TUTORIAL
SAÍDAS DIGITAIS

Autor: Luís Fernando Patsko e Tiago Lone
Nível: Intermediário
Criação: 27/12/2005
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 vamos aprender a utilizar as portas de saídas digitais presentes no *Módulo de Entradas, Saídas e Servo-Motores* do KDR5000. Esse módulo possui duas dessas portas, sendo que cada uma delas possui 8 bits. Através desse programa também é possível utilizar a porta de saídas digitais do MEC1000. Vamos criar um programa que permite definir o estado dos bits dessas saídas e veremos também um pouco sobre como trabalhar com elas. Dessa forma teremos total controle sobre elas e poderemos usá-las sem problemas em nossos projetos.

2 – Material

Para esse tutorial é necessário o *Módulo Principal* e o *Módulo de Entradas, Saídas e Servo-Motores*. Para a criação do programa será necessário o Borland Delphi 6. A seguir a imagem da montagem do Kit necessária para esse tutorial.

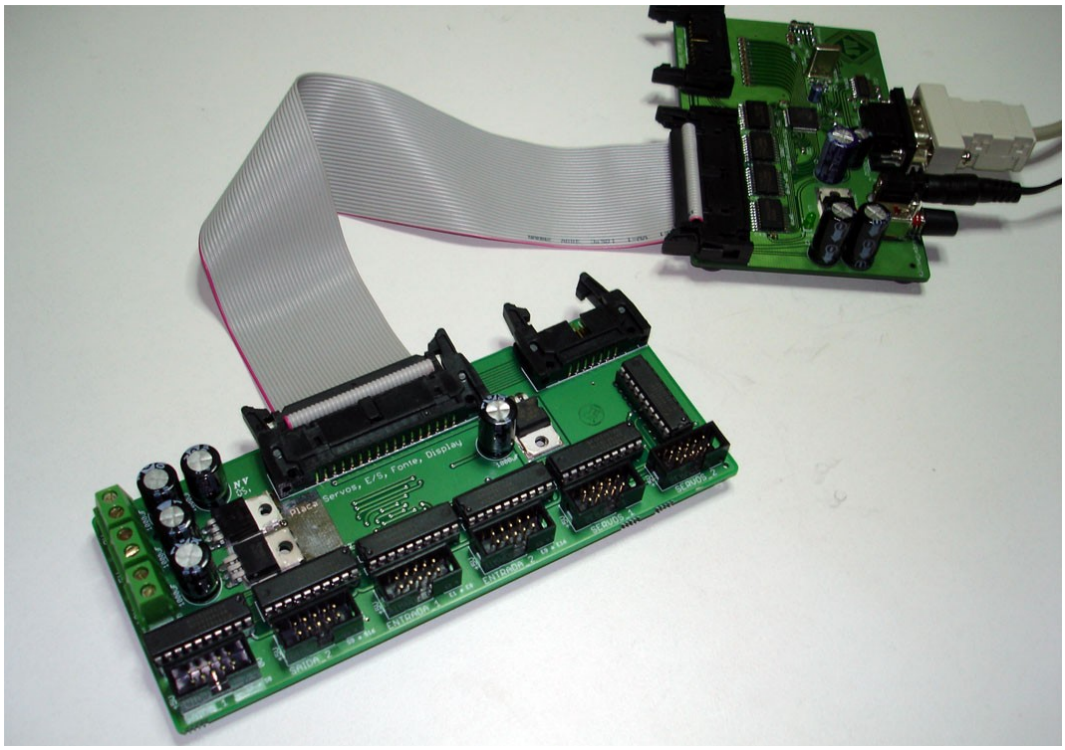

Figura 1: *Montagem do kit usada nesse tutorial.*

3 – Projeto

Nesse tutorial vamos desenvolver um programa para controlar o estado das portas de saídas digitais do *Módulo de Entradas, Saídas e Servo-Motores*. Será possível definir o estado de todos os bits das portas. A aparência desse programa será a seguinte.

Figura 2: *Aparência final do programa que será criado nesse tutorial.*

Nosso primeiro passo é criar essa interface gráfica. 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 possui um Label, um ComboBox, oito CheckBox e um Button. O ComboBox serve para selecionar a porta que iremos definir o estado, os CheckBox para setar o estado dos bits dessa porta e o Button para atualizar o estado dela. Todos esses componentes encontram-se na aba "Standard" da barra de componentes.

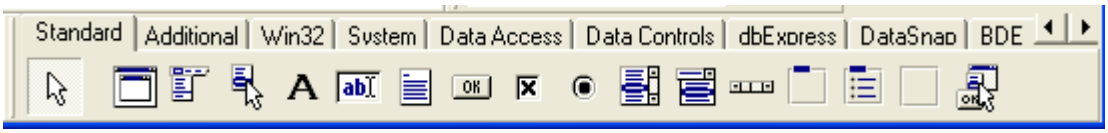

Figura 3: *Aba "Standard" da Barra de componente.*

Vamos adicionar um Label e um ComboBox para a seleção da porta que queremos definir o estado. O componente Label possui o seguinte ícone.

Figura 4: *Ícone do componente Label.*

E o componente ComboBox possui o seguinte ícone.

Figura 5: *Ícone do componente ComboBox.*

Adicionamos o Label e modificamos as seguintes propriedades.

Name	=	LabelPorta
Caption	=	Porta:
Font/Style/fsBold	=	true

Em seguida adicionamos um componente ComboBox e modificamos as propriedades a seguir.

Name	=	ComboBoxPorta
Style	=	csDropDownList
Items.Strings	=	Porta 0, Porta 1
ItemIndex	=	0

Com isso nosso Form irá se parecer com o seguinte.

Figura 6: *Aparência do Form após a inclusão do Label e do ComboBox.*

Vamos adicionar os componentes CheckBox para definir o estado dos bits da porta. O componente CheckBox, encontra-se na aba “Standard” da barra de componentes e possui o seguinte ícone.

Figura 7: *Ícone do componente CheckBox.*

Serão adicionados oito desses componentes e vamos modificar as seguintes propriedades neles.

Name	=	CheckBoxBit0
Caption	=	Bit 0:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit1
Caption	=	Bit 1:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit2
Caption	=	Bit 2:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit3
Caption	=	Bit 3:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit4
Caption	=	Bit 4:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit5
Caption	=	Bit 5:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit6
Caption	=	Bit 6:
Font/Style/fsBold	=	true

Name	=	CheckBoxBit7
Caption	=	Bit 7:
Font/Style/fsBold	=	true

Assim teremos a seguinte aparência no nosso Form principal.

Figura 8: *Form após a adição dos CheckBox de estado dos bits.*

Nesse ponto só falta adicionar o botão para atualização do estado da porta. Para isso adicionamos um componentes Button que pode ser encontrado na aba "Standard" da barra de componentes. Esse componente possui o seguinte ícone.

Figura 9: *Ícone do componente Button.*

Temos que modificar as seguintes propriedades do botão.

Name	=	ButtonAtualizar
Caption	=	Atualizar
Font/Style/fsBold	=	true

Um último detalhe que vamos modificar é a propriedade Caption do Form principal. Como copiamos o projeto do tutorial Base, essa propriedade possui o valor "Projeto Base". Vamos modificar essa propriedade para "Saídas Digitais". Com isso finalizamos a construção de nossa interface gráfica. A seguir a imagem dessa interface finalizada.

Figura 10: *Interface finalizada.*

O próximo passo é implementar o código que define os estado da porta. Isso é feito no manipulador do evento OnClick do botão "Atualizar". Para criar esse manipulador podemos selecionar o componente Button que possui o texto "Atualizar", ir no **Object Inspector**, selecionar a aba Events e dar um duplo clique sobre a linha que está escrito OnClick. Uma forma mais fácil de fazer isso é apenas dar um duplo clique sobre o botão no Form e com isso o Delphi irá criar automaticamente um manipulador para o evento OnClick. O seguinte código será criado.

```
Procedure TformMain.ButtonAtualizarClick(
                                        Sender: TObject);
begin
end;
```

Vamos adicionar dentro desse manipulador o código que irá definir o estado da porta de saída digital selecionada. O método que vamos utilizar para controlar a porta de saída digital é o DigitalPortWrite. Esse método possui a seguinte declaração.

```
Procedure DigitalPortWrite(port : Integer;
                                      value : Byte);
```

Ele possui dois parâmetros, um que indica a porta que queremos definir o estado e outro que define o estado da porta. O *Módulo de Entradas, Saídas e Servo-Motores* possui duas portas de saídas digitais, logo o primeiro parâmetro pode ser "0", para indicar a primeira porta, ou "1", para indicar a segunda.

Já o segundo parâmetro do método DigitalPortWrite define quais bits da porta deverão ser setados para o nível lógico 1 e quais deverão ser setados para o nível lógico 0. Esse parâmetro é um valor de 8 bits sendo que cada bit dele irá definir o estado de um bit da porta saída. O bit 0 da porta será representado pelo bit 0 do parâmetro, o bit 1 pelo bit 1 e assim por diante. Para maiores informações sobre o funcionamento e uso das portas de saídas digitais veja o tópico sobre essas portas ainda nesse documento.

Agora vamos iniciar a implementação do manipulador do evento OnClick do botão "Atualizar". O início do código ficará assim.

```
Procedure TformMain.ButtonAtualizarClick(
                                    Sender: TObject);
var
  // Armazena estado dos bits
  bits : Byte;

begin
  // Inicializa bits
  bits := 0;

  // Verifica estado de todos os bits
  if CheckBoxBit0.Checked then
    bits := bits OR 1;

  if CheckBoxBit1.Checked then
    bits := bits OR 2;

  if CheckBoxBit2.Checked then
    bits := bits OR 4;

  if CheckBoxBit3.Checked then
    bits := bits OR 8;

  if CheckBoxBit4.Checked then
    bits := bits OR 16;

  if CheckBoxBit5.Checked then
    bits := bits OR 32;

  if CheckBoxBit6.Checked then
    bits := bits OR 64;
```

```
    if CheckBoxBit7.Checked then
      bits := bits OR 128;


    ...
end;
```

Resumindo, nesse trecho de código declaramos e inicializamos a variável denominada "bits" com zero. Em seguida verificamos quais bits da porta de saída deverão estar setados para nível lógico "1" e armazenamos essa informação na variável denominada "bits".

Quando um CheckBox está marcado temos que setar o bit a que ele se refere para nível lógico "1". Para fazer isso sem interferir nos outros bits utilizamos a operação lógica "OR". Por exemplo, para setar o terceiro bit de uma valor para "1" fazemos uma operação "OR" desse valor com o valor 4. Utilizamos 4 porque esse valor em binário é representado por 00000100b, ou seja, possui setado para "1" apenas o terceiro bit, que é o bit que desejamos setar. Dessa forma setamos o terceiro bit do valor para "1" sem interferir no valor dos outros bits. Para entender melhor a operação "OR" procure alguma documentação sobre operações lógicas, mais especificamente a operação lógica "OR".

Para facilitar, a seguir apresentamos uma tabela com o valor que temos que utilizar com a operação "OR" para setar um determinado bit. A primeira coluna indica o bit que queremos setar e as outras 3 colunas indica o valor em decimal, hexadecimal e binário que devemos utilizar com a operação "OR".

Bit	***Decimal***	***Hexadecimal***	***Binário***
0	1	0x01	00000001b
1	2	0x02	00000010b
2	4	0x04	00000100b
3	8	0x08	00001000b
4	16	0x10	00010000b
5	32	0x20	00100000b
6	64	0x40	01000000b
7	128	0x80	10000000b

No nosso código utilizamos os valores em decimal. Primeiro zeramos a variável denominada "bits" para garantir que todos os bits dela iniciassem com "0" e em seguida fomos verificando, um por um, qual CheckBox estava selecionado.

Se o primeiro CheckBox, que representa o primeiro bit, estiver marcado, fazemos uma "OR" da variável "bits" com o valor 1 e assim setamos o primeiro bit para "1". Se o segundo CheckBox estiver marcado, fazemos uma "OR" do valor já existente na variável "bits" com o valor 2. Assim setamos o segundo bit para "1" sem interferir no valor dos outros bits.

Após verificar todos os CheckBox teremos na variável "bits" o valor correto para enviar

como parâmetro. Então é necessário apenas chamar o método DigitalPortWrite e passar esse valor como parâmetro. É o que vamos fazer a seguir.

```
    ...

    if CheckBoxBit6.Checked then
      bits := bits OR 64;

    if CheckBoxBit7.Checked then
      bits := bits OR 128;


    // Envia comando
    kit.DigitalPortWrite(ComboBoxPorta.ItemIndex,
                         bits);
end;
```

Pronto, com isso ao pressionarmos o botão "Atualizar" o estado da porta de saídas digitais, que estiver selecionada, será atualizado. A aparência final do programa será a seguinte.

Figura 11: *Aparência final do programa.*

Assim temos um programa que controla o estado das portas de saídas digitais. Podemos selecionar a porta que queremos controlar e definir exatamente o seu estado. A seguir vamos ver um pouco mais sobre as portas de saídas digitais e como utilizá-las em aplicações práticas.

4 – Portas de Saídas digitais

As portas de saídas digitais foram projetadas para uso genérico, dando ao usuário a possibilidade de utilizar o Kit para controlar um circuito externo e assim integrar as capacidades do Kit com as funcionalidades de um circuito projetado para um uso específico. Sua versatilidade e flexibilidade permitem inúmeras combinações e, com o devido conhecimento de eletrônica e programação, as possibilidades de utilização são ilimitadas.

O usuário apenas deverá respeitar os limites quanto a configuração das saídas digitais. Muita atenção deverá ser destinada no que diz respeito à capacidade e à pinagem destas. Os equipamentos a serem conectados deverão suportar convenientemente o sinal de 3,3V aplicado na saída. Caso ele possa ser alimentado com uma tensão de 5V, podemos utilizar as saídas disponíveis nos conectores. Caso contrário, é necessário utilizar outra fonte de alimentação.

Como o sinal da saída digital é fraco, para algumas aplicações é necessária a utilização de algum meio de amplificação para suprir o consumo do circuito a ser utilizado. Para o acionamento de um motor ou um relé por exemplo, a corrente máxima fornecida de 10mA é insuficiente, mas isso pode ser facilmente resolvido utilizando um transistor para amplificar o sinal da saída digital. Os tutoriais disponíveis no curso explicam passo a passo a montagem de um circuito compatível com as portas de saídas digitais.

A pinagem é também algo ao qual dada muita atenção. Qualquer descuido em relação à ligação pode danificar tanto o Kit quanto o equipamento a ser conectado. A imagem a seguir mostra a disposição dos conectores das saídas digitais.

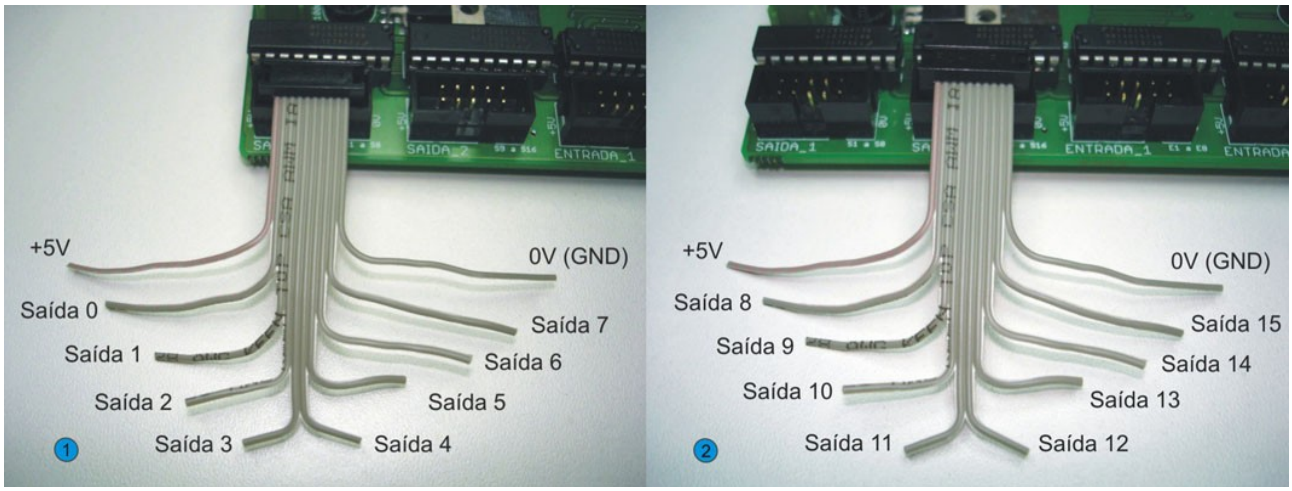

Figura 12: *Configuração das saídas digitais: 1-Saída digital 0. 2-Saída digital 1.*

Crimpando um cabo flat e abrindo-o, podemos compreender muito bem a pinagem dos conectores. Como pode ser observado, os conectores de alimentação estão nas extremidades em cada uma delas. Seu uso não é obrigatório caso o circuito a ser conectado já possua alimentação, entretanto deve-se tomar muito cuidado para evitar algum curto-circuito. Os fios disponíveis para a conexão com circuitos externos estão localizados no meio do cabo flat, sendo que cada um corresponde a um bit de uma porta de saída digital. Por exemplo, a Saída 1 corresponde ao bit de número 1 da porta 0 e a Saída 15 corresponde ao bit de número 7 da porta 1.

A única diferença presente entre o KDR5000 e o MEC1000, em relação às portas de saídas digitais, é a sua quantidade. Enquanto que o KDR5000 tem a disposição do usuário 2 portas,

presentes no *Módulo de Entradas, Saídas e Servo-Motores,* o MEC1000 possui apenas uma. Seu funcionamento e sua pinagem é a mesma.

Exemplos de circuitos simples que podem ser conectados às saídas digitais estão presentes em tutoriais que serão apresentados ao longo do curso. Entre eles, podemos destacar os que possibilitam o controle de um display de 7 segmentos, um relé e uma ponte H para controle de motores.

5 – Conclusão

Nesse tutorial vimos o que devemos fazer para controlar as portas de saídas digitais do *Módulo de Entradas, Saídas e Servo-Motores* do KDR5000 ou do MEC1000. Com o projeto que criamos foi possível entender o funcionamento do método DigitalPortWrite e o significado de seus parâmetros. Vimos também como utilizar as portas de saídas digitais em conjunto com montagens eletrônicas em aplicações práticas. Com isso já podemos utilizar essas portas em nossos projetos sem dificuldades.