



ELIPSE SCADA

---

HMI/SCADA SOFTWARE

# TUTORIAL



# Índice

<b>1.</b>	<b>INTRODUÇÃO</b>	<b>7</b>
1.1.	VERSÕES DO ELIPSE SCADA	7
1.1.1.	View	7
1.1.2.	MMI (Man Machine Interface)	8
1.1.3.	Pro (Professional)	8
1.1.4.	Power	9
1.2.	MÓDULOS DE OPERAÇÃO	9
1.3.	PLUG-INS	10
1.4.	OUTRAS INFORMAÇÕES	10
<b>2.</b>	<b>APRESENTAÇÃO</b>	<b>11</b>
<b>3.</b>	<b>INICIANDO O SCADA</b>	<b>15</b>
3.1.	CRIANDO A SUA APLICAÇÃO	16
3.2.	ORGANIZER	16
3.2.1.	<i>Ferramentas do Organizer</i>	17
3.3.	PROPRIEDADES DE UMA APLICAÇÃO	20
3.4.	TECLAS DE ATALHO	22
3.5.	OPÇÕES DE LINHAS DE COMANDO	23
<b>4.</b>	<b>TAGS</b>	<b>27</b>
4.1.	TIPOS DE TAGS	28
4.2.	CRIANDO TAGS	29
4.2.1.	<i>Regras para os nomes dos Tags</i>	29
4.3.	TAG PLC	29
4.3.1.	<i>Propriedades do tag PLC</i>	34
4.4.	TAG BLOCO	37
4.4.1.	<i>Propriedades do Tag Bloco</i>	37
4.5.	ELEMENTO DE BLOCO	38
4.6.	TAG BIT	40
4.7.	TAG RAM	42
4.8.	TAG MATRIZ	42
4.8.1.	<i>Propriedades do Tag Matriz</i>	43
4.9.	TAG DEMO	44
4.9.1.	<i>Propriedades do Tag Demo</i>	44
4.10.	TAG CRONO	46
4.11.	TAG DDE	47
4.11.1.	<i>Propriedades do Tag DDE</i>	47
4.12.	TAG EXPRESSÃO	48
4.13.	DICAS SOBRE TAGS	55
4.14.	PÁGINA DE ALARMES	57
4.15.	ALARMES E GRUPOS DE ALARMES	59
<b>5.</b>	<b>CRIAÇÃO DE TELAS</b>	<b>63</b>
5.1.	PROPRIEDADES GERAIS DE TELAS	63
5.2.	PROPRIEDADES DO ESTILO DA TELA	65
<b>6.</b>	<b>OBJETOS DE TELA</b>	<b>71</b>
6.1.	EDIÇÃO DOS OBJETOS DE TELA	72
6.2.	PROPRIEDADES DOS OBJETOS DE TELA	74
6.2.1.	<i>Página Tamanho e Posição</i>	74
6.2.2.	<i>Página de Moldura</i>	76
6.2.3.	<i>Página de Tags</i>	77
6.3.	INSERÇÃO DE OBJETOS E EXECUÇÃO	78
6.3.1.	<i>Utilização de Imagens</i>	78
6.3.2.	<i>Fazendo animações</i>	79
<b>7.</b>	<b>SCRIPTS</b>	<b>91</b>
7.1.	CONSIDERAÇÕES GERAIS	91

7.2.	APPBROWSER E REFERÊNCIA CRUZADA .....	92
7.3.	OPERADORES E CONSTANTES .....	94
7.4.	CONTROLE DE FLUXO .....	96
7.4.1.	Comando <i>If...Else...Elseif...Endif</i> .....	96
7.4.2.	Comando <i>For...Next</i> .....	97
7.4.3.	Comando <i>While...Wend</i> .....	97
7.4.4.	Comando <i>Repeat</i> .....	97
7.5.	FUNÇÕES ESPECIAIS .....	97
7.6.	DICAS DE OTIMIZAÇÃO .....	98
<b>8.</b>	<b>RECEITAS .....</b>	<b>107</b>
8.1.	PROPRIEDADES GERAIS DA RECEITA .....	107
8.2.	EDITANDO RECEITAS .....	109
8.3.	DICAS DE OTIMIZAÇÃO .....	110
<b>9.</b>	<b>HISTÓRICOS .....</b>	<b>115</b>
9.1.	TIPOS DE HISTÓRICOS .....	115
9.2.	ANÁLISE HISTÓRICA .....	117
9.2.1.	<i>Configurando a Análise Histórica</i> .....	119
9.3.	DICAS E OTIMIZAÇÃO .....	121
<b>10.</b>	<b>RELATÓRIOS .....</b>	<b>129</b>
10.1.	PROCEDIMENTOS COM RELATÓRIOS .....	129
<b>11.</b>	<b>USUÁRIOS E SENHAS .....</b>	<b>135</b>
<b>12.</b>	<b>BANCO DE DADOS .....</b>	<b>139</b>
<b>13.</b>	<b>DICAS E OTIMIZAÇÃO .....</b>	<b>143</b>

# Convenções

Estas são convenções utilizadas neste manual:

EXEMPLO	DESCRIÇÃO
SILO6.BMP	Nomes de arquivos e outros termos no nível do sistema operacional são indicados com o tipo de letra <b>Tahoma</b> , em maiúsculas.
Geral	Nomes de campos e opções que devem ser procurados na tela, em menus ou nas abas dos objetos são indicados com tipo de letra <b>Tahoma</b> .
“Agitação”	Caracteres entre aspas devem ser digitados no lugar mencionado, sem a presença das aspas.
Tela1.Show()	Partes de programas ( <i>scripts</i> ) são indicadas com o tipo de letra <i>Courier</i> . Eles deverão ser digitados nos lugares reservados e depois compilados para a verificação de erros.
Tank01.High	Caracteres em <b>negrito</b> indicam nomes de objetos do Elipse SCADA ou suas propriedades.
<nome do arquivo>	Expressões entre os sinais <> devem ser substituídas pelo nome do objeto em questão.
[Ctrl+Enter]	Expressões entre colchetes indicam nomes de teclas. Quando estiverem acompanhadas de um sinal +, você deve pressionar a segunda tecla enquanto pressiona a primeira.



Bem-vindo ao Elipse SCADA! A Elipse Software sente-se orgulhosa em apresentar esta poderosa ferramenta para o desenvolvimento de sistemas de supervisão e controle de processos.

O Elipse SCADA alia alto desempenho e grande versatilidade representados em seus diversos recursos que facilitam e agilizam a tarefa de desenvolvimento de sua aplicação. Totalmente configurável pelo usuário, permite a monitoração de variáveis em tempo real, através de gráficos e objetos que estão relacionados com as variáveis físicas de campo. Também é possível fazer acionamentos e enviar ou receber informações para equipamentos de aquisição de dados. Além disso, através de sua exclusiva linguagem de programação, o Elipse Basic, é possível automatizar diversas tarefas a fim de atender as necessidades específicas de sua empresa.

Agradecemos a sua preferência por nossos produtos e desejamos sucesso com sua nova ferramenta de trabalho!

*Equipe Elipse Software*

## 1.1. Versões do Elipse Scada

---

O Elipse SCADA está disponível em quatro versões, atendendo as demandas de personalização de nossos clientes. Estas versões se diferenciam na sua funcionalidade, cada uma acrescentando recursos em relação à versão anterior. A seguir, podemos observar as características de cada versão:

### 1.1.1. View

A versão **View** é indicada para aplicações simples, como por exemplo, uma interface com o operador para monitoração e acionamentos. As informações recebidas pelo View estão disponíveis também para outras aplicações que possam trabalhar com DDE (Dynamic Data Exchange). Neste módulo estão disponíveis:

- Funções de monitoramento e controle;
- Comunicação com PLCs e outros equipamentos via drivers DLL, inclusive em blocos;

- Objetos de tela para a produção de interfaces, como por exemplo, botões, medidores (*gauges*), caixas de texto, gráficos de barra e tendências, imagens, animações, alarmes e outros;
- Importação de imagens de editores gráficos, como por exemplo, Corel Draw! e Microsoft Paint;
- Alarmes;
- Controle de acesso através de lista de usuários (autenticação);
- Servidor e cliente DDE;
- Programação e automação de processos através de sua exclusiva linguagem de programação baseada em *scripts*, o Elipse Basic;
- Servidor para aplicações remotas.

### 1.1.2. MMI (Man Machine Interface)

Esta versão é indicada para aplicações de médio porte, onde é necessário o armazenamento de dados, tratamento de informações e criação de relatórios complexos.

Nesta versão, estão disponíveis além das características da versão View, as seguintes:

- Históricos;
- Receitas;
- Relatórios;
- CEP (Controle Estatístico de Processos);
- Novos objetos de tela: Browser e Alarmes tipo Histórico;
- *Log* de alarmes em disco.

### 1.1.3. Pro (Professional)

Esta versão é indicada para aplicações de qualquer porte, que envolvam comunicação em rede, seja local ou remota ou ainda que necessitem a troca de informações com bancos de dados. A versão Professional possui, além de todas as características da versão MMI, as seguintes funções:

- Suporte a ODBC (Open Database Connectivity);
- Suporte a DAO (Data Access Objects);
- Age como cliente de aplicações remotas.

### 1.1.4. Power

Versão especialmente desenvolvida para supervisão de subestações e sistemas elétricos. Permite conexão com IEDs (*Intelligent Electronic Device*) e RTU (*Remote Terminal Units*) através de qualquer protocolo de comunicação, inclusive IEC 870-5/DNP 3.0. Utiliza base de tempo local, permitindo seqüenciamento de eventos (SOE) com precisão de 1 ms e oscilografia, transferência e visualização de formas de onda, tanto em estações locais como em sistemas telesupervisionados.

## 1.2. Módulos de Operação

---

O Elipse SCADA possui três módulos para sua operação: **Configurador**, **Runtime** e **Master**. O módulo ativo é definido a partir de um dispositivo de proteção (*hardkey*) que é acoplado ao computador. Enquanto que os módulos Configurador e Master foram especialmente desenvolvidos para a criação e o desenvolvimento de aplicativos, o módulo Runtime permite apenas a execução destes. Neste módulo, não é possível qualquer alteração no aplicativo por parte do usuário.

Na ausência do hardkey, o software pode ainda ser executado em modo **Demonstração**. Como não necessita do hardkey, o modo **Demo** pode ser utilizado para a avaliação do software. Ele possui todos os recursos existentes no módulo Configurador, com exceção de que trabalha com um máximo de 20 *tags* (variáveis de processo) e permite a comunicação com equipamentos de aquisição de dados por até 10 minutos. Neste modo, o software pode ser livremente reproduzido e distribuído.

Os módulos Runtime e Master estão também disponíveis em versões **Lite** que possuem as mesmas características, porém são limitadas em número de *tags* (variáveis): Lite 75 com 75 tags e Lite 300 com 300 tags.

## 1.3. Plug-Ins

---

Plug-ins são ferramentas adicionais que permitem a expansão dos recursos do Elipse SCADA, acrescentando funcionalidades no software. Eles podem ser adquiridos separadamente e trabalham em conjunto com qualquer versão do software.

Atualmente, estão disponíveis os seguintes plug-ins:

Plug-Ins	
OPÇÕES	DESCRIÇÃO
	Permite a monitoração de sistemas através de recursos de captura, registro e transmissão digital de imagens em tempo real. Suporta diversos padrões (inclusive MPEG), possibilitando a visualização em janelas com tamanho e qualidade programáveis pelo usuário. Permite a criação de um banco de imagens com busca por período ou evento e transmissão de imagens em tempo real para estações remotas via TCP/IP ou linha discada.
	Sistema para supervisão de processos através da Internet. Utilizando qualquer navegador (Internet Explorer, Netscape e outros) é possível conectar-se a uma estação de supervisão remota, recebendo dados em tempo real. Com este recurso é possível visualizar processos de qualquer parte do mundo.

## 1.4. Outras Informações

---

Para saber mais sobre nossos produtos, acesse nossa página <http://www.elipse.com.br> ou entre em contato com nosso departamento de vendas na cidade mais próxima de você:

- **São Paulo, SP:** (11) 3061-2828
- **Porto Alegre, RS:** (51) 3346-4699
- **Curitiba, PR:** (41) 342-0120
- **EUA:** +1 (252) 995-6885

O conteúdo desse manual visa o apoio ao módulo de treinamento para a execução e programação do Elipse SCADA. Com este manual, você poderá acompanhar o conteúdo do curso. Durante as aulas, fique à vontade para praticar o que você aprendeu e resolver suas dúvidas com o professor.

No treinamento, é apresentado um estudo de caso que simula uma aplicação real: um sistema de supervisão e controle. O instrutor desenvolve a aplicação junto com os alunos passo-a-passo, facilitando o entendimento e aprendizado da ferramenta.

O sistema em questão apresenta um sinótico de uma fábrica de balas, exemplificando vários aspectos e recursos disponíveis no Elipse SCADA.



Figura 1: Tela de abertura

Para a produção, são necessários 4 produtos básicos: água, xarope, glucose e açúcar, cujas quantidades serão controladas a cada novo tipo de bala a ser produzida através da utilização de receitas pré-definidas e programadas.

Após a pesagem individual dos produtos, estes são homogeneizados no misturador que por sua vez transfere a mistura para um tanque de estocagem. Esta transferência entre tanques pode ser automática ou controlada pelo acionamento de uma válvula.

A partir do tanque de estocagem, a mistura é transferida para os cozinhadores por bombeamento, também controlado pelo aplicativo. O operador do sistema pode, nesta mesma tela, visualizar as temperaturas de cada tanque, controlar a frequência dos motores e abrir ou fechar as válvulas que levam a mistura para os cozinhadores.

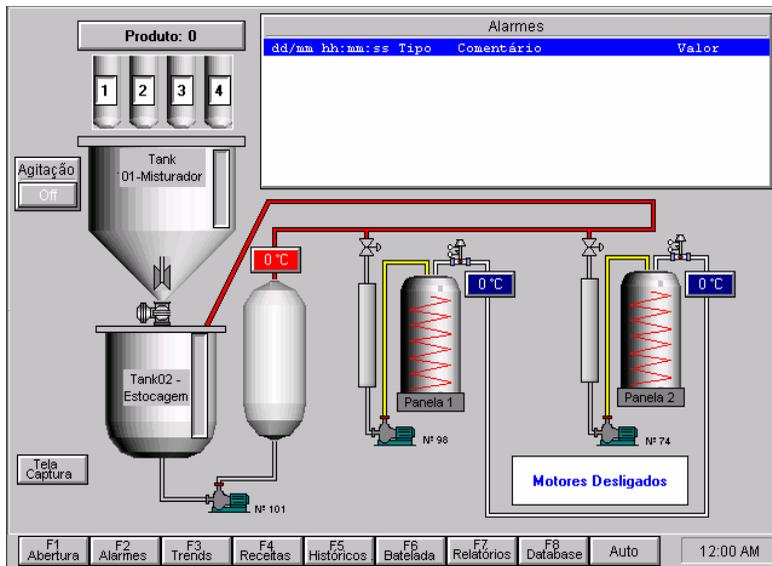


Figura 2: Tela de Dosagem

O sistema também mostrará condições de alarme no caso de algum parâmetro ultrapassar os limites estabelecidos (como por exemplo, um aumento excessivo de temperatura), além de criar gráficos de tendência das temperaturas, geração de base de dados de operação e respectivos relatórios.

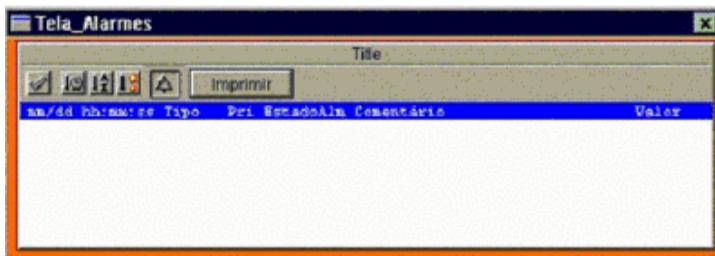


Figura 3: Tela de utilização dos alarmes históricos

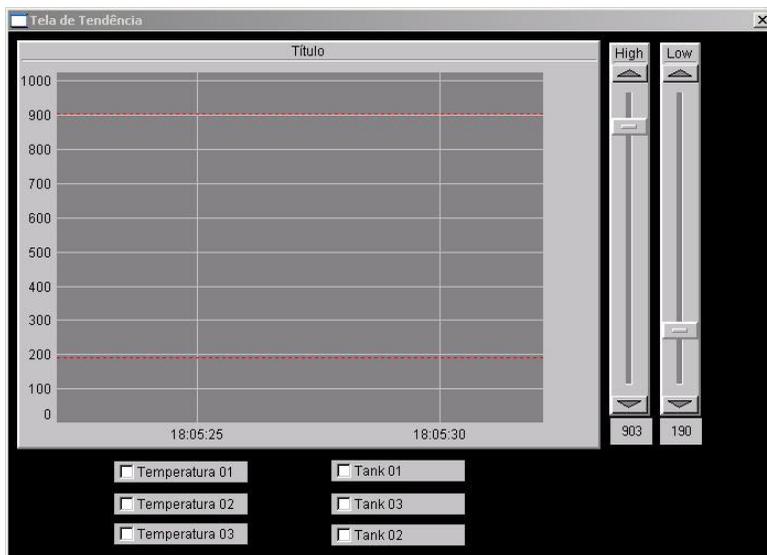


Figura 4: Tela de Tendência

Através da tela de receitas, podem ser criados novos produtos e editados aqueles já existentes.

Finalmente, um procedimento de consulta dos processos de batelada, que permite consulta, visualização e impressão dos dados de histórico.



Figura 5: Tela de Batelada



Figura 6: Tela de Impressão

Após proceder com a instalação do software, você terá em sua máquina um grupo de programas chamado Elipse SCADA com os ícones para chamar o sistema.

Para iniciar o Elipse SCADA, faça isso:

- ➔ Clique no botão Iniciar (*Start*) na barra de tarefas do Windows.
- ➔ Selecione Programas (*Programs*), Elipse SCADA e Elipse SCADA novamente.
- ➔ Você terá uma tela parecida com a figura abaixo.

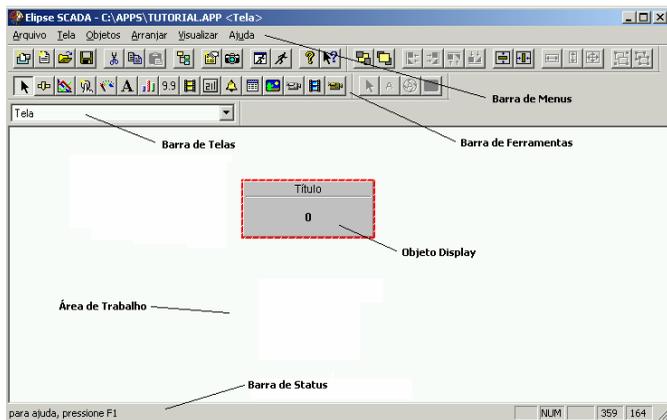


Figura 7: Tela de inicialização do SCADA

Na figura, podemos ver alguns elementos importantes da interface do Elipse SCADA:

- **Barra de Ferramentas:** apresenta botões para fácil acesso às funções do sistema.
- **Barra de Status:** mostra as mensagens do sistema.
- **Área de Trabalho:** área para desenvolvimento da aplicação.

- **Barra de Menus:** para escolha das funções do sistema.
- **Barra de Telas:** para a seleção das tela que se quer trabalhar.
- **Objeto Display:** exemplo de objetos de tela do Elipse SCADA.

### 3.1. Criando a sua aplicação

---

A criação de uma **aplicação** é o ponto de partida para montagem de um sistema utilizando o Elipse SCADA. Em uma aplicação, o usuário reúne todos os elementos necessários para execução das tarefas desejadas. As informações referentes a esta aplicação ficam armazenadas em um arquivo de extensão APP.

Para criar uma nova aplicação:

- ➔ Escolha no menu Arquivo a opção Nova Aplicação.
- ➔ No quadro Salvar Aplicação Nova! escolha um nome e o lugar onde a aplicação será salva.

Além dos arquivos de extensão APP, existem outros gerados e utilizados pelo Elipse SCADA:

**Extensões disponíveis**

EXTENSÃO	DESCRIÇÃO
.APX	Arquivo de senhas
.BAK	Backup da aplicação
.DAT	Arquivo de históricos
.HDR	Cabeçalhos de arquivos de históricos por batelada
.RCP	Arquivo de receitas
.DLL	Drivers de comunicação
.BMP, .JPG, .GIF	Arquivos de imagens

### 3.2. Organizer

---

A fim de permitir uma visão simples e organizada de toda a aplicação, o Elipse SCADA oferece uma poderosa ferramenta de programação chamada Organizer.

A partir do Organizer, você pode desenvolver toda a aplicação simplesmente navegando através de sua estrutura. Essa estrutura pode ser comparada a uma árvore de diretórios. Desta forma, a estrutura da aplicação começa no canto superior esquerdo com a raiz da aplicação. Todos os objetos da aplicação descem a partir da raiz agrupados de acordo com seu tipo: Tags, Telas, Alarmes, Receitas, Históricos, Relatórios e assim por diante.

Selecionando-se qualquer um de seus ramos, as propriedades do objeto selecionado serão mostradas no lado direito da janela onde poderão ser editadas. Por exemplo, se você selecionar Tags na árvore do Organizer, poderão ser criados novos tags e suas

propriedades poderão ser editadas selecionando-se a página desejada a partir das guias no topo da janela.

Você pode chamar o Organizer de diversas maneiras:

- ➔ ou pressionando o botão da barra de ferramentas;
- ➔ selecionando a opção Organizer do menu Arquivo ou
- ➔ apertando as teclas [Alt+O].

Note que você só pode chamar o Organizer quando houver uma aplicação aberta.



Figura 8: Tela do Organizer com as propriedades da aplicação

### 3.2.1. Ferramentas do Organizer

O Organizer possui um conjunto de ferramentas que permitem realizar determinadas tarefas rapidamente, sem a necessidade da utilização dos menus. Também existem botões que inserem comandos do Elipse Basic, facilitando a tarefa de programação de scripts.

Estas ferramentas estão dispostas em uma barra que está localizada na parte inferior da janela do Organizer.

Cada botão desta barra é descrito a seguir.

### Ferramentas do Organizer

ICONE	COMANDO	AÇÃO
	Deletar	Apaga um ou mais itens selecionados no Organizer.
	Duplicar	Duplica o item selecionado na árvore do Organizer.
	AppBrowser	Chama o AppBrowser.
	Referência Cruzada	Chama a Referência Cruzada.
	Compilar	Compila o script que está sendo editado.
	Compilar tudo	Compila todos os scripts que não estão compilados.
	Recompilar tudo	Recompila todos os scripts da aplicação, possibilitando ao usuário acessar cada script com um duplo clique. É gerada uma lista dos scripts compilados, mostrando em vermelho os que estão com erro.
	If	Inserir o comando IF no script selecionado, no ponto onde está o cursor.
	ElseIf	Inserir o comando ELSEIF no script selecionado, no ponto onde está o cursor.
	Else	Inserir o comando ELSE no script selecionado, no ponto onde está o cursor.
	For...Next	Inserir o comando FOR...NEXT no script selecionado, no ponto onde está o cursor.
	While...Wend	Inserir o comando WHILE...WEND (fim de While) no script selecionado, no ponto onde está o cursor.
	Repeat...Until	Inserir o comando REPEAT...UNTIL no script selecionado, no ponto onde está o cursor.

### AppBrowser

O AppBrowser é uma importante ferramenta do Organizer. Ele é composto de uma janela que apresenta a árvore da aplicação com seus objetos. Clicando em qualquer objeto, pode-se visualizar as funções e atributos relacionados a este objeto. Quando estamos escrevendo um script, um botão *Cópia no Script* --> fica disponível nesta janela, permitindo a cópia do atributo ou função em questão para as linhas de programação, facilitando essa tarefa.

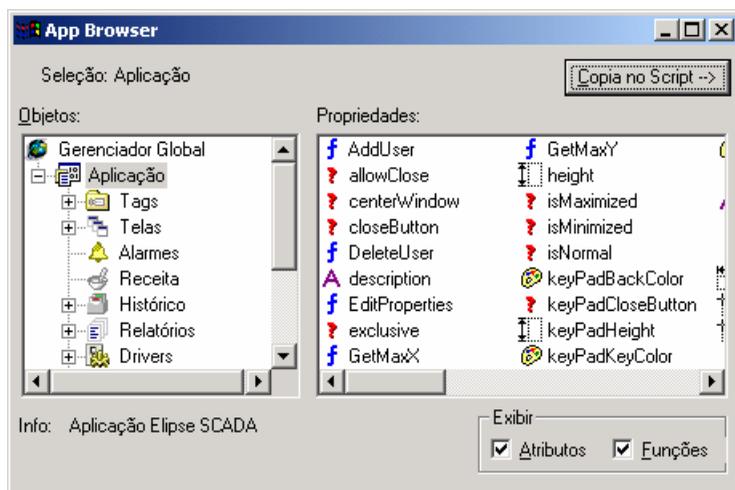


Figura 9: Ferramenta AppBrowser.

### Referência Cruzada

A ferramenta de Referência Cruzada permite visualizar em que locais os objetos indicados são referidos, facilitando a tarefa de configuração e depuração de aplicações.



Figura 10: Referência Cruzada

### 3.3. Propriedades de uma aplicação

Ao selecionar o item **Aplicação** na árvore do Organizer, suas propriedades serão mostradas do lado direito (ver figura 9). Aqui são configurados parâmetros genéricos sobre a aplicação, assim como seu comportamento em relação aos outros programas e ao próprio sistema operacional.

**Propriedades da aba Aplicação**

OPÇÃO	DESCRIÇÃO
Descrição	Define o nome da aplicação (que aparecerá na barra de título) caso a opção Barra de Título esteja habilitada.
Barra de título	Habilita a barra de título na janela da aplicação.

**Propriedades da guia Aplicação (quadro Estilos)**

OPÇÃO	DESCRIÇÃO
Desabilita troca de aplicação	Desabilita a troca entre programas, ou seja, desabilita o atalho [Alt+Tab] do Windows.
Desabilita proteção de tela	Desabilita qualquer protetor de tela ( <i>screen saver</i> ) enquanto o Elipse SCADA estiver sendo executado.
Botão de Fechar	Habilita o botão de Fechar e o Menu de Sistema na janela da aplicação.
Botão de Minimizar	Habilita o botão de Minimizar na janela da aplicação.
Botão de Maximizar	Habilita o botão de Maximizar na janela da aplicação.
Permite Fechar	Desligado, faz com que a execução termine apenas quando for chamada a função <code>StopRunning()</code> . Ligado, permite que a aplicação (e o Elipse SCADA) seja terminado via outros meios, como um clique no botão Fechar, desligar do Windows, etc.

**Propriedades da guia Aplicação (Botões)**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
Estatísticas...	Abre uma janela que mostra informações estatísticas da aplicação, como: tempo total de edição da aplicação, número de itens na aplicação, número total de tags, número de revisões e versão do Elipse SCADA em que foi gerada a aplicação.
Procura Item...	Abre uma janela que permite encontrar um item (objeto, propriedade) em qualquer lugar da aplicação e apresentá-lo para edição.
Importar...	Abre uma janela que permite escolher uma aplicação para a importação. Após a escolha da aplicação origem, uma nova janela é aberta com a árvore das duas aplicações, de modo que o usuário pode arrastar os objetos da aplicação origem para a aplicação destino. OBS: a aplicação-origem não é modificada.
Proteção...	Abre uma janela para proteção da aplicação. Existem duas proteções: para <b>configuração</b> (para editar e fazer qualquer tipo de modificação) e para <b>execução</b> . No caso da utilização de senha para a configuração, o usuário final não poderá alterar a aplicação, a não ser que conheça a senha utilizada. O mesmo vale para a execução, sendo que só pode haver esta senha, se houver uma para a configuração.

**Propriedades da guia Janela**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
Começa Maximizado /Minimizado /Normal	Define as configurações de tamanho iniciais da aplicação.

**Propriedades da guia Touch Screen**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
Habilitar "Key pad"	Habilita um teclado em tela (acessado por mouse ou touch screen).
Esconder mouse	Desaparece com o cursor (ponteiro) do mouse.
Usar botões grandes	Habilita o uso de botões grandes nos objetos de tela e no <i>Key Pad</i> .

### 3.4. Teclas de atalho

Algumas teclas de atalho estão disponíveis para facilitar e agilizar a utilização do Elipse SCADA.

#### Opções gerais

OPÇÃO	DESCRIÇÃO
Ctrl + O	Abrir aplicação
Ctrl + Shift + V	Informações “Sobre o Elipse SCADA”
F1	Chama a ajuda
Shift + F1	Chama a ajuda de contexto

#### Editando uma aplicação

OPÇÃO	DESCRIÇÃO
Ctrl + S	Salvar aplicação
F10	Rodar (executar) aplicação
Alt + O	Chama o Organizer
Ctrl + N	Nova tela
F8	Monitorar tela
Ctrl + Alt + Shift + I	Conta o número de itens da aplicação
Ctrl+Shift+F10	Edita as fontes dos scripts.

#### Editando telas

OPÇÃO	DESCRIÇÃO
Ctrl + F4	Fechar tela
Esc	Deseleccionar objeto
Ctrl + A	Selecionar todos objetos
Del	Apagar objeto
Ctrl + X	Recortar objeto
Ctrl + C	Copiar objeto
Ctrl + V	Colar objeto
Shift + Del	Recortar objeto
Ctrl + Ins	Copiar objeto
Shift + Ins	Colar objeto

#### Editando formulários (Forms)

OPÇÃO	DESCRIÇÃO
Ctrl + F4	Fecha o editor de relatórios
Esc	Deseleccionar objeto
Ctrl + A	Selecionar todos objetos

### 3.5. Opções de Linhas de Comando

---

É possível chamar o Elipse SCADA diretamente da linha de comando. O executável ELIPSE32.EXE possui a seguinte sintaxe:

ELIPSE32.EXE [-DEMO] [-SETUP] [-EDIT] [<NomeApp>]

Onde:

- DEMO (Opcional) Força o Elipse SCADA a rodar em modo de demonstração, sem verificar os mecanismos de proteção (*hardkey*). Esta opção reescreve o arquivo .INI configurando a seção [Protection]Type.
- SETUP (Opcional) Força o Elipse SCADA a rodar o programa de *Setup*, que permite a você configurar as opções no arquivo de preferências (.INI).
- EDIT (Opcional) Força o Elipse SCADA a rodar no modo Configurador. Se o nome de uma aplicação for informado na linha de comando, esta aplicação será aberta para configuração.
- NomeApp (Opcional) O nome da aplicação que irá rodar automaticamente ou será aberta para configuração (quando o -EDIT é especificado).







A supervisão de um processo com o Elipse SCADA ocorre através da leitura de variáveis de processos no campo. Os valores dessas variáveis são associados a objetos do sistema chamados **Tags**.

Para cada objeto inserido na tela, devemos associar pelo menos um **tag** ou **atributo**. Os **tags** são todas as variáveis (numéricas ou alfanuméricas) envolvidas num aplicativo. Os **atributos** são dados fornecidos pelo Elipse SCADA sobre parâmetros de sistema e componentes da aplicação. Como exemplo, podemos considerar um tag a temperatura de um forno. Um de seus atributos poderia ser o nível de alarme a partir do qual deva ser acionada uma sirene.

O valor do tag ou do atributo associado poderá por exemplo, ser mostrado pelos objetos de animação em uma tela, ser utilizado em cálculos em um script, ser modificado através de ações do operador e entre outras possibilidades.

Ao criar tags, o usuário poderá organizá-los livremente em **grupos**, de forma a facilitar a procura e identificação durante o processo de configuração. Para a criação de um grupos, basta selecionar o item **Tags** no Organizer e clicar em **Novo Grupo**.

Você pode criar grupos dentro de outros grupos, sem restrições. Para modificar a hierarquia dos grupos e mudá-los de posição (por exemplo, incluir um grupo em outro grupo) basta arrastar o grupo em questão para o lugar desejado.

Os exemplos deste tutorial informam procedimentos para a criação de tags. Caso você possua um equipamento e deseje realizar comunicação, dê preferência a variáveis tipo PLC ou Bloco; caso contrário, escolha tags do tipo Demo, que permitem a simulação de valores na ausência de dados reais.

## 4.1. Tipos de Tags

---

Os tags podem ter vários tipos, de acordo com o que se deseja armazenar e como se quer utilizá-los.

**Tipos de tags**

TIPOS	DESCRIÇÃO
PLC	É utilizado para trocar informações com os equipamentos de aquisição de dados (escrita e leitura) através dos drivers de comunicação. Os parâmetros solicitados são obtidos através do arquivo de ajuda que acompanha cada driver de comunicação.
Bloco PLC	Semelhante ao tag tipo PLC, porém permite a leitura de vários dados simultaneamente. Em muitos casos, a utilização de tags tipo Bloco otimiza em muito a comunicação.
RAM	Tag de utilização interna, para guardar valores em memória. Os tags RAM são voláteis, ou seja, só guardam os valores enquanto o aplicativo estiver aberto.
Matriz	São tags RAM arranjados de forma a permitir acesso vetorial ou matricial.
Demo	Tag para simulação de valores. Permite gerar curvas definidas ou valores aleatórios.
Crono	Permite a criação de contadores e temporizadores.
Expressão	Tag que permite a entrada de uma expressão numérica ou alfanumérica (permite a soma entre strings).
DDE	<i>(Dynamic Data Exchange)</i> Tag para troca de dados com outras aplicações. Representa uma das maneiras de trocar dados entre aplicações comuns (como o Microsoft Excel e Access) ou ainda entre drivers de comunicação ( <i>DDE Servers</i> ) fornecidos por um fabricante.

## 4.2. Criando Tags

Para a criação de novos tags, basta selecionar no Organizer o item Tags ou um grupo de tags previamente criado e clicar em **Novo Tag**. Será mostrado o quadro **Criar um novo tag**, onde deverá ser informado o nome do tag, a quantidade e o tipo. Para uma quantidade maior que 1, o sistema numera automaticamente os tags, acrescentando um número depois do nome.

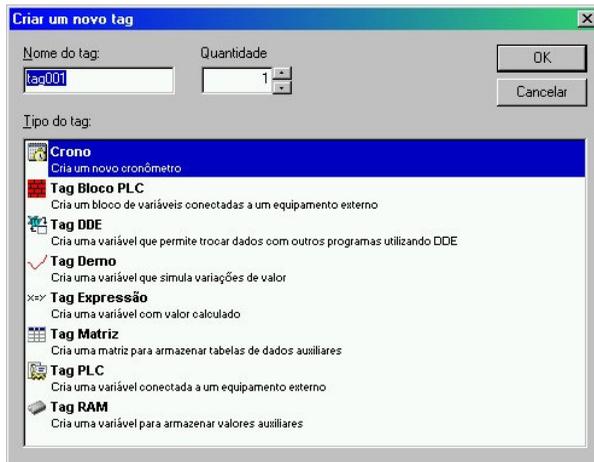


Figura 11: Criando um novo tag

### 4.2.1. Regras para os nomes dos Tags

Ao especificar o nome dos tags, algumas regras deverão ser seguidas:

- o nome não pode conter caracteres reservados, como operadores lógicos e aritméticos (+, -, \*, /) e caracteres especiais (? , ! , \ , | , & , % , \$ , # , @).
- o nome não pode conter espaço.
- o nome do tag não pode ser estritamente numérico, deverá ter uma letra inicial, pelo menos.

## 4.3. Tag PLC

Os tags tipo PLC são utilizados quando se deseja ler e escrever dados em um PLC (CLP), separadamente. Eles podem representar qualquer tipo de variável, como entrada ou saída digital ou analógica, a depender da configuração e endereçamento

requerido pelo driver. Antes de criar um tag PLC é necessário criar um objeto **Driver**, ao qual o tag será associado.

Os **drivers de comunicação** são bibliotecas (arquivos .DLL) reposnsáveis pela interligação do Elipse SCADA com algum equipamento externo. Na verdade, podemos utilizar um driver para se comunicar com qualquer coisa que possua uma interface de comunicação, seja uma máquina ou até mesmo um software (como no caso dos drivers de rede, como veremos mais adiante).

Cada driver de comunicação está associado um objeto **Driver** dentro do Elipse SCADA. Para criar um novo Driver, basta entrar no item **Drivers** a partir do Organizer e clicar no botão **Novo**. Na janela **Open**, indique o caminho para o arquivo .DLL desejado. Os arquivos de drivers podem ser instalados em separado, em qualquer diretório a ser definido pelo usuário.

Uma vez escolhido o arquivo de driver, deve-se fazer as configurações dos parâmetros de comunicação. Clicando no botão **Configurar**, vemos um tela onde podem ser especificados os dados gerais para a comunicação como: porta serial, taxa de comunicação e outros, de acordo com a documentação fornecida para cada driver. Para um auxílio à tarefa de configuração, pode-se apertar o botão **Ajuda**. O Elipse SCADA irá abrir o arquivo-texto com a documentação do driver.

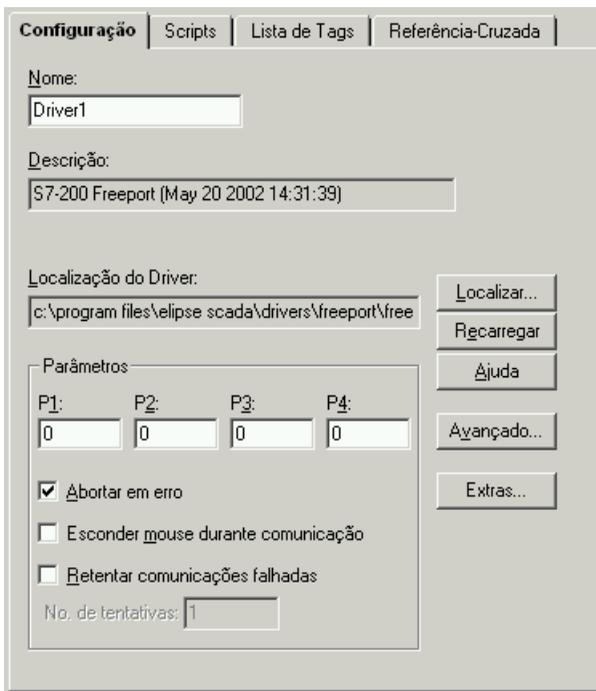


Figura 12: Janela Propriedades do Driver

### Propriedades do Driver

OPÇÃO	DESCRIÇÃO
Nome	Nome do objeto correspondente ao driver.
Localizar	Permite indicar um novo arquivo com o driver desejado. O nome e sua localização serão mostrados nos campos <i>Descrição</i> e <i>Localização do Driver</i> .
Recarregar	Carrega um novo driver escolhido, atualizando o sistema.
Ajuda	Chama a documentação do driver escolhido.

### Propriedades do Driver (parâmetros)

OPÇÃO	DESCRIÇÃO
P1, P2, P3 e P4	Campos para a entrada dos parâmetros para o PLC.
Abortar em erro	Esta opção faz com que seja mostrada uma caixa de diálogo requisitando o cancelamento da comunicação com o driver, no caso de erro de comunicação. Esta opção deve ser usada apenas em configuração, pois na execução pode ser perigosa, de modo que caso o operador responda “Sim” toda a comunicação será suspensa.
Esconder o mouse durante a comunicação	A opção “Esconder mouse durante comunicação” pode ser utilizada para verificar conflitos de interrupção na porta serial (normalmente não usado)
Retentar comunicações falhadas	A opção “Retentar comunicações falhadas” indica que o driver deve tentar reestabelecer uma comunicação perdida. Pode-se especificar um número de re-tentativas em caso de erro.
No. de tentativas	Número de tentativas no caso de erro de comunicação. Usar com cautela esta propriedade, pois se o equipamento apresenta erros de comunicação seguidos, é necessário uma revisão geral de toda a comunicação – caso contrário acarretará em atrasos na resposta geral do software.

Através do botão *Extra...*, ativo em alguns drivers, tem-se acesso a parâmetros especiais de configuração, como o uso de modems e geração de *debug* e *trace* da comunicação (para a depuração de aplicações). Consulte a documentação do driver para saber mais sobre estes parâmetros.

Podemos ver um exemplo de configuração extra na figura abaixo:

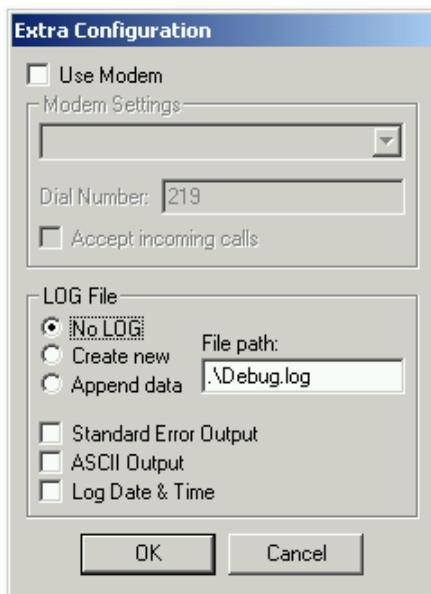


Figura 13: Janela Extras para o driver S7-200 Freeport da Siemens

Através do botão **Avançado...**, podemos abrir a janela para acesso às configurações avançadas de funcionamento do driver.



Figura 14: Quadro configuração avançada de driver

### Configurações avançadas para drivers de comunicação

OPÇÃO	DESCRIÇÃO
Manter comportamento 16-bits	Quando habilitado, é o modo de operação normal das versões 16 bits. Se desabilitado, opera de modo multitarefa híbrido, padrão das versões 32 bits. Este modo de trabalho dos drivers 32 bits coloca em um processo separado toda a tarefa de comunicação, de modo assíncrono à operação normal do Elipse SCADA, acelerando o processamento da aplicação. Este método é interrompido somente quando há requisições explícitas do usuário para obter o valor de uma variável, como num <i>script</i> (programa), onde o próximo passo depende da atualização do valor do tag. Nesse caso, o processo de troca de informações entre driver de comunicação e programa principal se torna síncrono.
Escrever em background...	Permite a realização de escrita em processamento paralelo. Permite que se escolha a prioridade da solicitação de escrita ao driver, com a mesma prioridade dos outros pedidos (prioridade baixa) ou no topo da lista de pedidos (prioridade alta).
Iniciar driver quando carregar aplicação	Esta opção executa automaticamente a função <code>StartComm()</code> ao iniciar a aplicação, fazendo com que o driver de comunicação esteja pronto para trocar informações. Caso fique desmarcada, o usuário deverá executar esta função via <i>script</i> , para permitir a comunicação.
Mantenha o driver na memória	Esta opção obriga o Elipse SCADA a não descarregar o driver da memória toda vez que se retorna ao ambiente de desenvolvimento.

### 4.3.1. Propriedades do tag PLC

Figura 15: Propriedades do Tag PLC

#### Propriedades do Tag PLC

OPÇÃO	DESCRIÇÃO
Nome	Nome do tag.
Mudar tipo para...	Permite que se mude o tipo do tag.
Acessar bits...	Permite desmembrar o tag em bits, criando tags <b>Bit</b> para um ou mais bits do tag.
Descrição	Uma breve descrição sobre o tag.
Driver	Permite a seleção do driver associado ao tag.
Ajuda	Mostra a ajuda do driver associado.

### **Mudando o tipo de tags**

Muitas vezes criamos tags que, de acordo com o desenvolvimento da aplicação, adquirem outras características do que as planejadas no início. O Eclipse SCADA facilita o reaproveitamento desses tags, dispondo da função **Mudar Tipo**, que permite a mudança do tipo do tag em qualquer momento.

Por exemplo, caso você tenha uma aplicação com muitos tags PLC e você queira transformá-los em tags elementos de bloco, pode-se usar o botão **Mudar tipo para**, onde deve ser escolhido o bloco de destino. A partir daí, os tags PLC serão transformados em elementos de bloco. O Eclipse SCADA irá mudar automaticamente todas as suas referências internas (utilização em telas, expressões, scripts e outros objetos).

### **Acessando os tags em bits**

Caso a variável lida seja uma palavra cujos bits são informações digitais relevantes, podemos separá-los. Clicar em **Acessar Bits** faz com que seja aberta uma janela para a especificação de quais bits serão expandidos. A utilização dos bits pode ser feita como sendo um tag normal, sendo apenas sua escrita “mascarada” com os outros bits antes de ser enviada ao equipamento. Tais características serão vistas com mais detalhes adiante, na seção **Tag Bit**.

**Propriedades dos tags PLC (parâmetros de I/O)**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
N1, N2, N3 e N4	Permite a configuração dos parâmetros para o driver associado.
Scan	Define de atualização dos valores do tag (em milisegundos).

### Endereçamento utilizando outros sistemas numéricos

Nos campos dos parâmetros do driver (N1, N2, N3 e N4), os valores podem ser expressos em decimais (de -32768 a 65535), octais (de 0o a 177777o) ou hexadecimais (de 0000h a FFFFh).

#### Propriedades dos tags PLC (opções de escala)

OPÇÃO	DESCRIÇÃO
Escala	Marcando esta opção os valores do tag serão convertidos para uma nova escala de valores conforme os limites definidos.
CLP Inferior	Define o valor mínimo a ser lido do CLP.
Sistema Inferior	Define o valor mínimo para a conversão na escala.
CLP Superior	Define o valor máximo a ser lido do CLP.
Sistema Superior	Define o valor máximo para a conversão na escala.
Testa conexão aqui	Permite a leitura e escrita de valores no CLP para testes.

### Escalas

Caso os valores que estão sendo lidos do equipamento estejam em uma escala diferente daquela que será utilizada em seu sistema, pode-se especificar uma conversão no próprio tag, determinando os níveis inferior e superior no equipamento (PLC) e inferior e superior no sistema. Ao utilizar a variável em qualquer parte do software (exceto no Organizer na função Testa Conexão Aqui, onde são mostrados os valores sem conversão), esta terá suas escalas automaticamente calculadas a cada leitura ou escrita.

#### Propriedades dos tags PLC (opções de I/O)

OPÇÃO	DESCRIÇÃO
Habilita leitura pelo scan	Caso esta opção esteja habilitada, este tag será lido (na taxa especificada no campo Scan) sempre que existir algum item na aplicação que esteja utilizando o tag.
Habilita leitura automática	Habilita o tag para ser lido caso seu valor se tornar necessário e a última leitura realizada maior que o tempo de varredura.
Habilita escrita automática	Quando de uma atribuição de valor ao tag, seja através de scripts ou por objetos de tela, indica se o driver irá enviar o novo valor automaticamente ao equipamento.

Através das opções de I/O dos tags PLC podemos otimizar a operação de nossa aplicação, fazendo acesso ao PLC somente quando necessário.

## 4.4. Tag Bloco

Os tags **Bloco PLC** (ou simplesmente bloco) têm a mesma finalidade dos tags PLC, ou seja, trocar informações com os equipamentos de aquisição de dados através dos drivers de comunicação fornecidos pela Elipse Software.

Sua vantagem porém, é permitir que vários tags tenham seus valores lidos ou escritos simultaneamente, otimizando o meio físico e diminuindo o tempo médio de varredura das variáveis. Em linhas gerais, cada bloco é associado a um driver de comunicação e possui um tempo de varredura que é o mesmo para todas as suas variáveis.

Na criação do tag bloco, o Elipse SCADA pergunta a quantidade de elementos que o bloco será composto. Uma vez feito isso, aparece na árvore do Organizer o tag Bloco e dentro dele, os elementos do bloco.

### 4.4.1. Propriedades do Tag Bloco

Figura 16: Propriedades de tag Bloco

### Propriedades de Tag Bloco

OPÇÃO	DESCRIÇÃO
Nome	Nome do tag.
Descrição	Uma breve descrição sobre o tag.
Driver	Permite selecionar o driver ao qual o tag estará associado.
Ajuda	Mostra a ajuda do driver selecionado.
B1, B2, B3 e B4	Permite a configuração dos parâmetros para driver associado.
Scan	Define de atualização dos valores do tag (em milisegundos).
<< Tamanho	Muda o tamanho do bloco a ser monitorado de acordo com o indicado na caixa "Tamanho", independente dos elementos.
Novo elemento...	Permite que você adicione novos elementos ao bloco.
Habilita leitura pelo scan	Habilita leitura em bloco a cada varredura.
Habilita leitura automática	Habilita leitura automática para o bloco.
Habilita escrita automática	Habilita escrita automática para o bloco.

## 4.5. Elemento de Bloco

---

Cada elemento do tag bloco possui propriedades que podem ser acessadas selecionando-se o elemento desejado na árvore da aplicação no Organizer. As mesmas considerações feitas para as tags PLC valem para cada um dos elementos do bloco, a saber:

- uso de escalas nos elementos;
- escrita e leitura nos elementos;
- transformação em bits;
- uso de alarmes.

Vemos as propriedades dos elementos de bloco a seguir:

Figura 17: Propriedades do Elemento de Bloco

### Propriedades do Elemento de Bloco

OPÇÃO	DESCRIÇÃO
Nome	Nome do elemento de bloco.
Descrição	Descrição do conteúdo do elemento.
Acessar bits...	Permite desmembrar o elemento em bits, criando tags <b>Bit</b> para um ou mais bits do elemento.
Bloco Index =	Permite mudar a ordem do elemento no bloco digitando um índice.
Testa conexão aqui	Permite a leitura e escrita de valores no CLP para testes.

**Propriedades do Elemento de bloco (Escala)**

OPÇÃO	DESCRIÇÃO
Escala	Marcando esta opção os valores do tag serão convertidos para uma nova escala de valores determinada pelo usuário conforme os limites definidos nos campos CLP Inferior, CLP Superior, Sist. Inferior e Sist. Superior.
CLP Inferior	Define o valor mínimo a ser lido do CLP.
Sistema Inferior	Define o novo valor mínimo para a conversão dos valores lidos.
CLP Superior	Define o valor máximo a ser lido do CLP.
Sistema Superior	Define o novo valor máximo para a conversão dos valores lidos.

## 4.6. Tag Bit

O **Tag Bit** somente pode ser criado a partir de outro tag e permite acessar individualmente cada bit do mesmo. Os tags que permitem o desdobramento em bits são: PLC, Demo, Expressão, Elemento de Bloco, RAM ou Remoto.

Este recurso é bastante útil quando um valor lido de um equipamento como um *byte* ou uma palavra, representa na verdade, 8 ou 16 (ou mais) estados digitais independentes (ligado ou desligado).

O valor do bit é obtido através do mascaramento do bit de sua posição com o tag ao qual ele pertence. Já a escrita, é feita de duas formas: mascaramento e escrita da palavra inteira ou escrita do bit individual, se o equipamento suportar tal comando. (Este comando é implementado de modo transparente ao usuário no driver de comunicação.)

Você pode criar um tag Bit a partir da página Geral. Clicando no botão **Acessar bits**, você poderá selecionar os bits que deseja mapear. A seleção dos bits é feita usando-se o mouse e as teclas [Shift] ou [Ctrl], da mesma forma em que se selecionam itens no sistema operacional Windows, por exemplo.



Figura 18: Criando um Tag Bit

O tag Bit pode ser tanto um único bit quanto um conjunto de bits, desde que sejam contínuos. Isto quer dizer que você pode mapear para um único tag Bit, por exemplo, os bits 0, 1 e 2, mas não os bits 10, 11 e 24. A opção existente nesta janela permite especificar se devem ser criados um tag para cada bit selecionado ou se os bits contínuos que estejam selecionados devem ser agrupados em um único tag.

Os tags Bit criados aparecem abaixo do respectivo tag na árvore da aplicação no Organizer. Ao selecionar um tag Bit específico, suas propriedades são mostradas ao lado direito da árvore. A página de propriedades gerais do tag Bit aparece quando selecionada a tab Geral no topo das páginas do tag Bit. Esta página é mostrada abaixo e seus respectivos campos são descritos na tabela que segue.

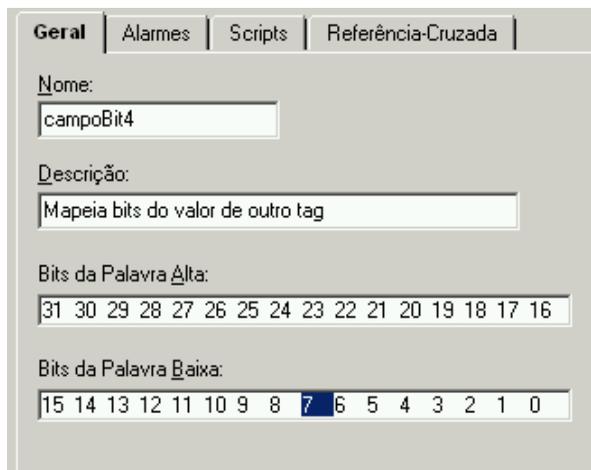


Figura 19: Propriedades do Tag Bit

**Propriedades de tag Bit**

OPÇÃO	DESCRIÇÃO
Nome	Nome do tag.
Descrição	Uma breve descrição sobre o tag.
Bits da Palavra Alta /Bits da Palavra Baixa	Define quais os bits que fazem parte do tag.

## 4.7. Tag Ram

---

**Tags RAM** são usados internamente para armazenar valores em memória. Este tipo de tag é volátil e por isso, mantém seus valores somente enquanto a aplicação está executando.

O tag RAM tem apenas o seu nome, descrição e valor inicial como propriedades que devem ser configuradas. Também é possível acessar os bits de um tag RAM, através do botão **Acessar bits...**

Para estabelecer um valor inicial para os tags RAM há duas maneiras:

1. Colocar o valor inicial (fixo) no campo **Valor Inicial**.
2. Armazenar o valor desejado em uma receita (cujos valores são modificáveis) e carregá-la ao iniciar a aplicação, o que faz com que os tags presentes na receita sejam não-voláteis.

Exemplos do uso de tag RAM serão vistos no capítulo sobre **Receitas**.

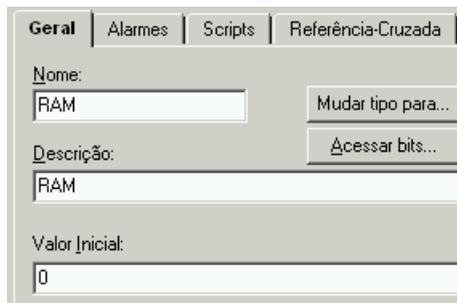


Figura 20: Propriedades do tag Ram

## 4.8. Tag Matriz

---

O **Tag Matriz** permite criar matrizes ou vetores de dados que podem ser usados em cálculos, armazenamentos e outras funções. É possível mapear cada célula de uma matriz como se fosse um tag e então associar cada uma a um tag ou propriedade. Neste caso, uma vez que o valor da célula muda, o tag ou propriedade associado assume o novo valor e vice-versa.

Importante: as operações sobre matrizes sempre tem linha e coluna começando com o índice 1.

### 4.8.1. Propriedades do Tag Matriz

The image shows a dialog box titled 'Propriedades do Tag Matriz' with two tabs: 'Geral' and 'Referência-Cruzada'. The 'Geral' tab is active. It contains the following fields and controls:

- Nome:** A text box containing the value 'Matriz'.
- Descrição:** A text box containing the value 'Matriz'.
- Colunas:** A spin box with the value '3'.
- Linhas:** A spin box with the value '3'.
- Below the spin boxes, there is a text label: 'Clique aqui para associar uma célula a um tag:'.
- At the bottom, there is a button labeled 'Associar...'.

Figura 21: Propriedades do Tag Ram

#### Propriedades do tag Matriz

OPÇÃO	DESCRIÇÃO
Nome	Nome do tag.
Descrição	Uma breve descrição sobre o tag.
Colunas	Define o número de colunas da matriz
Linhas	Define o número de linhas da matriz
Associar...	Mapeia todas ou somente algumas células da matriz para um tag.

#### ***Mapeando células para tags***

Você pode mapear uma célula da matriz ou vetor para um tag pressionando o botão **Associar...** na página **Geral** do tag e especificando qual ou quais as células que deseja mapear.

Cada célula mapeada aparece abaixo do tag na árvore da aplicação no Organizer. Ao selecionar uma célula específica, suas propriedades são mostradas ao lado direito da árvore. Cada célula mapeada possui 4 páginas de propriedades: **Geral**, **Alarmes**, **Scripts** e **Tags**. As 3 primeiras páginas são as mesmas de qualquer tag e a página de tags permite associar um tag ou propriedade à célula da matriz da mesma forma em que tags e propriedades são associados a objetos de tela.

## 4.9. Tag Demo

O **Tag Demo** é usado para a simulação de valores a partir de curvas pré-definidas ou aleatoriamente. A geração é feita conforme o tipo de curva selecionada nos seis botões da página **Geral** das propriedades do tag (ver figura a seguir).

Tags Demo podem ajudá-lo a testar sua aplicação ou podem ser usados, por exemplo, em um objeto de tela **Animação** para mostrar os quadros da animação de acordo com a variação do tag.

### 4.9.1. Propriedades do Tag Demo

The image shows a software interface for configuring a 'Tag Demo'. It features a tabbed menu at the top with 'Geral' selected. Below the tabs, there are several input fields and buttons. The 'Nome' field contains 'Demo', and the 'Descrição' field contains 'Tag Demo'. There are two buttons: 'Mudar tipo para...' and 'Acessar bits...'. A 2x3 grid of buttons displays different waveforms: a smooth curve, a square wave, a sine wave, a sawtooth wave, a triangular wave, and a zigzag wave. Below this grid are five input fields: 'Limite Inferior' (0), 'Limite Superior' (20000), 'Incremento' (1), 'Espera' (1), and 'Período' (100). At the bottom, there is a checked checkbox labeled 'Habilitado'.

Figura 22: Propriedades do Tag Demo

**Propriedades do Tag Demo**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
Nome	Nome do tag.
Descrição	Uma breve descrição sobre o tag.
Acessar bits...	Permite desmembrar o tag em bits.
Mudar tipo para...	Permite que se mude o tipo do tag.
Tipo	Define o tipo de curva a ser usada para a variação de valores.
Limite Inferior	Define um valor mínimo limite para o valor do tag .
Limite Superior	Define um valor máximo limite para o valor do tag .
Incremento	Define o incremento para a curva do tipo “dente de serra”.
Espera	Define o número de períodos entre cada geração de valor. Por exemplo, se for 2, gera um valor a cada dois períodos. É usado junto com o atributo <b>Período</b> para controlar o intervalo de tempo para a variação dos dados.
Período	Define um valor em milissegundos para o período da geração de valores. É usado em conjunto com o atributo <b>Espera</b> .
Habilitado	Os valores são atualizados apenas quando essa opção está ligada. Caso contrário, o valor do tag permanece o mesmo.

## 4.10. Tag Crono

---

O **Tag Crono** (cronômetro) permite realizar operações básicas para contagem de tempo (crescente e decrescente) e temporizações, permitindo executar tarefas quando um certo valor é atingido.

The image shows a configuration window for a 'Tag Crono' (timer) with the following elements:

- Three tabs: 'Geral' (selected), 'Scripts', and 'Referência-Cruzada'.
- 'Nome:' field containing 'Crono'.
- 'Descrição:' field containing 'Crono'.
- 'Incremento:' field containing '1' followed by 'seg'.
- 'Preset:' field containing '60' followed by 'seg'.
- Four checkboxes:
  - Iniciar cronômetro ao rodar aplicação
  - Zerar o acumulador ao ligar o cronômetro
  - Salvar acumulador ao sair
  - Reinício automático (execução em loop)

Figura 23: Propriedades do Tag Crono

## 4.11. Tag DDE

O **Tag DDE** é usado para troca de dados entre o Elipse SCADA e outras aplicações (Microsoft Excel ou Microsoft Access, por exemplo) usando DDE (*Dynamic Data Exchange*).

Em uma rede Windows, o Elipse SCADA pode usar NetDDE (DDE em rede), o que permite trocar dados com outro Elipse SCADA, dentre outras formas, através de tags DDE.

### 4.11.1. Propriedades do Tag DDE

Figura 24: Propriedades do Tag DDE

#### Propriedades do Tag DDE

OPÇÃO	DESCRIÇÃO
Nome	Nome do tag.
Mudar tipo para...	Permite que se mude o tipo do tag.
Descrição	Uma breve descrição sobre o tag.

**Propriedades do Tag DDE (Servidor)**

OPÇÃO	DESCRIÇÃO
Computador	Define o nome do computador onde se encontra a aplicação que é o Servidor DDE. O padrão é o computador corrente, mas outros computadores na rede poderão estar disponíveis através de NetDDE.
Servidor	Define o nome do Servidor DDE. Pode ser uma aplicação ou um driver DDE fornecido pelo fabricante do seu equipamento. Uma lista dos programas Servidor DDE disponíveis aparece quando a seta ao lado desta caixa é pressionada.
Tópico	Define o nome do tópico do Servidor DDE. A lista de tópicos disponíveis aparece quando a seta ao lado desta caixa é pressionada.
Item	Define o nome do item do Servidor DDE.
Testar Conexão	Permite que você teste a configuração DDE. Uma mensagem pode indicar um erro de conexão ou o valor recebido pelo item configurado.

**Propriedades do Tag DDE (Escala)**

OPÇÃO	DESCRIÇÃO
Escala	Marcando esta opção os valores do tag serão convertidos para a nova escala de valores definidos nos campos do quadro.
Servidor	Define os valores mínimo e máximo a serem lidos do servidor.
Sist. Inferior	Define o novo limite mínimo para a conversão dos valores lidos.
Sist. Superior	Define o novo limite máximo para a conversão dos valores lidos.

## 4.12. Tag Expressão

O Tag Expressão permite que você atribua uma expressão numérica ou alfanumérica a um Tag. Você pode criar equações envolvendo variáveis quaisquer, sejam elas numéricas, alfanuméricas, tags ou atributos.

Ao digitar a expressão, que será a operação que o tag realizará, automaticamente no campo Erros aparecerão os erros encontrados na edição até aquele momento. As mesmas funções, operadores e constantes usadas nos scripts (módulos de programação) podem ser usadas nos tags Expressão (ver capítulo sobre Scripts).

Para utilizá-los, ao editar a expressão basta chamar o AppBrowser, onde aparecerá uma janela que possibilita copiar todas as funções ou atributos disponíveis na aplicação para a linha de edição.

The image shows a software interface window titled 'Propriedades do tag Expressão'. It has a tabbed interface with 'Geral' selected. The 'Nome' field contains 'tag006' and has a 'Mudar tipo para...' button next to it. The 'Descrição' field also contains 'tag006' and has an 'Acessar bits...' button next to it. Below these is a large text area for 'Expressão', which is currently empty. At the bottom, there is a section for 'Erros' which displays the text 'No errors'.

Figura 25: Propriedades do tag Expressão

### Propriedades do tag Expressão

OPÇÃO	DESCRIÇÃO
Nome	Nome do tag.
Mudar tipo para...	Permite que se mude o tipo do tag.
Acessar bits...	Permite desmembrar o tag em bits.
Descrição	Uma breve descrição sobre o tag.
Expressão	Permite a entrada de qualquer expressão válida para o tag.
Erros	Lista erros de sintaxe encontrados na expressão. Os erros são mostrados durante a edição da expressão. Para que se tenha uma expressão válida a mensagem <b>No errors</b> deve aparecer neste campo.

As mesmas funções, operadores e constantes usadas nos Scripts podem ser usadas nos tags Expressão. Veja o capítulo Scripts em Constantes & Operadores, onde são listados os operadores e constantes que podem ser usados na expressão e que também são suportados nos *scripts*. Aqui temos alguns exemplos de constantes:

números inteiros	1234, -1234, 10011b (binário), 733o (octal), 0A100h (hexa)
números reais	1.2345
strings	“temperatura”, “pressão”

# Exercícios

## 1. Estabelecer o driver de comunicação a ser utilizado na aplicação.

- ➔ Carregue o driver de comunicação e com o auxílio do arquivo de ajuda, preencha os parâmetros P1 a P4, habilitando a opção **Abortar em erro** e desmarcando a opção **Esconder mouse durante comunicação**. Não usar retentativas.

## 2. Criar um tag tipo PLC para representação de uma entrada digital.

- ➔ Selecionar o objeto **Tags** no Organizer, clicar no botão **Novo Tag**.
- ➔ Digite “DI” na propriedade nome do Tag.
- ➔ Digite “1” no campo **Quantidade**.
- ➔ Escolha o tag tipo PLC, clicando depois no botão **OK**.
- ➔ Associe o driver de comunicação através do campo **Driver**.
- ➔ Especifique os parâmetros de N1 a N4.
- ➔ Lembre-se que na seção **Testa Conexão Aqui**, há possibilidade de ler e escrever valores diretamente no equipamento.
- ➔ Criar um tag tipo PLC para representação de uma saída digital.
- ➔ Selecionar o objeto **Tags** no Organizer, clicar no botão **Novo Tag**.
- ➔ Digite “DO” na propriedade **Nome do Tag**.
- ➔ Digite “1” no campo **Quantidade**.
- ➔ Escolha o tag tipo PLC, clicando depois no botão **OK**.

## 3. Criar um novo grupo de tags tipo PLC com três tags para representar níveis de tanques.

- ➔ Selecionar o objeto **Tags** no Organizer, clicar no botão **Novo Grupo**.
- ➔ Digitar “Níveis” na propriedade **Nome**.
- ➔ Selecionar o grupo **Níveis** e clique em **Novo Tag**.
- ➔ Digitar “Tank01” no campo **Nome**.
- ➔ Digite “3” no campo **Quantidade**;
- ➔ Escolha o tag tipo PLC, clicando depois no botão **OK**.

**OBS:** Quando geramos um grupo, são criados 3 tags do tipo PLC com parte do nome idêntico porém com índice numérico diferente (em ordem crescente), pois não podem existir dois tags com o mesmo nome.

4. **Criar um bloco de comunicação com 3 elementos.**
  - ➔ Seguir os mesmos procedimentos para a criação de tags, escolhendo tipo Bloco.
  - ➔ Digitar “Temperaturas” na propriedade Nome.
  - ➔ Escolher “1” em Quantidade.
  - ➔ Em Entre Tamanho Bloco, escolher “3”.
  - ➔ Dentro de Temperaturas, selecione os três elementos, digitando na propriedade Nome o texto “Temperatura01”. Automaticamente os outros elementos do bloco terão os nomes “Temperatura02” e Temperatura03”.
5. **Separar em bits os tags DI e DO.**
  - ➔ Selecione através do Organizer o tag DI e logo após Acessar Bits.
  - ➔ Escolha os bits indicados pelo instrutor, especificando a opção Criar um tag para cada bit. Serão criados bits associados ao tag DI, representando as entradas digitais.
  - ➔ Seguir o mesmo procedimento para o tag DO.
3. **Criar variáveis RAM para o cadastramento e armazenamento das quantidades de matérias primas.**
  - ➔ Criar um novo grupo de tags, chamado produtos.
  - ➔ Criar a partir deste grupo os tags RAM: codigo, agua, acucar, xarope, glucose e numero\_receita.
  - ➔ Não é necessário especificar um valor inicial.
4. **Criar um tag tipo Demo para animação do misturador no funil.**
  - ➔ Selecionar o objeto Tags no Organizer, escolher Novo Tag. Na propriedade Nome digite “Mix” e aceite, clicando OK.
  - ➔ Nas propriedades do tag Mix, escolha a opção de onda triangular, com limite inferior 0 e superior 9.
5. **Criar um tag expressão que será a combinação de três tags digitais, chamado Status.**
  - ➔ Este tag mostrará um exemplo útil quando se deseja mostrar na tela uma indicação ou animação que possui mais de dois estados (ligado, desligado, falha, etc.). Neste caso é necessário criar um tag expressão. Seguir os mesmos procedimentos para a criação de tags, escolhendo agora o tipo Expressão.
  - ➔ No campo Nome, digite “Status”.
  - ➔ Clique agora no campo Expressão. Neste momento, há dois caminhos: você pode digitar diretamente o texto desejado ou utilizar a ferramenta

AppBrowser para navegar pela aplicação, permitindo copiar a função, atributo ou objeto desejado diretamente para o local de edição.

- ➔ No primeiro caso, digite:
- ➔ Tags.DI.CampoBit1\*4 + Tags.DI.CampoBit2\*2 + Tags.DI.CampoBit3
- ➔ O resultado final será um valor de 0 a 7, segundo as seguintes possibilidades:

Bit1	Bit2	Bit3	Status
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- ➔ No segundo caso, acesse o botão AppBrowser e clique em Tags, selecionando o tag DI.
- ➔ Clique duas vezes e procure o item CampoBit1. Depois de selecionado, clique em Copiar para Script, onde o item desejado será transferido para a expressão no tag Status.
- ➔ Agora você deve digitar os sinais “\*” e “4” para completar a primeira parte da expressão.
- ➔ Complete o procedimento para a expressão ficar igual ao primeiro caso.

## 6. Criar uma ligação entre uma célula do Excel e um tag tipo DDE.

- ➔ Selecionar o objeto tags no Organizer, clicar no botão Novo Tag.
- ➔ Digite “Planilha” na propriedade Nome.
- ➔ Digite “1” no campo Quantidade.
- ➔ Escolha Tag DDE e clique OK.
- ➔ Abra o Microsoft Excel e numa nova planilha, digite um valor qualquer na primeira célula e salve-a.

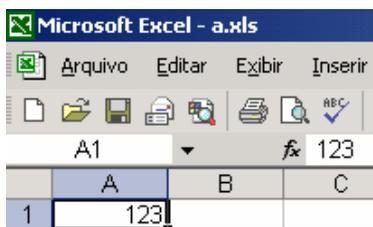


Figura 26: Célula do Excel

- ➔ Nas propriedades do tag **Planilha**, escolha “Excel” para **Nome do servidor**, **Sheet1** para **Tópico** e no campo **Item**: “R1C1” (para a versão do Excel em inglês) ou “L1C1” (português).

 A screenshot of the "Propriedades do Tag Planilha" dialog box. It has four tabs: "Geral", "Alarmes", "Scripts", and "Referência-Cruzada". The "Geral" tab is selected. The fields are:
 

- Nome:** "Planilha" (with a "Mudar tipo para..." button next to it)
- Descrição:** "tag001"
- Nome do servidor:** "Excel" (dropdown menu)
- Tópico:** "[a.xls]Plan1" (dropdown menu)
- Item:** "L1C1" (with a "Testar Conexão" button next to it)

Figura 27: Propriedades do Tag Planilha

- ➔ Clique em **Testar Conexão** e o valor digitado na célula aparecerá.

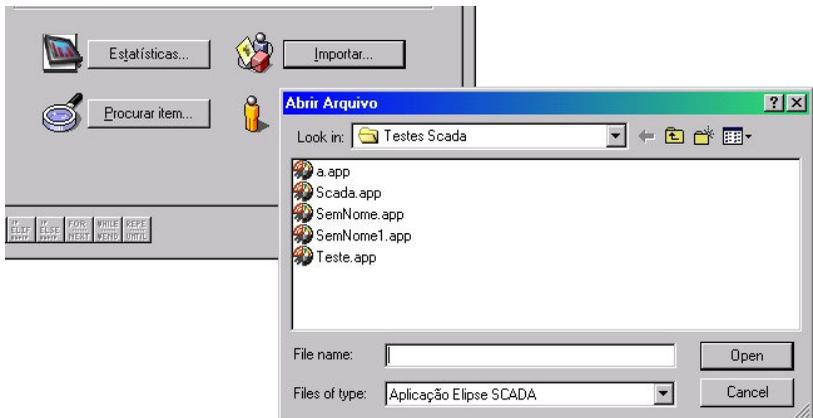


## 4.13. Dicas sobre Tags

### Importando tags de outras aplicações

A fim de permitir o aproveitamento do trabalho realizado em outra aplicação, ou mesmo permitir que mais de uma pessoa trabalhe no mesmo aplicativo, pode-se importar partes de outras aplicações. Utilize a ferramenta **Importar** presente no item **Aplicação** no Organizer, para realizar tal tarefa.

Será interrogado o caminho da aplicação de origem, que contém os objetos que deseja copiar. Selecione o item desejado e clique em **Open**.



### Copiando ou movendo tags ou grupo de tags

Utilize a ferramenta de duplicação  do Organizer para realizar cópias de tags ou grupos. O mesmo procedimento pode ser utilizado para os outros objetos, como telas ou objetos de tela.

Para mover um tag ou um grupo de tags através do Organizer, basta clicar sobre o tag ou grupo e arrastá-lo (sem soltar o botão do mouse) até a localidade destino, que pode ser outro grupo ou o item **Tags** e soltá-lo. Todas as suas referências (ligações com outros objetos ou scripts) serão atualizadas.

### Configurando múltiplos tags

Muitas vezes é necessário configurarmos vários tags de maneira idêntica para diversos atributos. Isso é possível, fazendo uma seleção múltipla. Primeiro, deve-se selecionar todos os tags desejados no Organizer - arraste o mouse iniciando no

primeiro tag até o último tag da lista. Desta maneira, todos os tags serão marcados. Caso queira selecionar somente alguns tags, pressione a tecla [Ctrl] enquanto clica sobre cada tag desejado, separadamente. Feito isso, cada ação ou digitação que for feita será realizada em todos os tags.

### ***Dê preferência aos tags de comunicação em bloco***

A utilização de tags Bloco permite a otimização do meio físico, já que num processo de comunicação serial genérico, boa parte dos caracteres transmitidos são de controle e verificação. Com o uso dos blocos fazemos com que tais caracteres sejam enviados um número menor de vezes devido ao encapsulamento de maior número de variáveis na mesma transmissão.

### ***Ajuste do tempo de varredura (scan) de tags de comunicação***

Procure programar o tempo de scan das variáveis com valores próximos do real. Caso seja especificado um tempo muito baixo de scan para todas as variáveis, o que provavelmente vai ocorrer é que, por limitações do meio físico, nem todas as variáveis poderão ser coletadas na taxa especificada, o que gerará queda de performance na comunicação, não permitindo que certas variáveis que realmente necessitam de uma busca mais rápida ocupem a comunicação.

### ***Informação de tempo nos tags***

Alguns equipamentos permitem o envio de informações de tempo, juntamente com os valores das variáveis. Os drivers de comunicação para tais equipamentos podem, a cada tag consultado, retornar também a informação deste relógio.

Esta informação pode ser obtida através da propriedade **TimeStamp**, presente em todos os tags, com precisão de 1 milisegundo. Quando o equipamento não suporta este tipo de informação ou quando o driver não está preparado para tal, o próprio programa principal realiza a tarefa de informação do instante de coleta, preenchendo com o valor do relógio do PC.

**OBS:** No caso dos tags tipo Bloco, é informado o mesmo TimeStamp para todos os elementos do bloco, já que foram consultados no mesmo instante de tempo. No caso de serem eventos distintos, devem ser lidos como tags tipo PLC.

### ***Escrita automática em tags PLC***

Ao atribuir um valor diretamente a um tag PLC ou elemento de bloco que possua a propriedade **escrita automática** habilitada, o comando é enviado diretamente ao driver de comunicação, que por sua vez o repassa ao equipamento associado. Tal ação não ocorre somente quando o valor atribuído for igual ao conteúdo que já estava no tag. Caso queira forçar uma escrita mesmo assim, deve ser executada a função `Write()` do tag, em algum script.

## 4.14. Página de Alarmes

Cada tag que é definido possui uma página de **Alarmes**, onde podem ser configurados quatro intervalos de valores e prioridades para alarmes.

Alarmes são usados para sinalizar algum evento que possa vir a ocorrer com a variável permitindo inclusive, a tomada de ações apropriadas através de scripts.

Para visualizar os alarmes configurados para um tag, você precisa inserir um objeto de tela **Alarme**. Este objeto pode mostrar também alarmes já ocorridos que estejam registrados em um arquivo de alarmes e outros alarmes ativos no sistema.

Para imprimir os alarmes ocorridos no sistema, você pode definir um **Relatório** através do Organizer e executar a função especial **Print** em um script.

A página de alarmes dos tags aparece quando selecionada a tab **Alarmes** no topo das páginas do tag.

	Valor:	Pri:	Comentários
<input type="checkbox"/> LoLo	1000	1	
<input type="checkbox"/> Low	5000	1	
<input type="checkbox"/> High	15000	1	
<input type="checkbox"/> HiHi	19000	1	
<input checked="" type="checkbox"/> Logar mensagens de retorno			

Grupo de Alarmes:  
 (grupo padrão) ▼

Manter valor do tag sempre atualizado

Usa outro nome de tag:

Figura 28: Opções de configuração de alarmes para tags

## Propriedades de configuração de alarmes para tags

OPÇÃO	DESCRIÇÃO
LoLo	Alarme Baixo Crítico. Define um intervalo de valores (menor igual) onde o Tag é considerado em um estado de Alarme Baixo Crítico. É usado quando o valor do Tag está abaixo de um mínimo, ou seja, extremamente baixo.
Baixo	Alarme Baixo. Define um intervalo de valores (menor igual) onde o Tag é considerado em estado de alarme baixo. É usado quando o valor do Tag está abaixo do normal.
Alto	Alarme Alto. Define um intervalo de valores (maior igual) onde o Tag é considerado em estado de Alarme Alto. É usado quando o valor do Tag está mais alto do que o normal.
HiHi	Alarme Alto Crítico. Define um intervalo de valores (maior igual) onde o Tag é considerado em estado de Alarme Alto Crítico. É usado quando o valor do Tag é está acima de um máximo, ou seja, extremamente alto.
Valor	Define os limites para cada situação possível de alarme (lolo, low, hi, hihhi).
Prioridade	Define a prioridade para cada situação de alarme. Números pequenos indicam alta prioridade (a prioridade deve ser um número entre 0 e 999). Para um melhor controle os alarmes de maior prioridade irão aparecer em primeiro plano na janela de alarmes (Objeto de Tela Alarme).
Comentário	Um comentário ou mensagem pode ser definido para cada alarme. Podem ser usados até 100 caracteres.
Mensagem de retorno	Habilita o log da mensagem de retorno de alarme.
Grupo de alarmes	Define o grupo de alarmes, cujo arquivo receberá as mensagens de ocorrências.

O intervalo entre o nível Low e High de alarme (se configurados) representam o estado de operação normal da variável. Ao ultrapassar um desses limites, a ocorrência é registrada (*log*) como um alarme ativo. Caso a variável retorne ao estado normal, é registrada uma ocorrência de retorno, caso esta opção esteja ativada.

## 4.15. Alarmes e Grupos de Alarmes

Cada vez que ocorre um alarme, são gravados todos os dados do evento, como data e hora, tipo de evento, valor do tag, etc. Cada alarme pode estar associado a um grupo de alarmes definido pelo usuário no item Alarmes do Organizer.

Figura 29: Propriedades do Grupo de alarmes

Nesta página, podemos configurar um arquivo para a gravação dos alarmes, bem como sons e mensagens de alerta, que será reconhecido como **grupo de alarmes padrão**.

### Propriedades do Grupo de alarmes

OPÇÃO	DESCRIÇÃO
Nome	Nome do Grupo de Alarmes
Descrição	Descrição sobre o grupo
Habilita reg	Habilita a gravação de dados em disco para o grupo
Gravação	Número máximo de registros. O arquivo é rotativo, ou seja, conterà somente o número aqui especificado, que serão os mais novos.
Nome do arquivo	Nome do arquivo em disco
Novo grupo de alarmes!	Cria um novo grupo de Alarmes

Um grupo de alarmes diferente pode ser utilizado quando desejar separar alguns tipos de alarmes de tags, de modo que sejam armazenados em arquivos separados. Para visualizá-los ou imprimi-los, neste caso, é necessário criar objetos e relatórios separados, um para cada arquivo.







Uma **Tela** pode ser definida como uma janela para monitoramento de um processo, onde serão inseridos os objetos que farão a interface do operador com o sistema. Cada aplicação pode ter um número ilimitado de telas.

As telas são o ponto-de-partida para a construção da interface de sua aplicação. Um bom desenho de tela garante uma compreensão melhor do processo supervisionado e utilização mais fácil dos recursos acrescentados à aplicação.

Você pode criar uma nova tela pressionando o botão  na barra de ferramentas ou usando o comando **NOVO** (*New*) no menu **Tela** (*Screen*). No Organizer, quando a opção **Telas** é selecionada, é mostrada uma janela contendo uma lista de todas as telas da sua aplicação. Você pode criar, apagar e navegar pelas telas da aplicação utilizando os botões à direita (**Criar**, **Deletar** e **Ir Para**).

Junto com estes botões existem os botões **Mostrar** e **Esconder** que permitem mostrar uma tela específica ou escondê-la durante o desenvolvimento. Para fazer isso em tempo de execução, pode-se modificar a propriedade **Visible** da tela. Por exemplo, é possível criar uma tela de aviso para indicar uma condição de alarme que só será mostrada quando essa condição for atingida (colocando o valor **TRUE** na propriedade **Visible**). No momento que a condição for desfeita, pode-se esconder novamente a tela.

## 5.1. Propriedades Gerais de Telas

---

Para cada nova tela, você pode acrescentar objetos de tela, definir imagens de fundo e entre outras propriedades. Para visualizar ou editar as propriedades da tela corrente, você tem diversas maneiras:

- clicando no botão  na barra de ferramentas;
- dando um duplo clique em um espaço vazio da tela em questão;
- usando o comando **Propriedades** (*Properties*) do menu **Tela** (*Screen*) ou
- quando se está editando a lista de telas que aparece ao selecionarmos o item **Telas** no Organizer.

A seguir temos um exemplo da guia **Geral** no Organizer, com as propriedades de telas.

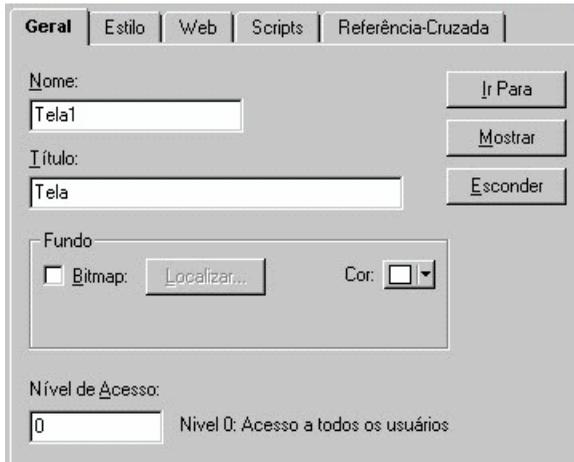


Figura 30: Propriedades Gerais da Tela

### Propriedades Gerais da Tela

OPÇÃO	DESCRIÇÃO
Nome	Define um nome para a tela corrente. Usando este nome você pode abrir a tela de qualquer parte da aplicação usando botões ou teclas de função, bem como associá-la a scripts.
Título	Define um título para a tela, usado também como sua descrição.
Nível de acesso	Define o nível de acesso para a Tela, que será verificado com o nível de acesso do usuário ao entrar na Tela.
Bitmap	Habilita / Desabilita o uso de um bitmap como fundo para a Tela corrente. Você pode usar o botão Browse para encontrar os bitmaps.
Localizar	Permite navegar na estrutura de diretórios a fim de encontrar os arquivos-imagem que serão usados como fundo para a Tela. O caminho e nome do bitmap aparecem abaixo do campo.
Cor	Define a cor de fundo para a tela corrente. Este parâmetro é usado quando não existe um bitmap selecionado ou quando o bitmap não preenche toda a Tela.

## 5.2. Propriedades do Estilo da Tela

A página propriedades do Estilo da Tela aparece quando selecionada a guia *Estilo* no topo das páginas da Tela. Esta página é mostrada abaixo e seus respectivos campos são descritos na tabela que segue.

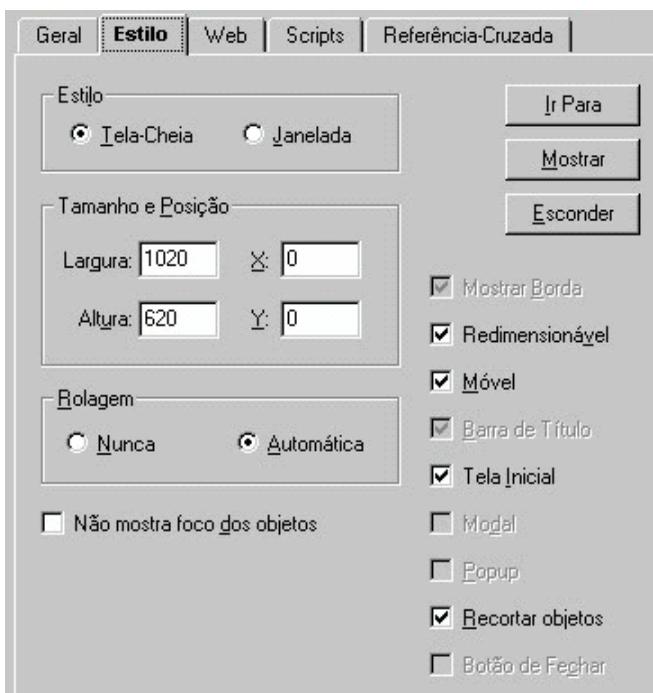


Figura 31: Propriedades de estilo em telas

### Propriedades de estilo de Telas

OPÇÃO	DESCRIÇÃO
Estilo/Tela Cheia	Determina que a tela ocupe toda a janela da aplicação.
Estilo/Janelada	Determina que a tela apareça dentro de uma janela com as dimensões especificadas nos campos Tamanho e a Posição.
Largura	Define a largura da tela em <i>pixels</i> .
Altura	Define a altura da tela em <i>pixels</i> .
X	Determina a coordenada X para o canto superior esquerdo da tela em <i>pixels</i> .
Y	Determina a coordenada Y para o canto superior esquerdo da Tela em <i>pixels</i> .
Rolagem/Nunca	Determina que a janela não tenha barras de rolagem, mesmo quando se faça necessário.
Rolagem/Automática	Barras de rolagem aparecerão automaticamente nas laterais da tela quando se fizer necessário.
Mostrar borda	Inserir borda na tela.
Redimensionável	Permite o redimensionamento da janela em tempo de execução.
Móvel	Permite que a janela seja movida em tempo de execução.
Barra de Título	Mostra ou esconde a Barra de Título.
Tela Inicial	Torna a tela a primeira a ser mostrada em execução.
Modal	Não permite que o usuário deixe a tela sem fechá-la.
Popup	Permite que, ao clicar fora da tela, esta seja automaticamente fechada.
Recortar Objetos	Permite abertura rápida de telas com número menor de objetos.
Botão de Fechar	Habilita o botão Fechar na janela (padrão Windows: um "x" no canto superior direito).

### Importando imagens para o fundo de telas

Para uma melhor qualidade gráfica da aplicação, pode ser inserido uma imagem como fundo de tela. O Elipse SCADA permite arquivos gráficos com formato BMP, JPEG ou GIF que pode ser gerado em diversos aplicativos de desenho ou manipulação de imagens como Microsoft Paint, Corel Draw!, Adobe Photoshop, AutoDesk AutoCAD e outros. Você pode criar seus desenhos (bitmaps) em qualquer tamanho e cores que desejar. O Elipse SCADA irá importá-los automaticamente sem a necessidade de qualquer processo de conversão.

# Exercícios

## 1. Criar a Tela Principal para monitoração da produção.

- ➔ Clique no botão  Nova Tela na barra de ferramentas. Caso você já tenha uma tela vazia criada (ao iniciar um novo aplicativo sempre é criada uma tela automaticamente), vá para o passo seguinte.
- ➔ A partir dessa nova tela pode-se definir os objetos de animação, o desenho de fundo do sinótico e todas as características específicas da tela. A lista das telas existentes na aplicação fica disponível na barra de ferramentas para o carregamento durante o processo de configuração e criação.
- ➔ Para configurar as propriedades da tela, clique no botão  Propriedades.
- ➔ Nas propriedades da tela nova digite “Dosagem” na propriedade Nome e “Tela de Dosagem” na propriedade Título.
- ➔ Marque a opção Bitmap, pressione o botão Localizar e selecione o arquivo fundomodelo.bmp.
- ➔ Na guia Estilo, marque as opções estilo Tela-Cheia e rolagem Automática.

## 2. Criar uma tela de alarmes para o sistema.

- ➔ Crie uma nova tela e configure com nome “Alarmes” e título “Tela de Alarmes”.
- ➔ Coloque a cor de fundo laranja, através da opção Outras Cores... na página principal.
- ➔ Configure os estilos Janelada, rolagem Automática e opções Botão de Fechar, Móvel e Barra de Título marcadas.
- ➔ Desmarque a opção Tela Inicial.

## 3. Criar Tela de Tendências, nos mesmos moldes na Tela de Alarmes.

- ➔ Uma das opções seria repetir o procedimento anterior. Porém, outra maneira interessante seria duplicar a tela anterior e modificar apenas os pontos necessários.
- ➔ Através do Organizer selecione a tela de alarmes.
- ➔ Clique no ícone Duplicar  na barra de ferramentas, que permite fazer cópias de qualquer objeto. Em seguida, aparece uma caixa de diálogo,







As telas de aplicação podem conter bitmaps de fundo e objetos. Os objetos que são inseridos sobre o bitmap constituem um plano secundário na tela, de modo que podem ser deletados, copiados, movidos, redimensionados e agrupados, sem prejudicar o desenho de fundo.

Os **Objetos de Tela** são elementos gráficos que estão relacionados com os tags de modo a realizar uma interface amigável com as variáveis. Os objetos previamente disponíveis são os seguintes:



**Slider**

Permite visualizar ou enviar valores para um tag através de um potenciômetro (botão deslizando).



**Tendência**

É utilizado para visualizar um gráfico de tendência com até 16 tags (que podem ser trocados em execução), executando a coleta em tempo-real ou em segundo plano. Pode desenhar gráficos de variáveis por tempo ou de variáveis em relação a outras (XY).



**Botão**

Para acionamentos ou execuções de tarefas especificadas pelo usuário através do mouse ou teclado.



**Gauge**

Mostrador de valores analógicos com ponteiros (medidor).



**Texto**

Este objeto permite atribuir mensagens a intervalos de valores dos tags denominados **Zonas**, definindo cores e textos para cada um deles.



**Barras**

Utilizado para visualizar dados na forma de barra. Podem ser mostrados até 16 tags em cada objeto de barras.



**Display**

Mostrador numérico/alfanumérico em tempo real.



**Animação**

Para criar animações usando bitmaps definidos pelo usuário.



**Setpoint**

É uma caixa de edição, para digitação e envio de valores para uma variável.



**Alarmes** Permite a visualização dos alarmes ativos (Sumário) ou dos alarmes logados no arquivo de alarmes (Histórico).



**Browser** Permite a visualização de arquivos de banco de dados na tela.



**Bitmap** Permite inserir imagens de qualquer tamanho sobre a tela.



**Watcher** Permite inserir objetos para a visualização de filmes ou vídeo ao vivo no sistema, através de placas de aquisição.

## 6.1. Edição dos objetos de Tela

---

Os objetos de tela podem ser criados a partir da barra de ferramentas ou através do menu **Objetos**. Uma vez selecionado o objeto que se deseja criar mantenha o botão esquerdo do mouse pressionado na área da tela enquanto movimenta o mouse (um retângulo pontilhado mostra o tamanho e a forma do objeto). Ao soltar o botão o objeto será posicionado dentro da área especificada.

Insira dois objetos quaisquer na tela (por exemplo, dois botões), de modo a verificar e utilizar as dicas abaixo:

### ***Copiando objetos de tela***

Podemos copiar objetos de tela pressionando a tecla **[Ctrl]** enquanto arrastamos o objeto que desejamos copiar. Esta ação irá criar um novo objeto de tela que terá as mesmas propriedades que o objeto copiado. Pode-se ainda utilizar os comandos tradicionais Copiar **[Ctrl+C]** e Colar **[Ctrl+V]** da interface do Windows.

### ***Sobreposição de objetos***

Se você possui dois objetos na tela, e parte de um precisa estar sob ou sobre o outro, você pode ajustar a disposição através dos botões **Trazer para Frente**  e **Levar para o Fundo** .

### ***Ordem de navegação entre objetos com entrada de teclado e mouse***

Ao inserir uma série de objetos na tela, é possível (em execução), através do uso da tecla **[Tab]** se deslocar de um objeto para outro. Inicialmente o deslocamento por tabs segue a ordem de criação dos objetos em tela. Porém, é possível modificar esta ordem através desses passos:

- ➔ Selecione os objetos na ordem desejada.
- ➔ Aperte sobre o botão **Enviar para frente** ou **Trazer para o fundo**.
- ➔ Os objetos que forem trazidos para a frente, serão colocados em primeiro lugar na ordem de navegação e aqueles enviados para trás, no último lugar na fila.
- ➔ A edição de propriedades dos objetos na tela como alinhamento, tamanho, posição e agrupamento é feita através da barra de ferramentas **Arranjar** ou através do menu. O último objeto selecionado fica com o foco em vermelho para ser usado como referência. Para deselegionar um objeto use a combinação de teclas [Ctrl+Shift+BotãoEsqMouse].



Figura 32: Barra de ferramentas Arranjar

### **Selecionando todos os objetos da tela**

Para selecionar todos os objetos contidos em uma tela, basta pressionar as teclas [Ctrl+A].

### **Selecionando objetos na tela**

Para selecionar os objetos contidos em uma área, use a ferramenta de seleção . Com ela, você pode selecionar os objetos dentro de uma área delimitada pelo mouse. É possível selecionar objetos individualmente usando a combinação de teclas [Ctrl+BotãoEsqMouse].

Uso de teclas direcionais

Para mover um objeto no Elipse SCADA com o teclado, selecione-o e utilize as teclas direcionais:

- ↑ Sobe o objeto 1 ponto para cima
- ↓ Desce o objeto 1 ponto para baixo
- Move o objeto 1 ponto para a direita
- ← Move o objeto 1 ponto para a esquerda

Combinando as teclas direcionais com [Ctrl], você faz com que fique 10 vezes mais rápido o deslocamento do objeto. Combinando as teclas direcionais com [Shift], você pode redimensionar o objeto.

Alguns exemplos:

- Ctrl + ↑ Sobe o objeto 10 pontos
- Shift + ← Diminui em 1 ponto a largura do objeto;
- Ctrl + Shift + → Aumenta em 10 pontos a largura do objeto.

## 6.2. Propriedades dos Objetos de Tela

Inserido um objeto na tela, suas propriedades podem ser acessadas de diversas formas:

- Através de um duplo clique sobre o objeto
- Selecionando o objeto e utilizando a opção de menu objetos/propriedades
- Selecionando o objeto e clicando no botão  na barra de ferramentas
- Via organizer, onde se pode acessar a tela; ao clicar sobre o símbolo  são mostrados os objetos pertencentes àquela tela.

Todos os objetos possuem três páginas de propriedades em comum, como veremos a seguir.

### 6.2.1. Página Tamanho e Posição

Através desta aba pode-se ajustar algumas características de posicionamento do objeto, além de outras especificações genéricas.

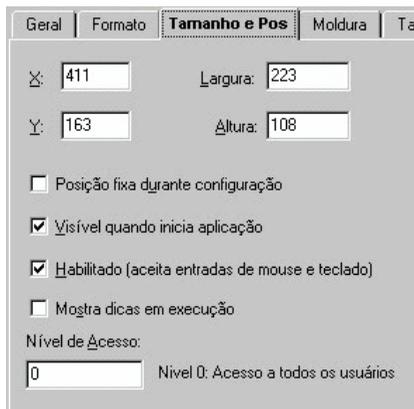


Figura 33: Propriedades de objetos de tela: Tamanho e Posição

**Propriedades de objetos de tela (tamanho e posição)**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
X	Define a coordenada X da posição do canto superior esquerdo do objeto. O ponto 0,0 é o canto superior esquerdo da tela.
Y	Define a coordenada Y da posição do canto superior esquerdo do objeto. O ponto 0,0 é o canto superior esquerdo da tela.
Largura	Determina a largura do objeto, em pixels.
Altura	Determina a altura do objeto, em pixels.
Posição fixa durante configuração	Impede que o objeto possa ser movido durante a configuração.
Visível quando inicia aplicação	Determina que o objeto seja visível no momento em que a aplicação iniciar.
Habilitado (aceita entradas de mouse ou teclado)	Habilita o acesso do teclado e mouse ao objeto (válido somente para aqueles objetos que permitem entradas via mouse ou teclado, como Setpoints e Sliders).
Mostra dicas em execução	Faz o objeto a mostrar uma "dica" quando o mouse está sobre ele. O texto da dica está no campo Descrição do objeto.
Nível de Acesso	Define o nível de acesso para o objeto.

## 6.2.2. Página de Moldura

Através da página de moldura podemos configurar estilos visuais para o objeto, como bordas, efeitos tridimensionais e títulos, dentre outros.



Figura 34: Página de Moldura

### Propriedades da Moldura

OPÇÃO	DESCRIÇÃO
Visível	Habilita ou desabilita uma moldura em volta do objeto.
Cor	Define a cor da moldura do objeto.
Título	Habilita ou desabilita um título na moldura do objeto.
Texto	Define o texto do título.
Fonte...	Define fonte, cor e tamanho da fonte do título.
Separador do título	Habilita ou desabilita uma linha separadora entre o título e o objeto.
Borda	Habilita ou desabilita a borda da moldura.
Espessura	Define a espessura da borda em pixels.
Cor	Define a cor da borda da moldura.
Efeito 3D	Seleciona um efeito 3D para dentro ou para fora para a moldura.
Espessura	Define a espessura em pixels para o efeito 3D.

### 6.2.3. Página de Tags

Através da página de tags podemos associar o objeto a uma ou mais variáveis, que podem ser tags ou atributos de um objeto qualquer. Na janela **Objetos** temos acesso aos objetos na árvore do Organizer, cujas propriedades aparecem na janela **Propriedades**. Os objetos que estão selecionados podem ser associados através de um clique no botão **Adicionar**. A operação mais comum, que é a associação de um tag, é feita selecionando-se o tag em questão e o adicionando à lista. Pode ser utilizado um procedimento semelhante para associar, ao invés do tag propriamente dito, seu nível de alarme ou seu tempo de scan ou qualquer outra propriedade que desejar.

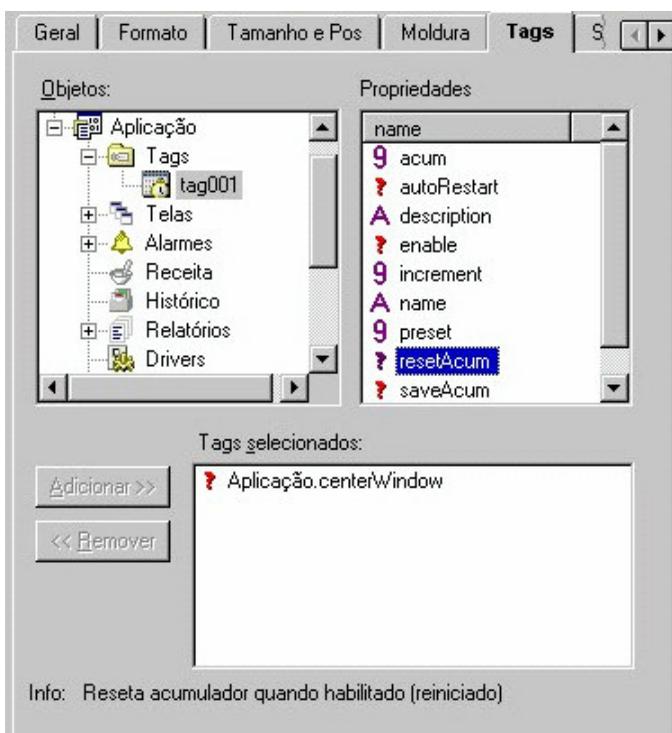


Figura 35: Propriedades dos Tags – Objetos de Tela

Propriedades de objetos de tela (tags associados)

OPÇÃO	DESCRIÇÃO
Objetos	Mostra a árvore da aplicação. Conforme o objeto selecionado, suas propriedades aparecerão na janela de Propriedades.
Propriedades	Permite a seleção de qualquer propriedade do objeto.
Tags selecionados	Lista os tags que estão associados ao objeto.
Adicionar	Adiciona os tags marcados à lista.
Remover	Remove os tags selecionados da lista.

## 6.3. Inserção de Objetos e Execução

Nosso objetivo nesta seção é listar métodos e comentários sobre a utilização dos objetos. Para conhecer as características específicas de cada um deles, consulte o capítulo referente ao assunto no Manual do Usuário.

Nos exercícios a seguir, para testar o comportamento dos objetos e da aplicação em si, você deverá **executar** a aplicação. Isso pode ser feito de duas maneiras:

1. Pressionar a tecla [F8] ou o ícone , que realiza monitoração de todas as telas que estiverem abertas
2. Pressionar a tecla [F10] ou o ícone , que realiza a execução total do aplicativo.

Para retornar ao modo de **configuração** (desenvolvimento da aplicação), basta pressionar a tecla [ESC], definida como padrão para parar a aplicação. Para alterar a tecla ou criar um novo método, é necessário modificar o script previamente associado à tecla [ESC]. Veremos este procedimento em capítulo posterior.

### 6.3.1. Utilização de Imagens

O Elipse SCADA permite a utilização de imagens nas telas das aplicações e em alguns objetos, como botões e animações. Estas imagens poderão estar no formato BMP (bitmap do Windows), JPEG ou GIF. Diversos programas no mercado permitem a edição de imagens nesses formatos, dentre os quais podemos citar o Adobe Photoshop, o Corel Draw! e o Autodesk AutoCAD.

No pacote do Elipse SCADA está incluída uma série de imagens e outros arquivos que podem ser utilizados sem restrição, para no desenvolvimento de suas aplicações. Elas se encontram no diretório **Lib**, dentro do diretório de instalação do sistema.

Para os exercícios deste tutorial, usaremos como imagem de fundo o arquivo **Fundo Modelo.bmp** que está no diretório **Elipse SCADA\Lib\Tutorial**.

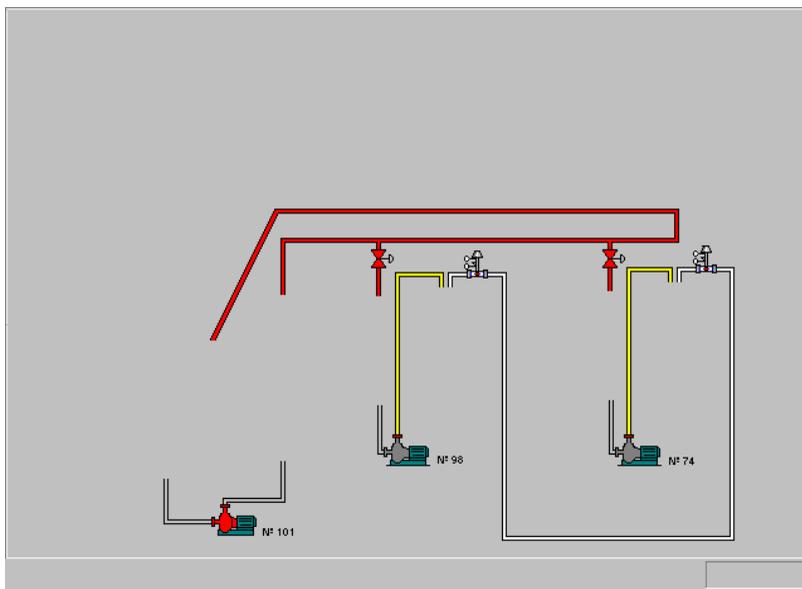


Figura 36: Fundo Aplicação-Exemplo

### 6.3.2. Fazendo animações

Outro recurso interessante é a possibilidade de criar animações a partir de um conjunto de imagens. O Elipse SCADA permite a seqüenciação de várias imagens para termos a sensação de movimento. Isso é muito útil para ilustrar diversos processos em uma aplicação, como por exemplo, a atividade de uma turbina ou peças andando em uma esteira.

Basicamente, para se fazer uma animação, devemos atribuir uma série de imagens para determinados intervalos de valores que um tag pode assumir. Estes intervalos são chamados de **Zonas**. Normalmente, utilizamos um tag demo para gerar os valores necessários a troca das imagens na animação automaticamente.

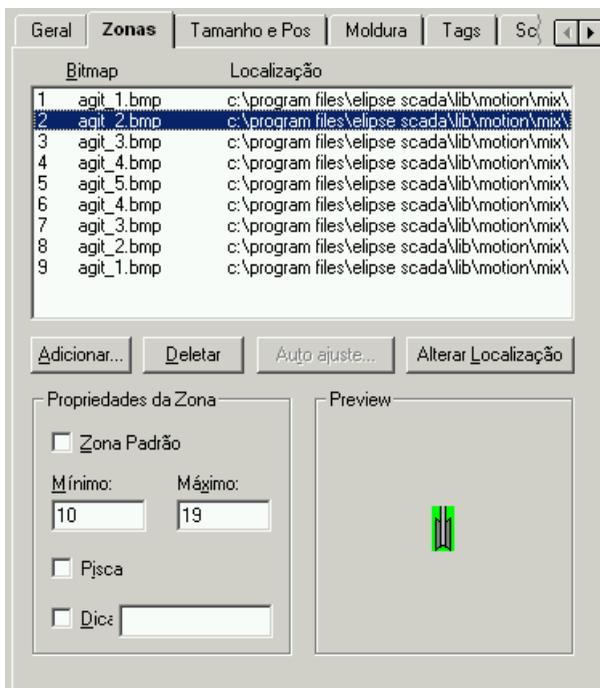


Figura 37: Propriedades das Animações - Zonas

# Exercícios

1. **Colocar reservatórios de abastecimento das matérias primas no canto esquerdo superior da tela de dosagem.**
  - ➔ Clique no ícone para inserir um objeto bitmap e marque a área na tela.
  - ➔ Clique duas vezes no objeto para chamar as propriedades.
  - ➔ No campo **Nome do Bitmap**, clique em **Localizar** e escolha o arquivo **funil2.bmp** no diretório **Lib\Hidraulic\Tanks**.
  - ➔ Clique em **Tamanho Original** para que o objeto se ajuste ao tamanho correto da imagem. Marque agora a opção **Transparente** e escolha como fundo a cor cinza.
  - ➔ Após a colocação de um reservatório, pode-se copiá-lo três vezes. Para isso, basta selecionar o objeto e arrastá-lo, pressionando juntamente a tecla [Ctrl] e soltando-o no local desejado.
2. **Colocar números de identificação dos reservatórios de matéria-prima.**
  - ➔ Escolha o ícone do objeto texto e selecione uma área na tela.
  - ➔ Para que fique sobreposto ao desenho do tanque, basta colocá-lo na região do tanque e trazê-lo para a frente, através do menu **Arranjar/Trazer para a Frente**.
  - ➔ Clique duas vezes no objeto para chamar as propriedades.
  - ➔ Selecione a guia **Zonas**.
  - ➔ Clique em **Adicionar**, para criar uma nova zona de mensagem.
  - ➔ Digite “1” no campo **Mensagem**, marcando a opção **Zona Padrão**.
  - ➔ Repita o processo para os outros reservatórios.
3. **Colocar o funil de mistura das matérias primas.**
  - ➔ Repetir o procedimento de inserção do objeto bitmap, escolhendo o arquivo **funil.bmp**, configurando a cor de fundo para **cinza claro**.
  - ➔ Insira um objeto **Texto** em cima do bitmap. Na opção **Zonas**, adicione a mensagem “Tank 01 - Misturador” e na aba **Moldura**, desabilite a opção **Visível**.
4. **Colocar o reservatório da mistura das matérias primas, no canto esquerdo inferior da tela de dosagem.**
  - ➔ Repetir o procedimento de inserção do objeto bitmap, escolhendo o arquivo **sil06.bmp**.

- ➔ Insira um objeto **Texto** em cima do bitmap. Na opção **Zonas**, adicione a mensagem “Tank 02 - Estocagem” e na aba **Moldura**, desabilite a opção **Visível**.
- 5. **Colocar o reservatório intermediário para transferência da mistura para os cozinhadore.**
  - ➔ Repetir o procedimento de inserção do objeto bitmap, escolhendo o arquivo **sil05.bmp**.
- 6. **Próximo aos motores 98 e 74, no lado esquerdo superior, inserir os condensadore.**
  - ➔ Repetir o procedimento inserindo o bitmap **condens.bmp**.
- 7. **Inserir os silos ao lado direito superior dos mesmos motore.**
  - ➔ Repetir o procedimento inserindo o arquivo **sil04.bmp**.
- 8. **Inserir a válvula de transferência de material do funil para o reservatório, para controle manual via mouse.**
  - ➔ Inserir um objeto botão no local mencionado.
  - ➔ Acesse as propriedades do objeto, clicando duas vezes.
  - ➔ Marque em **Funcionalidade**: “Liga/Desliga”.
  - ➔ Em **Botões**, aperte o tipo “Bitmap” (com o desenho de polígonos colorido).
  - ➔ No campo **Mensagens**, escolher para o estado **Normal** o arquivo **v\_vertical\_off.bmp** e para o estado **Pressionado** o arquivo **v\_vertical\_on.bmp**. Ambos arquivos estão no diretório **Lib\Hidraulic\Valv**.
- 9. **Inserir um botão para controle manual da agitação de material no funil.**
  - ➔ Inserir um botão no lado esquerdo central na tela, próximo ao funil.
  - ➔ Em **Funcionalidade**, marcar **Liga/Desliga**; em **Botões**, tipo **Mensagens de Texto** (primeira opção).
  - ➔ Na aba **Mensagem**, escreva para o estado normal o texto “Off” com fonte **Arial**, tamanho 9, cor branca.
  - ➔ Para o estado **Pressionado** coloque o texto “On”, cor de fundo azul escuro com a mesma fonte.
  - ➔ Na aba **Moldura**, marque **Visível** e no texto do título, escreva “Agitação”.
  - ➔ Na aba **Tags** adicione a propriedade **Mix.Enabled** do tag **Mix**.
- 10. **Inserir uma animação representando a agitação de material.**
  - ➔ Escolha o objeto tipo animação e coloque em qualquer lugar da tela.

- ➔ Na página Zonas, adicione os arquivos `agit_1.bmp`, `agit_2.bmp`, `agit_3.bmp`, `agit_4.bmp` e `agit_5.bmp` em uma seqüência crescente e depois de `agit_4.bmp` de volta a `agit_1.bmp`, em uma seqüência decrescente, totalizando 9 zonas diferentes. Os arquivos estão localizados no diretório `Lib\Motion\Mix`.
- ➔ Selecione agora todas as zonas (arraste com o mouse) e clique no botão *Auto Ajuste*, informando de 0 a 9 como limites. Agora cada zona está associada a uma faixa de valores do tag que será associado.
- ➔ Marque a Zona 1 como Zona Padrão.
- ➔ Na página Tags, adicione um tag de nome “Mix”.
- ➔ Na página Geral, faça os seguintes ajustes: marque Transparente; em Fundo, escolha a cor verde-limão; clique no botão Ajustar Tamanho.
- ➔ Leve a animação até o funil e clique no botão Trazer para Frente, para posicionar a animação em cima da imagem.

#### 11. Inserir a visualização da válvula do condensador através de animação.

- ➔ Sobre cada um dos condensadores, inserir um objeto de animação.
- ➔ Na página Zonas, insira duas imagens: `valv_off.bmp`, marcando como Zona Padrão e `valv_on.bmp`, marcado com mínimo 1 e máximo 1. Os arquivos estão localizados no diretório `Lib\Hidraulic\Valv`.
- ➔ Na página Geral, clique em Ajustar imagem.
- ➔ Na página Tags, associe cada um deles a um bit do tag DO. Assim, quando a saída digital associada a este bit se encontrar ligada, a animação mostrará a válvula acionada.

#### 12. Criar animações sobre os motores, de modo a monitorar sua operação.

- ➔ Sobre cada um dos motores, inserir um objeto tipo animação.
- ➔ Na página Zonas, escolha o arquivo `m&pumpoff.bmp` como zona Padrão e o arquivo `m&pumpon.bmp` com valor mínimo e máximo 1. Os arquivos se encontram em `Lib\Hidraulic\Motors&Pumps`.
- ➔ Clique no botão Ajuste imagem.
- ➔ Na página Tags, associe agora cada uma das três animações os três primeiros bits do tag DI.

#### 13. Criar botões de controle para as válvulas de saída.

- ➔ Repetir os procedimento anteriores, escolhendo na página Mensagens o bitmap `horizon_contr.bmp` para o quadro Normal e `horizon_contr_on.bmp` no quadro Pressionado. Os arquivos se encontram em `Lib\Hidraulic\Motors&Pumps`.

- ➔ Escolha na página de mensagens o valor 0 para Normal e 1 para Pressionado.
- ➔ Na página Tags, associe para cada uma das válvulas, um outro bit do tag DI.

**14. Inserir um objeto texto que irá indicar se os motores estão ligados ou desligados, a partir de três bits do tag DI.**

- ➔ Embaixo do terceiro motor (nº. 74) insira um objeto texto e desabilite sua moldura.
- ➔ Na aba Zonas, adicione zonas de mensagens de acordo com o que segue:
  - **Zona1:** Mensagem “Motores Desligados”, Zona Padrão, cor de fundo amarela, cor da fonte vermelha;
  - **Zona2:** Mensagem “Motor 3 Ligado”, valor mínimo 1 e valor máximo 1, cor de fundo preta, cor da fonte verde;
  - **Zona3:** Mensagem “Motor 2 Ligado”, valor mínimo 2 e valor máximo 2, cor de fundo preta e cor da fonte vermelha.
  - **Zona4:** Mensagem: “Motores 2 e 3 Ligados”, valor mínimo 3 e valor máximo 3, cor de fundo preta e cor da fonte azul.
  - **Zona5:** Mensagem: “Motor 1 Ligado”, valor mínimo 4 e valor máximo 4, cor de fundo preta e cor da fonte amarela.
  - **Zona6:** Mensagem: “Motores 1 e 3 Ligados”, valor mínimo 5 e valor máximo 5, cor de fundo preta e cor da fonte laranja.
  - **Zona7:** Mensagem: “Motores 1 e 2 Ligados”, valor mínimo 6 e valor máximo 6, cor de fundo preta e cor da fonte rosa.
  - **Zona8:** Mensagem: “Todos os motores ligados”, valor mínimo 7 e valor máximo 7, cor de fundo preta e cor da fonte branca.
- ➔ Na aba Tags, associe o objeto ao tag expressão **Status**.
- ➔ Supondo que os bits que compõem o tag **Status** indiquem que o motor está ligado ou desligado, o objeto texto mostrará vários tipos de mensagens, de acordo com o valor recebido:

BIT1	BIT2	BIT3	STATUS	TEXTO
0	0	0	0	Motores Desligados
0	0	1	1	Motor 3 Ligado
0	1	0	2	Motor 2 Ligado
0	1	1	3	Motores 2 e 3 Ligados
1	0	0	4	Motor 1 Ligado
1	0	1	5	Motores 1 e 3 Ligados
1	1	0	6	Motores 1 e 2 Ligados
1	1	1	7	Motores Ligados

#### 15. Inserir um display com o código do produto que está sendo processado.

- ➔ Logo acima dos silos de abastecimento de matéria prima, inserir um objeto display.
- ➔ Na aba Geral, desabilitar a moldura e escolher a fonte “MSSansSerif Regular”, tamanho 10.
- ➔ Na página Formato, marque Texto e no campo Prefixo, digite “Produto”.
- ➔ Na página Tags, associe o tag `codigo`. O valor de `codigo` será tratado em outro exemplo adiante.

#### 16. Inserir um relógio na tela principal.

- ➔ Insira um objeto display no canto inferior direito da tela principal.
- ➔ Escolha a fonte “Arial Regular”, tamanho 9.
- ➔ Em Moldura, desmarque a opção Visível.
- ➔ Na página de Tags, selecione o item Gerenciador Global. Escolha a propriedade `currentTime`.
- ➔ Na página Formato, marque Data/Hora e pelo botão Formato, selecione o formato “hh:mm”. No exemplo da janela, equivale ao “17:30”.

#### 17. Inserir um display para as temperaturas, ao lado dos cozinhadores e do silo de matéria-prima.

- ➔ Insira um objeto display para cada cozinhadores e para o silo.
- ➔ Em Moldura, desmarque a opção Visível.
- ➔ Coloque fonte “MsSansSerif Regular”, tamanho 8, cor branca.
- ➔ No fundo, escolha a cor azul;
- ➔ Em Formato, escolha numérico, tamanho 3, precisão 0. No campo Sufixo, digite “°C”.
- ➔ Na página Tags, associe o tag de temperatura apropriado.

- ➔ Faça o mesmo procedimento para todos os objetos display criados.

#### 18. Criar botões para navegação entre as telas.

- ➔ Criar um botão, que será inserido na parte inferior da tela de dosagem.
- ➔ Na aba **Geral**, escolha a funcionalidade do tipo momentâneo.
- ➔ Associe o botão à tecla F1 (configurada no campo **Tecla de Função**).
- ➔ No campo **Ir Para Tela**, escolha a tela **Abertura**.
- ➔ Na aba **Mensagens**, escolha a fonte “Arial Regular”, tamanho 9, cor amarela com cor de fundo verde-escuro. Digite o texto “F1 – Abertura” para as duas mensagens (em estado normal e pressionado).
- ➔ Através das ferramentas de cópia, copiar este botão mais cinco vezes, colocando os demais lado-a-lado. Os novos botões devem ter a mesma funcionalidade, porém levando as outras telas. Para os novos botões, escolha os textos: “F2 – Alarmes”, “F3 – Tendência”, “F4 – Receitas”, “F5- Histórico”, “F6-Batelada”, “F7- Relatório”, “F8- Receita” e “F9- DB”.

#### 19. Inserir um quadro de alarmes.

- ➔ No canto superior direito da tela de dosagem, inserir um objeto Alarmes.
- ➔ Marcar no tipo de alarme: **Resumido**.
- ➔ Em **Formato da Mensagem**, marcar as opções de **Data**, **Hora**, **Tipo de Alarme**, **Comentário** (tamanho 20) e **Valor** (tamanho padrão).

#### 20. Inserir um gráfico de barras para mostrar o nível dos cozinhadores e do silo de estocagem.

- ➔ Num espaço qualquer da tela, inserir um objeto Gráfico de Barras (*Bar Graph*).
- ➔ Na página **Geral**, marcar a faixa de valores de 0 a 100, orientação de baixo para cima e espaçamento 0.
- ➔ Desabilite a régua e a moldura.
- ➔ Na página **Tags**, associe ao tag **Tank01**.
- ➔ Posicione o objeto sobre o cozinhador e escolha a opção **Trazer para a Frente**.
- ➔ Repita o procedimento para o outro cozinhador, associando o tag **Tank02** e para o silo de estocagem de matéria prima, com o tag **Tank03**.

#### 21. Inserir um título na Tela de Abertura.

- ➔ Na tela de abertura, inserir um objeto texto, configurando uma zona de mensagens.
- ➔ Marcá-la como padrão, com cor de fundo vinho e fonte “Arial Negrito”, tamanho 20 e cor amarela.

- ➔ Digite como texto do objeto: “Aplicação de exemplo – Fábrica de Balas”.
- 22. Inserir uma barra de suporte para ferramentas.**
- ➔ Inserir outro objeto texto, de forma que ocupe toda a extensão inferior da tela.
  - ➔ Crie uma zona de mensagem, marcada como padrão. Não digite nenhuma mensagem.
  - ➔ Na página de moldura, desmarque o título e borda.
  - ➔ Na opção Efeito 3D, marque para dentro, com tamanho 4.
- 23. Inserir um gráfico de tendências na Tela de Tendências.**
- ➔ Insira um objeto tendência na Tela de Tendências.
  - ➔ Na página **Geral**, seção tipo de gráfico, marque **Tempo x Dado** e defina o intervalo de 10 segundos.
  - ➔ Na página **Avançado**, marque **Tempo-real**, somente quando a tendência está no topo.
  - ➔ Na página **Gráfico**, digite para o eixo Y os limites de 0 (inferior) a 250 (superior).
  - ➔ Para associar tags ao objeto Tendência, selecione a aba **Penas**. Associe as penas aos tags de temperatura. Configure o gráfico de acordo com as especificações do instrutor. Recomenda-se utilizar as cores vermelho, amarelo e azul para a criação das penas.
  - ➔ Repita o procedimento de inserção de penas para os tags de nível.
- 24. Inserir um alarme histórico na tela de Alarmes.**
- ➔ Insira o objeto de Alarmes na tela reservada para o mesmo.
  - ➔ Marcar tipo Histórico, com opções de Data, Hora, Tipo de Alarme, Comentário (tamanho 20) e Valor.
- 25. Inserir níveis de alarme no objeto de tendência, através do uso de marcas.**
- ➔ Na tela de tendência, selecione as propriedades da tendência.
  - ➔ Na página **Geral**, clique em **Adicionar Marca**, selecione **Linha Horizontal** e formate-a como uma linha tracejada. Na página de tags associe ao nível de alarme alto do tag **Temperatura01**, **Temperatura01.High.Limit**. Repita o procedimento para o alarme baixo com propriedade **Temperatura01.Low.Limit**.
- 26. Inserir na tela de tendência, dois botões deslizantes para modificar os níveis de alarme.**







Os **Scripts** são pequenos procedimentos escritos em linguagem de programação própria – Eclipse Basic – que permitem uma maior flexibilidade na sua aplicação. O scripts são sempre associados a eventos, isto é, eles são iniciados no momento da ocorrência deste evento.

## 7.1. Considerações Gerais

---

Em qualquer linguagem de programação, é necessária a criação de métodos, de modo a especificar e ordenar a execução das instruções desejadas. A própria estrutura dos scripts do Eclipse scada já organiza de certa maneira esta ordem, pois são orientados à **eventos**.

Os eventos são ocorrências relacionadas a um objeto, que podem ser tratadas de modo a se realizar uma ação específica. Eles podem ser **físicos**, como por exemplo, alguma ação no teclado ou no mouse. Em cada caso, temos diversas informações relevantes como a tecla pressionada ou a posição do cursor e o status dos botões. Os eventos podem ser **internos**, como a mudança do valor de uma variável. Estes eventos podem também ter associações físicas, como a mudança de uma temperatura de uma câmara de 10 para 11 graus quando temos um tag que recebe os valores dessa temperatura. O Eclipse SCADA já tem diversos eventos pré-definidos disponíveis para a ligação ou associação de scripts. Exemplos de alguns desses eventos são listados a seguir.

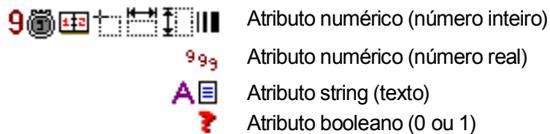
### Eventos comuns em aplicações SCADA

<b>EVENTO</b>	<b>DESCRIÇÃO</b>
OnKeyPress	Quando uma tecla é pressionada
OnKeyRelease	Quando uma tecla é liberada
OnSetFocus	Quando um objeto recebe o foco de edição
OnLButtonDown	Quando o botão esquerdo é pressionado sobre um objeto
OnRelease	Quando um objeto botão é liberado
WhileRunning	Enquanto uma aplicação está executando
OnAlarm	Ocorrência de qualquer tipo de alarme

A linguagem utilizada nos módulos de script, o **Elipse Basic**, é bastante similar às linguagens de programação, porém com recursos visuais como os encontrados no Visual Basic.

Algumas características da linguagem:

- *Não é necessário a declaração de variáveis ou funções no início do script.* As variáveis devem ser tags, objetos ou atributos previamente criadas ou importadas de outras aplicações. O Elipse SCADA já possui algumas variáveis de sistema pré-definidas.
- *O tipo de dado que se atribui a um tag é livre.* O valor suportado pode ser desde um inteiro de 8 bits até um tipo real de 64 bits ou ainda um string (texto). Em comunicação de dados com equipamentos externos, a conversão é feita automaticamente, de acordo com os tipos suportados pelo equipamento. No caso de propriedades, um ícone ao lado de cada uma (visualizado através do AppBrowser) indica o tipo de dado suportado:



As **variáveis** e **constantes** são os objetos básicos manipulados num script. Os **operadores** especificam o que será realizado com os mesmos. As **expressões** combinam variáveis e constantes para produzir novos valores.

Para facilitar a edição de scripts ou de tags expressão podem ser usadas as ferramentas AppBrowser e Referência Cruzada (X-Reference).

## 7.2. AppBrowser e Referência Cruzada

---

O **AppBrowser** permite navegar facilmente pela aplicação. Quando você seleciona um objeto na árvore ao lado esquerdo da janela, seus atributos e funções correspondentes são listados à direita.

Você pode usar o AppBrowser como referência durante a edição de um script. Uma característica bastante útil é a possibilidade de selecionar um objeto, atributo ou função que você deseja utilizar e copiar diretamente para o script pressionando o botão Copia no Script ->.

A ferramenta **Referência Cruzada** possui a mesma estrutura do AppBrowser com a diferença que quando você seleciona um objeto na árvore ao lado esquerdo da janela, suas respectivas referências é que são listadas à direita. Dê um duplo-clique sobre uma referência para ir ao objeto referido.

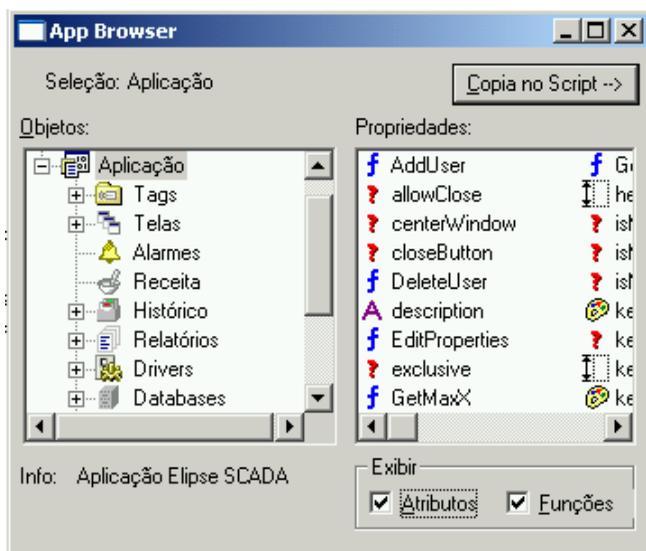


Figura 38: App Browser

### Opções disponíveis no AppBrowser

OPÇÃO	DESCRIÇÃO
Seleção	Mostra o nome do objeto, atributo e função selecionado, da mesma forma que será copiado para o script.
Objetos	Lista dos objetos em ordem hierárquica.
Info	Mostra uma descrição do item selecionado.
Propriedades	Lista dos atributos e funções do objeto selecionado.
Exibir	Permite filtrar a informação (atributos ou funções) listada na janela de propriedades.
Copia no Script -->	Copia o texto mostrado no campo Seleção para o script.

## 7.3. Operadores e Constantes

Listamos as constantes (com suas notações) e operadores que podem ser utilizados nos scripts.

### Constantes

TIPOS	EXEMPLOS
Inteiros (32 bits, dec)	1234, 1234d, -993
Inteiros (32 bits, bin)	11001110b (não permite sinal)
Inteiros (32 bits, octal)	7733o (não permite sinal)
Inteiros (32 bits, hex)	0A100h, 3B8h (não permite sinal) (se o primeiro dígito é A-F, coloque um zero na frente)
Números reais (64 bits)	133.443, 344.939 (não tem notação científica)
Strings	"Temperatura", "pressão"

### Operadores aritméticos

OPERADOR	EXEMPLO
+ (adição)	tag001 + 34
+ (concatenação de strings)	slider1.Frame.title + "<- PLC1"
- (subtração)	tag001 - screen1.x
* (multiplicação)	screen1.width * 3.141592
/ (divisão)	tag001 / tag002
% (resto da divisão)	tag001 % tag002
** (exponenciação)	tag001 ** 2 (tag001 ao quadrado)

### Operadores lógicos

OPERADOR	EXEMPLO
& (E bit-a-bit)	flags & 0F000h
(OU bit-a-bit)	flags   0F000h
^ (OU exclusivo bit-a-bit)	flags ^ 0F000h
~ (NÃO bit-a-bit)	~flags
<< (SHIFT à esquerda)	flags << 2 (desloca o valor de "flags" dois bits para a esquerda)
>> (SHIFT à direita)	flags >> 2 (desloca o valor de "flags" dois bits para a direita)
AND (E lógico)	tagOnOff AND (tag001 > 200)
OR (OU lógico)	tagOnOff OR (tag001 > 200)
XOR (OU exclusivo)	tagOnOff XOR (tag001 > 200)
NOT (negação)	NOT tagOnOff

**Operadores relacionais (retornam 1 para verdadeiro e 0 para falso)**

OPERADOR	EXEMPLO
= (é igual a)	tag001 = 8
> (é maior que)	tag001 > tag002
< (é menor que)	tag001 < tag002
>= (é maior ou igual que)	tag001 >= -1
<= (é menor ou igual que)	tag001 <= 16
<> (é diferente que)	tag001 <> 1

**Operadores especiais de atribuição (aritméticos)**

OPERADOR	EXEMPLO
+= (soma)	tag001 += 6 (ou seja, tag001 = tag001 + 6)
-= (subtração)	tag001 -= 3.246 (tag001 = tag001 - 3.246)
*= (multiplicação)	tag001 *= 4 ( tag001 = tag001 * 4)
/= (divisão)	tag001 /= 12 (tag001 = tag001 / 20)
**= (exponenciação)	tag001 **= 2 (tag001 = tag001 ** 2)

**Operadores especiais de atribuição (bit-a-bit)**

OPERADOR	EXEMPLO
&= (AND)	tag001 &= 00000000h
= (OR)	tag001  = 11111111h
^= (XOR)	tag001 ^= 10100100h
<<= (SHIFT à esquerda)	tag001 <<= 2 (tag001 = tag001 << 2)
>>= (SHIFT à direita)	tag001 >>= 2 (tag001 = tag001 >> 2)

**Precedência de operadores (Ordem de execução)**

A tabela abaixo lista as regras para precedência e associação de todos os operadores. Aqueles listados na mesma linha possuem a mesma precedência.

1. + - ~ NOT (operadores unários)
2. \*\*
3. \* / %
4. + -
5. >> <<
6. > >= < <=
7. == <>

8. &
9. ^
10. |
11. AND
12. XOR
13. OR
14. = += -= \*= /= %= &= |= ^= \*\*= <<= >>=

## 7.4. Controle de Fluxo

---

A fim de controlar a ordem na qual as instruções são processados o Elipse Basic disponibiliza uma série de comandos para fazer desvios e condições. Estes comandos são tratados a seguir.

### 7.4.1. Comando If...Else...Elseif...EndIf

Permite a tomada de decisões durante a execução de um script.

Sintaxe:

```
If <condição1>  
    <bloco de instruções 1>  
Else  
    <bloco de instruções n>  
EndIf
```

```
If <condição1>  
    <bloco de instruções 1>  
ElseIf <condição2>  
    <bloco de instruções 2>  
Else  
    <bloco de instruções n>  
EndIf
```

### 7.4.2. Comando For...Next

Repete um bloco de instruções um determinado número de vezes.

Sintaxe:

```
For <contador> = <início> To <fim>  
    <bloco de instruções>  
Next
```

### 7.4.3. Comando While...Wend

Executa um bloco de instruções enquanto uma determinada condição é verdadeira.

Sintaxe:

```
While <condição>  
    <bloco de instruções>  
Wend
```

### 7.4.4. Comando Repeat

Executa um bloco de instruções até que determinada condição seja verdadeira.

Sintaxe:

```
Repeat  
    <bloco de instruções>  
Until <condição>
```

Para saber mais a respeito da sintaxe e permissões de uso dos laços de controle, consulte o manual do usuário. Nosso objetivo aqui é fornecer exemplos e comentários sobre a performance na sua utilização.

## 7.5. Funções Especiais

---

O Eclipse SCADA possui uma série de funções especiais pré-definidas que auxiliam na edição de scripts, facilitando a execução de tarefas mais complexas e permitindo uma melhor configuração do seu sistema.

Através da ferramenta AppBrowser podemos ver as diversas funções especiais disponíveis para cada objeto durante a edição de scripts.

Destacamos o objeto Gerenciador Global, que traz funções de utilidade geral, como funções de datas e do relógio de tempo-real, manipulação de strings e conversões numéricas, arquivos, multimídia e outras.

## 7.6. Dicas de Otimização

---

### **Edição de scripts**

Para construir um script, além de utilizar o AppBrowser, você pode editar as linhas livremente como num editor de textos qualquer do Windows. Dessa maneira, as operações padrão como Recortar [Ctrl+X], Copiar [Ctrl+C], Colar [Ctrl+V], e Desfazer [Ctrl+Z] podem ser utilizadas.

Além disso, podem ser usadas as ferramentas de Procurar (*Find*) e Substituir (*Replace*) presentes em qualquer script.

### **Compilação de scripts**

Sempre antes de executar uma aplicação devemos verificar se esta não contém erros. Isso é uma tarefa muito importante, uma vez que os scripts que contiverem erros de sintaxe (por exemplo, nomes de propriedades ou objetos errados ou não existentes) não serão executados.

Utilizando os botões **Compilar**, **Compilar Script**, **Compilar Todos os Scripts** que estão na parte inferior do Organizer podemos fazer estas verificações de maneira rápida.

Utilize o botão **Compilar Scripts** para checar por erros no script que está sendo editado no momento. Este botão não irá verificar se os demais scripts da aplicação possuem erros.

O botão **Compilar** verifica somente os scripts que ainda não foram compilados. O botão **Compilar Todos os Scripts** verifica todos os scripts da aplicação sem levar em conta se foram modificados ou não. A diferença de tempo que esta operação leva para ser feita em relação a **Compilar** é sensivelmente maior para aplicações grandes, mas é bastante útil para evitar erros de execução. Quando utilizar estas duas últimas opções, aparecerá uma janela, indicando em vermelho as linhas de scripts que possuem erros. Via um duplo clique na linha vermelha, o script que contém o erro é automaticamente editado.

Fazendo a compilação de scripts evitamos erros comuns:

- a) Você está atribuindo parâmetros de tipos incompatíveis (exemplo: forçando uma string para um atributo digital);
- b) Houve uma divisão por zero;
- c) O script possui erros de sintaxe ou semântica (significado).

## Seqüenciamento entre scripts

Como os scripts são orientados a eventos, há a possibilidade da execução de scripts dentro de outros, devido ao regime de seqüenciamento de operações utilizado. Seja o seguinte exemplo:

Script tag001.OnAlarmHigh

```
tag002 = 1
```

```
tag003 = 0
```

Script tag002.OnValueChanged

```
Screen1.Activate()
```

Neste caso, no momento em que **tag001** entra em condição de alarme (que foi previamente definido pelo usuário) o primeiro script atribui o valor “1” para **tag002**. Ao realizar tal tarefa, o script de **tag001** é interrompido, de modo a verificar nas implicações da atribuição do valor à **tag002**. Quando isto ocorre, em linhas gerais é verificado se **tag002** possui algum script associado (de modo que é necessário executá-los também, pois ocorreu um evento com **tag002**) ou se há algum objeto de tela ligado ao mesmo.

Logo, o script OnValueChanged de **tag002** é executado, e somente então se retorna ao ponto original, de onde se tinha parado, no script de **tag001**.

## Comentários

Comentários nos scripts podem ser inseridos com duas barras simples colocados à esquerda, a partir do ponto onde se deseja comentar. Exemplo:

```
Tag001 = 25 // Isto é um comentário
```

Variáveis internas

Você pode criar variáveis internamente no script. Ao final da execução, a mesma será destruída. O comando utilizado para a criação de variáveis locais é DIM. Exemplo:

```
DIM Flag
```

```
FOR Flag = 0 To 10
```

```
tag001 = Flag
```

```
NEXT
```

No exemplo, criamos uma variável `Flag` para ser o contador do laço FOR...NEXT.

### ***Retorno de script***

Caso queira sair de um script antes de seu final, use o comando RETURN.

Exemplo:

```
IF tag001 > 10
RETURN
ENDIF
tag002 = 10
```

No exemplo acima, se **tag001** é maior que 10 o script é abandonado; caso contrário, sua execução prossegue normalmente.

### ***Restrições no uso de laços infinitos***

Os laços de controle como While e Repeat, se usados de modo a gerar laços infinitos, podem interromper a execução das outras tarefas do software. Seja o seguinte exemplo:

```
WHILE tag001
    tag002 += tag003
WEND
```

No script acima, a variável **tag001** é avaliada; caso verdadeira, isto é, se diferente de zero, leva à execução da primeira instrução, que incrementa **tag002** do valor de **tag003**. Logo após, **tag001** é avaliado novamente e a execução prossegue sem a possibilidade da recepção da resposta do driver de comunicação com o valor de **tag001**. A fim de eliminar o problema, uma das soluções seria forçar uma leitura de **tag001** ao final de cada loop, como no exemplo:

```
WHILE tag001
    tag002 += tag003
    tag001.Read()
WEND
```

### ***Limitações em scripts de execução por tempo***

Os scripts **WhileRunning** geralmente gastam mais tempo de processamento que outros scripts. Através da função **ScriptWindow()**, você pode verificar em execução quanto tempo cada script leva para ser executado. Logo, deve-se garantir que um script **WhileRunning** termine de ser executado antes de ser chamado novamente, a fim de evitar um acúmulo de pedidos de execução.

### **Scripts muito longos**

Evite a criação de scripts muito longos. Dividindo o script em outros menores, o tempo gasto diminui consideravelmente. Utilize também as funções de depuração para otimizar os scripts.

Uma sugestão de separação entre scripts poderia ser como a seguir.

Crie um tag tipo RAM, chamado, por exemplo, “FunctionA”. Em determinado momento, no script que deseja separar, faça uma atribuição a **FunctionA**.

```
Script Application.WhileRunning
...
FunctionA = 1
```

Em **FunctionA** crie um script **OnValueChanged** e copie a parte do primeiro script, que será executada neste. Como última instrução, retorne o tag **FunctionA** para seu valor inicial.

```
Script FunctionA.OnValueChanged
...
FunctionA = 0
```

### **Limite máximo de scripts**

No Elipse SCADA, há um limite de 32Kb para o tamanho dos scripts, ou seja, em torno de 32 mil caracteres, incluindo quebras de linha.

### **Scripts de botões**

Para os scripts dos botões, dê preferência ao evento **OnRelease** ao invés de **OnPress**, pois dessa forma o usuário percebe mais facilmente a ação.

# Exercícios

## 1. Substituir, no botão na tela de Dosagens, a chamada automática da tela de Alarmes por um script.

- ➔ Na página Geral, na lista Ir Para Tela, escolher nenhum.
- ➔ Na página de scripts, escolher OnRelease.
- ➔ Através do AppBrowser, procure a tela de alarmes, escolhendo no canto direito inferior suas funções.
- ➔ Escolha a função `Activate()`. Copie para o script.
- ➔ Compile o script. No resultado, deverá aparecer:  
`Alarmes.Activate()`
- ➔ Execute a aplicação, testando a funcionalidade.

## 2. Fazer a aplicação trocar para tela de Alarmes na ocorrência de um alarme específico.

- ➔ Em uma das variáveis de nível ou temperatura criar um script `OnAlarmHigh`, executando a função de troca de tela para a tela de alarmes, através de execução da função `Activate()` da mesma.

## 3. Fazer um objeto trocar de cor na ocorrência de um alarme.

- ➔ No tag `Temperatura01` fazer através de um script `OnAlarmHigh` mudar a cor de um display na tela, através da alteração da propriedade `backgroundColor` e da ajuda da função RGB (r, g, b) presente no Gerenciador Global.

Exemplo:

```
Script Temperatura01.OnAlarmHigh
Dosagem.Display01.backgroundColor = RGB(255,0,0)
// seta vermelho para cor de fundo
```

```
Script Temperatura01.OnAlarmReturn
Dosagem.Display01.backgroundColor = RGB(0,0,255)
// retorna para azul
```

## 4. Fazer a aplicação trocar para tela de Alarmes na ocorrência de qualquer alarme.

- ➔ Fazer a mesma troca para a tela de Alarmes, utilizando o script `OnAlarm` do item alarmes no Organizer, de modo que a tela de alarmes seja chamada na ocorrência de qualquer alarme com prioridade 2.
- ➔ Selecionar o item Alarmes no Organizer, escolhendo a aba Scripts.

- ➔ Criar um novo script associado ao evento OnAlarm.

Exemplo:

```
Script OnAlarm
IF lastAlarmPri == 2
Alarmes.Activate()
ENDIF
```

### 5. Criar um ícone de login na tela de abertura, que muda seu desenho ao se passar o mouse sobre o mesmo.

- ➔ Insira um objeto bitmap sobre a barra de ferramentas criada, trazendo-o para a frente (sobre a barra).
- ➔ Escolha como bitmap o arquivo Lib\Util\Login2.bmp, tipo transparente e cor de fundo branca.
- ➔ Crie um script para receber a movimentação do mouse sobre ele.

```
Script OnMouseMove
Abertura.Bitmap1.SetMouseCapture()
IF Abertura.Bitmap1.IsMouseInside()
Abertura.Bitmap1.fileName="Lib\Util\login.bmp"
ELSE
Abertura.Bitmap1.fileName="Lib\Util\login2.bmp"
ReleaseMouseCapture()
ENDIF
```

- ➔ O script OnMouseMove é executado quando o mouse é movido para dentro ou fora da área do objeto. Já a função SetMouseCapture faz com que todas as mensagens do Windows geradas pelo mouse sejam enviadas para o objeto em questão. Desta maneira, pode-se testar se o ponteiro está dentro ou fora da área, de modo a trocarmos os desenhos.
- ➔ Execute a aplicação e ao passar o mouse sobre o bitmap, verá que o desenho muda de preto e branco para colorido.

### 6. Criar um botão na tela de Dosagem, que liga e desliga o modo automático e manual, desabilitando os botões de controle dos motores e válvulas.

- ➔ Insira um botão, à direita da área dos botões na tela de Dosagem.
- ➔ Escolha um botão do tipo Liga/Desliga.
- ➔ Na página de mensagens digite “Auto” para Normal e “Manual” para Pressionado.
- ➔ Criar dois scripts: OnPress e OnRelease. As instruções do primeiro servem para habilitar os objetos, quando escolher operação manual e o segundo para desabilitá-los, de acordo com o script:

```
Script OnPress
Dosagem.Botão1.enabled = 1
Dosagem.Botão2.enabled = 1

Script OnRelease
Dosagem.Botão1.enabled = 0
Dosagem.Botão2.enabled = 0
```

### 7. Criar um sinal sonoro ao entrar em alarme.

- ➔ Crie um script **OnAlarm** no item Alarmes do Organizer.
- ➔ Insira o comando **StartSound**, presente no Gerenciador Global.
- ➔ Este comando começa a tocar um índice sonoro em intervalos regulares.

### 8. Criar um botão para desligar o alarme.

- ➔ Insira um botão sobre o objeto de Alarmes, trazendo-o para a frente.
- ➔ Escolha a funcionalidade **Momentâneo**. Escolha o tipo **Bitmap**
- ➔ Utilize os arquivos **Calaron.bmp** e **Calaroff.bmp** que estão em **Lib\Button**.
- ➔ Crie um script **OnRelease** para o botão, executando a função **StopSound()**, presente no Gerenciador Global.





Uma **Receita** é um conjunto de valores pré-definidos que podem ser carregados para um grupo de *tags* a fim de configurar um processo específico. Esta lista de tags também chamamos de **modelo de receita**.

Por exemplo, seja uma máquina que fabrica diferentes tipos de parafusos. As variáveis envolvidas no processo são sempre as mesmas, mas seus valores provavelmente irão mudar dependendo do tipo de parafuso que se quer produzir. Supondo que você tem diferentes configurações de máquina para cada tipo de parafuso, estes valores poderiam ser gravados em uma receita e serem posteriormente carregados em *tags* de controle, facilitando a tarefa do operador e evitando erros.

Dessa maneira, podemos criar um modelo de receita “Parafuso” com diversas receitas “Fenda Philips”, “Fenda Torx”, “Fenda Simples” e assim por diante.

Para que sejam recuperados quando necessário, os modelos e os dados de uma receita são armazenados em disco, em um “arquivo de receitas” com a extensão .RCP.

## 8.1. Propriedades Gerais da Receita

---

Cada receita que você cria para a aplicação aparece abaixo da opção **Receitas** na árvore do Organizer. Ao selecionar uma receita específica, suas propriedades são mostradas ao lado direito da árvore.

A seguir, podemos ver as propriedades das receitas.

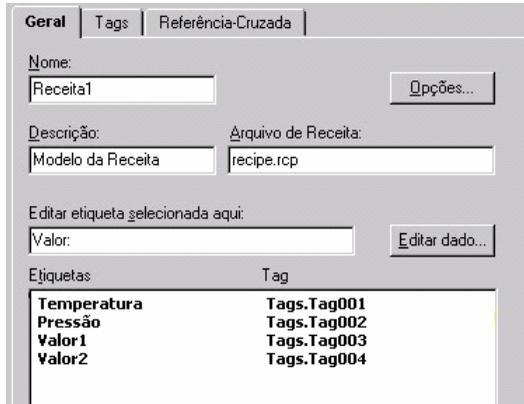


Figura 39: Propriedades da Receita

### Propriedades da Receita

OPÇÃO	DESCRIÇÃO
Nome	Define o nome do modelo de receita.
Descrição	Uma breve descrição sobre o modelo de receita.
Arquivo Receita	Define o nome do arquivo para o modelo de receita. O nome do arquivo pode ter até 8 caracteres e não deve conter extensão (o Eclipse SCADA irá sempre usar a extensão RCP). Você pode especificar também o caminho do arquivo, que poderá ser uma localização absoluta (“C:\ELIPSE\RECIPES\RCP1”) ou relativa (“RECIPES\RCP1”). Localizações relativas são recomendadas se você deseja copiar sua aplicação para outro computador.
Editar receita selecionada aqui	Permite a edição da etiqueta que identifica o tag selecionado na receita.
Editar Dado...	Abre a janela Editar Receita onde se pode acrescentar e modificar os valores das diversas receitas.
Etiquetas	Mostra os campos associados aos tags do modelo de receita.
Tag	Mostra os tags selecionados para o modelo de receita corrente.

## 8.2. Editando Receitas

Para acrescentar, editar ou apagar receitas já criadas, utilizamos a janela Editar Receita que é chamada através do botão **Editar dado...** na página de propriedades gerais de receitas. Todas as receitas criadas utilizando o modelo escolhido são listadas no campo **Receitas**, onde podem ser selecionados para edição. Selecionando qualquer receita da lista você poderá editar a sua descrição e os valores de cada tag associado.

Veremos os campos dessa janela a seguir.

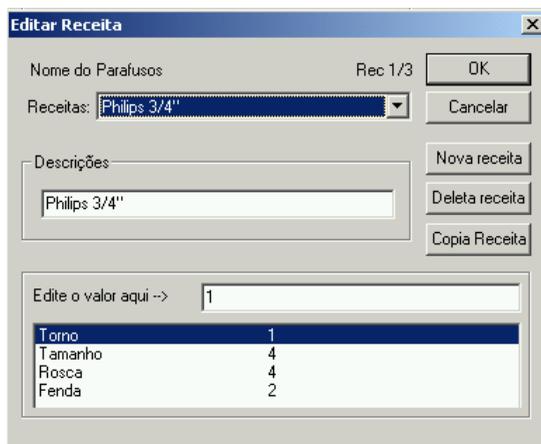


Figura 40: Editar Receita

### Opções disponíveis na janela Editar Receitas

OPÇÃO	DESCRIÇÃO
Receitas	Permite a seleção de uma receita no modelo corrente.
Descrição	Nome ou descrição da receita.
Nova receita	Cria uma nova receita. Cada modelo de receita pode ter várias receitas (conjunto de valores) relacionadas.
Deletar receita	Apaga a receita selecionada.
Copia receita	Permite copiar uma receita já definida, a fim de tornar mais fácil a criação de receitas com muitas variáveis.
Edite o valor aqui	Permite a edição do valor do tag para a receita corrente, que são listados no quadro inferior. Os tags são identificados por suas etiquetas associadas. Use as setas de direção do teclado ou o mouse para selecionar o tag a ser editado.

## 8.3. Dicas de otimização

---

### *Implementação de tags retentivos*

Esta dica mostra como implementar tags com valor persistente, isto é, os valores dos tags são gravados quando se sai da aplicação e são carregados quando se inicia a aplicação novamente.

Como as receitas são gravadas em disco e temos que associar seus componentes a tags em memória, podemos dispor delas para armazenar valores dos tags retentivos para utilizarmos em uma execução posterior.

Para tags retentivos, seguimos estes passos:

- ➔ Crie um novo modelo de receita e associe a ela os tags e propriedades que se deseja gravar.
- ➔ No script `OnStartRunning` da aplicação, adicione as seguintes linhas:

```
IF receita.GetRecCount() > 0
    receita.LoadRecipe(1)
ENDIF
```
- ➔ No script `OnStopRunning` da Aplicação ou ainda, via um outro procedimento qualquer no qual deseja garantir que os dados serão gravados, adicione as seguintes linhas:

```
IF receita.GetRecCount() < 1
    receita.CreateNewRecord("Tags persistentes")
ENDIF
receita.SaveRecipe(1)
```

Desta forma, sempre que a aplicação é terminada os valores dos tags são salvos no primeiro registro (`SaveRecipe(1)`) do arquivo de receitas. Quando a aplicação é rodada novamente, é verificado se existe algum registro gravado no arquivo. Se existir, o primeiro registro é carregado (`LoadRecipe(1)`).

### *Intercâmbio de receitas*

Outra característica interessante do modelo de receitas é que o mesmo arquivo de dados criado por uma receita ou modelo pode ser utilizado por outra receita, desde que possuam o mesmo número de variáveis associadas.

Isto permite carregar o arquivo de receitas em outro conjunto de variáveis que não as variáveis de campo, permitindo a preparação de fórmulas e etc. num processo à parte, para posteriormente serem utilizadas pelo operador.

# Exercícios

## 1. Criar um modelo de receita para cadastro de produtos.

- ➔ No item **Receitas**, criar uma nova receita, com o nome de “modelo1.rcp”.
- ➔ Especifique arquivo **modelo1.rcp**.
- ➔ Associe os tags **Água**, **Açúcar**, **Xarope** e **Glucose**.

## 2. Criar exemplos de receitas.

- ➔ Clicar no campo **Editar dado**, onde será aberta uma caixa de diálogo para o cadastro das receitas (conjunto de valores) que podem estar associados aos tags, além do nome de cada receita.

## 3. Criar na tela **Receitas**, setpoints para digitação de valores.

- ➔ Criar na tela **Receitas**, cinco setpoints para digitação e visualização de valores nos tags **Água**, **Açúcar**, **Xarope** e **Glucose**.
- ➔ Criar um setpoint associado ao tag **Codigo** (este último necessariamente com formato texto).

## 3. Criar na tela **Receitas** os procedimentos para manipulação das receitas.

- ➔ Criar na tela **Receitas**, quatro botões que executarão scripts para realizar operações básicas com as receitas. São eles:
- ➔ **Selecionar e Carregar**: permite escolher qual receita se deseja editar. No arquivo que foi criado, **modelo1.rcp**, podem existir várias receitas, ou seja, vários conjuntos de valores. Através de um procedimento de seleção, escolheremos qual das receitas que desejamos manipular. Para tal, devemos obter um número, que é a posição no arquivo ou número da receita, o que será armazenado no tag **numero\_receita**.

```
numero_receita= Modelo1.ChooseRecipe("Escolha o Produto",1)
```

- ➔ A linha acima faz com que seja aberta uma janela para a escolha da receita desejada. Ao adicionar no mesmo script as linhas abaixo, a receita selecionada será carregada, cujo nome será copiado para o tag **Codigo**.

```
Modelo1.LoadRecipe (numero_receita)
```

```
Produtos.Codigo=Modelo1.GetRecDescription (numero_receita)
```

- ➔ **Criar Nova Receita**: permite a abertura de um novo registro ou conjunto de dados no arquivo **modelo1.rcp**.

```
numero_receita=Modelo1.CreateNewRecord(Produtos.Codigo)
```

- ➔ **Deletar Receita**: a partir do número do registro de uma receita, podemos retirá-la do arquivo **modelo1.rcp**.

```
Modelo1.DeleteRecipe(numero_receita)
```

- ➔ Uma outra sugestão para deletar uma receita pode ser a seguinte:

```
IF MessageBox("Deseja Realmente Deletar a Receita?","_
"Deletar a Receita", 0124h) == 6
Modelo1.DeleteRecipe(numero_receita)
Modelo1.LoadRecipe(1)
Produtos.Codigo= Modelo1.GetRecDescription(1)
ENDIF
```

- ➔ A função **MessageBox** é usada para confirmar se o usuário deseja realmente deletar a receita. Esta função está presente no Gerenciador Global e serve como interface de diálogo com o usuário quando se faz necessário alguma informação ou intervenção. Além disso, este script de exemplo também carrega a primeira receita, de modo que os setpoints não fiquem com valores de uma receita que não existe mais.

- ➔ **Editar Receita:** é uma função já pronta, presente no software, que substitui os procedimentos anteriores. Realiza a abertura de uma janela padrão, onde o usuário pode criar, editar ou deletar receitas. No caso deste exemplo, permitiremos a manipulação dos dados de duas formas: através da janela padrão de edição, ou ainda através dos setpoints.

```
Modelo1.EditRecipe()
```

- ➔ **Salvar Receita:** permite carregar os valores, presente nos tags, para uma receita ou posição no arquivo de dados, a fim de armazená-los. Para tal, devemos informar o número da receita, que deve ter sido previamente criada.

```
Modelo1.SetRecDescription(numero_receita,Produtos.Codigo)
Modelo1.SaveRecipe(numero_receita)
```





Os **Históricos** são objetos responsáveis pelo armazenamento de valores de tags. O armazenamento pode ser feito por tempo ou por evento, que deve ser especificado para a gravação dos dados.

## 9.1. Tipos de Históricos

Os Históricos podem ser gerados de duas maneiras diferentes: **Contínua**, que armazena os dados continuamente durante a execução da aplicação (ex: gravação das temperaturas de uma câmara fria a cada quinze minutos) ou em **Batelada**, no caso de processos em lote (ex: gravação das temperaturas de um forno separadas por código de lote de produção e nome do operador). Na Batelada, os dados são armazenados acompanhados de uma ou mais referências, associadas ao um **Cabeçalho** do histórico pela qual uma Batelada pode ser localizada. Os processos de Batelada precisam de comandos específicos, via algum script, para ser iniciado (`StartBatchProcess()`) ou terminado (`FinishBatchProcess()`). Para criar ou editar um Histórico precisamos selecionar o item Histórico no Organizer.

The image shows a dialog box titled 'Propriedades gerais do Histórico' with three tabs: 'Geral', 'Tags', and 'Referência-Cruzada'. The 'Geral' tab is selected. The dialog contains the following fields and controls:

- Nome:** A text box containing 'Hist1'.
- Descrição:** A text box containing 'Arquivo histórico'.
- Arquivo:** A text box containing 'hist.dat'.
- Tempo Escr.:** A text box containing '1000' followed by 'ms'.
- Máx. Regs.:** A text box containing '10000'.
- Buttons:** 'Análises...', 'Atualizar', 'CEP...', and 'Localizar...'.
- Checkboxes:**
  - Habilita histórico por scan
  - Processo de Batelada
  - Suporte a rede

Figura 41: Propriedades gerais do Histórico

### Propriedades gerais dos Históricos

OPÇÃO	DESCRIÇÃO
Nome	Define o nome do Histórico (pode ser modificado em execução).
Descrição	Descrição do Histórico.
Análises...	Permite a visualização dos dados do Histórico em forma gráfica.
Atualizar	Atualiza a estrutura do arquivo de Histórico quando ocorrer alguma mudança na configuração do Histórico.
CEP...	Chama o Controle Estatístico de Processos (CEP).
Arquivo	Define um nome de arquivo para o Histórico.
Tempo Escr.	Define a frequência com que os dados serão escritos no arquivo.
Máx. Regs.	Define um número máximo de registros para o arquivo de Histórico. O arquivo de Histórico é rotativo, ou seja, quando os dados excederem o tamanho do arquivo os primeiros registros serão substituídos.
Habilita histórico por scan	Habilita a escrita no arquivo de Histórico a partir do início da execução da aplicação segundo a taxa de varredura (scan) definida. Deixe esta opção desmarcada se você deseja controlar manualmente (usando Scripts) a geração dos dados do Histórico.
Processo de Batelada	Define o tipo do Histórico como sendo Batelada. Quando esta opção está marcada, um arquivo de cabeçalho (.HDR) é criado. Este arquivo guarda informações sobre cada batelada.
Suporte a rede	Habilita o suporte a rede para o Histórico, isto é, permite que o Histórico seja acessado (somente para leitura) por outras aplicações Elipse na rede, através de um objeto Browser ou objeto Relatório do tipo Análise Histórica. Esta opção também faz com que cada modificação no histórico seja gravada instantaneamente, não permitindo que o Sistema Operacional realize um “agendamento” da tarefa para realização posterior.

Após a configuração do tipo de histórico devemos adicionar ao histórico os tags que serão armazenados, que é feito na aba **Tags**. A cada variável adicionada devemos determinar o formato do campo, que pode ser:

Tipos de variável

TIPO	INTERVALO DE VALORES	TAMANHO (BYTES)
char	-128 a 127	1
byte	0 a 255	1
short	-32768 a 32767	2
word	0 a 65535	2
long	-2147483648 a 2147483647	4
dword	0 a 4294967296	4
float	3.4E +/- 38 (7 dígitos)	4
double	1.7E +/- 308 (15 dígitos)	8
datahora	01/01/1970 a 05/02/2036	8

O tipo string tem tamanho variável, declarado no histórico.

Ao escolher o histórico como tipo **Batelada**, será criado mais um item no objeto Histórico, chamado **Cabeçalho** que pode ser acessado via Organizer. No cabeçalho devem inseridos tags, cujos valores serão usados para identificar cada um dos processos de batelada.

Seguindo o exemplo, teremos o armazenamento de temperaturas do forno, onde queremos ter referência via um código de lote e o nome do operador. Assim, devem existir em sua aplicação dois tags: **código** e **operador**, que serão inseridos associados ao item **cabeçalho** e terão seus valores gravados a cada início e fim de batelada, de modo a identificá-la.

Os dados de cabeçalho são gravados num arquivo com extensão **.HDR**, de mesmo nome do arquivo de dados do histórico. Por exemplo, se você informou **hist1.dat** como nome do arquivo de dados do histórico, será criado **hist1.hdr**.

Deve existir no cabeçalho pelo menos um tag, configurado como um campo texto, para que a busca da batelada possa ser realizada.

## 9.2. Análise Histórica

Quando criamos um novo histórico, é obtido como um subitem, um objeto de análise histórica automaticamente. Este objeto pode ser criado separadamente no item **Relatórios**, a fim de realizar análises diferentes no arquivo **Histórico** que está sendo gerado ou mesmo em outros arquivos que não tenham sido gerados por esta aplicação. Na análise, estão inclusos os objetos **Plotter** (que é a geração do gráfico) e **Consulta**, que cuida do filtro de dados que será aplicado no arquivo a ser visualizado.

Você pode chamar a análise histórica em tempo de execução, através da função **Analysis()** do objeto **Histórico**.

Vejamos as configurações da análise histórica:

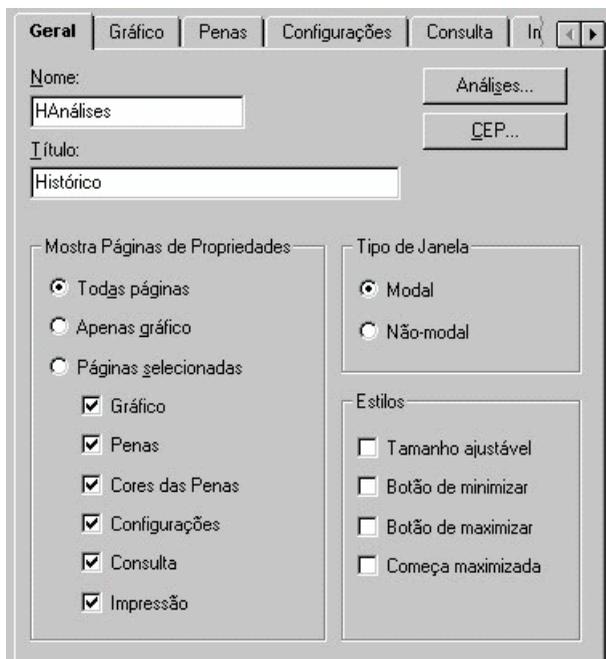


Figura 42: Propriedades da análise histórica

### Propriedades da análise histórica

OPÇÃO	DESCRIÇÃO
Nome	Define o nome da análise histórica.
Título	Define o título a ser mostrado na janela da análise histórica.
Análises...	Permite a visualização dos dados do histórico em forma gráfica.
CEP...	Chama o Controle Estatístico de Processos (CEP).
Mostrar Páginas de Propriedades	Define as páginas de propriedade a serem exibidas na janela da análise histórica, de acordo com a seleção.
Tipo de Janela	Define o tipo de diálogo da análise histórica: Modal, que não permite acessar nenhuma outra janela antes de ser fechado) e Não-Modal (que não necessita ser fechado).
Estilos	Define o estilo da janela da análise histórica, definindo elementos da janela.

### 9.2.1. Configurando a Análise Histórica

Podemos especificar o modo de trabalho da análise através das páginas Gráfico, Penas, Configurações, Consulta e Impressora, que são comuns entre os objetos que fazem consulta a base de dados.

Para os objetivos desse tutorial, faremos aqui uma breve explanação apenas sobre duas páginas: Configurações e Consulta.

#### ***Página de configurações da análise histórica***

Define arquivos e bateladas da análise histórica.

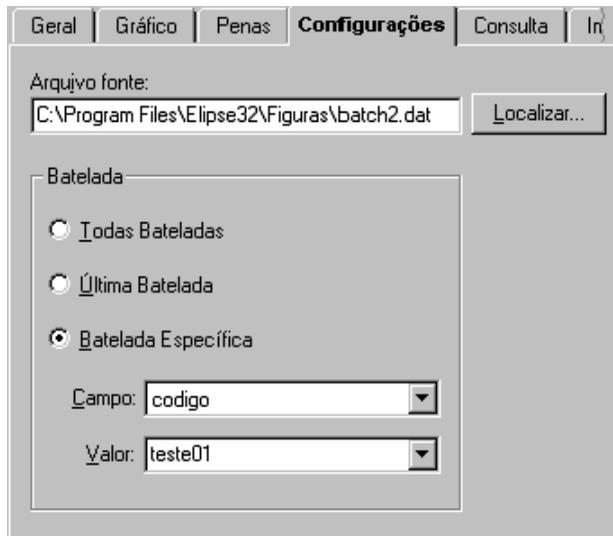


Figura 43: Configurações da análise histórica

### Configurações da análise histórica

OPÇÃO	DESCRIÇÃO
Arquivo fonte	Define o nome do arquivo fonte para a Análise (extensão .DAT).
Localizar...	Permite localizar o arquivo fonte a ser usado.
Todas Bateladas	Indica que todas bateladas do histórico serão impressas. Esta opção está disponível somente quando o arquivo-fonte é de histórico por bateladas.
Última Batelada	Indica que somente a última batelada será impressa. Esta opção está disponível somente se o arquivo-fonte é de histórico por bateladas.
Bateladas Específicas	Seleciona uma batelada específica a ser impressa, conforme o especificado em <b>Campo</b> e <b>Valor</b> . Esta opção está disponível somente quando o arquivo-fonte é de histórico por bateladas.
Campo	Lista os campos disponíveis para seleção de uma batelada.
Valor	Define o valor a ser buscado.

### **Página de Consulta**

Permite definir um intervalo de tempo para selecionar os dados do arquivo a ser visualizado.

**Erro! Não é possível criar objetos a partir de códigos de campo de edição.**

Figura 44: Propriedades da consulta para análise histórica

**Propriedades da consulta para análise histórica**

<b>OPÇÃO</b>	<b>DESCRIÇÃO</b>
Sem consulta	Não será usado filtro, selecionando todos os dados.
Intervalo de tempo	Seleciona os dados dentro de um intervalo de tempo especificado. Deve ser fornecida uma data inicial e uma data final para busca.
Data mais recente	Seleciona apenas os dados mais novos. Use esta opção quando o histórico deva mostrar as últimas aquisições num período de tempo especificado.
Data Inicial	Determina o dia inicial do intervalo de tempo.
Hora Inicial	Determina a hora inicial do intervalo de tempo.
Dia Final	Determina o dia final do intervalo de tempo.
Hora Final	Determina a hora final do intervalo de tempo.
Último	Define as unidades usadas para selecionar os dados mais recentes.

### 9.3. Dicas e Otimização

---

#### ***Tamanho do arquivo de dados***

O tamanho de um arquivo histórico que será gerado pode ser calculado assim:

$$\text{Tamanho do Arquivo de Dados} = \text{tamHeader} + \text{tamRegistro} * \text{numRegistros}$$

$$\text{Tamanho do Cabeçalho (Header)} = 24 + \text{numCampos} * 40$$

$$\text{Tamanho do Registro} = 2 + \text{somatórioTamanhoCampos}$$

Os 2 bytes que são somados correspondem a verificação do tipo CRC16 presente em cada registro.

#### ***Fazendo gravação por eventos***

Para realizar a gravação de dados por eventos, você deve desabilitar o histórico ao iniciar a aplicação (opção na página **Geral**). Assim, os dados só serão gravados num comando explícito de gravação, que deve ser feito através de um script.

Há duas maneiras diferentes em scripts que podem realizar tal operação. Por exemplo, supondo **Hist1** o nome de nosso histórico de exemplo, teríamos:

```
Hist1.Open()
Hist1.WriteRecord()
Hist1.Close()
```

Ou:

```
Hist1.enabled = 1  
Hist1.WriteRecord()  
Hist1.enabled = 0
```

Ambos realizam basicamente a mesma operação, porém o segundo habilita a gravação por tempo, enquanto a propriedade **enabled** estiver em 1. O comando de escrita **WriteRecord()** obtém o valor atualizado dos tags que estão associados ao histórico, realizando a inserção de mais uma linha de dados.

### **Informação de tempo no histórico**

Cada registro de dados que é gravado, possui uma marcação de tempo, que é o campo **DateTime**, que possui 8 bytes de dados (64 bits). Normalmente, ao gravar um dado, é consultado o relógio do computador e a informação obtida é inserida naquela linha de dados, com precisão de um milissegundo. Você pode porém, estipular outra fonte de data e hora.

Ao clicar no objeto **Histórico** no Organizer, aparecerão sob o mesmo todos os campos de dados associados a ele. O primeiro deles é sempre o campo **DateTime**. Ao clicar sobre este campo, há a possibilidade de associar um tag ou propriedade, através da aba **Tags**. Neste caso, pode ser associada a propriedade **TimeStamp** de algum tag, caso o equipamento com o qual se está trocando informações, suporte o envio de informações de seu relógio local.

Assim, o valor da data gravado no histórico não será o do microcomputador, mas sim, o de outra máquina ou equipamento.

### **Tendência histórica**

O objeto de tela **Tendência**, quando configurado como tendência histórica, pode visualizar os dados históricos bem como a análise. Os objetos associados à tendência nesse caso, não são os próprios tags, mas os campos do histórico, que são criados quando se associam tags ao histórico.

Para escolher dados sobre o arquivo a ser associado à tendência, bem como escolher a forma de consulta, é criada uma página de **Consulta**, que pode ser acessada na página **Avançado** na tendência.

A associação das penas aos campos é feita clicando-se sobre o campo **Eixo Y** e **Eixo X** na aba **Penas**, acessada pelas propriedades da **Tendência**. Ao clicar sobre este campo aparecerá uma lista com o nome dos campos do histórico.

Para fazer a tendência buscar os dados no disco ou atualizar seus dados, é necessário, via algum script, executar sua função **UpdateQuery()**. Caso queira que isto ocorra a intervalos fixos, basta inserir este comando no script **WhileRunning** da tela, informando no script o tempo em milissegundos da sua execução.

### **Modo de coleta de dados do gráfico de tendência**

Através da página **Avançado** do objeto **Tendência** podemos determinar o método de coleta de dados. A primeira opção de **Tempo Real** faz com que a tendência só colete dados quando a tela estiver aberta. Ao chamá-la novamente, aparecerá limpa.

A segunda faz com que a comunicação com seus tags aconteça independente da tela; logo ao chamá-la os dados já estarão carregados.

A terceira opção, **tendência histórica**, permite recuperar os dados após o computador ser desligado, ou ainda espelhar os mesmos dados gravados em disco pelo objeto histórico. Para isso, os objetos associados à tendência não são os próprios tags, mas os campos do histórico, como já foi mencionado.

### **Inserção de marcas no gráfico de tendência**

Através da página **Penas** do objeto **Tendência**, podemos associar marcas ao gráfico, que podem ser linhas verticais, horizontais ou pontos (marcas). Cada marca pode ser associada a um tag ou propriedade, ou ainda ser modificada via script.

### **Objeto Browser**

O objeto de tela **Browser** possui as mesmas características da tendência histórica, porém mostra os dados de modo texto, não graficamente. O **Browser** também não atualiza seus dados automaticamente, logo você deve executar sua função `UpdateQuery()` através de algum script, quando quiser que a atualização ocorra.

Este objeto, permite também, definir algumas propriedades como o número de linhas que podem ser vistas, a linha que está selecionada pelo usuário, etc., além de permitir a formatação de cada campo que será visualizado.

O **Browser** permite também a visualização do arquivo de bateladas, através da especificação do arquivo `.HDR` correspondente na página **Configurações**.

# Exercícios

## 1. Criar um objeto histórico, para gravação contínua.

- ➔ A partir do Organizer, criar um novo histórico, com o nome de “Hist1”.
- ➔ Especifique o nome do arquivo como `continuo.dat`, o tempo de escrita em 1000 ms e o número máximo de registros em 1000.
- ➔ Habilitar a gravação ao iniciar a aplicação (por scan).
- ➔ Acessar o HAnálises dentro do Hist1 pela árvore da aplicação do Organizer e na aba Consulta definir "sem consulta por data".
- ➔ Na aba Tags, inserir os tags de nível.

## 2. Criar um botão na tela, chamando a função Análise Histórica na tela de Dosagem.

- ➔ Insira um botão do tipo momentâneo, na tela Dosagem.
- ➔ Crie um script `OnRelease` para o botão, inserindo a função `Hist1.Analysis()`, para chamar a análise histórica.
- ➔ Na página Mensagens, digite “F5 – Análise” para o texto normal e pressionado.

## 3. Criar um objeto histórico com gravação por batelada.

- ➔ A partir do Organizer criar um novo histórico, com o nome “Hist2”.
- ➔ Especifique o nome do arquivo como `batch.dat`. Marque o histórico como batelada.
- ➔ No item **Cabeçalho**, que pode ser acessado via Organizer (dentro do objeto `Hist2`), associe o tag `Codigo` (que é o código do produto) como um string de 10 caracteres e a propriedade `Aplicação.UserName`, que é o nome do usuário que está logado no sistema, também como uma string de 10 caracteres.
- ➔ Na aba Tags do objeto `Hist2`, adicione os tags de temperatura.
- ➔ Clique no botão **Atualizar** da aba Geral, para gerar a estrutura dos arquivos.

## 4. Configurar a tela para cadastro das bateladas.

- ➔ Insira três botões do tipo momentâneo na tela de Bateladas, para executar três tarefas básicas das bateladas, que são o **Início**, **Fim** e **Reinício**. Tais ações poderiam ser executadas via algum sinal proveniente do campo, mas para efeitos de testes, o faremos manualmente através dos botões.

- ➔ Nas mensagens dos três botões digite “Iniciar”, “Finalizar” e “Reiniciar”.
- ➔ Para o primeiro, crie um script `OnRelease`, executando a função `Hist2.StartBatchProcess()`.
- ➔ Para o segundo, a função é `Hist2.FinishBatchProcess()`.
- ➔ Para o terceiro, a função será `Hist2.RestartLastBatch()`.

### 5. Criar através do uso de dois objetos browser, um sistema para escolha de análise por batelada.

- ➔ Inserir na tela de batelada, dois objetos browser. O browser superior será chamado de **Browser1** e o browser inferior será chamado de **Browser2**.
- ➔ Neste exemplo, o browser permitirá, através da navegação no arquivo `.HDR`, a escolha da batelada que desejamos visualizar. Assim, o primeiro deve estar associado ao arquivo `Batch.hdr`.
- ➔ Na página **Consulta**, deixe sem consulta.
- ➔ Na página **Banco de Dados**, clique em **Atualizar estrutura do arquivo**. Depois, clique no campo **Código**, especificando a palavra “Código” como **Etiqueta** e no campo **Aplicação.UserName** a palavra “Operador”.
- ➔ No segundo browser, faça a associação ao arquivo de dados `batch.dat`. Na página de configurações, escolha a opção **Batelada Específica**.
- ➔ Crie um script para o primeiro browser no evento `OnLButtonDbClick`, que será executado ao pressionar o botão esquerdo do mouse 2 vezes:

```

Cabeçalho.Open()
Cabeçalho.GoTo(Browser1.curSel)
Cabeçalho.Edit()
Browser2.Consulta.criteria = Cabeçalho.Codigo
HAnalysis.Consulta.criteria = Cabeçalho.Codigo
Cabeçalho.Close()
Browser2.UpdateQuery()

```

Basicamente, o script acima abre o arquivo `.HDR` na mesma linha que está sendo clicada pelo usuário na tela. Logo após, é ajustado como critério de busca para o `Browser2` e para a **Análise Histórica**, a batelada cujo código é o que está sendo visto pelo operador.

O objeto browser não possui atualização de dados automática, ou seja, o arquivo de dados não é reconsultado automaticamente a intervalos regulares. Esta tarefa é

realizada apenas ao entrar na tela que possui o objeto, ou ainda através de uma função de atualização, chamada de `UpdateQuery()`, que está presente na última linha de nosso exemplo.

- ➔ Para o segundo botão (Termina) podemos adicionar a função `UpdateQuery()` de modo que ao terminar a batelada os browser estarão atualizados. Neste caso, o script deste botão ficaria com a seguinte configuração:

```
Hist2.FinishBatchProcess()  
Browser1.UpdateQuery()
```

### 6. Criar um setpoint para a digitação do código do produto.

- ➔ Inserir na tela de batelada, um objeto setpoint, escolhendo na página de formato o dado como tipo texto. Na aba Tags, associe ao tag `Codigo`.







Os relatórios permitem realizar a impressão dos dados históricos, cabeçalhos e alarmes, e ainda dados instantâneos.

Existem quatro tipos de relatórios:

- **Relatório Texto:** Pode realizar impressão de dados no formato de linhas e colunas, inclusive de arquivos de alarmes. Permite também a impressão para arquivo em disco.
- **Relatório Gráfico:** Realiza impressão gráfica de dados históricos, com a criação de legenda.
- **Relatório Formatado:** Use esta opção para imprimir um formulário com textos quaisquer, valores instantâneos de variáveis e bitmaps.
- **Análise Histórica:** Relatório de tela, para visualização/impressão de dados do disco. Use este relatório quando desejar possuir várias análises diferentes para cada arquivo, ou mesmo para análise de dados remotos, gerados por outras aplicações.

Todos os relatórios possuem uma aba de Critério, para especificação do intervalo de busca dos dados e uma aba de Configurações para escolha do arquivo de dados e de bateladas, se esta opção estiver habilitada.

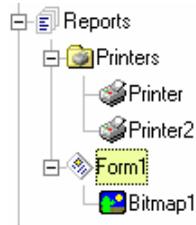
### 10.1. Procedimentos com relatórios

---

#### *Como trocar a impressora e reconfigurar os relatórios*

Os relatórios, quando salvos no aplicativo, contém informações quanto à impressora, página, etc. Ao trocar ou alterar algum dado sobre a impressão, estes dados devem ser reconfigurados.

Ao selecionar o item **Relatórios** via Organizer, será mostrado como subitem os relatórios já criados para aquela aplicação e a lista de impressoras cadastradas. Você pode determinar que um certo relatório utilize hora uma ou outra impressora da lista, a depender das condições desejadas.



Você pode associar, por exemplo, o Relatório Form1 à impressora Printer.

Em execução, podem ser alteradas as seguintes propriedades:

- Alterar a impressora do relatório Form1 de Printer para Printer2 - feito através da edição de propriedades do relatório.
- Alterar a impressora Printer de um certo modelo de impressora física para outro - feito através da edição de propriedades da impressora.

### **Impressão em arquivo**

Podemos gerar um relatório em arquivo usando a função PrintToFile. Sua sintaxe é:

```
PrintToFile(<arquivo> [, <bImprimeCabecalho> [, <separador> [, <bIndicarProgresso>]]])
```

Onde:

<arquivo> é um string com o nome do arquivo a ser impresso no disco;  
<bImprimeCabecalho> habilita a impressão do cabeçalho;  
<separador> é um string usado para separar os campos na impressão;  
<bIndicarProgresso> habilita uma barra para indicar o progresso da impressão.

### **Como imprimir uma tela**

Para isso, basta criar um relatório tipo formatado, inserindo um bitmap como único objeto, ocupando toda a extensão da página.

Suponha que você queira estabelecer a tecla [Alt+I] para realizar esta impressão. Para associar esta tecla a um script para impressão, siga os seguintes passos:

- ➔ Vá nas configurações da tela (caso queira a impressão somente em uma tela) ou na aplicação (caso queira em todas as telas da aplicação).
- ➔ Escolha na página de scripts, um novo script para o evento OnKeyRelease.
- ➔ Clique no botão de captura e digite as teclas [Alt+I].
- ➔ Neste script, acrescente as seguintes funções:

```

CaptureScreen("teste.bmp")
Form1.Bitmap1.bitmapName="teste.bmp"
Form1.Print()

```

### **Como escolher um filtro de data e hora para impressão**

Há duas maneiras de realizar esta tarefa:

- Criar um botão cujo script `OnRelease()` chamará uma função de configuração da consulta do relatório, como por exemplo, a função `Relatorio.Query.EditProperties()`. Em execução aparecerá uma janela semelhante àquela mostrada em configuração.
- Inserir 12 setpoints, que permitirão a escolha dos intervalos iniciais e finais. Cada setpoint será associado a uma das propriedades da consulta do relatório:

```

Relatorio1.Consulta.StartHour
Relatorio1.Consulta.StartMinute
Relatorio1.Consulta.StartSecond
Relatorio1.Consulta.StartDay
Relatorio1.Consulta.StartMonth
Relatorio1.Consulta.StartYear
Relatorio1.Consulta.FinalHour
Relatorio1.Consulta.FinalMinute
Relatorio1.Consulta.FinalSecond
Relatorio1.Consulta.FinalDay
Relatorio1.Consulta.FinalMonth
Relatorio1.Consulta.FinalYear

```

### **Como escolher um filtro de campos específicos para impressão**

Podemos utilizar a função `AddFilter` para antes de realizar a impressão, realizar um filtro em um campo específico, juntamente com a data. Sua sintaxe é:

```
AddFilter(<campo>, <valorMin>, <valorMax>)
```

Onde:

<campo> é o nome do campo do histórico sobre o qual quer se aplicar o filtro;  
 <valorMin> e <ValorMax> são os valores limites.

# Exercícios

1. **Criar um relatório tipo texto para a impressão de alarmes.**
  - ➔ Criar um relatório. Especificar nome “Relatorio1”. Escolher o arquivo `continuo.dat`. Escolher na consulta o critério Intervalo de tempo.
2. **Criar uma nova tela para seleção de intervalo de impressão.**
  - ➔ Criar uma nova tela do tipo janelada. Como bitmap de fundo, inserir o arquivo `Lib\Eletric\datahora.bmp`. Inserir os setpoints que permitirão a escolha dos intervalos iniciais e finais. Cada setpoint será associado a uma das propriedades da consulta do relatório:

```
Relatorio1.Consulta.StartHour
Relatorio1.Consulta.StartMinute
Relatorio1.Consulta.StartSecond
Relatorio1.Consulta.StartDay
Relatorio1.Consulta.StartMonth
Relatorio1.Consulta.StartYear
Relatorio1.Consulta.FinalHour
Relatorio1.Consulta.FinalMinute
Relatorio1.Consulta.FinalSecond
Relatorio1.Consulta.FinalDay
Relatorio1.Consulta.FinalMonth
Relatorio1.Consulta.FinalYear
```
3. **Criar objetos bitmap para a impressão.**
  - ➔ Inserir dois objetos tipo bitmap, nos cantos inferiores da tela. O primeiro deve ser associado ao bitmap `Lib\Util\disquete.bmp` e o segundo, `Lib\Util\impres.bmp`.
  - ➔ Criar um script do botão esquerdo do mouse como segue:
  - ➔ Script OnLButtonUp
  - ➔ `Relatorio1.PrintToFile("teste.txt",0,')`
  - ➔ Para o segundo bitmap, um script como segue:
  - ➔ Script OnLButtonUp
  - ➔ `Relatorio1.Print()`
4. **Criar na tela de impressão um procedimento de configuração da impressora.**
  - ➔ Criar um relatório formatado.
  - ➔ Inserir um objeto bitmap, com o arquivo `Lib\Util\tools.bmp`. Marcar como transparente, com fundo verde-limão.





Através das **Senhas**, podemos controlar o acesso a telas de uma aplicação. É possível guardar uma lista de usuários, sendo que para cada um é atribuído um nome, um login (identificação no sistema), uma senha e um nível de segurança.

Se nas propriedade gerais de uma tela, no campo **Nível de Acesso** for especificado um número diferente de “0” (o nível zero libera o acesso a todos os usuários) será necessário que um usuário esteja **logado** e que seu nível de segurança permita o acesso a tela em questão.

Para o usuário se logar no sistema, deve-se obrigatoriamente executar a função **Login()**. Caso um usuário já estiver logado, esta função providencia primeiramente um logout do usuário antigo através da função **Logout()** e depois a autenticação do novo usuário.

Conforme o nível de acesso delegado, cada usuário terá disponível algumas funções. O usuário com nível 1 é considerado **superusuário**. Este, quando logado pode criar, modificar e remover os atributos de todos os usuários. Estes métodos estão implementados na função **UserAdministration()**. Outros usuários podem, através da mesma função, fazer a troca de sua senha.

As senhas e níveis são gravadas num arquivo criptografado e independente dos demais no diretório da aplicação.

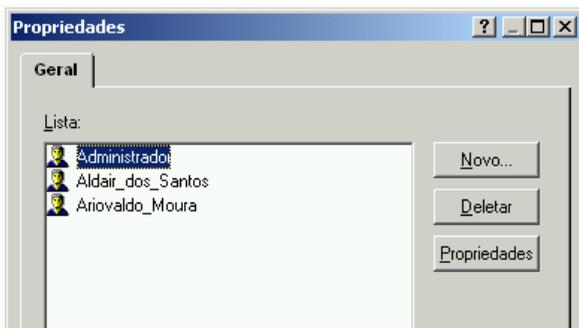


Figura 45: Propriedades dos Usuários

# Exercícios

## 1. Criar usuários e cadastrá-los.

➔ Criar usuários com vários níveis de acesso e alterar os níveis de acesso nas telas do sistema, menos na tela de Abertura que terá acesso liberado para todos os usuários.

## 2. Criar procedimento de login do usuário na tela de Abertura.

➔ No objeto bitmap de login da tela de abertura, criar um script OnLButtonUp, executando a função Login(), presente na aplicação.

## 3. Criar procedimento de manutenção de senhas.

➔ Insira um objeto bitmap na tela de abertura para chamar a manutenção de senhas.

➔ Associe os arquivos `manut.bmp` e `manut2.bmp`, que irão variar se o mouse estiver sobre o objeto ou não. Estes arquivos se encontram no diretório `Lib\Icons`.

➔ Criar um script OnLButtonUp, associado ao bitmap, que execute a função `Aplicação.UserAdministration()`.

## 4. Inserir nome do usuário que foi logado na tela de abertura.

➔ Inserir um display na tela de abertura, sobre a barra de ferramentas, trazendo-o para a frente.

➔ Configure a fonte como “Arial Regular”, tamanho 10, cor preta, com alinhamento à esquerda. No campo Prefixo, digite “Usuário:”.

➔ Na aba Tags, associe a variável de sistema `Aplicação.UserName`.

➔ Insira também um display mostrando o nível de acesso do usuário, via propriedade `UserAccessLevel`.

## 5. Criar botão para chamar a tela de dosagem, com o texto: “Entrar no Sistema”.

## 6. Criar verificação de usuário logado ou não.

➔ Utilize as funções `MessageBox`, para fazer a interface com o usuário.

## 7. Criar cadastro de log do usuário.

➔ Crie um tag RAM chamado “User”. Através da marcação de um alarme alto em 1 nessa variável, podemos registrar o login de um usuário.

➔ Através do uso do script `OnUserLogin` da aplicação, realizar o seguinte script:





A opção **Databases** (Bancos de Dados ou simplesmente BD) do Eclipse SCADA permite criar e manipular um ou mais bancos de dados usando o padrão ODBC. É possível a conexão com um banco de dados já existente ou criar um novo a partir de um assistente dentro do software.

Importante dizer que antes de utilizar um BD dentro do Eclipse SCADA, é necessário criar uma conexão ODBC para o BD desejado.

Para criar uma conexão com uma nova tabela:

- ➔ Escolha o driver ODBC que deseja usar.
- ➔ Configure o nome da conexão e o arquivo ou diretório que contém os dados.
- ➔ Crie cada um dos campos, escolhendo nome, tipo de dado e tamanho.
- ➔ No Organizer, aparecerá a tabela criada e seus respectivos campos, que poderão ser modificados usando as funções especiais do ODBC nos scripts.

Para criar uma conexão com um banco de dados já existente:

- ➔ Crie o banco de dados. No caso do Excel, deve-se utilizar a primeira linha de cada coluna como o nome do campo.
- ➔ Selecione a linha de cabeçalho (com o nome dos campos) e na caixa de nome, coloque um nome para essa tabela.
- ➔ Feche o banco de dados.
- ➔ No Organizer, em **Databases**, escolha “Conectar a uma tabela já existente”, pressione o botão **Novo** e escolha o driver ODBC que deseja usar.
- ➔ Escolha o diretório onde está o banco de dados.
- ➔ Na árvore devem aparecer a tabela e seus respectivos campos.

Consulte o manual para saber mais sobre Bancos de Dados.

# Exercícios

Para os exercícios deste capítulo, é necessário ter instalado o banco de dados Microsoft Access, rodando no sistema operacional Windows 2000.

## 1. Criar um arquivo de banco de dados do Access vazio.

- ➔ Execute o Microsoft Access e crie um novo MDB, através do comando **Novo** do Menu **Arquivo**, opção **Banco de Dados em Branco**. Coloque o nome de “exemplo.mdb”.
- ➔ Crie uma tabela em “Modo Estrutura” (Design View).
- ➔ Crie quatro campos:
  - **Matricula**, do tipo número;
  - **Nome**, do tipo texto.
  - **Cargo**, do tipo texto.
  - **Telefone**, do tipo texto.
- ➔ Coloque como chave primária o campo **Matricula**. Coloque o nome da tabela de “Cadastro”. Salve o arquivo e feche o Access.

## 2. Criar uma conexão ODBC.

- ➔ A partir do **Painel de Controle do Windows**, que pode ser acessado através da opção **Configurações** do Menu **Iniciar** do Windows, escolha **Ferramentas Administrativas** (Administrative Tools) e depois **Fontes de Dados ODBC** (ODBC Data Sources).
- ➔ Na aba **Sistema DSN** (DSN System), clique no botão **Adicionar** (Add).
- ➔ Selecione o driver do **Microsoft Access** e clique em **Concluir** (Finish).
- ➔ Digite o nome da fonte de dados "Cadastro".
- ➔ Clique em **Selecionar** (Select) e selecione o arquivo **exemplo.mdb** recém criado.
- ➔ Depois disso clique em **OK**,finalizando assim a criação da conexão ODBC.

## 3. Criar um Database ligado ao arquivo criado.

- ➔ De volta ao **Elipse SCADA**, a partir do **Organizer**, crie um novo Database. Utilize o tipo de conexão ODBC. Selecione a conexão chamada **Cadastro** recém criada. A tabela **Cadastro** deve aparecer na lista de tabelas existentes. Caso não apareça, clique no botão **Atualizar**.

- ➔ Para finalizar a criação do Banco de Dados, clique em OK. Repare que foi criado um novo item dentro de Databases que pode ser visualizado pela árvore da aplicação do Organizer com os campos da tabela.

#### 4. Criar uma tela para visualizar o Database.

- ➔ Crie uma tela chamada "Funcionários" com 4 setpoints e associe cada um deles ao respectivo campo da tabela.

Matrícula	Nome	Cargo	Telefone
0	Carlos	0	0

- ➔ Crie 6 botões na tela para navegar pela tabela e adicionar novos registros. Para cada botão, crie um script no evento *OnRelease*, usando as funções do Database mostradas abaixo:



: Função *MoveFirst()*



: Função *MovePrev()*



: Função *AddRecord()*



: Função *DeleteRecord()*



: Função *MoveNext()*



: Função *MoveLast ()*

- ➔ Pode-se criar um novo botão na tela de dosagem para abrir a "Funcionários", a exemplo dos outros botões de abertura de telas criados anteriormente.



Figura 46: Tela de Banco de Dados



Para executar as tarefas de configuração no menor tempo possível e obter uma melhor performance na execução de sua aplicação, listamos a seguir algumas dicas que podem ser úteis.

### ***Memória***

A quantidade de memória RAM necessária para uma boa performance de seu sistema depende do tamanho da aplicação. Geralmente cada item da aplicação gasta algo em torno de 2 KBytes. Para saber o total de itens de sua aplicação, pressione as teclas [Ctrl+Shift+Alt+I]. Caso sua máquina seja limitada, pode-se diminuir a quantidade de memória requisitada, trabalhando com desenhos de resolução menor (16 ou 256 cores) e criando animações somente nas áreas que ficarão animadas.

### ***Como configurar objetos de tela para melhorar sua performance***

Objetos como o gráfico de tendências podem realizar a coleta de dados em segundo plano e redesenhar o gráfico em intervalos maiores, sem perder a coleta. Logo, o tempo de “refresh” da tendência pode ser maior que o de varredura (scan) das variáveis sem prejudicar seu conteúdo. Outra consideração sobre objeto de tela é relativa ao objeto animação, que quando transparente gasta mais tempo de processamento que a animação normal.

### ***Imagens (bitmap) desaparecidas***

Se no caso do transporte de uma aplicação para outra máquina, as imagens sumirem, deve-se desabilitar a opção “Esconder extensões do MS-DOS para arquivos registrados”. Pode-se acessar essa opção a partir do Windows Explorer.

### ***Ordenando os “tabs” dos objetos do Elipse***

Com o uso da tecla [Tab] é possível se deslocar de um objeto para outro em uma série. Inicialmente, o deslocamento por [Tab] segue a ordem de criação dos objetos em tela. Porém, é possível modificar esta ordem selecionando os objetos na ordem desejada e apertando o botão Trazer para Frente ou Mandar para Trás.

## ***Criando atalhos para uma aplicação Elipse***

No Windows, a instalação do Elipse SCADA registra automaticamente os arquivos com a extensão .APP. Assim, as aplicações do Elipse SCADA podem ser executadas automaticamente com um duplo clique sobre o aplicativo gerado.

Para criar um ícone de acesso (atalho) na área de trabalho, basta localizar o diretório ou pasta de trabalho onde está o programa e arrastá-lo para o local desejado.

## ***Velocidade de comunicação***

Procure comunicar com os equipamentos de aquisição de dados em taxas mais altas sempre, desde que não hajam problemas de comunicação. Em se tratando de comunicação serial, em geral o Windows suporta bem a comunicação em 19200 bps. Procure usar esta velocidade, em detrimento a 9600 bps, mais comum.

## ***Número de tentativas de comunicação e scripts “OnCommError”***

Na configuração do driver de comunicação, na opção “Retentar Comunicações Falhadas” não é aconselhável executar um número alto de tentativas, pois se o equipamento apresenta erros de comunicação seguidos, faz-se necessário uma revisão de toda a comunicação, desde conexões físicas até o driver de comunicação. Simplesmente aumentar o número de tentativas faz com que cada tag lido com erro seja retentado um número *n* de vezes antes do software reconhecer o erro de comunicação, o que acaba degradando a performance da comunicação como um todo.

Para verificar e gerenciar com maior eficiência os erros de comunicação pode ser criado um script OnCommError no objeto Driver, que será chamado quando houver qualquer erro de comunicação.

Para manipular os erros, podem ser usadas algumas funções do driver, a saber:

- **GetErrorInfo(param)**: retorna informação sobre o último erro ocorrido. Se **param= 0**, retorna o tipo de comunicação que causou o erro (ver Manual do Usuário). Se **param= 1, 2, 3 ou 4**, retorna os parâmetros N1 a N4 ou B1 a B4 do tag PLC ou tag Bloco que causou o erro. Assim, seja uma rede de PLCs onde o parâmetro N1 ou B1 representa o número do PLC na rede, saberemos qual dos equipamentos está com falhas.
- **AddFilter(strFilter)**: Adiciona um filtro de comunicação ao Driver, desabilitando leituras ou escritas em tags que possuam parâmetros especificados no filtro. Assim podemos desabilitar a comunicação com um equipamento específico para manutenção, por exemplo, evitando erros enquanto estiver desconectado.

- **RemoveFilter(strFilter):** remove o filtro que foi adicionado com **AddFilter(strFilter)**. Por exemplo, suponha que estamos utilizando o driver Modbus, cujo parâmetro N1 indica o endereço do PLC na rede. Uma sugestão de script seria a seguinte. Primeiro, **crie um tag tipo RAM chamado "Erro"**. Depois, **Crie um script OnCommError** no driver de comunicação:

```

Erro = Driver1.GetErrorInfo (1)
Driver1.AddFilter (Str (Erro, 2, 0))
MessageBox (
  "Ocorreu um erro no PLC: "+StrZero (Erro, 2, 0),
  "Erro de Comunicação", 0040h
)

```

### ***Atributos “Habilitar leitura pelo scan”, “Habilitar leitura automática” e “Habilitar escrita automática”***

Procure configurar corretamente estes parâmetros em seus tags. O normal é que se uma variável deve ser normalmente buscada, as opções de leitura pelo scan e automática devem estar habilitadas. Caso você queira controlar diretamente a escrita e leitura através das funções **Read()** e **Write()**, deixe todas as opções desabilitadas.

### ***Advise de tags***

O Elipse realiza otimizações na comunicação a fim de evitar que variáveis sejam comunicadas sem necessidade, através do uso da propriedades **advise**, que indica que algum módulo (objeto de tela, alarme, histórico, relatório, script, etc...) está precisando do valor atualizado da variável. Assim, um tag que não está sendo utilizado por nenhum objeto não será lido. Caso queira que um tag seja lido independente de estar sendo usado ou não, você deve, por exemplo, habilitar um alarme ou mesmo forçar sua leitura através da função **Read()** do Tag. No caso dos tags expressão, sempre que uma das variáveis componentes mudar, o tag será reavaliado, caso este esteja em **advise**. Caso contrário, só será reavaliado quando algum módulo necessitar seu valor.

### ***Otimização de laços***

Caso não necessite aninhar vários laços, prefira separá-los, pois assim o script apresenta melhor performance. Por exemplo, laços **FOR...NEXT** aninhados:

```

For ...
  For ...
  Next
Next

```

são geralmente mais demorados que separados:

```
For ...  
Next  
For ...  
Next
```

### ***Manipulando datas no ELIPSE***

Você pode acessar a data do sistema pelo atributo `currentTime` ou separadamente, pelos atributos `day`, `month`, `year`, `hour`, `minute` e `second` que estão no Gerenciador Global.

Para mostrar uma data num formato qualquer na tela, basta inserir um `display` com o formato `Data/Hora`, associado à propriedade `currentTime`.

Note que os parâmetros mencionados retornam a data atual do sistema, o que significa que são atualizados a todo momento. Eles também podem ser substituídos por um valor (data absoluta) de outras fontes, como de um banco de dados, por exemplo.

### ***Arquivos de driver não aparecem***

No Windows, se ao tentar configurar um novo driver de comunicação na sua aplicação, não for possível encontrar o arquivo `.DLL`, deve-se mudar a opção de arquivos ocultos para “Mostrar todos os arquivos”. Esta opção encontra-se em Ferramentas/Opções de Pasta no Windows Explorer.

### ***Exibir imagens de vídeo***

Os objetos **AVI**, **Video** e **Preview** permitem a exibição de imagens geradas por câmeras de vídeo ou TV, ligadas ao computador. A resolução é definida pelo tamanho que o objeto é colocado na tela, com frequência de 30 quadros (frames) por segundo. Saiba mais sobre a utilização de vídeo no Manual do Usuário.



