Mini-Curso

Parte II

Programação Delphi para Eletrônica

Eduardo D. D. Vilela
eddv@mailbr.com.br

No artigo anterior pudemos ter uma visão básica do ambiente de desenvolvimento Delphi, entretanto, como você verá, apenas arranhamos a superfície. No presente artigo nos aprofundaremos um pouco mais no conhecimento desta ferramenta e na utilização dela envolvendo a Eletrônica, em um projeto prático onde utilizaremos a porta paralela do PC para a interface com o meio externo. Vale lembrar que este mini-curso é destinado a todos os que utilizam a eletrônica, desde profissionais que trabalham com ela em seu dia-a-dia, que têm uma oportunidade de adquirir conhecimentos numa área nova e bastante promissora, que é o uso de ferramentas programáveis em conjunto com a eletrônica; até empresas que precisam controlar/acompanhar processos de forma mais produtiva e dinâmica, onde é necessário possuir domínio do que se faz e ***como*** se faz, atuando eficazmente através do que é oferecido pelas informações do sistema.

Mais do que nunca, o uso eficiente da informação visando qualidade e produtividade, significa muitas vezes até mesmo a sobrevivência ou não no mercado.

Aba	Descrição dos Componentes
Standard	*Componentes padrão em uma interface Windows, tais como edit, botão e label.*
Additional	*Grupo adicional aos componentes padrão, tais como SpeedButton e Notebook.*
Win32	*Novos da interface de 32 bits do Win95, como abas, barra de status, progress bar.*
System	*Componentes que pertencem à tecnologia do sistema Windows. Exemplo: timer.*
Internet	*Específicos para desenv. em Delphi de aplicações para Internet: sockets e HTML*
Data Access	*Especializados para acesso a Banco de Dados, tais como Table, Query.*
Data Controls	*Especializados de BD utilizados para exibir dados, tais como Grid, Navigator.*
QReport	*Destinados à criação de relatórios e impressões em geral.*
Dialogs	*Janelas de diálogos comuns do Win95, que possuem uma aparência consistente para executar operações de arquivo, tais como abertura, gravação e impressão.*
Win31	*Componentes característicos do Win31, ainda utilizados no Win95*
ActiveX	*Controles da tecnologia ActiveX fornecidos com o Delphi ou de outros fabricantes.*
Samples	*Componentes diversos, tais como ColorGrid, Calendar e SpinButton.*

Componentes

No projeto prático que vimos na primeira parte do curso foram utilizados quatro componentes, um form e os outros três da aba 'standard', que é uma das 12 abas que acompanham o Delphi.

A aba 'standard' é uma das mais utilizadas em aplicações *for Windows* em geral, entretanto, como eventualmente faremos uso de componentes de outra abas, vejamos uma breve descrição de cada uma delas, conforme mostra a figura 1.

Fig. 1 - A paleta de componentes

Convém salientar que, juntamente com o Delphi, são fornecidos vários exemplos - projetos *DEMOS* englobando praticamente todos os tipos de componentes - que representam por si só uma vasta gama de informações destinadas àqueles que pretendem se aprofundar mais no uso desta ferramenta.

A programação modularizada

Um fator realmente impulsionador na criação de programas com o Delphi é o amplo uso da tecnologia de componentes. Cada componente do Delphi é criado 'apenas' utilizando o Object Pascal (em algumas partes, utiliza também uma pequena porção de assembly), servindo-se de *hierarquias*, onde existem componentes mais genéricos (componentes *pais*) que formam as bases da VCL *(Visual Component Library* - Biblioteca de Componentes Visuais do Delphi) e componentes *filhos*, que herdam características de seu pais, adicionadas novas propriedades e novos eventos.

Entretanto, você não precisa saber como foi escrito o código que define o comportamento de um botão: não é necessário saber porque e nem como ocorre todo o tratamento dado pelo Windows a cada componente para poder utilizá-lo. E melhor, se desejar utilizar o botão, não precisará escrever nenhum código para criá-lo ou desenhá-lo no seu vídeo; bastará arrastá-lo da paleta e excluí-lo: já estará tudo feito. É só codificar (programar) o comportamento dele frente aos eventos. Isso sim, é tarefa do programador.

Desta forma, um programador (algo que se você ainda não o é, será!) poderá criar inúmeros programas sem a necessidade de dar maiores aprofundamentos no conhecimento da linguagem Object Pascal e do próprio Windows.

Isto só é necessário caso você deseje criar um novo componente para adicionar ao conjunto de componentes do Delphi como, por exemplo, um componente para acessar a porta paralela do PC. (Nota: dentre os componentes que acompanham o Delphi não há nenhum com esta finalidade).

Você pode construir um novo componente, mas não é necessário. A modularidade proporcionada pela linguagem Object Pascal torna transparente a você e ao Delphi o uso de componentes desenvolvidos por outros programadores para realizar o seu projeto.

E é isso o que faremos: utilizaremos um componente *terceirizado* para o acesso à porta de conexão da impressora.

Utilizando componentes terceirizados

No projeto prático deste capítulo faremos a primeira conexão PC/mundo exterior: iremos acessar LEDs e chaves através da porta paralela. Como foi mencionado, o Delphi não possui em seu conjunto de componentes nenhum que possibilite acessar a porta paralela (a saída da impressora). Será necessário utilizarmos um componente à parte, o IOport, que deverá ser instalado no ambiente do Delphi.

Este componente está *disponível gratuitamente para download no site da Editora SABER*. O mesmo ocorre com o código fonte e o arquivo executável do projeto que será visto mais adiante.

Em suma, um componente é um conjunto de arquivos contendo definições, códigos e recursos de forma a instruir o Windows a manipular convenientemente eventos e propriedades agrupadas em torno de uma estrutura concisa.

A instalação de componentes desenvolvidos por outros programadores é um processo bastante simples, e consiste em copiar os arquivos que definem componentes para um subdiretório que deve ser criado dentro do diretório onde foi instalado o Delphi.

Procedimento: Por motivos de organização, criar um subdiretório dentro daquele onde está instalado o Delphi. Vou designar este novo diretório de 'MinhaLIB'. Copia-se então os arquivos do componente para este diretório e, instrui-se o Delphi a adicionar o novo componente à sua paleta, disponibilizando-o para você usar.

Componentes podem ser fornecidos *avulsos* (mais comum) ou em *pacotes.* A diferença deve ser levada em conta no momento da instalação *(Install Component/Install Package).*

Para instruir o Delphi a adicionar o novo componente avulso (que é o caso do IOport) à paleta, proceda da seguinte forma:

- Preste atenção ao formato dos nomes de arquivos do novo componente;

- Feche o form e o Code Editor aberto;

- Acesse o menu Component > Install Component...

- Abre-se uma caixa de diálogo chamada 'Install Component'. Você deve indicar qual é o arquivo de código fonte do componente (o ***.pas***) caso ele esteja disponível; senão, utilize o arquivo compilado do componente (o ***.dcu***). Para indicar o arquivo, dê um

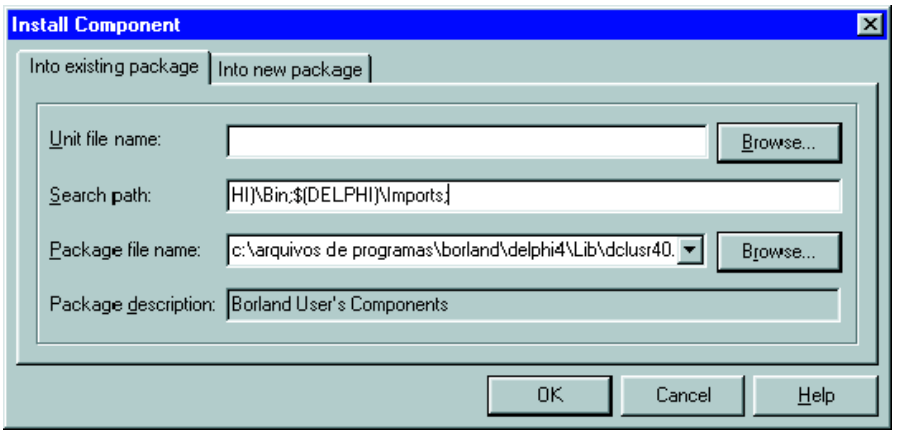

Fig. 2 - A janela *'Install Component'*

click no botão Browse ao lado e percorra, através da caixa de diálogo padrão de abertura de arquivo, todo o caminho até localizar o arquivo - que se você fez como mencionado anteriormente, vai estar dentro do diretório de instalação do Delphi/MinhaLIB. Localizando-o, selecione-o e mande abrí-lo, e você estará de volta à tela mostrada na figura 2. Dê um click em [OK] e responda afirmativamente (Yes/OK) às próximas perguntas.

Feche a janela 'Package' que se abre, respondendo sempre afirmativamente (Yes/OK).

Neste ponto o componente já estará instalado na paleta de componentes, na aba 'e-comp' e pronto pra ser utilizado. É só abrir um novo projeto e utilizá-lo. Para tanto, o faremos através de um projeto prático usando a porta paralela, tornando-se necessário um conhecimento básico sobre a mesma (vide artigo nesta mesma edição).

A montagem prática

Passaremos agora à implementação de um projeto prático de modo a fixarmos os novos conceitos, utilizando Delphi para acessarmos a porta paralela através do componente para acesso ao I/O do PC, o IOport, que a esta altura já deve estar instalado em seu ambiente Delphi. Se você ainda não instalou o componente, faça-o agora.

Os Componentes Label, SpeedButton e BitBtn

Como estes componentes são muito utilizados, vejamos algumas características deles. A função do label é geralmente de identificação: ele geralmente é usado para informar do que se trata um determinado campo. Veja o label com a caption 'Endereço Base' no projeto mais adiante. Sua função é apenas identificar, e para tanto a propriedade mais destacada é o Caption, que são os caracteres que aparecem na tela. Geralmente um label é definido em tempo de desenvolvimento e não é alvo de nenhuma alteração nas suas propriedades em run-time.

O SpeedButton e o BitBtn são dois tipos de botões, com algumas diferenças, mas quase sempre intercambiáveis. As propriedades preponderantes são o Caption e o Glyph, (alterar para) aquelas imagens que às vezes são mostradas nos botões. Para mostrar alguma imagem, você pode indicar através do Object Inspector o arquivo no formato BMP ou ainda, no caso do BitBtn, selecionar alguma imagem padrão, embutida na propriedade Kind. Observe a figura 3.

O Componente RadioGroup

Utilizaremos um componente bastante comum no Windows: os botões de seleção em grupo, denominados pelo Delphi de *RadioGroup.* O RadioGroup funciona da seguinte forma: ele exibe uma lista de itens com o *check,* onde cada um deles possui um índice, sendo que apenas um dos itens pode ser selecionado por vez, útil portanto, para indicar seleções exclusivas. Na figura 4, exemplificamos.

Para verificar como funciona o componente, inicie um novo projeto e coloque um RadioGroup no form.

O rótulo do componente é definido na propriedade Caption, e para definir os itens acesse através do Object Inspector a propriedade 'Items' dando um duplo clique sobre o conteúdo, fazendo aparecer o 'String list editor'. É neste editor que define os itens do componente. É só digitar um item por linha (é como um editor qualquer: teclar Enter para nova linha), e clicar OK quanto terminar de digitar todos os itens.

O Delphi relaciona cada item com um índice (onde o primeiro item corresponde ao índice 0), e isto fica disponível através da propriedade 'ItemIndex' do componente.

Note que a propriedade ItemIndex neste momento é –1. Isto significa que nenhum elemento está selecionado. Altere-a para 0 (zero) e você verá que agora o primeiro elemento do RadioGroup está selecionado. Altere

Fig. 4 - O componente RadioGroup.

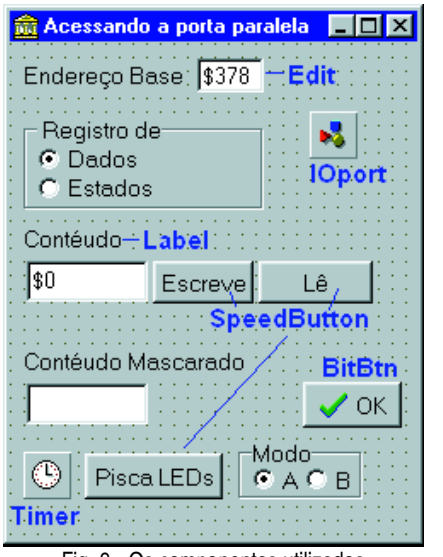

Fig. 3 - Os componentes utilizados

para 1 e o segundo elemento será selecionado. Isto basta para vermos o funcionamento do RadioGroup: iremos utilizar o RadioGroup para indicar qual registro da porta paralela será lido/escrito.

E será simples, pois basta utilizarmos a informação de qual índice do RadioGroup que está selecionado, e assim fazer leitura/escrita no endereço conveniente.

O Componente Timer

Outro componente muito útil é o Timer: ele, quando habilitado (propriedade *enabled = True)*, gera eventos em si mesmo, obedecendo a intervalos de tempo predeterminados (propriedade *interval* - dada em milissegundos).

Assim, você pode, por exemplo, soar um bip uma vez por segundo ou a cada 10 segundos, simplesmente programando uma linha no manipulador de eventos do timer. O caso mais comum é quando há a necessidade de se ter a execução de uma tarefa a intervalos determinados de tempo, por exemplo: a aquisição e *plotagem* dos dados de um sensor remoto a cada 500ms para a construção de uma curva em função do tempo.

Método

Já vimos a definição de propriedade e de evento, mas, e quanto a 'método'?

Método é um procedimento intrínseco ao componente com a finalidade de realizar uma determinada operação, podendo receber ou não um parâmetro. Exemplificando: analogamente ao ser humano, pode-se considerar métodos do 'componente' homem:

```
Homem1.Respirar; {Método sem parâmetro explícito}
Homem1.Beber(Café); {Parâmetro: CAFÉ }
Homem1.Beber(Água); {Parâmetro: ÁGUA }
```

Um método só é acessado via código, (as propriedades podem ser acessadas via código ou via Object Inspector), e deve ser da seguinte forma, s*e houver parâmetros*:

Componente.Método(Parâmetro1, Parâmetro2,...,ParâmetroN)

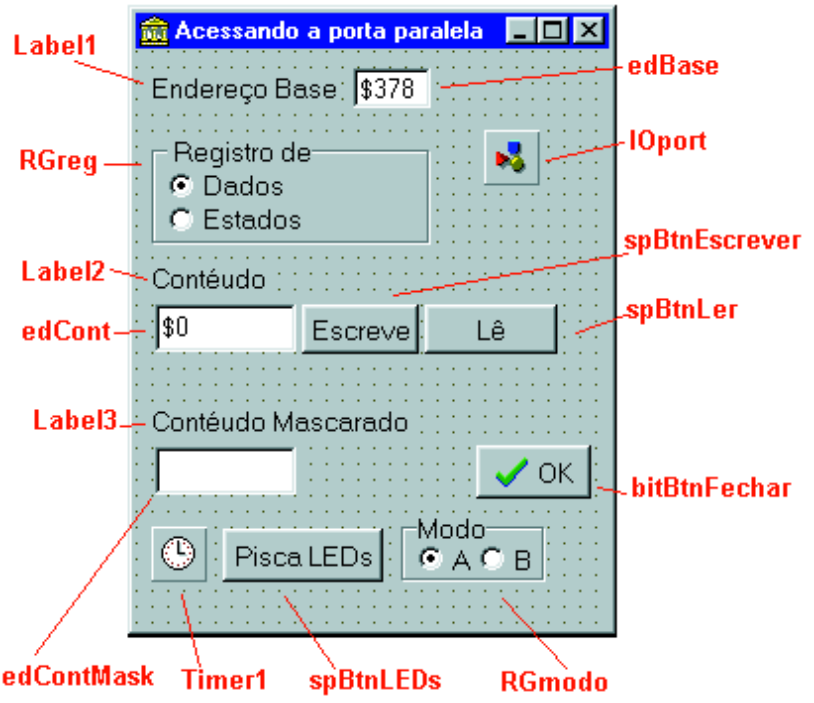

Fig. 5 - A definição visual do projeto - os componentes e seus nomes

Como funciona o componente IOport

Para fazer o acesso à porta paralela, o componente possui as seguintes propriedades:

PortAddress - Define o endereço a ser acessado;

PortData - Define o dado do acesso: se você for escrever no endereço especificado por PortAddress, este dado é que será escrito lá. Se for ler do endereço especificado, após a leitura, esta propriedade conterá o dado lido.

E para ler ou escrever, o componente dispõe de alguns métodos, e usaremos os seguintes:

Write - Escreve o conteúdo da propriedade PortData no endereço dado pela propriedade PortAddress;

Read - Lê o conteúdo do endereço dado pela propriedade PortAddress e coloca o mesmo na propriedade PortData;

InvBit(n) - Lê o conteúdo do endereço indicado por PortAddress, inverte o bitN, e escreve no mesmo endereço. Execute o programa e analise a rotina do Efeito1.

Finalidade

O programa consiste em possibilitar o manuseio da porta paralela através de comandos diretos e também o acesso programado.

Isto será feito através das propriedades e métodos do componente IOport instalado. Como vimos no artigo anterior, tendo a idéia em mente, o primeiro passo é definir a interface visual. De acordo com a figura 5, serão necessários vários componentes:

3 Labels, 3 Edits, 4 Botões (SpeedButtons e BitBtn), 2 RadioGroups, 1 Timer e 1 IOport.

(Estão nas abas *Standard*, *Additional*, *System* e *e-comp*)

Arrastando e ajustando as propriedades

Quando se arrasta um label da paleta para o form, o seu caption (rótulo) será por default, igual ao nome do componente. Assim, ao colocar o primeiro label no form, ele terá o nome padrão de 'Label1' e o seu Caption também será 'Label1' - confira isto no Object Inspector.

Você deve então alterar a propriedade Caption para exibir o rótulo que quer. Então, como base na figura 5, um a um, vejamos:

Inicie um novo projeto (File/New Application). O form (Form1) estará vazio, então ponha um label no form (arraste-o da paleta de componentes - o Delphi automaticamente o nomeará 'Label1'), altere seu caption para 'Endereço Base' (faça-o através do Object Inspector - aba Properties); coloque um edit e altere a sua propriedade Text para '$378' e altere também o seu nome (propriedade 'name') para 'edBase' - atenção: este edit será utilizado para armazenar o endereço base da porta paralela, portanto, certifique-se que é realmente este o endereço ou se é $278 (vide artigo sobre a porta paralela).

Adicione os demais labels e edits como na figura 5, alterando seus nomes conforme mostrado na figura.

Ao incluir o edit 'edContMask', altere (via Object Inspector) a sua propriedade Enabled para False, pois como ele será apenas para mostrar o conteúdo mascarado do dado lido da porta paralela, não deverá aceitar que o usuário atue diretamente sobre ele - para tanto, basta desabilitá-lo através da propriedade Enabled.

Adicione também os dois RadioGroups, definindo seu rótulos (propriedade *Caption*) e itens (propriedade *Items*) como mostrado na fig. 5, e altere de -1 para 0 a propriedade *ItemIndex* de ambos, fazendo com que o primeiro item de cada fique selecionado.

Apenas para facilitar na identificação quando escrevermos o código, altere os nomes dos RadioGroups para 'RGreg' e 'RGmodo', respectivamente, conforme os Captions que você já alterou. Adicione também um componente Timer, e altere as propriedades Enabled para *False* (isto significa que ele estará inicialmente desabilitado) e *Interval* para 100: significa que quando ele for habilitado (Enabled := True;) via algum evento, ele executará a sua rotina de evento (onTimer) a cada 100 milissegundos. Ponha os demais botões (SpeedButtons e BitBtn), Edits e Labels, alterando as propriedades deles conforme mostrado na figura 5. Lembre-se: No caso dos botões e dos labels, o texto que aparece no componente é o conteúdo da propriedade 'Caption', enquanto que no caso do componente Edit, é o conteúdo da propriedade 'Text' - veja no Object Inspector.

Para o caso do botão com caption 'Escreve', renomeie-o para 'spBtnEscrever'; o com caption 'Lê' para 'spBtnLer', o com caption 'Pisca LEDs' para 'spBtnLEDs'.

Por fim, para o botão com rótulo 'OK' - trata-se de um BitBtn: este é um outro tipo de botão, bastante semelhante ao SpeedButton, com propriedades mais específicas. Adicione um ao form, acesse sua propriedade *Kind*, alterando para bkOK - ele define automaticamente a imagem bitmap padrão para botão 'OK' na propriedade *Glyph* do botão, e a exibe. Altere à vontade a propriedade Kind para notar as configurações padrão. Se desejar alterar a imagem, faça através da propriedade Glyph. Esta propriedade permite exibir mais de uma imagem, dependendo do estado do botão: habilitado ou desabilitado - altere a propriedade NumGlyph de 2 para 1 e veja o que acontece! Altere seu nome para 'bitBtnFechar'.

Tendo criado a interface, o próximo passo é codificar convenientemente os eventos necessários. Para codificar os eventos, selecione o componente e, através do Object Inspector, na aba Events, dê um duplo clique sobre o evento desejado. Ao fazer isto o Delphi criará o esqueleto do manipulador do evento, então é só codificar o evento. Veja o código de cada manipulador na listagem a seguir.

Procure no código os eventos utilizados, com as finalidades descritas.

> ***Nota*** - É de uso comum os programadores nomearem os componentes da seguinte forma: uma palavra significativa prefixada de uma parte do nome padrão do componente. Veja por exemplo, o nome dado acima, 'edBase', 'ed' por se tratar de um componente Edit, e 'Base', pois este edit servirá para conter o endereço base da porta paralela. Dessa forma, fica fácil relacionar o componente com a sua função no programa. Assim, a mudança do nome dos componente é apenas para facilitar a compreensão por parte do programador, que ao ler/reler o código fonte contendo nomes sugestivos, terá maior facilidade na compreensão do algorítmo. Outro fato a considerar é que normalmente não se altera o nome de componentes tipo 'Label', pois raramente eles são acessados em tempo de execução. Portanto, não é sequer mencionado no código fonte. Observe isto no programa que faremos no projeto prático.

Você não precisa se preocupar com a parte de código gerado automaticamente, basta copiar os códigos para as procedures que farão o tratamento dos eventos. Inclusive não é necessário copiar os comentários, pois eles são apenas para facilitar a sua compreensão. Os textos entre '{ }' e após '//' são comentários feitos no Code Editor com a finalidade de tornar mais legível o código.

O Circuito

O circuito apresentado na figura 6 destina-se à visualização da manipulação da porta paralela: um conjunto de 8 LEDs conectados aos pinos referentes aos 8 bits do registro de dados, fornecendo a confirmação visual da escrita neste registro da porta.

Apresenta ainda 5 interruptores, a fim de ler o conteúdo do registro de estado. Não é necessária alimentação externa, pois o próprio circuito da porta é capaz de fornecer os níveis necessários. Como a leitura em aberto do registro de estado resulta em nível alto, a menos do bit /S7, apenas a conexão eventual via chave com o terra do circuito da porta é suficiente para se obter todos os estados possíveis.

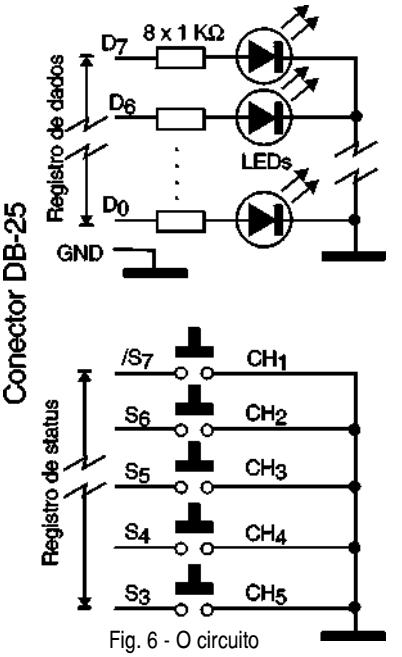

Fig. 6 - O circuito

Testando

Monte o circuito mostrado, confira, e conecte na porta paralela. Execute o comando 'RUN' ou a tecla de função F9. Veja a figura 7, a seguir.

O seu programa estará sendo executado. Faça escrita e leitura nos ports de Dados e de Status. Note que como o bit7 do registro de Status é lido invertido, é conveniente fazer o seu mascaramento de forma a tornar este detalhe irrelevante, e facilitar o tratamento do byte lido. Isto é feito no manipulador do evento spBtnLerClick.

Observe que:

Se você digitar 80 no Edit edCont, o Delphi trata o 80 como decimal, logo se mandar escrever 80 (decimal) na saída do porta, Data Register, acenderá os bits D6 e D4: 80d=64d+16d; entretanto, se digitar $80 no Edit edCont, ele interpreta como hexadecimal por causa do '$', e agora se o $80 for enviado para o port de Dados, acenderá o bit D7 apenas: $80=80h=10000000b. Por fim, após compreender totalmente o código, faça modificações e experimentos. Um destes consiste em gerar novos padrões de saída para o evento do timer, inclusive com a alteração da propriedade *interval* do mesmo.

PROGRAMA

```
{ ....  Código gerado automaticamente pelo Delphi ...
Acima deste ponto, estará a parte do código criado automatica-
mente pelo Delphi: as bibliotecas utilizadas, o código para criar os
componentes que você arrastou para o form, etc.}
...
var
  Form1: TForm1;        // Variável criada pelo Delphi.
  i    : Integer;       //* <- Contador - variável global ao Form
implementation         // Você deve criar a variável i (basta digi-
                       // tar a linha marcada com asterisco
{$R *.DFM}             // (Digite abaixo da variável Form1)
                       //... Evento gerado na criação do form
                       //(run-time).
                       // Define o conteúdo do edBase como
                       // o endereço da porta.
procedure TForm1.FormCreate(Sender: TObject);
begin                  //Como a propriedade Text é uma string,
                       //converte-a para inteiro
  IOport.PortAddress := StrToInt(edBase.Text);
end;


                       // .....  Botão 'OK' - Fecha a aplicação
procedure TForm1. bitBtnFecharClick(Sender: TObject);
begin
  Close;               // Close: encerra a aplicação
end;

procedure TForm1.spBtnLerClick(Sender: TObject);
begin

                       // Para fazer a leitura, deve-se primeiro
                       // definir qual será o endereço
                       // Para isto, converte-se o valor
                       // da propriedade Text do edBase (que
                       // contém o endereço base da porta
                       // paralela) e soma-se a ele o índice
                       // do item selecionado no RadioGroup
                       // Rgreg, pois se o item selecionado
                       // for o primeiro ('Dados'), o valor ItemIndex
                       // será igual a zero, que
                       // somado ao endereço base da porta,
                       // dará justamente $378, ou seja, a
                       // leitura será feita no registro de Dados.
                       // Se o item selecionado for o segundo
                       // ('Dados'), o valor ItemIndex
                       // será igual a 1, que somado ao
                       // endereço base da porta, dará $379,
                       // ou seja, a leitura será feita no registro
                       // de Estado.

 IOport.PortAddress := StrToInt(edBase.Text) + RGreg.ItemIndex;
 IOport.Read;
                       // Lê valor do registro ($378 ou $379),
                       // converte p/ Hexa edCont (veja help
                       // do Delphi para a função IntToHex)
                       // Basta, estando no Code Editor,
                       // pressionar F1 sobre a palavra

edCont.Text := '$'+ IntToHex((IOport.PortData),2);
                       // A função IntToHex converte um valor
                       // inteiro - no caso, o valor inteiro
                       // correspondente aos bits lidos da porta
                       // paralela. A sintaxe da função é
                       // IntToHex(Valor_Inteiro,Nº.deDígitos)
                       // Veja a descrição completa através
                       // no Help do Delphi - tecla F1
                       // Se a leitura for feita do registro de
                       // 'Dados', basta apenas
                       // converter para Hexa, senão, converte
                       // e mascara os bits invertidos
                       // A máscara XOR é feita devido ao fato de
                       // que alguns bits da porta são lidos
                       // invertidos. (Vide artigo referente
                       // à porta paralela)
  if (RGreg.ItemIndex = 0) then
    edContMask.Text := '$'+ IntToHex(IOport.PortData,2)
  else
   edContMask.Text := '$'+ IntToHex($80 XOR IOport.PortData,2);
end;

procedure TForm1.spBtnEscreverClick(Sender: TObject);
begin
                       // Note que a propriedade PortData requer
                       // um valor inteiro, logo, é necessário
                       // converter o conteúdo do edCont.Text -
                       // o edit que possui o valor que deve ser
                       // escrito na porta -  que é string
IOport.PortData := StrToInt(edCont.Text);// edCont.Text -> Dado
IOport.Write;          // Escreve o dado
end;

procedure TForm1.RGregClick(Sender: TObject);
begin
  { Define o endereço da porta a partir do conteúdo do edBase +
o valor do índice selecionado do RGreg:  $378 ou $379}
IOport.PortAddress := StrToInt(edBase.Text) + RGreg.ItemIndex;

if RGreg.ItemIndex = 1 then      // Se o item 'Estado' estiver
   spBtnEscrever.Enabled := False //  selecionado, desabilita o
else                   // botão btnEscrever, caso
   spBtnEscrever.Enabled := True;    // contrário, habilita
end;

                       // Ativa/Desativa o timer
procedure TForm1.spBtnLEDsClick(Sender: TObject);
begin
  IOport.PortAddress := $378;
                       // Seta endereço da porta (Dados)
  IOport.PortData := $0;      // Seta dado a ser enviado
  IOport.Write;        // Escreve dado D7..D0 = nível 0
  i:=0;                // Atribui 0 ao contador
  Timer1.Enabled := Not Timer1.Enabled; // Habilita o Timer
end;


procedure TForm1.Timer1Timer(Sender: TObject);
begin
 IOport.PortAddress := $378;
                       // Seta endereço da porta (Dados)

 case RGmodo.ItemIndex of
                       // Se o ItemIndex do RadioG2 = 0
  0: begin             // (se 1o. item está selecionado):
     IOport.InvBit(i);   //  -> Inverte o bit ordem 'i'
     if i = 8 then i:=-1; // Se..., reinicia o contador i
     end;               // Vide efeito na fig. 7
                        // Efeito1: loop de 8 passos
                        // 2o. Efeito:
                       // Se RadioG2.ItemIndex = 1
                        // (se 2o. item selecionado)
  1: begin             // então verifica contador e
      case i of        // define máscara de bits:
        0: IOport.PortData := $81; // 10000001  0 - LED Off
        1: IOport.PortData := $42; // 01000010  1 - LED On
        2: IOport.PortData := $24; // 00100100
        3: IOport.PortData := $18; // 00011000
        4: IOport.PortData := $24; // 00100100
        5: IOport.PortData := $42; // 01000010
       end;
       IOport.Write;           // Escreve máscara na porta
       if i=5 then i:=-1;      // Reinicia o contador
      end;                    // Efeito2: loop de 6 passos
   end;


    if i>10 then i:=-1;              // Segurança contra estouro
    inc(i);                    // Incrementa o contador i
end;
procedure TForm1.RGmodoClick(Sender: TObject);
begin
  i := -1; // Dá um Reset no contador i
end;
end.  // Fim do código
```

Formas de acesso ao 'mundo exterior' pelo PC

A forma de acesso apresentada, onde utilizamos a porta paralela, é uma das mais fáceis de obter, entretanto, dependendo da aplicação, existem limitações tais como número de bits disponíveis, número de vias necessárias para alcançar maiores distâncias, etc.

Com a finalidade de contornar possíveis limitações, pode-se utilizar outras formas de acesso a um circuito externo: a porta serial ou até mesmo o barramento do PC, sendo que esta última requer maior cautela, por ser necessária a abertura do equipamento e por levar os sinais externos até muito próximo dos circuitos do PC.

Entretanto, em todas as técnicas, você pode e por medidas de segurança, é aconselhável lançar mão de *técnicas de interfaceamento* apropriadas na conexão do PC com circuitos externos (veja mais em 'Técnicas de Interfaceamento' - SE 313 -fevereiro/99).

No próximo artigo nos aprofundaremos mais no uso dos componentes nativos do Delphi, e veremos como desenvolver um sistema de aquisição e manipulação de dados.

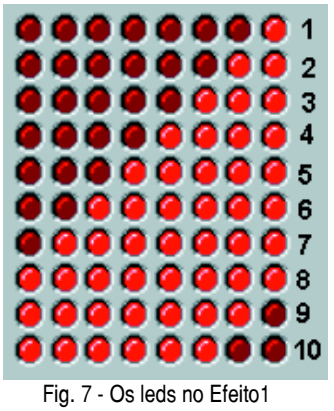

Fig. 7 - Os leds no Efeito1

Notas:

1.Esta montagem utiliza um componente de acesso ao hardware do PC, e portanto, você deve tomar os devidos cuidados para não colocar em risco a integridade do equipamento. Deverá checar os circuitos e diagramas que venha a montar ANTES de conectar. Lembre-se que o PC é a parte mais cara do projeto!

2.O componente de acesso à porta paralela utiliza características do padrão Windows95/98, diferentes dos padrões do WindowsNT. Dessa forma, este projeto é incompatível com o sistema operacional WindowsNT. ■