A tarefa de comunicação está no estado "dormir" enquanto não há dados de entrada através do link de comunicação. Quando necessário, esta tarefa obtém as informações sobre o aplicativo corrente com a ajuda do protocolo pergunta/resposta com a tarefa núcleo. A tarefa de comunicação envia

uma pergunta para o núcleo. No fim do ciclo (para obter uma imagem síncrona do aplicativo) o núcleo dá uma resposta à tarefa de comunicação.

As tarefas ISaGRAF não modificam as prioridades que lhes foram dadas. O usuário está livre para ajustar estas prioridades de acordo com o comportamento das tarefas ISaGRAF descritas acima e as necessidades globais do aplicativo.
Por exemplo, para assegurar que o ISaGRAF não está ocupado por uma tarefa de prioridade inferior, pode-se modificar os parâmetros de gerenciamento de tarefas OS-9 tais como **MIN_AGE** e **MAX_AGE**.

Modo terminal

O protocolo de comunicação serial destino reconhece uma seqüência de 3 caracteres retorno de carro-CR ($0D) e em seguida inicia uma tarefa OS-9 shell, se está disponível, no dispositivo de link serial.
Isto permite obter um prompt OS-9 shell em qualquer terminal, utilizando o link serial do ISaGRAF destino.

Exemplo:
À partir do PC host:
- Feche o depurador ISaGRAF.
- Comece uma sessão Terminal Windows (grupo acessórios) com os parâmetros de comunicação corretos
- Pressione 3 retorno de carro

Você está agora conectado a um OS-9 Shell
- Digite **logout** para sair do modo terminal.

ATENÇÃO:
Deve-se sempre sair de uma sessão modo terminal de uma maneira própria utilizando o logout e nada mais, senão a próxima conexão com o ambiente de trabalho não terá sucesso.

C.5 Iniciando com o ISaGRAF VxWorks destino

Para a execução do(s) ISaGRAF destino(s), deve-se executar alguns comandos sobre o sistema VxWorks, para definir a configuração do ambiente e iniciar o(s) ISaGRAF destino(s). Todos estes comandos podem ser iniciados à partir de um arquivo Script. Eles são descritos nos próximos capítulos.

C.5.1 O gerenciamento dos recursos de sistema: isassr.o

Este módulo é necessário para todas as configurações do ISaGRAF destino e deve ser carregado em primeiro lugar. Permite o gerenciamento dos recursos do sistema de execução multitarefa.

C.5.2 Recursos comuns ao isa.o, isakerse.o e isakeret.o

Para executar o ISaGRAF, um destes módulos deve ser carregado.

isa.o:	permite iniciar o ISaGRAF mono-tarefa destino (somente link de comunicação serial).
Isakerse.o:	permite iniciar o ISaGRAF multitarefa destino (somente link de comunicação serial).
Isakeret.o:	permite iniciar o ISaGRAF multitarefa destino (link de comunicação serial e/ou Ethernet)

Estes módulos são descritos em detalhes nos próximos capítulos

Configuração do link de comunicação serial

O ISaGRAF destino basicamente utiliza um link serial para a comunicação com o depurador. Quando este link está aberto, nenhuma configuração é executada no dispositivo de link serial especificado pelo ISaGRAF destino. Portanto, o usuário está totalmente livre para utilizar os parâmetros necessários. Entretanto, um modo de transferência de dados binários (modo RAW) é necessário. Por isto, a sub-rotina *ISAMOD ()* está disponível.

```
uchar ISAMOD
  (
  char *desc,          /* nome da unidade serial          */
  uint32 baudrate      /* taxa de transmissão de dados */
  )
```

Descrição:
Configura o dispositivo de link serial para uma transferência de dados binários com uma taxa de transmissão especificada (bauds)

Valor de retorno:
0 se não há erros, BAD_RET em caso de erros

Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Velocidade do relógio do sistema

A variável global CLKRATE (uint32) deve ser inicializada em função do relógio do sistema VxWorks. Você pode utilizar:

CLKRATE = sysClkRateGet ()
O valor padrão de CLKRATE é 60Hz.

C.5.3 Executando o ISaGRAF mono-tarefa: isa.o

O ISaGRAF destino pode ser executado como uma única tarefa. Mas as operações em tal configuração podem ser críticas. Por exemplo, é recomendado não sobrecarregar o link de comunicação para garantir o bom desempenho. No sistema multitarefa VxWorks, diferentes ISaGRAF destino mono-tarefas podem ser executados na mesma CPU desde que possuam diferentes números escravos e portas de comunicação.
Esta implementação mono-tarefa foi projetada principalmente para plataformas de hardware simples como placas de baixo custo ou PCs MS-DOS ou para criar um primeiro protótipo para a abordagem de uma nova plataforma. Por esta razão, a implementação ISaGRAF destino multitarefa seja a preferida.
O ISaGRAF destino mono-tarefa não impede a execução de processos de fundo ou de rotinas controladas por interrupção.

Registro do(s) número(s) escravo(s)

Um ISaGRAF destino é caracterizado por seu número escravo. Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Este número escravo é utilizado pelo protocolo de ligação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados. Portanto, antes de iniciar a(s) tarefa(s) ISaGRAF destino(s), ela(s) deve(m) ser registrada(s). Por isto, a sub-rotina *isa_register_slave()* está disponível.

```
uchar isa_register_slave
  (
  uchar slave      /* número escravo */
  )
```

Descrição:
Registra um novo escravo no sistema de gerenciamento multitarefas.

Valor de retorno:
0 se não há erros, BAD_RET em caso de erros

Unidade de armazenamento do arquivo cópia de segurança do aplicativo

A variável global TSK_FUNIT (char *) pode ser inicializada por uma cadeia de caracteres (“string”) contendo o caminho da unidade de armazenamento do arquivo cópia de segurança. O ISaGRAF destino simplesmente utiliza as rotinas padrões de gerenciamento de arquivos fopen, fread, fwrite, fclose para cópia de segurança do arquivo do aplicativo.
O valor padrão é uma “string” vazia ("") para especificar que não há nenhuma unidade de armazenamento.

Exemplo:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"
Especifica ISaGRAF\target\apl\, na raíz da unidade C:, sobre o PC *host_name*, como diretório de cópia de segurança do arquivo do aplicativo. Cuidado para não se esquecer da última barra (/) caso contrário a cópia de segurança é feita no diretório ISaGRAF\target\ com os nomes dos arquivos prefixados como apl .

Se necessário, esta variável pode ser configurada para diferentes unidades, para cada destino, antes de cada início.
Você encontrará mais informações sobre o arquivo cópia de segurança no item “Recursos específicos”, no capítulo “Cópia de segurança do aplicativo”.

Controle de fim de ciclo

A variável TSK_NBTCKSCHED (uint 32) pode ser configurada com um valor especificando um tempo de retardo do tic e utilizado pelo ISaGRAF destino no fim do ciclo.
O valor padrão é 0 (mesma prioridade da tarefa).
Se necessário, esta variável pode ser configurada para diferentes unidades, para cada destino, antes de cada início.
Você encontrará mais informações no item “Recursos específicos”, no capítulo “Duração do ciclo, comportamento e prioridades das tarefas”.

Iniciando o ISaGRAF destino

Uma vez que a configuração do ambiente está definida, a última etapa consiste em iniciar o(s) ISaGRAF destino(s): isa_main.

```
uchar isa_main
  (
  uchar slave,        /* Número escravo              */
  char *com           /* Nome do dispositivo serial */
  )
```

Descrição:
Inicia uma tarefa ISaGRAF destino.

Valor de retorno:
Retorna um valor diferente de zero se ocorre algum erro.

O número escravo é o mesmo já discutido no item “Registro do(s) número(s) escravo(s)”.
É possível iniciar mais que um destino com a condição de que os números escravos de suas portas de comunicação sejam diferentes.
Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho do escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem ao de um destino existente.

Exemplo

Este exemplo mostra como iniciar um ISaGRAF mono-tarefa destino com o número escravo 1 e com o dispositivo /tyCo/1 para o link serial.
O diretório host corrente é aquele em que o destino está instalado.

Carregar o módulo isassr.o
ld < RELS/isassr.o

Carregar o módulo isa.o
ld < CMDS/isa.o

Configuração da comunicação serial

ISAMOD ("/tyCo/1", 19200)

Velocidade do relógio do sistema
CLKRATE = sysClkRateGet ()

Registro do escravo
isa_register_slave (1)

Unidade de armazenamento de arquivo (pode ser pulada no caso de valor padrão)
TSK_FUNIT = ""

Controle de fim de ciclo (pode ser pulada no caso de valor padrão)
TSK_NBTCKSCHED = 0

Início do ISaGRAF destino
sp (isa_main, 1, "/tyCo/1")

C.5.4 Executando o ISaGRAF multitarefas: isakerse.o e isakeret.o

Para melhorar o tempo de resposta do ISaGRAF núcleo e o link de comunicação, o destino é dividido em duas tarefas separando o trabalho de comunicação (tarefas de comunicação) e a execução do aplicativo (tarefa de núcleo).
Tal arquitetura é mais flexível. Permite que o usuário execute mais que uma tarefa de comunicação ligada a uma só tarefa núcleo ou executar até 4 núcleos com a mesma tarefa de comunicação. Isto facilita certas integrações, como uma ligação de visualização do processo e o depurador do ambiente de trabalho sobre o mesmo aplicativo ou uma ligação simples de até 4 aplicativos diferentes através da mesma porta física.

As tarefas núcleo e comunicação são independentes e podem ser iniciadas separadamente. Só é necessário iniciar a(s) tarefa(s) núcleo antes para que inicie o seu sistema ambiente e a(s) tarefa(s) de comunicação possam se conectar.

O ISaGRAF destino multitarefa não impede a execução dos processos de fundo ou as rotinas controladas por interrupção.

Dois módulos estão disponíveis dependendo da capacidade de comunicação do hardware:
- Núcleo e link serial: isakerse.o
Este módulo permite iniciar a(s) tarefa(s) núcleo e a(s) tarefa(s) de comunicação serial.
- Núcleo e link serial e/ou Ethernet: isakeret.o
Este módulo permite iniciar a(s) tarefa(s) núcleo e a(s) tarefa(s) de comunicação serial e/ou Ethernet.

O início do ISaGRAF é o mesmo para os dois módulos isakerse.o e isakeret.o, exceto para o isakeret.o, você pode especificar, seja o nome de um dispositivo serial, seja o número de porta para o link Ethernet, como parâmetro para o dispositivo de comunicação quando iniciando a(s) tarefa(s) de comunicação ISaGRAF : tst_main_ex (veja a seguir).

Para o ISaGRAF, o destino VxWorks é o servidor e o depurador é o cliente que se conecta ao número de porta especificado.

- ***Registro de núcleo(s)***

Um ISaGRAF núcleo é caracterizado por seu número escravo . Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Este número escravo é utilizado pelo protocolo de ligação de comunicação e pela(s) tarefa(s) de comunicação ligada(s) ao núcleo. Serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são executados. Portanto, antes de iniciar a(s) tarefa(s) ISaGRAF núcleo(s), ela(s) deve(m) ser registrada(s). Por isto, a sub-rotina *isa_register_slave()* está disponível.

```
uchar isa_register_slave
  (
  uchar slave      /* número escravo */
  )
```

Descrição:
Registra um novo escravo no sistema de gerenciamento de multitarefas.

Valor de retorno:
0 se não há erros, BAD_RET em caso de erros

Registro de tarefa(s) de comunicação

Uma tarefa de comunicação ISaGRAF é caracterizada por seu número lógico. É utilizado para gerenciar diversas tarefas de comunicação ao mesmo tempo. Está compreendido entre 1 até 255, e deve ser diferente para cada tarefa de comunicação. Portanto, antes de iniciar a(s) tarefa(s) de comunicação ISaGRAF, ela(s) deve(m) ser registrada(s). Por isto, a sub-rotina *isa_register_com()* está disponível.

```
uchar isa_register_com
  (
  uchar com_id        /* identificador da tarefa de comunicação */
  )
```

Descripção:
Registra uma nova tarefa de comunicação no sistema de gerenciamento multitarefas.

Valor de retorno:
0 se não há erros, BAD_RET em caso de erros

Unidade de armazenamento do arquivo cópia de segurança do aplicativo

A variável global TSK_FUNIT (char *) pode ser inicializada por uma cadeia de caracteres (“string”) contendo o caminho da unidade de armazenamento do arquivo cópia de segurança. O ISaGRAF destino simplesmente utiliza as rotinas padrões de gerenciamento de arquivos fopen, fread, fwrite, fclose para cópia de segurança do arquivo do aplicativo.
O valor padrão é uma “string” vazia ("") para especificar que não há nenhuma unidade de armazenamento.

Exemplo:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"
Especifica ISaGRAF\target\apl\, na raíz da unidade C:, sobre o PC *host_name*, como diretório de cópia de segurança do arquivo do aplicativo. Cuidado para não se esquecer da última barra (/) caso contrário

a cópia de segurança é feita no diretório ISaGRAF\target\ com os nomes dos arquivos prefixados como apl .

Se necessário, esta variável pode ser configurada para diferentes unidades, para cada destino, antes de cada início.
Você encontrará mais informações sobre o arquivo cópia de segurança no item "Recursos específicos", no capítulo "Cópia de segurança do aplicativo".

Controle de fim de ciclo

A variável TSK_NBTCKSCHED (uint 32) pode ser configurada com um valor especificando um tempo de retardo do tic e utilizado pelo ISaGRAF destino no fim do ciclo.
O valor padrão é 0 (mesma prioridade da tarefa).
Se necessário, esta variável pode ser configurada para diferentes valores, para cada núcleo, antes do início de cada núcleo.
Você encontrará mais informações no item "Recursos específicos", no capítulo "Duração do ciclo, comportamento e prioridades das tarefas".

Iniciando o ISaGRAF núcleo

Uma vez que a configuração do ambiente está definida, uma das últimas etapas consiste em iniciar o(s) ISaGRAF núcleo(s): isa_main.

```
uchar isa_main
  (
  uchar slave,      /* Número escravo                */
  char *com         /* NOT USED "string" vazia é OK */
  )
```

Descrição:
Inicia uma tarefa ISaGRAF núcleo

Valor de retorno:
Retorna um valor diferente de zero se ocorre algum erro.

O número escravo é o mesmo já discutido no item "Registro do(s) número(s) escravo(s)".
É possível iniciar mais que um núcleo com a condição de que os seus números escravos sejam diferentes.

Iniciando a tarefa de comunicação ISaGRAF

Uma vez que a configuração do ambiente está definida, uma das últimas etapas consiste em iniciar a(s) tarefa(s) de comunicação ISaGRAF : tst_main_ex.

```
uchar tst_main_ex
  (
  char *com,        /* Nome do dispositivo de comunicação */
  uchar *slave,     /* Localização de um campo de 4 Bytes especificando o(s) núcleo(s) escravo
                       para ligar a */
  uchar com_id      /* identificador da tarefa de comunicação */
  )
```

Descrição:
Inicia uma tarefa de comunicação SaGRAF

Valor de retorno:
Retorna um valor diferente de zero se ocorre algum erro.

O campo de 4 Bytes especifica o(s) escravo(s) núcleo ao(s) qual(is) a tarefa de comunicação está ligada. Se menos que 4 escravos núcleo são necessários , o campo deve ser completado com zeros. Uma vez que a tarefa é iniciada, este campo não é mais necessário.
O nome do dispositivo de comunicação corresponde ao nome do dispositivo serial a ser utilizado para o link de comunicação.
É possível iniciar mais que uma tarefa de comunicação com a condição de que seus identificadores de tarefas sejam diferentes.
Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros do link de comunicação do ambiente de trabalho (veja o manual do usuário : Gerenciamento de Programas) correspondem ao de um destino existente (núcleo e tarefas de comunicação).

Exemplo:

Este exemplo mostra como iniciar:
Uma tarefa ISaGRAF núcleo com número escravo 1.
Uma tarefa de comunicação ISaGRAF identificada com o número 1, ligada ao núcleo escravo 1 e com o dispositivo /tyCo/1 para o link serial.
Uma tarefa de comunicação ISaGRAF identificada com o número 2, ligada ao núcleo escravo 1 e com o número de porta 1100 para o link de comunicação Ethernet.
O diretório host corrente é aquele em que o destino está instalado.

Carregar o módulo isassr.o
ld < RELS/isassr.o

Carregar o módulo isakeret.o (você pode carregar isakerse.o quando nenhum link de comunicação Ethernet é necessário)
ld < CMDS/isakeret.o

Configuração da comunicação serial
ISAMOD ("/tyCo/1", 19200)

Velocidade do relógio do sistema
CLKRATE = sysClkRateGet ()

Registro do escravo
isa_register_slave (1)

Registro da comunicação
isa_register_com (1)
isa_register_com (2)

Unidade de armazenamento de arquivo (pode ser pulada no caso de valor padrão)
TSK_FUNIT = ""

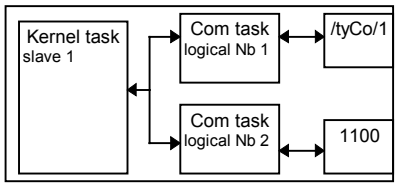
Controle de fim de ciclo (pode ser pulada no caso de valor padrão)
TSK_NBTCKSCHED = 0

Início do ISaGRAF núcleo
sp (isa_main, 1, "")

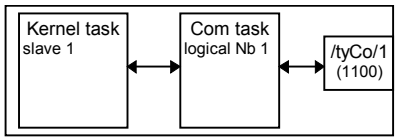
Tarefa de comunicação, links escravos
SlavesLink = 0x01000000

Início das tarefas de comunicação ISaGRAF
sp (tst_main_ex, "/tyCo/1", &SlavesLink, 1)
sp (tst_main_ex, "1100", &SlavesLink, 2)

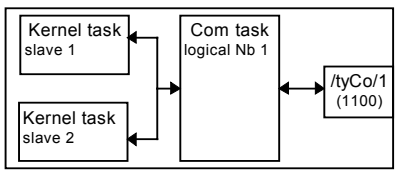
Este início corresponde à seguinte figura:



Você também pode escolher entre as seguintes configurações básicas:.



A configuração mais básica consiste em uma tarefa núcleo associada a uma tarefa de comunicação em um link serial (Ethernet).

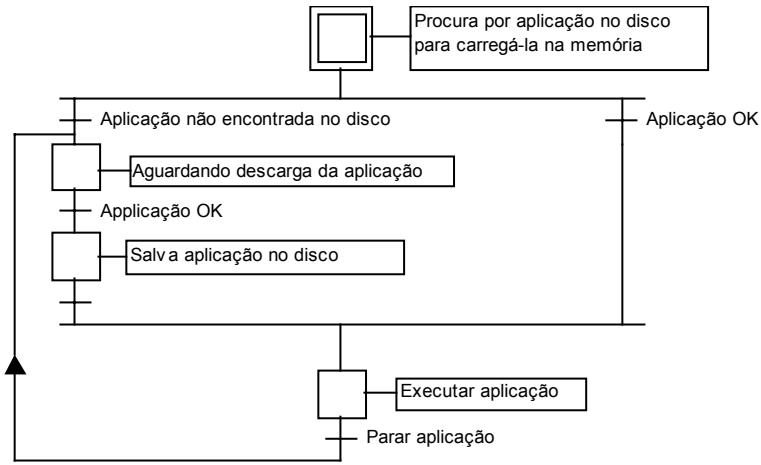


Uma outra configuração consiste em 2 núcleos associados a uma tarefa de comunicação em um link serial (Ethernet). Neste caso, SlavesLink = 0x01020000.

C.5.5 Recursos específicos

Iniciando o ISaGRAF

Quando o destino é iniciado, o seguinte algorítmo é executado.



- **Definições**

O código do aplicativo é a base dos dados binários gerados e carregados pelo ambiente de trabalho e em seguida executado pelo destino. Pode ser completado pela tabela de símbolos.
A tabela de símbolos de um aplicativo é uma base de dados ASCII gerado e carregado pelo ambiente de trabalho. Esta tabela faz a ligação entre os objetos simbólicos e os objetos internos do destino. Não é necessário no destino exceto para o gerenciamento de símbolos específicos do usuário. Para maiores informações sobre a tabela de símbolos veja o manual do usuário:Técnicas de programação avançada.
O caminho da unidade de disco é especificado no início do ISaGRAF destino utilizando a variável global TSK_FUNIT (valor padrão = "" para especificar que não há nenhuma unidade de disco).

- **ISaGRAF Multi-aplicativos**

Diferentes aplicativos (tarefas núcleo e comunicação) podem ser executados simultaneamente por uma CPU, desde que possuam diferentes números escravos e números lógicos de tarefa de comunicação. Quando diferentes aplicativos são executados, o usuário deve tomar cuidado com o compartilhamento de acesso a alguns objetos do aplicativo, do tipo placas de E/S. É possível que diferentes aplicativos (núcleos) utilizem placas físicas distintas a menos que um servidor E/S ou sinal seja implementado através da unidade E/S.

- **Cópia de segurança do aplicativo**

Quando um novo aplicativo é carregado do depurador do ambiente de trabalho para o destino, o código do aplicativo é gravado (o destino utiliza as rotinas padrões de gerenciamento de arquivos fopen,...) com o nome de arquivo:

*path***ISAx1** arquivo cópia de segurança do código do aplicativo ISaGRAF (x é o número escravo)

Se a tabela de símbolos do aplicativo é carregada antes, também é salva no diretório corrente do destino com o nome:

*path***ISAx6** arquivo de cópia de segurança dos símbolos do aplicativo ISaGRAF (x é o número escravo)

O caminho *path* é especificado no início do ISaGRAF destino utilizando a variável global TSK_FUNIT. Uma "string" vazia ("") significa que não há nenhuma unidade de disco (valor padrão).

Quando o ISaGRAF destino é iniciado, estes arquivos (código e símbolos do aplicativo) são localizados no diretório corrente e carregados na memória.

Se nenhum arquivo de símbolos está disponível na memória, o destino inicia a execução do código de aplicativo sem os símbolos.
Se nenhum código do aplicativo está disponível na memória, o destino aguarda o carregamento de um aplicativo.

Para iniciar o destino com um aplicativo específico no momento da alimentação, sem utilizar o depurador, existem duas maneiras:

- Copiar os arquivos a partir do PC Host, no qual o ambiente de trabalho está instalado, para a unidade de armazenamento do arquivo cópia de segurança do aplicativo, com a ajuda de qualquer ferramenta de transferência de dados. Você também pode utilizar a barra de ferramentas do ambiente de trabalho (veja o manual do usuário: Gerenciamento de programas) para facilitar estas manipulações.

- Armazenar o código do aplicativo (e se necessário, a tabela de símbolos) em uma memória não volátil (do tipo PROM ou EPROM), à partir dos arquivos no PC host no qual o ambiente de trabalho está instalado, utilizando suas próprias ferramentas.

  Se você desejar ter um acesso rápido ou um gerenciamento dos breakpoints, por exemplo, no momento da alimentação do sistema você pode carregar o código do aplicativo (e eventualmente a tabela de símbolos) da PROM para a RAM utilizando suas próprias ferramentas.

  Em seguida ao início do ISaGRAF (justo antes do início das tarefas) você deve especificar o(s) endereço(s) em que o código do aplicativo (e se necessário a tabela de símbolos do aplicativo) está localizado na memória. Assim, você deve iniciar a variável global SSR da seguinte maneira:
  SSR[x][1].space = *endereço do código do aplicativo*
  E se necessário:
  SSR[x][6].space = *endereço da tabela de símbolos do aplicativo*

  Desta maneira você pode escrever um procedimento curto. A variável global é declarada como uma estrutura do tipo str_ssr que é definida no arquivo tasy0ssr.h .

  ATENÇÃO:
  O gerenciamento do breakpoint do depurador ISaGRAF não pode ser executado corretamente se o código do aplicativo não está acessível para escrita. Isto não é um problema se o seu aplicativo tiver sido completamente testado anteriormente.

No PC Host, se o ambiente de trabalho ISaGRAF está instalado no diretório padrão \ISAWIN, o nome do arquivo de código do aplicativo do projeto MYPROJ é:
\ISAWIN\APL\MYPROJ\appli.x6m (correspondendo a isax1 no destino).
e o nome do arquivo de símbolos do aplicativo do projeto MYPROJ é:
\ISAWIN\APL\MYPROJ\appli.tst (correspondendo a isax6 no destino).

Gerenciamento de erros e mensagens de saída

O software do ISaGRAF destino permite o gerenciamento da detecção de erros. Você encontrará a lista de mensagens de erro com suas descrições no anexo.

A detecção de erros é tratada da seguinte maneira:
- Um erro é composto de um erro e de um número de argumento que são enviados à rotina de erros do ISaGRAF
- Se a detecção dos erros for validada nos parâmetros de execução do ambiente de trabalho, o erro é processado. Se não, a informação é perdida e o gerenciamento de erros interrompido.

Quando o erro é processado:
- Número do erro (valor decimal) e do argumento (valor hexadecimal) são mostrados na saída padrão stdout .
- Número do erro e do argumento são enviados para um "buffer" de erro do tipo FIFO de modo a serem recuperados mais tarde. O tamanho do "buffer" é definido nas opções "parâmetros de execução" do ambiente de trabalho. Quando o "buffer" está cheio, a cada novo erro que chega, o mais antigo é perdido.
- Erros podem ser chamados ou pelo depurador ou pelo aplicativo em execução utilizando a chamada SYSTEM (veja o manual do usuário).

Quando o depurador detecta um erro, uma mensagem descrevendo o erro é mostrada na janela de erro. Dependendo do contexto do aplicativo (em execução ou não) o depurador pode mostrar o nome do objeto (variável ou programa) do qual o erro vem, ou o erro do argumento (valor decimal) entre parênteses [x] que pode ter um significado diferente para cada erro.

No destino, quando um erro é detectado, os valores dos erros são mostrados na saída padrão stdout. A tela pode ser direcionada utilizando rotinas VxWorks, tais como:

ioGlobalStdSet()

ou *ioTaskStdSet()*

No último caso, nem o núcleo nem a tarefa de comunicação podem gerar erros.

Duração do ciclo, comportamento e prioridade das tarefas

- No fim de um ciclo ISaGRAF, justo antes de começar um novo ciclo, o seguinte algorítmo é executado:

  Se um tempo de ciclo é especificado (à partir do ambiente de trabalho, veja o manual do usuário: Gerenciamento de programas) então a CPU é liberada para o período de tempo restante (tempo de ciclo especificado – tempo de ciclo do aplicativo corrente). Se o período de tempo é negativo, um transbordo é gerado e a CPU é liberada para TSK_NBTCKSCHED (variável configurada no início do ISaGRAF) para forçar o scheduling.

  Se o tempo de ciclo não é especificado, ou o tempo restante é menor ou igual a 1 tic ou igual a zero, então a CPU é liberada para TSK_NBTCKSCHED tic(s) para forçar o scheduling.

  A precisão de tempo do destino corresponde a um tic padrão do sistema VxWorks.

  Um tempo de ciclo especificado é normalmente utilizado para sincronizar ciclos ou para colocar a CPU à disposição das outras tarefas executadas no sistema VxWorks.

- A tarefa de comunicação está no estado "dormir" enquanto não há dados de entrada através do link de comunicação. Quando necessário, esta tarefa obtém as informações sobre o aplicativo corrente com a ajuda do protocolo pergunta/resposta com a tarefa núcleo. A tarefa de comunicação envia

uma pergunta para o núcleo. No fim do ciclo (para obter uma imagem síncrona do aplicativo) o núcleo dá uma resposta à tarefa de comunicação.

As tarefas ISaGRAF não modificam as prioridades que lhes foram dadas. O usuário está livre para ajustar estas prioridades de acordo com o comportamento das tarefas ISaGRAF descritas acima e as necessidades globais do aplicativo.

C.6 Iniciando com o ISaGRAF NT destino

C.6.1 Executando o ISaGRAF

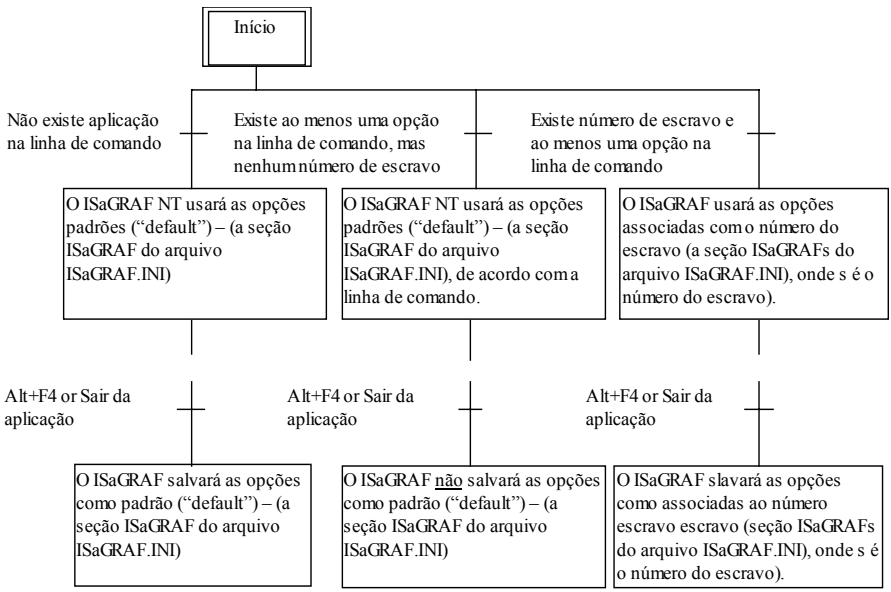
Na implementação NT, o destino é executado em um só programa: WISAKER.EXE, que pode ser iniciado diversas vezes. Permite executar tantos ISaGRAF NT destino quantos forem desejados já que cada instância tem um número escravo diferente.

O programa destino não impede a execução de rotinas de controladas por interrupção.

O software WISAKER é desenvolvido sob a plataforma Windows NT 3.51 ou superior.

C.6.2 Informações gerais sobre as opções

A opções são gravadas e restauradas de acordo com o seguinte diagrama:



Note que o arquivo ISAGRAF.INI é gravado no diretório de trabalho corrente.

Número escravo: Opção -s

Esta opção define o(s) número(s) escravo(s) do destino. Está compreendido entre 1 até 255, com exceção do número 13 ($0D). Este número escravo é utilizado pelo protocolo de ligação de comunicação e serve principalmente para distinguir entre os diferentes escravos quando muitos destinos são conectados ao mesmo ambiente de trabalho host ou quando muitos destinos são executados no mesmo PC. Quando utilizando o depurador do ambiente de trabalho, certifique-se de que os parâmetros de comunicação do ambiente de trabalho escravo (veja o manual do usuário : Gerenciamento de Programas) correspondem aos do destino.

Valor padrão: O valor padrão do número escravo é 1 ou o especificado no arquivo ISaGRAF.INI.

Exemplo:
WISAKER.EXE -s=2

Interface de usuário: Esta janela é mostrada à partir do comando "Opções/Escravo" da janela principal do ISaGRAF NT destino.

Utilizando o mouse ou as teclas de seta (para cima e para baixo) é possível modificar o valor desta opção. Para utilizá-la, o ISaGRAF NT destino deve ser reinicializado.

Conexão ("link") de comunicação e Configuração: Opção -t

O ISaGRAF destino utiliza uma link serial ou um link Ethernet para a comunicação com o depurador. O nome da porta é especificado com a opção **-t** . Como a interface de comunicação é desenvolvida para ser compatível com qualquer tipo de máquina, as portas COM1, COM2 ou COM3 podem ser utilizadas para comunicação serial, e os números de porta começando por 1100 podem ser utilizados para comunicação Ethernet.

Valor padrão: A porta de comunicação padrão é 1100 para Ethernet e COM1 para serial ou o especificado no arquivo ISaGRAF.INI.

IMPORTANTE: O link de comunicação padrão é o Ethernet.

Exemplos:
WISAKER -t=COM2
WISAKER -t=1101

Configuração serial:
Algumas opções só podem ser utilizadas se a opção -t=COMx é especificada.

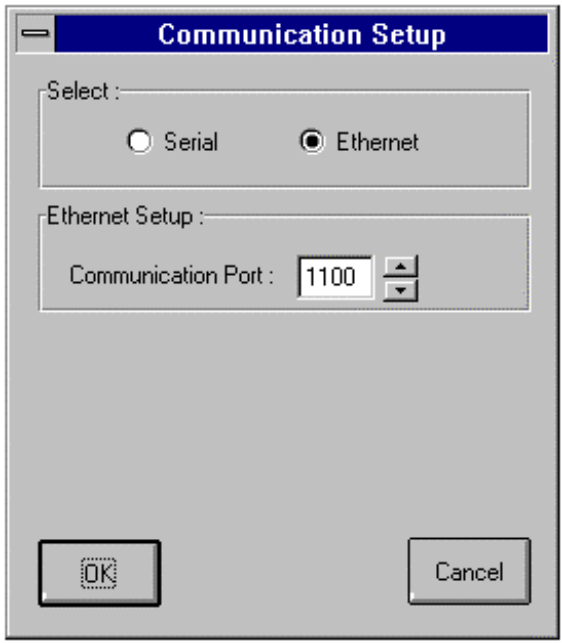
A seguir, as opções utilizadas para a configuração do link serial:

Opção	Valores	Significado
baud	600 1200 2400 4800 9600 **19200**	Taxa de transmissão de dados
parity	**n** e o	Sem paridade Par Ímpar
data	7 or **8**	Número de bits
stop	**1** or 2	Número de stop bit
flow	h **n**	Controle por hardware Sem controle

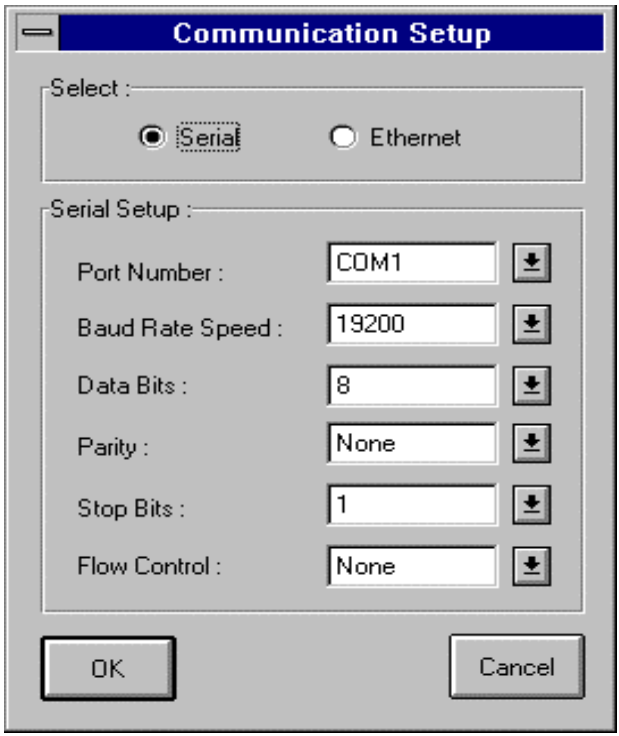
Os valores padrões são 19.200, sem paridade, 8 bits de dados, 1 stop bit, sem controle de fluxo.

Exemplo:
WISAKER -t=COM1 baud=1200 data=8 parity=n stop=2

Interface de usuário: Esta janela é mostrada à partir do comando "Opções/Comunicação" da janela principal do ISaGRAF NT destino.



É possível escolher entre a comunicação serial e a Ethernet. A comunicação Ethernet permite a modificação do número da porta. Este número de porta deve ser o mesmo da especificação do link PC-PLC do ambiente de trabalho.



Pela escolha da comunicação serial, a configuração será mostrada. Esta configuração deve ser a mesma da especificação do link PC-PLC do ambiente de trabalho.

Simulação gráfica de placas virtuais: opção -x

Se esta opção é utilizada, as placas declaradas virtuais, no editor de conexão E/S (veja parte A), são simuladas.
Os valores possíveis são 0 ou 1; 0 significa sem simulação e 1 significa simulação validada.

Valor padrão: O valor padrão é 0 ou o configurado no arquivo ISaGRAF.INI.

Exemplo:
WISAKER -x=1 simulará as placas virtuais,

Interface de usuário: O elemento do menu será validado ou não segundo o estado da opção. As placas simuladas aparecem em um painel gráfico.

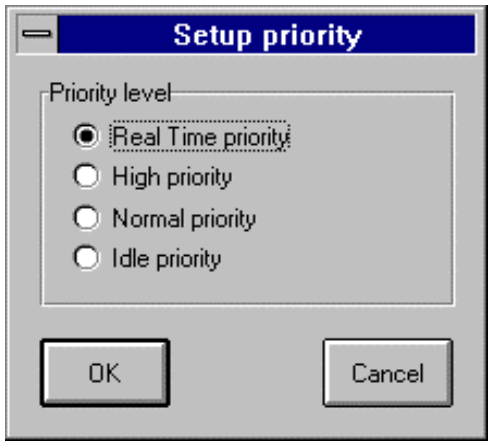
Prioridade do ISaGRAF NT destino: opção -p

Como o destino é executado sob a plataforma NT, é muito útil especificar o nível de prioridade . Por exemplo, é possível executar um aplicativo ISaGRAF de temporização crítica em um destino com uma prioridade mais alta ou executar um ou mais destinos em plano de fundo com prioridades menores.

Os valores possíveis são 0, 1, 2 ou 3. 0 é a prioridade mais alta e 3 é a prioridade mais baixa.

Exemplos:
WISAKER -p=0
WISAKER -p=1

Interface de usuário: Esta janela é mostrada à partir do comando "Opções/Prioridade" da janela principal do ISaGRAF NT destino.



A prioridade mais alta é o tempo real (real time) e a mais baixa é a inativa (idle).
0: “Real time priority” (prioridade em tempo real)
1: “High priority” (prioridade alta)
2: “Normal priority” (prioridade baixa)
3: “Idle priority” (prioridade inativa)

Exemplos:

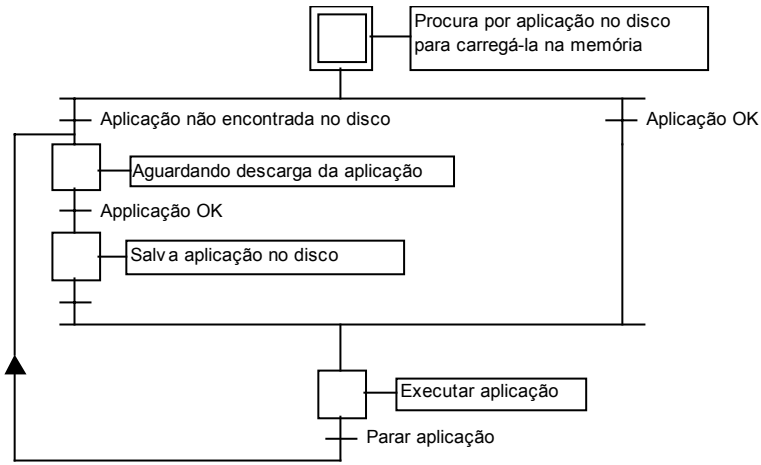
wisaker -t=COM1 Inicia o ISaGRAF destino com o valor padrão do número escravo (1) e com COM1 como porta de comunicação.

wisaker -s=3 -t=COM1 Inicia o ISaGRAF destino com o número escravo 3 e com COM1 como porta de comunicação.

C.6.3 Recursos específicos

Iniciando o ISaGRAF

Quando o destino é iniciado, o seguinte algorítmo é executado.



- **Definições**

O código do aplicativo é a base dos dados binários gerados e carregados pelo ambiente de trabalho e em seguida executado pelo destino. Pode ser completado pela tabela de símbolos.
A tabela de símbolos de um aplicativo é uma base de dados ASCII gerado e carregado pelo ambiente de trabalho. Esta tabela faz a ligação entre os objetos simbólicos e os objetos internos do destino. Não é necessário no destino exceto para o gerenciamento de símbolos específicos do usuário como por exemplo a característica DDE oo a simulação de E/Ss com a facilidade nome de símbolo. Para maiores informações sobre a tabela de símbolos veja o manual do usuário:Técnicas de programação avançada.

- **ISaGRAF Multi-aplicativos**

Diferentes aplicativos podem ser executados simultaneamente por uma CPU, desde que possuam diferentes números escravos e números lógicos de tarefa de comunicação. Quando diferentes aplicativos são executados, o usuário deve tomar cuidado com o compartilhamento de acesso a alguns objetos do aplicativo, do tipo placas de E/S. É possível que diferentes aplicativos (núcleos) utilizem placas físicas distintas a menos que um servidor E/S ou sinal seja implementado através da unidade E/S.

- **Cópia de segurança do aplicativo**

Quando um novo aplicativo é carregado do depurador do ambiente de trabalho para o destino, o código do aplicativo é gravado no diretório corrente do destino com o nome de arquivo:

ISAx1 arquivo cópia de segurança do código do aplicativo ISaGRAF (x é o número escravo)

Se a tabela de símbolos do aplicativo é carregada antes, também é salva no diretório corrente do destino com o nome de arquivo:

ISAx6 arquivo de cópia de segurança dos símbolos do aplicativo ISaGRAF (x é o número escravo)

Quando o ISaGRAF destino é iniciado, estes arquivos (código e símbolos do aplicativo) são localizados no diretório corrente e carregados na memória.

Se nenhum arquivo de símbolos está disponível, o destino inicia a execução do código de aplicativo sem os símbolos.
Se nenhum código de aplicativo está disponível, o destino aguarda a carga de um aplicativo.

Para iniciar o destino com um aplicativo específico no momento da alimentação, sem utilizar o link depurador, este arquivos podem ser copiados à partir do mesmo disco, se o ambiente de trabalho está no mesmo PC ou utilizando um disquete, diretamente para o disco do diretório corrente do destino.

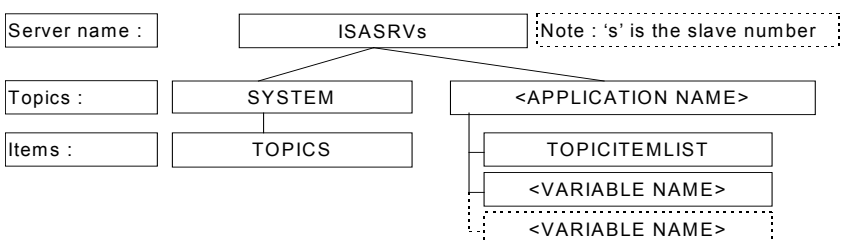
Se o ambiente de trabalho ISaGRAF está instalado no diretório padrão \ISAWIN:
o nome do arquivo de código do aplicativo do projeto MYPROJ é:
\ISAWIN\APL\MYPROJ\appli.x8m
e o nome do arquivo de símbolos do aplicativo do projeto MYPROJ é:
\ISAWIN\APL\MYPROJ\appli.tst

Exemplo:
À partir do diretório em que WISAKER.EXE está instalado, se o seguinte comando é inserido:
copy \ISAWIN\APL\MYPROJ\appli.x8m isa11
Então o arquivo WISAKER.EXE localizará e executará o aplicativo 'myproj'.

Todos estes comandos podem ser agrupados, por exemplo, em um arquivo Batch e então iniciado à partir do menu de ferramentas do ambiente de trabalho (veja o Manual do usuário: Gerenciamento de programas).

Especificações DDE

O ISaGRAF NT destino é um servidor DDE (Dynamic Data Exchange). Todo software cliente pode ser conectado ao destino para troca de variáveis. Por exemplo, MSEXCEL pode animar gráficos com valores provenientes do ISaGRAF destino via DDE.
A função DDE requer uma tabela de símbolos do aplicativo no destino.
Os assuntos DDE são definidos da seguinte forma:



« ISASRVs » é o nome do servidor DDE, 's' é o número escravo.
« SYSTEM » é um tópico padrão que dá acesso ao item « TOPICS »,
« TOPICS » dá a lista de tópicos definidos: sistema e o nome do aplicativo em execução no destino ISaGRAF NT.
« APPLICATION NAME » é o nome do aplicativo.
« TOPICITEMLIST » é a lista de itens disponíveis sob o tópico corrente, que dá a lista de variáveis acessíveis pelo DDE.
« VARIABLE NAME » é o nome de uma variável.

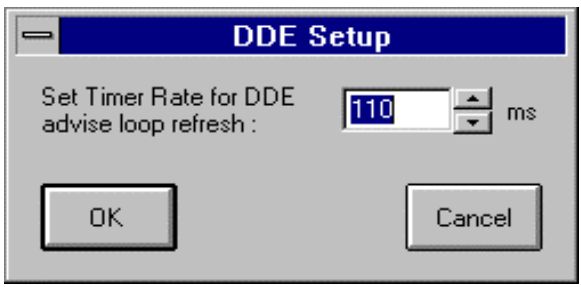
DDE Advise Loop Rate para o ISaGRAF NT destino: opção -d
O DDE cliente geralmente chama as variáveis cada vez que delas necessita. Se as variáveis de um aplicativo são muito numerosas, isto pode gastar um tempo considerável. Há um outro modo chamado de Advise Mode (advise loop), no qual o próprio servidor envia unicamente as variáveis modificadas. Desta forma, as comunicações são minimizadas e eficientes. Neste modo, as variáveis marcadas são periodicamente verificadas pelo servidor para decidir quais variáveis devem ser enviadas. Este período é chamado de DDE Advise Loop Rate.

Com esta opção, é possível especificar a taxa (em **ms**) para o DDE Advise Loop.

Valor padrão: O valor padrão é 1.000 ms ou o configurado no arquivo ISaGRAF.INI

Exemplo:
WISAKER -d=100

Interface de usuário: Esta janela é mostrada à partir do comando "Opções/DDE" da janela principal do ISaGRAF NT destino.



Gerenciamento de erros e mensagens de saída

O software do ISaGRAF destino permite o gerenciamento da detecção de erros. Você encontrará a lista de mensagens de erro com suas descrições no anexo.

A detecção de erros é tratada da seguinte maneira:
- Um erro é composto de um erro e de um número de argumento que são enviados à rotina de erros do ISaGRAF
- Se a detecção dos erros for validada nos parâmetros de execução do ambiente de trabalho, o erro é processado. Se não, a informação é perdida e a detecção de erros interrompida.

Quando o erro é processado:
- Número do erro (valor decimal) e do argumento (valor hexadecimal) são mostrados na saída (janela do WISAKER.EXE)
- Número do erro e do argumento são enviados para um “buffer” de erro do tipo FIFO de modo a serem recuperados mais tarde. O tamanho do “buffer” é definido nas opções “parâmetros de execução” do ambiente de trabalho. Quando o “buffer” está cheio, a cada novo erro que chega, o mais antigo é perdido.
- Erros podem ser chamados ou pelo depurador ou pelo aplicativo em execução utilizando a chamada SYSTEM (veja o manual do usuário).

Quando o depurador detecta um erro, uma mensagem descrevendo o erro é mostrada na janela de erro. Dependendo do contexto do aplicativo (em execução ou não) o depurador pode mostrar o nome do objeto (variável ou programa) do qual o erro vem, ou o erro do argumento (valor decimal) entre parênteses [x] que pode ter um significado diferente para cada erro.

Uma mensagem de saudação é mostrada na saída quando o destino inicia. É composta de número escravo, a configuração de comunicação e o nome do servidor DDE.

Relógio do sistema

Como o ISaGRAF NT destino é projetado para ser executado em qualquer sistema, a referência de tempo utilizada para a sincronização do ciclo bem como para a atualização das variáveis de temporização, é o tic padrão de 10 milisegundos.

Assim, não é possível obter uma precisão maior que 10ms para as variáveis de temporização. Por esta razão, um tempo de ciclo com duração menor ou igual a 10 ms e diferente de zero gera um erro de transbordo de tempo de ciclo (error 62). Veja o capítulo a seguir para mais informações.

Solicite ao seu fornecedor uma implementação específica se seu aplicativo necessita de maior precisão.

Duração do ciclo e comportamento do destino

No fim de um ciclo ISaGRAF, justo antes de começar um novo ciclo, o seguinte algorítmo é executado:

Se um tempo de ciclo é especificado (à partir do ambiente de trabalho, veja o manual do usuário: Gerenciamento de programas) então a CPU é liberada para o período de tempo restante (tempo de ciclo especificado – tempo de ciclo do aplicativo corrente). Se o período de tempo é negativo, um transbordo é gerado e a CPU é liberada por 1 tick para forçar o scheduling.

Se o tempo de ciclo não é especificado, ou o tempo restante é menor ou igual a 1 tic, então a CPU é liberada por 1 tic para forçar o scheduling.

A precisão de tempo do destino corresponde a um tic padrão do sistema Windows NT.

Um tempo de ciclo especificado é normalmente utilizado para sincronizar ciclos ou para colocar a CPU à disposição das outras tarefas executadas no sistema Windows NT.

Sair do ISaGRAF pelo teclado

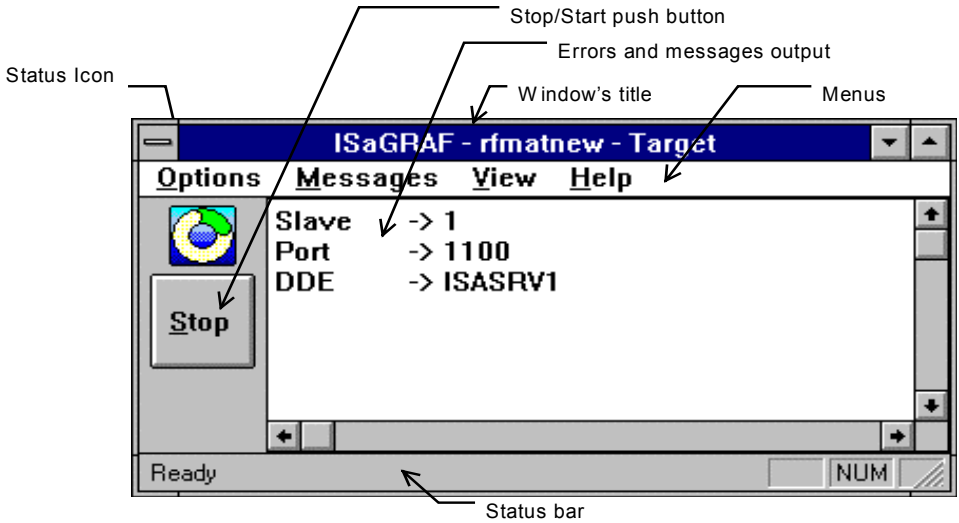
Durante o teste de um aplicativo sob condições não industriais em um PC, o usuário pode interromper o ISaGRAF através de uma combinação de teclas, evitando as paradas não desejadas. A seqüência de teclas é a seguinte:

alt + F4

Um efeito secundário perigoso devido a estas saídas rápidas é o seguinte: a interface das placas de E/S não é fechada. Por esta razão, a maneira correta de interromper o ISaGRAF destino é a seguinte:
- interromper o aplicativo à partir do depurador ou à partir do botão Iniciar/Parar (isto fechará as placas de E/S)
- interromper o ISaGRAF destino à partir do menu de sistema.

C.6.4 Interface de usuários

Esta é a interface de usuários do ISaGRAF NT destino:



Estes são os principais elementos:

a barra de título da janela
a barra de menu
o ícone do estado da execução
o botão Iniciar/Parar
a saída de erros e mensagens
a barra de estado.

O título da janela contém « ISaGRAF - name_of_appli - target », no qual name_of_appli é o nome do aplicativo corrente. Se nenhum aplicativo está sendo executado, o título contém « ISaGRAF - - Target ».

Barra de menu do ISaGRAF NT destino:

A barra de menu contém quatro menus:

Opções
Mensagens
Visualização
Ajuda

- **Menu "Opções"**

(veja também a primeira seção NT: Informações Gerais sobre as opções)
O menu "**Opções**" dá acesso às opções de execução. As seguintes opções estão disponíveis:

Número da estação escrava dá acesso à modificação do número escravo . A opção modificada será ativada somente após o próximo início do destino. Esta função não está disponível se o destino tiver sido iniciado com pelo menos uma opção na linha de comando.

Comunicação dá acesso à configuração de comunicação. A opção modificada será ativada somente após o próximo início do destino. Esta função não está disponível se o destino tiver sido iniciado com pelo menos uma opção na linha de comando (exceto a opção –s).

DDE dá acesso à modificação do DDE Advise Loop Rate. A opção modificada será ativada somente após o próximo início do destino. Esta função não está disponível se o destino tiver sido iniciado com pelo menos uma opção na linha de comando (exceto a opção –s).

Simular E/S valida a simulação das E/S. A opção modificada será ativada somente após o próximo início do destino.

Prioridade dá acesso à modificação das prioridades. A modificação da opção é ativada imediatamente.

Opções padrão recupera as opções de execução padrões mas somente para:
- Comunicação
- DDE
- coordenadas da janela na tela

As opções modificadas serão ativadas somente após o próximo início do destino. Esta função não está disponível se o destino tiver sido iniciado com pelo menos uma opção na linha de comando (exceto para opção –s).

- **Menu "Mensagens"**

O menu "**Mensagens**" serve para o gerenciamento das saídas. Contém os seguintes comandos:

Confirmar: interrompe o vermelho piscante no caso de erros ou de mensagens.

Apagar: apaga totalmente a saída.

O ícone do ISaGRAF NT destino:

O ícone reflete o estado do destino:

- o aplicativo está em execução, o símbolo gira
- não há aplicativo em execução (ou o aplicativo foi interrompido) o ícone não gira (parado)
- erros ou mensagens são apresentados na janela de saída. O centro do ícone pisca em vermelho. Para parar a pisca-pisca, selecione o item « **Reconhecer** » do menu « **Mensagens** » ou o item « **Limpar** » do mesmo menu (atenção, este comando apaga **todas** as mensagens de saída). Você encontrará mais informações sobre erros no capítulo "Gerenciamento de erros e mensagens de saída".

Os diferentes estados estão reunidos na tabela a seguir:

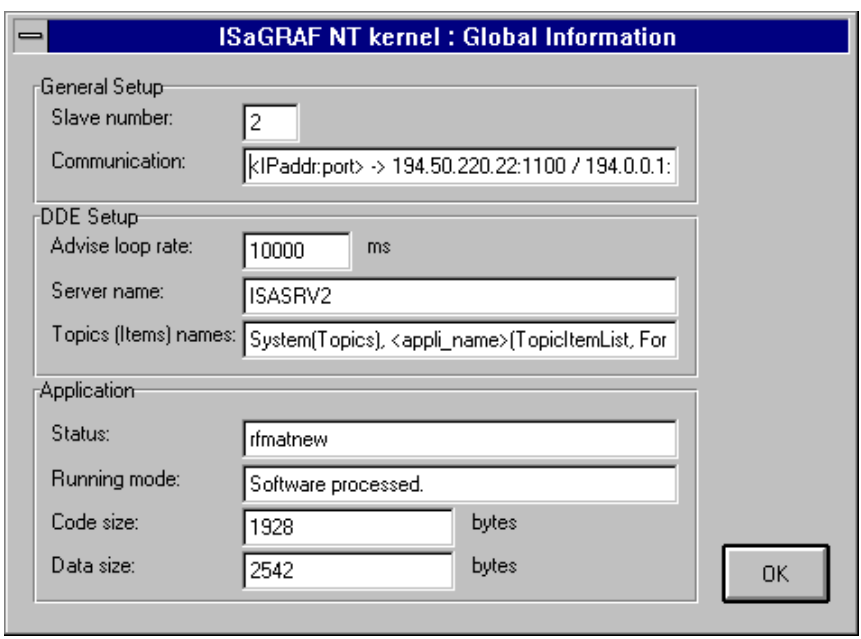
	Sem erros	**Erros ou mensagens (o miolo é vermelho)**
Aplicativo em execução		
Sem aplicativo		

O botão Iniciar/Parar do ISaGRAF NT destino:

O botão **Iniciar**/**Parar** é idêntico ao da função **Iniciar**/**parar** do depurador. O texto fixado no botão reflete o estado da execução do aplicativo. Se o aplicativo está em execução, o texto será « **Parar** », se o aplicativo está parado (ou se não há nenhum aplicativo), o texto será « **Iniciar** » (Note que se hão há nenhum aplicativo disponível e uma ação "**EXECUTAR**" é solicitada, o botão alterna para o modo "**Parar**" e então volta para o modo "**EXECUTAR**").

Informações gerais sobre o ISaGRAF NT destino

O comando "**Exibir / Informações**" apresenta a caixa de diálogo a seguir que fornece as informações gerais sobre a configuração do destino e do aplicativo corrente:



Há três tópicos:

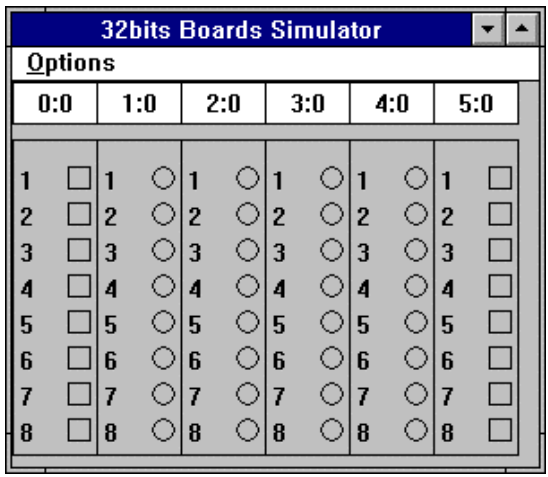
a) Configuração geral:
   - Número escravo
   - A configuração da comunicação (se o link de comunicação é o Ethernet, uma lista de endereços IP disponíveis no destino corrente é mostrada em adição ao número da porta)

b) Configuração DDE
   - Advise Loop Rate
   - Nome do servidor DDE
   - Nomes dos tópicos e dos itens DDE. São informações gerais que não refletem os valores reais. De fato, os campos entre < > devem ser preenchidos com os valores reais.

c) Aplicação
   - O estado do aplicativo que é o seu nome quando há um aplicativo em execução ou a cadeia de caracteres ("string") '**Aplicação inexistente**' quando não há um aplicativo em execução.
   - O modo de execução do aplicativo que indica se o aplicativo está sendo executado através do processador de software. Contém, neste caso, a cadeia de caracteres ("string"): « Software processado ». Ou se o aplicativo foi compilado com um compilador C. Contém, neste caso, a cadeia de caracteres ("string"): « C compilado ». Se não há nenhum aplicativo sendo executado, contém a cadeia de caracteres ("string"): « **Aplicação inexistente** ».
   - O tamanho do código em bytes. Se o modo de execução é « C compilado », este campo é zero.
   - O tamanho dos dados em bytes. É a soma dos dados internos de tempo de execução e as variáveis da base de dados.

Simulação das placas virtuais do ISaGRAF NT destino:

Quando a opção « Simulação de E/S » é selecionada, no próximo início do aplicativo a seguinte janela é apresentada:



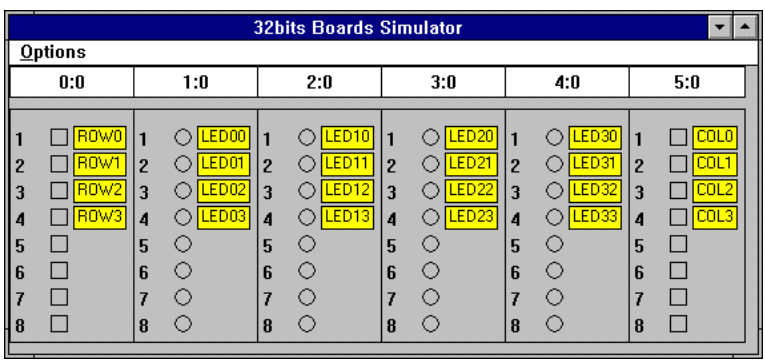
Dependendo da sua configuração de E/S, um certo número de placas e variáveis serão mostrados. Os números « s:b » no topo de cada placa representa os identificadores do slot (s) e o identificador da placa (b) Numeração à partir de zero e não pode ser modificado.
A janela 'Simulador de placas 32bits' trabalha com o estado do aplicativo **Iniciar**/**Parar**. Portanto, se há um aplicativo em execução, que possua (ou utilize) placas virtuais e a opção « Simulação de E/S » está validada, a janela é mostrada. Caso contrário, se o botão "**Parar**" é pressionado, ela é fechada. Esta janela trabalha paralelamente com as chamadas E/S.

O menu "**Opções**" contém dois itens:

Nomes das variáveis mostra os nomes das variáveis se e somente se a tabela de símbolos tiver sido carregada antes do código Tic.

Valores hexadecimais mostra cada número inteiro no formato hexadecimal ao invés do formato padrão decimal.

Os nomes das variáveis terão a seguinte aparência:



C.7 Programando em "C"

C.7.1 Apresentação

Este manual é dedicado aos usuários com boa experiência dos conceitos do ISaGRAF e ferramentas do Ambiente de Trabalho. Após o desenvolvimento de aplicativos puramente de automação utilizando **funções de conversão**, **funções "C"** e **blocos de função** das bibliotecas padrões da CJ International, o usuário pode criar outras conversões, funções e blocos de funções. Também é possível ampliar o ISaGRAF destino PLC criando novas bibliotecas para tirar melhor partido da flexibilidade do ambiente de trabalho ISaGRAF e da plataforma de hardware.

Com um sistema de desenvolvimento "C" e com uma boa prática em programação "C", este manual permitirá que o usuário personalize o seu ISaGRAF destino PLC para a melhor possibilidade de controle. As bibliotecas desenvolvidas permitem o aumento do desempenho do destino PLC e permite que o programador de automação desenvolva mais facilmente os aplicativos de alta qualidade.

A informação contida neste documento não é dedicada a um sistema destino especial. Entretanto, algumas facilidades (tais como capacidade multitarefa) não podem ser aplicadas em certos sistemas mono-tarefa.

Recursos do ambiente de trabalho ISaGRAF

O ambiente de trabalho ISaGRAF inclui diversas funções dedicadas ao gerenciamento de bibliotecas de componentes "C" e à sua incorporação nos aplicativos de automação. Para o programador de automação, uma conversão "C", uma função ou um bloco de função é uma "**caixa preta** ", completamente definida por sua interface.

O Gerenciamento de Bibliotecas ISaGRAF é utilizado para a adição de componentes nas bibliotecas existentes e a definição da interface entre a implementação "C" e a utilização destes componentes nos programas **ST**/**FBD**. O Gerenciamento de Biblioteca ISaGRAF também oferece as funções de geração automática do código fonte "C" para o desenvolvimento das conversões, funções e blocos de função, e inclui ferramentas para a edição dos arquivos fonte "C". Veja a primeira parte deste manual (**Guia do Usuário ISaGRAF**) para mais informações sobre as funções do Gerenciamento de Biblioteca.

Desenvolvimento em linguagem "C"

O Ambiente de Trabalho ISaGRAF não inclui qualquer compilador "C" ou ferramenta de compilador cruzado. O usuário deve possuir um compilador "C", dedicado ao sistema ISaGRAF destino, para integrar os novos componentes "C" ao ISaGRAF núcleo.

Se a ferramenta utiliza um sistema de desenvolvimento cruzado, o Ambiente de Trabalho ISaGRAF oferece os pontos de entrada disponíveis para executar os comandos de compilação e de edição das ligações em uma janela MS-DOS (arquivo de comando .BAT). Neste caso, o usuário deve assegurar que as ferramentas de desenvolvimento “C” possam funcionar no contexto de uma janela MS-DOS do Windows. Caso contrário, o Windows deve ser fechado antes de executar os compiladores num ambiente puramente MS-DOS.

Observações técnicas

O gerenciamento de Biblioteca ISaGRAF permite que o usuário escreva um texto de descrição para cada componente da biblioteca. Esta **observação técnica** é o guia de utilização do componente desenvolvido em "C". É dedicado ao programador de automação e descreve as conversões, funções ou blocos de funções correspondentes num aplicativo ISaGRAF.

A conversão, a função ou o bloco de função deve ser precisamente descrito numa observação técnica para que o programador de automação possa realmente utilizá-la como uma função padrão das linguagens ISaGRAF. Para uma função "C", a observação técnica deve ser descrita:

- a função detalhada processada pela função
- a descrição completa dos seus parâmetros de chamada
- o significado do seu valor de retorno
- a digitação detalhada dos seus parâmetros de chamada e valor de retorno
- o contexto do aplicativo

Para um bloco de função "C", a observação técnica deve ser descrita:

- a função detalhada processada pela ativação do bloco
- a descrição completa dos seus parâmetros de chamada
- o significado dos parâmetros de retorno
- a digitação detalhada dos seus parâmetros de chamada e valor de retorno
- o contexto do aplicativo

Para uma função de conversão, A observação técnica descreve:

- o significado exato da conversão quando utilizada com um variável de entrada
- o significado exato da conversão quando utilizada com um variável de saída
- os limites dos valores que a conversão pode processar

Observações técnicas também podem conter informações sobre:

- a identificação completa da conversão, função ou bloco de função
- qualquer informação sobre sua manutenção e atualizações
- o(s) sistema(s) destino(s) suportado(s)
- os recursos especiais multitarefa
- os serviços de sistema requeridos, memória, drivers...

C.7.2 Funções de conversão

O Ambiente de Trabalho ISaGRAF inclui um utilitário de **conversão linear** para executar conversão analógica simples de E/S no tempo de execução do ISaGRAF destino PLC. Este utilitário não necessita nenhum desenvolvimento em "C". Está limitado às funções contínuas crescentes ou decrescentes. Veja o Manual do Usuário ISaGRAF para uma descrição completa destas ferramentas.

As funções de conversão permitem que o usuário aplique qualquer conversão complexa, com operações específicas descritas em linguagem "C". Basicamente, uma função de conversão pode ser definida tanto para **Entradas** quanto para **Saídas**. Mesmo que uma direção (entrada ou saída) não seja utilizada, ela deverá ser implementada e testada antes da integração da conversão em um aplicativo, para proteger o sistema ISaGRAF núcleo de toda utilização indevida.

As funções de conversão são escritas em linguagem “C”, compiladas e ligadas com o ISaGRAF núcleo executivo. O núcleo ampliado deve ser instalado no ISaGRAF destino PLC antes de utilizar as novas funções de conversão nos aplicativos ISaGRAF. Novas funções de conversão não podem ser integradas no ISaGRAF Simulador. Os aplicativos ISaGRAF devem ser simulados **antes** da inserção de funções de conversão não padrões.

O código fonte "C" das conversões padrões escritas pelo CJ International é instalado com o Ambiente de Trabalho ISaGRAF. Podem ser utilizados como exemplos para a criação de novas funções. Recomenda-se **não modificar nunca** as funções padrões para que possam ser utilizadas em qualquer aplicativo ISaGRAF. As conversões padrões instaladas com o Ambiente de Trabalho ISaGRAF são suportadas pelo simulador ISaGRAF.

Atenção: As funções de conversão são operações **síncronas**, ativadas no tempo de execução pelo gerenciamento de E/S ISaGRAF, durante as fases de entrada ou de saída do ciclo do aplicativo. O tempo gasto com a execução de uma função de conversão é incluído no **tempo de ciclo** do aplicativo ISaGRAF. O usuário deve assegurar-se que nenhuma “operação de espera” está programada na função de conversão para preservar o mecanismo de execução do ISaGRAF núcleo.

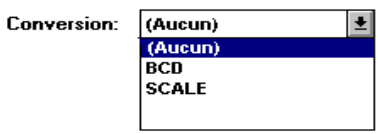
Inserindo uma função na biblioteca ISaGRAF

O Gerenciamento de Biblioteca ISaGRAF deve ser utilizado para inserir uma nova função de conversão na Biblioteca ISaGRAF. Com o ambiente de trabalho, utilize o comando "**Novo**" do menu "**Arquivos**", quando a biblioteca das funções de conversão é selecionada. Nenhum parâmetro tem que ser definido no Ambiente de Trabalho, porque as funções de conversão utilizam uma interface padrão predefinida.

Quando uma nova função de conversão tem que ser criada, sua **Observação Técnica** deve ser escrita. O quadro do código fonte "C" para a nova função de conversão é automaticamente gerada pelo Gereciamento de Biblioteca ISaGRAF.

Utilizando uma conversão em um projeto ISaGRAF

As funções de conversão definidas podem ser utilizadas para filtrar os valores de quaisquer variáveis de entrada ou de saída analógica do projeto selecionado. Para atrelar uma função de conversão a uma variável, utilize o editor de dicionário, selecione a variável de E/S analógica e abra a caixa de diálogo para a edição de seus parâmetros. Utilize o campo "**conversão**" da caixa de diálogo para selecionar a função de conversão que deve ser atrelada a esta variável analógica de E/S:



As tabelas e as funções de conversão aparecem juntas na lista. Isto implica que o mesmo nome pode ser dado a uma tabela e a uma função. Uma variável não pode ser atrelada a uma função de conversão que ainda não foi definida ou integrada no núcleo ISaGRAF.

Interface "C" padrão

A interface de uma função de conversão sempre tem o mesmo formato. Os parâmetros de chamada e de retorno são passados através de uma estrutura. Esta estrutura é definida no arquivo "**TACN0DEF.h**":

```
/*
  Nome: tacn0def.h
  Arquivo de definições padrões - conversões
*/

#define DIR_INPUT 0             /* direção = conversão de entrada */
#define DIR_OUTPUT 1            /* direção = conversão de saída */

typedef int32   T_ANA;          /* tipo ANA inteiro */
typedef float   T_REAL;         /* tipo ANA real    */

typedef struct {                /* estrutura de conversão          */
     uint16 number;             /* número de conversão (reservado) */
     uint16 direction;          /* direção da conversão            */
     T_REAL *before;            /* valor depois da conversão       */
     T_REAL *after;             /* valor antes da conversão        */
   } str_cnv;

#define ARG_BEFORE (*(arg->before))
#define ARG_AFTER  (*(arg->after))
#define DIRECTION  (arg->direction)

/* fim do arquivo */
```

A estrutura "**str_cnv**" contém todos os elementos da interface. Somente o parâmetro de uma conversão "C" é um ponteiro para tal estrutura. O campo "**número**" é o número lógico da conversão (número na biblioteca ISaGRAF) e não é jamais utilizado na programação da função de conversão.

O campo "**direção**" indica se a conversão deve ser aplicada na variável de entrada ou de saída. Contém o valor **DIR_INPUT** para uma conversão entrada, ou o valor **DIR_OUTPUT** para uma conversão de saída.

O campo "**antes**" aponta para o valor anterior à conversão. Este campo tem um significado diferente para uma conversão de entrada ou de saída. Representa o valor elétrico (lido no dispositivo de entrada) para uma conversão de entrada, quando o campo **direção** tem o valor **DIR_INPUT**. Representa o valor físico (utilizado nas equações programadas) para uma conversão de saída, quando o campo **direção** tem o valor **DIR_OUTPUT**.

O campo "**depois**" aponta para o valor após a conversão. Este campo tem um significado diferente para uma conversão de entrada ou de saída. Representa o valor físico (utilizado nas equações programadas) para uma conversão de entrada, quando o campo **direção** tem o valor **DIR_INPUT**. Representa o valor elétrico (enviado para o dispositivo de saída) para uma conversão de saída, quando o campo **direção** tem o valor **DIR_OUTPUT**.

O programador pode utilizar as definições "**ARG_BEFORE**" e "**ARG_AFTER**" para o acesso direto aos campos **antes** e **depois** da estrutura passada pela função de conversão "C". Os valores processados são **valores de ponto flutuante de precisão simples**. O resultado é convertido para um número inteiro longo (32 bits) quando a conversão é aplicada a uma variável analógica inteira. Isto significa que a mesma conversão pode ser utilizada para as variáveis de E/S analógicas, inteiras e reais.

Código fonte

Porque a função de conversão pode ser utilizada por ambas as variáveis analógicas, de entrada e de saída, o código fonte "C" é dividido em duas partes principais: a conversão de entrada e a conversão de saída. O campo **direção** é utilizado para selecionar o tipo de conversão a realizar. O Gerenciamento de Biblioteca ISaGRAF gera, automaticamente, todo o quadro do código fonte "C" da função, quando uma nova função de conversão é criada. Do mesmo modo é gerada a estrutura "**IF**" principal para a seleção da direção. A seguir, o formato padrão do código fonte de uma função de conversão:

```
/*
  função de conversão
  nome: sample
*/

#include <tasy0def.h>
#include <tacn0def.h>

void CNV_sample (str_cnv *arg)
{
        if (DIRECTION == DIR_INPUT) { /*INPUT CONV*/

        }
        else { /*OUTPUT CONV*/

        }
}

/* A função a seguir mostra a ligação com o Gerenciamento de E/S ISaGRAF
manager, utilizando o nome da conversão. Esta função é completamente gerada
pelo Gerenciamento de Biblioteca ISaGRAF. */

UFP cnvdef_sample (char *name)
{
        sys_strcpy (name, "SAMPLE");        /* dá o nome da conversão */
        return (CNV_sample);        /* endereço da função implementada */
}
```

É recomendado, para completar o quadro gerado automaticamente, descrever duas funções isoladas (não exportadas) no mesmo arquivo, para as conversões de entrada e de saída. Estas funções serão chamadas pela função principal, na estrutura de seleção principal **IF**, como mostra o exemplo anterior.

O arquivo "**TASY0DEF.H**" inclui as definições comuns a todos os arquivos fonte do núcleo ISaGRAF. Deve ser obrigatoriamente incluído para assegurar a independência relativamente ao sistema de exploração do destino. Também contém a definição do tipo **UFP**, que representa um ponteiro "far" para uma função "void", e é utilizada para a função de declaração.

Ligações entre os projetos e a implementação "C"

A ligação lógica entre a implementação de uma função de conversão e sua utilização nos projetos ISaGRAF é feita com o nome da conversão. Uma função de "declaração" é acrescentada ao código fonte " C" da função de conversão. Esta função é chamada uma só vez quando do início do aplicativo, para realizar uma ligação dinâmica entre o gerenciamento ISaGRAF de E/S e a função a ser implementada. Este é o formato padrão de uma função de declaração:

```
UFP cnvdef_xxx (char *name)
{
        strcpy (name, "XXX");       /* dá o nome da conversão */
        return (CNV_xxx);           /* endereço da função implementada */
}
/* (xxx é o nome da conversão) */
```

O nome da função utilizada na instrução **strcpy** deve ser escrito em letras **maiúsculas**. Deve ser escrito em letras minúsculas no nome das funções de implementação e de declaração.

A utilização dos prefixos "**CNV_**" e "**cnvdef_**" para as funções de implementação e de declaração permite a criação das conversões com o mesmo nome da palavra chave da linguagem "C" ou o nome de uma função já definida nas bibliotecas do "C" e do ISaGRAF.

Outras instruções podem ser adicionadas na função de declaração para executar as operações de inicialização específicas relativas a esta conversão. O sistema ISaGRAF assegura ao usuário que esta função é chamada **uma só vez** na inicialização do aplicativo.

A função de declaração é chamada por qualquer função de conversão integrada, até mesmo se não é utilizada no aplicativo ISaGRAF. O ISaGRAF núcleo gera um erro fatal se uma conversão utilizada no aplicativo não está integrada ao núcleo.

Antes de ligar novas funções com o núcleo, o usuário tem que escrever outro arquivo fonte em "C", com o nome "**GRCN0LIB.C**", e inseri-lo na lista de arquivos para a construção das bibliotecas. O " **GRCN0LIB.C** " só contém um "array" (matriz) de funções de declaração. Este "array" (matriz) é lido durante a fase de inicializações do aplicativo, para estabelecer uma ligação dinâmica com as funções de conversão em " C ". A seguir, um exemplo do conteúdo deste arquivo:

```
/* File "GRCN0LIB.c" – Exemplo com as conversões da biblioteca padrão */

#include <tasy0def.h>               /* necessário para as declarações de tipo */
```

```
extern UFP cnvdef_scale (char *name);      /* declaração da conversão SCALE */
extern UFP cnvdef_bcd (char *name);        /* declaração da conversão BCD */

UFP_LIST CNVDEF[ ] = {          /* "array" de funções de declaração para */
        /* funções de conversão integradas */
        cnvdef_scale,
        cnvdef_bcd,

NULL };

/* fim de arquivo */
```

O array **CNVDEF** deve ser terminado por um ponteiro NULL. Alguns problemas sérios podem ocorrer quando esta condição não é encontrada. As funções de conversão não incluídas no array **CNVDEF** não são reconhecidas pelo núcleo mesmo se os seus códigos são considerados na edição das ligações.

Ao escrever este arquivo, um novo núcleo pode ser construído, incluindo todas as conversões existentes. Um núcleo também pode ser construído dedicado a um projeto, inserindo no array **CNVDEF** só as conversões utilizadas no projeto. O arquivo "**GRCN0LIB.C**" é gerado automaticamente pelo Gerador de Código ISaGRAF, quando o código de um aplicativo é construído. O arquivo gerado é colocado no diretório do projeto ISaGRAF e agrupa só as conversões utilizadas no projeto.

Um outro arquivo, com o nome "**GRCN0LIB.68K**" é automaticamente gerado pelo Gerador de Código ISaGRAF, quando o código do aplicativo é construído. Este arquivo tem exatamente o mesmo significado do arquivo "**GRCN0LIB.C**" mas utiliza as convenções da escrita em assembler para declarar o array **CNVDEF**. Ele é colocado no diretório contendo os dados do projeto ISaGRAF e só agrupa as conversões referenciadas no dicionário do aplicativo. A seguir, um exemplo deste arquivo:

```
#include <tasy0def.h>

extern UFP cnvdef_scale ();
extern UFP cnvdef_dcb ();

_asm ("
CNVDEF:
        dc.l  cnvdef_scale
        dc.l  cnvdef_dcb
        dc.l  0
");

/* fim do arquivo */
```

Para maiores informações sobre este arquivo, veja o arquivo "**Leiame**" no disquete contendo o software destino ISaGRAF.

Limites

A biblioteca ISaGRAF pode conter até **128** funções de conversão. Qualquer tipo de operação pode ser processada em uma função de conversão. Deve-se notar que as funções são chamadas pelo ciclo ISaGRAF de um modo síncrono, de forma que a execução da função tenha efeito direto no tempo de ciclo do aplicativo.

C.7.3 Funções "C"

As funções "C" são utilizadas para aumentar as possibilidades das linguagens **ST** e **FBD**. Podem ser utilizadas para realizar qualquer cálculo específico, chamadas de sistema, comunicações ou instalar um conjunto de serviços para o diálogo entre um aplicativo ISaGRAF e outras tarefas. As funções são escritas em linguagem "C", compiladas e ligadas com o ISaGRAF núcleo. O núcleo ampliado deve ser instalado no ISaGRAF destino PLC antes de utilizar as novas funções em projetos ISaGRAF.

As novas funções não podem ser integradas no Simulador ISaGRAF. Os aplicativos ISaGRAF têm que ser simulados **antes** da utilização nos seus programas de novas funções.

Atenção: As funções são operações **síncronas**, ativadas no tempo de execução pelo ISaGRAF núcleo, durante o ciclo de aplicativo. O tempo gasto na execução de uma função está incluído no **tempo de ciclo** do aplicativo ISaGRAF. O usuário deve assegurar que nenhuma "operação de espera" está programada em uma função para preservar o mecanismo de execução do ISaGRAF núcleo.

Inserindo uma função na biblioteca ISaGRAF

O Gerenciamento de Biblioteca ISaGRAF deve ser utilizado para inserir uma nova função "C" na Biblioteca ISaGRAF. Com o ambiente de trabalho, utilize o comando "**Novo**" do menu "**Arquivos**", quando a biblioteca das funções "C" é selecionada. Quando uma nova função é criada, sua **Observação Técnica** deve ser escrita. O quadro do código fonte "C" para a nova função é automaticamente gerado pelo Gerenciamento de Biblioteca ISaGRAF.

O comando "**Parâmetros**" do menu "**Editar**" é utilizado para definir os parâmetros de chamada e de retorno da nova função.

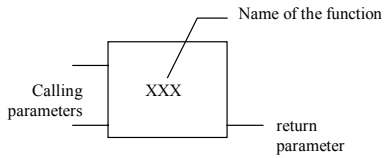
Utilizando uma função "C" em um projeto ISaGRAF

Qualquer função "C" integrada pode ser utilizada como uma função padrão nos programas de um projeto ISaGRAF. As funções em "C" podem ser utilizadas com as linguagens **ST** e **FBD**, e depois de certos enunciados da linguagem SFC.

A chamada de uma função "C" em linguagem **ST** segue as convenções de chamada de função da linguagem. Os parâmetros de chamada da função são escritos após o nome da função, entre parênteses e separados por vírgulas. A expressão representa o valor de retorno para a função. Uma chamada de função "C" pode ser inserida em qualquer enunciado de atribuição ou expressão complexa. A seguir, um exemplo de uma chamada de função "C" em um enunciado de atribuição:

```
result := ProcName (par1, par2, ... parN);
```

Um programa **FBD** pode chamar qualquer função em "C". Uma função é representada como uma caixa de função. Seus parâmetros de chamada são conectados à esquerda da caixa de função. O parâmetro de retorno é conectado à direita da caixa de função. A seguir, a representação padrão de uma função em um diagrama FBD:



Uma função "C" pode ser chamada à partir de qualquer bloco de ação **SFC** (com o atributo **P** ou **N**), ou de qualquer condição booleana atrelada a uma transição.

Definindo a interface de uma função "C"

O comando "**Parâmetros**" do menu "**Editar**" é utilizado para definir os parâmetros de chamada e de retorno de uma nova função. Uma função pode ter até **31** parâmetros de chamada e sempre **um** parâmetro de retorno.

A lista na parte superior da janela mostra os parâmetros da função "C", de acordo com a ordem definida pelo protótipo de chamada da função: primeiro os parâmetros de chamada e por último o parâmetro de saída. A parte inferior da janela mostra a descrição detalhada do parâmetro selecionado na lista:
- o nome do parâmetro
- a direção (chamada/retorno) do parâmetro
- o tipo do parâmetro

Todos os tipos de dados do ISaGRAF podem ser utilizados por um parâmetro: Booleano, Analógico inteiro, Analógico real, Temporização ou Mensagem. Os analógicos inteiros e reais devem ser diferenciados.
A seguir, a correspondência entre os tipos ISaGRAF e os tipos da linguagem "C":

BOOLEANO	unsigned long	Palavra de 32 bits s/ sinal: 1=true / 0=false
ANALÓGICO	Long	Palavra inteira c/ sinal de 32 bits
REAL	Float	Valor ponto flutuante precisão simples (32bits)
TEMPORIZAÇÃO	unsigned long	Palavra inteira s/ sinal 32 bits (unidade = 1 ms)
MENSAGEM	char *	Cadeia de caracteres (“string”).

Quando um valor de mensagem é passado para uma função "C", não pode conter caracteres nulos (código ASCII 00). A “string” passada para o código "C" é terminada por um caractere nulo. Não esqueça, o parâmetro de retorno deve ser o último na lista. As regras a seguir devem ser respeitadas pela nomenclatura dos parâmetros:
- o comprimento do nome não pode exceder **16** caracteres
- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes devem ser **letras**, **algarismos** ou o caractere sublinhado ‘_’
- o nome não faz distinção entre maiúscula/minúscula

O mesmo nome não pode ser utilizado para mais de um parâmetro da mesma função. Um parâmetro de chamada não pode ter o mesmo nome de um parâmetro de retorno. O mesmo nome **pode** ser utilizado

para parâmetros de funções diferentes. O nome padrão (proposto) para o parâmetro de retorno é "**Q**". Este nome pode ser modificado livremente. O nome de um parâmetro é utilizado para identificar o parâmetro no código fonte "C" .

O comando "**Inserir**" é utilizado para inserir um novo parâmetro antes do parâmetro selecionado na lista. O comando "**Apagar**" é utilizado para apagar o parâmetro selecionado. O comando "**Organizar**" automaticamente classifica os parâmetros e posiciona o parâmetro de retorno no fim da lista. Pressione o botão "**OK**" para gravar a definição dos parâmetros e fechar a caixa de diálogo. Pressione a tecla "**Cancelar**" para fechar a caixa de diálogo sem modificar a definição dos parâmetros.

Interface "C" de uma função

A interface de uma função depende da definição de seus parâmetros. Os parâmetros de chamada e de retorno são passados por uma estrutura. Esta estrutura está definida no arquivo "**GRUS0nnn.H**" em que "**nnn**" é o número lógico da função na biblioteca ISaGRAF. Este é um exemplo da interface "C", para a função "SIN" (cálculo de seno):

```
/* Arquivo: GRUS0255.h - função "sample" */

typedef  long                   T_BOO;
typedef  long                   T_ANA;
typedef  float                  T_REAL;
typedef  long                   T_TMR;
typedef  char                   *T_MSG;

typedef struct {
        /* CALL   */            T_REAL   _param1;
        /* RETURN */            T_REAL   _param2;
} str_arg;

#define PARAM1                  (arg->_param1)
#define PARAM2                  (arg->_param2)

/* fim do arquivo */
```

A relação entre os tipos ISaGRAF e os tipos da linguagem "C" é mostrada a seguir. Os tipos ISaGRAF são definidos como tipos "C" no arquivo de definição da função.

Booleano	T_BOO	long (32 bits)
Analógico	T_ANA	Long
Real	T_REAL	float (32 bits – precisão simples)
Temporização	T_TMR	Long
Mensagem	T_MSG	char * (32 bits – ponteiro de caracteres)

Cada campo da estrutura "**str_arg**" corresponde a um parâmetro de função. O parâmetro de retorno é o último na estrutura. Os parâmetros de chamada aparecem na estrutura com a mesma ordem da

estabelecida para a definição de função. Um identificador de maiúscula é definido para cada parâmetro permitindo o acesso diretamente ao valor do parâmetro na estrutura passada para função "C". Os nomes dos identificadores são os inseridos durante a definição da função com o Gerenciamento de Biblioteca ISaGRAF.

O arquivo de definição "C" é atualizado toda vez que a interface da função é modificada utilizando o Gerenciamento de Biblioteca ISaGRAF. Isto assegura uma correspondência completa entre a implementação da função e sua utilização nos programas dos aplicativos ISaGRAF.

Código fonte

A seguir, o quadro padrão da implementação de uma função "C":

```
/* Exemplo de função de usuário - Número "255" – Nome : "SAMPLE" */

#include "tasy0def.h"              /* Definições comuns do ISaGRAF núcleo */
#include "grus0255.h"              /* Definição da interface da função 255 */

void USP_sample (str_arg *arg)
{
        /* corpo da função */
}

/ * A função a seguir é utilizada para a inicialização da função e a declaração de
sua implementação. Realiza a ligação com o ISaGRAF núcleo, utilizando o nome
da função. Esta função é automaticamente gerada pelo Gerenciamento de
Biblioteca ISaGRAF. */

UFP uspdef_sample (char *name)
{
        strcpy (name, "SAMPLE"); /* dá o nome da função */
        return (USP_sample);     /* endereço da função implementada */
}

/* fim de arquivo */
```

O arquivo "**TASY0DEF.H**" reúne as definições comuns a todos os arquivos “fonte” do ISaGRAF núcleo. Também contém a definição do tipo **UFP**, que representa um ponteiro “far” para uma função “void” e é utilizada para a função de declaração.

Ligações entre projetos e a implementação em "C"

A ligação lógica entre a implementação em “C” de uma função e sua utilização nos projetos ISaGRAF é feita com o nome da função. Uma função de "declaração" é acrescentada ao código fonte "C" da função. Esta função é chamada uma só vez quando do início do aplicativo, para realizar uma ligação

dinâmica entre o ISaGRAF núcleo e a função a implementada. Este é o formato padrão de uma função de declaração:

```
UFP uspdef_xxx (char *name)
{
        strcpy (name, "XXX");       /* dá o nome da conversão */
        return (USP_xxx);           /* endereço da função implementada */
}
/* (xxx é o nome da conversão) */
```

O nome da função utilizada na instrução **strcpy** deve ser escrito em letras **maiúsculas**. Deve ser escrito em letras minúsculas no nome das funções de implementação e de declaração. A utilização dos prefixos "**USP_**" and "**uspdef_**" para as funções de implementação e de declaração permite a criação das funções com o mesmo nome da palavra chave da linguagem "C" ou o nome de uma função já definida nas bibliotecas do "C" e do ISaGRAF.

Outras instruções podem ser adicionadas na função de declaração para executar as operações de inicialização específicas relativas a esta função. O sistema ISaGRAF assegura ao usuário que esta função é chamada **uma só vez** na inicialização do aplicativo. A função de declaração é chamada por qualquer função integrada ao núcleo, até mesmo se não são utilizadas no aplicativo ISaGRAF. O ISaGRAF núcleo gera um erro fatal se uma conversão utilizada no aplicativo não está integrada ao núcleo.

Antes da edição das ligações do núcleo com as novas funções, o programador deve escrever outro arquivo fonte "C", com o nome"**GRUS0LIB.C**", e inseri-lo na lista de arquivos para a construção das bibliotecas. O "**GRUS0LIB.C**" só contém um array de funções de declaração. Este array é lido durante a fase de inicialização do aplicativo, para estabelecer uma ligação dinâmica com as funções de conversão em " C ". A seguir, um exemplo do conteúdo deste arquivo:

```
/* File "GRUS0LIB.c" – Exemplo utilizando funções trigonométricas */

#include <tasy0def.h>               /* necessário para as definições de tipo */

extern UFP uspdef_fc1 (char *name);     /* declaração de funções */
extern UFP uspdef_fc2 (char *name);
extern UFP uspdef_fc3 (char *name);
extern UFP uspdef_fc4 (char *name);

UFP_LIST USPDEF[ ] = {          /* array de declaração de funções */
        /* para funções integradas */
        uspdef_fc1,
        uspdef_fc2,
        uspdef_fc3,
        uspdef_fc4,
```

NULL };

/* fim de arquivo */

O array **USPDEF** deve ser terminado por um ponteiro NULL. Alguns problemas sérios podem ocorrer quando esta condição não é encontrada. As funções não incluídas no array **USPDEF** não são reconhecidas pelo núcleo mesmo se os seus códigos são considerados na edição das ligações. Ao escrever este arquivo, um novo núcleo pode ser construído, incluindo todas as novas funções. Também é possível criar um núcleo dedicado a um projeto inserindo no array **USPDEF** só as funções utilizadas no projeto. O arquivo "**GRUS0LIB.C**" é gerado automaticamente pelo Gerador de Código ISaGRAF, quando o código de um aplicativo é construído. O arquivo gerado é colocado no diretório do projeto ISaGRAF e agrupa só as funções utilizadas no projeto.

Limites

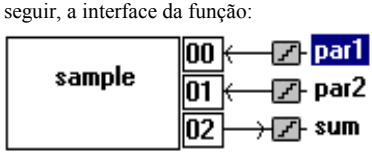
A biblioteca ISaGRAF pode conter até **255** funções "C". Qualquer tipo de operação pode ser processada em uma função. Deve-se notar que as funções são chamadas pelo ciclo ISaGRAF de um modo **síncrono**, de forma que a execução da função tenha efeito direto no tempo de ciclo do aplicativo.

Exemplo completo

A seguir, a programação completa de uma função "**sample**", que executa uma simples adição analógica inteira. Eis a observação técnica da função:

nome:	SAMPLE
descrição:	Calcula uma adição analógica inteira
data de criação:	1st July 1992
autor:	CJ International
chamada:	par1, par2: operandos inteiros
retorno:	soma: resultado da adição
protótipo:	sum := sample (par1, par2);

A seguir, a interface da função:



a seguir, o arquivo de definições "C" para a função:

/* File: GRUS0255.h – função “C” - Nome: sample */

/* definição dos tipos padrão ISaGRAF */

```
typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definição da estrutura dos parâmetros de chamada e de retorno */

typedef struct {
        T_ANA _par1;            /* parâmetro de chamada #1 */
        T_ANA _par2;            /* parâmetro de chamada #2 */
        T_ANA _sum;             /* parâmetro de chamada */
} str_arg;

/* identificadores utilizados para acessar os parâmetros de chamada e de retorno */

#define PAR1                    (arg->_par1)
#define PAR2                    (arg->_par2)
#define SUM                     (arg->_sum)

/* fim de arquivo */
```

A seguir, o arquivo código fonte "C". Somente as linhas impressas com caracteres em negrito são inseridas manualmente pelo programador.

```
/* File: GRUS0255.c – função C - Nome: SAMPLE */

#include "tasy0def.h"           /* necessário para as definições de tipo */
#include "grus0255.h"           /* arquivo de definições C */

/* serviço principal: calcula a adição */

void USP_sample (str_arg *arg)
{
        SUM = PAR1 + PAR2;
}

/* declaração para a ligação dinâmica com o núcleo ISaGRAF */

UFP uspdef_sample (char *name)
{
```

```
        strcpy (name, "SAMPLE");
        return (USP_sample);
}
/* fim do arquivo */
```

C.7.4 Blocos de função em "C"

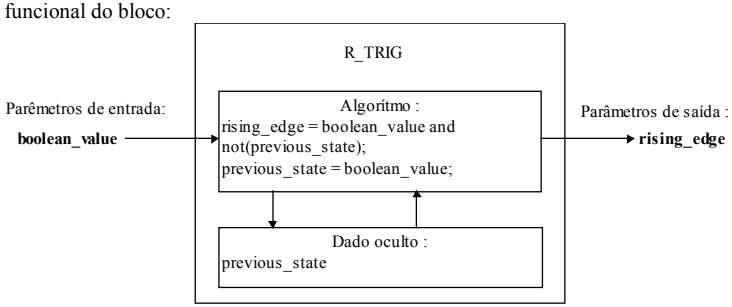
Os blocos de função associam operações e dados estáticos. Complementam o conjunto das funções "C", permitindo o processamento de objetos estáticos. São geralmente utilizados para aumentar as possibilidades das linguagens **ST** e **FBD**. Ao contrário das funções, que processam valores, os blocos de função podem processar dados estáticos. Isto significa que um algorítmo de bloco de função pode gerenciar as variações no tempo destes dados.

Os blocos de função são escritos em linguagem "C", compilados e ligados com o ISaGRAF núcleo. O núcleo ampliado deve ser instalado no ISaGRAF destino PLC antes de utilizar os novos blocos de função em projetos ISaGRAF. Os novos blocos de função não podem ser integrados no Simulador ISaGRAF. Os aplicativos ISaGRAF têm que ser simulados **antes** da utilização nos programas dos novos blocos.

Atenção: Os blocos de função realizam operações elementares **síncronas**, ativadas pelo ISaGRAF núcleo, durante o ciclo de aplicativo. O tempo gasto na execução de um bloco de função está incluído no **tempo de ciclo** do aplicativo ISaGRAF. O usuário deve assegurar que nenhuma "operação de espera" está programada em um bloco de função, para preservar o mecanismo de execução do ISaGRAF núcleo.

Declarando instâncias

Um bloco de função é um objeto que combina operações e dados estáticos. A seguir, um exemplo de bloco de função "**R_TRIG**" que detecta a borda de subida de uma expressão booleana. Eis a descrição funcional do bloco:



O valor estático oculto "**previous_state**" é necessário para o cálculo da borda. Esta variável deve ser diferenciada toda vez que o bloco de função "**R_TRIG**" é utilizado no mesmo aplicativo. As instâncias (cópias) dos blocos de função utilizadas na linguagem ST devem ser declaradas no dicionário do aplicativo. Porque o bloco de função inclui os dados, cada cópia (instância) de um bloco de função deve ser identificado por um único nome. O tipo de bloco de função é identificado com o gerenciamento da biblioteca. O nome das instâncias é dado com a ferramenta de edição do Dicionário.

As instâncias dos blocos de função utilizados nos diagramas FBD não precisam ser declarados já que o editor ISaGRAF FBD automaticamente declara as novas instâncias toda vez que um bloco de função é inserido no diagrama. As instâncias de blocos de função declaradas automaticamente pelo editor FBD são sempre **LOCAIS** para o programa editado.

Inserindo um bloco de função na biblioteca ISaGRAF

O Gerenciamento de Biblioteca ISaGRAF deve ser utilizado para inserir um novo bloco de função "C" na Biblioteca ISaGRAF. Com o ambiente de trabalho, utilize o comando "**Novo**" do menu "**Arquivos**", quando a biblioteca dos blocos de função "C" é selecionada. Quando um novo bloco de função é criado, sua **Observação Técnica** deve ser escrita. O quadro do código fonte "C" para o novo bloco de função é automaticamente gerado pelo Gerenciamento de Biblioteca ISaGRAF. O comando "**Parâmetros**" do menu "**Editar**" é utilizado para definir os parâmetros de chamada e de retorno do novo bloco de função.

Utilizando um bloco de função "C" em um projeto ISaGRAF

Qualquer bloco de função integrado ao ISaGRAF núcleo pode ser utilizado nos projetos ISaGRAF. Os blocos de função podem ser utilizados nos programas escritos nas linguagens **ST** e **FBD**.

A chamada de um bloco de função à partir da linguagem **ST** segue as convenções de chamada de bloco de função da linguagem. Os parâmetros de chamada do bloco são escritos após o nome da função, entre parênteses e separados por vírgulas. Os parâmetros de retorno são chamados um por um. Cada parâmetro de retorno é identificado por um nome, combinando o nome da instância do bloco e o nome lógico do parâmetro. Os componentes do nome são separados por um ponto. Por exemplo, o nome:

FBINSTNAME.parname

É utilizado para representar o parâmetro de retorno cujo nome é "**parname**", de uma instância de bloco de função de nome "**FBINSTNAME**".

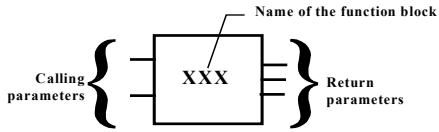
As instâncias dos blocos de função utilizadas na linguagem ST devem ser declaradas no dicionário do aplicativo. Cada cópia (instância) de um bloco de função deve ser identificado por um único nome. A seguir, um exemplo de declaração de instâncias no dicionário ISaGRAF:

instance:	TRIG1	type:	R_TRIG
	TRIG2		R_TRIG

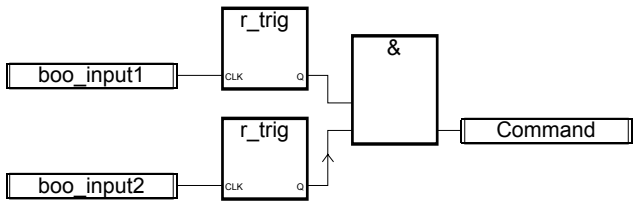
E um exemplo utilizando estas instâncias declaradas em um programa ST:

```
TRIG1 (boo_input1);
TRIG2 (boo_input2);
Command := (TRIG1.Q & TRIG2.Q);
```

Um programa **FBD** pode chamar qualquer bloco de função "C" da biblioteca. Um bloco de função é representado como uma caixa. Seus parâmetros de chamada são conectados à esquerda da caixa. Os parâmetros de retorno são conectados à direita da caixa. A seguir, a representação padrão de um bloco de função em um diagrama FBD:



As instâncias de blocos de função utilizadas em diagramas FBD não precisam ser declaradas já que o editor ISaGRAF FBD automaticamente declara as instâncias dos blocos utilizados. As instâncias de blocos de função automaticamente declaradas pelo editor FBD são sempre **LOCAIS** para o programa editado. A seguir, o exemplo anterior programado na linguagem FBD:



Definindo a interface de um bloco de função "C"

O comando "**Parâmetros**" do menu "**Editar**" é utilizado para definir os parâmetros de chamada e de retorno de um bloco de função. Um bloco de função pode ter até **32** parâmetros, livremente repartidos em parâmetros de chamada ou de retorno. Ao contrário de função "C", um bloco de função pode ter diversos parâmetros de retorno.

A lista na parte superior da janela mostra os parâmetros do bloco de função, de acordo com a ordem definida pelo protótipo de chamada: primeiro os parâmetros de chamada e por último os parâmetros de retorno. A parte inferior da janela mostra a descrição detalhada do parâmetro selecionado na lista:
- o nome do parâmetro
- a direção (chamada/retorno) do parâmetro
- o tipo do parâmetro

Todos os tipos de dados do ISaGRAF podem ser utilizados por um parâmetro: Booleano, Analógico inteiro, Analógico real, Temporização ou Mensagem. Os analógicos inteiros e reais devem ser diferenciados.
A seguir, a correspondência entre os tipos ISaGRAF e os tipos da linguagem "C":

BOOLEANO	unsigned long	Palavra de 32 bits s/ sinal: 1=true / 0=false
ANALÓGICO	long	Palavra inteira c/ sinal de 32 bits
REAL	float	Valor ponto flutuante precisão simples (32bits)
TEMPORIZAÇÃO	unsigned long	Palavra inteira s/ sinal 32 bits (unidade = 1 ms)
MENSAGEM	char *	Cadeia de caracteres (“string”).

Quando um valor de mensagem é passado para uma função "C", não pode conter caracteres nulos (código ASCII 00). A “string” passada para o código "C" é sempre terminada por um caractere nulo. Não esqueça, os parâmetros de retorno devem ser os últimos na lista. As regras a seguir devem ser respeitadas pela nomenclatura dos parâmetros:
- o comprimento do nome não pode exceder **16** caracteres

- o primeiro caractere deve ser uma **letra**
- os caracteres seguintes devem ser **letras**, **algarismos** ou o caractere sublinhado '_'
- o nome não faz distinção entre maiúscula/minúscula

O mesmo nome não pode ser utilizado por diversos parâmetros do mesmo bloco de função. Um parâmetro de chamada não pode ter o mesmo nome de um parâmetro de retorno. O mesmo nome **pode** ser utilizado para parâmetros de blocos de função diferentes. O nome de um parâmetro é utilizado para identificar o dado correspondente no código fonte "C" do bloco de função.

O comando "**Inserir**" é utilizado para inserir um novo parâmetro antes do parâmetro selecionado na lista. O comando "**Apagar**" é utilizado para apagar o parâmetro selecionado. O comando "**Organizar**" automaticamente classifica os parâmetros e posiciona os parâmetros de retorno no fim da lista. Pressione o botão "**OK**" para gravar a definição dos parâmetros e fechar a caixa de diálogo. Pressione a tecla "**Cancelar**" para fechar a caixa de diálogo sem modificar a definição dos parâmetros.

Interface "C" de um bloco de função

A interface de um bloco de função depende da definição de seus parâmetros. Os parâmetros de chamada são passados por uma estrutura. Esta estrutura está definida no arquivo "**GRFB0nnn.H**" em que "**nnn**" é o número lógico do bloco de função na biblioteca ISaGRAF. Os parâmetros de retorno são identificados por números lógicos também definidos no arquivo "**GRFB0nnn.H**". Este é um exemplo da interface "C", para o bloco de função "**LIM_ALRM**" (alarmes dos limites):

```
/* interface de bloco de função - nome: sample */

/* tipos padrão ISaGRAF */

typedef                 long    T_BOO;
typedef                 long    T_ANA;
typedef                 float   T_REAL;
typedef                 long    T_TMR;
typedef                 char    *T_MSG;

/* estrutura de parâmetros de chamada */

typedef struct {
        /* CALL  */             T_BOO _par1;
        /* CALL  */             T_BOO _par2;
} str_arg;

/* acesso aos campos da estrutura str_arg */

#define                 PAR1    (arg->_par1)
#define                 PAR2    (arg->_par2)
```

```
/* números lógicos dos parâmetros de retorno */

#define                 FBLPNO_Q1    0
#define                 FBLPNO_Q2    1

/* fim de arquivo */
```

A relação entre os tipos ISaGRAF e os tipos da linguagem "C" é mostrada a seguir. Os tipos ISaGRAF são definidos como tipos "C" no arquivo de definição da função.

Booleano	T_BOO	long (32 bits)
Analógico	T_ANA	Long
Real	T_REAL	float (32 bits – precisão simples)
Temporização	T_TMR	Long
Mensagem	T_MSG	char * (32 bits – ponteiro de caractere)

Cada campo da estrutura "**str_arg**" corresponde a um parâmetro de chamada do bloco de função. Os parâmetros de chamada aparecem na estrutura com a mesma ordem da estabelecida para a definição do bloco de função. Um identificador de maiúscula é definido para cada parâmetro permitindo o acesso diretamente ao valor do parâmetro na estrutura passada para função "C". Os nomes dos identificadores são os inseridos durante a definição da função com o Gerenciamento de Biblioteca ISaGRAF.

A ordem utilizada para a numeração dos parâmetros de retorno é a estabelecida para a definição do bloco de função. O número lógico do primeiro parâmetro de retorno é sempre **0**.

Os identificadores definidos para representar estes números devem ser utilizados na fonte "C" (e não diretamente os números). Isto assegura que o arquivo fonte "C" é facilmente recompilado após uma modificação da definição de interface do bloco funcional.

O arquivo de definição "C" é atualizado automaticamente toda vez que a interface do bloco de função é modificada utilizando o Gerenciamento de Biblioteca ISaGRAF. Isto assegura uma coerência entre a implementação em "C" dos blocos de função e sua utilização nos programas dos aplicativos ISaGRAF.

Código fonte

A implementação em linguagem "C" de um bloco de função é dividida em três serviços principais:

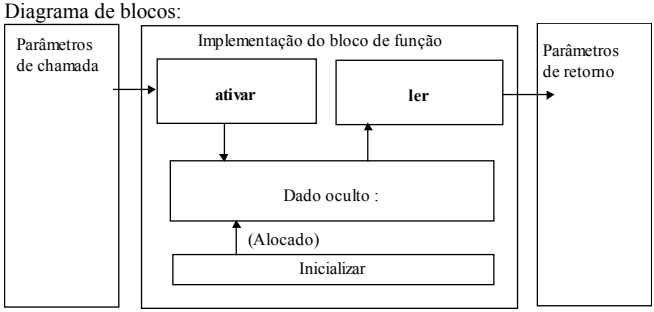
- serviço de inicialização
- serviço de ativação – processamento dos parâmetros de chamada
- serviço de leitura – acesso aos parâmetros de retorno

O mesmo código é utilizado para cada instância de um mesmo bloco de função e não é duplicado na memória. Uma estrutura de dados estáticos está associada a cada instância. Tais dados não podem ser acessados diretamente pelos programas ISaGRAF, e contêm as "variáveis ocultas" associadas à instância.

O "serviço de ativação " é chamado uma vez para cada instância de cada bloco utilizado, em cada ciclo de execução. Processa os parâmetros de chamada do bloco de função e atualiza os dados associados. Representa o "algoritmo principal" do bloco de função.

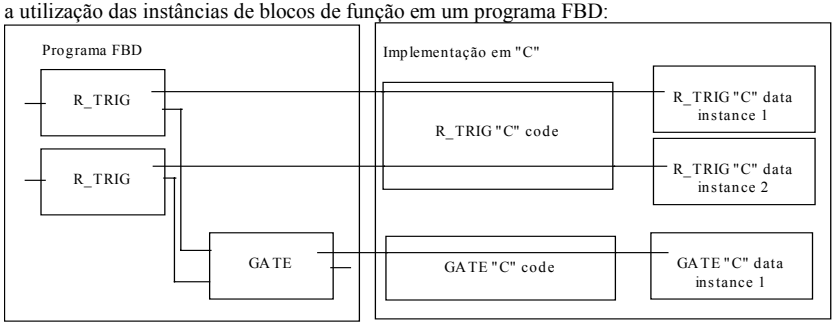
O "serviço de leitura" é chamado pelo ISaGRAF núcleo para ler o valor corrente de um parâmetro de retorno para uma instância do bloco de função. Nenhum cálculo especial tem que ser executado neste serviço. Só opera transferência entre os dados ocultos da instância e o aplicativo ISaGRAF.

Diagrama de blocos:



- **Bloco de função dados estáticos**

Um bloco de função associa operações e dados estáticos. Uma estrutura de dados é associada a cada instância de um mesmo bloco de função. Cada vez que um bloco de função é utilizado em programação ST ou FBD, corresponde a uma instância e uma estrutura de dados associada. O exemplo a seguir mostra a correspondência entre estruturas de dados manipulados pelo código "C" do bloco de função e a utilização das instâncias de blocos de função em um programa FBD:



A memória necessária para armazenar cada estrutura de dados é alocada pelo sistema ISaGRAF, na inicialização do aplicativo. Um ponteiro para a estrutura de instância associada é passado em argumento a cada chamada de serviços de ativação e de leitura do bloco de função correspondente.

O Gerenciamento de Biblioteca ISaGRAF gera automaticamente o quadro de definição do tipo da estrutura no código fonte "C" do bloco de função. Este tipo sempre é chamado "**str_data**". O programador não deve mudar este nome, assegurando assim a coerência com as definições de serviço. Os dados ocultos geralmente agrupam as variáveis internas de cálculo e uma imagem dos parâmetros de retorno. O serviço de "leitura" do bloco de função só é utilizado para acessar o parâmetro de retorno e não deve ser utilizado para executar outras operações.

- **O serviço de inicialização**

O serviço de "inicialização" de um bloco de função é chamado pelo ISaGRAF núcleo quando o aplicativo é iniciado. Permite ao bloco de função indicar ao sistema ISaGRAF o tamanho da memória que deve ser alocada para uma instância do bloco. A seguir, a programação padrão deste serviço de inicialização:

```
uint16 FBINIT_xxx (uint16 hinstance)
/* "xxx" é o nome do bloco de função */
{
  return (sizeof (str_data));
}
```

O argumento "**hinstance**" é o número lógico da instância. É reservado para as operações internas ISaGRAF e não deve ser utilizado na programação do serviço. O serviço de inicialização retorna o número de bytes de memória alocados para os dados de uma instância. A quantidade de memória exigida (valor de retorno) não pode exceder **64** kbytes. Nenhuma outra operação deve ser executada neste serviço. O código fonte "C" deste serviço é gerado automaticamente pelo Gerenciamento de Biblioteca ISaGRAF quando o bloco de função é criado.

- **O serviço de ativação**

O serviço de "ativação" é chamado em cada ciclo de execução, para cada instância de bloco de função utilizado no aplicativo. Este serviço processa os parâmetros de chamada e executa o algorítmo principal do bloco de função para atualizar os dados internos ocultos e o valor de parâmetros de retorno. A seguir, o quadro padrão do serviço de ativação:

```
void FBACT_xxx (
uint16 hinstance,          /* "xxx" é o nome do bloco de função */
                           /* número lógico da instância */
str_data *data,            /* data: aponta para os dados da instância */
str_arg *arg               /* aponta para a estrutura dos parâmetros de
chamada */
)
{
}
```

O argumento "**hinstance**" é o número lógico da instância. É reservado para as operações internas ISaGRAF e não deve ser utilizado na programação do serviço. O argumento "**data**" é um ponteiro para a estrutura de dados contendo as "variáveis ocultas" associadas à instância. O argumento "**arg**" é um ponteiro para a estrutura que contém os valores correntes dos parâmetros de chamada do bloco de função. O programador deve utilizar os identificadores definidos no arquivo de definições "C" associado para ter acesso aos campos da estrutura "**arg**".

O algorítmo de "ativação" trata os parâmetros de chamada do bloco (armazenado na estrutura "**arg**"), e atualiza os campos da estrutura "**data**". O exemplo a seguir mostra o serviço de "ativação" do bloco de função **TRIG** (detecção de borda de subida):

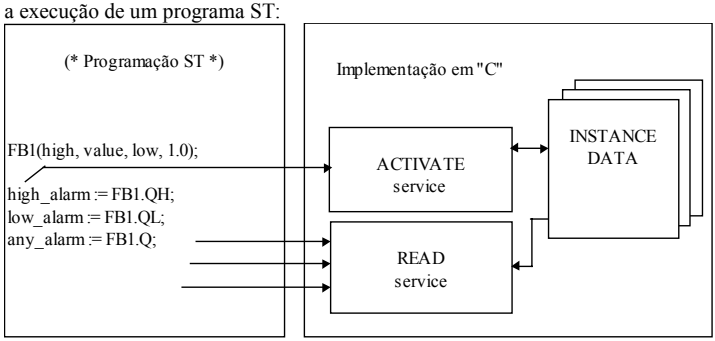
```
/* definições escritas no arquivo de definições "C" associado */
```

```
typedef struct {                    /* parâmetros de chamada */
        T_BOO  _clk;                /* sinal de entrada */
} str_arg;

#define CLK      (arg->_clk)

/* definição da estrutura de dados associada a uma instância */

typedef struct {
        T_BOO prev_state;           /* estado anterior do sinal de entrada */
        T_BOO edge_detect;          /* detecção da borda de subida */
} str_data;

/* serviço de ativação */

void FBACT_trig (uint16 hinstance, str_data *data, str_arg *arg)
{
        data->edge_detect = (T_BOO)(CLK && !data->prev_state);
        data->prev_state = CLK;  /* parâmetro de chamada */
}
```

O quadro do código fonte "C" deste serviço é automaticamente gerado pelo Gerenciamento de Biblioteca ISaGRAF quando o bloco de função é criado.

- **Acesso aos parâmetros de retorno**

O serviço de "leitura" é chamado cada vez que um parâmetro de retorno de uma instância de bloco de função é referenciado num programa ST ou FBD. É utilizado para obter um valor de **um único** parâmetro de retorno. O seguinte exemplo mostra a chamada do serviço de "leitura" executado durante a execução de um programa ST:



Porque o serviço de leitura pode ser chamado diversas vezes no mesmo ciclo de execução para o acesso ao mesmo parâmetro de retorno ou à mesma instância de bloco, não deve efetuar nenhum cálculo que

modifique o valor dos dados associados à instância. Só efetua transferência entre dados ocultos e o aplicativo ISaGRAF. a seguir, o quadro padrão do serviço de leitura:

```
/* conjunto de operações utilizadas para copiar o valor do parâmetro de retorno */

#define BOO_VALUE          ((T_BOO *)value)
#define ANA_VALUE          ((T_ANA *)value)
#define REAL_VALUE         ((T_REAL *)value)
#define TMR_VALUE          ((T_TMR *)value)
#define MSG_VALUE          ((T_MSG *)value)

/* serviço de leitura: um só parâmetro lido numa chamada */

void FBREAD_xxx (          /* "xxx" é o nome do bloco de função */
uint16 hinstance,          /* número lógico da instância */
str_data *data,            /* ponteiro para os dados da instância */
uint16 parno,              /* número lógico do parâmetro lido */
void *value)               /* endereço do "buffer" em que o */
                           /* valor do parâmetro deve ser copiado */
{
        switch (parno) {
            case FBLPNO_XX: /* ... */ break;
            case FBLPNO_YY: /* ... */ break;
             /* .... */
        }
}
```

O argumento "**hinstance**" é o número lógico da instância. É reservado para as operações internas ISaGRAF e não deve ser utilizado na programação do serviço. O argumento "**data**" é um ponteiro para a estrutura de dados contendo as "variáveis ocultas" associadas à instância.

O argumento "**parno**" é o número lógico do parâmetro solicitado. Utilize os identificadores definidos no arquivo de definições "C" associado para distinguir os parâmetros de retorno. Tais identificadores sempre começam com o prefixo "**FBLPNO_**". O argumento "**value**" é o endereço do "buffer" interno do ISaGRAF núcleo em que o valor atual do parâmetro solicitado deve ser copiado. O tipo de dados apontado por este argumento depende do tipo declarado para o parâmetro considerado . A tabela a seguir mostra a correspondência entre o tipo ISaGRAF do parâmetro e o tipo "C" de dados apontados pelo argumento "**value**":

booleano	long	Palavra s/ sinal 32 bits (1=true / 0=false)
analógico	long	Palavra c/ sinal de 32 bits
real	float	Valor ponto flutuante precisão simples (32bits)
temporização	long	Palavra s/ sinal 32 bits (unidade = 1 ms)
mensagem	char *	array de caracteres

As seguintes macros são utilizadas para o acesso ao "buffer" interno do ISaGRAF. de acordo com o tipo de parâmetro acessado:

```
#define     BOO_VALUE       ((T_BOO *)value)
#define     ANA_VALUE       ((T_ANA *)value)
#define     REAL_VALUE      ((T_REAL *)value)
#define     TMR_VALUE       ((T_TMR *)value)
#define     MSG_VALUE       ((T_MSG *)value)
```

A seguir, as operações programadas mais comumente utilizadas para copiar o valor ou o parâmetro para o "buffer" ISaGRAF:

```
/* para um parâmetro booleano: */
  *BOO_VALUE = parameter_value;
/* para um parâmetro analógico inteiro: */
  *ANA_VALUE = parameter_value;
/* para um parâmetro real inteiro: */
  *REAL_VALUE = parameter_value;
/* para um parâmetro temporização: */
  *TMR_VALUE = parameter_value;
/* para um parâmetro mensagem: */
  strcpy (*MSG_VALUE, parameter_value);
```

O quadro do código fonte "C" deste serviço é automaticamente gerado pelo Gerenciamento de Biblioteca ISaGRAF quando o bloco de função é criado.

- **Exemplo de código fonte "C"**

A seguir, o quadro padrão de uma implementação de um bloco de função "C":

```
/* function block (xxx é o nome do bloco de função) */

#include <tasy0def.h>
#include <grfb0nnn.h>   /* nnn é o número do bloco de função na biblioteca */

/* estrutura dos dados ocultos para cada instância do bloco */
typedef struct {
        /* definição dos campos */
} str_data;

/* serviço de inicialização: retorna o tamanho dos dados ocultos necessários */
word FBINIT_xxx (uint16 hinstance)
{
        return (sizeof (str_data));
```

```
}

/* serviço de ativação: processa os parâmetros de chamada */
void FBACT_xxx (uint16 hinstance, str_data *data, str_arg *arg)
{
        /* ... */
}

/* conjunto de operações utilizadas para copiar o valor do parâmetro de retorno */
#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* serviço de leitura: um só parâmetro lido numa chamada */
void FBREAD_xxx (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
        switch(parno)
        {
          case FBLPNO_XX: *???_VALUE = ...; break;
          case FBLPNO_YY: *???_VALUE = ...; break;
        ....

}
```

/* O serviço a seguir é utilizado para a ligação dinâmica entre os serviços implementados e o aplicativo ISaGRAF. A ligação é executada na inicialização do aplicativo com a ajuda do nome do bloco de função. O código deste serviço é completamente gerado pelo Gerenciamento de Biblioteca ISaGRAF. Os tipos "?BP" são definidos no arquivo "grta0def.h" como os ponteiros para as funções. */

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");
        *initproc = (IBP)FBINIT_xxx;
        *readproc = (RBP)FBREAD_xxx;
        return ((ABP)FBACT_xxx);
}
```

```
/* fim do arquivo */
```

O arquivo "**TASY0DEF.H**" inclui as definições comuns a todos os arquivos fonte do núcleo ISaGRAF. Deve ser obrigatoriamente incluído para assegurar a independência relativamente ao sistema de exploração do destino. Também contém a definição do tipo **?FP**, utilizado pela função de declaração.

Ligações entre os projetos e a implementação "C"

A ligação lógica entre a implementação de um bloco de função "C" e sua utilização nos programas de um projeto ISaGRAF é feita com o nome do bloco de função. Um serviço de "declaração" é acrescentado ao código fonte " C" do bloco de função. Este serviço é chamado uma só vez quando do início do aplicativo e indica ao ISaGRAF núcleo o nome do bloco de função "C" que corresponde aos serviços implementados. Este é o formato padrão de tal serviço de declaração:

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");                   /* nome do bloco de função */
        *initproc = (IBP)FBINIT_xxx;                    /* serviço de inicialização */
        *readproc = (RBP)FBREAD_xxx;                    /* serviço de leitura */
        return ((ABP)FBACT_xxx);                        /* serviço de ativação */
}
/* xxx é o nome do bloco de função */
```

O nome do bloco de função utilizado na instrução **strcpy** deve ser escrita em letras **maiúsculas**. Deve ser escrito em letras minúsculas no nome dos serviços implementados.

A utilização dos prefixos "**FBACT_**", "**FBINIT_**", "**FBREAD_**" e "**FBLDEF_**" para os serviços implementados e a definição de serviço permite que o usuário dê o nome a um bloco de função com uma palavra chave reservada da linguagem "C" ou o nome de uma função já definida nas bibliotecas do "C" e do ISaGRAF. Nenhuma outra instrução deve ser adicionada ao serviço de declaração.

O serviço de declaração é chamado por qualquer bloco de função "C" integrado mesmo se não é utilizado nos programas do aplicativo ISaGRAF. O ISaGRAF núcleo gera um erro fatal se um bloco de função "C" utilizado no aplicativo não está integrado ao núcleo.

Antes da edição das ligações do núcleo com os novos blocos de função, o programador deve escrever outro arquivo fonte "C", com o nome"**GRFB0LIB.C**", e inseri-lo na lista de arquivos para a construção das bibliotecas. O arquivo "**GRFB0LIB.C**" só contém um array de serviços de declaração. Este array é lido durante a fase de inicialização do aplicativo ISaGRAF núcleo para estabelecer uma ligação dinâmica com os blocos de função implementados. A seguir, um exemplo do conteúdo deste arquivo:

```
/* Arquivo: grfb0lib.c - blocos de função implementados */

#include <tasy0def.h>

extern ABP fbldef_fb1(char *name, IBP *init, RBP *read);
```

```
extern ABP fbldef_fb2(char *name, IBP *init, RBP *read);

FBL_LIST FBLDEF[ ] = {
        fbldef_fb1,
        fbldef_fb2,

NULL };

/* fim de arquivo */
```

O array **FBLDEF** deve ser terminado por um ponteiro NULL. Alguns problemas sérios podem ocorrer quando esta condição não é encontrada. Os blocos de função não incluídos no array **FBLDEF** não são reconhecidos pelo núcleo mesmo se os seus códigos são considerados na edição das ligações.

Ao escrever este arquivo, um novo núcleo pode ser construído, incluindo todos os novos blocos de função. Também é possível criar um núcleo dedicado a um projeto inserindo no array **FBLDEF** só os blocos de função utilizados no projeto. O arquivo "**GRFB0LIB.C**" é gerado automaticamente pelo Gerador de Código ISaGRAF, quando o código de um aplicativo é construído. O arquivo gerado é colocado no diretório do projeto ISaGRAF e agrupa só o blocos de função utilizados no projeto.

Limites

A biblioteca ISaGRAF pode conter até **255** blocos de função "C". Qualquer tipo de operação pode ser processada em um bloco de função. Cada tipo de bloco de função pode ser copiado até **255** vezes no mesmo projeto.

Deve-se notar que os serviços bloco de função são chamadas pelo ciclo ISaGRAF de um modo **síncrono**, de forma que a execução do bloco de função tenha efeito direto no tempo de ciclo do aplicativo.

Exemplo completo

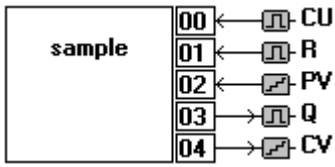
A seguir, a título de exemplo, a programação completa do bloco de função "**sample**", que é um contador incremental.

Eis a observação técnica deste bloco de função:

nome:	SAMPLE
descrição:	contador incremental
data da criação:	01 February 1994
autor:	CJ International
chamada:	CU : entrada da contagem
	R : comando de reset
	PV : valor máximo programado
retorno:	Q : detecção do máximo
	CV : resultado atual da contagem

protótipo:	SAMPLE (count, reset_command, maximum_value);
	max_detect := SAMPLE.Q;
	count_result := SAMPLE.CV;

A seguir, a definição da interface do bloco de função:



A seguir, o arquivo de definições "C" para o bloco de função:

```
/* interface do bloco de função - nome: SAMPLE */

/* definição dos tipos padrão ISaGRAF */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definição da estrutura composta dos parâmetros de chamada */

typedef struct {
  T_BOO  _cu;
  T_BOO  _r;
  T_ANA  _pv;
} str_arg;

/* identificadores utilizados para o acesso direto aos parâmetros de chamada */

#define CU                  (arg->_cu)
#define R                   (arg->_r)
#define PV                  (arg->_pv)

/* numeração dos parâmetros de retorno */

#define FBLPNO_Q            0
```

```
#define FBLPNO_CV                1

/* fim do arquivo */
```

A seguir, o arquivo de código fonte "C" do bloco de função. Somente as linhas mostradas em negrito são inseridas manualmente pelo programador.

```
/* function block - name: SAMPLE */

#include <tasy0def.h>            /* necessário para as definições de tipo */
#include <grfb0255.h>            /* definição "C" da interface */

/* definição da estrutura de dados associada a uma instância */

typedef struct {
T_BOO overflow;                  /* verdadeiro: valor do contador >= valor
programado */
        T_ANA value;             /* valor atual do contador */
} str_data;

/* serviço de inicialização: requer alocação de memória para os dados de instância
*/

word FBINIT_sample (uint16 hinstance)
{
        return (sizeof (str_data));
}

/* serviço de ativação: algoritmo de contagem incremental */

void FBACT_sample (uint16 hinstance, str_data *data, str_arg *arg)
{
        if (R) data->value = 0;
        else if (CU && data->value < PV) (data->value)++;
        data->overflow = (data->value >= PV) ? (T_BOO)1 : (T_BOO)0;
}

/* conjunto de operações requeridas para a cópia dos parâmetros para o buffer
ISaGRAF */

#define BOO_VALUE   ((T_BOO *)value)
```

```
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* serviço de leitura dos parâmetros de retorno */

void FBREAD_sample (uint16 hinstance, str_data *data, uint16 parno, void
*value)
{
        switch (parno) {
          case FBLPNO_Q   : *BOO_VALUE = data->overflow; break;
          case FBLPNO_CV : *ANA_VALUE = data->value; break;
        }
}

/* serviço de declaração para a ligação dinâmica com o ISaGRAF núcleo */

ABP fbldef_sample (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "SAMPLE");
        *initproc = (IBP)FBINIT_sample;
        *readproc = (RBP)FBREAD_sample;
        return ((ABP)FBACT_sample);
}

/* fim do arquivo */
```

C.7.5 Técnicas de compilação e de integração

O ambiente de trabalho ISaGRAF não inclui qualquer compilador de "C" ou ferramenta de edição de ligações. Porém este capítulo explica as técnicas principais que podem ser aplicadas para a fácil utilização dos arquivos criados pelo Gerenciamento de Biblioteca ISaGRAF e os passa para as outras ferramentas tais como compiladores e de edição de ligações.

- ***Arquivos de código "C"***

Os arquivos de código fonte "C" de conversões, funções e blocos de função são colocados pelo Gerenciamento de Biblioteca ISaGRAF nos diretórios **ISAWIN\LIB\DEFS** e **ISAWIN\LIB\SRC**. O nome de um arquivo de código fonte é construído com o número correspondente da conversão, função ou bloco de função na biblioteca ISaGRAF. Estes são os nomes de arquivo utilizados:

\isawin\lib\defs\TACN0DEF.H	arquivo de definição padrão para todas as funções de conversão
\isawin\lib\src\GRCN0nnn.H	arquivo fonte de uma função de conversão
\isawin\lib\defs\GRUS0nnn.H	arquivo de definições de uma função
\isawin\lib\src\GRUS0nnn.C	arquivo fonte de uma função
\isawin\lib\defs\GRFB0nnn.H	arquivo de definições de um bloco de função
\isawin\lib\src\GRFB0nnn.C	arquivo fonte de bloco de função

(**nnn** é o número da conversão, função ou bloco de função)

Atenção: Se você renomeia ou copia um elemento com o Gerenciamento de Biblioteca ISaGRAF, os textos e as linhas de programação associadas não serão atualizadas automaticamente com o novo nome ou número do elemento. Eles devem ser atualizados manualmente no arquivo de código fonte "C".

O arquivo **\ISAWIN\LIB\USPNUMS** dá a relação entre os nomes e os números lógicos para todas as funções da biblioteca ISaGRAF. Eis um exemplo deste arquivo:

```
1       funct_A
10      funct_B
16      funct_C
```

O arquivo **\ISAWIN\LIB\FBLNUMS** dá a relação entre os nomes e os números lógicos para os blocos de função da biblioteca ISaGRAF. Eis um exemplo deste arquivo:

```
0       fbl_A
1       fbl_B
2       fbl_C
```

O arquivo **\ISAWIN\LIB\CNVNUMS** dá a relação entre os nomes e os números lógicos para as funções de conversão existentes na biblioteca ISaGRAF. Como um exemplo, eis o conteúdo deste arquivo para as conversões da biblioteca padrão:

```
0       SCALE
1       BCD
```

Estes arquivos são atualizados automaticamente pelo Gerenciamento de Biblioteca ISaGRAF toda vez que uma conversão, função ou bloco de função é criado, renomeado, copiado ou apagado. O Gerador de Código ISaGRAF automaticamente gera os seguintes arquivos quando um aplicativo é construído:

\isawin\apl\ppp\GRCN0LIB.C	Array de declaração de conversões referenciado no projeto
\isawin\apl\ppp\GRUS0LIB.C	Array de declaração das funções referenciado no projeto
\isawin\apl\ppp\GRFB0LIB.C	Array de declaração dos blocos de função referenciado no projeto

(**ppp** é o nome do projeto ISaGRAF)

Estes arquivos podem ser utilizados durante as operações de edição de ligações para construir um novo ISaGRAF núcleo dedicado a um projeto, que contém somente as conversões, funções e blocos de função utilizados no projeto.

Carregamento dos arquivos fonte para um sistema nativo

Os arquivos de código fonte e de definição em "C" criados pelo Gerenciamento de Biblioteca ISaGRAF podem ser carregados no sistema ISaGRAF destino, se suporta uma ferramenta de compilação nativa. Para fazer isto, a ferramenta padrão **TERMINAL** fornecida com o Windows pode ser utilizada.

Quando os arquivos fontes são gerenciados no sistema destino, os arquivos de definição têm que ser atualizados com uma nova operação de carregamento toda vez que uma interface de função é modificada com o Gerenciamento de Biblioteca ISaGRAF.

As linhas de comandos para carregar arquivos pode ser agrupado por exemplo em um arquivo batch e então pode ser iniciado do menu de ferramenta do ambiente de trabalho (veja o guia de usuário: Gerenciamento de programas).

Utilização de um compilador cruzada

Os arquivos fontes também podem ser gerenciados diretamente no seu PC, se o destino é também um PC, ou um compilador cruzado está disponível, executando no PC e gerando código para o sistema destino.

Neste caso, o usuário pode executar o Gerenciamento de Biblioteca ISaGRAF para completar e modificar as fontes de conversões, funções ou blocos de função.
As linhas de comandos para executar o compilador e a ligação podem ser agrupadas por exemplo em um arquivo batch e então iniciado do menu de ferramenta do ambiente de trabalho (veja o guia de usuário: Gerenciamento de programas).

Quando as conversões, funções e blocos de função são compilados no PC, o usuário tem que simplesmente carregar o novo ISaGRAF núcleo gerado (ligado com novos componentes) no sistema destino antes de executar os aplicativos. Se o destino é outro PC, o novo ISaGRAF núcleo gerado pode ser carregado na máquina destino utilizando um disquete ou por uma rede.

Edição de ligações com o ISaGRAF núcleo

Atenção:
O que está descrito a seguir são informações gerais que podem não corresponder exatamente com o seu sistema destino.
Em todo caso você pode consultar os arquivos readme e .TXT fornecidos com o disco destino.

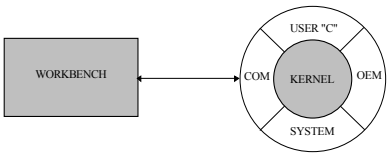
O disquete ISaGRAF destino contém muitos arquivos de utilidades para compilar e ligar as conversões, funções e blocos de função com as bibliotecas ISaGRAF núcleo.

Duas implementações existem:
- ISaGRAF monotarefa: todas as funções são executadas no mesmo programa
- ISaGRAF multitarefa: uma tarefa separada (ou linha) é dedicada para comunicação

Em qualquer caso, os componentes "C" desenvolvidos são agrupados nas mesmas bibliotecas: para o programador "C", não faz nenhuma diferença entre as versões monotarefa ou multitarefa. Para uma

versão monotarefa, as bibliotecas "C" são ligadas à monotarefa (geralmente chamada **isa**), sendo que para a versão multitarefa as bibliotecas são ligadas à tarefa núcleo (geralmente chamada **isaker**).

Sistema Desenvolvimento **Sistema Destino**

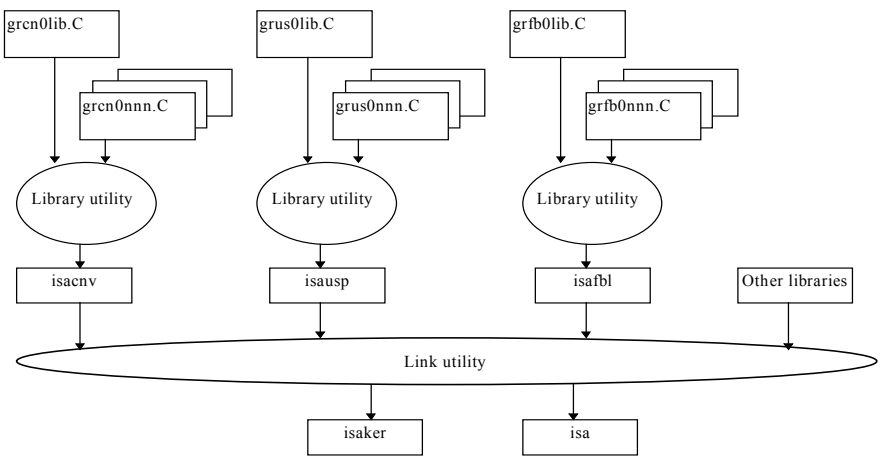


A parte interna do software ISaGRAF (KERNEL) é independente do hardware utilizado. Executa as instruções das linguagens IEC e gera seu próprio banco de dados.

A primeira etapa, ao fazer a ligação com o ISaGRAF núcleo, é construir bibliotecas de todas as conversões, funções e blocos de função necessárias para o projeto específico:

Biblioteca	conteúdo
ISAUSP	- arquivo objeto GRUS0LIB (array de funções declaradas) - arquivo objeto de cada função integrada
ISAFBL	- arquivo objeto GRFB0LIB (array de blocos de função declarados) - arquivo objeto de cada bloco de função integrado
ISACNV	- arquivo objeto GRCN0LIB (array de conversões declaradas) - arquivo objeto de cada função de conversão integrada

Então o programador tem que executar a edição de ligações entre estas bibliotecas e os outros arquivos objeto e bibliotecas do ISaGRAF núcleo. As diferentes fases dos componentes "C" são descritas no diagrama a seguir:



Esta é a lista exata dos módulos objetos e das bibliotecas que intervêm na edição das ligações:

Para construir isaker:		
Módulo Objeto:	**tast0mai**	
Módulo Objeto:	**tats0com**	
Biblioteca núcleo:	**isaker**	
Biblioteca núcleo:	**isaoem**	
Biblioteca de usuário:	**isausp**	funções definidas p/ usuário
Biblioteca de usuário:	**isafbl**	blocos de função definidos p/ usuário
Biblioteca de usuário:	**isacnv**	conversões definidas p/ usuário
Biblioteca núcleo:	**isasys**	
Bibliotecas de sistema:	(veja o manual do compilador utilizado)	

Para construir isa:		
Módulo Objeto:	**tast0mai**	
Módulo Objeto:	**tast0com**	
Biblioteca núcleo:	**isaker**	
Biblioteca núcleo:	**isatst**	
Biblioteca núcleo:	**isaoem**	
Biblioteca de usuário:	**isausp**	funções definidas p/ usuário
Biblioteca de usuário:	**isafbl**	blocos de função definidos p/ usuário
Biblioteca de usuário:	**isacnv**	conversões definidas p/ usuário
Biblioteca núcleo:	**isasys**	
Bibliotecas de sistema:	(veja o manual do compilador utilizado)	

O programador deve respeitar exatamente a ordem descrita dos módulos objetos e as bibliotecas nas figuras anteriores. Os arquivos de código de objetos e de bibliotecas têm extensões padrão (**.lib** ", **.obj** ", **.l** ", **.r** "...) de acordo com o sistema designado.

Opções de compilação e de edição de ligações

As opções a especificar nas linhas de comandos do compilador e do editor de ligações dependem do sistema destino e do tipo de operações realizadas nas conversões, funções e blocos de função. Algumas operações necessitam de outras bibliotecas de sistema (matemáticas, gráficas...) durante a edição de ligação.

Todos os arquivos de código fonte "C" do ISaGRAF núcleo têm que ser compilados no modelo de memória **LARGE**. O programador deve utilizar o mesmo modelo para a compilação das conversões, funções e blocos de função.

Uma constante especial deve ser definida na linha de comando para a compilação dos procedimentos e das funções de conversão. Indica o tipo do sistema destino e processadores e permite descrever as

conversões, funções e blocos de função independentes do sistema destino. A seguir, os nomes destes valores constantes:

DOS .....para sistemas ambiente DOS (processador INTEL)
ISAWNT .....para sistemas ambiente Windows-NT (processador INTEL)
OS9.....para sistema ambiente OS9 (processador MOTOROLA)
VxWorks.....para sistema ambiente VxWorks (processador MOTOROLA)

Os arquivos de comando **ISACC** fornecidos com o software ISaGRAF destino mostra como definir a constante conveniente ao seu sistema.

Compiladores suportados

Os seguintes compiladores podem ser utilizados para o desenvolvimento dos procedimentos e das funções de conversão e suas ligações com a bibliotecas do ISaGRAF Núcleo:

Microsoft MSC 7.00 compiler	para os sistemas ambiente DOS
Microsoft MSVC 4.00 compiler	para os sistemas ambiente Windows-NT
Microware ULTRA-C compiler	para os sistemas ambiente OS-9
Tornado 1.0; GNU Toolkit 2.6	para os sistemas ambiente VxWorks

Entre em contato com a CJ International sobre a utilização de outros compiladores.

Resumo

A seguir, o resumo das operações a efetuar para o desenvolvimento de um novo elemento (conversão, função e bloco de função).

⇨**1.** com o Gerenciamento de Biblioteca ISaGRAF, criar um novo elemento: dê a ele um nome e um texto de comentário. O quadro do código fonte “C” é automaticamente gerado.

⇨**2.** Com o Gerenciamento de Biblioteca ISaGRAF, descrever a interface (parâmetros de chamada e de retorno), se o elemento é uma função ou bloco de função. O arquivo de definições "C" é automaticamente gerado.

⇨**3.** Com o Gerenciamento de Biblioteca ISaGRAF, entrar com o texto da observação técnica (Manual do Usuário) para este elemento.

⇨**4.** Com o Gerenciamento de Biblioteca ISaGRAF, completar o arquivo de código fonte "C", entrando com a programação "C" do algoritmo de conversão, função ou bloco de função. O código fonte do elemento está agora completo. Note que outro editor pode ser utilizado.

⇨**5.** Selecionar a opção "**Show logical number**" do Gerenciamento de Biblioteca para saber o número utilizado nos nomes dos arquivos de código “C” associados ao elemento. Este número é utilizado no nome dos caminhos dos arquivos fonte “C” e “H” correspondentes.

⇨**6.** Copiar / Carregar os arquivos “C” e ”H” para o seu destino (se compilador nativo) ou para o ambiente correspondente (se compilador cruzado) no qual as bibliotecas ISaGRAF destino e tarefas estão instaladas.

⇨**7.** Executar o compilador "C" no novo arquivo fonte e corrigir os erros de sintaxe, se existirem.

⇨**8.** Inserir o nome do serviço de declaração associado ao elemento no arquivo "**GR??0LIB.C**" que contém o array dos elementos inseridos.

⇨**9.** Executar o compilador "C" para compilar o arquivo "**GR??0LIB.C**".

⇨**10.** Inserir o nome do módulo objeto na lista de arquivos objetos utilizados para construir a biblioteca correspondente.

⇨**11.** Executar o construtor de biblioteca "C". Executar o editor de ligações "C" para o novo Núcleo.

⇨**12.** Instalar o novo núcleo gerado sobre o sistema destino.

⇨**13.** Escrever um aplicativo ISaGRAF de teste para a validação e a manutenção do elemento desenvolvido.

C.8 O link Modbus

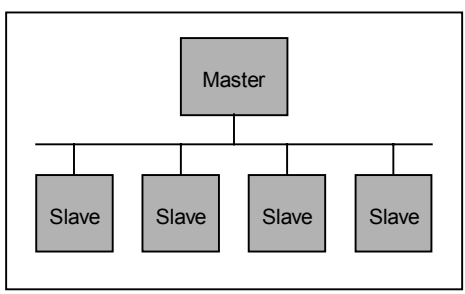
Uma vez que o aplicativo está completamente desenvolvido e testado, você pode conectá-lo a um processo de controle de processo.

O ISaGRAF é um sistema aberto dispondo de uma grande variedade de possibilidades de rede.
A rede industrial mais simples é o protocolo padrão MODBUS/MODICON que está disponível na maior parte dos sistemas de controle de processo e que não necessita de nenhum link serial (RS232, RS485, Current Loop).

O protocolo de comunicação do depurador ISaGRAF é compatível com o MODBUS permitindo o acesso de leitura/escrita às variáveis a partir de um Modbus principal.

C.8.1 Rede e protocolo MODBUS

Uma rede Modbus é composta de só uma estação principal (usualmente um sistema de controle de processo) e uma ou mais estações escravas (usualmente PLCs).



O Principal envia uma solicitação por vez (utilizando um número escravo) e aguarda a resposta do escravo antes de enviar uma nova solicitação.

Cada quadro contém um número escravo, um número de solicitação e uma data correspondente, e um checksum de 16 bits (CRC).

Se nenhuma resposta é recebida após um timeout, a solicitação pode ser repetida um certo número de vezes antes do principal declarar o escravo “desconectado”.
O valor do timeout e o número de tentativas são definidos na estação principal para ajustar as necessidades dos escravos (dependendo do aplicativo, Tc...).

Se um erro ocorre em um durante o processamento da solicitação, o escravo pode emitir uma mensagem, de erro ao invés de enviar um quadro de resposta esperado.

O Modbus é um protocolo Modicon mas não um padrão internacional. Há diversas implementações diferentes dos protocolos compatíveis 'Modbus', com diversas possibilidades, por exemplo:

– Lista de códigos de funções implementadas
– Mapa de endereços
– RTU (códigos binários) u protocolo ASCII
Tc...

C.8.2 Implementação do ISaGRAF

Acesso às variáveis do aplicativo

O link de comunicação ISaGRAF reconhece cinco códigos de função Modbus:

1	ler N bits
3	ler N palavras
5	escrever 1 bit
6	escrever 1 palavra
16	escrever N palavras

É possível o acesso às variáveis do aplicativo ISaGRAF através de seus 'endereços de rede' se, obviamente, eles estão definidos no dicionário do ambiente de trabalho. Estas variáveis podem ser:
– Variáveis booleanas ou analógicas
– Variáveis internas, de entradas ou de saída
– Variáveis locais ou globais.

Para escrever uma variável booleana, as funções 5, 6 ou 16 podem ser utilizadas. Um valor TRUE = escrever é qualquer um diferente de zero.

Para ler uma variável booleana, as funções 1 ou 3 podem ser utilizadas. Com a função 1, os valores são recuperados em um campo de bit; com a função 3, eles são recuperados em bytes (um valor TRUE corresponde à 0xFFFF).

Para escrever uma variável analógica, as funções 6 ou 16 podem ser utilizadas. O valor é um inteiro de 16 bits compreendido entre -32.768 até +32.767 (as variáveis do ISaGRAF destino têm 32 bits).

Para ler uma variável analógica, a função 3 deve ser utilizada. O valor lido é um inteiro de 16 bits compreendido entre -32.768 até +32.767. No lado do destino, as variáveis analógicas são de 32 bits, por isso, um valor no destino que ultrapassa o limite de 16 bits (positivo ou negativo) é lido com o valor máximo de 16 bits (positivo ou negativo).

Não é possível acessar as variáveis reais com uma solicitação Modbus.

atenção:
A implementação ISaGRAF não gerencia os códigos de erros do tipo 'endereço Modbus desconhecido'.

Anotações:

slv número escravo
nbw número de palavras

nbb	número de bytes
nbi	número de bits
addH	endereço rede (Byte Superior)
addL	endereço rede (Byte Inferior)
vH	valor (Byte Superior)
vL	valor (Byte Inferior)
V	valor do Byte
bfd	campo de bit (nbb Bytes)
crcH	checksum (Byte Superior)
crcL	checksum (Byte Inferior)

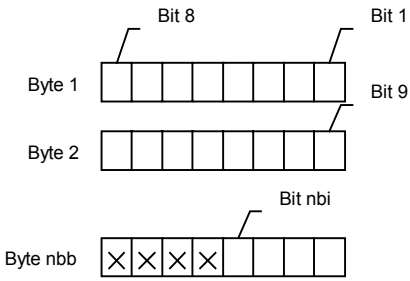
FUNÇÃO 1: ler N bits

Ler nbi bits (Booleanos) começando pelo endereço de rede addH/addL

Pergunta	slv	01	addH	addL	00	nbi	crcH	crcL

Resposta	slv	01	nbb	bfd	...		crcH	crcL
				Byte 1		Byte nbb		

bfd é um campo de bits de nbb Bytes com o seguinte formato:



O bit 1 corresponde ao valor da variável no endereço de rede addH/addL.
O bit nbi corresponde ao valor da variável no endereço de rede addH/addL + nbi -1.
X significa valor indefinido.

FUNÇÃO 3: ler N palavras

Ler nbw palavras começando pelo endereço de rede addH/addL

Pergunta	slv	03	addH	addL	00	nbw	crcH	crcL

Resposta	slv	03	nbb	vH	vL	...	crcH	crcL

nbb corresponde ao número de bytes vH, vL

FUNÇÃO 5: escrever 1 bit

Escrever um bit (Booleano) no endereço de rede addH/addL

Pergunta	slv	05	addH	addL	vH	00	crcH	crcL

Resposta	slv	05	addH	addL	vH	00	crcH	crcL

FUNÇÃO 6: escrever 1 palavra

Escrever uma palavra no endereço de rede addH/addL

Pergunte	slv	06	addH	addL	vH	vL	crcH	crcL

Resposta	slv	06	addH	addL	vH	vL	crcH	crcL

FUNÇÃO 16: escrever N palavras

Escrever nbw palavras começando pelo endereço de rede addH/addL (nbb = 2nbw)

Pergunta	slv	10	addH	addL	00	nbw	nbb	vH	vL	...	crcH	crcL

Resposta	slv	10	addH	addL	00	nbw	crcH	crcL

Exemplos:

– Função 1: ler 15 bits começando pelo endereço de rede 0x1020, no escravo 1

Pergunta	01	01	10	20	00	0F	79	04

Resposta	01	01	02	00	12	39	F1

O valor lido é 0x0012, que é igual a 00000000 00010010 na representação campo de bits.
Neste exemplo, as variáveis 0x1029 e 0x102C são TRUE, todas as outras são FALSE.

– Função 16: escrever 2 palavras no endereço 0x2100, no escravo 1. Os valores escritos são 0x1234 e 0x5678.

Pergunta	01	10	21	00	00	02	04	12	34	56	78	1C	CA

Resposta	01	10	21	00	00	02	4B	F4

Transferência de arquivo

Comparado aos barramentos de campo modernos, o protocolo Modbus oferece serviços muito pobres se ele não é complementado pelos códigos de função de fabricantes específicos.

Na nossa situação, a execução do ISaGRAF em uma base computadorizada poderosa e flexível gera duas restrições ao protocolo Modbus:
- Só é possível o acesso às variáveis ISaGRAF
- É difícil executar transferências rápidas de uma grande quantidade de dados

Por estas razões o ISaGRAF oferece um conjunto de solicitações "do tipo Modbus" para a transferência de arquivos ou um protocolo de "gerenciamento de arquivos remoto". Estas facilidades devem ser implementadas para validar:
- o carregamento de arquivo Binário ou ASCII
- a releitura de arquivo Binário ou ASCII
- a troca dinâmica de dados através de um arquivo compartilhado virtual ou físico

Portanto, com o link de comunicação ISaGRAF, qualquer aplicativo "independente do ISaGRAF" pode facilmente comunicar-se com um destino remoto.

O protocolo é baseado nos seguintes conceitos:
- O arquivo no ISaGRAF destino é chamado **remote file**
- O arquivo no ISaGRAF principal é chamado **local file**
- Cada byte em um arquivo é acessado com uma **base address** de 32 bit e um **byte address** de 16 bit

Existem solicitações para a seleção do nome do arquivo remoto, o endereço de base e para ler ou escrever os dados do arquivo remoto utilizando o endereço de byte de 16 bits.

FUNÇÃO 17: escrever dados

nbb corresponde ao número de bytes vH, vL

Pergunta	slv	11	addH	addL	00	nbb	nbb	vH	vL	...	crcH	crcL

Reposta	slv	11	addH	addL	00	nbb	crcH	crcL

O significado destas solicitações diferem de acordo com o alcance do endereço addH/addL:

- **0xF000: Inicializa o nome do arquivo remoto**
  nbb corresponde ao número de caracteres para o nome do arquivo, especificado nos campos vH vL (neste caso Superior (H) e Inferior (L) não têm significado) e **incluindo \0** para o fim da cadeia de caracteres.
  Se o arquivo não existe, ele é criado com os atributos "writable + readable + executable" (de escrita, de leitura, executável).
- **0xF002: Modificar o endereço de base com um valor especificado**
  nbb deve ser igual a 4.O primeiro byte vH/vL corresponda à palavra Superior do valor especificado. Qualquer valor de 32 bits é aceito.
  Todas as solicitações futuras de leitura e de escrita utilizarão este endereço de base. Quando esta solicitação não é utilizada, o valor do endereço de base padrão é zero.
- **0xF004: Apaga o arquivo**
  nbb deve ser igual a zero.
  O arquivo é apagado se ele existe e se é possível fazê-lo.
- **Maior que 0xF004: Reservado**
- **Menor que 0xF000: Escreve os bytes**

O endereço especificado no qual os bytes devem ser escritos é especificado no addH/addL. Deve ser menor que F000. Os bytes especificados (nbb bytes especificado nos campos vH vL em que Superior (H) e Inferior (L) não podem mais ter significado) são escritos nas ordem dada (da esquerda para a direita) para o nome do arquivo remoto selecionado previamente. O endereço de começo escrito, é o endereço especificado acrescentado ao endereço básico previamente selecionado. Se os endereços de acesso resultantes excedem o tamanho de arquivo atual, o arquivo é estendido. Você não pode reduzir o tamanho de arquivo.

FUNÇÃO 18: ler dados

Pergunta	slv	12	addH	addL	00	nbb	crcH	crcL

Resposta	slv	12	nbb	V	V	...	crcH	crcL

O endereço especificado no qual os bytes são lidos está especificado em addH/addL. Deve ser menor que F000. Leia o número especificado (nbb) de bytes, do nome de arquivo remoto previamente selecionado, a partir de endereço especificado (addH/addL com qualquer valor de 16 bits) acrescido ao endereço de base previamente selecionado.
Os Valores são recuperados (campos V da esquerda para a direita) na ordem em que são lidos no arquivo.

Exemplo:

Selecionar o nome do arquivo remoto: 'target.fil'.

Pergunta	01	11	F0	00	00	0B	0B	74	...	00	25	9F

Resposta	01	11	F0	00	00	0B	8F	0E

Selecionar os endereços de base: 0x10000.

Pergunta	01	11	F0	02	00	04	04	00	01	00	00	76	11

Resposta	01	11	F0	02	00	04	6E	CA

Escrever 4 bytes: endereço absoluto 0x107D0, valores 01,02,03,04.

Pergunta	01	11	07	D0	00	04	04	01	02	03	04	28	6F

Resposta	01	11	07	D0	00	04	FC	87

Ler 4 bytes: endereço absoluto 0x107D0.

Pergunta	01	12	07	D0	00	04	B8	87

Resposta	01	12	04	01	02	03	04	58	7D

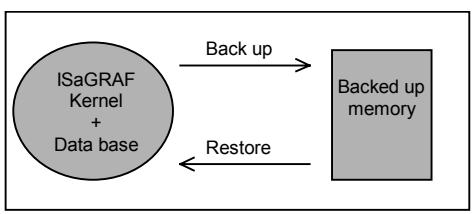
C.9 Gerenciamento da falta de energia

C.9.1 Bases

O gerenciamento das falhas por falta de energia é algo muito crítico em um aplicativo, por três razões:
– depende das especificação do processo
– depende das capacidades do hardware
– depende dos métodos de programação

Portanto, a solução do ISaGRAF para o gerenciamento de falta de energia não é um método completo e universal, mas um conjunto de princípios, métodos e ferramentas que devem ser combinados de uma maneira particular para cada aplicativo, ou pelo menos para cada hardware.

Para permitir que o sistema de controle de processo reinicie corretamente após uma falta de energia, 3 problemas devem ser solucionados:



– Efetuar uma cópia de segurança dos dados (backup)
– Detectar a ocorrência de uma falta de energia na partida (início)
– Restaurar os dados da cópia de segurança

O segundo problema não permite a solução de software padrão, mas o fornecedor do sistema pode fornecer as ferramentas para ter acesso ao estado do hardware a partir de um aplicativo ISaGRAF ou de um programa C.

Além disso, é importante decidir que dados salvar e recuperar. Vamos definir dois tipos de dados:
– Variáveis do aplicativo:
  Variáveis de processo tais como número de itens processadas, data da falta de energia, valores dos parâmetros do aplicativo etc. ...
  Variáveis de programa tais como contadores, temporizadores, valores intermediários e flags.

– Estado do programa:
  Tais como referência de etapas ativas, estado de cada programa C, etc. ...

Estes dois casos e as soluções ISaGRAF correspondentes são descritas nos capítulos a seguir.

C.9.2 Cópia de segurança das variáveis do aplicativo

Variáveis não voláteis

O editor de variável do ambiente de trabalho permite selecionar o atributo "não volátil" para cada variável interna (exceto de E/S) .

No fim de cada ciclo do destino os valores não voláteis são copiados em uma memória especial, normalmente uma RAM alimentada por bateria.

Na partida, se pelo menos uma variável tem um atributo "não volátil", o ISaGRAF procura pelas variáveis não voláteis:
- se o mesmo aplicativo estava em execução antes, o ISaGRAF reconhece os valores salvos e os atribui a todas as variáveis 'não voláteis'.
- se o aplicativo anterior era diferente ou se nenhum aplicativo estava em execução, o ISaGRAF reconhece que os valores 'não voláteis' não são válidos e reconfigura todas as variáveis 'não voláteis' com o valor zero (reset).

A especificação da área de memória utilizada para armazenar os diferentes tipos de variáveis é especificado na área de trabalho, no menu **Make**: **Application run time option ; retained variables.**
A cadeia especificada deve ter o seguinte formato:

boo_add , boo_size , ana_add , ana_size , tmr_add , tmr_size , msg_add , msg_size

em que:

boo_add: Endereço hexadecimal utilizado para armazenar variáveis booleanas. Deve ser sempre diferente de zero.

boo_size: Tamanho hexadecimal, em bytes, disponível para este endereço. É necessário um byte por variável booleana a ser salva.

ana_add: Endereço hexadecimal utilizado para armazenar variáveis analógicas. Deve ser sempre diferente zero.

ana_size: Tamanho hexadecimal, em bytes, disponível para este endereço. No mínimo são sempre necessários quatro bytes por variável analógica a ser salva.

tmr_add: Endereço hexadecimal utilizado para armazenar variáveis de temporização. Deve ser sempre diferente de zero.

tmr_size: Tamanho hexadecimal, em bytes, disponível para este endereço. São necessários cinco bytes por variável de temporização a ser salva.

msg_add: Endereço hexadecimal utilizado para armazenar variáveis de mensagem. Deve ser sempre diferente de zero.

msg_size: Tamanho hexadecimal, em bytes, disponível para este endereço. São necessários 256 bytes por mensagem a ser salva.

Exigências:
- Todos os campos de todos os tipos devem ser especificados mesmo se você não precisar fazer uma cópia de segurança de todos os tipos de variáveis. Neste caso, você precisa especificar um tamanho igual a zero (com exceção das variáveis analógicas para as quais você deve especificar um tamanho

de quatro bytes) e qualquer endereço diferente de zero deve ser especificado para o(s) tipo(s) de variável a não ser salva.

Exemplo:
Suponhamos que necessitemos fazer uma cópia de segurança de:

20 Variáveis booleanas
0 Variável analógica
0 Variável de temporização
3 Variáveis de mensagem

A memória de segurança (backup) está localizada no endereço hexadecimal 0xA2F200.
Vamos supor que:

As variáveis booleanas serão armazenadas no endereço 0xA2F200 com o tamanho exato de 20 bytes.
Um tamanho mínimo de 4 bytes para as variáveis analógicas a salvar são armazenadas no endereço 0xA2F214.
O endereço dummy para as temporizações é 0xA2F200 e especificado com um tamanho igual a zero.
As mensagens são armazenadas no endereço 0x A2F218 com o tamanho exato necessário de 256*3 bytes.

Portanto, a cadeia inserida no ambiente de trabalho deve ser:

```
A2F200,14,A2F214,4,A2F200,0,A2F218,300
```

Chamada da função SYSTEM

Se a maioriadas variáveis do aplicativo devem ser salvas, é preferível utilizar as facilidades da função SYSTEM para a manipulação de um conjunto de variáveis (para mais informações sobre a função SYSTEM veja o manual do usuário). Note que neste caso, o salvamento e a restauração são gerenciados pelo programados no nível do aplicativo.

Em primeiro lugar, você precisa definir a localização da memória de segurança para um tipo especificado de variável ou todos os tipos de variáveis:

<new_address> := SYSTEM(SYS_INITxxx,<address>);

em que:

- <address> é a localização do endereço de memória de segurança (valor 16# para o formato hexadecimal). O endereço deve ser par, caso contrário as operações não são executadas.
- SYS_INITxxx pode ser:
  - SYS_INITBOO para definir a localização da memória de segurança para todas as variáveis booleanas.
  - SYS_INITANA para definir a localização da memória de segurança para todas as variáveis analógicas.
  - SYS_INITTMR para definir a localização da memória de segurança para todas as variáveis de temporização.
  - SYS_INITALL para definir a localização da memória de segurança para todas as variáveis booleanas, analógicas e de temporização.
- <new_address> localiza o próximo endereço livre, que dizer <address> + tamanho das variáveis salvas (backup) (em bytes) de acordo com SYS_INITxxx. Permite verificar o tamanho necessário da memória de segurança. Se a operação não foi executada <new_address> assume o valor zero.

Em seguir, você pode solicitar uma cópia de segurança (backup). Este procedimento pode ser chamado a qualquer momento no aplicativo. A cópia de segurança só é efetuada no fim do ciclo atual e uma única vez. Se o hardware libera uma entrada booleana ou uma função C informando o usuário quando a falta de energia acontece, e permite pelo menos um retardo de ciclo ISaGRAF antes do corte de energia, portanto a cópia de segurança só é possível quando uma falta de energia é detectada:

<error> :=SYSTEM(SYS_SAVxxx,0);

em que:
- SYS_SAVxxx pode ser:
  - SYS_SAVBOO para solicitar a cópia de segurança de todas as variáveis booleanas.
  - SYS_SAVANA para solicitar a cópia de segurança de todas as variáveis analógicas.
  - SYS_SAVTMR para solicitar a cópia de segurança de todas as variáveis de temporização.
  - SYS_SAVALL para solicitar a cópia de segurança de todas as variáveis booleanas, analógicas e de temporização.
- <error> obtém um estado de erro diferente de zero quando a operação não foi executada (SYS_INITxxx não foi chamado).

Finalmente as variáveis devem ser restauradas. Este procedimento pode ser chamado a qualquer momento no aplicativo. A restauração só é efetuada no fim do ciclo atual e uma única vez. Para assegurar que os dados salvos são válidos, uma variável analógica deve ser configurada com um valor constante utilizada como uma assinatura:

<error> := SYSTEM(SYS_RESTxxx,0);

em que:
- SYS_RESTxxx pode ser:
  - SYS_RESTBOO para restaurar todas as variáveis booleanas.
  - SYS_RESTANA para restaurar todas as variáveis analógicas.
  - SYS_RESTTMR para restaurar todas as variáveis de temporização.
  - SYS_RESTALL para restaurar todas as variáveis booleanas, analógicas e de temporização.
- <error> obtém um estado de erro diferente de zero quando a operação não é executada (SYS_INITxxx não foi executado).

A seguir, o resumo dos comandos da função SYSTEM para o gerenciamento das cópias de segurança das variáveis:

Comando		**Significado**
palavra chave predefinida	**Valor**	
SYS_INITBOO	16#20	init cópia segurança booleana
SYS_SAVBOO	16#21	salvar as booleanas
SYS_RESTBOO	16#22	restaurar as booleanas
SYS_INITANA	16#24	init cópia segurança analógica
SYS_SAVANA	16#25	salvar as analógicas
SYS_RESTANA	16#26	restaurar as analógicas
SYS_INITTMR	16#28	init cópia segurança temporizações
SYS_SAVTMR	16#29	salvar as de temporizações
SYS_RESTTMR	16#2A	restaurar as de temporizações
SYS_INITALL	16#2C	init cópia segurança todos os tipos
SYS_SAVALL	16#2D	salvar todos os tipos

SYS_RESTALL	16#2E	restaurar todos os tipos

Comando (palavra chave predefinida)	**Argumento**	**Valor de retorno**
SYS_INITxxx	endereço memória	próximo endereço livre
SYS_SAVxxx	0	zero se OK
SYS_RESTxxx	0	zero se OK

Implementação personalizada

Finalmente, utilizando as funções ou os blocos de função em C você pode desenvolver procedimentos de usuário específicos para ter acesso à memória de segurança sustentada por bateria, para armazenar e restaurar variáveis a qualquer momento no aplicativo.

Exemplos:

1) Procedimento dedicado a um aplicativo:
backup, restore_temp, restore_date, restore_cpt são procedimento de usuário em C.

```
backup(temperature, date, cnt);        armazena 3 dados críticos

temperature := restore_temp();         restaura temperatura
date := restore_date();                restaura data
cnt := restore_cnt();                  restaura contador
```

2) Procedimentos de uso geral:
backup_init, backup, backup_link, restore são procedimentos de usuário em C.

```
save_id := backup_init(address, size);     aloca um array armazenado na
                                           memória de segurança.
backup(save_id, cpt1, 3);                  salva cpt1 como o 3º elemento.

rest_id := backup_link(address, size)      ligação da memória de segurança.
cpt1 := restore(rest_id, 3);               restaura o valor salvo de cpt1.
```

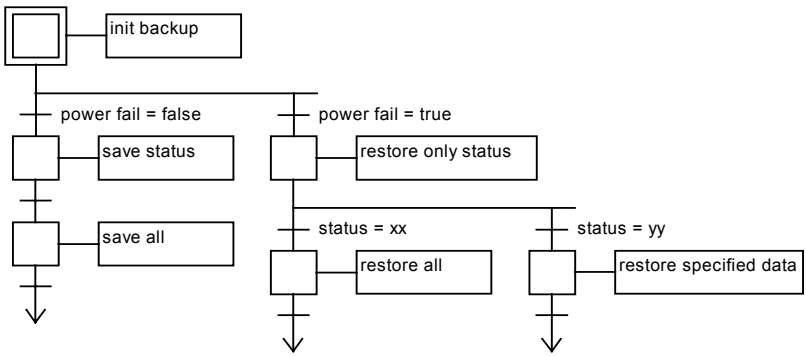
C.9.3 Cópia de segurança do estado de um programa

É possível armazenar cada estado de qualquer programa de aplicativo, mas pode ser perigoso restaurar cada programa com o estado da última cópia de segurança, por pelo menos três razões:

- Alguns processos requerem operações específicas antes de reiniciar
- É enfadonho lidar com cada estado de um aplicativo completa
- Alguns recursos externos, como programas C, periféricos, etc. não podem ser reiniciados automaticamente

A melhor solução é fazer uma cópia de segurança das variáveis analógicas ou booleanas para descrever o estado do processo se o programador pensa que estas informações podem ser úteis nas etapas de reinicialização.
Então deve ser possível, a partir de uma 'imagem' simples e inteligente do processo, iniciar, interromper ou congelar os programas SFC e inicializar as variáveis de modo a colocar o aplicativo num estado adequado. Mas nenhum procedimento de início automático é fornecido pelo ISaGRAF.

Exemplo:


init backup
power fail = false
power fail = true
save status
restore only status
save all
status = xx
status = yy
restore all
restore specified data

C.10 Anexo: Lista de erros e descrição

Lista de erros:

Código	Mensagem	Tipo
1	cannot allocate memory for run time data base	sistema
2	incorrect application data base (Motorola/Intel)	aplicativo
3	cannot allocate communication mailbox	sistema
4	cannot link kernel data base	sistema
5	time-out sending question to kernel	sistema
6	time-out waiting answer from kernel	sistema
7	cannot init communication	sistema
8	cannot allocate memory for retained variables	aplicativo
9	application stopped	aplicativo
10	too many simultaneous N or P actions	aplicativo
11	too many simultaneous setting actions	aplicativo
12	too many simultaneous resetting actions	aplicativo
13	unknown TIC instruction	aplicativo
16	cannot answer read data request	sistema
17	cannot answer write data request	sistema
18	cannot answer debugger session request	sistema
19	cannot answer modbus request	sistema
20	cannot answer debugger application request	sistema
21	cannot answer debugger	sistema
23	unknown request code	sistema
24	Ethernet communication error	sistema
25	communication synchro error	sistema
28	cannot allocate memory for application	sistema
29	cannot allocate memory for application update	sistema
30	unknown OEM key code	aplicativo
31	cannot init boolean input board	aplicativo
32	cannot init analog input board	aplicativo
33	cannot init message input board	aplicativo
34	cannot init boolean output board	aplicativo
35	cannot init analog output board	aplicativo
36	cannot init message output board	aplicativo
37	cannot input boolean board	aplicativo
38	cannot input analog board	aplicativo
39	cannot input message board	aplicativo
40	cannot output boolean output variable	aplicativo
41	cannot output analog output variable	aplicativo
42	cannot output message output variable	aplicativo
43	cannot operate boolean variable	aplicativo
44	cannot operate analog variable	aplicativo
45	cannot operate message variable	aplicativo

46	cannot open board	aplicativo
47	cannot close board	aplicativo
50	cannot overwrite boolean output variable	programa
51	cannot overwrite analog output variable	programa
52	cannot overwrite message output variable	programa
61	unknown system request code	programa
62	sampling period overflow	programa
63	user function not implemented	aplicativo
64	integer divided by zero	programa
65	conversion function not implemented	aplicativo
66	function block not implemented	aplicativo
67	standard function not implemented	aplicativo
68	real divided by zero	programa
69	invalid operate parameters	aplicativo
72	application symbols cannot be modified	aplicativo
73	cannot update: different set of boolean variables	aplicativo
74	cannot update: different set of analog variables	aplicativo
75	cannot update: different set of timer variables	aplicativo
76	cannot update: different set of message variables	aplicativo
77	cannot update: cannot find new application	aplicativo
> 100	specific OEM error code, ask your supplier for more details	

Pode-se distinguir 3 tipos de erros:

– Erros de sistema:
Tais problemas provavelmente são devidos ao software ou hardware destino, não tendo relação com a configuração do aplicativo ou com a execução dos programas.
Tente uma reinicialização drástica (desligue a alimentação) do seu destino e tente executar outros aplicativos.
Estes erros devem ser informados para o seu suporte técnico ISaGRAF.

– Erros de aplicativo:
Tais problemas são devido aos parâmetros, tamanho ou conteúdo do aplicativo.
Estes erros devem desaparecer quando um aplicativo conhecido e validado previamente é carregado. Se o problema ainda persiste, ele faz parte dos erros de sistema (veja a lista acima).

– Erros de programa:
Tais problemas são causados por uma seqüência específica de um programa.
Este tipo de erro deve desaparecer quando o aplicativo é executado no modo ciclo-a-ciclo ou quando o programa crítico é interrompido.

Descrição dos erros:

1. cannot allocate memory for run time data base	***sistema***

Não há memória disponível. Verifique o hardware.

2. incorrect application data base (Motorola/Intel)	***aplicativo***

O arquivo do aplicativo, carregado ou salvo (backup) não está correto. Este erro aparece se o aplicativo é gerado para a INTEL e carregado na MOTOROLA (e vice-versa) ou se o arquivo foi alterado.

3. cannot allocate communication mailbox	***sistema***

Este erro é gerado pela tarefa de comunicação se não pode alocar espaço 3 para comunicação entre as tarefas.

4. cannot link kernel data base	***sistema***

Este erro é gerado pela tarefa de comunicação se não pode localizar um núcleo em execução com o número escravo especificado na sua linha de comando.

5. time-out sending question to kernel	***sistema***

A tarefa de comunicação não pode enviar uma solicitação para o núcleo. O núcleo provavelmente não está em execução ou está ocupado.

6. time-out waiting answer from kernel	***sistema***

A tarefa de comunicação não pode receber uma resposta do núcleo. O núcleo provavelmente não está em execução ou está ocupado.

7. cannot init communication	***sistema***

Este aviso é gerada quando a camada de comunicação não pode inicializar a ligação física. Este aviso também é mostrado se nenhum caminho de comunicação é especificado. Isto não impede a execução correta do destino, mas não pode comunicar.

8. cannot allocate memory for retained variables	***aplicativo***

O ISaGRAF não pode gerenciar as variáveis não voláteis. Pode ser por duas razões:
- a cadeia de caracteres passada como um parâmetro ao host destino não está sintaticamente correta
- o tamanho da memória especificado para cada bloco não é suficiente

Verifique a sintaxe do seu parâmetro 'variável não volátil' e assegure-se de reduzir o número de variáveis não voláteis.

9. application stopped	***aplicativo***

Esta mensagem é mostrada cada vez que o aplicativo é interrompido a partir do depurador.

10. too many simultaneous N or P actions	***aplicativo***

Este erro é gerado quando um dos ciclos do destino deve executar muitas ações P ou blocos cíclicos não armazenados. É possível localizar o problema no modo CC. O número máximo de ações simultâneas é 2 + 4 por programa SFC.

11. too many simultaneous setting actions	***aplicativo***

Este erro é gerado quando um dos ciclos do destino deve executar muitas ações Set (executada quando a etapa torna-se ativa). Mesmo procedimento do anterior.

12. too many simultaneous resetting actions	***aplicativo***

Este erro é gerado quando um dos ciclos do destino deve executar muitas ações Reset (executada quando a etapa torna-se não ativa). Mesmo procedimento do anterior.

13. unknown TIC instruction	***aplicativo***

O núcleo detectou um erro no código do aplicativo de um programa (chamado **T**arget **I**ndependent **C**ode). Pode ser por duas razões:
- um programa externo escreve no código do aplicativo. Tente localizar o problema no modo ciclo-a-ciclo (modo CC) e assegure-se de que todas as interfaces de E/S tenham parâmetros corretos.
- seu destino opera com um conjunto de instruções reduzido e seu aplicativo utiliza uma instrução ou um tipo de variável que não está autorizado.

16. cannot answer read data request	***sistema***

A detecção de um erro de comunicação em resposta a uma função de solicitação específica ISaGRAF Modbus código 18 (ler arquivo). Verifique a conexão e a configuração em ambos os lados, principal e destino.

17. cannot answer write data request	***sistema***

A detecção de um erro de comunicação em resposta a uma função de solicitação específica ISaGRAF Modbus código 17 (escrever arquivo). Verifique a conexão e a configuração em ambos os lados, principal e destino.

18. cannot answer debugger session request	***sistema***

A detecção de um erro de comunicação em resposta a uma solicitação do depurador. Verifique a conexão e a configuração em ambos os lados, principal e destino.

19. cannot answer modbus request	***sistema***

A detecção de um erro de comunicação em resposta a uma solicitação Modbus. Verifique a conexão e a configuração em ambos os lados, principal e destino.

20. cannot answer debugger application request	***sistema***

A detecção de um erro de comunicação em resposta a uma solicitação do depurador. Verifique a conexão e a configuração em ambos os lados, principal e destino.

21. cannot answer debugger	***sistema***

A detecção de um erro de comunicação em resposta a uma solicitação do depurador. Verifique a conexão e a configuração em ambos os lados, principal e destino.

23. unknown request code	***sistema***

Uma solicitação do depurador não faz sentido.

24. Ethernet communication error	***sistema***

Esta mensagem aparece toda vez que a conexão é fechada quando o depurador é fechado: o sistema está funcionando OK. Por outro lado, esta mensagem significa que um erro de comunicação Ethernet foi detectado. Verifique a conexão e a configuração em ambos os lados, principal e destino.
Uma mensagem suplementar pode ser mostrada:
1: error while sending or receiving
2: error while creating the socket
3: error while binding or listening the socket
4: error while accepting a new client

25. communication synchro error	***sistema***

Sincronização ruim entre a tarefa de comunicação no destino e no principal. Verifique a conexão e a configuração (parâmetros de comunicação) em ambos os lados, principal e destino.

28. cannot allocate memory for application	***sistema***

Sem memória disponível. Verifique o hardware, de acordo com o tamanho do aplicativo.

29. cannot allocate memory for application update	***sistema***

Sem memória disponível. Verifique o hardware, de acordo com o tamanho do aplicativo.

30. unknown OEM key code	***aplicativo***

O aplicativo está utilizando uma placa que tem um código de fabricante não reconhecido pelo destino. Verifique a conexão de E/S no ambiente de trabalho e utilize o atributo 'VIRTUAL' para localizar a placa incorreta. É possível que a biblioteca do seu ambiente de trabalho não corresponda à versão do seu destino.

31. cannot init boolean input board	***aplicativo***

A inicialização de uma placa de entrada booleana falhou. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros das suas placas de entrada booleanas.

32. cannot init analog input board	***aplicativo***

A inicialização de uma placa de entrada analógica falhou. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros das suas placas de entrada analógicas.

33. cannot init message Entrada board	***aplicativo***

A inicialização de uma placa de entrada de mensagem falhou. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros das suas placas de entrada de mensagem.

34. cannot init boolean output board	***aplicativo***

A inicialização de uma placa de saída booleana falhou. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros das suas placas de saída booleanas.

35. cannot init analog output board	***aplicativo***

A inicialização de uma placa de saída analógica falhou. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros das suas placas de saída analógicas.

36. cannot init message output board	***aplicativo***

A inicialização de uma placa de saída de mensagem falhou. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros das suas placas de saída de mensagem.

37. cannot input boolean board	***aplicativo***

Um erro foi detectado durante a atualização de uma placa de entrada booleana. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros de suas placas.

38. cannot input analog board	***aplicativo***

Um erro foi detectado durante a atualização de uma placa de entrada analógica. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros de suas placas.

39. cannot input message board	***aplicativo***

Um erro foi detectado durante a atualização de uma placa de entrada de mensagem. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros de suas placas.

40. cannot output boolean output variable	***aplicativo***

Um erro foi detectado durante a atualização de uma placa de saída booleana. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros de suas placas.

41. cannot output analog output variable	***aplicativo***

Um erro foi detectado durante a atualização de uma placa de saída analógica. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros de suas placas.

42. cannot output message output variable	***aplicativo***

Um erro foi detectado durante a atualização de uma placa de saída booleana. Verifique a conexão de E/S no ambiente de trabalho e os parâmetros de suas placas.

43. cannot operate boolean variable	***aplicativo***

Um erro foi detectado durante a execução da chamada OPERATE de uma variável booleana. Verifique os seus parâmetros OPERATE e a nota de usuário da placa.

44. cannot operate analog variable	***aplicativo***

Um erro foi detectado durante a execução da chamada OPERATE de uma variável analógica. Verifique os seus parâmetros OPERATE e a nota de usuário da placa.

45. cannot operate message variable	***aplicativo***

Um erro foi detectado durante a execução da chamada OPERATE de uma variável de mensagem. Verifique os seus parâmetros OPERATE e a nota de usuário da placa.

46. cannot open board	***aplicativo***

O aplicativo utiliza uma referência de placa desconhecida pelo destino. Verifique a conexão de E/S no ambiente de trabalho. É possível que a biblioteca de seu ambiente de trabalho não corresponda à versão do seu destino.

47. cannot close board	***aplicativo***

O aplicativo utiliza uma referência de placa desconhecida pelo destino. Verifique a conexão de E/S no ambiente de trabalho.

50. cannot overwrite boolean output variable	***programa***

Duas seqüências SFC estão escrevendo a mesma variável de saída booleana no mesmo ciclo destino. Isto deve ser evitado para prevenir um comportamento perigoso dos E/Ss. No caso de tal conflito, a prioridade é dada ao programa mais alto na hierarquia. Se os dois programas de SFC estão localizados no mesmo nível, o resultado é imprevisível.

51. cannot overwrite analog output variable	***programa***

Duas seqüências SFC estão escrevendo a mesma variável de saída analógica no mesmo ciclo destino. Veja o comentário acima.

52. cannot overwrite message output variable	***programa***

Duas seqüências SFC estão escrevendo a mesma variável de saída de mensagem no mesmo ciclo destino. Veja o comentário acima.

61. unknown sistema request code	***programa***

Um programa está utilizando a chamada SISTEMA com um código inválido.

62. sampling period overflow	***programa***

O tempo de ciclo é maior que o especificado no menu.
Em um sistema multitarefa, significa que não há tempo suficiente de CPU para executar um ciclo, mesmo se a 'duração do ciclo atual' é menor que o período especificado.
Num sistema mono-tarefa, isto sempre significa que há muitas operações em um dos ciclos do destino.
Há muitas maneiras possíveis de remover esta mensagem:
- reduzir o número de operações executadas no momento da apresentação da mensagem.
- reduzir o número de marcas e transições válidas, otimizar os processamentos complexos, etc.
- reduzir a carga da CPU das outras tarefas para dar mais tempo ao ISaGRAF.
- reduzir o tráfego de comunicação para dar mais tempo ao ISaGRAF.
- utilizar a modificação dinâmica da duração do ciclo para adaptar a duração do ciclo às diferentes etapas do processo.
- forçar a zero o tempo de ciclo para permitir que o ISaGRAF núcleo funcione o mais rápido possível, sem qualquer verificação de transbordo.

63. user function not implemented	***aplicativo***

Um programa utiliza uma função C desconhecida no destino. É possível que a biblioteca do seu ambiente de trabalho não corresponda à versão do seu destino.

64. integer divided by zero	***programa***

Um programa tenta dividir um inteiro analógico por zero (divisão por zero). Seu aplicativo deve prevenir tal evento que pode ter efeitos imprevisíveis.
Quando isto ocorre, o ISaGRAF apresenta o maior valor analógico (máximo) como resultado.
Quando o operando é negativo, o resultado é invertido.

65. conversion function not implemented	***aplicativo***

Um programa está utilizando uma função de conversão C que é desconhecida no destino. É possível que a biblioteca do seu ambiente de trabalho não corresponda à versão do seu destino.
Quando isto ocorre, o ISaGRAF não converte o valor.

66. function block not implemented	***aplicativo***

Um programa está utilizando um bloco de função que é desconhecido no destino. É possível que a biblioteca do seu ambiente de trabalho não corresponda à versão do seu destino.

67. standard function not implemented	***aplicativo***

Um programa está utilizando um bloco de função que é desconhecido no destino embora tivesse que estar disponível na maioria dos destinos. Entre em contato com o seu fornecedor.

68. real divided by zero	***programa***

Um programa tenta dividir um real analógico por zero (divisão por zero). Seu aplicativo deve prevenir tal evento que pode ter efeitos imprevisíveis.
Quando isto ocorre, o ISaGRAF apresenta o maior valor real analógico (máximo) como resultado.
Quando o operando é negativo, o resultado é invertido.

69. invalid operate parameters	***aplicativo***

Seu aplicativo utiliza uma chamada OPERATE com parâmetros incorretos. Em geral, isto é filtrado pelo compilador. Pode ser um parâmetro de temporização ou uma variável que não é uma entrada ou uma saída.

72. application symbols cannot be modified	***aplicativo***

Ao tentar efetuar uma atualização no aplicativo, o aplicativo modificado não pode ser iniciado porque os símbolos são diferentes. Uma ou mais variáveis ou instâncias de blocos de função podem ter sido adicionadas, removidas ou modificadas em comparação com o aplicativo corrente.

73. cannot update: different set of boolean variables	***aplicativo***

O aplicativo modificado não pode ser iniciado porque algumas variáveis booleanas podem ter sido adicionadas ou removidas em comparação ao aplicativo corrente.

74. cannot update: different set of analog variables	***aplicativo***

O aplicativo modificado não pode ser iniciado porque algumas variáveis analógicas podem ter sido adicionadas ou removidas em comparação ao aplicativo corrente.

75. cannot update: different set of timer variables	***aplicativo***

O aplicativo modificado não pode ser iniciado porque algumas variáveis de temporização podem ter sido adicionadas ou removidas em comparação ao aplicativo corrente.

76. cannot update: different set of message variables	***aplicativo***

O aplicativo modificado não pode ser iniciado porque algumas variáveis de mensagem podem ter sido adicionadas ou removidas em comparação ao aplicativo corrente.

77. cannot update: cannot find new application	***application***

O aplicativo modificado não pode ser encontrado na memória. Pode ser que alguma coisa errada tenha acontecido durante a carga.

D. Glossário

Ação (FC) Símbolo de um diagrama de fluxo. Uma ação representa uma lista de instruções a serem executadas quando o fluxo dinâmico encontra o símbolo de ação.

Ação de pulso Ação SFC: lista de enunciados executada somente uma vez quando a etapa correspondente é ativada.

Ação não armazenada Ação SFC: lista de enunciados executada uma só vez quando a etapa correspondente está ativa.

Ação Lista de declarações ou tarefas executadas quando uma etapa de um programa SFC está ativa.

Ações booleanas Ação SFC: uma variável booleana é nomeada com o sinal de atividade de uma etapa.

Alcance Conjunto de programas que podem utilizar um objeto. O alcance predefinido para o ISaGRAF é: comum, global ou local.

Analógica Tipo de variáveis. São variáveis contínuas, reais ou inteiras.

Atividade de uma etapa Atributo de uma etapa que é marcado por um símbolo SFC. As ações anexadas à etapa são executadas de acordo com sua atividade.

Atributo Classe de variáveis. Os atributos disponíveis são internos, de entrada e de saída.

Barra de energia Barra vertical principal à esquerda ou à direita de uma rede LD.

Biblioteca Conjunto de recursos de hardware ou de software que podem ser utilizados diretamente em qualquer aplicativo ISaGRAF.

Bloco de função Componente gráfico da linguagem FBD que representa uma função elementar padrão das bibliotecas ISaGRAF.

Bobina Componente gráfico de um programa LD utilizado para representar a atribuição de uma variável de saída.

Booleana Tipo de variáveis. Tais variáveis só podem assumir valores verdadeiro ou falso.

Borda Mudança de uma variável booleana. Uma borda de subida significa a mudança de falso para verdadeiro. Uma borda de descida significa a mudança de verdadeiro para falso.

Breakpoint Marca colocada pelo usuário no tempo de depuração, em uma etapa ou transição SFC. O sistema destino para quando um símbolo SFC vai para o breakpoint.

Cadeia Conjunto de caracteres armazenado em uma mensagem variável.

Caixa de ferramenta Pequena janela associada à uma janela de edição que reúne os principais botões para a seleção dos componentes gráficos.

Canal de E/S Ponto de conexão simples de uma placa de E/S. Um canal de E/S pode receber uma variável de E/S.

Célula Área elementar da matriz gráfica para linguagens gráficas tais como SFC, FBD ou LD.

Cíclico Atributo de um programa que sempre é executado.

Ciclo destino Conjunto de operações executadas toda vez que o sistema ISaGRAF destino é ativado. Os ciclos são sincronizados com um período programável.

Código fonte C Arquivo texto que contém o código fonte " C " de uma função ou uma função de conversão.

Código OEM (Placa de E/S) Código hexadecimal de 16 bits definido para cada placa E/S da biblioteca ISaGRAF. O código OEM identifica o fornecedor da placa.

Comentário Texto incluído em um programa não tendo nenhuma atuação na execução do programa.

Comentário (SFC) Texto atrelado a uma etapa ou transição SFC não tendo nenhuma atuação na execução do programa.

Comum Gama de palavras definidas. Tais objetos podem ser utilizados em qualquer programa de qualquer projeto.

Condição (para uma transição) Expressão Booleana atrelada a uma transição SFC. A transição não pode ser apagada quando sua condição for falsa.

Conector (FC) Componente gráfico FC que representa um link de um ponto do fluxograma para uma ação ou teste FC. O símbolo gráfico de um desvio é um círculo pequeno, numerado com a referência do elemento de destino.

Conexão de E/S Definição dos links entre as variáveis do aplicativo e os canais das placas existentes no sistema destino.

Contato Componente gráfico de um programa LD. Representa o estado de uma variável de entrada.

Conversão Filtro atrelado a uma variável analógica de entrada ou de saída. A conversão é automaticamente aplicada toda vez que a variável é de entrada ou de saída.

Decisão (FC) (Também chamado teste) Símbolo de fluxograma atrelado a uma expressão booleana. O fluxo é direcionado ou para o símbolo de saída SIM ou para o símbolo de saída NÃO dependendo do estado da expressão.

Definições C Arquivo texto que contém as definições de constantes e dos tipos "C" requeridos para a programação de uma função C ou uma função de conversão.

Destino Máquina destino ISaGRAF, que suporta o núcleo do software ISaGRAF.

Desviar para uma etapa Componente gráfico SFC que representa um link de uma transição para uma etapa. O símbolo gráfico de um desvio é uma seta, numerada com a referência da etapa de destino.

Diagrama de Blocos Linguagem gráfica: as equações são construídas com blocos elementares padrões da biblioteca ISaGRAF, interligadas no diagrama.

Diagrama de contatos (Esquema de contatos) Linguagem gráfica que combina os contatos e os relés (as bobinas) em uma rede, para a modelagem de equações booleanas.

Diário Arquivo de texto que contém todas as observações sobre as mudanças feitas em um programa. Cada observação é completada com sua data de edição.

Dicionário Conjunto de variáveis interna, de entrada e de saída, e palavras definidas, utilizadas nos programas de um projeto.

E/S bloqueado Variável de entrada ou de saída, desconectada logicamente do dispositivo de E/S correspondente, por um comando " Trava " enviado pelo usuário do depurador.

Endereço de rede Endereço hexadecimal opcional definido livremente para cada variável. Este endereço é utilizado pelo protocolo Modbus para identificar a variável, para a comunicação com um sistema externo.

Entrada Atributo de uma variável. Tais variáveis são ligadas a um dispositivo de entrada.

Erro de execução Erro de um aplicativo detectado, na execução, pelo sistema destino ISaGRAF.

Etapa de fim Última etapa do corpo de uma macro etapa SFC. Uma etapa final não está ligada a qualquer transição seguinte.

Etapa de início Primeira etapa do corpo de uma macro etapa. Uma etapa inicial não está ligada uma transição de nenhuma precedente.

Etapa inicial Etapa especial de um programa SFC que é ativada quando o programa inicia.

Etapa Componente gráfico da linguagem SFC. Uma etapa representa um estado estável do processo. É representada por um quadrado e referenciada por um número. A atividade da etapa é utilizada para controlar a execução das ações definidas nas etapas.

Etiqueta (IL) Identificador colocado no início de uma linha de instrução IL que identifica a instrução e pode ser utilizada como um operando para as operações JMP.

Expressão constante Expressão literal utilizada para descrever um valor constante. Uma expressão constante é dedicada a um tipo.

Expressão Conjunto de operadores e identificadores que representa a evolução de um valor.

FBD Abreviatura de Functional Block Diagram (Diagrama em Blocos Funcionais) .

FC Abreviatura de Flow Chart (Fluxograma).

Fim de seção Grupo de programas cíclicos executado ao término de cada ciclo destino.

Fluxograma seqüencial Linguagem gráfica (GRAFCET): o processo é descrito em um número de etapas ligadas pelas transições. As ações estão atreladas às etapas. As transições estão detalhadas em condições booleanas.

Fluxograma Linguagem gráfica utilizada para projetar um fluxo. O gráfico consiste na ação a ser executada e a decisão que permite a seleção entre vários caminhos no fluxo. A linguagem Fluxograma habilita o projeto de loops a executar em ciclos sucessivos

Função de conversão Função escrita "C" que descreve uma conversão. Tal conversão pode estar atrelada a qualquer variável analógica de entrada ou saída.

Função em C A função escrita em linguagem " C ", chamada dos programas ISaGRAF (escrita com outras linguagens), de um modo síncrono. Funções C são entregues pela CJ International ou desenvolvidas pelo usuário.

Global Gama de variáveis ou palavras definidas. Tais objetos podem ser utilizados em qualquer programa de um projeto.

Hierarquia Arquitetura de um projeto, dividida em vários programas. A árvore hierárquica representa os links entre os programas principal e secundário.

Identificador Palavra única utilizada para representar uma variável ou uma expressão constante na programação.

IL Abreviatura de Instruction List (lista de instruções).

Início da seção Grupo de programas cíclicos executados no início de cada ciclo.

Instrução Operação completa da linguagem ST.

Instrução Operação elementar de um programa IL, introduzida em uma linha de texto.

Inteiro Classe de variáveis analógicas, armazenada no formato de valor absoluto, inteiro de 32 bits.

Interno Atributo de uma variável que não está ligada a um dispositivo de entrada ou de saída.

LD Abreviatura de Ladder Diagram (esquema de contatos).

Linguagem Linguagem de alto nível utilizada para descrever as operações de computador tais como

C	funções C e funções de conversão.
Lista de instruções	Linguagem de baixo nível, introduzida como uma lista seqüencial de operações elementares.
Local	Gama de variáveis ou palavras definidas. Tais objetos só podem ser utilizados em um programa de um projeto.
Macro etapa	Componente gráfico SFC. Uma macro etapa é um grupo único de etapas e transições representado como um único símbolo e descrita separadamente dentro do mesmo programa SFC.
Marca (SFC)	Marcador gráfico utilizado para indicar a atividade de uma etapa SFC durante a execução.
Matriz	Divisão lógica da área de edição em células retangulares, durante a edição de um programa com linguagem gráfica.
Mensagem	Tipo de variável. Tais variáveis contêm a cadeia de caracteres de comprimento variável
Modbus	Protocolo do tipo mestre-escravo. Um sistema destino ISaGRAF pode ser um escravo Modbus ligado a um sistema externo (como sistemas supervisores).
Modificador (IL)	Caractere complementar do nome de uma operação IL e que modifica o significado da instrução.
Modo ciclo a ciclo	Modo de execução: neste modo, os ciclos são executados um por um, de acordo com as ordens dadas pelo usuário do depurador.
Modo em Tempo real	Modo de execução normal: os ciclos são automaticamente acionados de acordo com os tempos de ciclo programados.
Nível 1 do SFC	Descrição geral de um programa SFC. O nível 1 agrupa o desenho do gráfico, os números de referência e os comentários atrelados às etapas e às transições.
Nível 2 do SFC	Descrição detalhada de um programa SFC. É a programação das ações das etapas e das condições associadas às transições.
Número de referência (SFC)	Número decimal (de 1 até 65.535) que identifica uma etapa SFC ou transição em um programa SFC.
Observação técnica	Guia de utilização para um elemento da biblioteca ISaGRAF (função C ou bloco de função, função de conversão ou placa E/S). A observação técnica é escrita pelo projetista do elemento.
Operação (IL)	Instrução básica do idioma IL. Uma operação geralmente está associada a um operando em uma linha de instrução.
Operação em	Operação de um programa IL, modificado pelo caractere modificador "(", que não será

espera (IL)	executada até a aparição da próxima instrução “)”.
Operando (IL)	Variável ou expressão constante tratada por uma instrução da linguagem IL.
Palavra chave	Palavra reservada da linguagem.
Palavras definidas	Identificador único utilizado para substituir qualquer expressão em um programa.
Parâmetro (Função em C)	Argumento de entrada ou de saída de uma função “C” . Um parâmetro é definido por um tipo e um nome de parâmetro formal.
Parâmetro (Placa de E/S)	Parâmetro constante ou programável de uma placa de E/S. Um parâmetro programável será informado durante a conexão de E/S para um aplicativo.
Parâmetro OEM (Placa de E/S)	Parâmetro de uma placa E/S, definido pelo projetista da placa. Pode ser um valor constante ou um parâmetro variável inserido pelo usuário durante a conexão de E/S, para cada aplicativo.
Placa de E/S	Recurso de hardware. Uma placa de E/S é caracterizada por um tipo e uma direção (entrada ou saída). Os parâmetros de uma placa de E/S são descritos na biblioteca ISaGRAF.
Placa real	Placa E/S fisicamente conectada a um dispositivo E/S da máquina destino.
Placa virtual	Placa E/S que não está fisicamente conectada a um dispositivo E/S da máquina destino e cujo funcionamento é simulado pelos comandos do depurador.
Programa de nível mais alto	Programa situado no topo da árvore hierárquica (nível mais alto). Este programa é ativado pelo sistema.
Programa principal SFC	Programa SFC que controla outro programa SFC, chamado seus secundários (children).
Programa principal	Programa escrito em qualquer linguagem que controla (chama) um outro programa (não SFC) chamada de subprograma.
Programa secundário SFC	Programa SFC controlado por outro programa SFC, denominado seu principal (father).
Programa	Unidade de programação. Um programa é descrito com uma linguagem e tem um lugar definido dentro da árvore hierárquica que modela a arquitetura do projeto.
Projeto	Área de programação que reúne toda a informação (programas, variáveis, código destino ...) para um aplicativo ISaGRAF
Real	Classe de variáveis analógicas armazenadas em um formato IEEE de ponto flutuante de

	32 bits (precisão simples) e podem aceitar os valores não inteiros.
Referências cruzadas	Informação calculada pelo Ambiente Trabalho ISaGRAF sobre o dicionário de variáveis e onde essas variáveis são usadas em um projeto.
Registro (IL)	Resultado atual de uma seqüência de programação em linguagem IL.
Resultado corrente (IL)	Resultado de uma instrução em um programa IL. O resultado atual pode ser modificado por uma instrução ou utilizado para definir uma variável.
Saída	Atributo de variável. Tais variáveis são ligadas a um dispositivo de saída (acionadores).
Seção seqüencial	Grupo de programas de um projeto. A execução destes programas segue regras dinâmicas da linguagem SFC.
Seção	Grupo de programas executado com as mesmas regras.
Separador	Caractere especial (ou grupo de caracteres) utilizado para separar os identificadores em uma linguagem literal. Um separador pode representar uma operação.
SFC	Abreviatura de Sequential Function Chart (GRAFCET).
Situação inicial	Conjunto de etapas iniciais de um programa SFC que representa o contexto do programa quando este é iniciado.
ST	Abreviatura de Structured Text (Texto estruturado).
Subprograma	Programa escrito em linguagem não SFC, ativado por um outro programa chamado programa principal (father), e executado de modo síncrono.
Tabela de conversão	Conjunto de pontos que define uma conversão linear (pelo segmento). Tal conversão pode ser aplicada a qualquer variável analógica de entrada ou saída.
Tempo de ciclo	Duração do ciclo de execução de destino.
Temporização	Tipo de variáveis. Tais variáveis contêm um valor de tempo e podem ser automaticamente atualizados pelo sistema ISaGRAF durante o tempo de execução.
Teste (FC)	(Também chamada decisão) Componente de um fluxograma atrelado a uma expressão booleana. O fluxograma está dirigido ou para a saída SIM (YES) ou para a saída NÃO (NO) dependo do estado da expressão.
Texto estruturado	(Texto estruturado) Linguagem literal estruturada de alto nível, combinando os enunciados de designação, as estruturas de controle do tipo If/Then/Else e as chamadas de funções.
Tipo	Classe de variáveis com o mesmo formato. Os tipos básicos predefinidos são: booleana, analógica, temporização e mensagem.

Transição	Componente gráfico da linguagem SFC. Uma transição representa a passagem de um grupo de etapas para um outro grupo de etapas e está condicionado a uma expressão booleana.
Transpondo uma transição	Tempo de execução: todos os símbolos que existem nas etapas precedentes são removidas. Um símbolo é criado em cada uma das etapas seguintes.
Validade de uma transição	Atributo de uma transição. Uma transição é validada quando todas as etapas anteriores estão ativas.
Valor de retorno de um subprograma	Valor retornado por um subprograma ao término de sua execução. O valor de retorno é utilizado nas equações do programa solicitante.
Variável de E/S	Variável conectada a um dispositivo de entrada ou de saída. Uma variável de E/S deve ser conectada em um canal de uma placa de E/S.
Variável	Representação de um dado elementar tratado pelos programas de um projeto.

E. Índice Geral

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z