Vivado Hello World Tutorial

Embedded Processor Hardware Design September 9, 2013

Table of Contents

Part 1: Building a Zynq-7000 Processor Hardware3Introduction3Step 1: Start the Vivado IDE and Create a Project3Step 2: Create an IP Integrator Design5Customize Instantiated IP9Use Block Designer Assistance10Step 4: Generate HDL Design Files14Step 7: Implement Design and Generate Bitstream15Step 8: Export Hardware to SDK17Export to SDK17
Introduction3Step 1: Start the Vivado IDE and Create a Project3Step 2: Create an IP Integrator Design5Customize Instantiated IP9Use Block Designer Assistance10Step 4: Generate HDL Design Files14Step 7: Implement Design and Generate Bitstream15Step 8: Export Hardware to SDK17Export to SDK17
Step 1: Start the Vivado IDE and Create a Project3Step 2: Create an IP Integrator Design5Customize Instantiated IP9Use Block Designer Assistance10Step 4: Generate HDL Design Files14Step 7: Implement Design and Generate Bitstream15Step 8: Export Hardware to SDK17Export to SDK17
Step 2: Create an IP Integrator Design 5 Customize Instantiated IP 9 Use Block Designer Assistance 10 Step 4: Generate HDL Design Files 14 Step 7: Implement Design and Generate Bitstream 15 Step 8: Export Hardware to SDK 17 Export to SDK 17
Customize Instantiated IP9Use Block Designer Assistance10Step 4: Generate HDL Design Files14Step 7: Implement Design and Generate Bitstream15Step 8: Export Hardware to SDK17Export to SDK17
Use Block Designer Assistance 10 Step 4: Generate HDL Design Files 14 Step 7: Implement Design and Generate Bitstream 15 Step 8: Export Hardware to SDK 17 Export to SDK 17
Step 4: Generate HDL Design Files 14 Step 7: Implement Design and Generate Bitstream 15 Step 8: Export Hardware to SDK 17 Export to SDK 17
Step 7: Implement Design and Generate Bitstream 15 Step 8: Export Hardware to SDK 17 Export to SDK 17
Step 8: Export Hardware to SDK 17 Export to SDK 17
Export to SDK
Part 2: Build Zynq-7000 Processor Software18
Step 1: Start SDK and Create a Software Application
Step 2: Run the Software Application
Add a Breakpoint
Step 3: Executing the Software

Requirements

The following is needed in order to follow this tutorial:

- Vivado w/ Xilinx SDK (tested, version 2013.2)
- Zedboard (tested, version D)

Part 1: Building a Zynq-7000 Processor Hardware

Introduction

In this part of the tutorial you create a Zynq-7000 processor based design and instantiate IP in the processing logic fabric (PL) to complete your design. Then you take the design through implementation, generate a bitstream, and export the hardware to SDK.

If you are not familiar with the Vivado Integrated Development Environment Vivado (IDE), see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893).

Step 1: Start the Vivado IDE and Create a Project

1. Start the Vivado IDE (FIGURE 1) by clicking the Vivado desktop icon or by typing **vivado** at a terminal command line.



Figure 1: Getting Started Page

- 2. From the Getting Started page, select **Create New Project**. The New Project wizard opens (FIGURE 2).
- 3. Click Next

🚴 New Project	×
	Create a New Vivado Project
	This wizard will guide you through the creation of a new project
	To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.
	To continue, click Next.
	< Back Next > Finish Cancel

Figure 2: Create New Project Wizard

- 4. In the **Project Name** dialog box, type the project name and location. Ensure that **Create project subdirectory** is checked, and then click **Next**.
- 5. In the **Project Type** dialog box, select **RTL Project**, then click **Next**.
- 6. In the **Add Sources** dialog box, ensure that the **Target language** is set to **VHDL**, then click **Next**.
- 7. In the **Add Existing IP** dialog box, click **Next**.
- 8. In the **Add Constraints** dialog box, click **Next**.
- 9. In the **Default Part** dialog box select **Boards** and choose "ZedBoard Zynq Evaluation and Development Kit". Make sure that you have selected the proper Board Version to match your hardware because multiple versions of hardware are supported in the Vivado IDE. Click **Next**.
- 10. Review the project summary in the **New Project Summary** dialog box before clicking **Finish** to create the project.

Step 2: Create an IP Integrator Design

1. In the Flow Navigator, select Create Block Design.



Figure 3: Create Block Design from Flow Navigator

2. In the **Create Block Design** popup menu, specify a name for your IP subsystem design.



Figure 4: Create Block Design Dialog Box

3. Right-click in the Vivado IP integrator diagram window, and select Add IP.



4. Alternatively, you can click the **Add IP** link in the IP integrator diagram area.



Figure 6: Add IP Link in IP Integrator Canvas

The IP Catalog opens.

5. In the search field, type **zynq** to find the ZYNQ7 Processing System IP, and then press **Enter** on the keyboard.

Search: Q- zynq (2 matches)										
Name 1	Version 2	AXI4	Status	License	Vendor	1				
ZYNQ7 Processing System	5.2	AXI4-Stream, AXI4	Production	Included	Xilinx, Inc.	x				
ZYNQ7 Processing System BFM	1.0	AXI4	Pre-produ	Purchase	Xilinx, Inc.	x				
Select and press ENTER or drag and drop, ESC	to cancel									

Figure 7: The IP Integrator IP Catalog

Because you selected the ZedBoard when you created the project, the Vivado IP integrator configures the design appropriately.

In the Tcl Console, you see the following message:

```
create_bd_cell -type ip -vlnv
xilinx.com:ip:processing_system7:5.2 processing_system7_1
INFO: [PS7-6] Configuring Board Preset zed. Please wait .....
```

There is a corresponding Tcl command for all actions performed in the IP integrator block diagram. Those commands are not shown in this document. See the Tcl Console for information on those commands.

6. In the IP integrator diagram header, click **Run Block Automation**.

🖫 Diagram 🗙 🔣 Address Editor 🗙	
注 📥 zynq_design_1	
🔍 🗔 Designer Assistance available. Run Block Automation	
/processing_system7_1	

Figure 8: Run Block Automation on Zync

The **Run Block Automation** dialog box opens, stating that the FIXED_IO and DDR interfaces will be created for the Zynq core.

7. Click OK.



Figure 9: Zync7 Run Block Automation Dialog Box

After running block automation on the Zynq processor, the IP integrator diagram should look as follows:



- 8. Now you can add peripherals to the processing logic (PL). To do this, rightclick in the IP integrator diagram, and select **Add IP**.
- 9. In the search field, type gpi to find the AXI GPIO IP, and then press **Enter** to add the AXI GPIO IP to the design.
- 10. Repeat the action, typing axi bram to find and add AXI BRAM Controller, and typing block to find and add Block Memory Generator.

The Block Design window matches FIGURE 11. The relative positions of the IP will vary.



Figure 11: Block Design after Instantiating IP

Customize Instantiated IP

1. Double-click the Block Memory Generator IP, or right-click and select **Customize Block** (FIGURE 12).

	gen	7	
Block Memory	9	Block Properties	Ctrl+E
axi gpi	×	Delete	Delete
 	Ð	Сору	Ctrl+C
s_axi_adk	1	Paste	Ctrl+V
s_axi_aresetn	R.	Select All	Ctrl+A
AXI GP	9	Add IP	Ctrl+I
	<u>∎</u> X	Customize Block	
		Orientation	•

Figure 12: Customize Block Option

The **Re-customize IP** dialog box opens. 2.

- 2. On the Basic tab of the dialog box, set:
 - Mode to BRAM Controller
 - Memory Type to True Dual Port RAM Click OK.

🖵 Re-customize IP		
Block Memory Generator (8.0)		
💕 Documentation 🚞 IP Location		
IP Symbol Power Estimation Show disabled ports	Component Name zynq_design_1_blk_mem_gen_1_0 Basic Port A Ontione Port B Options Other Options Summary	
r S	Mode BRAM Controller BRAM Controller Generate address interface with 32 bits Memory Type True Dual Port RAM Common Clock FCC Options	
	ECC Type No ECC Error Injection Pins Single Bit Error Injection	

Figure 13: Set Mode and Memory Type

The AXI BRAM Controller provides an AXI memory map interface to the Block Memory Generator.

3. Connect the Block Memory Generator to the AXI4 BRAM Controller by clicking the connection point and dragging a line between the IP.



The AXI BRAM Controller provides an AXI memory map interface to the Block Memory Generator.

Use Block Designer Assistance

Block Designer Assistance helps connect the AXI GPIO and AXI BRAM Controller to the Zynq-7000 PS.

1. Click **Run Connection Automation** and then select **/axi_gpio_1/s_axi** to connect the BRAM controller and GPIO IP to the Zynq PS and to the external pins on the ZedBoard (FIGURE 15).



Figure 15: Run Connection Automation

The Run Connection Automation dialog box opens and states that it will connect the master AXI interface to a slave interface.

In this case, the master is the Zynq Processing System IP (FIGURE 16).



Figure 16: Run Connection Automation Message

Click OK.

This action instantiates an AXI Interconnect IP as well as a Proc Sys Reset IP and makes the interconnection between the AXI interface of the GPIO and the Zynq-7000 PS.

2. Select Run Connection Automation again, and the /axi_gpio_1/gpio shown in FIGURE 17.

ŀ	Diagram × 🛚 🔀 Address Editor ×		
₹	🚠 zynq_design_1 →		
4	G Designer Assistance available. Run Connection Automation	1	
Q-		Axi_gpio_1/gpio	
٩		/axi_bram_ctrl_1/S_AXI	

Figure 17: axi_gpio Selection

The Run Connection Automation dialog box includes options to hook up to the GPIO port. 4.

3. Select leds_8bits (FIGURE 18).



Figure 18: Select Board Interface Options

- 4. Click **OK**. This step also configures the IP so that during netlist generation, the IP creates the necessary Xilinx Design Constraints (XDC).
- 5. Click **Run Connection Automation** again, and select the remaining option /axi_bram_ctrl_1/S_AXI (FIGURE 19).



Figure 19: axi_bram_ctrl Selection

This completes the connection between the Zynq7 Processing System and the AXI BRAM Controller.

The IP integrator subsystem looks like FIGURE 20. Again, the relative positions of the IP can differ slightly.



6. Click the Address Editor tab to show the memory map of all the IP in the design.

In this case, there are two IP: the AXI GPIO and the AXI BRAM Controller. The IP integrator assigns the memory maps for these IP automatically. You can change them if necessary.

7. Change the range of the AXI BRAM Controller to **64K**, as shown in FIGURE 21.

20	Diagram 🗙 🔣 Address Editor 🗙				
٩,	Instance	Base Name	Offset Address	Range	High Address
X	# /processing_system7_1				
	🖻 🏥 Data				
-	/axi_gpio_1	Reg	0x41200000	64K	0x4120FFFF
	🗖 🚥 /axi_bram_ctrl_1	Mem0	0x40000000	4K 🔻	0x40000FFF
				4K 🔺	
				8K _	
				16K =	
				32K	
				64K	
				128K	
				256K 🔻	
				512K	
				512K //	

Figure 21: axi_bram_ctrl to 64k Range

- 8. Save your design by pressing Ctrl-S, or select File > Save Block Design.
- 9. Click the Address Editor tab to make sure that the memory mappings for the GPIO and BRAM controller have been auto populated.
- 10. From the toolbar, run Design-Rules-Check (DRC) by clicking the **Validate Design** button (FIGURE 22). Alternatively, you can do the same from the menu by:
 - Selecting **Tools** > **Validate Design** from the menu.
 - Right-clicking in the Diagram window and selecting **Validate Design**.



Figure 22: Validate Design Button

The Validate Design Successful dialog box opens (FIGURE 23).



Figure 23: Validate Design Message

11. Click **OK**.

Step 4: Generate HDL Design Files

You now generate the HDL files for the design.

1. In the Source window, right-click the top-level subsystem design and select **Generate Output Products** (FIGURE 24). This generates the source files for the IP used in the block diagram and the relevant constraints file.



Figure 24: Generate Output Products Option

2. The Manage Output Products dialog box opens. Click OK.



- 3. In the Sources window, select the top-level subsystem source, and select **Create HDL Wrapper** to create an example top-level HDL file (FIGURE 25).
- 4. Click **OK** when the **Create HDL Wrapper** dialog box opens.

Block Design - zynq_	desi	gn_1						
Sources	皆 Diagram	× 🔣 Addre						
🔍 🔀 😂 🔂 🛙		Cell						
🖃 🛜 Design Sources	🕞 🔂 Design Sources (1)							
📗 🖃 🎄 zynq_design	1 (zvna desian 1.hd)(1))ata			
🗄 😰 zynq_des	6	Source Node Properties		Ctrl+E	axi_gpio_1			
⊕ Constraints (1) ⊕ Constraints (1) ⊕ Constraints (1)	•	Open File		Alt+O	axi_bram_o			
		Create HDL Wrapper		N				
		View Instantiation Template		13				

Figure 25: Create HDL Wrapper

Step 7: Implement Design and Generate Bitstream

1. In Flow Navigator, click **Generate Bitstream** to implement the design and generate a BIT file.

Note: If the system requests to re-synthesize the design before implementing, click **No**. The previous step of saving the constraints caused the flow to mark synthesis out-of-date. Ordinarily, you might want to re-synthesize the design if you manually changed the constraints, but for this tutorial, it is safe to ignore this condition (FIGURE 26).



Figure 26: Generate Bitstream

You might see a dialog box stating no implementation results are available.

2. Click Yes.



Figure 27: No Implementation Results Available Dialog Box

3. After the design implementation, click **Open Implemented Design**, (FIGURE 28).



Figure 28: Bitstream Generation Completed

4. You might get a warning that the implementation is out of date. Click **Yes**.



Figure 29: Implementation Is Out-of-Date Dialog Box

Step 8: Export Hardware to SDK

In this step, you export the hardware description to SDK. You use this in Part 2. The IP integrator block diagram, and the Implemented design, must be open to export the design to SDK.

IMPORTANT: For the Digilent driver to install, you must power on and connect the board to the host PC before launching SDK.

Export to SDK

1. In the Flow Navigator, click **Open Block** to invoke the IP integrator design (FIGURE 30).



Figure 30: IP Integrator - Open Block Design

Now you are ready to export your design to SDK.

2. From the main Vivado File menu, select Export Hardware for SDK (FIGURE 31).

8	Add Sources Open Source File	Alt+A Ctrl+N		Type: Size:	Block Designs	
	Export	•		Export Hardware	for SDK	53 c/
	Open Log File			Export Block Desi	ign	5/
	Open Journal File		61	Export Bitstream	File	

Figure 31: Export Hardware for SDK

The Export Hardware for SDK dialog box opens, ensure that Export Hardware, Include Bitstream, and Launch SDK are checked (FIGURE 32).

1	Lexport Hardware for SDK										
	Export hardware platform for SDK.										
	Options										
	Source:	Å zynq_design_1.bd	Ŧ								
	Export to: 🔂 <local project="" to=""></local>										
	Workspace: 💿 <local project="" to=""></local>										
	🔽 Export Ha	ardware									
	🔽 Include b	itstream (Note: an implemented design n	n								
	Launch SDK										
		ОК Са	ncel								

Figure 32: Export Hardware for SDK

Part 2: Build Zynq-7000 Processor Software

In this portion of the tutorial you will build an embedded software project that prints "Hello World" to the serial port.

Step 1: Start SDK and Create a Software Application

- If you are doing this lab as a continuation of Part 1 then SDK should have launched in a separate window (if you checked the Launch SDK option while exporting hardware). You can also start SDK from the Windows Start menu by clicking on Start > All Programs > Xilinx Design Tools > Vivado 2013.2 > SDK > Xilinx SDK 2013.2. When starting SDK in this manner you need to ensure that you in the correct workspace.
- You can do that by clicking on File > Switch Workspace > Other in SDK. In the Workspace Launcher dialog box in the Workspace field, point to the SDK_Export folder where you had exported your hardware from lab 1. Usually, this is located at

..\project_name\project_name.sdk\SDK\SDK_Export.

Now you can create a hello world application.

3. Select **File > New > Application Project** (FIGURE 33).

•	C/C++	- hw_pla	tform_0/sy	stem.xml -	Xilinx SD	К							
File	Edit	Source	Refactor	Navigate	Search	Run	Pro	oject	Xilinx Too	ols \	Window	Help	
	New				Alt+Shift	+N ►	2	Mak	efile Projec	ct wit	h Existin	g Code	
	Open	File					2	C++	Project				
	Close	Class Chile		Ctrl IN	Ċ	C Project							
	Close				Curr		4	App	lication 🖓	oject			
	Figure 33: File->New->Application Project												

New Project dialog box opens

4. In the Project Name field, type **Zync_Design**, and click **Next** (FIGURE 34).

🐵 New Project								
Application Project								
Project name: Zynq_D	lesign							
Use default location	n							
Location: C:\tutorials\;	zynq_debug_design\zynq_debug_design.sdk\SDK\SDK_E							
Choose file sy	ystem: default 💌							
Target Hardware								
Hardware Platform h	Hardware Platform hw_platform_0							
Processor	vs7_cortexa9_0							
Target Software								
OS Platform	standalone							
Language								
Board Support Packag	ge Create New Zynq_Design_bsp							
	⊙ Use existing							
(?)	< Back Next > Finish Cancel							
•								

Figure 34: SDK Application Project

5. From the Available Templates, select **Hello World** (FIGURE 34) and click **Finish**.

New Project	
Templates Create one of the available templates to gene application project.	rate a fully-functioning
Available Templates: Dhrystone Empty Application Hello World WIP Echo Server Memory Tests Peripheral Tests Zynq FSBL	Let's say 'Hello World' in C.
	ext > Finish Cancel

Figure 35: SDK New Project Template

When the program finish compiling, you will see the following (FIGURE 36).



Step 2: Run the Software Application

Now, you must run the hello world application on the ZedBoard. To do so, you need to configure the JTAG port. Make sure that your hardware is powered on and a Digilent Cable is connected to the host PC. Also, ensure that you have a USB cable connected to the UART port of the ZedBoard.

1. Click Xilinx Tools and select Configure JTAG Settings (FIGURE 37).



Figure 37: Configure JTAG Settings

2. In the Configure JTAG Settings dialog box, select the **Type** as **Auto Detect**, and click **OK** (FIGURE 38).

🐵 Configu	ure JTAG Settings			×					
Configu	re JTAG Settings								
Specify t These se	he JTAG cable to use fo ttings affect how XMD	or communication and JTAG connects to the FPGA.	Device Chain configuration of th	e target board. 🖣 🕂					
JTAG Cab	ole								
	Type: Auto Detect								
Hostr	stname:								
	Port:								
Frequ	lency:	•							
Other On	tions:								
Other Op	uons:								
JTAG Dev	vice Chain								
 Autom 	natically Discover Device	es on JTAG Chain							
Imanua (al Configuration of JTA	3 Chain		ឧភ្សុង					
EDC A2	Davias Nama	ID Cada	ID I anoth	2 @ V X					
FPGA	Device Name	ID Code	ik Length						
?			ОК	Cancel					

Figure 38: Configure JTAG Settings

3. Next, download the bitstream into the FPGA by selecting Xilinx **Tools** > **Program FPGA** (FIGURE 39).



This opens the Program FPGA dialog box.

4. Ensure that the path to the bitstream that you created in Step 7 of Lab 1 is correct and then click **Program**.

Note: The DONE LED on the board turns blue if the programming is successful.

- 5. Select and right-click the **Zynq_Design** application.
- 6. Select **Debug As** and **Debug Configurations** (FIGURE 40).

🐵 C/C	2++ - :	zynq	_design_bsp/	system.mss	- Xilinx S	DK								
File E	dit S	Sourc	e Refactor	Navigate	Search	Run	Project	Xilin	x Tools	Window	/ Help			
-		à 🖻	🛛 👻 🕶 🚳 -	- 🗟 🖆	- 🚳 -	c -	ଙ ◄	参 ▼	0 -	% - 1		2	× 6	😂 🛷 👻
Proj	ject Ex	kplor	er 🛛	6	system.x	ml	👔 syste	m.mss	×					
			E 🕏	\checkmark		p37	/_unry /afi2.o	eneric						
4 🕼	hw_pl	atfor	m_0			057	/afi3 g	eneric						
		_init	.с			ps7	_can_0 ca	anps	Docur	mentation	Example	es		
	🖹 ps		New				×	neric				_		
	🥹 ps		Go Into					neric						
	ps Open in New Windo							vcfg	<u>Docu</u>	mentation	Example	<u>es</u>		
	sy sy	P	Copy				Ctrl+C	naps	<u>Docu</u>	mentation	Example	<u>es</u>		
4 😂	Zynq		Paste				Ctrl+V	naps	<u>Docu</u>	mentation	Example	<u>es</u>		
Þ	🖧 Bi	×	Delete			1	Delete	nacps	<u>Docu</u>	mentation	Example	<u>es</u>		
Þ	றி In		Source				+	iops	<u>Docu</u>	mentation	Example	<u>es</u>		
Þ (🔁 D(Move					ps	<u>Docu</u>	mentation	Example	<u>es</u>		
Þ	🔁 sre		Rename				F2	neric						
A 📠 :	✓ [™] zynq → i BS		Import					pips	<u>Docu</u>	mentation	Example	<u>es</u>		
			Export					neric						
			Build Project					neric						
	🗎 lit		Clean Project	:				aic	Docu	mentation	Example	es		
[Ъ М	8	Refresh				F5	itime	Docur	mentation	Example	es		
	🔥 sy		Close Project					Jwdt	Docu	mentation	Example	es		
			Close Unrelat	ted Projects				neric						
			Build Configu	urations			×	neric						
			Make Targets	5			+	rtps	Docur	mentation	Example	<u>es</u>		
			Index				+	ops	<u>Docu</u>	mentation	Example	<u>es</u>		
			Show in Rem	ote System	s view			I						
			Convert To					-	D					
			Run As			×	раго Support Package.							
			Debug As				+	GDB 1	Launcl	h on Hard	ware (GD	B)		
			Profile As				+	fer 2	Launc	n on Hard	ware (Sys	tem D	ebugger)
			Team				+	S 4	Remo	te ARM Lii	plication	ication		
			Compare Wit	th			+	** T	Neline (Canfin Li	ian Appli	cation		
		***	Restore from	Local Histo	ory ic				ebug (Configurat	lions			
		1	Null C/C++ C	oue Analys	15			cess	ing c	ommand	line op	tion	-hwspe	ec C:/tut

Figure 40: Launch on Hardware

 In the Debug Configurations dialog box, right-click Xilinx C/C++ Application (GDB) and select New.



Figure 41: Debug Configuration Dialog Box

8. In the Debug Configurations dialog box, click Debug.

Bebug Configurations		×
Create, manage, and run configurations		\$\$.
Yes >> Type filter text € (C/c++ Application E (C/c++ Attach to Application € (C/c++ Ostmotem Debugger) C (C/c++ Ronet& Application ► Launch Group K Rennet A&M Linux Application Target Communication (GDB) ▲ Minux (C++ application (GDB) K Minux (C/c++ application (System Debugger)	Name: Zynq_Design Debug Main by Source (Device Initialization or STDIO Connection Remote Debug Endet Connect to gdbserver on a different machine. Endet Remote GDB Server, launch XMD on the remote machine, and connect to the processor. XMD will then report the port at which the gdbserver is open. IP Addres: Iocalhost Port 1234	Debugger Options 🖾 Common 🗋
Filter matched 10 of 10 items]	Apply Revert
0		Debug Close

Figure 42: Run Debug Configurations

9. The Confirm Perspective Switch dialog box opens. Click Yes.



Figure 43: Confirm Perspective Switch Dialog Box

10. Set the terminal by selecting the Terminal 1 tab and clicking the Settings button (FIGURE 44).



Figure 44: Settings Button

11. Use the following settings for the ZedBoard (FIGURE 45). Click **OK**.

Connection Type:	
Serial 🔹	The Port should be
Settings:	port for the Cypres
Port: COM1 -	USB-to-Serial.
Baud Rate: 115200 🔻	
Data Bits: 8	
Stop Bits: 1	
Parity: None 🔻	
Flow Control: None	
Timeout (sec): 5	

12. Verify the **Terminal** connection by checking the status at the top of the tab (FIGURE 46).

Problems	🖉 Tasks	🖳 Console	Properties	🖉 Terminal 1 🛛
Serial: (COM1	, 115200,	8, 1, None, N	None - CONNE	CTED) - Encoding: (ISO-8859-1)

Figure 46: Terminal Connection Verification

13. In the **Debug** tab, expand the tree, and select the processor core on which the program is to be run (FIGURE 47).



Figure 47: Processor Core to Debug

14. If it is not already open, select .../src/helloworld.c, line 41, and double click that line to open the source file.

Add a Breakpoint

You add a breakpoint on line 43.

1. Select **Navigate > Go To Line** (FIGURE 48).



Figure 48: Go to Line

- 2. In the Go To Line dialog box, type **43**.
- 3. Double click on the left pane of line 43, which adds a breakpoint on that line of source code (Figure 49).



Figure 49: Add a Breakpoint

Step 3: Executing the Software

This step will take you through executing the code up to and past the break point.

- 1. Click the Resume button or press F8
- 2. Click the Step Over button or press F6
- 3. You should see "Hello World" in the terminal if everything worked correctly (FIGURE 50).



Figure 50: Terminal Output