Discovery Session 4
Scrolling the LEDs

Overview

In this session we'll explore some alternate ways to control the tricolor LEDs introduced in our earlier discussion. In this tutorial each LED will be driven via a pulse width modulator, allowing the LED to be faded on and off. Using the Configurable Digital IO Instrument introduced in the first session we will create a custom interface driven from a script, that creates a running, fading pattern on the LEDs, commonly known as a Knight Rider pattern (in respect to the TV show of the same name). Control of the LED Controller and subsequently the LEDs will be managed through this custom interface and we'll explore just a few of the almost limitless ways we might interact with our system using this technology.

Prerequisites

This tutorial assumes you have a basic understanding of the process of placing and wiring objects in Altium Designer (including components, net labels / net connectivity, and wires / buses) and a basic understanding of the process of configuring and building a design using the Devices View (for specific details on this process, see **Discovery Session 1 – Exploring a Simple LED Driver**). No additional information is required.

Design detail

This exercise uses the components listed in Table 1, to create the circuit shown in Figure 1.

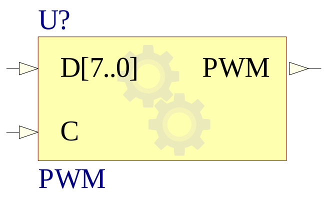
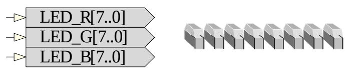
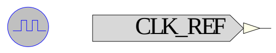
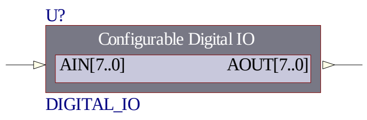
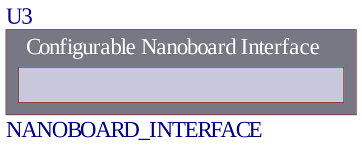
Component	Library	Name in Library
	FPGA Configurable Generic.IntLib	PWM
	FPGA NB3000 Port-Plugin.IntLib	LEDS_RGB
	FPGA NB3000 Port-Plugin.IntLib	CLOCK_REFERENCE
 U? Configurable Digital IO AIN[7..0] AOUT[7..0] DIGITAL_IO	FPGA_Instruments.IntLib	DIGITAL_IO
 U3 Configurable Nanoboard Interface NANOBOARD_INTERFACE	FPGA_Instruments.IntLib	NANOBOARD_INTERFACE

Table 1. List of components required by the design

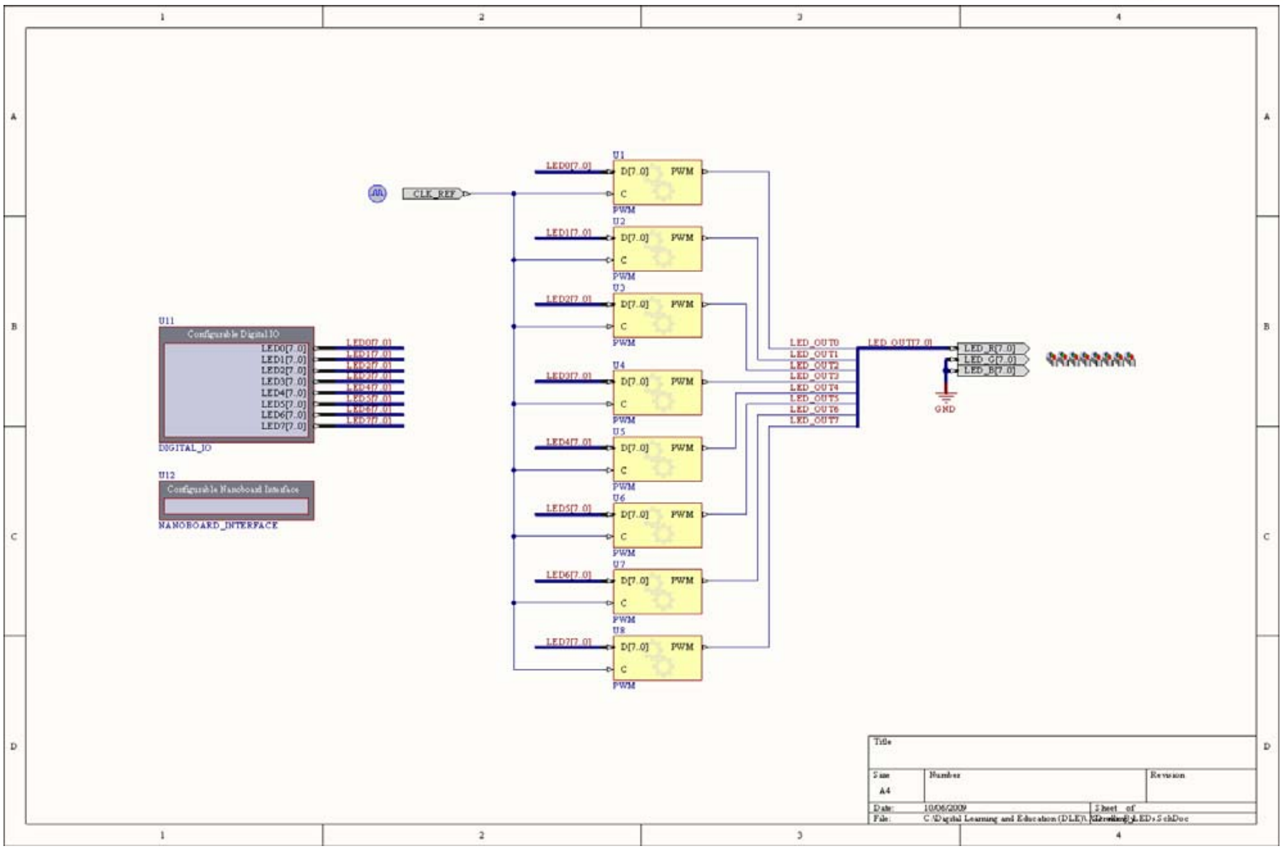
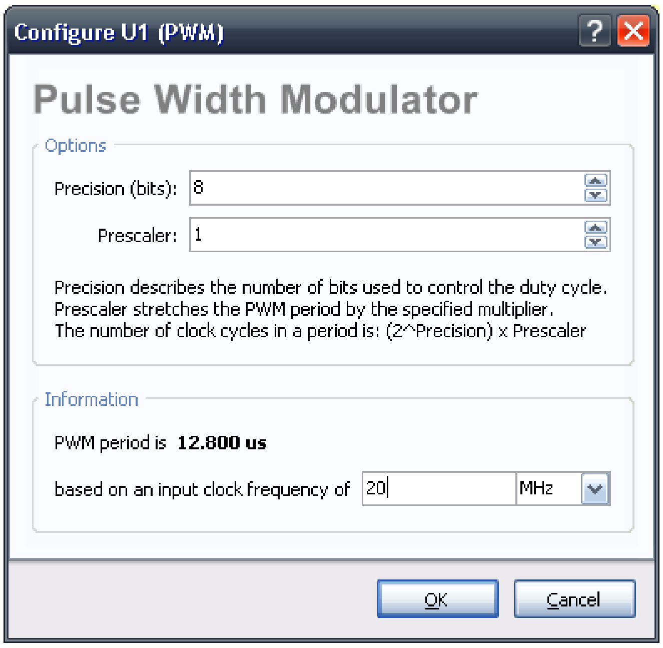


Figure 1. Schematic for the Scrolling_LEDs design.

Tutorial steps – preparing the hardware

1. Select **File»New»Projects»FPGA Project** to create a new, blank FPGA Project.
2. Select **File»New»Schematic** to add a new blank schematic to your FPGA project.
3. Select **File»Save Project As** to save the project documents. You will first be prompted to save the schematic, followed by the project itself. Name the documents `Scrolling_LEDs.SchDoc` and `Scrolling_LEDs.PrjFpg` respectively.
4. From the Libraries panel, locate and select the PWM component and click **Place**.
5. While the component is floating on the cursor, press the **Tab** key to open the *Component Properties* dialog.
6. Change the component's **Designator** property to `U1`,
7. then click the **Configure** button on the lower left of the *Component Properties* dialog and confirm that the Pulse Width Modulator is configured as shown in Figure 2, with a base frequency of 20 MHz. This will mean each LED output has a pulse-width-modulated square wave of 78.125 KHz.



Precision (bits): 8

Prescaler: 1

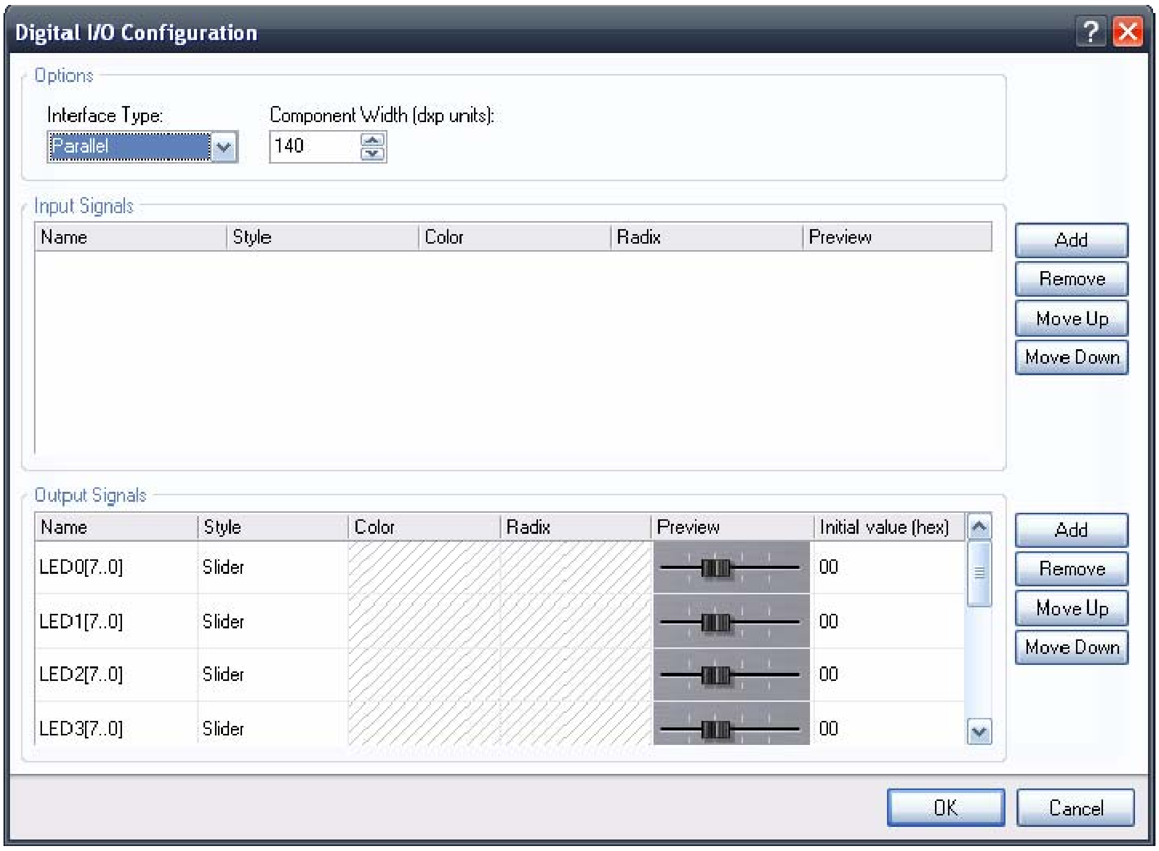
Precision describes the number of bits used to control the duty cycle.
Prescaler stretches the PWM period by the specified multiplier.
The number of clock cycles in a period is: (2^Precision) x Prescaler

PWM period is **12.800 us**

based on an input clock frequency of 20 MHz

Figure 2. Configuration of the Pulse Width Modulator.

8. Close the *PWM* dialog and place the first PWM component in the center of the schematic sheet, toward the top.
9. After placing the first PWM another will appear on the cursor, place it below the first one, and then place another 6, to give a total of 8 PWM components.
10. From the Libraries panel, locate and select the LEDS_RGB component and click **Place**.
11. Place it to the right of the schematic sheet, leaving room to wire it up. Its position can be changed later if needed.
12. Place the CLOCK_REF, DIGITAL_IO and NANOBOARD_INTERFACE components to the left of the PWM components, as shown in Figure 1. Their position is not critical, if needed they can be moved when you wire them up.
13. Select **Tools»Annotate Schematics Quietly** to set the designators of the new components you just placed.
14. Right-click on the DIGITAL_IO component, and select **Configure** from the floating context menu.



Digital I/O Configuration

Options

Interface Type: Parallel

Component Width (dxp units): 140

Input Signals

Name	Style	Color	Radix	Preview

Output Signals

Name	Style	Color	Radix	Preview	Initial value (hex)
LED0[7..0]	Slider				00
LED1[7..0]	Slider				00
LED2[7..0]	Slider				00
LED3[7..0]	Slider				00

Figure 3. Configure the Digital IO to have no inputs, and 8 slider style outputs.

15. The *Digital IO Configuration* dialog will open, by default the instrument will have 1 input signal and 1 output signal, we will now reconfigure it to have no inputs and 8 outputs.
16. Select the one Input Signal and click the adjacent **Remove** button to delete it.
17. Edit the **Name** of the Output Signal, setting it to `LED0(7..0)`, and edit the **Style** of the Output to set it to Slider.
18. **Add** 7 more Output Signals to give a total of 8, setting their style to Slider and their names to `LED1(7..0)` through to `LED7(7..0)`, respectively. The dialog will now look like Figure 3.
19. Click OK to close the dialog, the DIGITAL_IO component will now look like Figure 4.

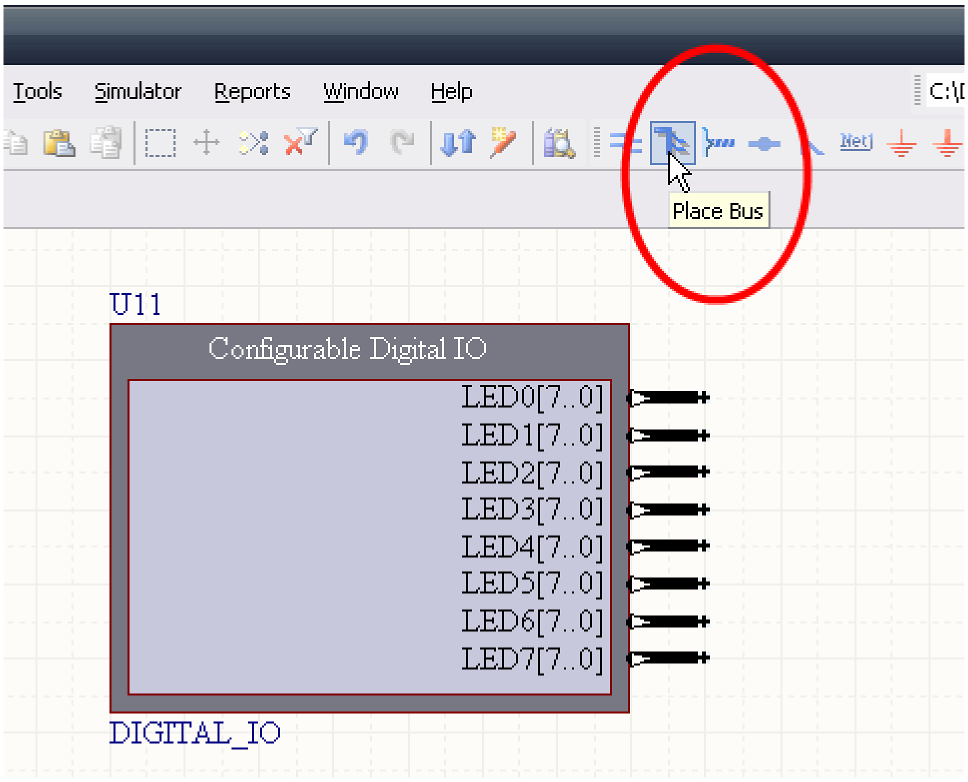


Figure 4. The Digital IO instrument has been configured, ready for wiring.

20. Click the Place Bus button (as shown in Figure 4), and place a short straight length of Bus line (approximately 3 to 4 times the length of the pin) out from each of the pins on the DIGITAL_IO component, as shown in Figure 5.
21. Click the Place Net Label button, then press **Tab** to open the *Net Label* dialog where you can configure the net label before placing it. Type in the net label name, `LED0[7..0]`.
22. Close the dialog and place the net label on the bus line connected to pin LED0(7..0) on the DIGITAL_IO component.
23. After placing the first net label, another will appear floating on the cursor. Press **Tab** again to edit the value, changing it to `LED1[7..0]`. Close the dialog and place the second net label below the first one you placed, as shown in Figure 5.

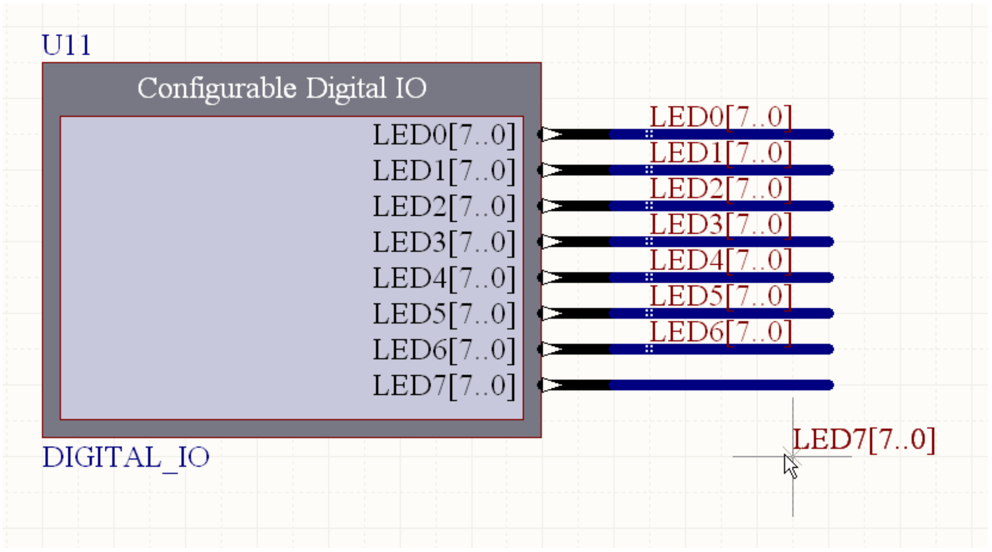


Figure 5. Place a net label on each of the bus lines.

24. Continue to edit and place net labels on each of the bus lines, until they are all labelled, as shown in Figure 5.
25. Each of these buses now needs to be connected to the data input on one of the PWM components. This will be done by placing a bus line+net label on each PWM data input, which you will do by copying and pasting the set of bus+net labels you just connected to the Digital IO. To make a copy of these bus lines and net labels, first click and drag a selection box that completely encloses them. Each selected object will highlight in the current selection color as soon as you release the mouse button.
26. To copy them all, hold the **Shift** key down and click the **Left Mouse Button** anywhere on any of the selected net labels, then while holding the mouse button down, move the mouse to a clear area of the schematic sheet, and release the mouse button to place the set of copied objects, as shown in Figure 6.

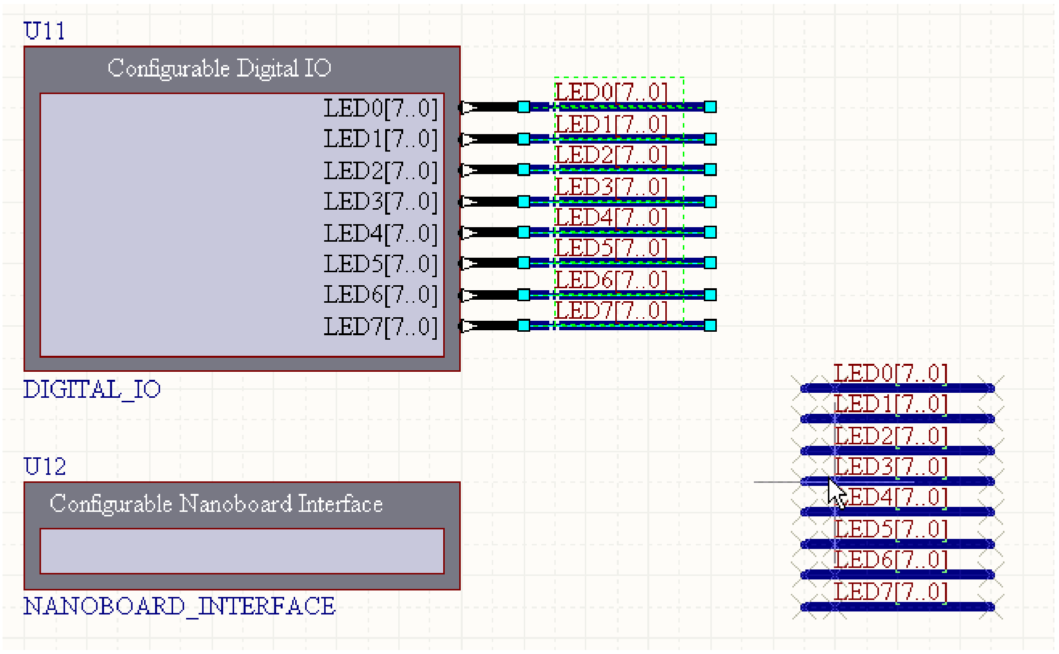


Figure 6. Hold Shift to drag a copy of the current selection.

27. Select the first bus+net label combination, then click and drag to position it so that the bus line touches the D[7..0] pin of the first PWM component, U1, as shown in Figure 7.

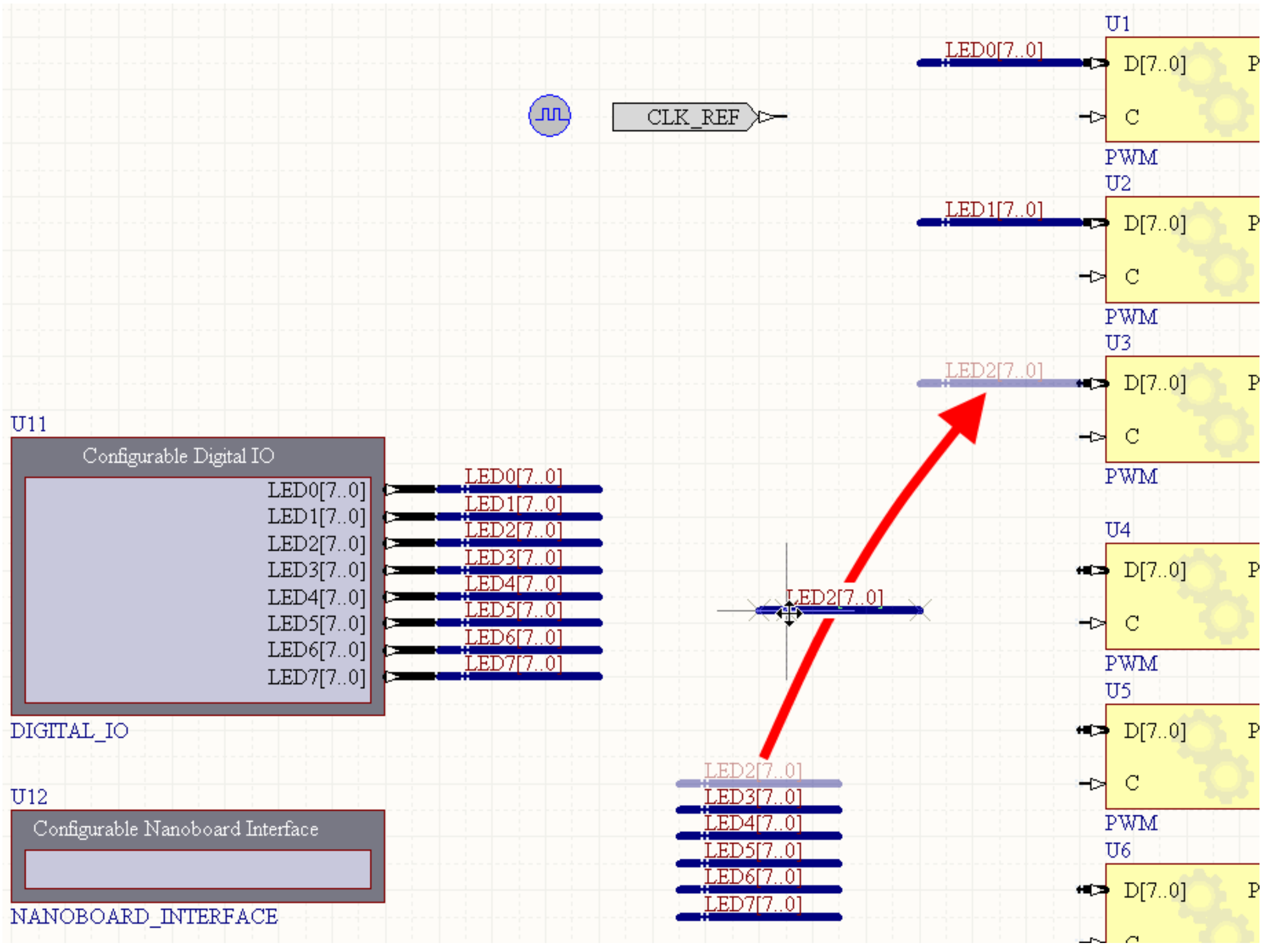


Figure 7. Select and move the copy of each bus+net label in turn.

28. Continue to re-position the remaining seven bus+net label combinations, as shown in Figure 7.
29. Now wire all of the PWM clock pins, labeled C, to the CLK_REF port plugin component, as shown in Figure 1.
30. The output from each PWM will be wired into a bus, which then connects to the red pin of the LEDS_RGB port plugin component, as shown in Figure 8. Note that the wire from each output must include a net label, in the format `LED_OUT0` through to `LED_OUT7`, and the bus line must include a net label in the format `LED_OUT[7..0]`.

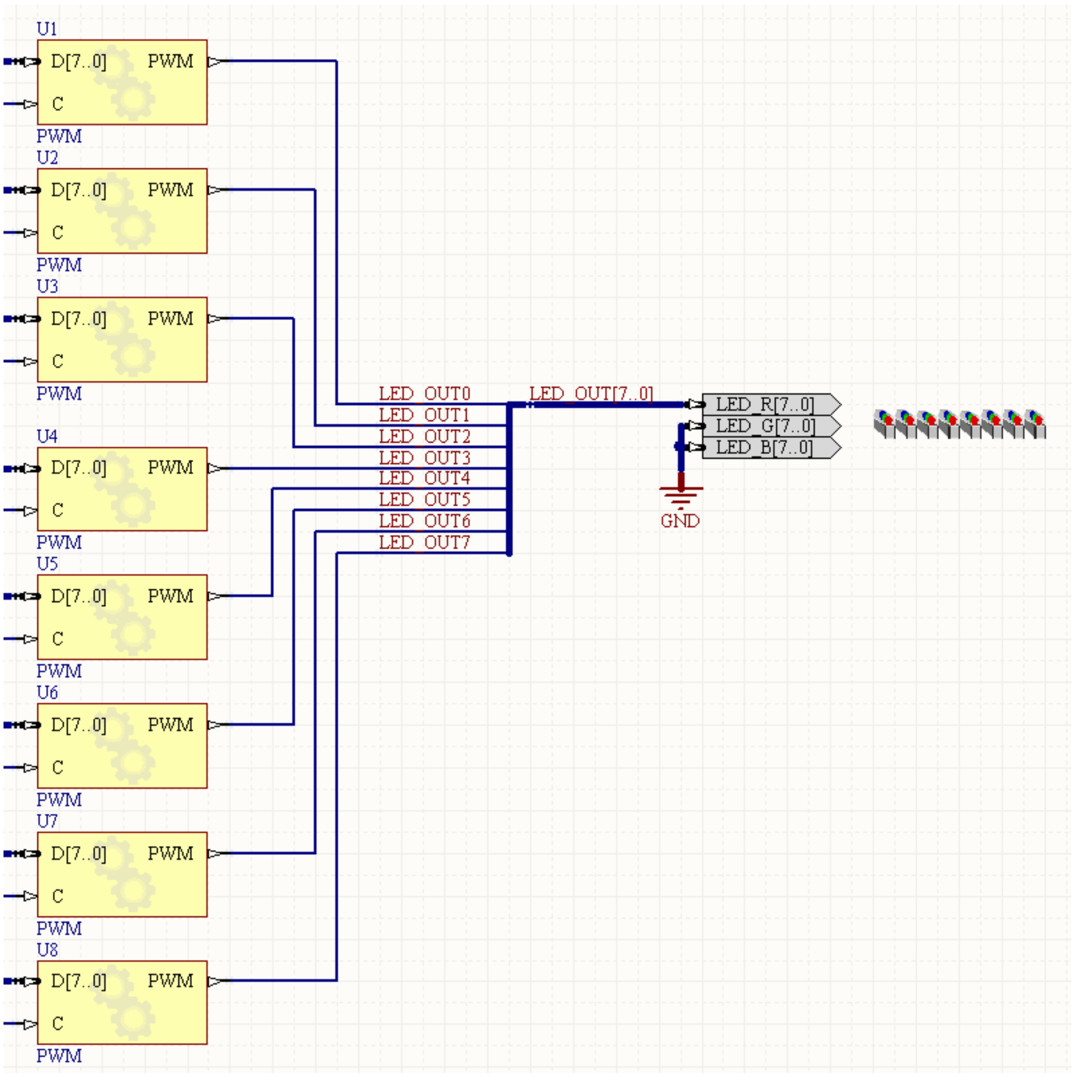


Figure 8. Wire the output of each PWM to the LEDs.

31. Connect the unused green and blue pins on the LEDS_RGB port plugin component to ground by placing a bus style ground port (as shown in Figure 9), and connecting it to the green and blue pins with a bus.

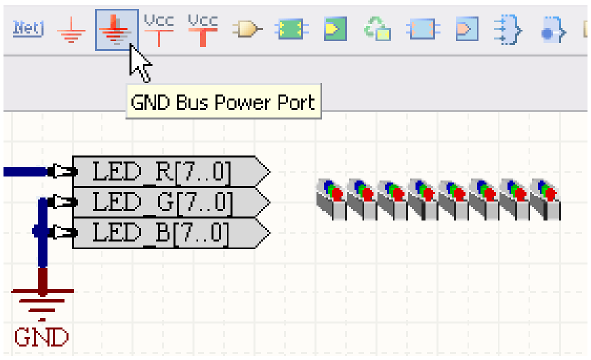


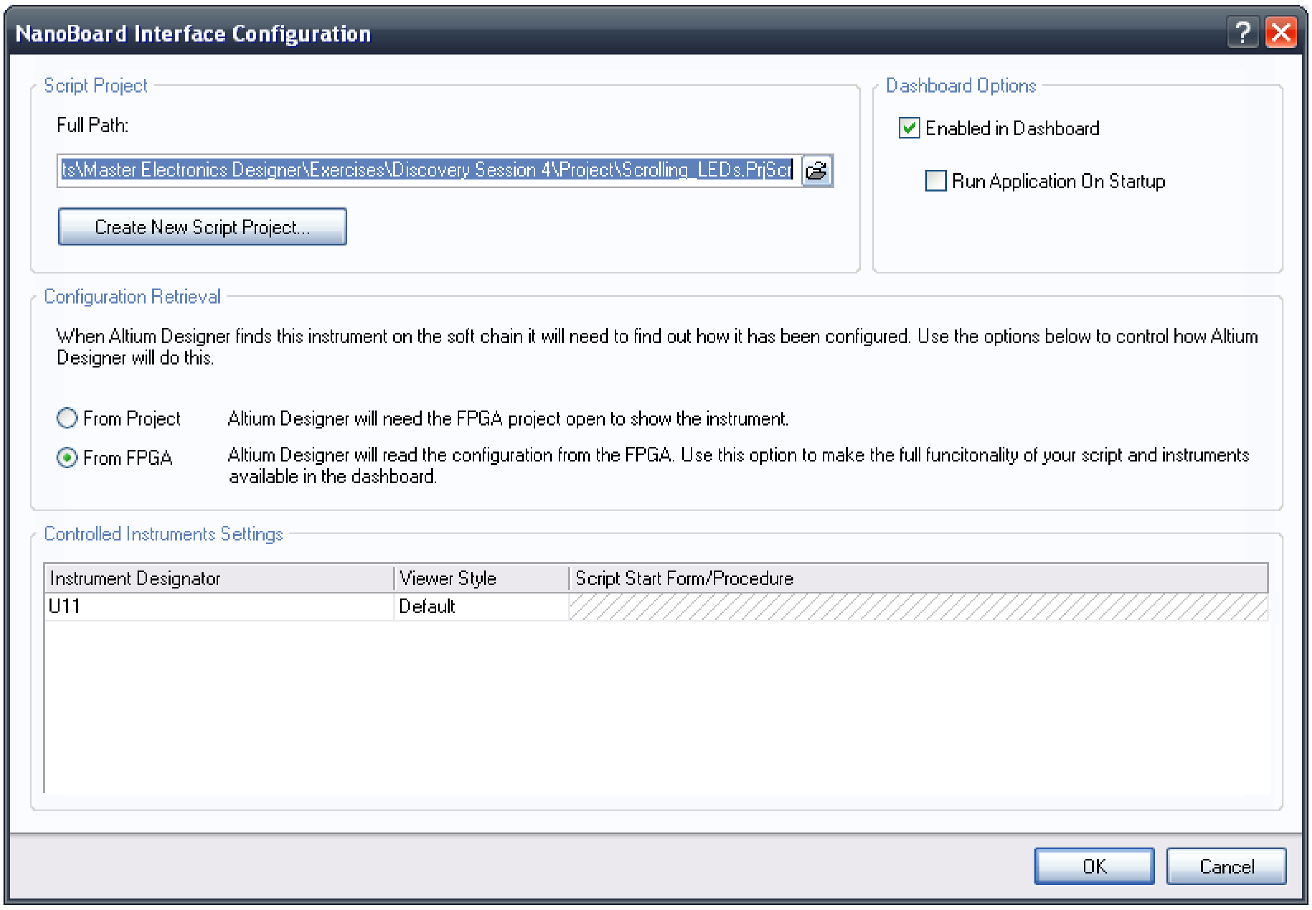
Figure 9. Connect the unused pins to ground.

32. The hardware is now complete, **Compile** the project and check the Messages panel for errors and warnings – resolve these before continuing.

33. Save the schematic and the project file (right-click on each in the Projects panel to save).

Tutorial steps – adding a script to control the Digital IO

34. It is now time to add a script to the Digital IO instrument. The function of the script is to control the Digital IO instrument programmatically, instead of you controlling it manually through the GUI. A script is attached to the Digital IO instrument through the NanoBoard Interface Configuration component. To attach a new script project right-click on the Interface Configuration component and select **Configure** from the floating menu.
35. The *NanoBoard Interface Configuration* dialog will open (Figure 10). Note that the lower region of the dialog shows which instrument is being controlled by the Interface Configuration component – it should display the current designator of this component. Click the **Create New Script Project** button to create and link an Altium Designer Delphi Script project to that instrument.
36. The *Save* dialog will open, navigate to the same folder as the FPGA project, and save the script project with the name `Scrolling_LEDs.PrjScr`.
37. Set the **Configuration Retrieval** option in the *NanoBoard Interface Configuration* dialog to **From FPGA**, as shown in Figure 10. This stores the configuration information in FPGA block ram, allowing the script to be run directly from the Altium Instrument Dashboard (a small application that gives access to FPGA instruments without needing Altium Designer to be installed). Click **OK** to close the dialog.


NanoBoard Interface Configuration

Script Project

Full Path: ts\Master Electronics Designer\Exercises\Discovery Session 4\Project\Scrolling_LEDs.PrjScr

Dashboard Options

[✓] Enabled in Dashboard

[ ] Run Application On Startup

Configuration Retrieval

When Altium Designer finds this instrument on the soft chain it will need to find out how it has been configured. Use the options below to control how Altium Designer will do this.

From Project — Altium Designer will need the FPGA project open to show the instrument.

[✓ From FPGA] — Altium Designer will read the configuration from the FPGA. Use this option to make the full funcitonality of your script and instruments available in the dashboard.

Controlled Instruments Settings

Instrument Designator	Viewer Style	Script Start Form/Procedure
U11	Default	

Figure 10. The script is connected to the Digital IO via the NanoBoard Interface component, and configured to be stored in FPGA block RAM.

38. This new script project is actually a child of the FPGA project, but will not be displayed as a child project yet. To force it to be displayed in the correct position, right-click on the NanoBoard Interface Configuration component again and select **Configure** from the floating menu to re-open the *NanoBoard Interface Configuration* dialog. When it opens the script project will move to its correct location, as shown in Figure 11.

Click **OK** to close the dialog again, then save schematic sheet and the new Script project (right-click on each in the Projects panel to save).

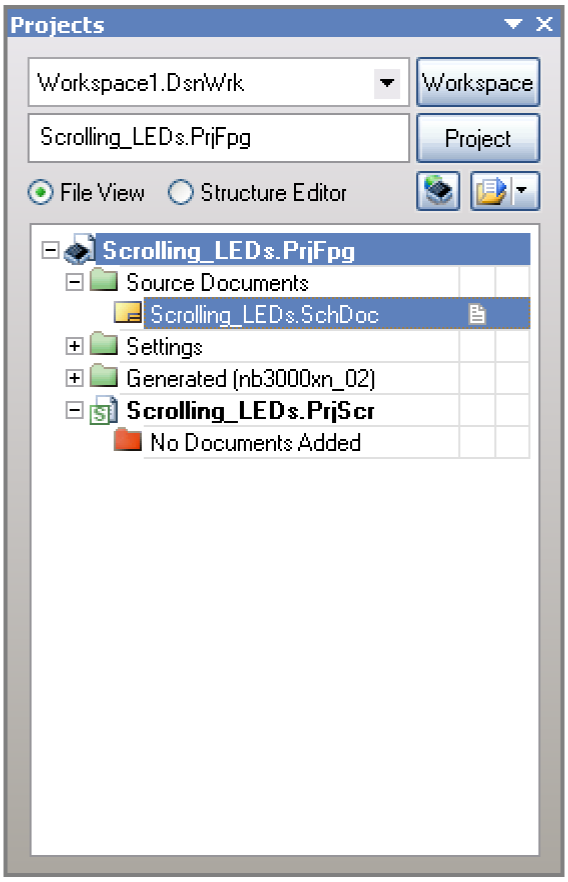


Figure 11. The new script project, shown as a child of the FPGA project.

39. To add a new Delphi script form to the project, right-click on the project in the Projects panel and select **Add New to Project»Delphi Script Form** from the menu.

40. Save the new script form document with the name `Scrolling_LEDs.pas`, and save the script project as well.

41. Select the Form tab along the bottom of the main workspace to enable the Form view, as shown in Figure 12. The blank form will be displayed at the top left of the workspace.

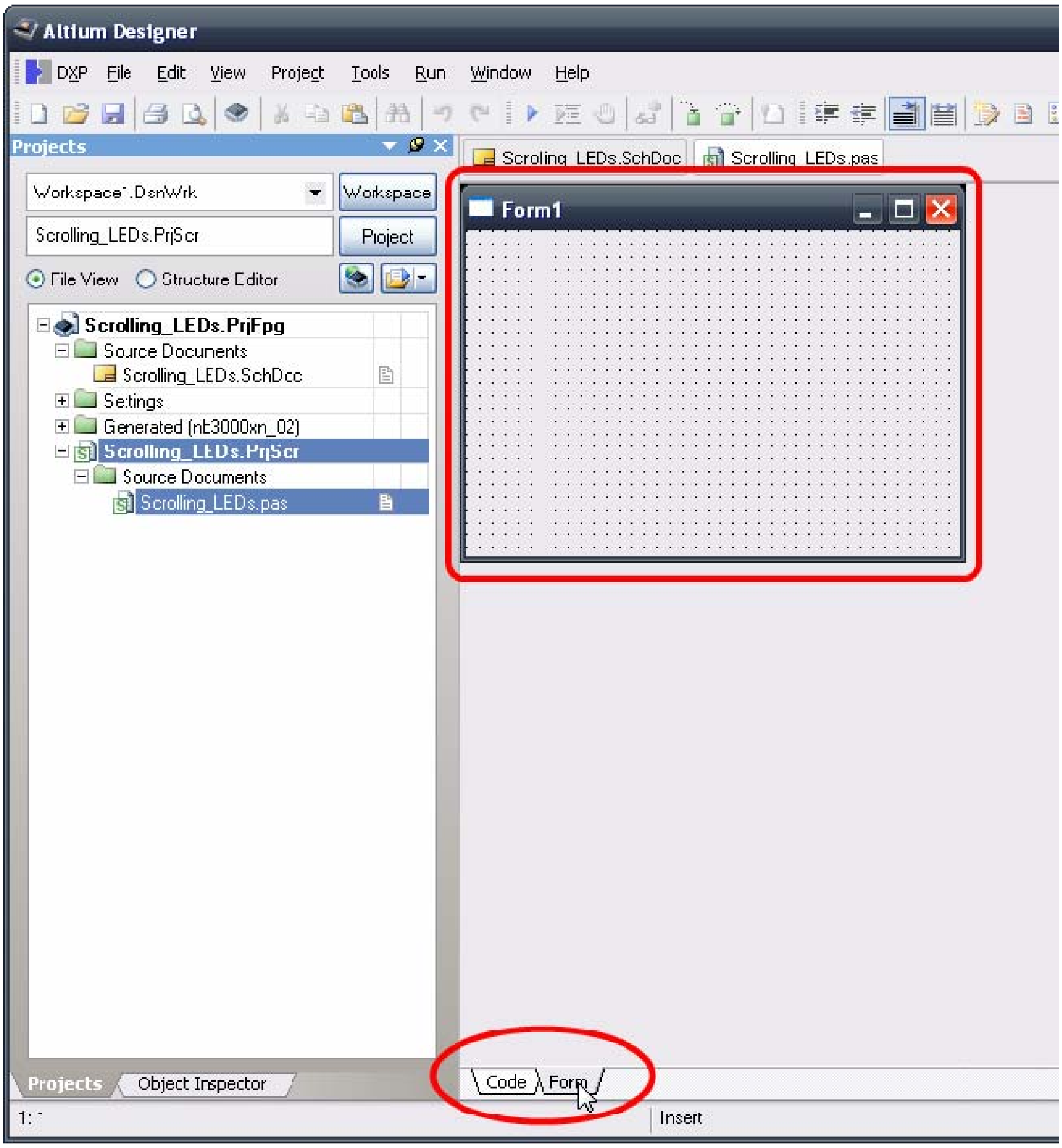
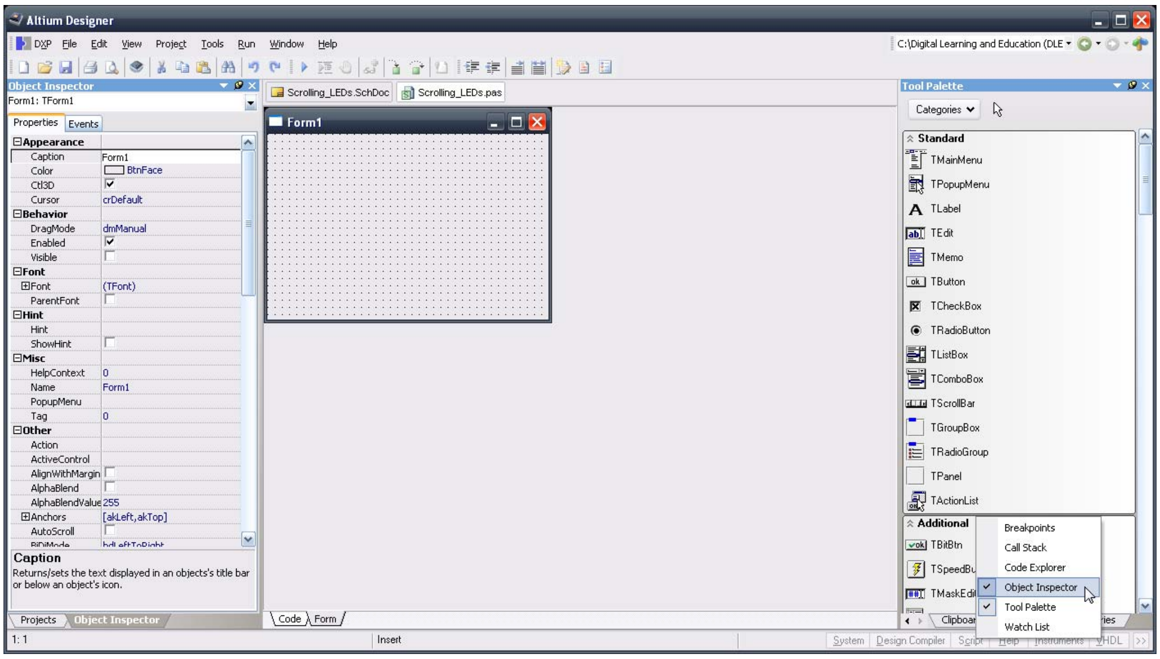


Figure 12. Switch to the Form view, to display the blank form.

42. Open the Tool Palette panel, and dock it on the right hand side of the workspace, as shown in Figure 13. This panel can be opened via the Script menu at the bottom right of the main window, or by selecting **View»Workspace Panels»Script»Tool Palette** from the main menu.
43. Open the Object Inspector, and dock it on the left hand side of the workspace, as shown in Figure 13. This panel can be opened via the Script menu at the bottom right of the main window, or by selecting **View»Workspace Panels»Script» Object Inspector** from the main menu.


Form1: TForm1

Appearance	
Caption	Form1
Color	BtnFace
Ctl3D	✓
Cursor	crDefault
Behavior	
DragMode	dmManual
Enabled	✓
Visible	☐
Font	
Font	(TFont)
ParentFont	☐
Hint	
Hint	
ShowHint	☐
Misc	
HelpContext	0
Name	Form1
PopupMenu	
Tag	0
Other	
Action	
ActiveControl	
AlignWithMargin	☐
AlphaBlend	☐
AlphaBlendValue	255
Anchors	[akLeft,akTop]
AutoScroll	☐

Caption
Returns/sets the text displayed in an objects's title bar or below an object's icon.

Figure 13. Main workspace configured for scripting

44. Click once in the center of the form, then in the Object Inspector panel locate the form's **Caption** property. Modify the **Caption** property to `Scrolling LEDs`. Press **Enter** to confirm the change.

45. Locate the form's **Name** property and change the form's name from `Form1` to `xMainForm`. Press **Enter** to confirm the change.

46. From the **Tool Palette** panel, locate the **Instrument Controls** section at the bottom of the list of controls. Click once on the `TInstrumentBackgroundPanel` control, then click once on the form, to place that control onto the form.

47. Position the cursor over the lower right corner of the form until the cursor changes to a double-headed diagonal arrow ⤡. Resize the form to be approximately the same size as it appears in Figure 14.

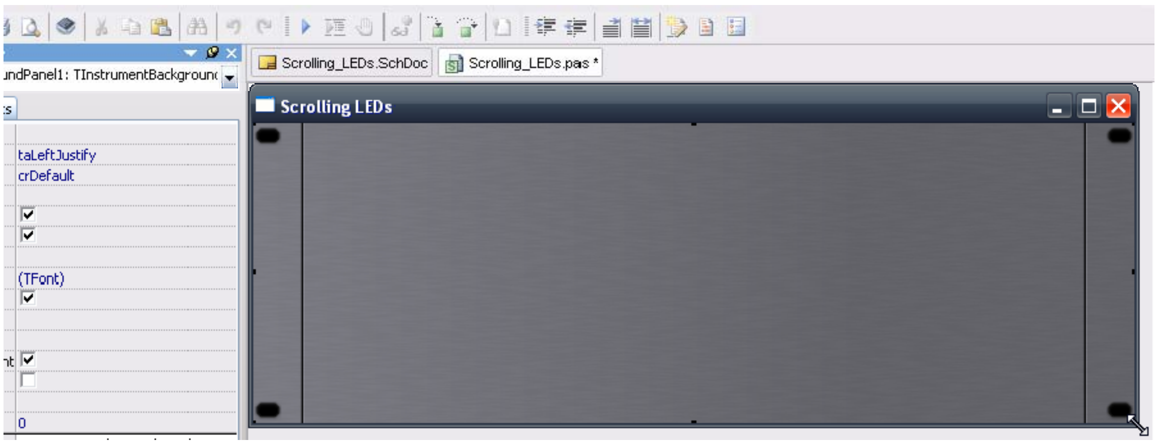
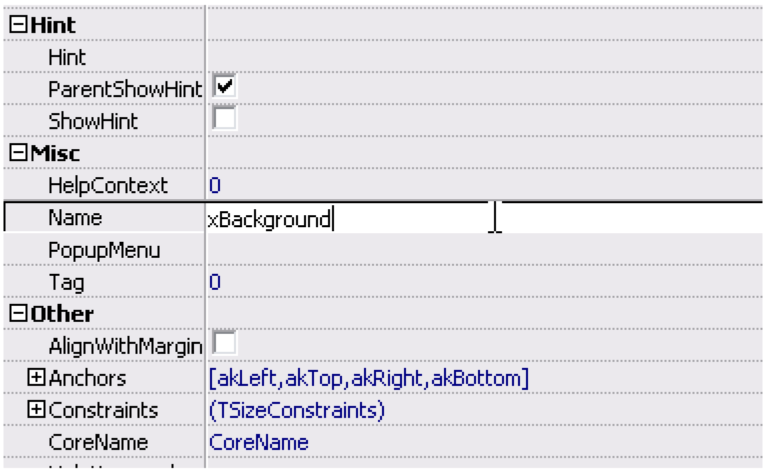


Figure 14. Resizing the form with the Instrument Panel applied

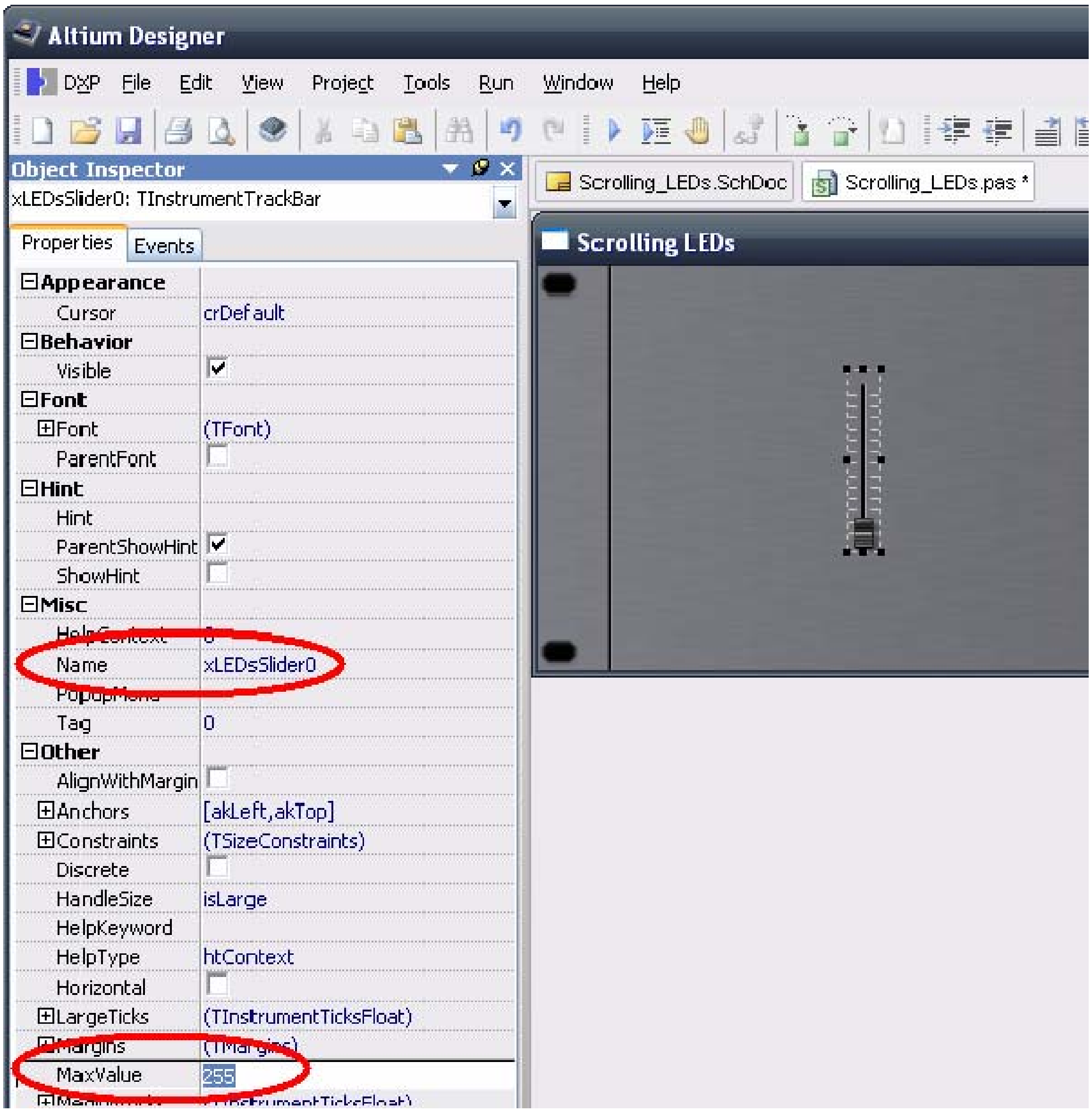
48. Click once in the center of the form to select the background, then from the Object Inspector panel locate the background's **Name** property. Change it to `xBackground`, and press **Enter** to confirm the change (Figure 15).



⊟**Hint**	
Hint	
ParentShowHint	✓
ShowHint	☐
⊟**Misc**	
HelpContext	0
Name	xBackground
PopupMenu	
Tag	0
⊟**Other**	
AlignWithMargin	☐
⊞Anchors	[akLeft,akTop,akRight,akBottom]
⊞Constraints	(TSizeConstraints)
CoreName	CoreName

Figure 15. Naming the background control.

49. The next step is to place 8 track bars. Scroll down the Tool Palette to the Instrument Controls section, then locate the **TInstrumentTrackBar**, click once on it, then click on the form to place the first slider on it.
50. Click once to select the new TrackBar slider, then in the Object Inspector panel set the Name of this first one to `xLEDsSlider0`, and the MaxValue to `255`, as shown in Figure 16.



xLEDsSlider0: TInstrumentTrackBar

⊟Appearance	
Cursor	crDefault
⊟Behavior	
Visible	✓
⊟Font	
⊞Font	(TFont)
ParentFont	☐
⊟Hint	
Hint	
ParentShowHint	✓
ShowHint	☐
⊟Misc	
HelpContext	0
Name	xLEDsSlider0
PopupMenu	
Tag	0
⊟Other	
AlignWithMargin	☐
⊞Anchors	[akLeft,akTop]
⊞Constraints	(TSizeConstraints)
Discrete	☐
HandleSize	isLarge
HelpKeyword	
HelpType	htContext
Horizontal	☐
⊞LargeTicks	(TInstrumentTicksFloat)
⊞Margins	(TMargins)
MaxValue	255

Figure 16. Name the TrackBar and set the MaxValue.

51. Right-click the new TrackBar and select **Copy** from the floating menu.
52. Right-click away from the first TrackBar and select **Paste** from the floating menu – a second TrackBar slider will appear – now click and hold on the second TrackBar and drag to position it next to the first one.
53. Continue to **Paste** in TrackBar sliders, repositioning each one, until there is a total of eight arranged in a row, as shown in Figure 19.
54. Now click on each TrackBar slider in turn, and set the **Name** property to `xLEDsSlider0` through to `xLEDsSlider7`.
55. Save the Scrolling_LEDs.pas file.
56. The next step is to connect the IO signals of the DIGITAL_IO instrument through to the variables in your code.

    This is done using a TSignalLinkManager. Locate the **TSignalLinkManager** control in the Instrument Controls section of the Tool Palette panel. Click once on the control, then again on the form to place the control onto the form. Since this control will not be visible it can be placed anywhere on the form, typically it is placed to one side, as shown in Figure 19.

57. Click once on the icon on the form to select the control and from the Object Inspector panel, modify the control's **Name** property to `xSignals`, and the **PollingInterval** property to `100`. Press **Enter** to confirm each change.

58. Double click the icon on the form to launch the **Editing xSignals.SignalLinks** dialog seen in Figure 17.

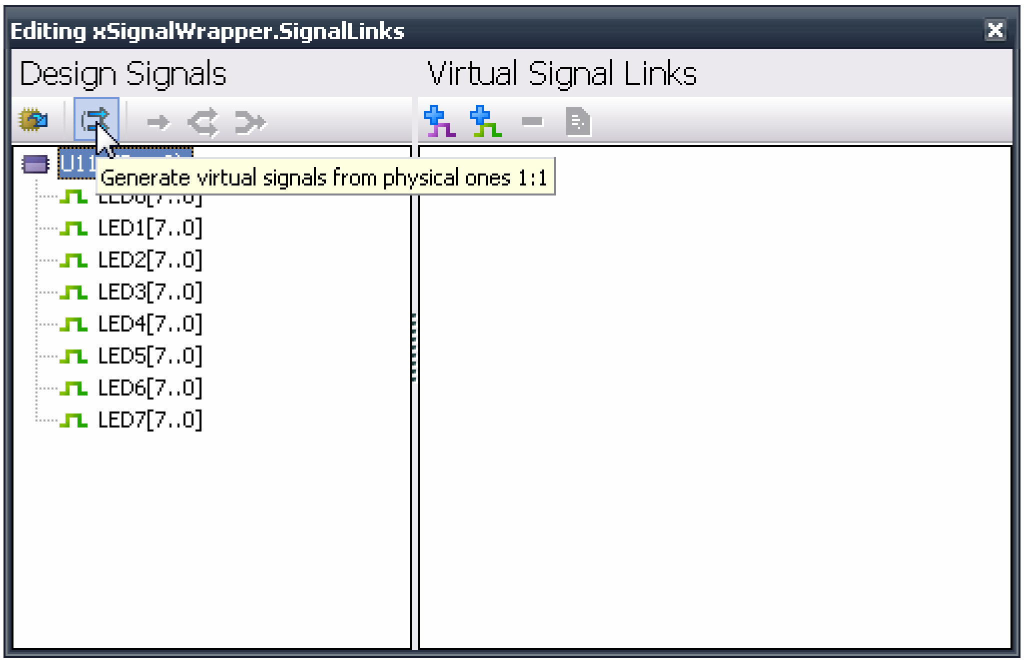
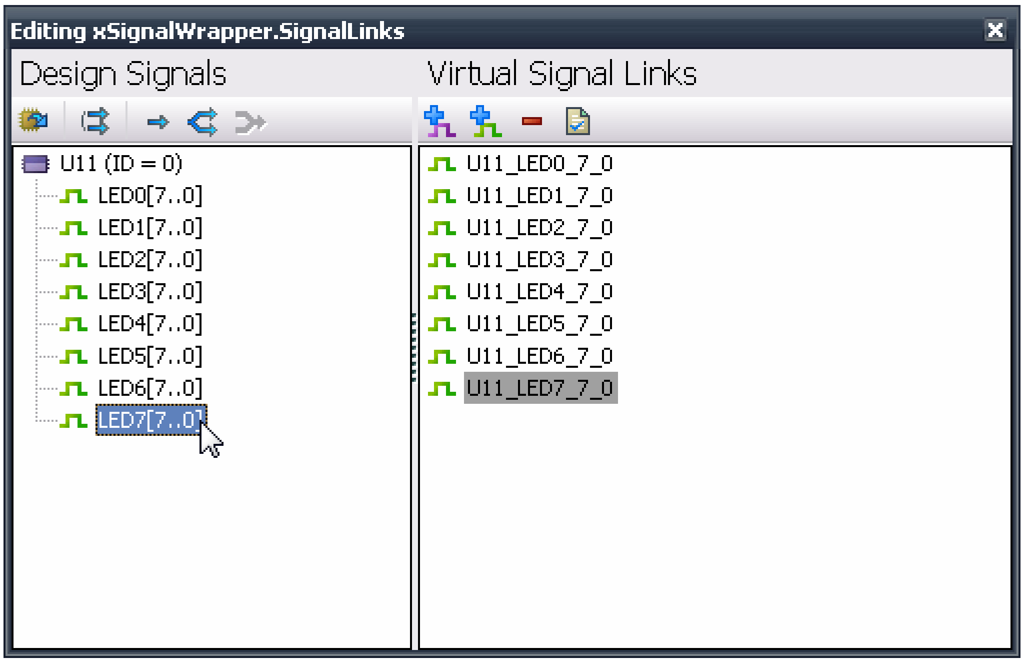


Figure 17. Existing design signals are connected to virtual signals.

59. This dialog is used to connect the signals in your design to internal (virtual) signals used in your code. Click once on the button to automatically generate virtual signals for each of the design signals, as shown in Figure 17. As you click on a design signal name in the dialog, its corresponding virtual signal will be highlighted, indicating that they are connected, as shown in Figure 18.

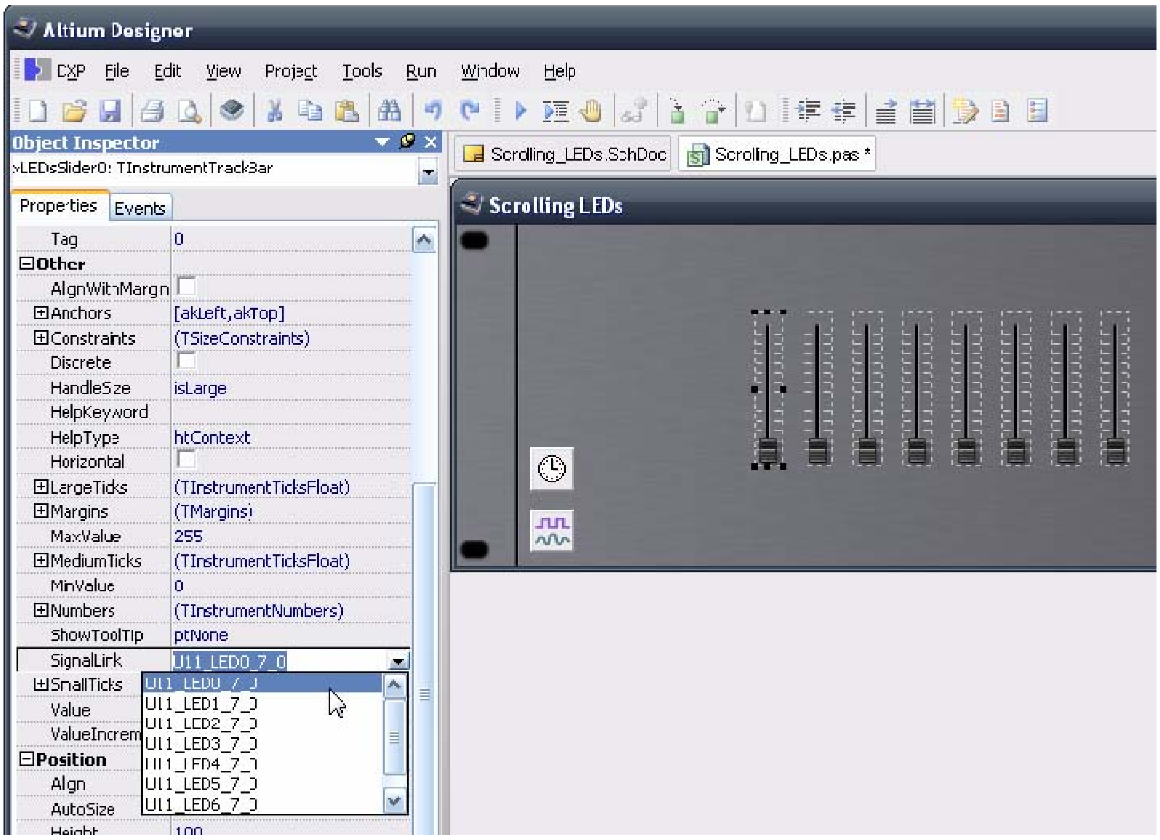


Editing xSignalWrapper.SignalLinks

Design Signals	Virtual Signal Links
U11 (ID = 0)	U11_LED0_7_0
LED0[7..0]	U11_LED1_7_0
LED1[7..0]	U11_LED2_7_0
LED2[7..0]	U11_LED3_7_0
LED3[7..0]	U11_LED4_7_0
LED4[7..0]	U11_LED5_7_0
LED5[7..0]	U11_LED6_7_0
LED6[7..0]	U11_LED7_7_0
LED7[7..0]	

Figure 18. Design signals have been linked to virtual signals.

60. The last step to bind each TrackBar slider to the appropriate signals in the design. To do this, click to select the first TrackBar, then in the Object Inspector locate the **SignalLink** property and set it to the first of the Virtual Signals, namely `U11_LED0_7_0`, as shown in Figure 19.



xLEDsSlider0: TInstrumentTrackBar

Properties | Events

Tag	0
⊟Other	
AlignWithMargin	☐
⊞Anchors	[akLeft,akTop]
⊞Constraints	(TSizeConstraints)
Discrete	☐
HandleSize	isLarge
HelpKeyword	
HelpType	htContext
Horizontal	☐
⊞LargeTicks	(TInstrumentTicksFloat)
⊞Margins	(TMargins)
MaxValue	255
⊞MediumTicks	(TInstrumentTicksFloat)
MinValue	0
⊞Numbers	(TInstrumentNumbers)
ShowToolTip	ptNone
SignalLink	U11_LED0_7_0
⊞SmallTicks	U11_LED0_7_0
Value	U11_LED1_7_0
ValueIncrem	U11_LED2_7_0
⊟Position	U11_LED3_7_0
Align	U11_LED4_7_0
AutoSize	U11_LED5_7_0
Height	U11_LED6_7_0

Figure 19. Link each TrackBar slider to the virtual signal.

61. Repeat this for the remaining 7 TrackBars, selecting the appropriate signal name in each case.

62. The code uses a timer to trigger when the outputs are activated. To include the timer, locate and click to select the TTimer from the System section of the Tool Palette. Once it is active, click once on the form to place it above the TSignalLinkManager icon.

63. While the TTimer is still selected, edit the **Name** property in the Object Inspector panel to `xTimer`, and the Interval to `75`. Press **Enter** to confirm each change. The form should now look like Figure 20.

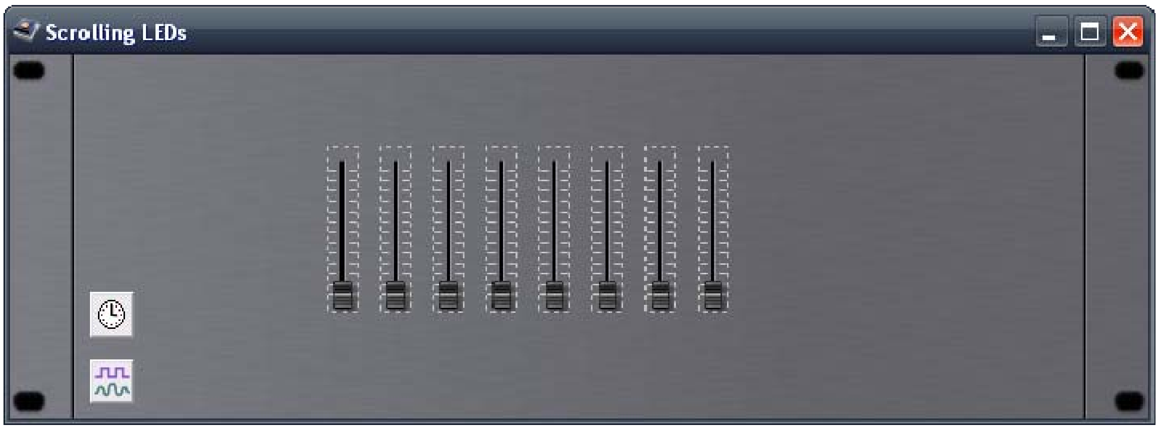


Figure 20. The instrument with 8 TrackBars, the TSignalLinkManager and the TTimer.

64. Now it's time to write the code. First we will write a procedure that runs automatically when the form is created. To be able to link code to the form creation you must first select the actual form itself, not the background you placed on top of the form. When you click once to select the form it is actually the background that is selected, as indicated by the small selection squares that appear in each corner and along the middle of each edge. To select the parent form, hold Ctrl as you click on the form. The Object Inspector will now show the name of the form, **Scrolling LEDs**, indicating that you have correctly selected the parent form.

65. Change to the Events Tab in the Object Inspector, and edit the **OnCreate** event, entering the string `OnCreate`. When you press **Enter** to confirm the change the Code Tab will be made active, showing the shell of the new procedure you have just created. Note that you can switch between the Code and the Form at any time, using the small Tabs at the bottom of the *.pas file.

66. Before entering the code for the new procedure, you will need to define the constants and the variables, as shown below in Figure 21.

67. Once that is done, enter the code for the **OnCreate** procedure, as shown in Figure 21. This routine will initialize the `LEDScanIndex` and `LEDScanIndexModifier` variables on startup.

```
const
   ScanArrayLimit = 7;

var
   ScanArray : array[0..ScanArrayLimit] of Integer;
   LEDScanIndex : Integer;
   LEDScanIndexModifier : Integer;

procedure TxMainForm.OnCreate(Sender: TObject);

begin
     LEDScanIndex := 0;
     LEDScanIndexModifier := 1;

end;
```

Figure 21. Declare the constants & variables, and start writing the code.

68. The next step is to connect code to the event that occurs each time the Timer ticks. To do that, switch back to the Form view using the Tab at the bottom of the code editing window, and click once to select the **TTimer** object. In the Events Tab of the Object Inspector, edit the **OnTimer** property value to be `TimerTick`. When you press Enter to apply the value you will be switched back to the Code view, ready to enter the code, which is shown below in Figure 22.

```
//Scanning lights
procedure TxMainForm.TimerTick(Sender: TObject);

var
   i : integer;
   PreScanValue : integer;


begin
     PreScanValue := 100;

     for i := 0 to ScanArrayLimit do
     begin
         if ScanArray[i]  = PreScanValue then
            ScanArray[i] := 255
         else
            ScanArray[i] := ScanArray[i] * 0.5;
     end;


     //current item = 100
     ScanArray[LedScanIndex] := PreScanValue;


     //modify the index on this clock tick
     LEDScanIndex := LEDScanIndex + LEDScanIndexModifier;


     //when it gets to the end, send it back the other way
     if (LEDScanIndex = ScanArrayLimit) or (LEDScanIndex = 0) then
        LEDScanIndexModifier := LEDScanIndexModifier * -1;

     xLEDSSlider0.Value := ScanArray[0];
     xLEDSSlider1.Value := ScanArray[1];
     xLEDSSlider2.Value := ScanArray[2];
     xLEDSSlider3.Value := ScanArray[3];
     xLEDSSlider4.Value := ScanArray[4];
     xLEDSSlider5.Value := ScanArray[5];
     xLEDSSlider6.Value := ScanArray[6];
     xLEDSSlider7.Value := ScanArray[7];

end;
```

Figure 22. Enter the code that runs the pattern back and forth along the LEDs.

69. Select **File»Save All** to save your work.

70. Switch to the Devices view by selecting **View»Devices View** from the menu, or clicking the button on the main toolbar.

71. Ensure the NanoBoard 3000 is connected to your PC via a suitable USB cable, and that the **Live** checkbox in the Devices view is checked. If it is correctly connected an icon for the NanoBoard will appear at the top of the Devices view and the small **Connected** icon at the top left of the Devices view will go green, as shown in Figure 23.

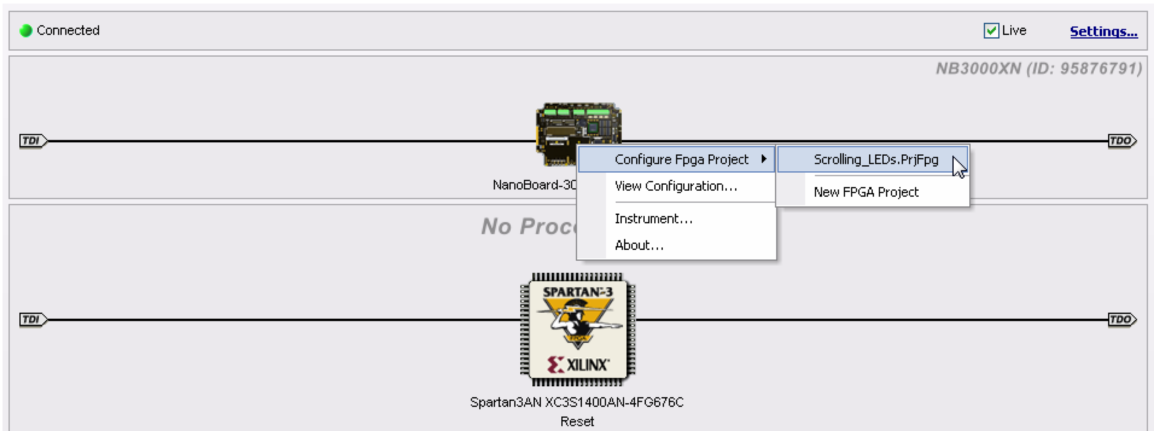


Figure 23. Configuring the project, this connects the signals on the NanoBoard through to the pins used in the FPGA design.

72. Right-click the NanoBoard icon and select **Configure FPGA Project»Scrolling_LEDs.PrjFpg** to auto-configure the design, as shown in Figure 23. The *Configuration Manager* dialog will open, click **OK** to accept the configuration and close the dialog. The build flow buttons should now appear just below the NanoBoard icon, as shown in Figure 24.
73. Click the **Program FPGA** button in the build flow to run through the stages up through and including programming the device, closing the *Result Summary* dialog that appears at the end of the process.

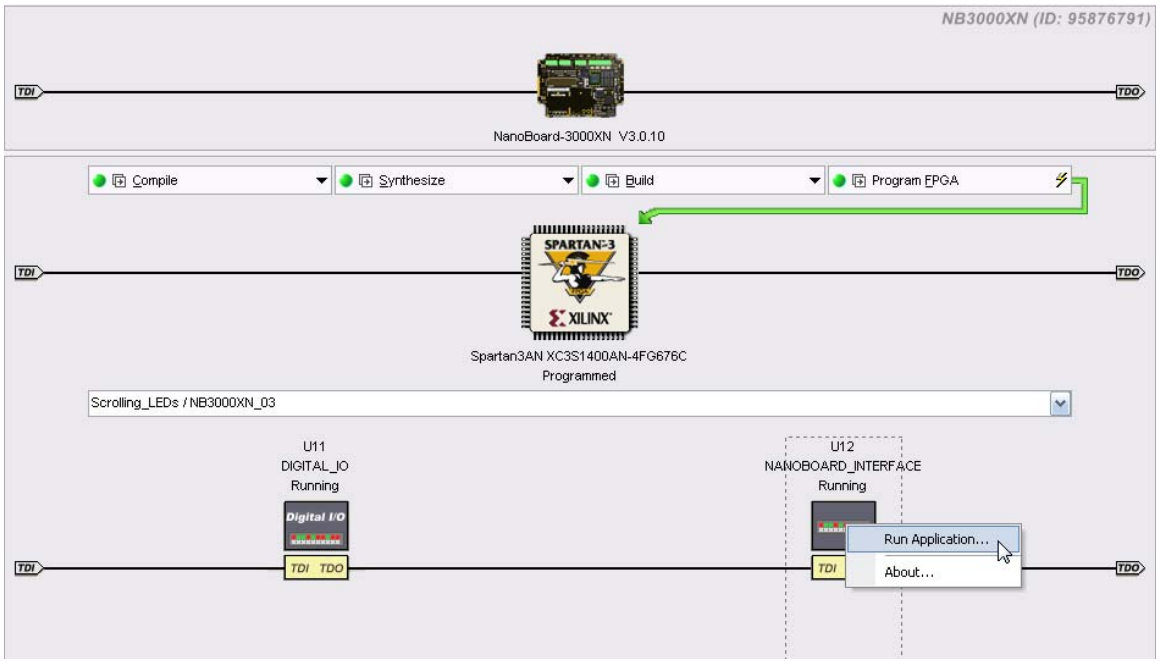


Figure 24. The Devices view, note that the Build flow is all green, indicating that the design has been successfully compiled and programmed into the target FPGA.

74. In the Devices view, right-click the **NANOBOARD_INTERFACE** component in the Soft Devices chain and select **Run Application**, as shown in Figure 24. The component's interface will open and the sliders will slide up and down, under the control of the script you just wrote. The LEDs will also illuminate, creating the desired Knight Rider pattern.

Some additional items for you to try

Below are some suggestions for additional things you might try as you further explore the tools and the new environment. These changes are not required by later exercises and it's recommended you save a backup of your current source files first

- Extend the project hardware to include Red, Green *and* Blue LEDs.
- Extend the arrays and instrument sliders to scroll through all three LED colors.
- Extend the script code to have three independent timers and scrolling routines for Red, Green and Blue arrays.

Revision History

Date	Revision No.	Changes
30-Jul-2009	1.0	New document release