TUTORIAL
DISPLAY DE 7 SEGMENTOS

Autor: Tiago Lone
Nível: Básico
Criação: 16/12/2005
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 aprenderemos a manipular os dois displays de 7 segmentos do *Módulo de Motores e Displays*. Esses displays podem ser utilizados para transmitir informações em uma montagem e vamos ver o que é necessário fazer para controlar, através de um programa, o que será apresentado por esses displays. Para auxiliar nesse estudo, será desenvolvido um programa que permite controlar o que os displays irão apresentar.

2 – Material

O programa desenvolvido nesse tutorial precisará do *Módulo Principal* e o *Módulo de Motores e Displays* sem nenhum ítem extra, pois os displays de 7 segmentos ficam soldados na placa principal desse módulo. Para a criação do programa será necessário o Borland Delphi 6. A seguir a imagem da placa principal do *Módulo de Motores e Displays* do Kit onde podemos visualizar os dois displays de 7 segmentos.

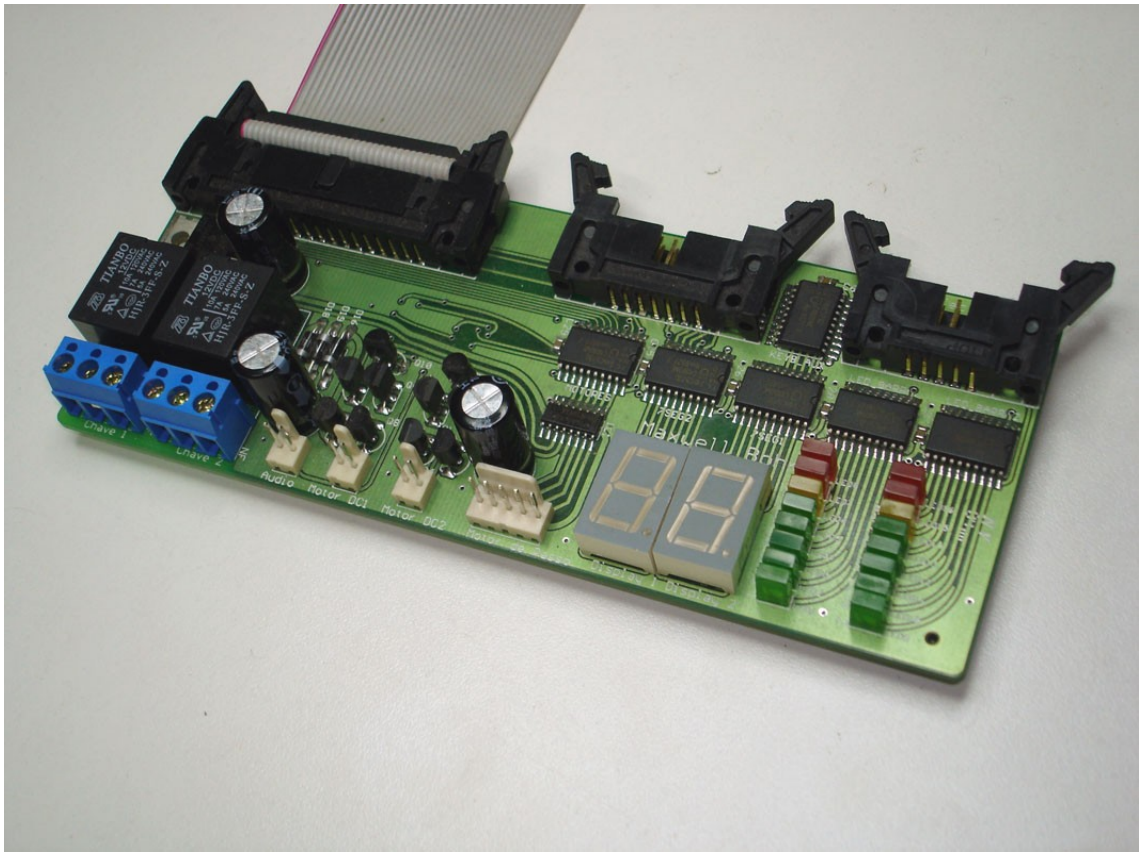


Figura 1: *Placa do Módulo de Motores e Displays.*

3 – Projeto

Nesse tutorial vamos criar um programa que permite o controle total sobre o que é apresentado nos displays de 7 segmentos. Ele irá permitir que apresentemos números nos displays ou que sejam especificados quais segmentos de LED dos displays devem ficar ligados ou desligados. A interface gráfica desse programa será a seguinte.

Figura 2: *Interface gráfica do programa que será criado nesse tutorial.*

Nosso primeiro passo na criação desse programa é desenvolver a interface gráfica. Vamos utilizar o projeto criado no tutorial Base, que já nos fornece algumas funcionalidades interessantes, e em cima dele vamos adicionar alguns componentes gráficos extras. Para isso temos apenas que copiar o projeto daquele tutorial.

A primeira coisa que modificaremos no projeto é a propriedade Caption do Form principal, que possui o texto "Projeto Base". Vamos modificar para "Display de 7 Segmentos". Com isso já podemos começar a adicionar os componentes gráficos ao Form.

Para modificar a interface gráfica criada no tutorial Base para a interface do programa desse tutorial vamos precisar adicionar alguns componentes ao Form principal. Adicionaremos alguns Labels, ComboBox, CheckBox e um Button. Os Labels são utilizados apenas para identificar a funcionalidade de três ComboBox. Esses ComboBox são utilizados para selecionar o display que queremos controlar, o modo como queremos controlá-lo e por fim um para selecionar o número que o display deverá apresentar. Os CheckBox são utilizados para definir quais segmentos de LED do display serão ligados e o Button, que é um botão, é utilizado para atualizar o estado do display de acordo com a configuração atual.

Todos esses componentes podem ser encontrados na aba "Standard" da barra de componentes.

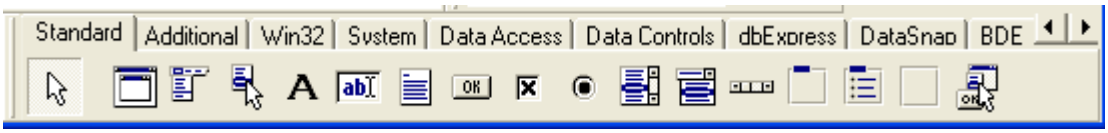


Figura 3: *Aba "Standard" da Barra de componentes.*

Vamos adicionar um Label e um ComboBox para a seleção do display que será controlado. 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.*

Após adicionar os dois componentes, modificamos as seguintes propriedades do Label.

Name	=	LabelDisplay
Caption	=	Display:
Font/Style/fsBold	=	true

E as seguintes propriedades do ComboBox.

Name	=	ComboBoxDisplay
Style	=	csDropDownList
Items.Strings	=	Display I, Display II
ItemIndex	=	0

Com isso o Form terá a seguinte aparência.

Figura 6: *Aparência do Form após a inclusão dos controles para selção do display que será controlado.*

Vamos adicionar mais um conjunto de Label e ComboBox para selecionar o modo de controle do display. Vamos oferecer dois modos, um em que iremos apresentar números e outro em que vamos controlar o estado de segmento por segmento. Então adicionamos um Label e modificamos as propriedades a seguir.

Name	=	LabelModo
Caption	=	Modo:
Font/Style/fsBold	=	true

Em seguida adicionamos um ComboBox e modificamos as propriedades listadas abaixo.

Name	=	ComboBoxModo
Style	=	csDropDownList
Items.Strings	=	Números, Segmentos
ItemIndex	=	0

Com isso já temos na interface os controles para seleção de modo de operação. O Form deve estar semelhante ao seguinte.

Figura 7: *Interface gráfica com os controles para seleção do display controlado e do modo de operação.*

Agora vamos incluir os componentes para seleção do número que será apresentado no display quando o modo de operação selecionado for "Números". Vamos utilizar novamente o conjunto Label com um ComboBox. Adicionamos o Label e modificamos as seguintes propriedades.

Name	=	LabelNumero
Caption	=	Número:
Font/Style/fsBold	=	true

Em seguida adicionamos o ComboBox e modificamos as seguintes propriedades.

Name	=	ComboBoxNumero
Style	=	csDropDownList
Items.Strings	=	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
ItemIndex	=	0

Assim temos a interface gráfica com a seguinte aparência.

Figura 8: *Interface gráfica após adição dos componentes para seleção do número apresentado.*

Temos que adicionar os componentes para a seleção dos segmentos que serão ligados ou que permanecerão desligados quando o modo de operação selecionado for "Segmentos". Vamos utilizar para isso vários componentes do tipo CheckBox, sendo que cada um deles irá representar um segmento de LED do display.

Esses componentes podem estar em dois estados, "Selecionados" ou "Não selecionados". Se estiverem selecionados então o segmento de LED a que eles se referem será ligado, caso contrário, ele será desligado.

Podemos encontrar o componente CheckBox na aba "Standard" da barra de componentes. Ele possui o seguinte ícone.

Figura 9: *Ícone do componente CheckBox.*

Vamos adicionar oito componentes desses, um para cada segmento. Vamos modificar algumas propriedades deles. A seguir a lista de propriedades que vamos modificar nos oito componentes.

Name	=	CheckBoxSegmento0
Caption	=	Segmento 0:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento1
Caption	=	Segmento 1:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento2
Caption	=	Segmento 2:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento3
Caption	=	Segmento 3:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento4
Caption	=	Segmento 4:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento5
Caption	=	Segmento 5:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento6
Caption	=	Segmento 6:
Font/Style/fsBold	=	true

Name	=	CheckBoxSegmento7
Caption	=	Segmento 7:

Font/Style/fsBold = true

Após a inclusão desses componentes o Form irá se parecer com o seguinte.

Figura 10: *Form após a adição dos CheckBox.*

Para finalizar a interface desse programa vamos adicionar um botão para atualizar o display. Toda vez que ele for pressionado será enviado um comando para o Kit que irá atualizar a apresentação do display de acordo com as seleções feitas na interface gráfica.

Para isso temos que adicionar um componente Button, que pode ser encontrado na aba "Standard" da barra de componentes. Esse componente possui o seguinte ícone.

Figura 11: *Ícone do componente Button.*

Temos que modificar as seguintes propriedades dos botões.

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

Com isso finalizamos a construção de nossa interface gráfica. Ela terá a seguinte aparência.

Figura 12: *Aparência final da interface gráfica.*

Agora vamos implementar o código que controla os displays de 7 segmentos. Para isso precisamos criar um manipulador para o evento OnClick do botão "Atualizar". Para fazer isso podemos selecionar o componente Button com 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;
```

Dentro desse manipulador implementaremos o código para controlar os displays. Quando o modo de operação selecionado for "Números", então utilizaremos o método Disp7SegNum para atualizar o display. Se o modo for "Segmentos", então utilizaremos o método Disp7SegStatus. A seguir a declaração desses dois métodos.

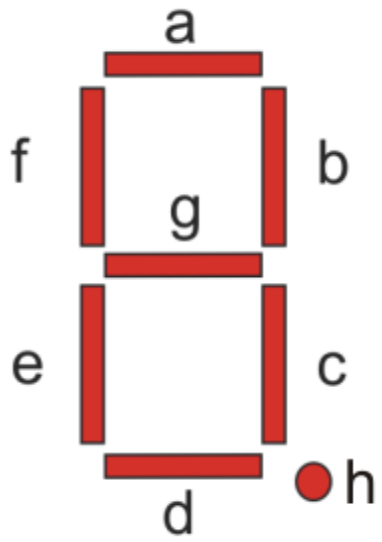
```
Procedure Disp7SegSetNum(disp, num : Integer);

Procedure Disp7SegSetStatus(disp : Integer;
                                    status : Byte);
```

Esses dois métodos possuem como primeiro parâmetro o display que queremos controlar. Se esse parâmetro tiver um valor "0" indica que queremos controlar o primeiro display de 7 segmentos do *Módulo de Motores e Displays*. Se o valor for "1" então queremos controlar o segundo display.

O segundo parâmetro do método Disp7SegNum será o número que queremos que o

display apresente. Esse parâmetros pode variar de 0 à 15 sendo que os valores 10, 11, 12 ,13, 14 e 15 serão representados por A, B, C, D, E e F respectivamente. Com isso podemos apresentar até mesmo valores no formato hexadecimal.

Já o segundo parâmetro do método Disp7SegStatus é um pouco mais complexo e define quais segmentos do display deverão ser ligados ou desligados. Ele é um valor de 8 bits sendo que cada bit desse valor irá definir o estado de um dos segmentos de LED do display. A seguir o esquema de um display de 7 segmento e os bits que representam seus segmentos.



Segmento	Descrição	Bit
a	Superior	0
b	Superior Direito	1
c	Inferior Direito	2
d	Inferior	3
e	Inferior Esquerdo	4
f	Superior Esquerdo	5
g	Central	6
h	Ponto	7

Por exemplo, para apresentar o número 1 é necessário que os segmentos 'b' e 'c' do display sejam acesos. Dessa forma o parâmetro que devemos utilizar é 6, que em binário é representado por 00000110b. Como podemos ver na forma binária do valor 6, os bits 1 e 2 são “1”, o que indica que os segmentos representados por esses bits devem ser acesos. Os outros segmentos ficam apagados pois os bits que os representam são “0”. Sabendo disso, vamos iniciar a implementar nosso manipulador do evento OnClick.

```
Procedure TformMain.ButtonAtualizarClick(Sender:
```

```
                                                    TObject);
var
  // Armazena os segmentos que devem ser ligados
  segmentos : Byte;

begin
  // Verifica o modo
  if ComboBoxModo.ItemIndex = 0 then
  begin
    // Código para o modo "Números"
  end
  else
  begin
    // Código para o modo "Segmentos"
  end;
end;
```

Nesse trecho de código, que foi adicionado ao manipulador do evento OnClick, declaramos uma variável denominada "segmentos", que irá armazenar quais segmentos serão ligados ou desligados, e em seguida verificamos com uma estrutura "if" qual modo de operação está selecionado. Vamos agora adicionar o código para quando o modo selecionado for "Números".

```
Procedure TformMain.ButtonAtualizarClick(Sender:
                                                    TObject);
var
  // Armazena os segmentos que devem ser ligados
  segmentos : Byte;

begin
  // Verifica o modo
  if ComboBoxModo.ItemIndex = 0 then
  begin
    // Código para o modo "Números"
    // Envia o número selecionado
    kit.Disp7SegSetNum( ComboBoxDisplay.ItemIndex,
                        ComboBoxNumero.ItemIndex);
```

```
    end
    else
    begin
      // Código para o modo "Segmentos"
    end;
  end;
```

Essa linha de código que adicionamos utiliza o método Disp7SegNum e passa como parâmetro o display selecionado e o número que deve ser apresentado por esse display.

Os ítens dos ComboBox de seleção de display e de número foram adicionados na ordem correta para que a posição deles dentro do ComboBox fosse equivalente ao valor que deve ser passado como parâmetro para representá-los. Dessa forma podemos utilizar diretamente como parâmetro a propriedade ItemIndex dos ComboBox.

A propriedade ItemIndex de um ComboBox indica a posição do ítem selecionado no momento. Por exemplo, no ComboBox de seleção do display temos dois ítens, "Display I" e "Display II". Eles aparecem nessa ordem no ComboBox. Se o primeiro ítem estiver selecionado, isto é, o ítem "Display I", então a propriedade ItemIndex será "0". Caso a segunda opção esteja selcionada, isto é, "Display II", então a propriedade ItemIndex será "1".

Vamos agora implementar o código para quando o modo selecionado é "Segmentos". O código será o seguinte.

```
  ...

  // Verifica o modo
  if ComboBoxModo.ItemIndex = 0 then
  begin
    // Código para o modo "Números"
    // Envia o número selecionado
    kit.Disp7SegSetNum( ComboBoxDisplay.ItemIndex,
                        ComboBoxNumero.ItemIndex);
  end
  else
  begin
    // Código para o modo "Segmentos"
    // Inicializa variável com "0"
    segmentos := 0;

    // Verifica quais segmentos devem ser ligados
```

```
      if CheckBoxSegmento0.Checked then
        segmentos := segmentos OR 1;

      if CheckBoxSegmento1.Checked then
        segmentos := segmentos OR 2;

      if CheckBoxSegmento2.Checked then
        segmentos := segmentos OR 4;

      if CheckBoxSegmento3.Checked then
        segmentos := segmentos OR 8;

      if CheckBoxSegmento4.Checked then
        segmentos := segmentos OR 16;

      if CheckBoxSegmento5.Checked then
        segmentos := segmentos OR 32;

      if CheckBoxSegmento6.Checked then
        segmentos := segmentos OR 64;

      if CheckBoxSegmento7.Checked then
        segmentos := segmentos OR 128;

      ...

    end;
  end;
```

Esse novo trecho de código inicializa a variável denominada “segmentos” com zero e em seguida verifica quais segmentos de LED devem ser ligados ou desligados e armazena essa informação na variável denominada “segmentos”.

Quando um LED tiver que ser ligado, temos que setar para “1” o bit correspondente a ele. Para fazer isso sem interferir nos outros bits do valor 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 que desejamos setar, 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 “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 “segmentos” 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 segmento de LED, estiver selecionado fazemos uma “OR” da variável “segmentos” com o valor 1 e assim setamos o primeiro bit para “1”. Se o segundo CheckBox estiver selecionado fazemos uma “OR” do valor já existente na variável “segmentos” 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 “segmentos” o valor correto para enviar como parâmetro. Então é necessário apenas chamar o método Disp7SegStatus e passar esse valor como parâmetro. É o que vamos fazer a seguir.

```
...

if CheckBoxSegmento6.Checked then
  segmentos := segmentos OR 64;

if CheckBoxSegmento7.Checked then
  segmentos := segmentos OR 128;

// Envia comando para o Kit
kit.Disp7SegSetStatus(
```

```
                ComboBoxDisplay.ItemIndex,segmentos);
        end;
    end;
```

Com isso finalizamos o código e já podemos testar nosso programa. A aparência final do programa ficou assim.

Figura 13: *Programa finalizado.*

Se selecionarmos o modo "Números" podemos escolher que número o display irá apresentar. No modo "Segmentos" podemos definir um a um quais segmentos de LEDs do display serão ligados. A seguir algumas imagens dos displays ligados.

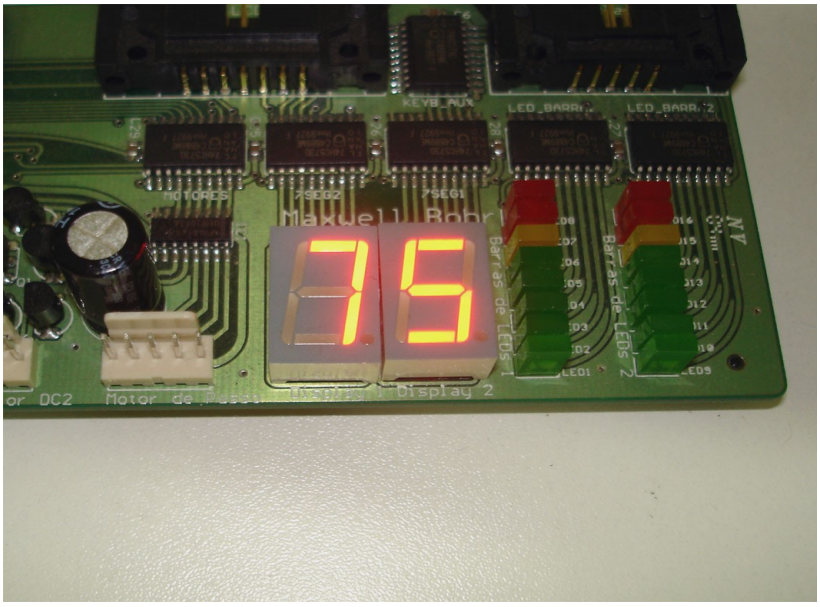


Figura 1: *Displays apresentando os números 7 e 5.*

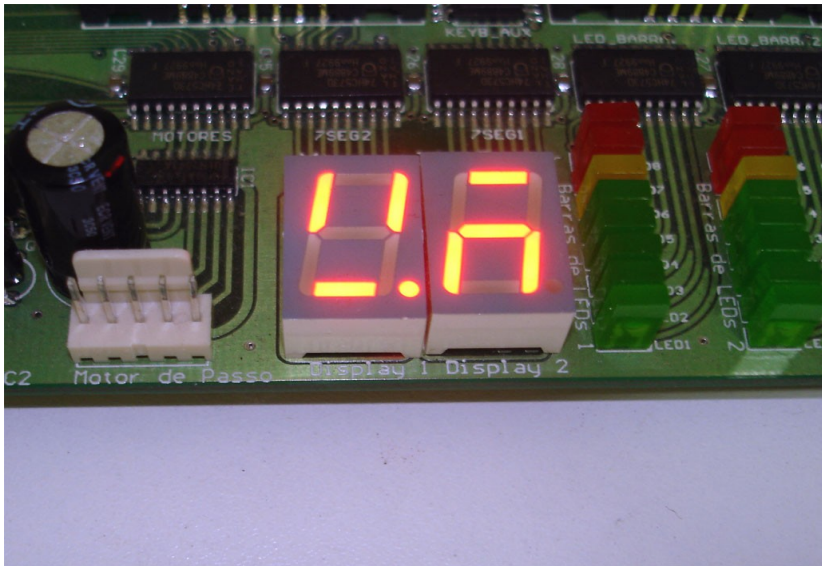


Figura 2: *Displays com segmentos de LED controlados individualmente.*

Pronto, temos um programa que controla os displays de 7 segmentos e nos oferece total controle sobre o que eles apresentam.

4 – Dicas

Como quando selecionamos o modo de apresentação de números os CheckBox da nossa interface não tem função ou quando selecionamos o modo de controle dos segmentos o ComboBox de seleção de números que não tem função, então seria interessante que os controles que não tenham função em um determinado modo de operação fiquem desabilitados.

Fazer isso é simples. Vamos utilizar o evento OnChange do ComboBox de seleção do modo. Criamos o manipulador para esse evento dando um duplo clique sobre o ComboBox de seleção de modo. O seguinte código será criado.

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

Esse manipulador será chamado toda vez que o modo for modificado. Temos que colocar dentro dele as rotinas para desabilitar e reabilitar os componentes de acordo com o modo selecionado. Esse código ficará assim.

```
Procedure TformMain.ComboBoxModoChange(Sender:
                                              TObject);
begin
  // Verifica o modo
  if ComboBoxModo.ItemIndex = 0 then
  begin
    // Desabilita todos CheckBox
    CheckBoxSegmento0.Enabled := false;
    CheckBoxSegmento1.Enabled := false;
    CheckBoxSegmento2.Enabled := false;
    CheckBoxSegmento3.Enabled := false;
    CheckBoxSegmento4.Enabled := false;
    CheckBoxSegmento5.Enabled := false;
    CheckBoxSegmento6.Enabled := false;
    CheckBoxSegmento7.Enabled := false;

    // Habilita numeros
    ComboBoxNumero.Enabled := true;
  end
  else
  begin
    // Desabilita numeros
    ComboBoxNumero.Enabled := false;

    // Habilita todos CheckBox
    CheckBoxSegmento0.Enabled := true;
    CheckBoxSegmento1.Enabled := true;
    CheckBoxSegmento2.Enabled := true;
    CheckBoxSegmento3.Enabled := true;
    CheckBoxSegmento4.Enabled := true;
    CheckBoxSegmento5.Enabled := true;
    CheckBoxSegmento6.Enabled := true;
    CheckBoxSegmento7.Enabled := true;
  end;
end;
```

Com isso falta apenas um detalhe, a condição inicial dos componentes. Como nosso modo padrão quando o programa inicia é “Números”, então temos que deixar os CheckBox desabilitados na inicialização do programa. Logo, temos que modificar a propriedade Enabled de todos eles para false. Fazemos isso ainda no ambiente de desenvolvimento. Dessa forma não precisamos adicionar mais código ao nosso programa. Com isso finalizamos essa implementação. A seguir a imagem da aparência do programa nos dois modos. Observe que dependendo do modo os controles que não são utilizados ficam desabilitados.

Figura 16: *Modo "Números". Todos CheckBox desabilitados.*

Figura 17: *Modo "Segmentos". O ComboBox de seleção de número está desabilitado.*

5 – Conclusão

Nesse tutorial vimos como manipular os display de 7 segmentos do *Módulo de Motores e Displays*. Com o projeto que criamos foi possível entender o funcionamento do método Disp7SegNum e Disp7SegStatus e o significado dos seus parâmetros. Assim já podemos utilizar os displays de 7 segmentos do Kit Didático de Robótica para transmitir informações em qualquer projeto.