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
U? → D[70] PWM → C PWM	FPGA Configurable Generic.IntLib	PWM
-⇒ LED_R[7.0] -> LED_G[7.0] -> LED_B[7.0]	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.



Configure U1 (P)	NM)	? 🔀
Pulse W	/idth Modulator	
Precision (bits): Prescaler:	8	
Precision describ Prescaler stretch The number of c	es the number of bits used to contro nes the PWM period by the specified lock cycles in a period is: (2^Precision	l the duty cycle. multiplier. n) × Prescaler
Information PWM period is 1	2.800 us	
based on an inp	ut clock frequency of 20	MHz 💌
	<u>o</u> k	Cancel

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.



igital I/O Cont	figuration						? 🔀
Options Interface Type: Parallel	Com 140	oonent Width (dxp	units):				
Input Signals —	Stule	L.a.	lor	Badiv	Provinu		
							Remove Move Up Move Down
Output Signals -	Chile	Color	Dadiu	Proviou	Initial value (bev)		
LED0[70]	Slider						Remove
LED1[70]	Slider						Move Up
LED2[70]	Slider			//// —	00		Move Down
LED3[70]	Slider			//// 	00	~	
					OK		Cancel

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.



_								
<u>T</u> ools	<u>S</u> imulator	<u>R</u> eports	<u>W</u> indow	<u>H</u> elp		1		[⊂:\
à 🖺		+ 🛪 🛪	7 9 (*	11 🎾	1	- 🍡 }••	-	<u>Net</u>) 🛓 🛓
						Place	Bus	
					{			
	U11					\sim	/	
		Configu	able Dig	ital IO				
				LED	0[70]			
				LED	1[70]	(+		
				LED	2[70] 3[7.01			
				LED	4[70]			
				LED	5[70]			
				LED	6[70]	(
				LED	o7[70]			
	DIGII	AL IO						

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 *integrable*, 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.



Configurable Digital IO	
LED0[70] LED1[70] LED2[70] LED3[70] LED4[70] LED5[70] LED6[70] LED6[70]	LED0[70] LED1[70] LED2[70] LED3[70] LED4[70] LED5[70] LED5[70]
DIGITAL_IO	LED7[70]

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.

U11	
Configurable Digital IO	
LED0[70] LED1[70] LED2[70] LED3[70] LED4[70] LED5[70] LED6[70] LED6[70]	LED0[7.0] LED1[7.0] LED2[7.0] LED3[7.0] LED4[7.0] LED5[7.0] LED6[7.0] LED6[7.0]
DIGITAL_IO	
	LED1[7.0]
U12	LED3[70]
Configurable Nanoboard Interface	LED5[70] LED5[70]
NANOBOARD_INTERFACE	

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			/ Dashboard Options
Full Path:		Enabled in Dashboard	
ts\Master Electronics [Designer\Exercises\Discovery Session	4\Project\Scrolling_LEDs.PriScr	
			Run Application Un Startup
Create New Script	Project		
Configuration Rotrieval			
Autor Alive Designed	Gooda their instrument and the applications in		mund the she actions below to control how Alt
Designer will do this.	rinds this instrument on the soft chain it	t will need to find out how it has been config	gured. Use the options below to control how Altium
O From Project A	Itium Designer will need the FPGA proj	ject open to show the instrument.	
From FPGA A a	ltium Designer will read the configurati vailable in the dashboard.	on from the FPGA. Use this option to make	the full funcitonality of your script and instruments
Controlled Instruments Se	ettings		
Instrument Designator	Viewer Style	Script Start Form/Procedure	
U11	Default		
			OK Cancel

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.





S Altium Designer	
DXP File Edit View Project Tools Run	<u>W</u> indow <u>H</u> elp
1 D 🐸 🖬 🖨 🖪 🗢 👗 🖛 🛤 🔊	N I ► E 0 2 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
Projects 🔹 🗸 🖉 🗙	Scroling LEDs.SchDoc 🚮 Scrolling LEDs.pas
Workspace".DsnWrk 👻 Workspace	Form1
Scrolling_LEDs.PrjScr Project	
⊙ File View ○ Structure Editor	
Scrolling_LEDs.PrijFpg Source Documents Scrolling_LEDs.SchDcc Settings Generated (nt3000xn_02) Scrolling_LEDs.PrijScr Source Documents Scrolling_LEDs.pas	
1:-	Insert

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.

💐 Altium Desig	ner		
DXP Eile E	dit <u>V</u> iew Proje <u>c</u> t <u>T</u> ools <u>R</u> un	<u>Window</u> Help	🛛 C:\Digital Learning and Education (DLE 🔻 🔇 👻 💮
		日日の創業を行う使用しているという	
Object Inspector	× 2 ▼		Tool Palette 🔷 👻 🖉 🗙
Form1: TForm1	-	Scrolling_cebs.scroller Spiscrolling_cebs.pas	Estamin M
Properties Events	s	💻 Form1 💦 🗖 🔀	Categories • K
□Appearance			🕆 Standard
Caption	Form1		"≣ TMainMenu
Color	BtnFace		
CtI3D			TPopupMenu
Cursor	crDefault		A TLabel
DragMode	draftaceus		
Enabled			[ab]] TEdit
Visible			TMemo
⊡Font			
⊞Font	(TFont)		_ok_ TButton
ParentFont			TCheckBox
⊟Hint		·····	
Hint			IHadioButton
EMisc	-		TListBox
HelpContext	0		
Name	Form1		E I ComboBox
PopupMenu			TScrolBar
Tag	0		
⊡0ther			LaroupBox
Action			TRadioGroup
ActiveControl			
Alighteend			I Panel
AlphaBlendValu	JE 255		R TActionList
⊞Anchors	[akLeft,akTop]		Additional
AutoScroll			× Additional Breakpoints
RiDiMode	hdi aftToDiabt		vok) TBitBtn Call Stack
Deturorisete the to	wet displayed in an objecte's Pile have		TSpeedBu Code Explorer
or below an object	s icon.		V Object Inspector
			TMaskEdit Tool Palette
Projects Obje	ect Inspector /	\Code \Form /	Clipboar Watch List
1:1		Insert	System Design Compiler Script Herp Instruments YHDL >>

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 \mathbb{Z}_2 . 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.



DSP Ele Edit Yiew Project Iools Run Window Help Diject Inspector Ispector Ispector Ispector Ispector xLEDsSlider0: InstrumentTrackBar Ispector Ispector Ispector Properties Events Ispector Ispector Ispector Cursor crober ault Ispector Ispector Ispector Visible Image: Ispector Ispector Ispector Properties Events Ispector Ispector Visible Image: Ispector Ispector Ispector ParentFont Ispector Ispector Ispector Bisc Ispector Ispector Ispector Name xLEDsSlider0 Popuprise Ispector Popuprise Ispector Ispector Ispector AlignWithMargin Ispector Ispector HandleSize IsLarge IsLarge HelpType htContext Ispector Horizontal Iterarge Ispector BlargeTicks (InstrumentTicksFloat) Image: Ispector MaxWalue Isse Isse	💐 Altium Design	er	
Diject Inspector xLEDsSlider0: TInstrumentTrackBar Properties Events Bappearance Cursor cursor cursor crDef ault Behavior Visible Visible Front BFont (TFont) ParentFont Bhint Hint Handesize AlignWithMargin BAnchors Balkeft, akTop] Biother Handesize BLargeTicks (TinstrumentTicksFloat) BuargeTicks (TinstrumentTicksFloat)	D <u>X</u> P <u>File</u> <u>E</u> c	lit <u>V</u> iew Proje <u>c</u> t <u>T</u> ools <u>R</u> un	<u>W</u> indow <u>H</u> elp
Object Inspector ▼ Ø × xLEDsSilder0: TInstrumentTrackBar ▼ Properties Events □Appearance ✓ Cursor crDefault □Behavior ✓ Visible ✓ □Front □Front □Front □Front □Front □ □PrentFront □ □Hint □ HolpGontext ○ Name xLEDsSider0 Popophone □ Tag 0 □Other □ AlignWithMargin □ □Anchors [akLeft,akTop] □Constraints (TiszeConstraints) □Discrete □ □HandeSize islarge HelpKeyword KCortext Hoizontal □ □BLargeTicks (TinstrumentTicksFloat) □Dwargins (TinstrumentTicksFloat)	0 🗃 🖬 🖉	0	0 > EE 🕘 2 2 2 0 4 # # 2 1
xLEDsSlider0: TInstrumentTrackBar Properties Events Cursor Cursor crDefault Behavior Visible Ifont Ifont Bront Inint Hint ParentShowHint Imint Hint HolpGontext Name xLEDsSlider0 Poperties IdagnWithMargin BAnchors Idkleft,akTop BConstraints Discrete HandleSize IsLarge HelpKyword HelpKyword Horizontal BLargeTicks (TinstrumentTicksFloat) MaxValue	Object Inspector	▼ 🖉 ×	Scrolling EDo SchDool 🖓 Scrolling EDo neo t
Properties Events EAppearance	xLEDsSlider0: TInstru	imentTrackBar 🚽	
EAppearance Cursor crobefault Behavior Visible Image: Starge HandleSize AlignWithMargin BAnchors [akLeft,akTop] BiConstraints (TSeeConstraints) Discrete HandleSize HandleSize HelpType htContext Horizontal BiLargeTicks (TinstrumentTicksFloat) BilargeTicks (TinstrumentTicksFloat)	Properties Events		Scrolling LEDs
Cursor crDef ault Behavior Visible Visible Visible Font Bront (TFont) ParentFont Hint Hint ParentShowHint V ShowHint ParentShowHint Visible Bront Used Haint Visible ParentShowHint Visible ParentShowHint Visible ParentShowHint Philot Philot ParentShowHint BlargeTicks (TimstrumentTicksFloat) Phargins ParentShowHint	□Appearance		
Behavior Visible ✓ Bront (TFont) ParentFont □ Phint □ Hint □ ParentShowHint ✓ ParentShowHint ✓ ShowHint □ Bisc □ HolpGontext □ Name xLEDsSlider0 Nome xLEDsSlider0 Popophone □ Tag 0 BOther □ AlignWithMargin □ BAnchors [akLeft,akTop] Discrete □ HandleSize isLarge HelpKeyword □ HelpKeyword □ HelpType htContext Horizontal □ BLargeTicks (TinstrumentTicksFloat) Imargins (Timargins) MaxValue 255	Cursor	crDefault	
Visible ✓ □Font (TFonk) □ParentFont □ □Hint □ Hint □ ParentShowHint ✓ ShowHint ✓ ShowHint ✓ BMisc □ HalpSontext S Name xLEDsSlider0 Popoprises □ Tag 0 BOther □ AlignWithMargin □ □Constraints [TSizeConstraints) Discrete □ HandleSize isLarge HelpKeyword HelpKeyword HelpKeyword □ Horizontal □ BLargeTicks (TinstrumentTicksFloat) MaxYalue ≥ SG	⊡Behavior		
EFont (TFont) ParentFont □ EHint □ Hint □ ParentShowHint ✓ ShowHint □ EMisc □ HalpContext ○ Name xLEDsSlider0 PopopHame □ Tag 0 BOther □ AlignWithMargin □ EConstraints [Aikeft,akTop] BConstraints [Size2constraints) Discrete □ HandleSize isLarge HelpKeyword □ HelpType htContext Horizontal □ BLargeTicks (TInstrumentTicksFloat) MarkValue 255	Visible		
⊞Font (TFont) ParentFont	⊡Font		
ParentFont EHint Hint Hint ParentShowHint ParentShowHint Misc Hintsc BalpContext Name xLEDsSlider0 Popaphisme Tag 0 EOther AlignWithMargin ElAnchors [akLeft,akTop] ElConstraints Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal ElargeTicks (TinstrumentTicksFloat) Ewargins MaxValue	⊞Font	(TFont)	53
Hint Hint ParentShowHint ✓ ShowHint ■Misc HolpContext Name xLEDsSlider0 Popophises Tag 0 BOther AlignWithMargin ■Anchors [akLeft,akTop] BConstraints (TsizeConstraints) Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal ■LargeTicks (TInstrumentTicksFloat) ■Margins (Imargins)	ParentFont		÷ +
Hint ParentShowHint ParentShowHint ShowHint ■Misc ■Misc HelpContext ShowHint ■Misc HelpContext Tag 0 ■Other AlignWithMargin ■Anchors [akLeft,akTop] BConstraints (TSizeConstraints) Discrete HelpKeyword HelpKeyword HelpType htContext Horizontal ■LargeTicks (TInstrumentTicksFloat) ■Margins (Timargins)	⊟Hint		
ParentShowHint ShowHint Imisc Imisc HelpSontext Name xLEDsSlider0 Popopriona Tag 0 Imisc Imisc AlignWithMargin Imisc AlignWithMargin Imisc Imisc AlignWithMargin Imisc Imisc Imisc AlignWithMargin Imisc	Hint		
ShowHint Imisc Holp Context Holp Context Holp Context Horizontal Constraints (TSizeConstraints) Discrete ImandeSize isLarge HelpKeyword HelpType htContext Horizontal Imargins (TinstrumentTicksFloat)	ParentShowHint		.Ŧ.
Holp Context Holp Context Name xLEDsSlider0 Popoprises Tag 0 BOther AlignWithMargin EAnchors [akLeft,akTop] EConstraints (TSizeConstraints) Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal ELargeTicks (TInstrumentTicksFloat)	ShowHint		
HelpContext Name XLEDsSlider0 Popoprisus Tag Tag O ElOther AlignWithMargin ElAnchors [akLeft,akTop] ElConstraints (TsizeConstraints) Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal ElLargeTicks (TInstrumentTicksFloat)	⊡Misc		
Name xLEDsSlider0 Popoprions Tag Tag O Other AlignWithMargin EAnchors [akLeft,akTop] EConstraints (TSizeConstraints) Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal ELargeTicks (TInstrumentTicksFloat) Emargins (Thargins) MaxValue	HelpContext		
Popophiona Tag 0 ElOther	Name	xLEDsSlider0	
Tag0ElOther	Popophona		
BOther AlignWithMargin AlignWithMargin [akLeft,akTop] BAnchors [akLeft,akTop] BConstraints (TSizeConstraints) Discrete	Tag	0	
AlignWithMargin ⊞Anchors [akLeft,akTop] [BConstraints (TSizeConstraints) Discrete ⊞Constraints (TSizeConstraints) Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal [LargeTicks (TInstrumentTicksFloat) [Emargins (TMargins) MaxValue 255	⊡Other		
Hanchors [akLeft,akTop] HandleSize (TSizeConstraints) Discrete Image: Starge HandleSize isLarge HelpKeyword Image: Starge HelpType htContext Horizontal Image: Starge HargeTicks (TInstrumentTicksFloat) Hargins (TMargins) MaxValue 255	AlignWithMargin		
	⊞Anchors	[akLeft,akTop]	
Discrete HandleSize isLarge HelpKeyword HelpType htContext Horizontal ⊞LargeTicks (TInstrumentTicksFloat) ™mangins (TMargins) MaxValue 255	⊞Constraints	(TSizeConstraints)	
HandleSize isLarge HelpKeyword	Discrete		
HelpKeyword HelpType htContext Horizontal Image: Context (TinstrumentTicksFloat) Image: Context (TinstrumentTicksFloat) Image: Context (TinstrumentTicksFloat) Image: Context (TinstrumentTicksFloat) Image: Context (TinstrumentTicksFloat) Image: Context (TinstrumentTicksFloat) Image: Context (TinstrumentTicksFloat)	HandleSize	isLarge	
HelpType htContext Horizontal ELargeTicks (TInstrumentTicksFloat) Maryalue 255	HelpKeyword		
Horizontal ElargeTicks (TInstrumentTicksFloat) Elimangins (TMangins) MaxValue 255	НеірТуре	htContext	
ELargeTicks (TInstrumentTicksFloat) Elemangins (Timargins) MaxValue 255	Horizontal		
MaxValue 255	⊞LargeTicks	(TInstrumentTicksFloat)	
MaxValue 255	Pittargins	(Ilmanginc)	
	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 micro 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 *m* icon on the form to launch the **Editing xSignals.SignalLinks** dialog seen in Figure 17.

Editing xSignalWrapper.SignalLinks	×
Design Signals	Virtual Signal Links
	<u> </u>
Generate virtual signals from phy 	rsical ones 1:1

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.



tual Signal Links
% - D
U11_LED0_7_0 U11_LED1_7_0 U11_LED2_7_0 U11_LED3_7_0 U11_LED4_7_0 U11_LED5_7_0 U11_LED6_7_0 U11_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.



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 C **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;
   LEDScanIndex := 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 solution 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.



			Live	Settinas
		NB3	000XN (ID	: 95876791)
TTO	Configure Fpga Project View Configuration	Scrolling_LEDs.PrjFpg		
No Proc	Instrument About			
T20 Spartan3AN X33514 Reset	K NUDAN-4FG676C			

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 autoconfigure 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.

				NB3000XN (ID: 95876791)
		NanoBoard-3	000XN V3.0.10		<u>100</u>
	🜒 🗊 Compile 🗸 🗸	Synthesize	🖌 🕒 Build	🔻 🌒 🗊 Program <u>F</u> PGA	#
	Scrolling_LEDs / NB3000XN_03	Spartan3AN XC3 Prog	ST1400AN-4FG676C rammed		
TDI	U11 Digital Runnir Digital TDI Ti	_Ю 19 10 20	Ν	U12 IANOBOARD_INTERFACE Running Run Application	

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

Software, hardware, documentation and related materials:

Copyright © 2009 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, Altium Designer, Board Insight, Design Explorer, DXP, LiveDesign, NanoBoard, NanoTalk, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.

