

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

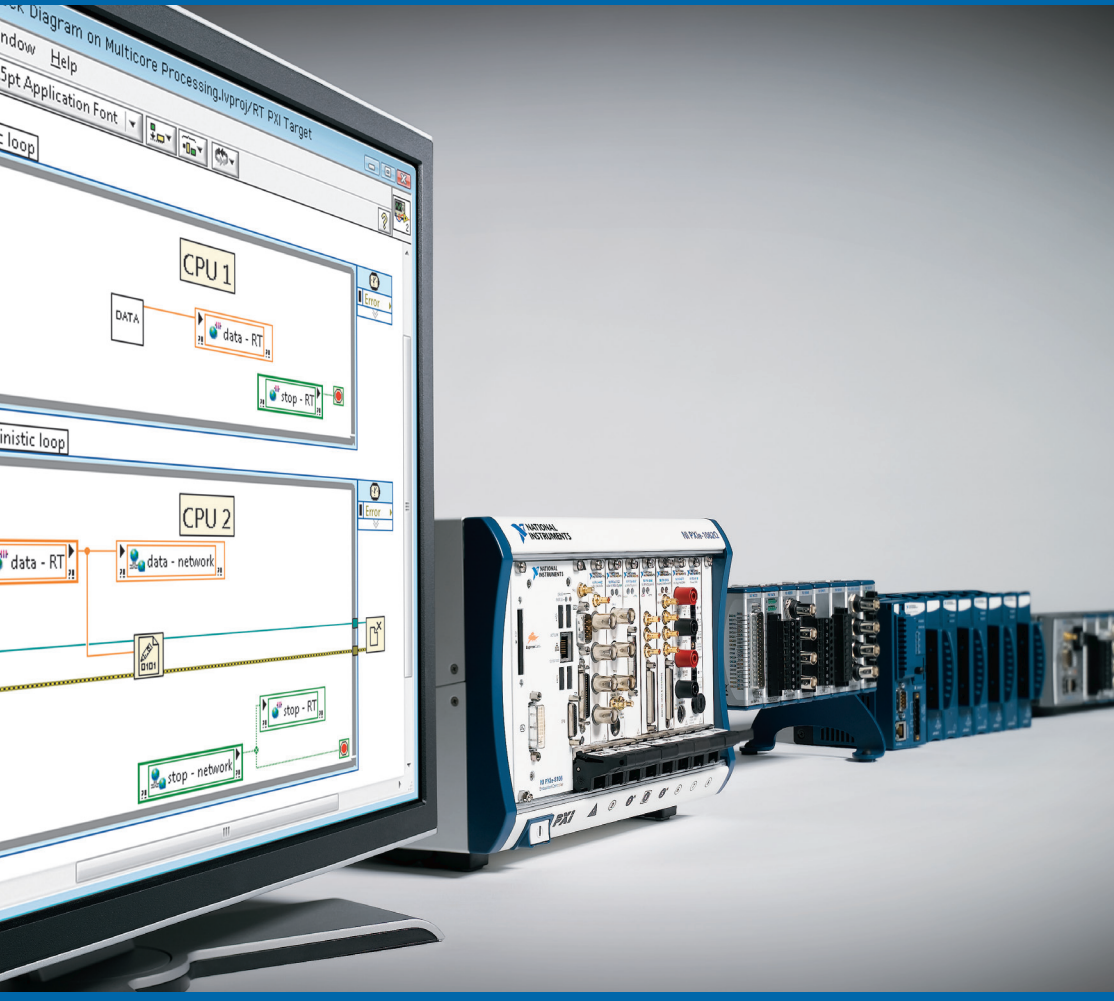
- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

NI PXIe-5641R
Getting Started Guide



ni.com/rf

NATIONAL INSTRUMENTS

GETTING STARTED GUIDE

NI PXIe-5641R

RIO IF Transceiver

This document explains how to install, configure, and program an NI PXIe-5641R IF transceiver.

The NI 5641R offers two 100 MS/s, 14-bit input channels with built-in digital downconversion and two 200 MS/s, 14-bit output channels with built-in digital upconversion. The NI 5641R is supported by the NI-5640R software that is included with your device.

Contents

Verifying the System Requirements....2
Unpacking the Kit....2
Verifying the Kit Contents....2
Installing the Software....3
- Choosing an Application Development Environment (ADE)....3
- Installing LabVIEW and NI-5640R Software....5

Installing the Hardware....6
Configuring the NI 5641R in MAX....7
Connecting Signals....8
- Connecting Analog Input (AI) Signals....8
- Connecting Analog Output (AO) Signals....8
- Connecting Digital Input/Output (DIO) Signals....9

Programming the NI 5641R....9
- NI-5640R Instrument Driver Examples....10

Troubleshooting....10
- The ni5640R Check Thermal Shutdown VI Indicated an Overtemperature Condition and My Device Shut Down. What Should I Do Next?....10
- My Signal Looks Very Noisy....11
- It Takes a Long Time to Access Controls and Indicators Using the Read/Write Control Function in LabVIEW FPGA....12
- An Error Was Detected in the Communication between the Host Computer and the FPGA Target....13
- When I Open a Host VI, LabVIEW Cannot Find a File Named filename.lvbit or filename.lvbitx....14
- When I Run an FPGA VI, a Compilation Error Instantly Occurs....14
- I Received a “Clock domain crossing not supported for FIFO memory” Error....14
- Connecting Two Asynchronous Data Wires Produces a Broken Wire....15
- Connecting Two Asynchronous Timing Wires Produces a Broken Wire....15
- I Lost My Harness Node Settings....15

NATIONAL INSTRUMENTS

Appendix A: NI PXIe-5641R Front Panels....16
Appendix B: Creating an IF Transceiver Application in LabVIEW 18
Creating an Application Using LabVIEW FPGA....19
Creating an Application Using the NI-5640R Instrument Driver....34
Where Do I Go Next?....34
Worldwide Support and Services....34

Verifying the System Requirements

To use the NI-5640R instrument driver, your system must meet certain requirements.

For more information about minimum system requirements, recommended system, and supported application development environments (ADEs), refer to the product readme, which is available on the driver software DVD or online at *ni.com/updates*.

Unpacking the Kit

Caution To prevent electrostatic discharge from damaging the device, ground yourself using a grounding strap or by holding a grounded object, such as your computer chassis.

1. Touch the antistatic package to a metal part of the computer chassis.
2. Remove the device from the package and inspect the device for loose components or any other sign of damage.

   Caution Never touch the exposed pins of connectors.

   Notify NI if the device appears damaged in any way. Do not install a damaged device.
3. Unpack any other items and documentation from the kit.

Store the device in the antistatic package when the device is not in use.

Verifying the Kit Contents

Verify that the kit contains the following items:

- NI-5640R software DVD, which includes the *NI IF Transceivers Help*
- *NI 5641R Getting Started Guide* (this document)
- NI 5641R
- Other documentation included with the NI 5641R and driver software:
  - *NI PXIe-5641R Specifications*
  - *Read Me First: Safety and Electromagnetic Compatibility*
  - *Maintain Forced-Air Cooling Note to Users*

Installing the Software

Choosing an Application Development Environment (ADE)

You create applications for your NI 5641R using LabVIEW.

You can use the LabVIEW FPGA Module to program your device, or you can use the NI-5640R LabVIEW instrument driver. More information about each method is provided in the following sections.

LabVIEW FPGA Module

Using the LabVIEW FPGA module, you can program the NI 5641R FPGA to match the requirements of your system.

Using LabVIEW FPGA, you can create user-defined behavior for the NI 5641R using a virtual instrument (VI), thereby creating an application-specific I/O device. However, using LabVIEW FPGA requires more programming time and more advanced programming skills than using the NI-5640R instrument driver.

When using the LabVIEW FPGA module, you have two methods of developing code that runs on your FPGA, which are the traditional LabVIEW FPGA programming or the NI-5640R Asynchronous Programming Palette.

Related Information

For more information about the NI LabVIEW FPGA Module, navigate to http://www.ni.com/fpga/

Traditional LabVIEW FPGA Programming

Traditional LabVIEW FPGA programming uses a dataflow model for running VIs and other nodes.

Nodes execute after they receive all required inputs. When a node executes, it generates data and passes the data to the next node in the dataflow path. The movement of data through the nodes determines the execution order of the VIs and functions on the block diagram.

NI-5640R Asynchronous Programming Palette

You can use the NI-5640R Asynchronous Programming palette, an additional palette of LabVIEW FPGA nodes, to simplify creating LabVIEW FPGA applications for your NI 5641R.

Using the asynchronous programming model provides free-running actors that can execute independently of the dataflow dependencies created by traditional LabVIEW programming.

The nodes and wires used in this model handle some aspects of programming in LabVIEW FPGA for you.

- Required clocks, data buffers, and other I/O items are automatically added to the LabVIEW FPGA project.
- The A/D converter (ADC) and D/A converter (DAC) nodes automatically choose the correct clock domain.
- The ADC and DAC nodes account for the way the device ADCs and DACs pack and unpack I/Q data.
- Common data exchange techniques are abstracted to simplify how data buffers are created and used.

NI-5640R Instrument Driver

The NI-5640R instrument driver API features a set of operations that exercise the basic functionality of the device, including configuration, control, and other device-specific functions.

With the NI-5640R API, you program the NI 5641R with its default personality—two synchronized input and two synchronized output channels.

Related Information

For more information about programming with the NI-5640R instrument driver, refer to the NI IF Transceiver Help. This help file contains hardware information, concepts, a detailed VI reference for the NI-5640R instrument driver API, and information specific to your device.

Programming Methods Comparison

Table 1. NI 5641R Programming Methods Comparison

Feature	LabVIEW FPGA Module		NI-5640R LabVIEW Instrument Driver
	Traditional LabVIEW FPGA	NI-5640R Asynchronous Programming	
Programming Complexity	Advanced LabVIEW and LabVIEW FPGA programming skills required. Using the LabVIEW FPGA Module allows you to create programs that exercise the maximum capabilities of the device.		Easy-to-use application programming interface (API).
Input/Output Accessibility	User-defined I/O.		Two synchronous input and two synchronous output channels.
Triggering Capabilities	Ability to create custom signal processing and custom triggering in the FPGA.		Support for software and digital edge triggering using the NI-5640R API.

Table 1. NI 5641R Programming Methods Comparison (Continued)

Feature	LabVIEW FPGA Module		NI-5640R LabVIEW Instrument Driver
	Traditional LabVIEW FPGA	NI-5640R Asynchronous Programming	
Compilation Cycles	Required FPGA compilation cycles.		No FPGA compilation cycles.
Programming Paradigm	FPGA code: Has only dataflow dependencies.	FPGA code: Has free-running actors that can execute independently of dataflow dependencies.	Host-based API only: No user-defined FPGA code.
Data Movement Policy	FPGA code: Uses only dataflow wires and user-created FIFOs.	FPGA code: Asynchronous data wires pass data between nodes independently of dataflow dependencies.	Host-based API only: No user-defined FPGA code.
Clock Configuration Policy	FPGA code: Uses only dataflow wires.	FPGA code: Asynchronous timing wires pass clock information between nodes independently of dataflow dependencies.	Host-based API only: No user-defined FPGA code

Installing LabVIEW and NI-5640R Software

You must be an Administrator to install NI software on your computer.

Select your programming method before you install your ADE.

1. Install LabVIEW.
2. If you choose to use the LabVIEW FPGA programming method, install the LabVIEW FPGA Module.
3. Insert the driver software media into your computer. The installer should open automatically.

   If the installation window does not appear, navigate to the drive, double-click it, and double-click `autorun.exe`.
4. Follow the instructions in the installation prompts.

   Note Windows users may see access and security messages during installation. Accept the prompts to complete the installation.
5. When the installer completes, select **Restart** in the dialog box that asks if you want to restart, shut down, or restart later.

Installing the Hardware

You must install the software before installing the hardware.
Before you install the hardware, refer to the guidelines in the *Maintain Forced-Air Cooling Note to Users* included with the module to ensure that the device can cool itself effectively. This document is also available at *ni.com/manuals*.

Caution NI 5641R modules are sensitive instruments that should be handled carefully. To prevent damage to the device caused by ESD or contamination, handle the device using the edges or the metal bracket.

Table 2. PXI/PXI Express Compatibility Glyphs

	You can install PXI or PXI Express modules in any PXI hybrid slot marked with a peripheral slot compatibility glyph (the letter "H" and a circle with a solid black background containing the slot number).
	You can also install PXI Express modules in any PXI Express slot marked with a peripheral slot compatibility glyph (a circle with a solid black background containing the slot number).

The NI 5641R is a one-slot module. It can be installed in a PXI Express-compatible slot.

1. Connect the AC power source to the chassis before installing the system controller or modules. The AC power cord grounds the chassis and protects it from electrical damage while you install the system controller.
2. Power off the chassis.
3. If the chassis has multiple fan speed settings, ensure the fans are set to the highest setting.

   Note Inadequate air circulation could cause the temperature inside the chassis to rise above the optimal operating temperature for the device, potentially causing thermal shutdown, shorter lifespans, or improper performance.
4. Position the chassis so that inlet and outlet vents are not obstructed. For more information about optimal chassis positioning, refer to the chassis documentation.
5. Remove the black plastic connectors from all the captive screws on the module front panel.
6. Ensure the ejector handle is in the unlatched (downward) position.
7. Hold the module by the ejector handle and slide it into an empty compatible slot. Ensure the base engages with the guides in the chassis.
8. After sliding the module completely into the chassis, latch it by pulling up on the ejector handle.

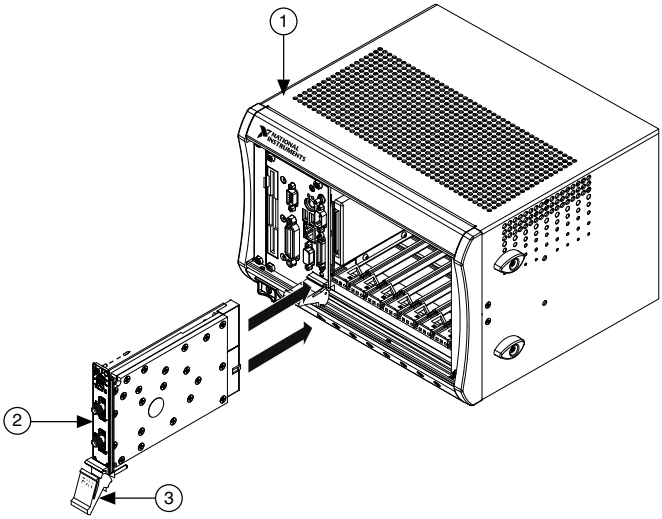
9. Tighten the captive screws at the top and bottom of the module front panel.

Note Device performance may suffer if screws are not tightened properly.

10. Verify that the chassis fans are operable and free of dust and other contaminants that may restrict airflow.
11. Cover all empty slots using filler panels and slot blockers before operating the module.

    NI recommends that you install slot blockers between the device and adjacent hardware modules to maximize cooling air flow.
12. Power on the chassis.

Figure 1. Installing the Module



1. PXI Hybrid Chassis
2. PXI Module
3. Ejector Handle in Down Position

Configuring the NI 5641R in MAX

Use Measurement & Automation Explorer (MAX) to configure your National Instruments hardware. MAX informs other programs about which devices reside in the system and how they are configured. MAX is automatically installed with NI-5640R.

1. Launch Measurement & Automation Explorer (MAX).

   MAX should automatically detect the device you installed.
2. In the Configuration pane, double-click **Devices and Interfaces** to see the list of installed devices. Installed devices appear under the name of their associated chassis.
3. Expand **NI-RIO Devices**.

4. Check that NI 5641R appears under NI-RIO Devices

   Note If you do not see your hardware listed, press <F5> to refresh the list of installed devices. If the device is still not listed, power off the system, ensure the device is correctly installed, and restart.

5. Click the **Save** button.
6. Record the device identifier MAX assigns to the hardware. Use this identifier when programming the NI 5641R.

Connecting Signals

You can connect three kinds of signals to the NI 5641R: analog input (AI) signals, analog output AO) signals, and digital input/output (DIO) signals.

Related Information

Appendix A: NI PXIe-5641R Front Panels on page 16

Connecting Analog Input (AI) Signals

- Use a connector saver (a replaceable SMA adapter used on test equipment) to connect AI signals to the NI 5641R.
- Ensure that analog signals do not exceed the maximum input voltage ratings specified in the specifications document to avoid damage to the device.
- Use external lowpass or bandpass filters when necessary to avoid aliasing effects.

Caution Observe the maximum input thresholds specified in the specifications document for your device. NI is not liable for any damage resulting from such signal connections.

Related Information

Refer to the Alias Effects topic and the Digital Downconverter (DDC) Operation topic for your device in the NI IF Transceivers Help for more information about aliasing and undersampling.

Connecting Analog Output (AO) Signals

- Use a connector saver (a replaceable SMA adapter used on test equipment) to connect AO signals to the NI 5641R.
- Terminate AO signals in 50 Ω impedance for best performance.
- Use external lowpass or bandpass filters when necessary to avoid imaging effects. For DAC update rates of 200 MS/s, images should be less than -50 dBc without any external lowpass filters.

Related Information

Refer to the Digital Upconversion (DUC) Operation topics for your device in the NI IF Transceivers Help for more information about imaging.

Connecting Digital Input/Output (DIO) Signals

The NI 5641R front panel DIO connector has nine pins. The DIO lines are direction-configurable by pin as input or output. If you are using LabVIEW FPGA, you can write an application to customize the functionality of this connector for your application.

Caution Exceeding the maximum input voltage ratings, which are listed in the device specifications, can damage the NI 5641R and the host computer or chassis. NI is not liable for any damage resulting from such signal connections.

- Connect signals to the DIO (AUX) connector.
- If required by your application, you can connect multiple NI 5641R digital output lines in parallel to provide higher current sourcing or sinking capability.

  If you connect multiple digital output lines in parallel, your application must drive all these lines simultaneously to the same value. If you connect digital lines together and drive them to different values, excessive current may flow through the DIO lines and damage the device.

+5 V Supply Pin on the DIO Connector

The +5 V supply pin on the DIO connector is connected to circuitry that protects the host computer or chassis in which it is installed. The +5 V pin on the DIO connector supplies +5 V to external circuitry. The +5 V output current is electronically limited.

If the current limit is exceeded, the +5V supply automatically shuts down within 10 µs. Approximately five seconds after the shutdown, and every five seconds thereafter, the internal circuitry attempts to restart the output supply, even if the condition that caused the shutdown still exists. For example, if the +5 V pin is permanently loaded such that it draws more than 500 mA at 5 V, every five seconds, the +5 V supply attempts a restart, which drives the +5 V signal with approximately 500 mA for 10 µs. This condition does not harm the NI 5641R; however, external circuitry could be damaged if it cannot tolerate the 500 mA restart pulse.

Caution Do not connect the +5 V power pins directly to analog or digital ground or to any other voltage source on the NI 5641R or any other device under any circumstance. Doing so can damage the NI device and the host computer or chassis. NI is not liable for damage resulting from such a connection.

Programming the NI 5641R

You have two options for programming the NI 5641R. You can either use the LabVIEW FPGA Module, or you can use the NI-5640R LabVIEW instrument driver API to generate or acquire data.

Related Information

Programming Methods Comparison on page 4

NI-5640R Instrument Driver Examples

The NI-5640R software includes many example programs. Examples demonstrate the functionality of the device and serve as programming models and building blocks for your own applications.

Locating Examples

You can use the NI Example Finder to easily browse and search installed examples. You can see descriptions and compatible devices for each example or see all the examples compatible with one particular device.

1. Launch LabVIEW.
2. Select **Help**»**Find Examples** and navigate to **Hardware Input and Output**»**Modular Instruments**»

Troubleshooting

This section describes how to troubleshoot common issues. If an issue persists after you complete the troubleshooting procedure, contact NI technical support or visit *ni.com/support*.

The ni5640R Check Thermal Shutdown VI Indicated an Overtemperature Condition and My Device Shut Down. What Should I Do Next?

The NI 5641R has an onboard sensor that monitors the device operating temperature and can shut down the device. The sensor temperature is accessible using the ni5640R Check Thermal Shutdown VI, which is available both in the NI-5640R instrument driver or in the LabVIEW FPGA Template library. The *NI PXIe-5641R Specifications* lists typical and maximum operating temperatures.

The NI 5641R also has a hardware fail-safe protection sensor that disables clocks if the temperature exceeds the maximum safe operating limits of the module. When running the included single-tone generator example in an NI PXIe-1065 chassis at room temperature, the sensor typically reads a temperature of 40°C to 45°C. Temperature sensor readings approaching 90°C indicate that the NI 5641R is operating near its maximum temperature.

In the case of a thermal shutdown, review the following recommendations to correct common overheating causes.

- Verify that all chassis covers, filler panels and/or slot blockers are installed. Replace any missing items.
- Adjust the chassis fan speed to a higher setting.

  Typically you change the fan speed using a switch on the back of the chassis.
- Move the chassis to unblock the fan air inlet and/or clean the fan filters.
- Move the chassis to a cooler ambient temperature location.

The ambient temperature around the chassis is too high. This condition may be caused by high ambient air temperature or proximity to heat-generating devices.

- Ensure the NI 5641R is not installed Slot 7 or 8 in an eight-slot chassis. If it is, move the module to a different slot.

If the problem persists, refer to *Re-enabling a Device After Thermal Shutdown*.

Re-Enabling a Device After Thermal Shutdown

To re-enable your device after a thermal shutdown, you must perform a power-on reset.

1. Power down the chassis that contains the device.
2. Review the procedure in *Installing the Hardware* and make any necessary adjustments to correct the cause of the overheating.
3. Restart the chassis.

My Signal Looks Very Noisy

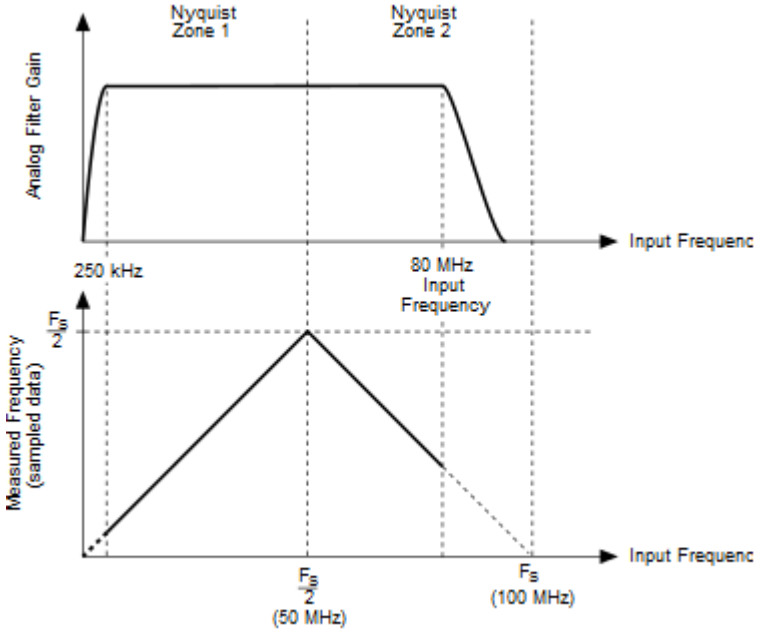
Acquired high-frequency signals can appear as lower frequency signals, which is also known as aliasing. You can use lowpass filtering to minimize the effects of aliasing.

Apply a lowpass filter (or bandpass filter for undersampling applications) in the analog signal chain before digitization.

The NI 5641R includes a lowpass filter in the onboard circuitry.

The following figure illustrates aliasing in an undersampled application. In this figure you can observe aliasing at input frequencies greater than $0.5 \times F_S$.

Figure 2. Aliasing in an Undersampled Application



Related Information

For more information about avoiding alias effects, refer to the Alias Effects topic for your device in the NI IF Transceivers Help.

It Takes a Long Time to Access Controls and Indicators Using the Read/Write Control Function in LabVIEW FPGA

When you select the Read/Write Control function in LabVIEW FPGA, LabVIEW locates the corresponding controls and indicators in your FPGA VI. If your FPGA VI is not loaded into memory at this time, LabVIEW loads it now and unloads it after the operation completes. You can prevent LabVIEW from loading and unloading the FPGA VI in two ways.

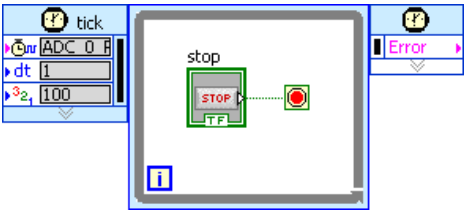
- Keep the FPGA VI open while you work on your host VI.
- Target the bitfile directly.
  a) Right-click the Open FPGA VI Reference function and select **Configure Open FPGA VI Reference** from the shortcut menu.

     The **Configure Open FGPA VI Reference** dialog box appears.
  b) Select the **Bitfile** control.
  c) Browse to its location.

An Error Was Detected in the Communication between the Host Computer and the FPGA Target

`Error -61046` may occur when your application tries to read or write values that are in a slow clock domain. For example, you may see this error if you have a control or indicator inside a single-cycle Timed Loop that is clocked by the ADC_0_Port_A_Clk and have configured the ADC 0 to decimate by a large value (larger than 128). Such a setup can cause this problem because the control or indicator is updated slowly, and the communication times out while trying to access it. This configuration is shown in the following figure.

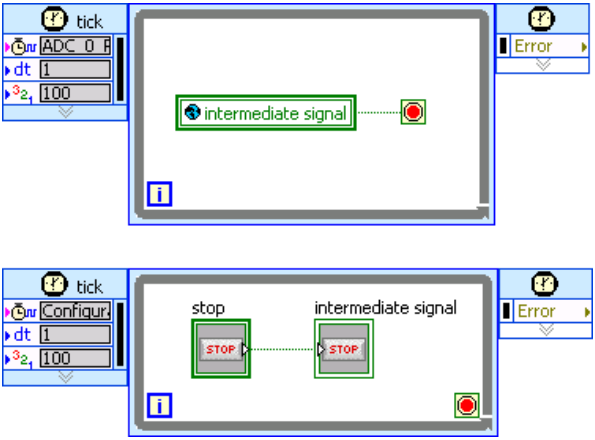
Figure 3. Controls in a Slow Clock Domain



To solve this problem, complete the following steps:

1. Place the control or indicator in a clock domain that is faster, such as the default Top-Level Clock.
2. Use global variables to transfer the value to a slower clock domain, as shown in the following figure.

Figure 4. Using Global Variables to Transfer the Value



When I Open a Host VI, LabVIEW Cannot Find a File Named `filename.lvbit` or `filename.lvbitx`

Before you try to resolve the error, you must have write permissions to the `typedef.ctl` file, and it must not be flagged as read-only.

LabVIEW sometimes cannot locate `.lvbit` or `.lvbitx` files if your subVIs have been saved with the type definition control pointing to another FPGA VI. In this case, when your subVI loads, it attempts to locate the original FPGA VI.

If LabVIEW reports this error, complete the following steps:

1. If LabVIEW reports that it cannot find one of these files, click **Cancel**.

   The VI should load properly. When the VI is finished loading, if the **Bind FPGA host reference to type definition** option is selected in the **Configure Open FPGA Reference** dialog box, the typedef is updated to point to the correct FPGA VI.
2. If you are using the Open FGPA VI Reference function in dynamic mode (configured by selecting the **Dynamic mode** checkbox in the **Configure Open FPGA VI Reference** dialog box), you do not need to bind the FPGA host reference to the type definition. Otherwise, if the type definition is not updated, complete the following steps:
   a) Right-click the Open FPGA VI Reference function, and select **Configure Open FPGA VI Reference**.
   b) Verify the **Open** and **Bind FPGA host reference to type definition** settings in the **Configure Open FGPA VI Reference** dialog box.

When I Run an FPGA VI, a Compilation Error Instantly Occurs

- Check the compilation report and ensure your FPGA VI is not violating any timing constraints.
- Force a compilation by right-clicking your build specification and selecting **Rebuild**.

  The compilation process may abnormally abort without being detected by LabVIEW. In this case, LabVIEW caches the error report; and, because LabVIEW detects that your FPGA VI has not changed, it simply returns the cached result for every compilation request unless you force rebuilding the compilation.

I Received a “Clock domain crossing not supported for FIFO memory” Error

FIFOs are created from three types of FPGA memory: flip flops, lookup tables, and block memory. The memory type is automatically chosen based on the FIFO depth, regardless of the bit width of your samples.

Table 3. FIFO Depth and Type

FIFO Depth	Selected FPGA Memory Type
1 to 2 samples	Flip flops
<300 bytes	Lookup tables
≥300 bytes	Block memory

Of these three types, only block memory FIFOs can be read and written from different clock domains. If you receive the **`Clock domain crossing not supported for FIFO memory type`** error, try the following suggestions to resolve the error:

- Increase your FIFO size to greater than 300 bytes.
- Find the FIFO in the `Generated FIFOs` folder in your project and change its type to block memory.

Connecting Two Asynchronous Data Wires Produces a Broken Wire

Some asynchronous input terminals only accept certain data exchange policies, so you see a broken wire if you attempt to connect two such terminals. For example, if you connect two square terminals and get a broken wire, you may have connected a register to an input that requires a FIFO.

Related Information

Refer to the Data Exchange Policies section of the NI IF Transceivers Help for more information about the types of data exchange policies.

Connecting Two Asynchronous Timing Wires Produces a Broken Wire

A wire between two clock terminals breaks if the connected node does not support the timing type of the wire.

Asynchronous timing wires come in three types: base clocks, derived clocks, and pulse generator-created.

Related Information

Refer to the NI IF Transceivers Help for more information about these clock types.

I Lost My Harness Node Settings

After you drop a VI onto the Harness node, the Harness node displays the adopted VI in the center and creates asynchronous wire terminals for the input and output data. Redropping a VI onto the Harness node erases any data exchange policy configuration changes you made using the embedded VI terminals.

Appendix A: NI PXIe-5641R Front Panels

The NI 5641R front panels contain seven connectors—four SMA jack connectors, two SMB connectors, and one 9-pin mini-circular DIN connector.

Figure 5. NI PXIe-5641R Front Panel Connectors

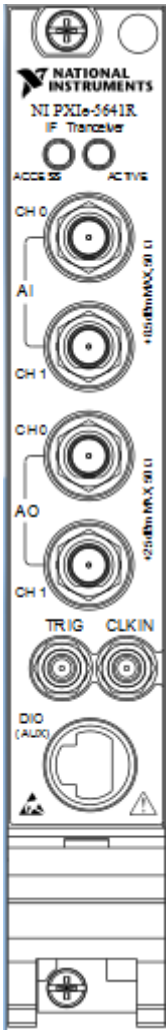


Table 4. NI 5641R Connectors

Connector Name	Type	Description
AI CH <0..1>	SMA	Analog input terminals for the NI 5641R .
AO CH <0..1>	SMA	Analog output terminals for the NI 5641R .
CLK IN	SMA	Input terminal for an external Reference or Sample Clock.
TRIG	SMA	Input or output terminal for device trigger signals.
DIO (AUX)	9-pin DIN mini-circular	Input or output terminal for device digital I/O (DIO) channels. DIO lines are direction-configurable by pin as input or output. The following figure and table provide the detailed pinout for this connector.

Figure 6. DIO 9-Pin DIN Connector

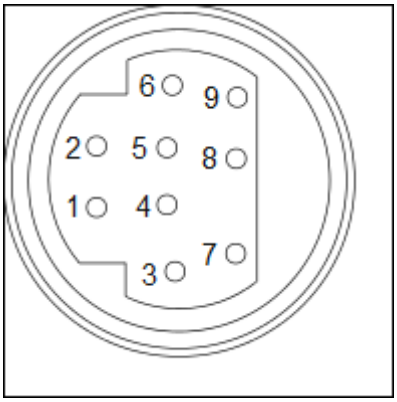


Table 5. DIO 9-Pin DIN Connector Pinout

Pin Number	Connection
1	+5 V
2	Digital ground (DGND)
3	DIO_01
4	DIO_02
5	DIO_03

Table 5. DIO 9-Pin DIN Connector Pinout (Continued)

Pin Number	Connection
6	DIO_04
7	DIO_05
8	DIO_06
9	DIO_07

The following two tables describe the different conditions indicated by the two PXI Express LEDs.

Table 6. ACTIVE LED Indicators

Color	Indications
Off	Device not transmitting or is experiencing an error.
Amber	Either of the DAC outputs or the Aux DIO port is enabled.
Green	Device is transmitting or writing to DIO lines.
Red	Error condition.

Table 7. ACCESS LED Indicators

Color	Indications
Off	Device not ready or FPGA has not been loaded.
Amber	Device being accessed by software.
Green	Device ready to be used.
Red	Error condition.

Related Information

Connecting Signals on page 8

You can connect three kinds of signals to the NI 5641R: analog input (AI) signals, analog output AO) signals, and digital input/output (DIO) signals.

Appendix B: Creating an IF Transceiver Application in LabVIEW

You can program the NI 5641R using LabVIEW FPGA or the NI-5640R instrument driver.

Creating an Application Using LabVIEW FPGA

To develop and use an FPGA VI with the NI 5641R, you must complete four major steps.

1. Create the LabVIEW project.
2. Add the hardware target.
3. Develop and compile the FPGA VI.

   Note You can develop the FPGA VI using either the traditional LabVIEW FPGA method or the NI-5640R Asynchronous Programming palette.

4. Create the host VI to communicate with the target.

> **Simple Spectrum Analyzer Example**
>
> The following sections walk you through creating a simple spectrum analyzer using one of the analog input channels on the NI 5641R. You can configure the center frequency and the bandwidth to analyze. The application also generates a sine tone that can be used as the signal to analyze.

Creating a Project Using the ni5640R Template

The ni5640R template contains all the elements needed to develop an application using the NI 5641R. It contains a LabVIEW project that has been preconfigured to target the NI 5641R. It also contains a Template FPGA and a Template Host VI. Complete the following steps to use the ni5640R Template:

1. Create a folder named `My Simple Spectrum Analyzer` in a new location, such as your desktop.
2. In LabVIEW, select **File**»**New Project** to display the **Project Explorer** window.
3. Right-click the **My Computer** target and select **New**»**Targets and Devices** to display the **Add Targets and Devices** dialog box.
4. Select **New target or device**.
5. Expand **IF Transceivers**.
6. Select the NI 5641R and click **OK**.
7. Save the generated top level FPGA VI as `My Simple Spectrum Analyzer(FPGA).vi`.
8. In the **Project Explorer** window, verify that the NI 5641R FPGA target and the **ni5640R Template.lvlib** appear. The FPGA target contains a VI that has the code necessary for hardware configuration.
9. Right-click the **My Computer** target in the **Project Explorer** window and select **New**» **VI**. Verify that an untitled VI appears in the **Project Explorer** window.
10. In the **Project Explorer** window, right-click the untitled VI and select **Save As**.
11. Save the VI as `My Simple Spectrum Analyzer (HOST).vi` to the `My Simple Spectrum Analyzer` folder created in step 1.

12. Click **OK**.
13. In the **Project Explorer** window, select **File»Save All (this Project).**
14. Save the project as `My Simple Spectrum Analyzer.lvproj` to the `My Simple Spectrum Analyzer` folder created in step 1.
15. Click **OK**.

Developing the FPGA VI

FPGA VIs run on FPGA targets and define the functionality and features of the targets on which they run. In this section, you use a custom VI to build an FPGA VI that acquires data using one of the analog input channels and then sends packets of data to the host VI using direct memory access (DMA).

You have two choices when creating an FPGA VI using LabVIEW FPGA—the traditional LabVIEW FPGA method or the NI-5640R Asynchronous Programming palette. In general, using the NI-5640R Asynchronous Programming palette requires fewer configuration steps to develop the FPGA VI.

Note Asynchronous programming works only on the NI PCI-5640R and NI PXIe-5641R targets and should not be used for code you may need to adapt for other classes of LabVIEW FPGA targets.

Developing an FPGA VI Using Traditional LabVIEW FPGA

Adding Resources to the Project

To use traditional LabVIEW FPGA programming to develop the FPGA VI, all resources used by the FPGA VI must be added to the project. Complete the following steps to add FPGA I/O, a DMA FIFO, and an FPGA base clock resource.

1. Right-click **FPGA Target** in the **Project Explorer** window and select **New»FPGA I/O**.

   The **New FPGA I/O** dialog box appears and displays the I/O resources available on the NI 5641R.
2. In the **New FPGA I/O** dialog box, expand **Analog Input**.
3. Select **ADC_0_Port_A_I** and **ADC_0_Port_A_Q**.
4. Click the right arrow to add these resources to the New FPGA I/O table.
5. Click **OK**.

   The FPGA I/O items you configured appear in the **Project Explorer** window in a folder below the FPGA target.
6. Right-click the FPGA target in the **Project Explorer** window and select **New»FIFO**.

   This FIFO transfers the acquired data to the host computer.
7. In the **FPGA FIFO Properties** dialog box, select the **General** category.
8. Set **Type** to **Target to Host – DMA**.
9. Set **Requested Number of Elements** to `32767` .
10. In the **FPGA FIFO Properties** dialog box, select the **Data Type** category.
11. Set **Data Type** to **U32**.
12. Click **OK**.

13. Right-click the FPGA target in the **Project Explorer** window and select **New»FPGA Base Clock**.
14. In the **FPGA Base Clock Properties** dialog box, set **Resource** to **ADC_0_Port_A_Clk**.

    The ADC_0_Port_A_Clk ticks every time the ADC has a new I/Q sample.
15. Click **OK**.

Developing Your FPGA VI

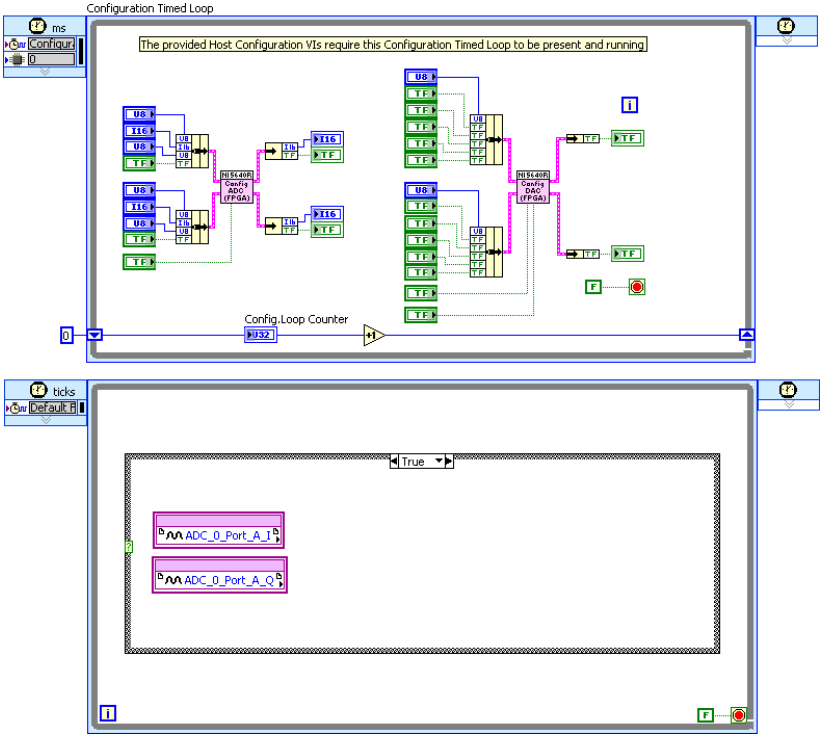
1. Double-click **My Simple Spectrum Analyzer (FPGA).vi** in the **Project Explorer** window to open the VI.
2. Create a Timed Loop beneath the Configuration Timed Loop.

    The Timed Loop is located on the Functions palette under **Programming»Structures» Timed Structures**.
3. Wire a FALSE constant to the Loop Condition terminal.
4. Select both Analog Input FPGA I/O items in the **Project Explorer** window and drag them onto the block diagram inside the timed loop you created.

    Two FPGA I/O Nodes appear on the block diagram configured with the specific FPGA I/O item.
5. Create a Case structure around the two FPGA I/O Nodes.

    The Case structure is located on the Functions palette under **Programming»Structures**. Your block diagram should now look like the following figure.

Figure 7. Adding I/O Nodes to the FPGA VI



6. Select the FIFO item in the **Project Explorer** window and drag it onto the block diagram inside the Case structure, to the right of the FPGA I/O Nodes.
7. Place a Join Numbers function between the FPGA I/O Nodes and the FIFO.

   The Join Numbers function is located on the Functions palette under **Programming**» **Numeric**»**Data Manipulation**.
8. Wire the outputs of the FPGA I/O Nodes to the inputs of the Join Numbers function.
9. Wire the output of the Join Numbers function to the **Element** input of the FIFO.
10. Create a numeric constant of 0 and wire it to the **Timeout** input of the FIFO.
11. Add a global variable to your loop by completing the following steps:
    a) Place a global variable on the block diagram to the left of the ADC nodes inside the Timed Loop but outside the Case structure.

       The global variable is located on the Functions palette under **Programming**» **Structures**.
    b) Double-click the global variable to open its front panel.
    c) Create a Boolean control called **Acquire**.

d) From the front panel, go to **File»Save As** and save the global VI as `My Simple Spectrum Analyzer (Global Acquire).vi` to the `My Simple Spectrum Analyzer` folder created in step 1 of *Using the ni5640R Template*.

e) Click **OK** and close the VI.

f) Return to the block diagram for the `My Simple Spectrum Analyzer (FPGA).vi` and right-click the global variable node.

g) Select **Item»Acquire**.

h) Right-click the global variable node and select **Change to Read**.

i) Wire **Acquire** to the Case structure.

12. Add a second global variable to your loop by completing the following steps:

a) Place a global variable on the block diagram inside the case structure to the right of the FIFO I/O Node.

b) Double-click the global variable to open its front panel.

c) Create a Boolean indicator called **Timed Out?**.

d) From the front panel, select **File»Save As** and save the global VI as `My Simple Spectrum Analyzer (Global Timed Out).vi` to the `My Simple Spectrum Analyzer` folder created in step 1 of *Using the ni5640R Template*.

e) Click **OK** and close the VI.

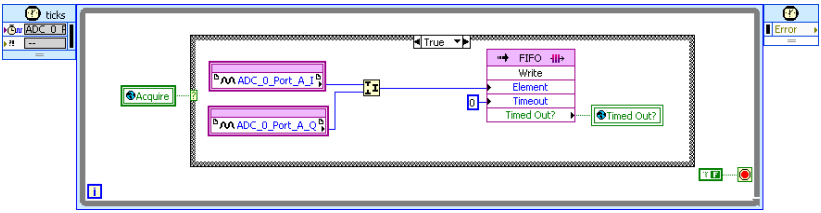
f) Return to the block diagram for your `My Simple Spectrum Analyzer (FPGA).vi` and right-click the second global variable node.

g) Select **Select Item»Timed Out?**.

13. Wire the FIFO I/O Node **Timed Out?** output to the **Timed Out?** global variable.

Your block diagram should now look like the following figure.

Figure 8. Timed Loop Setup



The acquired I and Q data, which are 16-bit values, are packed into a 32-bit value by the Join Numbers function. The packed value is then sent to the FIFO, which transfers the data to the host computer using DMA.

Specifying How Often to Acquire I and Q Samples

1. On the FPGA VI block diagram, double-click the **Input Node** (top left corner of the loop) on the Timed Loop to view the **Configure Timed Loop** dialog box.

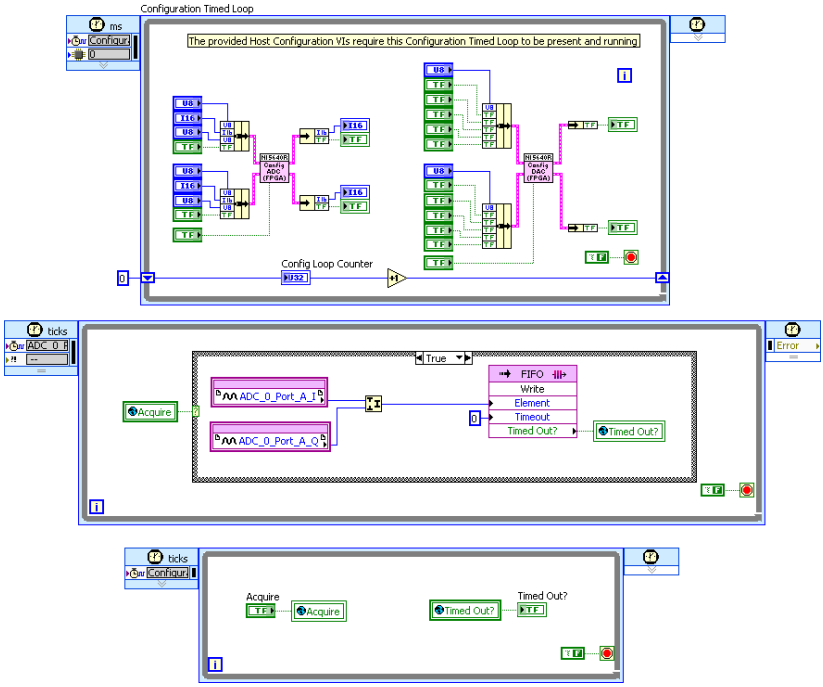
This dialog box allows you to configure which clock runs the Timed Loop. Notice that only those clocks that have been added to the project appear in the **Available Timing Sources** list.

2. In the **Configure Timed Loop** dialog box, click **Select Timing Source** and select **ADC_0_Port_A_Clk**.
3. Click **OK**.
4. Create a new Timed Loop beneath the second Timed Loop.

   The Timed Loop is located on the Functions palette under **Programming»Structures» Timed Structures.**
5. Wire a FALSE constant to the **Loop Condition** terminal.
6. Copy the **Acquire and Timed Out?** global variables and place them inside the new Timed Loop created in step 4.
7. Right-click the **Acquire** global variable and select **Change to Write**.
8. Right-click the **Timed Out?** global variable and select **Change to Read**.
9. Wire the **Acquire** global variable to a Boolean control.
10. Wire the **Timed Out?** global variable to a Boolean indicator.
11. On the FPGA VI block diagram, double-click the input node (top left corner of the loop) on the third Timed loop to view the **Configure Timed Loop** dialog box.
12. In the **Configure Timed Loop** dialog box, click **Select Timing Source** and select **Configuration_Clk**.

    Your block diagram should now look like the following figure.

Figure 9. Completed FPGA Block Diagram Using Traditional LabVIEW FPGA Programming



13. Save the VI and the project.

Your FPGA VI is now complete using traditional LabVIEW FPGA programming. Skip to the *Compiling the FPGA VI* section for the next steps you should take.

Using the NI-5640R Asynchronous Programming Palette

Before using the NI-5640R Asynchronous Programming palette, complete all the steps in the *Using the ni5640R Template* section.

1. Double-click the `My Simple Spectrum Analyzer(FPGA).vi` to open the VI and then open the block diagram.
2. Scroll down the block diagram to the area labeled "ADD YOUR CODE HERE."
3. Place the ADC node on the block diagram in this area.

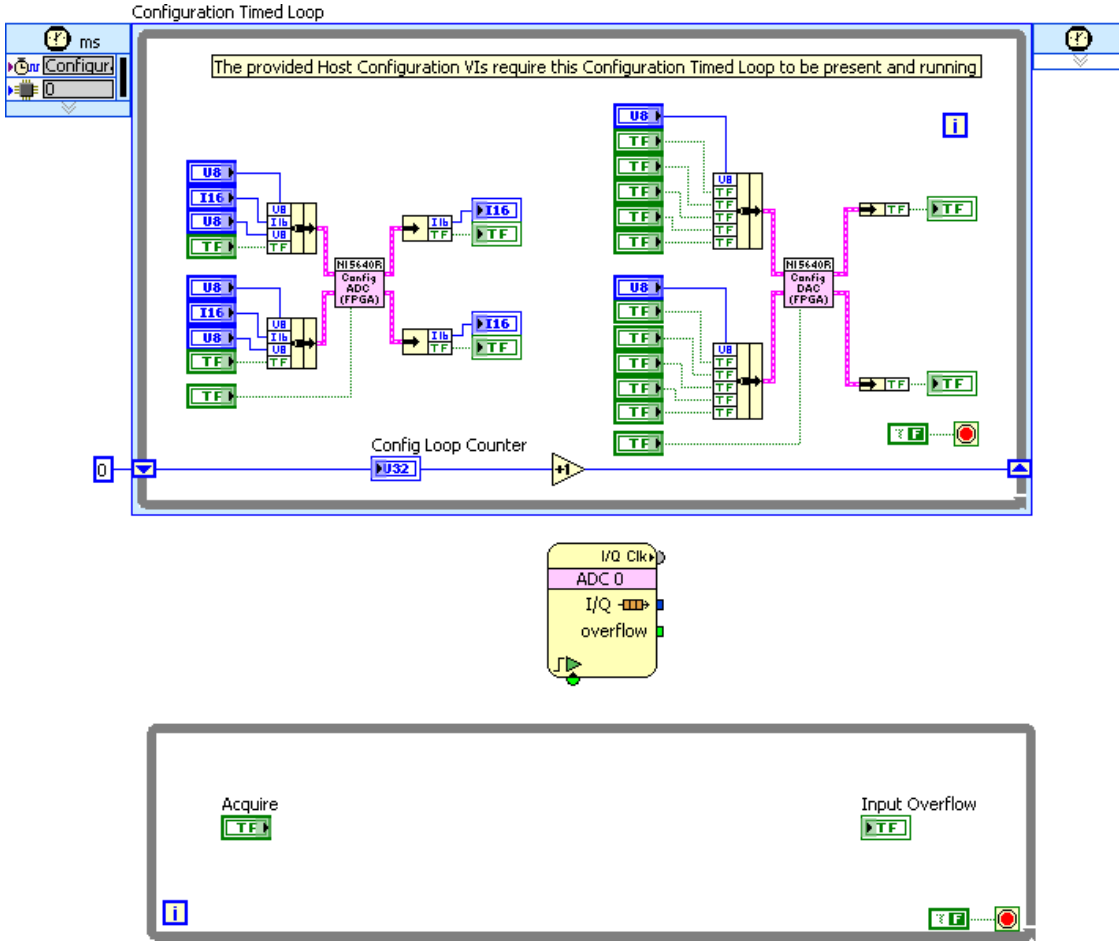
   The ADC node is located on the Functions palette under **Instruments**»**NI-5640R Asynchronous Programming**.
4. Create a While Loop below the ADC node. The While Loop is located on the Functions palette under **Programming**»**Structures**»**Timed Structures**.
5. Wire a FALSE constant to the Loop Condition terminal.
6. Open the front panel of the My Simple Spectrum Analyzer (FPGA) VI and create a Boolean Push Button control.

   This control is located on the Controls palette under Boolean.
7. Rename the control as `Acquire`.
8. Create a Boolean LED indicator.

   This indicator is located on the Controls palette under Boolean.
9. Rename the indicator as `Input Overflow`.
10. On the LabVIEW block diagram, drag the **Acquire** control and the **Input Overflow** indicator into the While Loop.

    Your block diagram should now look like the following figure.

Figure 10. My Simple Spectrum Analyzer (FPGA) VI Block Diagram



11. Place a Write Accessor node in the While Loop to the right of the **Acquire** control.

    The Write Accessor node is located on the Functions palette under NI-5640R Asynchronous Programming.
12. Wire the **Acquire** control to the Write Accessor node.
13. Wire the Write Accessor node to the **start trigger** parameter on the ADC node.
14. Place a Read Accessor node inside the While Loop to the left of the **overflow** indicator on the ADC node.

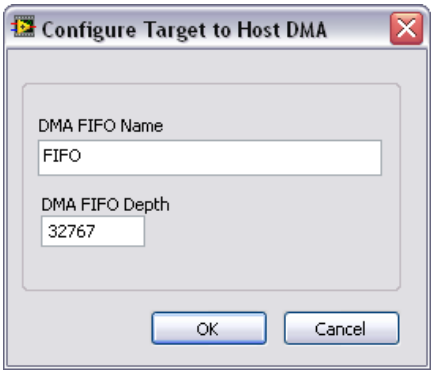
    The Read Accessor node is located on the Functions palette under NI-5640R Asynchronous Programming.
15. Wire the **overflow** indicator on the ADC node to the Read Accessor node.
16. Wire the Read Accessor node to the **Input Overflow** indicator.

17. Place a DMA to Host node outside the While Loop to the right of the ADC node.

    The DMA to Host node is located on the Functions palette under NI-5640R Asynchronous Programming.

18. Click the DMA FIFO glyph on the DMA to Host node to launch the **Configure Target to Host** dialog box and make the changes shown in the following figure.

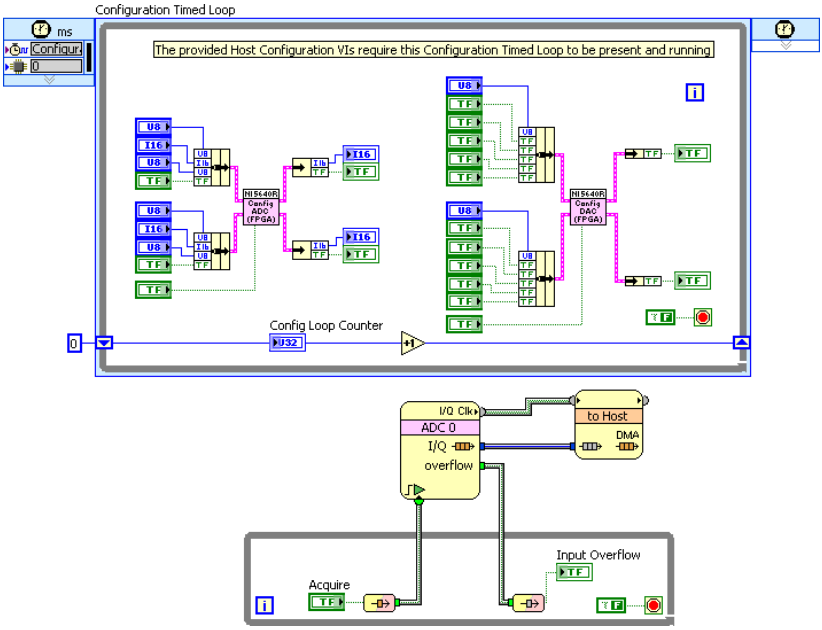
Figure 11. Configure Target to Host DMA Dialog Box



19. Wire the ADC **I/Q Data** terminal to the **data in** parameter of the DMA to Host node.
20. Enable the **clock domain** terminal of the DMA to Host node by right-clicking the node and selecting **Add Clock Domain Terminal**.
21. Wire the **I/Q Clk** terminal to the **clock domain** parameter of the DMA to Host node.

    Your block diagram should now look like the following figure.

Figure 12. Completed FPGA VI Block Diagram Using NI-5640R Asynchronous Programming Palette



22. Save the VI and the project.

Your FPGA VI is now complete using the NI-5640R Asynchronous Programming palette.

Refer to the *Compiling the FPGA VI* section for the next steps you should take.

NI-5640R FPGA and Asynchronous Programming Examples

The NI-5640R FPGA and asynchronous programming examples demonstrate some of the functionality of the NI IF transceiver that you can use or integrate into your applications.

These examples serve as interactive tools, programming models, and building blocks for your own applications. For the installation location of the example files, refer to the *NI-5640R Readme*.

Compiling the FPGA VI

Now that you have written an FPGA VI using either traditional or asynchronous programming, complete the following steps to compile the FPGA VI.

1. Right-click `My Simple Spectrum Analyzer (FPGA).vi` in the **Project Explorer** window and select **Create Build Specification**. Verify that My Simple Spectrum Analyzer (FPGA) appears under the Build Specifications folder in the **Project Explorer** window.
2. Right-click **My Simple Spectrum Analyzer (FPGA)** under the **Build Specifications** folder and select **Properties**.

3. In the **My Simple Spectrum Analyzer (FPGA) Properties** dialog box, select the **Information** category.
4. In the **Bitfile name** field, shorten the name to `MySimpleSpectrumAnalyzer(FPGA).lvbitx`.
5. Click **OK**.
6. Right-click the **My Simple Spectrum Analyzer (FPGA)** build specification in the **Project Explorer** window and select **Build**.

   When the compilation process is complete, the LabVIEW FPGA Compile Server displays a report indicating that the compilation was successful. Click **OK** to close the dialog box.

   Note Compilation can take a significant amount of time. The length of time depends on system processing power and available memory. Compilation could take several hours for complex VIs. You can develop the host VI, as described in the following section, while the FPGA VI compiles.

Developing the Host VI

To communicate programmatically with the FPGA VI, you must develop a host VI that runs on a Windows or an RT target. The host VI communicates with the FPGA VI running on the NI 5641R.

Opening a Reference to the FPGA VI

1. Open the block diagram for the `My Simple Spectrum Analyzer (HOST).vi`.
2. Place the Open FPGA VI Reference function on the block diagram.
3. Right-click the Open FPGA VI Reference function and select **Configure Open FPGA VI Reference**.
4. Select the **VI** button to launch the **Select VI** dialog box.
5. Select the `My Simple Spectrum Analyzer (FPGA) VI` you created in the *Developing the FPGA VI* section.
6. Ensure the **Run the FPGA VI** checkbox is selected.
7. Ensure the **Dynamic mode** checkbox is selected.
8. Click **OK**.
9. Right-click the resource name connector of the Open FPGA VI Reference function and create a control.

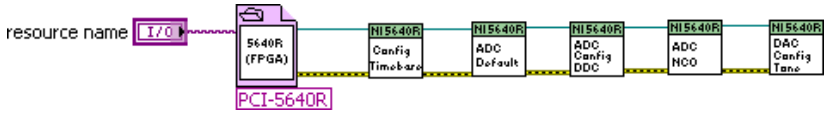
Adding NI-5640R VIs

1. In the **Project Explorer** window, expand the `Template` library and double-click the ni5640R VI Tree to open it.
2. Open the ni5640R VI Tree block diagram.
3. Copy the following VIs from the ni5640R VI Tree block diagram to the block diagram of `My Simple Spectrum Analyzer (HOST).vi`.
   - ni5640R Configure Timebase VI—Configures clocks.
   - ni5640R ADC Default VI—Configures analog input default settings.
   - ni5640R ADC Configure DDC VI—Configures analog input digital downconverter settings.

- ni5640R ADC Configure NCO VI—Configures analog input center frequency settings.
- ni5640R DAC Configure for Single-Tone Mode VI—Generates sine tone stimulus.

4. Connect the preceding VIs as shown in the following figure.

Figure 13. Host VI Block Diagram Connections



Acquiring Data From the FPGA VI

1. Place an Invoke Method function on the block diagram to the right of the ni5640R DAC Config for Single Tone VI.

   The Invoke Method function is available on the **Functions»FPGA Interface** palette.
2. Wire the Invoke Method function to the ni5640R DAC Config for Single Tone VI.
3. Select the word **Method** on the node and select **FIFO»Configure**.

   This method configures the size of the software FIFO and limits the maximum number of elements that your VI can read.
4. Place and wire another Invoke Method function on the block diagram.
5. Select the node and select **FIFO»Start**.

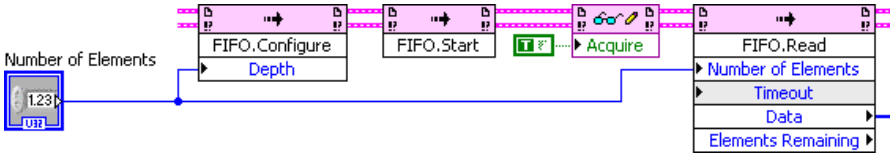
   This method starts DMA transfer from the DMA FIFO that you created in the FPGA VI to a software DMA FIFO.
6. Place and wire a Read/Write Control function on the block diagram.

   The Read/Write Control function is available on the **Functions»FPGA Interface** palette.
7. Select the word **Unselected** on the Read/Write Control function and select **Acquire**.
8. Wire a Boolean TRUE constant to the **Acquire** input on the Read/Write Control function.
9. Place and wire another Invoke Method function on the block diagram.
10. Select the Invoke Method function and select **FIFO»Read**.

    This method reads the contents of the software DMA FIFO.
11. Create a numeric control named `Number of Elements` and wire it to the **Depth** input in the FIFO.Configure method and to the Number of Elements parameter in the FIFO.Read method, as shown in the following figure.

The VIs you placed and wired in this section are shown in the following figure. The code in this figure should be wired to the code shown in the preceding section.

Figure 14. Wiring the FPGA VI for Data Acquisition



Analyzing the Data

1. Place a Split Number VI on the block diagram to the right of the Invoke Method Node you placed and wired in step 9 in the preceding section.

   The Split Number VI is located on the Functions palette under **Programming»Numeric» Data Manipulation**.
2. Wire the Data output from the **Method»FIFO»Read** to the input of the Split Number VI.
3. Place and wire two To Word Integer functions on the block diagram to convert the outputs of the Split Number VI to I16 format.

   The To Word Integer function is located on the Functions palette under **Programming» Numeric»Conversion**.
4. Place and wire a Real-Imaginary to Complex Number VI on the block diagram.

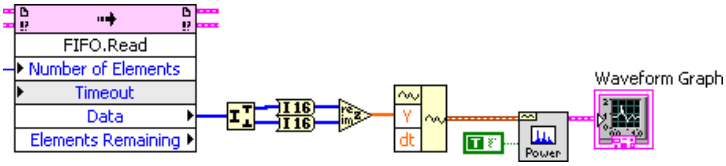
   The Real-Imaginary to Complex Number VI is located on the Functions palette under **Programming»Numeric»Complex**.
5. Place and wire a Build Waveform VI on the block diagram.

   The Build Waveform VI is located on the Functions palette under **Programming» Waveform**.
6. Place and wire an FFT Power Spectrum VI and PSD VI on the block diagram.

   The FFT Power Spectrum VI is located on the Functions palette under **Signal Processing»Waveform Measurements**.
7. Wire a TRUE constant to the dB On parameter of the FFT Power Spectrum VI.
8. Place a Waveform Graph indicator, located on the Waveform palette, on the front panel.

The VIs you placed and wired in this section are shown in the following figure. The code in this figure should be wired to the code shown in the preceding section.

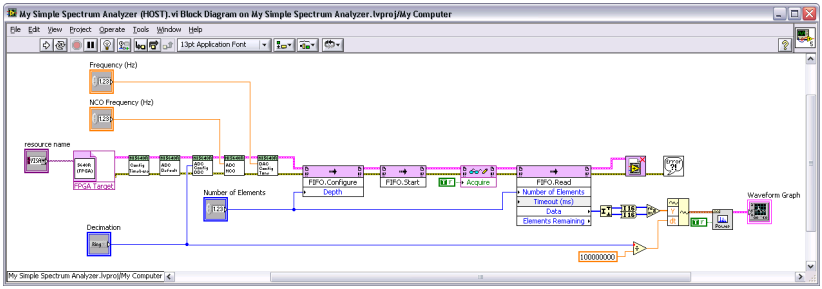
Figure 15. Analyzing the Data



Adding Controls and Indicators

1. Add the following controls and indicators, as shown in the following figure:
   - **Decimation**—Specifies how much decimation is performed on the analog input signal. This control determines the maximum bandwidth of your signal.
   - **NCO frequency**—Specifies the center frequency for the analog input waveform.
   - **Frequency**—Specifies the frequency at which to generate a sine tone. This tone can serve as a test signal for the simple spectrum analyzer.

Figure 16. Host VI Block Diagram With Controls and Indicators



2. Divide the decimation rate value by 100M, which is the default Sample Clock rate of the ADCs, and wire the output to the **dt** input of the Build Waveform VI.

Closing the Reference to the FPGA VI

1. Close the reference to the FPGA VI running on the FPGA target by placing the Close FPGA VI Reference VI on the block diagram as shown in the following figure.

   The Close FPGA Reference VI is available on the Functions palette under FPGA Interface.

2. Place a Simple Error Handler VI on your block diagram.

   The Simple Error Handler VI is available on the Functions palette under **Programming**» **Dialog & User Interface**.

3. Wire the Close FPGA VI Reference function to the Simple Error Handler VI.

4. Save the VI.

   Your host VI block diagram and front panel should now look like the following two figures.

Figure 17. Host VI Block Diagram

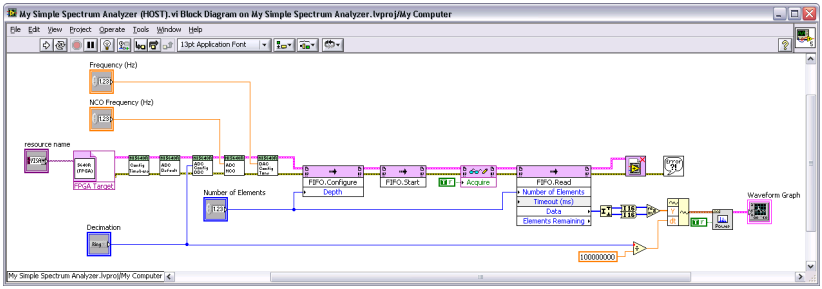
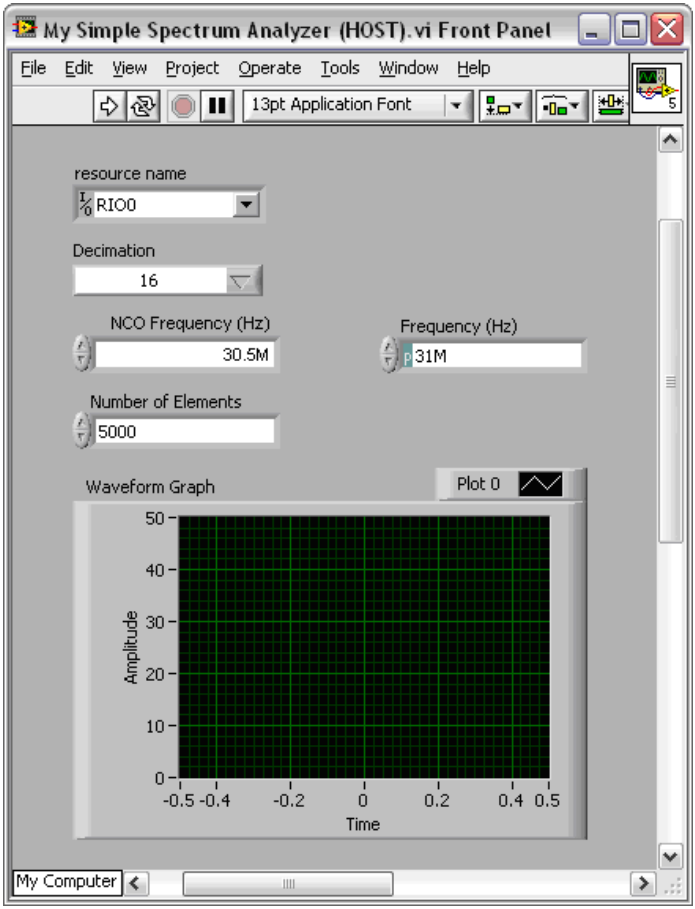


Figure 18. Host VI Front Panel



5. Populate your application with the control values shown in the *Host VI Front Panel* .
6. On the NI 5641R front panel, connect the AO 0 connector to the AI 0 connector using an SMA-to-SMA cable so that the output tone from AO 0 is fed into AI 0.
7. From the **Project Explorer** window, select **File»Save All (this Project)** when the FPGA compile is complete.
8. Click the **Run** button to run the host VI. The host VI programmatically downloads and runs the FPGA VI on the NI 5641R.

The spectrum of the generated tone appears on the host VI front panel waveform graph. The tone appears at 500 kHz because the generated tone frequency and the analyzer center frequency are 500 kHz apart.

Creating an Application Using the NI-5640R Instrument Driver

You can control NI 5641R programmatically using the supplied NI-5640R instrument driver.

The NI-5640R instrument driver controls the NI 5641R in its default personality of two synchronous input and two synchronous output channels. You also can run the NI-5640R examples to demonstrate the functionality of your device.

Related Information

For more information about programming with the NI-5640R instrument driver, refer to the NI IF Transceiver Help. This help file contains hardware information, concepts, a detailed VI reference for the NI-5640R instrument driver API, and information specific to your device.

NI-5640R Instrument Driver Examples

The NI-5640R software includes many example programs. Examples demonstrate the functionality of the device and serve as programming models and building blocks for your own applications.

Locating Examples

You can use the NI Example Finder to easily browse and search installed examples. You can see descriptions and compatible devices for each example or see all the examples compatible with one particular device.

1. Launch LabVIEW.
2. Select **Help»Find Examples** and navigate to **Hardware Input and Output»Modular Instruments»**

Where Do I Go Next?

For more detailed information about the NI 5641R and the NI-5640R, you can use the following resources.

- *NI IF Transceivers Help* —Contains more information about hardware features and programming. You can access this document at **Start»All Programs»National Instruments»NI-5640R»Documentation»NI IF Transceivers Help**.
- *NI PXIe-5641R Specifications*—Contains detailed specifications. This document is printed and included in your kit and is also available online.

Worldwide Support and Services

The National Instruments website is your complete resource for technical support. At *ni.com/support* you have access to everything from troubleshooting and application development self-help resources to email and phone assistance from NI Application Engineers.

Visit *ni.com/services* for NI Factory Installation Services, repairs, extended warranty, and other services.

Visit *ni.com/register* to register your National Instruments product. Product registration facilitates technical support and ensures that you receive important information updates from NI.

A Declaration of Conformity (DoC) is our claim of compliance with the Council of the European Communities using the manufacturer's declaration of conformity. This system affords the user protection for electromagnetic compatibility (EMC) and product safety. You can obtain the DoC for your product by visiting *ni.com/certification*. If your product supports calibration, you can obtain the calibration certificate for your product at *ni.com/calibration*.

National Instruments corporate headquarters is located at 11500 North Mopac Expressway, Austin, Texas, 78759-3504. National Instruments also has offices located around the world. For telephone support in the United States, create your service request at *ni.com/support* or dial 512 795 8248. For telephone support outside the United States, visit the *Worldwide Offices* section of *ni.com/niglobal* to access the branch office websites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Refer to the *NI Trademarks and Logo Guidelines* at ni.com/trademarks for information on National Instruments trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering National Instruments products/technology, refer to the appropriate location: **Help**»**Patents** in your software, the patents.txt file on your media, or the *National Instruments Patent Notice* at ni.com/patents. You can find information about end-user license agreements (EULAs) and third-party legal notices in the readme file for your NI product. Refer to the *Export Compliance Information* at ni.com/legal/export-compliance for the National Instruments global trade compliance policy and how to obtain relevant HTS codes, ECCNs, and other import/export data.

© 2013 National Instruments. All rights reserved.

376058A-01 Aug13

♻ This document represents a commitment
from National Instruments to the environment.

376058A-01 Aug13

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP