TUTORIAL
EXPANSÃO DE ENTRADAS DIGITAIS

Autor: Luís Fernando Patsko
Nível: Intermediário
Criação: 13/01/2006
Última versão: 18/12/2006

Maxwell Bohr
Instrumentação Eletrônica

http://www.maxwellbohr.com.br
contato@maxwellbohr.com.br

PdP
Pesquisa e Desenvolvimento de Produtos

http://www.automato.com.br
atendimento@automato.com.br

1 - Introdução

Nesse tutorial, veremos um sistema de expansão de portas de entrada digitais. Tal sistema é baseado numa aplicação onde é necessária a utilização de 40 sensores magnéticos. Tais sensores magnéticos comportam-se como chaves digitais que, ao invés de serem acionadas pelo contato mecânico, são acionadas pelo campo magnético gerado por um imã.

O sistema de expansão de entradas digitais conta com uma placa que fará a interface com os sensores e um programa criado especificamente para realizar o controle dessa placa. Tal sistema pode ser utilizado com o MEC1000 ou com o *Módulo de Entradas, Saídas e Servomotores* do KDR5000.

Baseado no funcionamento desse sistema, é possível elaborar sistemas de expansão de entradas mais simples, de 16 entradas, ou maiores, de até 128 entradas, de acordo com a necessidade. Além disso, os sensores magnéticos podem ser substituídos por outros componentes, como chaves digitais comuns ou pares ópticos.

2 – Placa de expansão

A placa conta com 5 registradores 74 573. Podem ser utilizados circuitos integrados das famílias HC, LS ou outras, desde que compatíveis com os níveis de tensão utilizados. Cada chave ou sensor deverá ser ligado a uma entrada de um registrador.

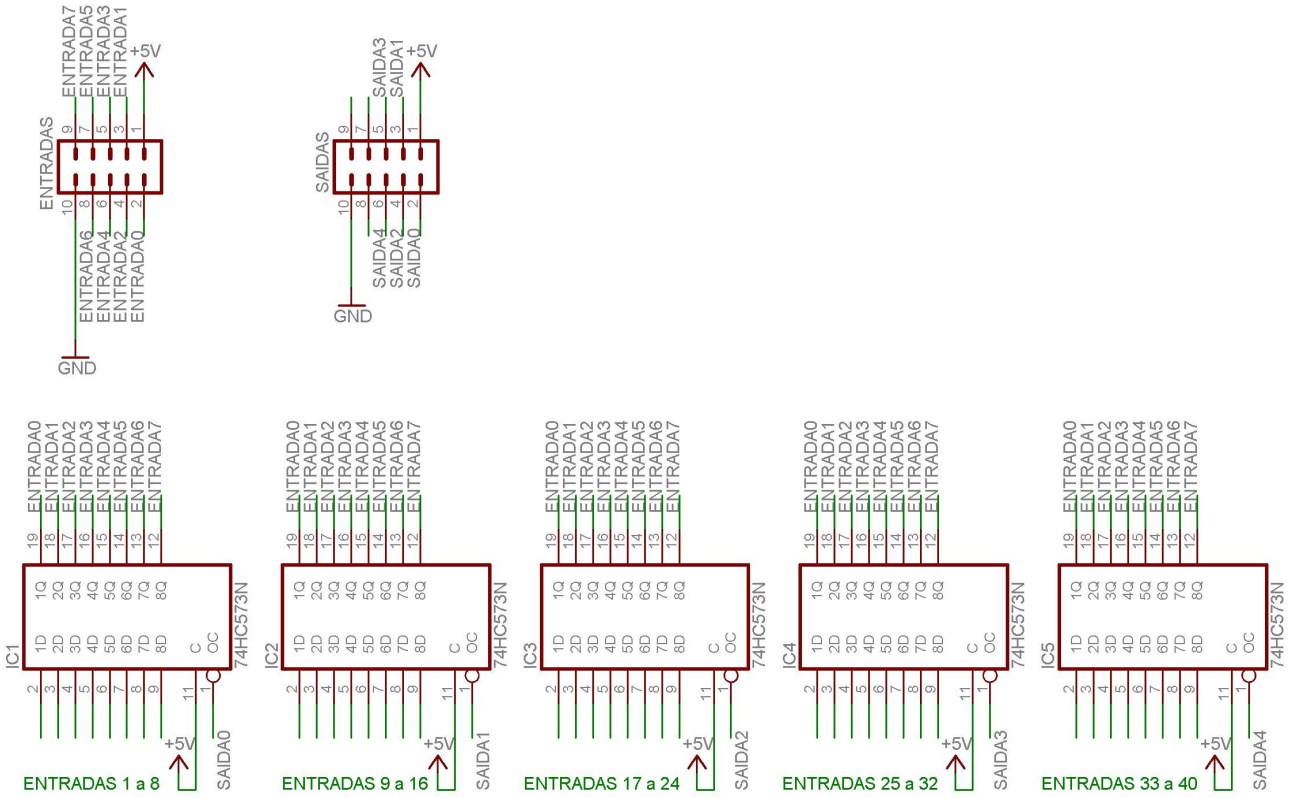

Figura 1: Esquema da placa de expansão de entradas digitais.

O conector Entradas presente nessa placa deverá ser ligado ao conector Entradas 0 do

MEC1000 ou do *Módulo de Entradas, Saídas e Servomotores,* enquanto que o conector Saídas deverá ser ligado ao conector Saídas 0. Os conectores são do tipo Header de 10 vias, semelhantes aos conectores do MEC1000 e o do KDR5000. As ligações deverão ser feitas através de cabos flat.

Serão utilizados as oito entradas digitais disponíveis no conector Entradas 0 e cinco das oito saídas presentes no conector Saídas 0. Cada uma das cinco saídas fará o controle de um dos registradores.

Ao manter uma das cinco saídas digitais no nível 0, será feita a seleção de um dos registradores. Então, será possível verificar nas entradas digitais o estado dos sensores ligados ao registrador selecionado.

O programa de controle deverá ser elaborado para fazer uma leitura seqüencial das chaves. Primeiro, deve ser selecionado o registrador 1 e serem verificados as oito chaves ou sensores ligados a ele. Esse registrador deverá ser desativado, para que o próximo seja selecionado. Será feita então uma leitura sequencial, de modo a permitir a leitura de oito entradas de cada vez. A tabela abaixo relaciona os níveis lógicos das saídas digitais e os sensores lidos nas entradas digitais.

Entradas a serem verificadas	Saída digital 0 (Registrador 1)	Saída digital 1 (Registrador 2)	Saída digital 2 (Registrador 3)	Saída digital 3 (Registrador 4)	Saída digital 4 (Registrador 5)
1 a 8	0	1	1	1	1
9 a 16	1	0	1	1	1
17 a 24	1	1	0	1	1
25 a 32	1	1	1	0	1
33 a 40	1	1	1	1	0

Para evitar possíveis danos aos componentes, não se deve acionar mais de um registrador simultaneamente. Apenas uma das cinco saídas digitais pode ser mantida no nível lógico 0.

Para a conexão dos sensores magnéticos ou chaves digitais, é sugerido o seguinte esquema de ligação:

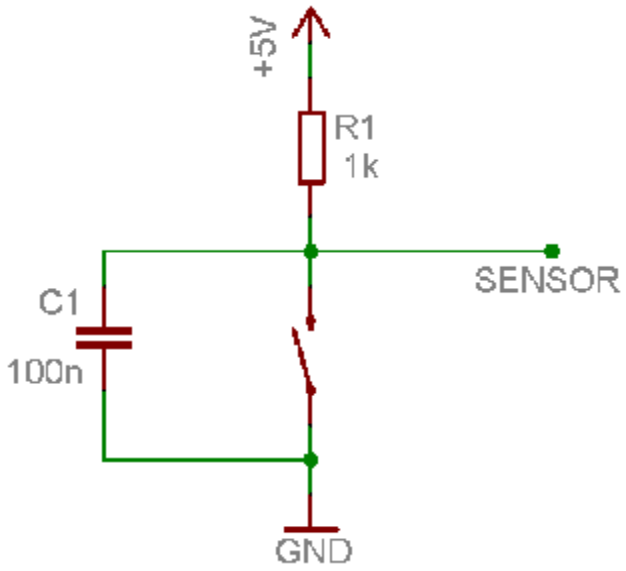

Figura 2: Sugestão de ligação de uma chave digital ou sensor magnético.

O capacitor tem a função de reduzir eventuais ruídos. Cada sensor deverá ser conectado a uma entrada de um registrador. O capacitor e o resistor podem ser montados na placa de expansão, deixando apenas o sensor fora da placa.

A imagem abaixo mostra uma sugestão de layout da placa. O layout mostra a placa vista pelo lado dos componentes. As trilhas que encontram-se em azul estarão no lado oposto, enquanto que as trilhas em vermelho devem ficar no lado dos componentes (no caso de ser utilizada uma placa dupla face). Caso seja utilizada uma placa que contém cobre em apenas um lado, uma sugestão é fazer "jumpers" com fios, no lugar das trilhas em vermelho.

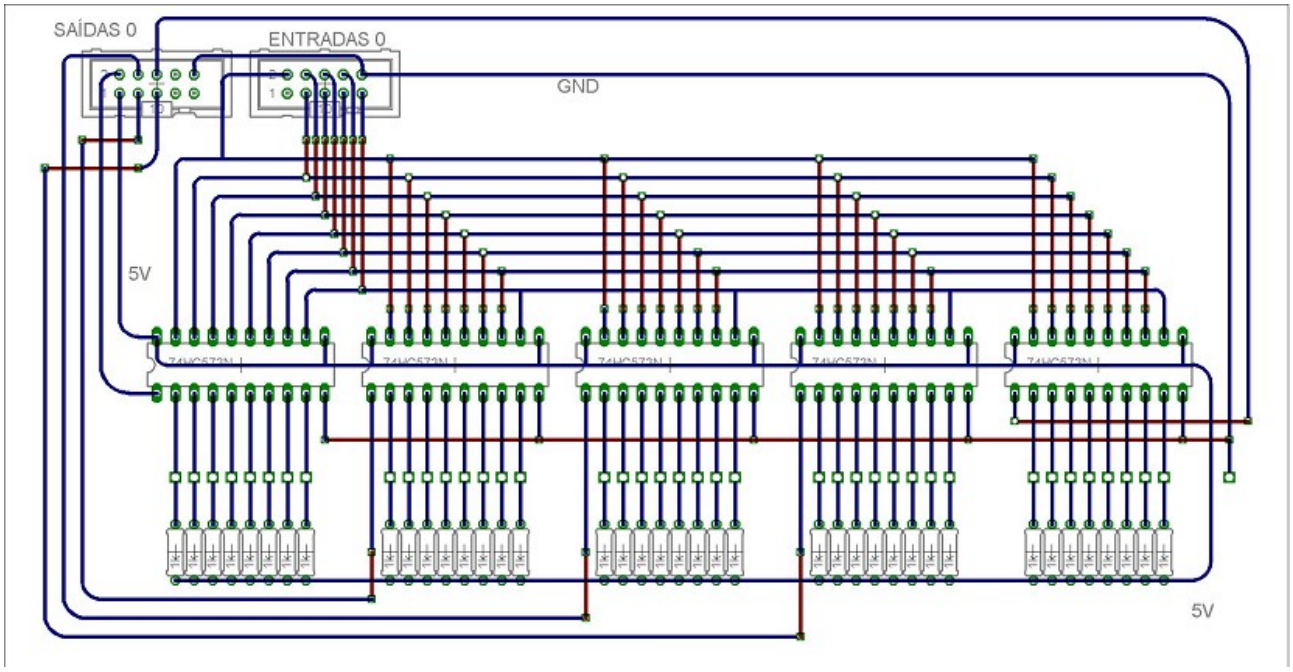

Figura 3: Sugestão de layout para a placa de expansão.

Tal placa pode ser alterada de acordo com a necessidade do projeto, adicionando ou retirando registradores do esquema. Lembre-se de que cada registrador deverá ter seu terminal OE ligado a uma saída digital.

Observe também que quanto maior for o número de entradas digitais expandidas, serão utilizadas mais portas de saídas digitais. Dessa forma, nessa aplicação de 40 entradas, serão utilizadas 5 saídas digitais, enquanto que numa aplicação de 16 entradas seriam utilizadas apenas 2 saídas digitais. No MEC1000, o sistema pode ser expandido para até 64 entradas, o que exigirá 8 portas de saídas digitais. Já o KDR5000, por possuir 16 saídas digitais, possibilita que o sistema seja expandido até 128 entradas.

3 – Software

O controle da placa de expansão será feito através de um programa específico. É possível utilizar o programa de demonstração do MEC1000 para fazer as leituras, porém um programa criado especificamente para essa finalidade simplifica algumas tarefas de leitura. O

programa será desenvolvido de forma a permitir ao usuário a leitura de uma das 40 entradas, ou a leitura simultânea de todas as entradas.

Primeiramente, vamos elaborar a interface gráfica do programa. Vamos utilizar o projeto criado no tutorial Base que já nos fornece algumas funcionalidades interessantes. Para isso copiamos o projeto daquele tutorial e em cima dele vamos adicionar alguns componentes gráficos extras.

Essa interface possuirá um ComboBox, através do qual será possível fazer a seleção da entrada a ser lida; e dois Buttons, sendo que um faz a leitura da entrada selecionado no ComboBox, e o outro faz a leitura de todas as entradas.

Vamos primeiramente modificar a propriedade Caption do Form principal, alterando-a de “Projeto Base” para “Expansão de entradas digitais”. A seguir, vamos colocar os componentes ComboBox e Button na interface do programa. Tais componentes se encontram na aba “Standard”.

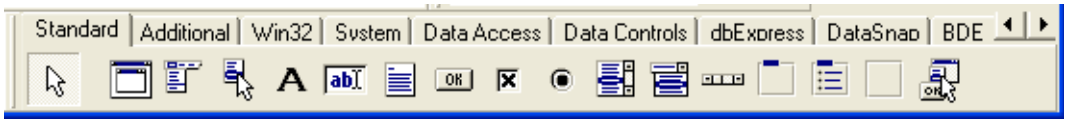

Figura 4: Aba "Standard" da Barra de componentes.

O componente ComboBox possui o seguinte ícone:

Figura 5: Ícone do componente ComboBox.

E o componente Button possui o seguinte ícone:

Figura 6: Ícone do componente Button.

Vamos agora alterar as propriedades desses componentes. As configurações do ComboBox devem ser alteradas para os parâmetros a seguir:

Name	=	ComboBoxPorta
Style	=	csDropDownList
Items.Strings	=	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ,15 ,16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40
ItemIndex	=	0

Já as propriedades dos Buttons devem ser alteradas para os valores a seguir:

Name	=	ButtonLer
Caption	=	Ler

Name	=	ButtonLerTodos
Caption	=	Ler todos

Nesse momento, a interface do programa estará finalizada. Ele terá a seguinte aparência:

Figura 7: *Interface do programa.*

Agora, vamos implementar as linhas de código responsáveis por fazer a leitura das entradas. Primeiro, vamos elaborar um método chamado "InitEntradaDeExpansao". Esse método envia o valor 255 para as portas de saídas digitais do MEC1000, o que fará com que todas elas fiquem no nível lógico 1. O objetivo disso é evitar que dois ou mais registradores estejam selecionadas ao mesmo tempo, o que poderia danificar a placa de expansão. Esse método então servirá como uma medida de proteção do nosso sistema.

A declaração do método deverá ser feito desse modo:

```
// Inicializa porta de expansão de entradas digitais do MEC1000
procedure TFormMain.InitEntradaDeExpansao();
begin
          // Envia 255 para a porta para evitar que 2 ou mais entradas
          // estejam selecionadas ao mesmo tempo, o que poderia
          // danificar o equipamento
          kit.DigitalPortWrite(0, 255);
end;
```

Esse método deverá ser chamado em dois momentos do nosso programa: na inicialização do programa e quando o ComboBox da serial for alterado. No método FormCreate, que já havia no programa Base, devemos adicionar a seguinte linha de código. Isso garante que as saídas digitais fiquem no nível 1 e nenhum registrador esteja selecionado.

```
procedure TFormMain.FormCreate(Sender: TObject);
begin
  // Cria instância de TKit
 kit := TKit.Create;

  // Recupera todas a portas seriais instaladas no sistema
  kit.GetInstalledSerialPorts(ComboBoxSerial.Items);

  try
    if (ComboBoxSerial.Items.Count>0) then
    begin
      // Seleciona o primeiro ítem
      ComboBoxSerial.ItemIndex := 0;
```

```
      // Tenta se conectar com um Kit
      kit.OpenCommunication(ComboBoxSerial.Text);

      // Inicializa porta de expansão
      InitEntradaDeExpansao();
    end;
  except
  end;
end;
```

No método ComboBoxSerialChange, que também já existia no programa Base, vamos adicionar a seguinte linha de código. Assim, quando o ComboBox for modificado, o programa tentará se comunicar com um kit através da nova porta serial selecionada, e também chamará o método InitEntradaDeExpansao, garantindo que os registradores não estejam selecionados.

```
procedure TFormMain.ComboBoxSerialChange(Sender: TObject);
begin
  // Fecha a comunicação caso essa já estivesse aberta
  kit.CloseCommunication();

  // Inicia comunicação com serial selecionada no ComboBox
  kit.OpenCommunication(ComboBoxSerial.Text);

  // Inicializa porta de expansão
  InitEntradaDeExpansao();
end;
```

Agora, vamos elaborar uma função chamado LerEntradaDeExpansao. Esse método será responsável por selecionar o registrador e fazer a leitura da entrada digital adequada, de acordo com a entrada selecionado no ComboBoxPorta.

Esse método tem como parâmetro uma variável chamada “bit”, que indicará qual das 40 entradas será verificada. O método retornará um valor do tipo Byte, o qual indicará o estado da entrada lida.

Primeiramente, vamos declarar nessa função uma variável do tipo Byte, chamada “leitura”. Essa variável armazenará o valor da leitura da porta de entradas digitais.

```
function TFormMain.LerEntradaDeExpansao(bit : Integer) : Byte;
var
  leitura : Byte;
begin


end;
```

O parâmetro “bit” será utilizado ao longo dessa função para determinar a entrada a ser lida. É interessante fazer uma verificação dessa variável, para nos assegurarmos que seu valor se encontra entre 1 e 40. Caso isso não ocorra, será exibida uma mensagem de alerta.

```
function TFormMain.LerEntradaDeExpansao(bit : Integer) : Byte;
var
  leitura : Byte;
begin
  // Verifica se número do bit está no intervalo válido
  if (bit<1) OR (bit>40) then
  Raise Exception.Create('Número do bit deve estar no intervalo de 1 à
                                                                  40.');
```

```
end;
```

O próximo passo será fazer a seleção do registrador apropriado. A placa de expansão conta com 5 registradores, sendo que cada um é responsável por um conjunto de 8 entradas. Se a entrada selecionada se encontrar entre 1 e 8, o primeiro registrador deverá ser ativado. Isso é feito atribuindo o nível lógico 0 na saída digital ligada a ele. O mesmo deverá ser feito com os outros registradores.

```
function TFormMain.LerEntradaDeExpansao(bit : Integer) : Byte;
var
  leitura : Byte;
begin
  // Verifica se número do bit está no intervalo válido
  if (bit<1) OR (bit>40) then
  Raise Exception.Create('Número do bit deve estar no intervalo de 1 à
                                                                  40.');

  // Envia o valor correto para a porta de saída
  if bit<=8 then                          // Bit de 1 à 8
  // Envia o valor 254 (11111110) para a porta
         kit.DigitalPortWrite(0, 254)
  else if bit<=16 then                    // Bit de 9 à 16
  // Envia o valor 253 (11111101) para a porta
         kit.DigitalPortWrite(0, 253)
  else if bit<=24 then                    // Bit de 17 à 24
  // Envia o valor 251 (11111011) para a porta
         kit.DigitalPortWrite(0, 251)
  else if bit<=32 then                    // Bit de 25 à 32
  // Envia o valor 247 (11110111) para a porta
         kit.DigitalPortWrite(0, 247)
  else                                    // Bit de 33 à 40
  // Envia o valor 239 (11101111) para a porta
         kit.DigitalPortWrite(0, 239);



end;
```

Com o registrador selecionado, podemos fazer a leitura da porta de entradas digitais. Isso será feito utilizando a função DigitalPortRead. Essa função retorna um valor de 0 a 255, que indica o estado de cada uma das entradas digitais. O tipo do valor retornado é um byte, que contém 8 bits. O valor de cada bit corresponde ao estado de uma entrada digital.

```
function TFormMain.LerEntradaDeExpansao(bit : Integer) : Byte;
var
  leitura : Byte;
begin
  // Verifica se número do bit está no intervalo válido
  if (bit<1) OR (bit>40) then
  Raise Exception.Create('Número do bit deve estar no intervalo de 1 à
                                                                  40.');

  // Envia o valor correto para a porta de saída
  if bit<=8 then                          // Bit de 1 à 8
  // Envia o valor 254 (11111110) para a porta
         kit.DigitalPortWrite(0, 254)
```

```
  else if bit<=16 then                  // Bit de 9 à 16
  // Envia o valor 253 (11111101) para a porta
        kit.DigitalPortWrite(0, 253)
  else if bit<=24 then                  // Bit de 17 à 24
  // Envia o valor 251 (11111011) para a porta
        kit.DigitalPortWrite(0, 251)
  else if bit<=32 then                  // Bit de 25 à 32
  // Envia o valor 247 (11110111) para a porta
        kit.DigitalPortWrite(0, 247)
  else                                  // Bit de 33 à 40
  // Envia o valor 239 (11101111) para a porta
        kit.DigitalPortWrite(0, 239);

  // Lê porta de entradas digitais
  leitura := kit.DigitalPortRead(0);


end;
```

Dessa forma, a variável leitura armazenará o estado das entradas digitais do MEC1000. De acordo com o registrador que foi selecionado, será feita a leitura de um dos blocos de 8 chaves digitais ou sensores ligados a ele. A tabela da página 3 esclarece a relação entre o valor das saídas digitais, o registrador selecionado e as entradas a serem lidas.

Porém, falta determinar o valor exato da entrada que queremos ler. A variável leitura possui o valor das 8 entradas ligadas a um registradores. Devemos extrair dessa variável apenas o valor da entrada que queremos. Se, por exemplo, quisermos ler a entrada 7, a variável leitura terá armazenado os valores das entradas 1 a 8, pois são essas entradas que estão ligados ao registrador 0.

Devemos fazer a leitura exata da entrada digital que queremos obter o valor. Isso pode ser feito através da seguinte linha de código. Esse trecho fará a extração do bit correspondente à entrada digital que queremos fazer a leitura.

```
function TFormMain.LerEntradaDeExpansao(bit : Integer) : Byte;
var
  leitura : Byte;
begin
  // Verifica se número do bit está no intervalo válido
  if (bit<1) OR (bit>40) then
  Raise Exception.Create('Número do bit deve estar no intervalo de 1 à
                                                                  40.');

  // Envia o valor correto para a porta de saída
  if bit<=8 then                        // Bit de 1 à 8
  // Envia o valor 254 (11111110) para a porta
        kit.DigitalPortWrite(0, 254)
  else if bit<=16 then                  // Bit de 9 à 16
  // Envia o valor 253 (11111101) para a porta
        kit.DigitalPortWrite(0, 253)
  else if bit<=24 then                  // Bit de 17 à 24
  // Envia o valor 251 (11111011) para a porta
        kit.DigitalPortWrite(0, 251)
  else if bit<=32 then                  // Bit de 25 à 32
  // Envia o valor 247 (11110111) para a porta
        kit.DigitalPortWrite(0, 247)
  else                                  // Bit de 33 à 40
  // Envia o valor 239 (11101111) para a porta
        kit.DigitalPortWrite(0, 239);
```

```
    // Lê porta de entradas digitais
    leitura := kit.DigitalPortRead(0);

    // Extrai apenas o bit que se deseja ler
    LerEntradaDeExpansao := (leitura SHR (((bit-1) mod 8))) AND $01;

end;
```

Dessa forma, finalizamos a função LerEntradaDeExpansao. Durante o programa, essa função será chamada, tendo como parâmetro a entrada que queremos ler. Ela então verificará esse parâmetro, fará a leitura da placa de expansão de entradas e retornará o valor correspondente ao estado da entrada que foi lida. Caso a entrada esteja em nível lógico 0, a função retornará o valor 0. Se a entrada estiver em nível 1, a função retornará o valor 1.

Falta adicionar ao nosso programa os métodos relacionados aos botões Ler e Ler Todos. Na interface gráfica do programa, vamos dar dois cliques sobre o botão Ler, o que criará as seguintes linhas de código. Elaboraremos o código a ser executado quando o botão for pressionado, durante a execução do programa.

```
procedure TFormMain.ButtonLerClick(Sender: TObject);
begin

end;
```

Vamos criar uma variável do tipo byte, que será chamada “valor”. Essa variável armazenará o valor a ser retornado pela função LerEntradaDeExpansao.

```
procedure TFormMain.ButtonLerClick(Sender: TObject);
var
  valor : Byte;
begin

end;
```

O próximo passo será chamar a função LerEntradaDeExpansão e atribuir o dado que ela retorna à variável “valor”. Como parâmetro dessa função, será enviado o item selecionado da ComboBox de seleção.

Vamos adicionar também o trecho responsável por mostrar a informação ao usuário. Utilizaremos a função ShowMessage, que tem como parâmetro o texto a ser exibido. Observe que para mostrar o valor lido, devemos utilizar a função IntToStr para que a variável “valor” seja convertida de um valor inteiro para uma string.

```
procedure TFormMain.ButtonLerClick(Sender: TObject);
var
  valor : Byte;
begin
  // Lê a entrada de expansão selecionada e armazena na variável valor
  valor := LerEntradaDeExpansao(ComboBoxPorta.ItemIndex+1);

  //Mostra o valor da entrada selecionada
  ShowMessage('O valor lido foi: ' + IntToStr(valor));
end;
```

Com essas linhas de código, ao pressionar o botão Ler, o programa fará a leitura da entrada selecionada na ComboBox e exibirá o seu valor na forma de uma caixa de mensagem.

Agora, vamos implementar o código do botão Ler Todos. Na interface, ao clicarmos duas vezes sobre o botão, será criada a seguinte linha de código.

```
procedure TFormMain.ButtonLerTodosClick(Sender: TObject);
begin

end;
```

Primeiro, devemos criar as variáveis a serem utilizadas durante a execução desse trecho de código. Além da variável "valor", que também criamos no código do botão Ler, vamos criar duas variáveis: "i", também do tipo Byte, e "str", do tipo String.

```
procedure TFormMain.ButtonLerClick(Sender: TObject);
var
  valor, i : Byte;
  str : String;
begin

end;
```

Agora, vamos elaborar o código para a leitura de todas as entradas. Vamos utilizar um laço for, que realizará uma contagem de 1 a 40. Em cada contagem, o programa fará a leitura de uma entrada digital e criará um texto, a ser armazenado na variável "str".

Observe que ao atribuirmos o valor da variável "str", adicionamos os trechos referentes à entrada que foi lida e ao seu valor, e também colocamos o seguinte trecho: "#13#10". Esse trecho faz com que seja criada uma nova linha de texto. Caso ele não seja colocado, todo o texto armazenado na variável "str" será exibido em uma única linha. Para verificar a função desse trecho, execute o programa sem ele.

Utilizamos também a função Sleep. Essa função interrompe a execução do programa durante um certo tempo, determinado em milisegundos. Nesse caso, utilizamos o parâmetro 25, o que fará com que o programa seja interrompido durante 25 milisegundos entre cada leitura das entradas digitais.

No término da contagem, utilizaremos a função ShowMessage para exibir o conteúdo dessa variável "str".

```
procedure TFormMain.ButtonLerTodosClick(Sender: TObject);
var
  valor, i : Byte;
  str : String;
begin
  // Lê todas as entradas e armazena valores em uma string
  for i:=1 to 40 do
  begin
    valor := LerEntradaDeExpansao(i);
     str := str + 'O valor do bit ' + IntToStr(i) + ' é igual à: ' +
                                          IntToStr(valor) + ''#13#10'';
    Sleep(25);
  end;

  //Exibe o valor das entradas digitais
  ShowMessage(str);
end;
```

Dessa forma, o programa estará finalizado. Basta apenas que ele seja compilado e

executado. Através desse programa, é possível fazer a leitura de 40 chaves ou sensores digitais ligados à placa de expansão de entradas digitais, porém esse número pode ser facilmente aumentado para até 128 (caso a placa seja criada com 16 registradores) ou reduzido para até 16 (se a placa for elaborada com 2 registradores apenas).

4 – Dicas

Para esse projeto, existem algumas dicas muito interessantes, tanto em relação ao hardware quanto ao software envolvido.

Tal projeto foi desenvolvido para uma aplicação onde era necessário o uso de 40 sensores magnéticos. Esses sensores são chaves digitais que são acionadas pela presença de um campo magnético, originado por um imã, por exemplo. As aplicações mais comuns desses sensores são sistemas de segurança ou de monitoramento de ambientes. A placa de expansão de entradas digitais pode ser utilizada também com chaves digitais comuns (push-bottons, chaves DIP ou chaves fim-de-curso, por exemplo) ou pares ópticos.

Uma outra dica interessante é alterar o número de entradas digitais de acordo com a aplicação da placa. O projeto demonstrado nesse tutorial disponibiliza 40 entradas digitais, mas ele pode ser facilmente expandido até 128 entradas ou reduzido até 16 entradas. Para isso, basta adicionar ou retirar registradores e alterar o programa de modo que ele realize o controle adequado da placa.

É importante observar que para obter 128 entradas, serão utilizados 16 registradores e, consequentemente, serão necessárias 16 saídas digitais. É possível obtê-las no *Módulo de Entradas, Saídas e Servomotores* do KDR5000. Porém, caso a placa seja utilizada com o MEC1000, será possível obter no máximo 64 entradas, pois o MEC1000 possui 8 saídas digitais para o controle dos registradores.

É recomendável colocar um capacitor em cada entrada, para reduzir oscilações e interferências que possam interferir no valor da leitura. O capacitor terá a função de estabilizar a tensão presente na porta de entrada. Com esse objetivo, é comum utilizar capacitores entre 100 nF e 1uF.

Também é interessante fazer a proteção desses terminais de entrada digital. Essa proteção pode ser feita através de diodos zener. Um diodo zener de 5,1V pode ser aplicado na entrada, protegendo os registradores de uma tensão superior a essa.

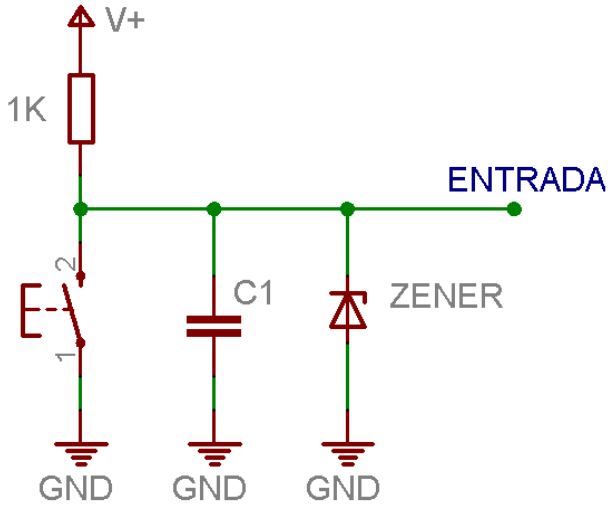

Figura 8: Capacitor de estabilização e diodo zener de proteção ligados a uma entrada da placa de expansão.

Em relação ao programa, pode ser interessante que a leitura seja feita automaticamente, bastando para isso adicionar um componente Timer e elaborar a rotina necessária para implementar isso.

Outra coisa importante é fazer confirmações das leituras do programa. Durante a utilização de chaves e sensores digitais, é comum ocorrer leituras falsas, devido à interferências ou ruídos. O próprio funcionamento mecânico das chaves pode ocasionar leituras falsas, no momento em que elas são pressionadas. Por isso, torna-se importante fazer uma confirmação das leituras. A utilização de um capacitor ligado à entrada digital reduz esssas interferências, porém, não as elimina completamente.

Esses ruídos e interferências se caracterizam por ocorrer durante intervalos rápidos, mas podem causar falsas leituras. Para ter a certeza de que se trata de um sinal válido, e não de um ruído, podemos fazer uma ou mais leituras de confirmação, em intervalos de algumas dezenas de milisegundos. Caso o valor se mantenha o mesmo nessas leituras, teremos a certeza de que se trata de uma leitura válida, e não de um ruído qualquer.

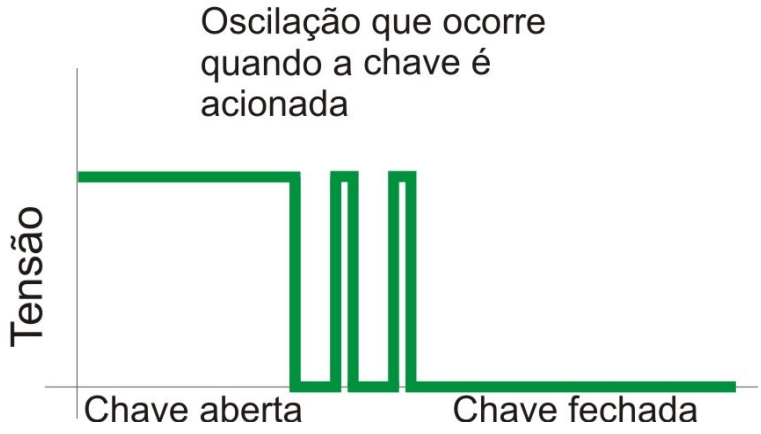

Figura 9: Exemplo de interferência que leva a leituras falsas. Tais oscilações ocorrem durante um intervalo de tempo muito curto, mas podem causar leituras erradas.

Vamos elaborar um método simples para verificar a validade da leitura, incrementando o trecho onde fazemos a leitura da porta de entradas digitais. Devemos utilizar duas variáveis para armazenar os valores das entradas, chamadas entrada1 e entrada2. Primeiro, o programa fará uma verificação da entrada digital selecionada e armazenará seu valor na variável entrada1.

O programa fará então uma pausa de alguns milisegundos e realizará uma nova leitura da entrada digital, armazenando o valor obtido na variável entrada2. A pausa foi realizada através da função Sleep, cujo parâmetro 50 determina uma pausa de 50 milisegundos. Esse tempo pode ser alterado, se houver necessidade.

Em seguida, o programa verificará se o valor de entrada1 é igual ao de entrada2. Se os valores forem iguais, temos uma garantia de que a entrada digital permaneceu estável durante essas duas leituras. Portanto, o valor provavelmente é válido, e a função LerEntradaDeExpansao terá como resposta esse valor.

Caso os dois valores sejam diferentes, pode ter ocorrido uma interferência ou ruído. Não podemos saber qual dos dois valores é válido. Portanto, a função LerEntradaDeExpansao retornará o valor 255. Mais adiante, faremos com que seja exibida uma caixa de mensagem ao usuário, indicando que houve uma leitura inválida.

```
...

  // Lê porta de entradas digitais
  leitura := kit.DigitalPortRead(0);

  // Extrai o valor do sensor e armazena na variável entrada1
  entrada1 := (leitura1 SHR (((bit-1) mod 8))) AND $01;

  // Realiza uma espera de 50 milisegundos
  Sleep (50);

  // Faz uma nova leitura da porta de entradas digitais
  leitura := kit.DigitalPortRead(0);

  // Extrai o valor do sensor e armazena na variável entrada2
  entrada2 := (leitura1 SHR (((bit-1) mod 8))) AND $01;


  // Verifica se o valor de entrada1 é igual ao de entrada2
  if (entrada1 = entrada2) then
    LerEntradaDeExpansao := entrada1;
  else
    LerEntradaDeExpansao := 255;

end;
```

Vamos alterar também as linhas de código relacionadas ao botão Ler. Devemos modificar o trecho que mostra ao usuário a leitura da porta de entrada. O programa verificará se a variável "valor" é igual a 255. Se ela possuir esse valor, retornado pela função LerEntradaDeExpansao, sabemos que houve uma leitura inválida. O programa então mostrará essa informação ao usuário. Caso contrário, será exibido o valor da entrada digital selecionada.

```
procedure TFormMain.ButtonLerClick(Sender: TObject);
var
  valor : Byte;
begin
  // Lê a entrada de expansão selecionada
  valor := LerEntradaDeExpansao(ComboBoxPorta.ItemIndex+1);
```

```
  // Se a variável valor for 255, houve uma leitura inválida
  if (valor = 255) then
        ShowMessage('Leitura inválida');
  else
        //Mostra o valor da entrada selecionada
        ShowMessage('O valor lido foi: ' + IntToStr(valor));
end;
```