COMPREHENSIVE SERVICES

We offer competitive repair and calibration services, as well as easily accessible documentation and free downloadable resources.

SELL YOUR SURPLUS

We buy new, used, decommissioned, and surplus parts from every NI series. We work out the best solution to suit your individual needs.

Sell For Cash Get Credit Receive a Trade-In Deal

OBSOLETE NI HARDWARE IN STOCK & READY TO SHIP

We stock **New, New Surplus, Refurbished,** and **Reconditioned** NI Hardware.

APEX WAVES

Bridging the gap *between the manufacturer and your legacy test system.*

1-800-915-6216

www.apexwaves.com

sales@apexwaves.com

Request a Quote CLICK HERE *NI-7932*

FlexRIO™

NI-7931R/7932R/7935R User Manual

August 2015
375181B-01

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information
ni.com

Worldwide Offices
Visit ni.com/niglobal to access the branch office websites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

National Instruments Corporate Headquarters
11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

For further support information, refer to the *NI Services* appendix. To comment on National Instruments documentation, refer to the National Instruments website at ni.com/info and enter the Info Code feedback.

Legal Information

Limited Warranty

This document is provided ‘as is’ and is subject to being changed, without notice, in future editions. For the latest version, refer to `ni.com/manuals`. NI reviews this document carefully for technical accuracy; however, NI MAKES NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS.

NI warrants that its hardware products will be free of defects in materials and workmanship that cause the product to fail to substantially conform to the applicable NI published specifications for one (1) year from the date of invoice.

For a period of ninety (90) days from the date of invoice, NI warrants that (i) its software products will perform substantially in accordance with the applicable documentation provided with the software and (ii) the software media will be free from defects in materials and workmanship.

If NI receives notice of a defect or non-conformance during the applicable warranty period, NI will, in its discretion: (i) repair or replace the affected product, or (ii) refund the fees paid for the affected product. Repaired or replaced Hardware will be warranted for the remainder of the original warranty period or ninety (90) days, whichever is longer. If NI elects to repair or replace the product, NI may use new or refurbished parts or products that are equivalent to new in performance and reliability and are at least functionally equivalent to the original part or product.

You must obtain an RMA number from NI before returning any product to NI. NI reserves the right to charge a fee for examining and testing Hardware not covered by the Limited Warranty.

This Limited Warranty does not apply if the defect of the product resulted from improper or inadequate maintenance, installation, repair, or calibration (performed by a party other than NI); unauthorized modification; improper environment; use of an improper hardware or software key; improper use or operation outside of the specification for the product; improper voltages; accident, abuse, or neglect; or a hazard such as lightning, flood, or other act of nature.

THE REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND THE CUSTOMER’S SOLE REMEDIES, AND SHALL APPLY EVEN IF SUCH REMEDIES FAIL OF THEIR ESSENTIAL PURPOSE.

EXCEPT AS EXPRESSLY SET FORTH HEREIN, PRODUCTS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND AND NI DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, WITH RESPECT TO THE PRODUCTS, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, AND ANY WARRANTIES THAT MAY ARISE FROM USAGE OF TRADE OR COURSE OF DEALING. NI DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF OR THE RESULTS OF THE USE OF THE PRODUCTS IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NI DOES NOT WARRANT THAT THE OPERATION OF THE PRODUCTS WILL BE UNINTERRUPTED OR ERROR FREE.

In the event that you and NI have a separate signed written agreement with warranty terms covering the products, then the warranty terms in the separate agreement shall control.

Copyright

End-User License Agreements and Third-Party Legal Notices

You can find end-user license agreements (EULAs) and third-party legal notices in the following locations:

- Notices are located in the `<National Instruments>\_Legal Information` and `<National Instruments>` directories.
- EULAs are located in the `<National Instruments>\Shared\MDF\Legal\license` directory.
- Review `<National Instruments>\_Legal Information.txt` for information on including legal information in installers built with NI products.

U.S. Government Restricted Rights

If you are an agency, department, or other entity of the United States Government (“Government”), the use, duplication, reproduction, release, modification, disclosure or transfer of the technical data included in this manual is governed by the Restricted Rights provisions under Federal Acquisition Regulation 52.227-14 for civilian agencies and Defense Federal Acquisition Regulation Supplement Section 252.227-7014 and 252.227-7015 for military agencies.

Trademarks

Refer to the *NI Trademarks and Logo Guidelines* at `ni.com/trademarks` for more information on National Instruments trademarks.

ARM, Keil, and µVision are trademarks or registered of ARM Ltd or its subsidiaries.

LEGO, the LEGO logo, WEDO, and MINDSTORMS are trademarks of the LEGO Group.

TETRIX by Pitsco is a trademark of Pitsco, Inc.

FIELDBUS FOUNDATION™ and FOUNDATION™ are trademarks of the Fieldbus Foundation.

Patents

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.

Export Compliance Information

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.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

YOU ARE ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY AND RELIABILITY OF THE PRODUCTS WHENEVER THE PRODUCTS ARE INCORPORATED IN YOUR SYSTEM OR APPLICATION, INCLUDING THE APPROPRIATE DESIGN, PROCESS, AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

PRODUCTS ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING IN THE OPERATION OF NUCLEAR FACILITIES; AIRCRAFT NAVIGATION; AIR TRAFFIC CONTROL SYSTEMS; LIFE SAVING OR LIFE SUSTAINING SYSTEMS OR SUCH OTHER MEDICAL DEVICES; OR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, “HIGH-RISK USES”). FURTHER, PRUDENT STEPS MUST BE TAKEN TO PROTECT AGAINST FAILURES, INCLUDING PROVIDING BACK-UP AND SHUT-DOWN MECHANISMS. NI EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES.

Compliance

Electromagnetic Compatibility Information

This hardware has been tested and found to comply with the applicable regulatory requirements and limits for electromagnetic compatibility (EMC) as indicated in the hardware's Declaration of Conformity (DoC)[1]. These requirements and limits are designed to provide reasonable protection against harmful interference when the hardware is operated in the intended electromagnetic environment. In special cases, for example when either highly sensitive or noisy hardware is being used in close proximity, additional mitigation measures may have to be employed to minimize the potential for electromagnetic interference.

While this hardware is compliant with the applicable regulatory EMC requirements, there is no guarantee that interference will not occur in a particular installation. To minimize the potential for the hardware to cause interference to radio and television reception or to experience unacceptable performance degradation, install and use this hardware in strict accordance with the instructions in the hardware documentation and the DoC[1].

If this hardware does cause interference with licensed radio communications services or other nearby electronics, which can be determined by turning the hardware off and on, you are encouraged to try to correct the interference by one or more of the following measures:

- Reorient the antenna of the receiver (the device suffering interference).
- Relocate the transmitter (the device generating interference) with respect to the receiver.
- Plug the transmitter into a different outlet so that the transmitter and the receiver are on different branch circuits.

Some hardware may require the use of a metal, shielded enclosure (windowless version) to meet the EMC requirements for special EMC environments such as, for marine use or in heavy industrial areas. Refer to the hardware's user documentation and the DoC[1] for product installation requirements.

When the hardware is connected to a test object or to test leads, the system may become more sensitive to disturbances or may cause interference in the local electromagnetic environment.

Operation of this hardware in a residential area is likely to cause harmful interference. Users are required to correct the interference at their own expense or cease operation of the hardware.

Changes or modifications not expressly approved by National Instruments could void the user's right to operate the hardware under the local regulatory rules.

[1] The Declaration of Conformity (DoC) contains important EMC compliance information and instructions for the user or installer. To obtain the DoC for this product, visit `ni.com/certification`, search by model number or product line, and click the appropriate link in the Certification column.

Contents

Chapter 5
Programming the High-Speed Serial Ports

Chapter 6
Programming with the Real-Time Target

About This Manual

The *NI-7931R/7932R/7935R User Manual* describes how to use the NI-7931R, NI-7932R, and NI-7935R controllers for FlexRIO to develop high-performance embedded applications.

This manual provides detailed information about the electrical and mechanical requirements of component-level IP (CLIP) and LabVIEW FPGA design.

Related Documentation

The following documents contain information that you may find helpful as you read this manual.

Table 1. Documentation Overview

Document	Location	Description
Getting started guide for your controller for FlexRIO	Available from the Start menu (**Start»All Programs»National Instruments»NI FlexRIO**) and at ni.com/manuals.	Contains information about installing, configuring, and troubleshooting your controller for FlexRIO.
Specifications for your controller for FlexRIO		Contains specifications for your controller for FlexRIO.
FlexRIO Help		Contains information about the controller for FlexRIO front panel connectors and I/O, programming instructions, and adapter module component-level IP (CLIP).
LabVIEW High-Performance FPGA Developer's Guide	Available at ni.com/tutorial.	Summarizes the most effective techniques for optimizing throughput, latency, and FPGA resources when using the LabVIEW FPGA Module and the RIO hardware platform.

Table 1. Documentation Overview (Continued)

Document	Location	Description
FPGA Module Help	This document is a book within the *LabVIEW Help*. Access this document by navigating to **Start»All Programs»National Instruments»LabVIEW 201x» LabVIEW 201x Help**, or by searching for *FPGA Module Help* at ni.com/manuals. Browse to the **FPGA Module** book in the **Contents** tab for information about using the LabVIEW FPGA Module.	With the LabVIEW FPGA Module and LabVIEW, you can create VIs that run on National Instruments FPGA targets. The *Getting Started with the LabVIEW FPGA* book provides links to the top resources that you can use to get started with LabVIEW FPGA. The *Integrating Third-Party IP (FPGA Module)* book contains information about adding custom HDL code to your LabVIEW project.
Real-Time Module Help	This document is a book within the *LabVIEW Help*. Access this document by navigating to **Start»All Programs»National Instruments»LabVIEW 201x» LabVIEW 201x Help**, or by searching for *Real-Time Module Help* at ni.com/manuals. Browse to the **Real-Time Module** book in the **Contents** tab for information about using the LabVIEW Real-Time Module.	The Real-Time Module combines LabVIEW graphical programming with the power of a real-time operating system, enabling you to build real-time applications. Use this help to access information about real-time programming concepts, step-by-step instructions for using LabVIEW with the Real-Time Module, reference information about Real-Time Module VIs and functions, and information about LabVIEW features on real-time operating systems.
LabVIEW FPGA Module Release and Upgrade Notes	Available at ni.com/manuals. You can also view this document by selecting **Start»All Programs»National Instruments»LabVIEW» LabVIEW Manuals**.	Contains information about installing the LabVIEW FPGA Module, describes new features, and provides upgrade information**.**

Xilinx Documentation

Xilinx FPGA documentation provides information required for the successful development of your controller for FlexRIO. The following table provides a list of specific Xilinx documentation resources.

Table 2. Xilinx Documentation

Document	Document Part Number	Description
7 Series FPGAs Overview	DS180	Outlines the features and product selection of the Xilinx 7 series FPGAs, including Kintex-7 devices.
Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics	DS182	Contains the DC and AC switching characteristic specifications for the Kintex-7 FPGAs.
Vivado Design Suite: Release Notes, Installation, and Licensing	UG973	Provides an overview of the new release of the Vivado Design Suite, including information on new and changed features, installation requirements for the software, and licensing information.
High-Speed Serial I/O Made Simple: A Designer's Guide, with FPGA Applications	—	Recommended for users new to high-speed serial.
7 Series FPGAs GTX/GTH Transceivers User Guide	UG476	Technical reference describing the 7 series FPGAs GTX/GTH transceivers.
Vivado Design Suite User Guide: Using Constraints	UG903	Describes using Xilinx Design Constraints (XDC) in Vivado tools.

All Xilinx documentation can be found at www.xilinx.com.

Additional Resources

Table 3. FlexRIO Development Resources

Development Resource	Location	Description
FlexRIO website	ni.com/flexrio	Contains information about FlexRIO devices, application areas, and technical resources.
FlexRIO Instrument Development Library	https://decibel.ni.com/content/docs/DOC-15799	The FlexRIO Instrument Development Library is a set of host and FPGA code that provides FPGA capabilities commonly found in instruments such as acquisition engines, DRAM interfaces, and trigger logic, along with the associated host APIs.
LabVIEW examples	Available in NI Example Finder. In LabVIEW, click **Help»Find Examples»Hardware Input and Output»FlexRIO**.	Contains examples of how to run FPGA VIs and Host VIs on your device.
IPNet	ni.com/ipnet	Contains LabVIEW FPGA functions and intellectual property to share.

1 Before You Begin

This section contains information you need before developing high-performance embedded applications using the NI-7931R, NI-7932R, and NI-7935R devices.

Development Requirements

Successful system design with the NI-793xR devices may require knowledge in the following areas, depending on your application.

- Real-time programming
- VHDL code design
- LabVIEW and LabVIEW FPGA programming

If you are unfamiliar with any of these concepts, refer to the following table for a list of resources for learning the fundamentals required for NI-793xR development.

Table 1-1. Fundamentals Resources

Concept	Resources
Real-time programming	Real-time programming courses are available at ni.com/training. You can also refer to the *LabVIEW Real-Time Module Help* at ni.com/manuals.
VHDL code design	Some VHDL training or experience is required before implementing custom protocols with the high-speed serial transceivers. Do not attempt to develop Component-Level IP (CLIP) without VHDL knowledge. Refer to the *FlexRIO Help* for more information about CLIP.
LabVIEW and LabVIEW FPGA programming	LabVIEW and LabVIEW FPGA training are available at ni.com/training. You can also refer to the *NI LabVIEW High-Performance FPGA Developer's Guide*, available at ni.com/tutorials.

Xilinx Licensing Information

Refer to the *Xilinx Documentation* section of *About This Manual* for a list of Xilinx documentation that contains important Xilinx licensing information.

2

Mounting the NI-793xR

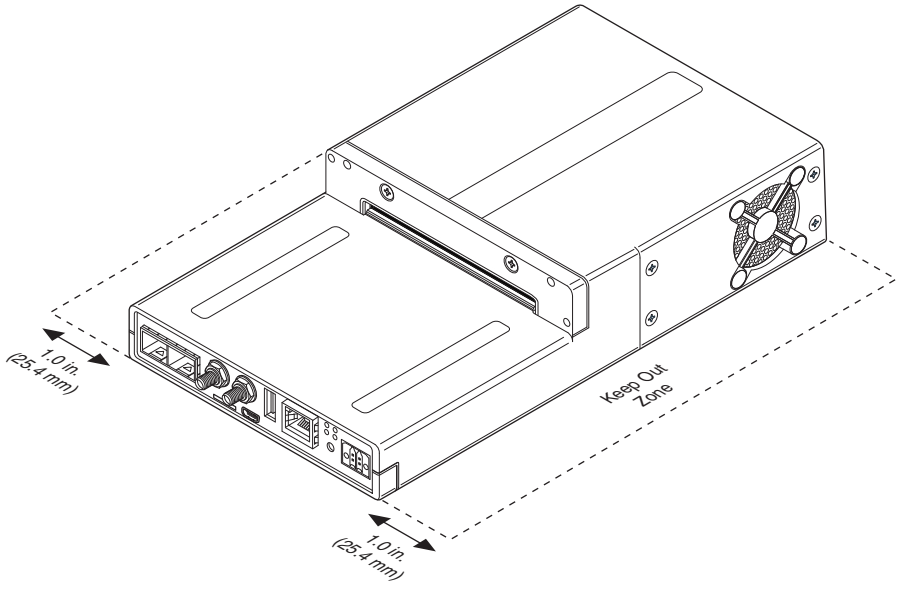
This section contains information about mounting the NI-793xR devices.

Note Before you begin mounting the NI-793xR, refer to the getting started guide for your NI-793xR for instructions about wiring power to the NI-793xR, powering on the NI-793xR, and connecting the NI-793xR to a host computer.

Caution The NI-793xR mounting orientation is not restricted; however, when mounting the NI-793xR upside-down, ensure that the FlexRIO adapter module is supported if you expect shock greater than 30 g/11 ms.

Caution In order to obtain the maximum allowable ambient temperature as specified in your device's specifications document, you must maintain at least 1 in. of clearance on either side of the NI-793xR. Refer to Figure 2-1 for fan clearance information.

Figure 2-1. Fan Clearance



You can mount the NI-793xR in a variety of configurations. The following table lists the ecommended mounting methods.

Table 2-1. Mounting Options

Method	Required Accessory Kit	NI Part Number
Direct mounting	—	—
Panel	Panel Mount Accessory Kit	784365-01

The following sections contain instructions for the mounting methods. Before using any of these mounting methods, record the serial number from the back of the device. You will be unable to read the serial number from the back of the device after you have mounted it.

Caution You must provide physical support for any FlexRIO adapter modules during the mounting process.

Mounting the NI-793xR Directly on a Flat Surface

For applications sensitive to shock and vibration, NI recommends mounting the device directly on a flat, rigid surface using the mounting holes in the device.

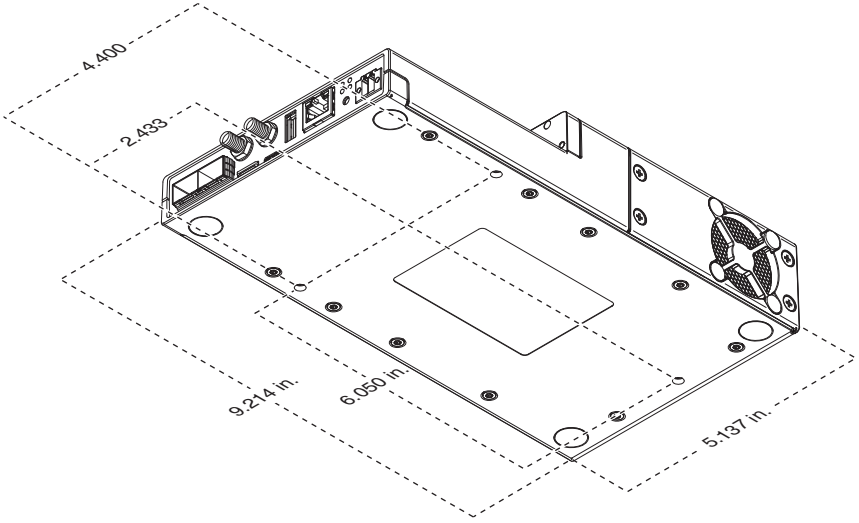
You will need the following items to mount the device directly on a flat surface:

- Three screws, M4, 7 mm + thickness of mounting surface

Complete the following steps to mount the device.

1. Use the dimensions shown in Figure 2-2 to drill the holes required for mounting the device.
2. Drill clearance holes 4.5 mm in diameter.
3. Align the device on the surface.
4. Fasten the device to the surface with the screws.

Figure 2-2. NI-793xR Dimensions

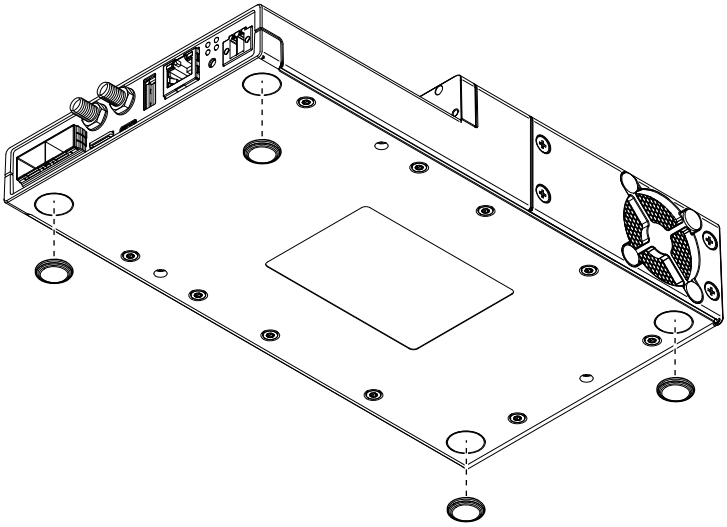


Installing the Rubber Feet

The NI-793xR ships with optional rubber feet. Install the rubber feet to the bottom of the device, as shown in Figure 2-3.

Caution Do not install rubber feet when directly mounting the NI-793xR. The rubber feet will prevent full contact between the unit and the mounting surface.

Figure 2-3. Installing the Rubber Feet



3

Hardware Architecture

This chapter contains information about the NI-793xR hardware architecture.

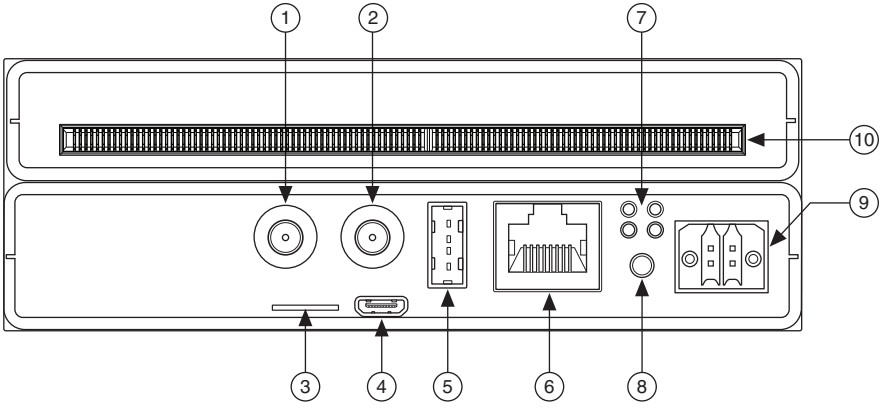
NI-7931R

The NI-7931R is an embedded FlexRIO controller with an embedded processor and reconfigurable FPGA.

Note The NI-7931R hardware does not require calibration.

The following figure shows the NI-7931R front panel connectors. For more information about the front panel connectors, refer to your device's specifications document and the *FlexRIO Help*. For information about connecting the device to a host computer, refer to the *NI-7931R Getting Started Guide*.

Figure 3-1. NI-7931R Front Panel Connectors

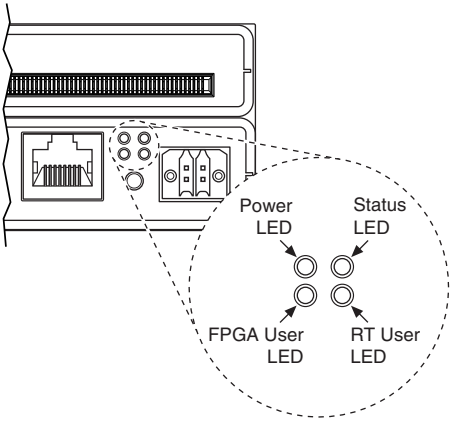


1	TRIG	6	1 Gb Ethernet
2	REF IN	7	LED indicators*
3	µSD card	8	Reset
4	USB device port	9	DC power source†
5	USB host	10	FlexRIO adapter module connector‡

* Refer to Figure 3-2 for LED placement.
† Refer to the *NI-7931R Getting Started Guide* for instructions about how to wire power to the NI-7931R.
‡ Refer to Figure 3-3 for the pinout.

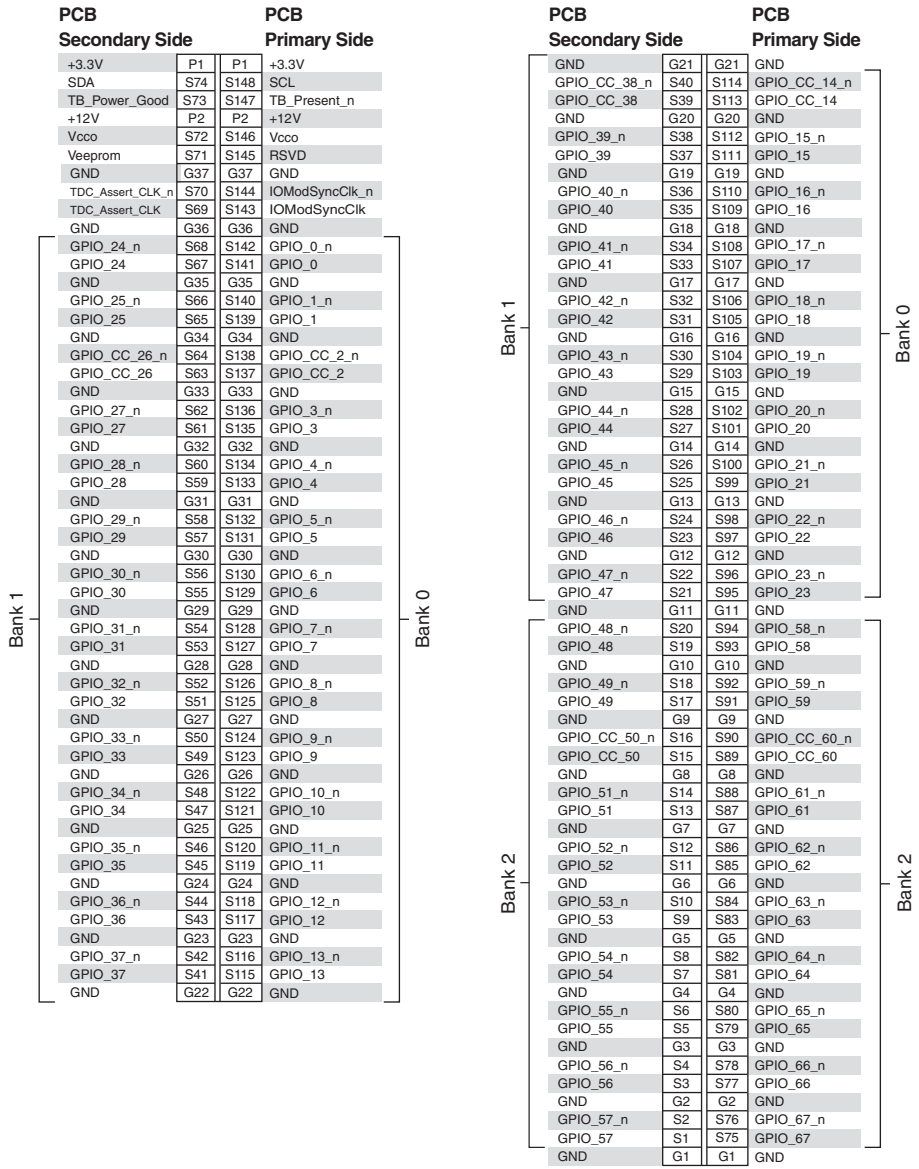
The following figure shows the NI-7931R LEDs in more detail.

Figure 3-2. NI-7931R LEDs



The following figure shows the available signals on the NI-7931R adapter module connector.

Figure 3-3. NI-7931R FPGA Connector Pinout



Note Pins S72 and S146 are shorted together.

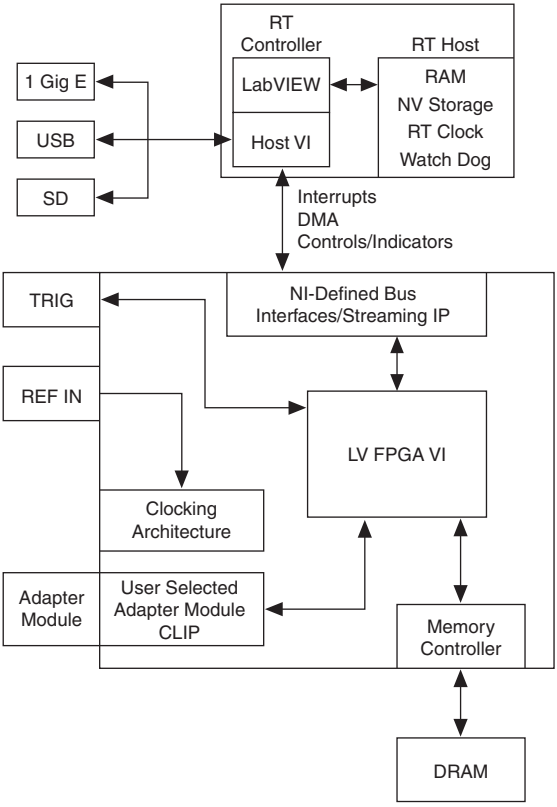
NI-7931R Key Features

The NI-7931R device includes the following key features. Refer to the *NI-7931R Specifications* for more details.

- Kintex-7 XC7K325T FPGA
- 2 GB onboard FPGA-accessible DRAM
- NI Linux Real-Time (32-bit) controller
- FPGA to host data transfer rates of 200 MB/s (single direction), 150 MB/s (bidirectional)
- Real-Time processor to USB external storage data transfer rates of 60 MB/s
- Real-Time processor to SD external storage data transfer rates of 12.0 MB/s (read), 9.0 MB/s (write)

The following figure illustrates the key components of the NI-7931R architecture.

Figure 3-4. NI-7931R Architecture Key Components



Clocking Architecture

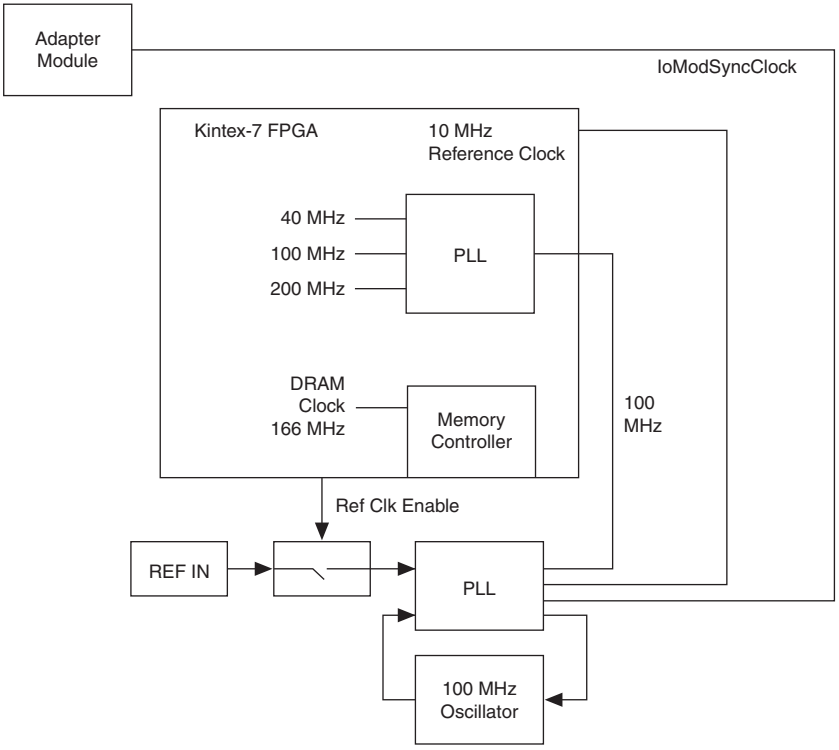
The NI-7931R device includes dedicated clocking hardware to provide a flexible clocking solution for your FlexRIO system. Refer to Chapter 4, *Developing with LabVIEW FPGA*, for information about configuring clocks with LabVIEW FPGA.

The NI-7931R clocking architecture includes the following clocks:

- 10 MHz Reference Clock
- 40 MHz Onboard Clock (default)
- 100 MHz Clock
- 200 MHz Clock
- DRAM Clock

The following figure illustrates the clocking circuitry on the NI-7931R.

Figure 3-5. NI-7931R Clocking Diagram



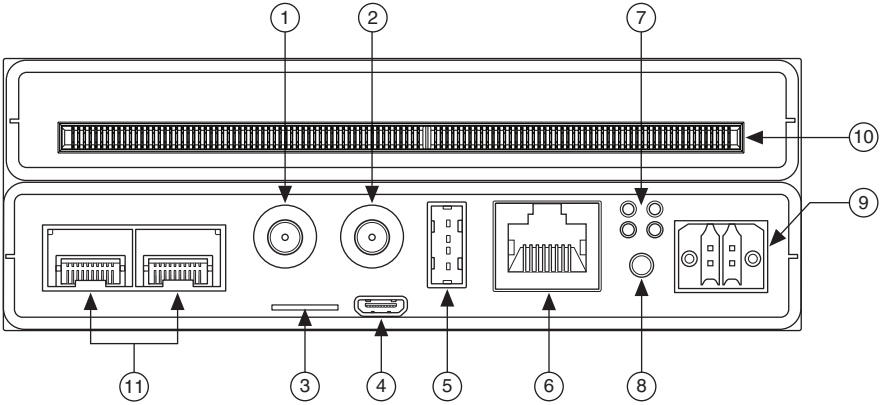
NI-7932R

The NI-7932R is an embedded FlexRIO controller with a LabVIEW Real-Time processor and reconfigurable FPGA. The NI-7932R includes a high-speed serial interface that uses Xilinx multi-gigabit transceiver (MGT) technology; you can reuse existing protocol IP that works with MGTs, or you can develop your own protocol IP. Refer to Chapter 5, *Programming the High-Speed Serial Ports*, for information about interfacing with MGTs via the SFP+ ports.

Note The NI-7932R hardware does not require calibration.

The following figure shows the NI-7932R front panel connectors. For more information about the front panel connectors, refer to your device's specifications document and the *FlexRIO Help*. For information about connecting the device to a host computer, refer to the *NI-7932R Getting Started Guide*.

Figure 3-6. NI-7932R Front Panel Connectors

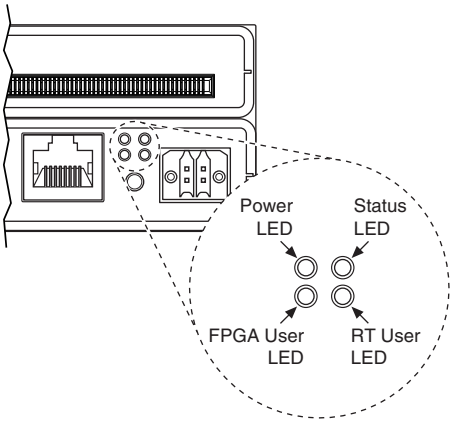


1 TRIG
2 REF IN
3 uSD card
4 USB device port
5 USB host
6 1 Gb Ethernet
7 LED indicators*
8 Reset
9 DC power source†
10 FlexRIO adapter module connector‡
11 SFP+ connectors

* Refer to Figure 3-7 for LED placement.
† Refer to the *NI-7932R Getting Started Guide* for instructions about how to wire power to the NI-7932R.
‡ Refer to Figure 3-8 for the pinout.

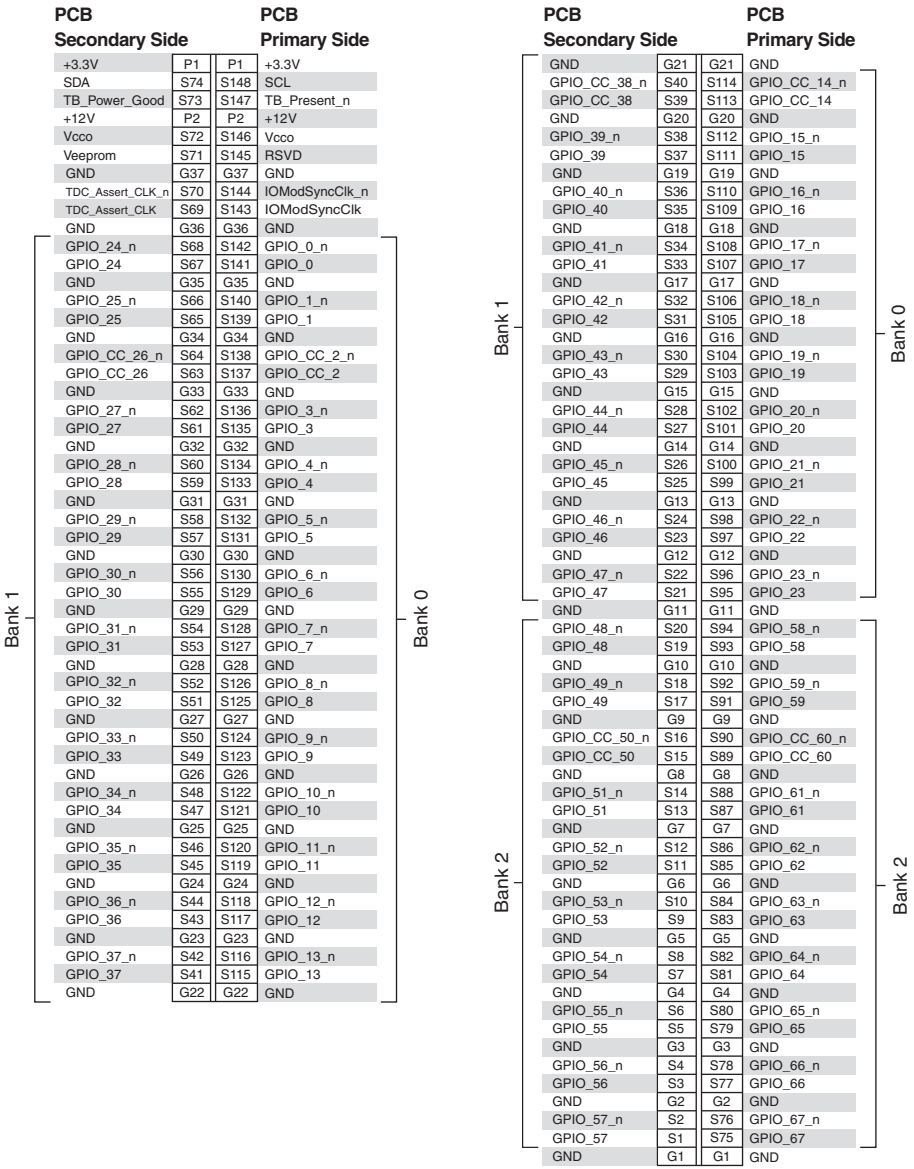
The following figure shows the NI-7932R LEDs in more detail.

Figure 3-7. NI-7932R LEDs



The following figure shows the available signals on the NI-7932R adapter module connector.

Figure 3-8. NI-7932R FPGA Connector Pinout



Note Pins S72 and S146 are shorted together.

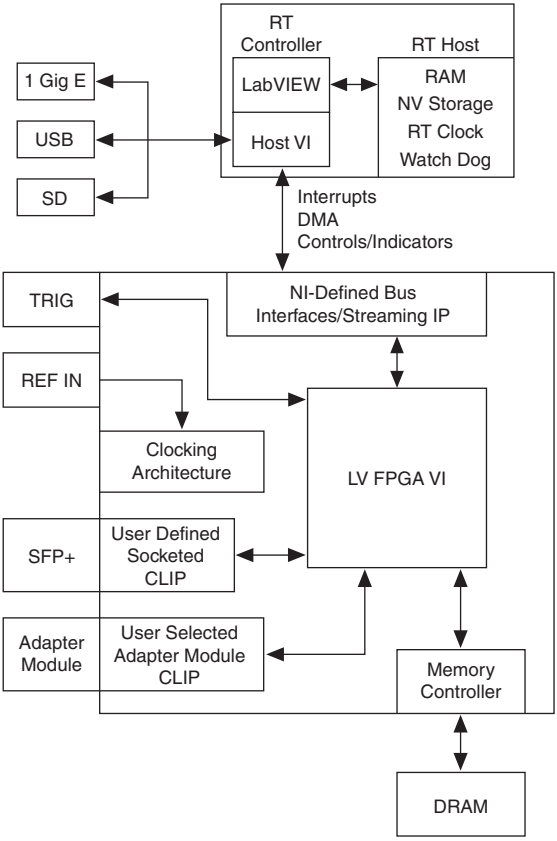
NI-7932R Key Features

The NI-7932R device includes the following key features. Refer to the *NI-7932R Specifications* for more details.

- SFP+ line rates of 3.125 Gbps, 6.25 Gbps, and 10.3125 Gbps
- Kintex-7 XC7K325T FPGA
- 2 GB onboard FPGA-accessible DRAM
- NI Linux Real-Time (32-bit) controller
- FPGA to host data transfer rates of 200 MB/s (single direction), 150 MB/s (bidirectional)
- Real-Time processor to USB external storage data transfer rates of 60 MB/s
- Real-Time processor to SD external storage data transfer rates of 12.0 MB/s (read), 9.0 MB/s (write)

The following figure illustrates the key components of the NI-7932R architecture.

Figure 3-9. NI-7932R Architecture Key Components



Clocking Architecture

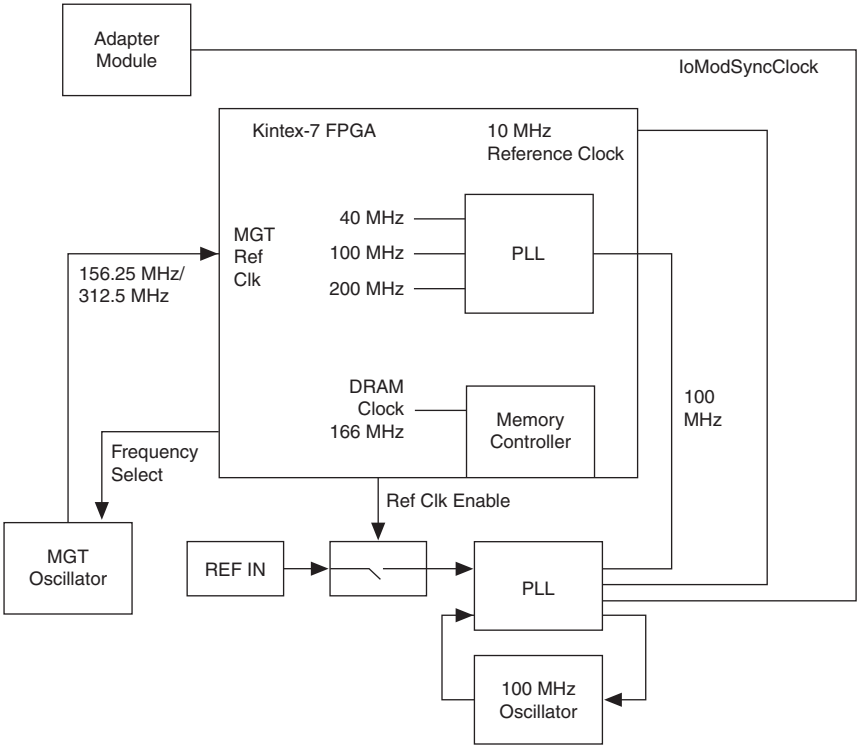
The NI-7932R device includes dedicated clocking hardware to provide a flexible clocking solution for your FlexRIO system. Refer to Chapter 4, *Developing with LabVIEW FPGA*, for information about configuring clocks with LabVIEW FPGA.

The NI-7932R clocking architecture includes the following clocks:

- 10 MHz Reference Clock
- 40 MHz Onboard Clock (default)
- 100 MHz Clock
- 156.25 MHz Clock/312.5 MHz MGT Clock[1]
- 200 MHz Clock
- DRAM Clock

The following figure illustrates the clocking circuitry on the NI-7932R.

Figure 3-10. NI-7932R Clocking Diagram



[1] This clock is user-selectable for either 156.25 MHz or 312.5 MHz.

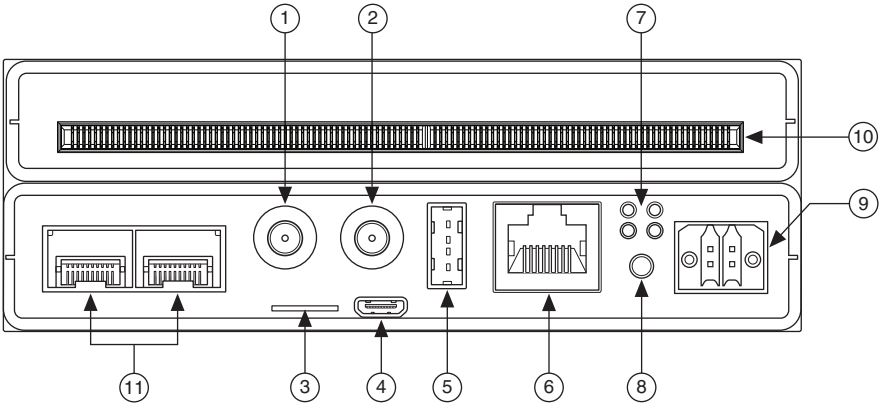
NI-7935R

The NI-7935R is an embedded FlexRIO controller with a LabVIEW Real-Time processor and reconfigurable FPGA. The NI-7935R includes a high-speed serial interface that uses Xilinx multi-gigabit transceiver (MGT) technology; you can reuse existing protocol IP that works with MGTs, or you can develop your own protocol IP. Refer to Chapter 5, *Programming the High-Speed Serial Ports*, for information about interfacing with MGTs via the SFP+ ports.

Note The NI-7935R hardware does not require calibration.

The following figure shows the NI-7935R front panel connectors. For more information about the front panel connectors, refer to your device's specifications document and the *FlexRIO Help*. For information about connecting the device to a host computer, refer to the *NI-7935R Getting Started Guide*.

Figure 3-11. NI-7935R Front Panel Connectors

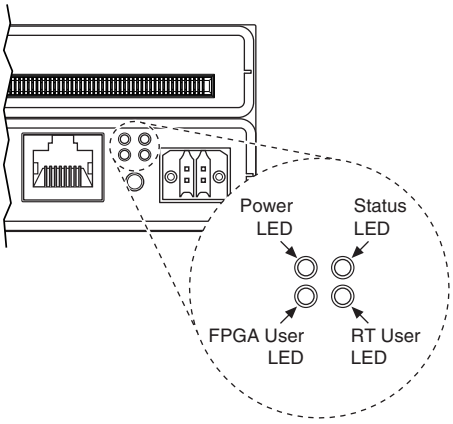


1 TRIG
2 REF IN
3 uSD card
4 USB device port
5 USB host
6 1 GB Ethernet
7 LED indicators*
8 Reset
9 DC power source†
10 FlexRIO adapter module connector‡
11 SFP+ connectors

* Refer to Figure 3-12 for LED placement.
† Refer to the *NI-7935R Getting Started Guide* for instructions about how to wire power to the NI-7935R.
‡ Refer to Figure 3-13 for the pinout.

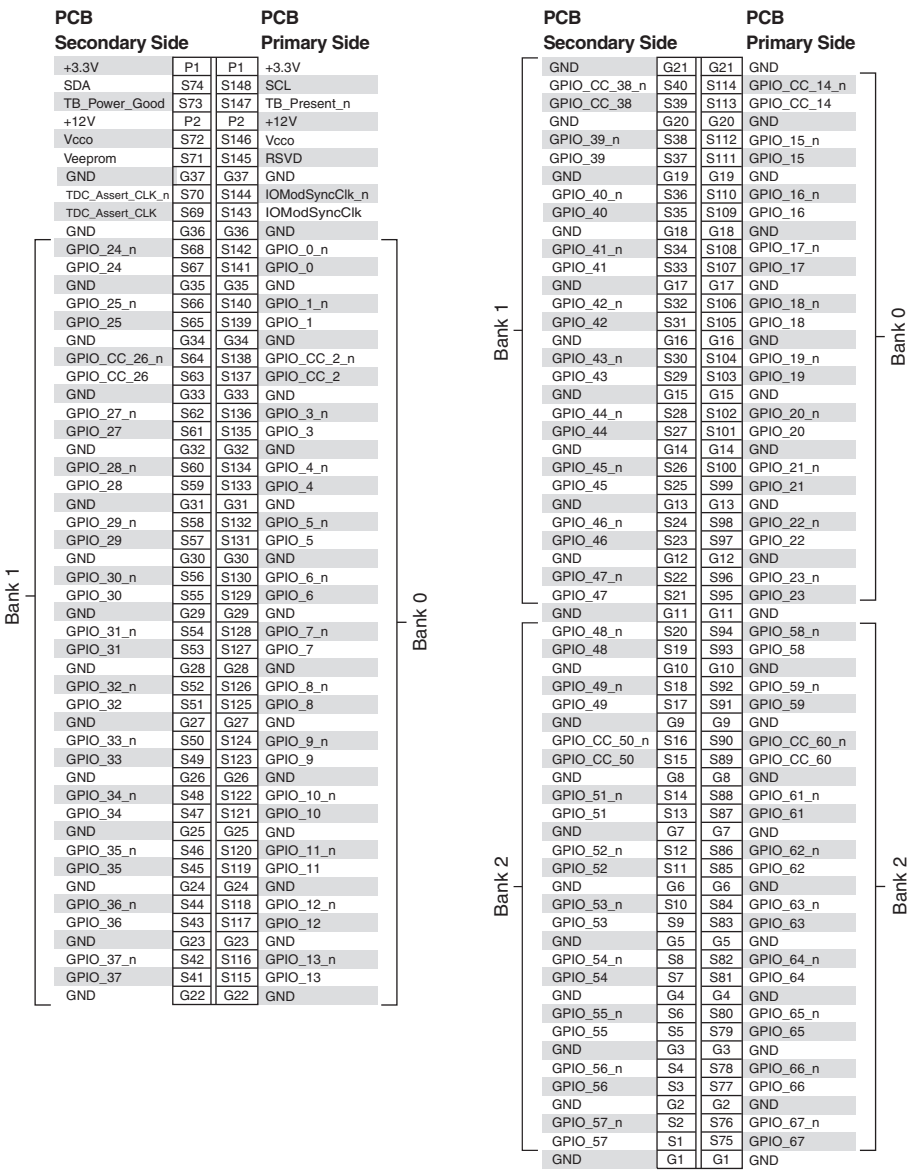
The following figure shows the NI-7935R LEDs in more detail.

Figure 3-12. NI-7935R LEDs



The following figure shows the available signals on the NI-7935R adapter module connector.

Figure 3-13. NI-7935R FPGA Connector Pinout



Note Pins S72 and S146 are shorted together.

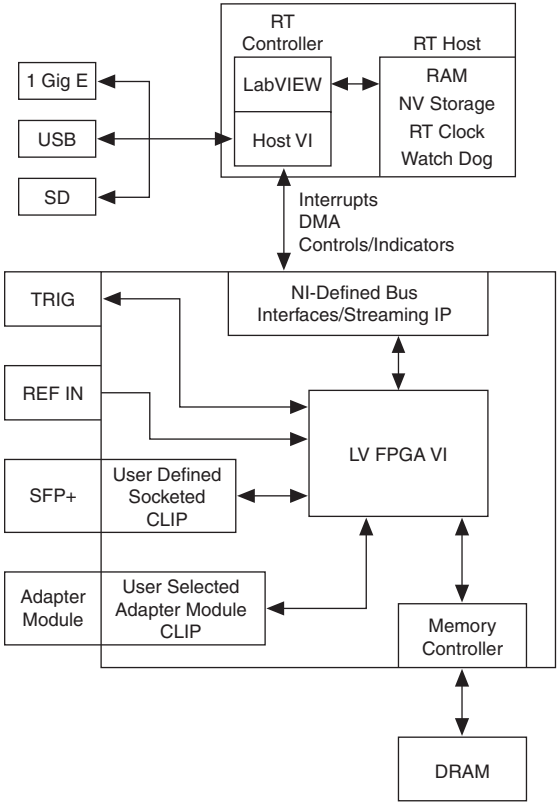
NI-7935R Key Features

The NI-7935R device includes the following key features. Refer to the *NI-7935R Specifications* for more details.

- SFP+ line rates of 3.125 Gbps, 6.25 Gbps, and 10.3125 Gbps
- Kintex-7 XC7K410T FPGA
- 2 GB onboard FPGA-accessible DRAM
- NI Linux Real-Time (32-bit) controller
- FPGA to host data transfer rates of 200 MB/s (single direction), 150 MB/s (bidirectional)
- Real-Time processor to USB external storage data transfer rates of 60 MB/s
- Real-Time processor to SD external storage data transfer rates of 12.0 MB/s (read), 9.0 MB/s (write)

The following figure illustrates the key components of the NI-7935R architecture.

Figure 3-14. NI-7935R Architecture Key Components



Clocking Architecture

The NI-7935R device includes dedicated clocking hardware to provide a flexible clocking solution for your FlexRIO system. Refer to Chapter 4, *Developing with LabVIEW FPGA*, for information about configuring clocks with LabVIEW FPGA.

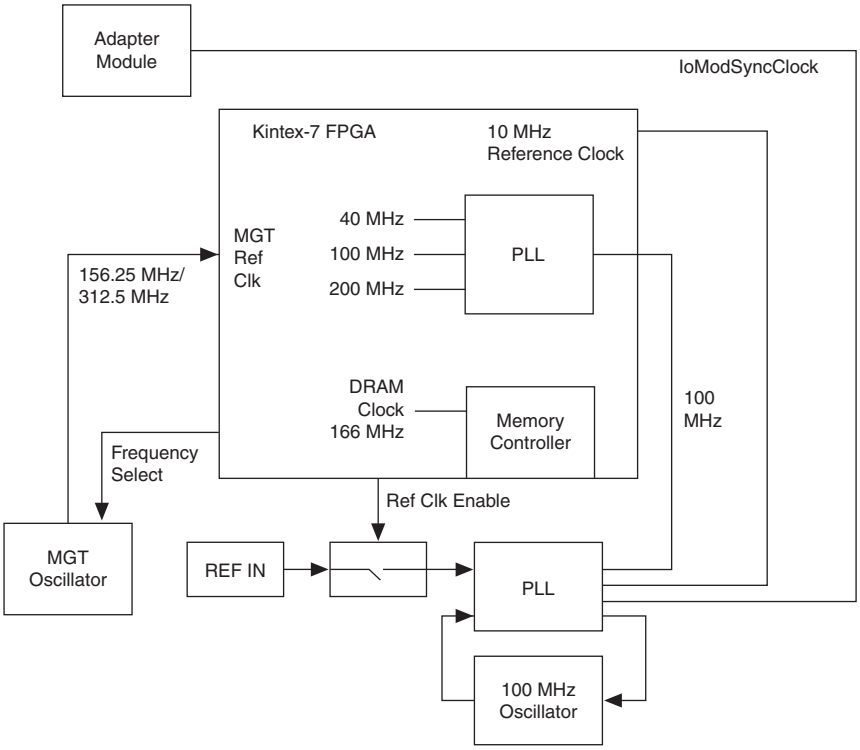
The NI-7935R clocking architecture includes the following clocks:

- 10 MHz Reference Clock
- 40 MHz Onboard Clock (default)
- 100 MHz Clock
- 156.25 MHz Clock/312.5 MHz MGT Clock[1]
- 200 MHz Clock
- DRAM Clock

[1] This clock is user-selectable for either 156.25 MHz or 312.5 MHz.

The following figure illustrates the clocking circuitry on the NI-7935R.

Figure 3-15. NI-7935R Clocking Diagram



4

Developing with LabVIEW FPGA

This chapter contains information about developing your NI-793xR-based project with LabVIEW FPGA. LabVIEW FPGA provides FPGA target support, configuration for clocking and routing, and interfacing with LabVIEW on your host computer for a fully integrated development experience.

Refer to the *NI LabVIEW High-Performance FPGA Developer's Guide* for information about techniques to optimize throughput, latency, and FPGA resources. Refer to the *Related Documentation* section of this manual for a full list of LabVIEW FPGA documentation that you may find helpful as you develop your application.

Developing with LabVIEW FPGA

For information about installing FlexRIO Support, installing the NI-793xR, and installing an adapter module, refer to the getting started guide for your NI-793xR device.

Adding the NI-793xR to a LabVIEW Project

1. Launch LabVIEW. The LabVIEW Getting Started window appears.
2. Click **Create Project** or open an existing project.
3. Right-click the project root in the Project Explorer window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
   a. If the hardware is connected to the host, select **Existing target or device**. Select your device under **Real-Time FlexRIO** and click **OK**.
   b. If the hardware is not connected to the host, select **New target or device**. Select your device under **Real-Time FlexRIO** and click **OK**.

Adding an Adapter Module to the Target

Skip this section if you are not using an adapter module.

1. Expand the FPGA target by clicking the + button, then right-click **IO Module** and select **Properties**.
2. Select the **General** category and check the **Enable IO Module** box.
3. Select your adapter module from the **IO Modules** list, and select the CLIP you want to use from the **Component Level IP** box.
4. Click **OK**.

Adding Items to the NI-793xR Target

You can add new or existing FPGA VIs, FPGA I/O items, FPGA FIFO, or FPGA clocks to the NI-793xR target in the **Project Explorer** window. You can also use folders to organize items under the FPGA target in the **Project Explorer** window. You might use the folder option for organizing items if you intend to use multiple FPGA I/O items.

Complete the following steps to add an item to the NI-793xR target in the **Project Explorer** window.

1. Right-click the FPGA target and select **New** from the shortcut menu to add a new item, such as a VI, FPGA I/O item, or folder. Then select the item you want to add to the project. The item appears in the **Project Explorer** window under the FPGA target.
2. Double-click the new item in the **Project Explorer** window to edit or configure the item. If you added an FPGA base clock, right-click the FPGA base clock and select **Properties** from the shortcut menu to configure the clock.

Note You also can drag and drop existing items into the FPGA target in the **Project Explorer** window.

Adding NI-793xR Target I/O

Complete the following steps to add target I/O for the NI-793xR and to access signals from any instantiated CLIP on the block diagram:

1. Place an FPGA I/O node on the FPGA target block diagram. The FPGA I/O node is located on the palette under **Functions»FPGA I/O»FPGA I/O Node**.
2. Right-click the FPGA I/O node and select **Add New FPGA I/O**.
3. In the **New FPGA I/O** dialog box, select resources under **Available Resources** and add them to **New FPGA I/O** using the right arrow button.
4. To remove a resource, select the resource under **New FPGA I/O** and click the left arrow button.
5. Click **OK**.

Configuring a 10 MHz Reference Clock

Note By default, the NI-793xR derives its 10 MHz Reference Clock from an internal oscillator.

To source an external 10 MHz reference clock from the REF IN front panel, complete the following steps.

1. Once the target has been added to the project, expand the FPGA target and right-click Reference Clock Source (Onboard 10 MHz Clock) and select **Properties**.
2. Enable the **Use the external front panel clock input as the reference clock** checkbox.
3. Click **OK**.

For information about optimizing your LabVIEW FPGA code for throughput, latency, or resource utilization, refer to the *High-Performance LabVIEW FPGA Developer's Guide*.

Auto-loading Bitfiles on Power-up

You can configure the NI-793xR to auto-load a bitfile on power-up, or you can use a startup executable on the Real-Time controller to load a specific bitfile when the device powers up. Complete the following steps to auto-load a bitfile on the NI-793xR.

1. In MAX, expand **Remote Systems** and select your NI-793xR target from the list of Real-Time targets.
2. Expand **Devices and Interfaces** and select the NI-793xR FPGA target.

Note You must select the NI-793xR FPGA target under **Devices and Interfaces**. Selecting the NI-793xR target directly under **Remote Systems** updates the Real-Time controller firmware and not the FPGA firmware.

3. Navigate to your bitfile and select **Open**.
4. In the **Update Firmware** window, select **Begin Update**. This process may take a few minutes to complete.
5. Restart your controller.

In addition to using MAX to auto-load bitfiles for FPGA, you can use the system configuration API to programmatically set the bitfile that is auto-loaded on power-up.

Interactive Front Panel Communication

Use interactive front panel communication to communicate with an FPGA VI running on an FPGA target with no additional programming. With interactive front panel communication, the host computer displays the FPGA VI front panel window and the FPGA target executes the FPGA VI block diagram.

The LabVIEW front panel window communicates with the FPGA target block diagram through the controls and indicators. You can communicate with an FPGA target connected directly to host computer or connected to a remote system over the network. As the FPGA target block diagram continues to run, the host computer updates values on the FPGA VI front panel window as often as possible. The execution rate of the FPGA VI is not affected by communication with the host computer. However, the front panel data you share during interactive front panel communication is not deterministic.

Use interactive front panel communication between the FPGA target and the host computer to control and test VIs running on the FPGA target. After downloading and running the FPGA VI, keep LabVIEW open on the host computer to display and interact with the front panel window of the FPGA VI.

During interactive front panel communication, you cannot use LabVIEW debugging tools, including probes, execution highlighting, breakpoints, and single-stepping. To identify errors before you compile, download, and run the FPGA VI on the FPGA target, consider using a test bench.

Note You cannot use interactive front panel communication while the FPGA is configured to execute on a third-party simulator. You can either use a host VI to execute the FPGA VI or change the execution mode of the FPGA target by right-clicking the FPGA target in the Project Explorer window and selecting **Select Execution Mode**.

Using the NI Common Instrument Design Libraries

NI provides instrument design libraries that you can use to create application-specific instrumentation designs for NI-793xR devices. The following sections provide an overview of the instrument design libraries. The instrument design libraries are located at `<LVDir>\instr.lib\_niInstr`. For information about the VIs in each instrument design library, refer to the Programming section of the *FlexRIO Help*.

Using niInstr Instruction Framework

Use the Instruction Framework instrument design library to build a communication network in LabVIEW FPGA. Standard communication methods, such as using controls and indicators to pass information between the host and the FPGA, may not scale well for large applications. Use the Instruction Framework to provide a scalable communication framework that larger applications may require, at the cost of increased complexity. Certain instrument design libraries require the use of the Instruction Framework.

Streaming Overview

The Streaming Instrument Design Library provides a consistent mechanism to handle both finite and continuous transfer streams. It provides stream monitoring and handshaking. It contains VIs for both the Host and FPGA.

CLIP Adapters Overview

The CLIP Adapters instrument design library includes AXI4-Lite and AXI4-Stream wrappers. These wrappers implement protocol timing and signaling into simple reader or writer endpoints that present 4-wire handshaking to the diagram. This handshaking allows for easier transition to many FPGA features without the need to implement this state logic on your own.

Data Trigger Overview

This instrument design library can be used to generate a trigger on an input signal under various conditions. The triggers produced by this library are typically consumed by the acquisition block in order to determine when to start and stop acquiring data.

This library supports multiple trigger types, data types, and samples per cycle.

Basic Elements Overview

This instrument design library contains several low-level elements, such as edge detectors, latches, and FIFOs. Using this library can be beneficial when developing new FPGA logic for your software-designed instrument. These basic elements are used in other instrument design libraries and the sample projects for your device.

Memory Overview

Use the Memory instrument design library to access DRAM and BRAM on the device in a consistent manner. This library provides a basic read and write interface to DRAM and BRAM.

In addition to the basic memory interface, you can use this instrument design library to reset the DRAM or BRAM. When memory read operations are posted to memory, there is some amount of latency before the associated data is retrieved from memory and presented to the FPGA diagram. Furthermore, multiple read operations can be queued up at once. You can use the Memory instrument design library to reset those queued memory operations.

This instrument design library also adds support for arbitration between the read and write ports of DRAM.

Compiling LabVIEW FPGA VIs

You may need to purchase and install additional licenses to compile FPGA designs that incorporate licensed cores from Xilinx or third-party IP vendors. Refer to *UG 973: Vivado Design Suite: Release Notes, Installation, and Licensing* at `xilinx.com` for information about managing licenses.

The NI-793xR targets include large FPGA devices that require a 64-bit compile worker. Refer to the *FlexRIO Support Readme* for more information about what platforms to use to compile bitfiles.

You cannot add additional licenses to remote compile workers in the NI LabVIEW FPGA Compile Cloud Service. You cannot use NI LabVIEW FPGA Compile Cloud Service to compile designs that incorporate Xilinx or other third-party licensed cores.

Download, Reset, and Run Side Effects in the LabVIEW FPGA Host Interface

When the NI-793xR FPGA loads, it performs a power-on self-configuration sequence that configures various on-board hardware. This configuration occurs at the following times:

- At device power-up after the bitfile loads.
- At the first time **Run** is called after a new bitfile is downloaded and the bitfile is not set to **Run on Load**.
- When **Run** is called after **Reset**.

For more information about **Run**, **Reset**, and other Invoke methods, refer to the *LabVIEW FPGA Module Help*.

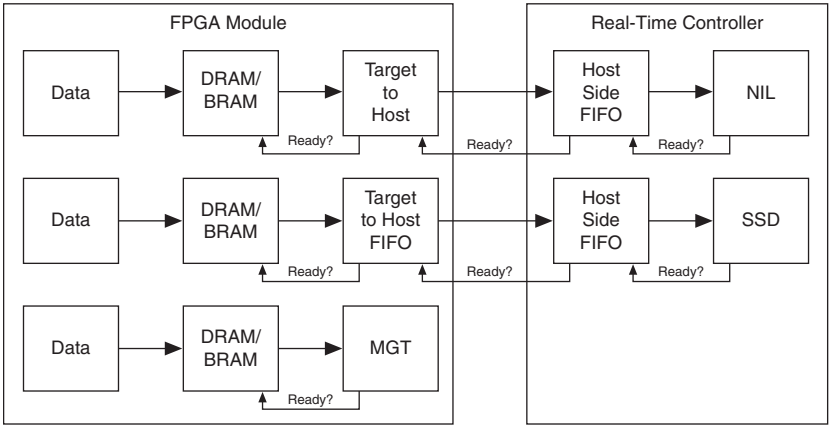
Note When self-configuration executes, the clocking configuration enters an indeterminate state. When the clocking configuration is in an indeterminate state, you cannot rely on clocking stability from the clocking and routing hardware on the NI-793xR.

Streaming

Flow Control

Any application that logs information must have rigorous flow control because the FlexRIO adapter module can generate far more data than the application nodes can process. The FPGA-to-Host FIFO uses Ready for Input signals to communicate to the DRAM whether it is ready to process more data. The following figure demonstrates how you can implement flow control on an NI-793xR target.

Figure 4-1. Host-Side FIFO to FPGA Flow Control



For information about data transfer rates, refer to the following sections:

- *NI-7931R Key Features*
- *NI-7932R Key Features*
- *NI-7935R Key Features*

DMA Streaming

The NI-793xR devices support both host-to-target streaming and target-to-host streaming through DMA channels that connect the host to your target. Use DMA streaming to allow the maximum throughput of data from your host application to be streamed to the target at high rates of speed.

The NI-793xR provides up to 16 DMA channels that can be accessed by your Host. These channels can be used in a variety of ways to meet your application's needs. The total overall bandwidth of the device limits your DMA use, whether you use 1 DMA channel or 16.

The maximum width of a DMA channel is 256 bits. To use the full width of the DMA channel to achieve maximum throughput, create a data construct that matches the 256-bit data width of the DMA channel. You can either create a cluster that contains 4 U64s, or an array of 4 U64s. To use an array, the FIFO must be configured to return multiple elements per read/write. You can also write up to 1,024 bits at a time from LabVIEW FPGA, and the Ready for Input connection throttles the connection to the FIFO to prevent overflow.

Theoretically, DMA throughput is maximized and is most consistent when the DMA FIFO buffer is sized as large as possible to absorb variations in the readiness of the host memory. However, sizing the FIFO larger consumes block RAM resources on the FPGA and increases the timing pressure on the FIFO. NI recommends making the FIFO as large as you can successfully compile with, in order to sustain throughput through the PCIe bus to and from host memory. You can change the size of the FIFO by configuring the Requested Number of Elements for the FIFO in the project properties. You can validate the DMA sizing through benchmarking, and you can use VIs in the Streaming Design Library to monitor the health of a FIFO.

For more detailed information about using DMA, DMA best practices, and how to make design decisions on how to implement DMA in your application, refer to the *Transferring Data Using Direct Memory Access* topic of the *LabVIEW FPGA Help*.

Total throughput depends on the SCTL rate from the FPGA that is reading or writing the DMA channels. The data throughput is calculated by the following equation:

(Data Width × Samples per Cycle) × Number of DMA FIFOs × SCTL Clock Rate = Data Throughput

Note The total data throughput cannot exceed the maximum data specification for your device. Refer to the Specifications document for your device for information about data throughput limits.

Note The number of array elements fed into the DMA FIFO from the Host can limit the maximum throughput for your application. Use large array subsets and set your FIFO depths to be deep enough to sustain high throughput.

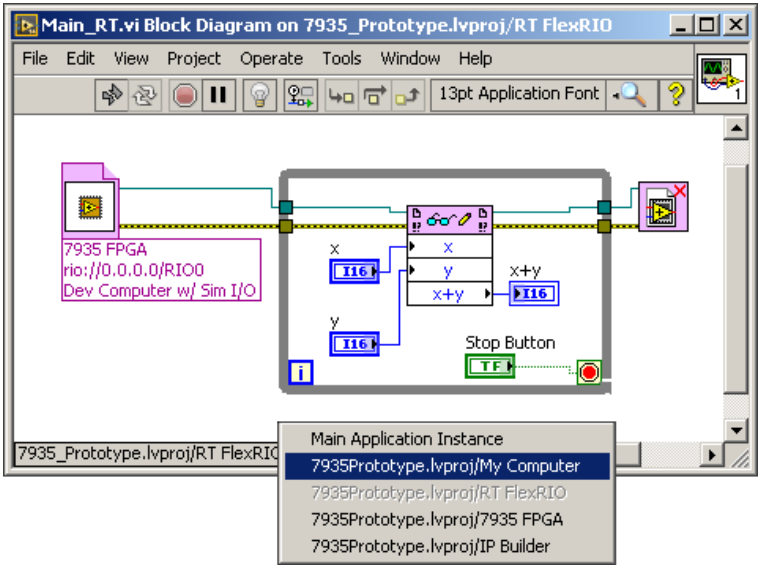
Simulating FPGA Behavior

You can simulate an FPGA VI that has been added to an NI-793xR target; however, you cannot open a reference to the simulated FPGA VI from the NI-793xR target. Instead, you must open a reference to the simulated FPGA VI by changing the application instance to My Computer. You can select the application instance for a VI by using the application instance shortcut menu.

Note If you attempt to open a reference to a simulated FPGA target from an NI-793xR target, a broken run arrow and an error message appear in your VI.

Complete the following steps to change the application instance for your simulated FPGA VI.

1. Navigate to the bottom left corner of the front panel window or block diagram. The application instance selection shortcut menu displays the current application instance of the VI.
2. Right-click the shortcut menu and select the My Computer instance in which to run the VI, as shown in the following figure.



Note Selecting a new application instance reopens the VI in the selected application instance. The VI also remains open in the original application instance.

You also can use the `Application:Default:Application` property to return the default application reference programmatically. Use the Application property to open the target application instance programmatically.

5

Programming the High-Speed Serial Ports

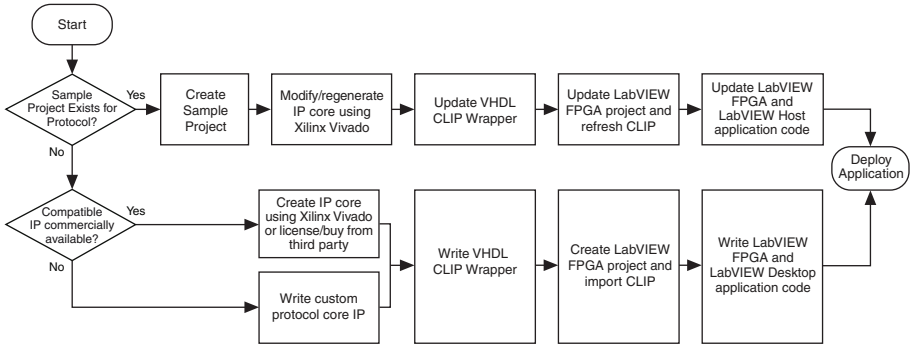
This chapter provides information about programming the multi-gigabit transceivers (MGTs) for the NI-7932R and NI-7935R, including information about creating socketed CLIP and using LabVIEW.

Note The NI-7931R does not have MGTs or high-speed serial ports.

Development Flow

Refer to the following diagram for an overview of the NI-793xR development process for implementing a high-speed serial protocol.

Figure 5-1. NI-793xR Development Process



If the sample project code is sufficient for your application, you do not have to modify the IP core, update the VHDL CLIP wrapper, or refresh the CLIP.

Developing MGT Socketed CLIP

This section provides steps for creating socketed CLIP for use with your application. Socketed CLIP provides the following functionality:

- Allows you to insert HDL IP into an FPGA target, enabling VHDL code to communicate directly with an FPGA VI.
- Allows the CLIP to communicate directly with circuitry external to the FPGA.
- Allows your IP to communicate directly with both the FPGA VI and the external adapter module connector interface.

Socketed CLIP Architecture

Figure 5-2 shows an overview of the NI-7932R socketed CLIP interface. Figure 5-3 shows an overview of the NI-7935R socketed CLIP interface.

Figure 5-2. NI-7932R Socketed CLIP Architecture

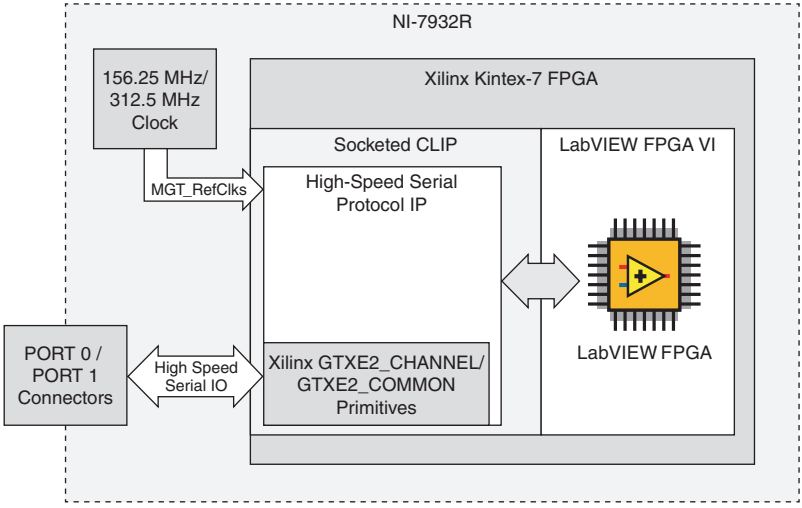
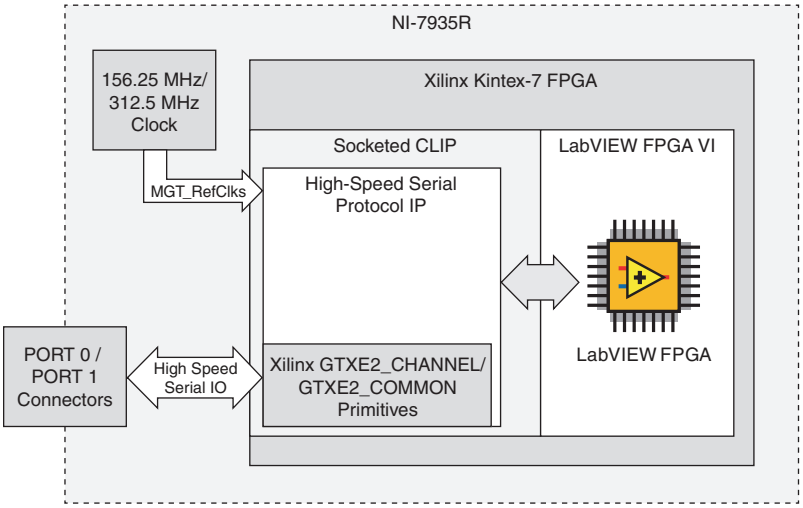


Figure 5-3. NI-7935R Socketed CLIP Architecture



Accessing the Xilinx Vivado Tools

Complete the following steps to run Xilinx Vivado:

1. If you installed Xilinx Vivado separately from LabVIEW FPGA, use this version. Otherwise, LabVIEW FPGA installs LabVIEW FPGA Xilinx Tools.

Note If Vivado is installed by LabVIEW FPGA, it does not appear in **Programs and Features**.

2. Open the Xilinx Vivado Tool directory by navigating to `C:\NIFPGA\programs\VivadoXXXX_Y`, where `XXXX` and `Y` refer to the Xilinx Vivado tool versions. For example, <VIVADO_DIR> version 2013.4 is located at `C:\NIFPGA\programs\Vivado2013_4`.
3. Run the Xilinx Vivado batch file: `<XilinxVivadoDir>\bin\vivado.bat`.

   You may receive the following warning when launching Vivado.

   ```
   Your XILINX_EDK environment variable is undefined. You may not be able to run some features properly. Please set up your XILINX_EDK environment to get full functionality.
   ```

   This error message is expected. You can ignore the error message if you are not using the Xilinx Embedded Development Kit (EDK). The EDK is not required for development with the NI-793xR.
4. Click **New Project** and follow the instructions in the wizard.

Generating an IP Core from the Xilinx Vivado IP Catalog

You may need to purchase and install additional licenses to generate some protocol IP core from Xilinx or third-party IP vendors. Refer to *UG 973: Vivado Design Suite: Release Notes, Installation, and Licensing* at `xilinx.com` for information about managing licenses.

Complete the following steps to create a Xilinx Vivado project:

1. Refer to the *Xilinx Documentation* section of this manual for information about licensing before creating a Xilinx Vivado project.
2. Launch the Xilinx Vivado IP catalog.
   a. Select **Manage IP** on the Vivado start screen.
   b. Locate the appropriate IP core to launch the configuration dialog. For example, the Aurora 64B66B IP core is located in **Communication and Networking»Serial Interfaces»Aurora 64B66B**.
3. Select the IP core settings. NI recommends that you select AXI4-Stream for high-speed data streams when possible.

Note NI does not recommend selecting AXI4-Lite for DRP accesses in the Xilinx IP cores because compatibility with LabVIEW FPGA AXI4-Lite adapters cannot be guaranteed. Refer to the Aurora sample projects for an example of how to use the LabVIEW FPGA AXI4-Lite adapters to connect to DRP within the CLIP.

Modifying Third-Party IP Core Logic

If you modify a third-party IP core for your high-speed serial protocol, consult the *Xilinx Product Guide* for the IP you are using before attempting to make any modifications.

Adhere to the following guidelines when modifying third-party IP core logic:

- Ensure all clocks are connected.
- Ensure AXI4-Lite management signals are connected correctly to the Xilinx DRP signals on the GTXE2_CHANNEL and GTXE2_COMMON primitives.
- Select **Include Shared Logic in example design** in the IP wizard to access various resources outside of the IP core logic, such as MGT_RefClk input buffers and QPLL wrappers.

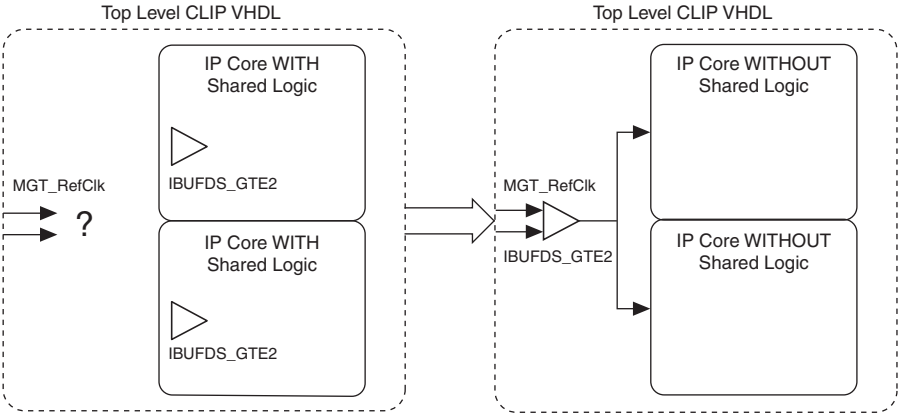
The following examples explain the differences in how the IBUFDS_GTE2 resource is exposed with and without the **Include Shared Logic in example design** option.

- Option 1: Include the IBUFDS_GTE2 input buffer primitive inside the core by selecting **Include Shared Logic in core** in the IP wizard. The image on the left in Figure 5-4 shows this option.
- Option 2: Instantiate a single IBUFDS_GTE2 input buffer in your top level CLIP VHDL, connect its output signal to both cores, and select **Include Shared Logic in example design** in the IP wizard. The image on the right in Figure 5-4 shows this option.

Note Do not modify the IP core unless you understand the required reference clock(s) and clocking resources.

The following figure shows the difference between the top-level CLIP VHDL with shared logic in the core (left) and without shared logic (right).

Figure 5-4. Top-Level CLIP VHDL and Shared Logic

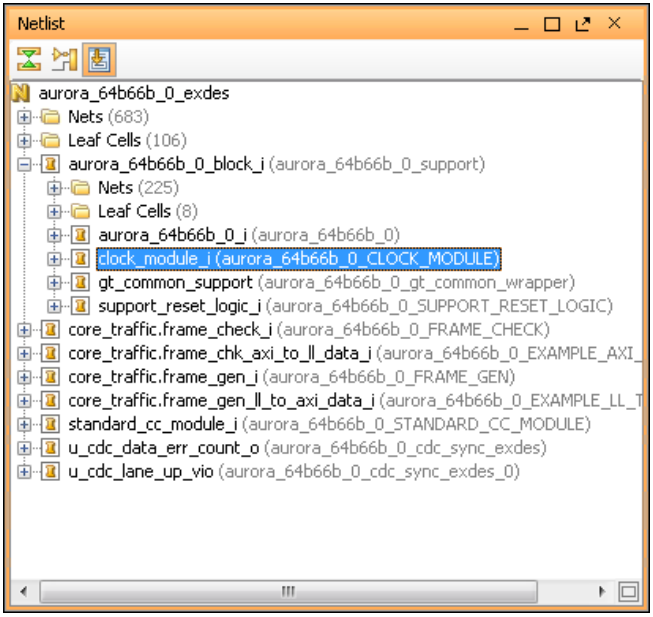


Building a Netlist from the IP Core

LabVIEW FPGA does not support Verilog source files in Component Level IP. However, you can generate EDIF netlists from any synthesized Verilog components in the IP you're using and instantiate the netlist in a VHDL wrapper.The following steps are an example of how to generate an EDIF netlist from the IP core:

1. Open the example project for your IP core in Vivado.
2. Set the appropriate top-level source file for which you plan to generate a netlist.
3. Run synthesis.
4. Open the Synthesized Design using one of the following methods.
   - Select **Open Synthesized Design** in the **Synthesis Completed** pop-up window.
   - Select the **Design Run** tab, then select **Open Synthesized Design** in the left hand pane.
5. In the Tcl Console, enter `write_edif <name of entity>.edf` to create the netlist that you use when you import the IP core into your LabVIEW project. The netlist location is indicated by the Tcl Console window.

6. The following figure shows the cells associated with the design in the **Netlist** window.



7. To build `.edf` files for an associated cell, enter the following command:
   `write_edif -cell <name of cell> <file name>.edf`
   For example, to create an `.edf` for `clock_module_i`, enter the following command:
   `write_edif -cell clock_module_i aurora_64b66b_clock_module.edf`

Note You may have to specify a longer path name depending on the location of the cell in your project. For example, clock_module_i may be located under `aurora_64b66b_0_block_i/clock_module_i`.

8. Copy the netlist into your LabVIEW FPGA CLIP directory.
9. Include your netlist in the list of synthesis files when running the CLIP Wizard.

Writing a VHDL Wrapper Around the Protocol IP Core

A VHDL wrapper is generally necessary to adapt the protocol signals to the dataflow semantics used within the LabVIEW FPGA diagram. NI recommends that you adhere to the following guidelines when writing a VHDL wrapper around the protocol IP core:

- Keep the interface between the CLIP and the LabVIEW FPGA diagram as simple as possible.

Note LabVIEW stores values in big-endian format, and your IP may accept only little-endian format. NI recommends performing any conversions in the CLIP and keeping endian conversions off the LabVIEW diagram for ease of use.

- Do not pass asynchronous signals to the LabVIEW FPGA diagram. Register the signals in a clock domain in the VHDL logic before passing them to the LabVIEW FPGA diagram.
- Use AXI4-Stream and AXI4-Lite interfaces for streaming data and register accesses. NI provides AXI4-Stream and AXI4-Lite wrappers to use on the LabVIEW FPGA diagram. Refer to the *Generating an IP Core from the Xilinx Vivado IP Catalog* section of this document for more information about IP core logic.
- If you expose an AXI4-Lite endpoint, use Xilinx AXI4 interconnect IP to expose only one AXI4-Lite endpoint to the LabVIEW FPGA diagram.
- Document the frequency of clocks coming from CLIP. Consider supporting enable chain removal.
- Implement a state machine that allows asynchronous resets. If you declare an input signal as a reset signal in the CLIP wizard, then that signal is asserted when the LabVIEW FPGA VI is not running.
- Implement a state machine that resets the protocol cores when the **PORT#** module is absent if your state machine does not already account for this.
- Connect various clocks from your CLIP to the DebugClks std_logic_vector in order to use host-side frequency counter debugging utilities.
- Provide timing constraints in XDC for your CLIP. Include timing constraints for clocks within your CLIP, but do not include pin/location constraints on MGTs transceiver lanes and RefClks. Refer to *UG 903: Vivado Design Suite User Guide: Using Constraints* at `xilinx.com` for more information about timing constraints in XDC for your CLIP.
- Use the TXOUTCLK and/or RXOUTCLK clock constraints for your high-speed serial CLIP if your protocol uses it directly.
  - The following is an example syntax for the constraint: `create_clock -period <period in ns> [get_pins %ClipInstancePath%/<path to your clock pin relative to the top level CLIP VHDL>]`.

- If you generate an asynchronous reset within your CLIP VHDL, create a false path constraint from the register that generates the reset signal. Include a “don’t touch” attribute for any false path constraints.
  - The following is an example syntax for the “don’t touch” attribute: `attribute dont_touch : string; attribute dont_touch of <signal name> : signal is "true";`
  - The following is an example syntax for the false path constraint: `set_false_path -from [get_cells %ClipInstancePath%/<path to your register>]`
- When writing constraints, you may need to refer to the CLIP’s instance name or the absolute path to the CLIP instance in the VHDL hierarchy. Refer to the *Constraints and Hierarchy* or more information about using the search-and-replace keywords **%ClipInstanceName%** and **%ClipInstancePath%**.

Constraints and Hierarchy

You can include CLIP-specific user constraints in the compilation using a constraints file, depending on your specific FPGA target. You can use this mechanism for all constraints except pin placement constraints. For example, you can access a clock directly from a global clock input pin through a global clock buffer for socketed CLIP. You must constrain the period of this clock.

For constraints on specific components within CLIP, you might need to specify the location of the component within the overall VHDL hierarchy. In such cases, consider prefacing the constraints with the following macros. Prefacing allows the constraints to be applied regardless of the component location in the VHDL hierarchy. If you want to use this example code, copy the code to a text file and save the file as `DemoClipAdder.xdc`. Add this constraints file along with the VHD file as synthesis files in the Configuring CLIP wizard to implement this constraint.

Xilinx Vivado

```
create_clock -period 10.000 -name %ClipInstanceName%Clk -waveform
{0.000 5.000} -add [get_pins %ClipInstancePath%/clk]

set_clock_latency -clock [get_clocks {%ClipInstanceName%CLK}] 10.0
[get_pins {%ClipInstancePath%/cAddOut[0]}]
```

To instantiate the CLIP multiple times, each CLIP instance must have a unique name, and the name must follow VHDL naming conventions. When you include these macros, you do not need to include a separate constraints file for each instance because the FPGA Module creates a unique instance name.

If a CLIP signal is not used, the Xilinx compilation tools might remove the signal from the bitstream. In such cases, you might get an NGBuild error during compilation. To resolve this issue, remove the constraint or use the signal in an FPGA VI.

Caution In order to guarantee data integrity and timing closure, verify that I/O nodes from the CLIP are written in the same clock domain in which they are read on the LabVIEW diagram and that I/O nodes to the CLIP are read in the same clock

domain in which they are written on the LabVIEW diagram. In rare cases where crossing clock domains is desirable, refer to KnowledgeBase `6OB8E8FM` at `ni.com/kb` for more information about how to write timing constraints between the CLIP and the LabVIEW diagram in order to specify timing exceptions on these paths and achieve timing closure. Note that data corruption might still occur when crossing clock domains.

Documenting Your IP

NI recommends documenting the behavior of your CLIP. Refer to the following guidelines for information about how to document your CLIP and how documenting your CLIP can affect the rest of your design:

- Document the endianness of your CLIP in order to properly interface your CLIP to the LabVIEW FPGA diagram. Refer to the *Writing a VHDL Wrapper Around the Protocol IP Core* section of this chapter for more information about how CLIP endianness affects the design process.
- Clearly define the portion of your entity interface that is facing the diagram, and which portion of your entity is facing the front panel.
- Document the connector signals by describing which signals are used, which signals are unused, and the manner in which the signal is used. Signal use can affect which ports are active with your IP and the behavior of cables upon ingestion and removal.
- Document how you integrate AXI4-Lite signals with LabVIEW data types. Some AXI4-Lite signals do not integrate easily with LabVIEW data types; for example, address ports can have widths of 11, but LabVIEW only provides addresses with widths of 8, 16, 32, and 64. Additionally, the AXI4-Lite and AXI4-Stream adapters are configured for use with fixed-point I/O.
- Document how clocks are used and how they are routed in your CLIP for use with the IP. You must route clocks to the diagram for use with the single-cycle timed loop (SCTL) in LabVIEW FPGA.
- Document the address map of individual components within any AXI4-Lite interfaces.

Adding MGT Socketed CLIP to the LabVIEW Project

After configuring the MGT Socketed CLIP in VHDL, you can use LabVIEW FPGA to continue the development process. LabVIEW FPGA provides FPGA target support, configuration for clocking and routing, and interfacing with LabVIEW on your host computer for a fully integrated development experience.

Refer to the *Related Documentation* section of this manual for a list of LabVIEW FPGA documentation that you may find helpful as you develop your application.

Configuring MGT Socketed CLIP in the NI-793xR LabVIEW FPGA Targets

Complete the following steps to configure MGT Socketed CLIP in your NI-793xR LabVIEW project:

1. Create a new project by selecting **File»New»Project**, or open an existing project by selecting **File»Open**.
2. Right-click the project in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
3. Select **New target or device** and select your device.
4. Right-click the device in the **Project Explorer** window and select **New»FPGA Target** to add an FPGA target to the Controller for FlexRIO.
5. Add the protocol IP through your CLIP. Right-click the device name and select **Properties»Component-Level IP.**

Note If you are using example CLIP or pre-made CLIP, you can import the CLIP using the dialog box, or you can click on the **Create File** icon to create a new CLIP using the CLIP Wizard.

Note You can modify a CLIP by selecting the preexisting CLIP Declaration Name and clicking **Modify File**.

6. If you are generating new CLIP, follow the instructions in the CLIP Wizard to interface your CLIP with LabVIEW FPGA. You do not need to use the CLIP Wizard if you are reusing an existing CLIP. Refer to the *FPGA Module Help* for more detailed information about the CLIP Wizard. The CLIP Wizard guides you through the following tasks.
   - Adding VHDL source, XDC constraints, and EDF/EDN/EDIF netlists
   - Configuring device types
   - Configuring generics
   - Performing syntax checks
   - Specifying how to use the signals in your CLIP

Note In Step 2 of the CLIP Wizard, select the appropriate Component Level IP Type for your target.

Note After you create the CLIP and add the files, you do not need to modify the CLIP for any changes to take place if you do not change the source paths. If you change the source paths or modify the CLIP source files, you must use the CLIP Wizard.

7. Instantiate the CLIP in the MGT Socket. When you add a new target to the project, LabVIEW automatically creates a compatible MGT Socket in the project. Right-click the socket and select **Properties**, then select **General** under **Category**.
8. Select a declaration from the drop-down menu under **Socketed Component Level IP Declaration**.
9. Click **OK**. The user-defined signals in your CLIP appear under the socket item in the **Project Explorer** window.
10. Right-click the MGT Socket and select **Clocking Selections** under **Category** to configure the Clocking and IO Configuration properties for your device.

Note Clocking and routing information is compile-time static and cannot be reconfigured at runtime.

Note The NI-793xR devices support empty sockets.

11. Select the clock that your CLIP requires and explicitly assign it a connection. You must add the clock to your LabVIEW project in order to select it from the **Connections** window. If your CLIP does not require any clocks, leave this page blank.
12. Click **OK**.

Refer to Chapter 3, *Hardware Architecture*, for more information about NI-793xR clocking capabilities.

Using Existing VHDL IP inside CLIP or IPIN

To use existing IP in your project, refer to the *Importing External IP Into LabVIEW FPGA* white paper at `ni.com`.

CLIP does not support custom user libraries in the VHDL. If your VHDL uses custom user libraries, use one of the following workarounds:

- Create a netlist from the VHDL and integrate the netlist using CLIP.
- Reference the default reference library instead of a custom user library.

Refer to the *Creating or Acquiring IP (FPGA Module)* topic in the *LabVIEW FPGA Module Help* for more information about using existing VHDL IP inside CLIP or IPIN.

Improving Performance in Larger Designs through Enable Chain Removal

By default, LabVIEW adds code to the FPGA code to enforce data flow. This code addition is referred to as the enable chain. In larger applications, the enable chain can create routing congestion and limit performance. You can remove the enable chain under certain circumstances. Refer to *Improving Timing Performance in Large Designs (FPGA Module)* in the *LabVIEW FPGA Module Help* for more information about how to remove enable chains and when to do so.

6

Programming with the Real-Time Target

This chapter contains information about programming with the LabVIEW Real-Time target. For information about developing LabVIEW Real-Time applications, refer to the *LabVIEW Real-Time Module Help*.

Best Practices

For information about LabVIEW Real-Time programming best practices, refer to the Real-Time Module Best Practices topic of the *Real-Time Module Help*. This page includes an overview of best practices for designing, developing, and deploying applications with LabVIEW Real-Time.

Key Concepts

The following key concepts provide the basic information you need to start using the Real-Time FlexRIO Target.

- **Real-time (RT) application**—An application designed for stable execution and precise timing.
- **Determinism**—The characteristic of a real-time application that describes how consistently the application responds to external events or performs operations within a given time limit. Maximizing determinism is often a priority when designing real-time applications.
- **Jitter**—The time difference between the fastest and slowest executions of the application. Minimizing jitter is often a priority when designing real-time applications.
- **Real-time operating system (RTOS)**—An operating system designed to run applications with increased determinism and reduced jitter. A general-purpose operating system, like Microsoft Windows, completes operations at unpredictable times. In contrast, each operation an RTOS performs has a known maximum completion time. By designing an application for an RTOS, you can make sure an application will run deterministically.
- **RT target**—A controller, such as an NI-793xR, that runs an RTOS.
- **Stand-alone RT application**—An RT application that runs automatically when you power on an RT target.
- **Device driver software**—A software component that translates commands from LabVIEW into a format appropriate for a particular RT target and any installed I/O devices. You install the appropriate device driver software as a part of configuring your RT target.

- **Host computer**—The computer you use to design a real-time application. You deploy a real-time application from the host computer to the RT target. You can also communicate with the RT target through a user interface running on the host computer.
- **NI Measurement & Automation Explorer (MAX)**—The software you use to configure RT targets. After you install the Real-Time Module on the host computer, you can use MAX to install the Real-Time Module, the RTOS, and device driver software on the RT target.
- **Subnet**—A subdivision of a network over which devices can communicate using TCP/IP protocol. MAX automatically detects RT targets connected to the same subnet as the host computer.
- **Shared variable**—A memory space that you can read data from and write data to. You can read and write shared variables on a single computer with single-process shared variables or on multiple computers with network-published shared variables. Use shared variables to publish only the latest values in a data set to one or more computers.
- **RT FIFO**—Acts like a fixed-size queue, where the first value you write to the FIFO queue is the first value that you can read from the FIFO queue. An RT FIFO ensures deterministic behavior by imposing a size restriction on the data you share and by pre-allocating memory for the data. Use RT FIFO functions to share data between VIs or parallel loops running on an RT target.
- **Network stream**—A lossless, unidirectional, one-to-one communication channel that consists of a writer endpoint and a reader endpoint. Use network streams to stream lossless data over a network.

Installing and Configuring the NI-793xR

Refer to the getting started guide for your NI-793xR for instructions about how to perform the following tasks before developing a real-time application for your NI-793xR:

1. Install support for the NI-793xR on the host computer.
2. Detect and configure the NI-793xR.
3. Install software on the NI-793xR.

Creating a Real-Time Application

For step by step instructions about creating a project and adding a Real-Time target to it, refer to the *Creating a Real-Time FlexRIO Project* topic of the *FlexRIO Help*.

For conceptual information about real-time applications, refer to *Tutorial: Creating a Real-Time Application* topic in the *Real-Time Module How-To* book of the *Real-Time Module Help.*

Real-Time System Integration

The following sections contain information about integrating your Real-Time system with LabVIEW.

Querying Fan Speed and Temperature Sensors

Use the System Configuration API to query the fan sensors or the temperature sensor on the NI-793xR. The System Configuration API can read device properties remotely from a development machine or monitoring system, or you can access these properties locally through the device for self-monitoring.

The NI-793xR includes four temperature sensors and one fan. Three of the temperature sensors monitor the CPU, and one temperature sensor monitors the FPGA. Refer to the following table for the resource you must use to access each temperature sensor and fan, as well as each component's operating range.

Note All temperatures are reported in degrees Celsius (°C).

Table 6-1. NI-793xR Temperature Sensors and Fan

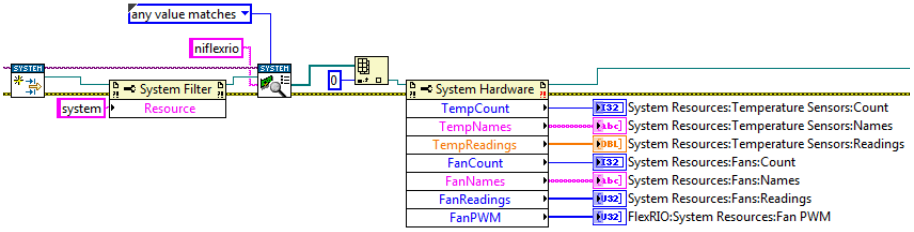
Sensor Name	Resource	Operating Range
CPU Temp 1	System	<98 °C
CPU Temp 2	System	<85 °C
CPU Temp 3	System	<85 °C
System Fan	System	For information about the fan, refer to the *Using the Fan* section.
FPGA Temp	System	<96 °C
Current Temp	RIO0	<96 °C

Note CPU Temp 1 and FPGA Temp are both on-die temperature sensors for their respective component. CPU Temp 2 and CPU Temp 3 are onboard temperature sensors near the CPU. Use CPU Temp 2 and CPU Temp 3 as redundant sensors, or for monitoring internal ambient temperature.

To query the System Fan properties, including the speed reading and PWM (pulse width modulation) duty cycle, filter for the `system` resource and query the properties under the `System Resources::Fans` category and the `FlexRIO::System Resources::FanPWM` property. The speed reading property is in units of RPM, and the PWM property is in units of percentage.

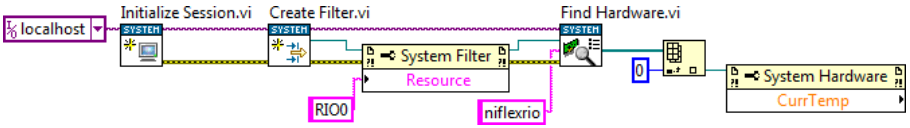
To query the CPU Temp *x* and FPGA Temp sensors, filter for the `system` resource and query the properties under the `System Resources::Temperature Sensors` category.

Figure 6-1. Querying Fan and CPU Temperatures



You can also monitor the FPGA Temp sensor on the RIO0 resource. To do this, filter for the `RIO0` resource and query the `Devices & Chassis::Current Temp` property.

Figure 6-2. Querying FPGA Temperature



Power/Thermal Protection and Shutdown

If the FPGA overheats or the temperature monitor cannot be read, FPGA communication is shut down and accesses to the FPGA do not work. Additionally, a device status message appears in MAX under the FPGA item that has been shut down. If the FPGA communication shuts down, power cycle the system and contact NI customer support at `ni.com/support`. In order to avoid seeing this error again, improve the airflow to your chassis or consider reduced FPGA logic in your design.

You can also query the device status through the *LabVIEW System Configuration API*.

LabVIEW System Configuration API

The LabVIEW System Configuration API allows you to gather information and perform tasks programmatically on both local and remote systems. The System Configuration palette is located on the functions palette in LabVIEW under **Measurement I/O**.

Note If Measurement I/O does not appear on the functions palette, you can enable it by selecting **Customize»Change Visible Palette**.

Complete the following steps to use the LabVIEW System Configuration API with your NI-793xR Real-Time project.

1. Open a session and point to your target using its IP address.
2. Enter your user name and password, if applicable.
3. Open the **System Configuration** palette in LabVIEW.
4. Open the **Property Node (Hardware)** to obtain information such as the device temperature and device model name.

Refer to the *NI System Configuration API Help* topic of the *LabVIEW Help* for more information about using the LabVIEW System Configuration API. For information about the FlexRIO System Configuration API, refer to the FlexRIO System Configuration Expert topic in the *FlexRIO Help*.

Communicating with Applications on an RT Target

The RT Engine on the RT target does not provide a user interface for applications. You can use one of two communication protocols, front panel communication or network communication, to provide a user interface on the host computer for RT target VIs.

Front Panel Communication

With front panel communication, LabVIEW and the RT Engine execute different parts of the same VI. LabVIEW on the host computer displays the front panel of the VI while the RT Engine executes the block diagram. A user interface thread handles the communication between LabVIEW and the RT Engine.

Use front panel communication between LabVIEW on the host computer and the RT Engine to control and test VIs running on an RT target. After downloading and running the VIs, keep LabVIEW on the host computer open to display and interact with the front panel of the VI.

You also can use front panel communication to debug VIs while they run on the RT target. You can use LabVIEW debugging tools—such as probes, execution highlighting, breakpoints, and single stepping—to locate errors on the block diagram code. Refer to the *Building, Deploying, and Debugging Applications (Real-Time Module)* topic of the *Real-Time Module Help* for information about debugging applications.

Front panel communication is a good communication method to use during development because front panel communication is a quick method for monitoring and interfacing with VIs running on an RT target. However, front panel communication is not deterministic and can affect the determinism of a time-critical VI. Use network communication methods to increase the efficiency of the communication between a host computer and VIs running on the RT target.

Network Communication

With network communication, a host VI runs on the host computer and communicates with the VI running on the RT target using specific network communication methods such as TCP, VI Server, and in the case of non-networked RT Series plug-in devices, shared memory reads and writes. You might use network communication for the following reasons:

- You want to run another VI on the host computer.
- You want to control the data exchanged between the host computer and the RT target. You can customize communication code to specify which front panel objects get updated and when. You also can control which components are visible on the front panel because some controls and indicators might be more important than others.
- You want to control timing and sequencing of the data transfer.
- You want to perform additional data processing or logging.

For more information about interacting with the front panels of RT target VIs, refer to the *Interacting with the Front Panels of RT Target VIs* topic in the *LabVIEW Real-Time Module Help*.

Note The *Interacting with the Front Panels of RT Target VIs* topic in the *LabVIEW Real-Time Module Help* contains information about an embedded UI, which is not available on NI-793xR targets.

Where to Go from Here

The Real-Time Module includes a comprehensive documentation set designed to help you create deterministic applications to run on RT targets.

LabVIEW Help

The *LabVIEW Help*, available by selecting **Help»LabVIEW Help** in LabVIEW, contains the following information that is specific to the Real-Time Module:

- **Real-Time Module Best Practices**—Information about best practices for designing, developing, and deploying applications with the Real-Time Module.
- **Real-Time Module Concepts**—Information about programming concepts, application architectures, and Real-Time Module features you can use to create deterministic applications.
- **Real-Time Module How-To**—Step-by-step instructions for using Real-Time Module features.
- **Real-Time VIs**—Reference information about Real-Time Module VIs, functions, and error codes.

- **Real-Time Operating Systems**—Information about using LabVIEW on real-time operating systems.
- **Real-Time Module Error Codes**—Information about error codes specific to the Real-Time Module.

LabVIEW Real-Time Module Release and Upgrade Notes

The *LabVIEW Real-Time Module Release and Upgrade Notes* contains information to help you install and configure the Real-Time Module and a list of upgrade issues and new features. Complete the following steps to access this document:

1. Open the `labview\manuals` directory.
2. Double-click `RT_Release_Upgrade_Notes.pdf` to open this manual.

A

CLIP Signals

This chapter contains lists of CLIP signals for the NI-7932R and NI-7935R devices.

NI-7932R

Refer to the following table for a list of the NI-7932R socketed CLIP signals.

Table A-1. NI-7932R CLIP Signals

Port	Direction	Clock Domain	Description
MGT_RefClk0_p	In (pad)	—	Differential input clock that you must connect to an IBUFDS_GTE2 input buffer primitive when this input clock is used in your design
MGT_RefClk0_n	In (pad)		
SocketClk40	In	Clock	A 40 MHz clock that runs continuously regardless of connectivity. This signal is connected to the **40 MHz Onboard Clock** signal, which is the default top-level clock for the LabVIEW FPGA VI.

Table A-1. NI-7932R CLIP Signals (Continued)

Port	Direction	Clock Domain	Description
aResetSl	In	Async	This signal is not required. This signal is an asynchronous reset signal from the LabVIEW FPGA environment. If you create an input signal to your CLIP and assign it as **Reset** in the CLIP wizard, that signal is driven as an asynchronous reset signal. Reset all CLIP state machines and logic whenever this signal is logic high. This signal is driven high when you call the LabVIEW FPGA **Reset** invoke method. Call **Run** on the FPGA VI to deassert this signal. Do not use CLIP inputs from the LabVIEW FPGA VI in the CLIP until **aResetS1** is deasserted.
Port<0..1>_RX_p	In (pad)	—	Dedicated MGT receive signals for Port <0..1>.
Port<0..1>_RX_n	In (pad)	—	
Port<0..1>_TX_p	Out (pad)	—	Dedicated MGT transmit signals for Port <0..1>.
Port<0..1>_TX_n	Out (pad)	—	
Port<0..1>_Tx_Fault	In	Async	When high, indicates a laser fault. Low indicates normal operation.
Port<0..1>_LOS	In	Async	When high, this input indicates that the received optical power is below the worst-case receiver sensitivity. Low indicates normal operation.
Port<0..1>_ABS	In	Async	When high, this input indicates that a module is plugged into the SFP+ socket. Low indicates that a module has been detected.

Table A-1. NI-7932R CLIP Signals (Continued)

Port	Direction	Clock Domain	Description
Port<0..1>_Tx_ Disable	Out	Async	When high, this output shuts down the transmitter optical transmitter. When low, operation is enabled.
Port<0..1>_Rs<0..1>	Out	Async	Rate selection pins.
Port<0..1>_SCL	In/Out	Async	Bidirectional serial clock signal for the two-wire communication interface on the Port <0..1> connector. Valid values: `0` and `Z` (open drain). This signal is also called MODDEF1.
Port<0..1>_SDA	In/Out	Async	Bidirectional serial data signal for the two-wire communication interface on the Port <0..1> connector. Valid values: `0` and `Z` (open drain). This signal is also called MODDEF2.
Port<0..1>_ MacAddress	In	Async	Unique 48-bit MAC address assigned to Port<0..1>. Use this address when implementing a network interface controller on Port<0..1>.
Port<0..1>_ MacAddressValid	In	Async	When asserted, this signal indicates that Port<0..1>_MacAddress is valid.

Table A-1. NI-7932R CLIP Signals (Continued)

Port	Direction	Clock Domain	Description
sPort<0..1>_ EnablePower	Out	SocketClk40	Enables or disables the power supply to Port <0..1>. This signal is active high.
sPort<0..1>_ PowerGood	In	SocketClk40	Indicates that the power supply to the cable for Port <0..1> is enabled. This signal may deassert if an over-power condition is detected.

NI-7935R

Refer to the following table for a list of the NI-7935R socketed CLIP signals.

Table A-2. NI-7935R CLIP Signals

Port	Direction	Clock Domain	Description
MGT_RefClk0_p	In (pad)	—	Differential input clock that you must connect to an IBUFDS_GTE2 input buffer primitive when this input clock is used in your design
MGT_RefClk0_n	In (pad)	—	
SocketClk40	In	Clock	A 40 MHz clock that runs continuously regardless of connectivity. This signal is connected to the **40 MHz Onboard Clock** signal, which is the default top-level clock for the LabVIEW FPGA VI.

Table A-2. NI-7935R CLIP Signals (Continued)

Port	Direction	Clock Domain	Description
aResetSl	In	Async	This signal is not required. This signal is an asynchronous reset signal from the LabVIEW FPGA environment. If you create an input signal to your CLIP and assign it as **Reset** in the CLIP wizard, that signal is driven as an asynchronous reset signal. Reset all CLIP state machines and logic whenever this signal is logic high. This signal is driven high when you call the LabVIEW FPGA **Reset** invoke method. Call **Run** on the FPGA VI to deassert this signal. Do not use CLIP inputs from the LabVIEW FPGA VI in the CLIP until **aResetS1** is deasserted.
Port<0..1>_RX_p	In (pad)	—	Dedicated MGT receive signals for Port <0..1>.
Port<0..1>_RX_n	In (pad)	—	
Port<0..1>_TX_p	Out (pad)	—	Dedicated MGT transmit signals for Port <0..1>.
Port<0..1>_TX_n	Out (pad)	—	
Port<0..1>_Tx_Fault	In	Async	When high, indicates a laser fault. Low indicates normal operation.
Port<0..1>_LOS	In	Async	When high, this input indicates that the received optical power is below the worst-case receiver sensitivity. Low indicates normal operation.
Port<0..1>_ABS	In	Async	When high, this input indicates that a module is plugged into the SFP+ socket. Low indicates that a module has been detected.

Table A-2. NI-7935R CLIP Signals (Continued)

Port	Direction	Clock Domain	Description
Port<0..1>_Tx_ Disable	Out	Async	When high, this output shuts down the transmitter optical transmitter. When low, operation is enabled.
Port<0..1>_Rs<0..1>	Out	Async	Rate selection pins.
Port<0..1>_SCL	In/Out	Async	Bidirectional serial clock signal for the two-wire communication interface on the Port <0..1> connector. Valid values: 0 and Z (open drain). This signal is also called MODDEF1.
Port<0..1>_SDA	In/Out	Async	Bidirectional serial data signal for the two-wire communication interface on the Port <0..1> connector. Valid values: 0 and Z (open drain). This signal is also called MODDEF2.
Port<0..1>_ MacAddress	In	Async	Unique 48-bit MAC address assigned to Port<0..1>. Use this address when implementing a network interface controller on Port<0..1>.
Port<0..1>_ MacAddressValid	In	Async	When asserted, this signal indicates that Port<0..1>_MacAddress is valid.

Table A-2. NI-7935R CLIP Signals (Continued)

Port	Direction	Clock Domain	Description
sPort<0..1>_ EnablePower	Out	SocketClk40	Enables or disables the power supply to Port <0..1>. This signal is active high.
sPort<0..1>_ PowerGood	In	SocketClk40	Indicates that the power supply to the cable for Port <0..1> is enabled. This signal may deassert if an over-power condition is detected.

B

Using the Fan

The NI-793xR includes a low power consumption DC fan for cooling the device. The following table lists the fan specifications.

Table B-1. NI-793xR Fan Specifications

Manufacturer	Sanyo Denki
Manufacturer part number	9GA0412G7001
Rated voltage	12 V
Operating voltage range	7 V to 13.8 V
Rated speed	13,100 rpm
Air flow	0.36 m^3/min (12.7 CFM)
Operating temperature	-10 °C to 70 °C
Life expectancy (continual operation)	40,000 h (60 °C)
	70,000 h (40 °C)

Refer to the Sanyo Denki website for complete specifications.

Replacing the Fan

The NI-793xR includes a replaceable fan assembly. For fan troubleshooting information and to order replacement parts , refer to ni.com/support.

C

NI Services

National Instruments provides global services and support as part of our commitment to your success. Take advantage of product services in addition to training and certification programs that meet your needs during each phase of the application life cycle; from planning and development through deployment and ongoing maintenance.

To get started, register your product at ni.com/myproducts.

As a registered NI product user, you are entitled to the following benefits:

- Access to applicable product services.
- Easier product management with an online account.
- Receive critical part notifications, software updates, and service expirations.

Log in to your National Instruments ni.com User Profile to get personalized access to your services.

Services and Resources

- **Maintenance and Hardware Services**—NI helps you identify your systems' accuracy and reliability requirements and provides warranty, sparing, and calibration services to help you maintain accuracy and minimize downtime over the life of your system. Visit ni.com/services for more information.
  - **Warranty and Repair**—All NI hardware features a one-year standard warranty that is extendable up to five years. NI offers repair services performed in a timely manner by highly trained factory technicians using only original parts at a National Instruments service center.
  - **Calibration**—Through regular calibration, you can quantify and improve the measurement performance of an instrument. NI provides state-of-the-art calibration services. If your product supports calibration, you can obtain the calibration certificate for your product at ni.com/calibration.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.

- **Training and Certification**—The NI training and certification program is the most effective way to increase application development proficiency and productivity. Visit ni.com/training for more information.
  - The Skills Guide assists you in identifying the proficiency requirements of your current application and gives you options for obtaining those skills consistent with your time and budget constraints and personal learning preferences. Visit ni.com/skills-guide to see these custom paths.
  - NI offers courses in several languages and formats including instructor-led classes at facilities worldwide, courses on-site at your facility, and online courses to serve your individual needs.
- **Technical Support**—Support at ni.com/support includes the following resources:
  - **Self-Help Technical Resources**—Visit ni.com/support for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on. Registered users also receive access to the NI Discussion Forums at ni.com/forums. NI Applications Engineers make sure every question submitted online receives an answer.
  - **Software Support Service Membership**—The Standard Service Program (SSP) is a renewable one-year subscription included with almost every NI software product, including NI Developer Suite. This program entitles members to direct access to NI Applications Engineers through phone and email for one-to-one technical support, as well as exclusive access to online training modules at ni.com/self-paced-training. NI also offers flexible extended contract options that guarantee your SSP benefits are available without interruption for as long as you need them. Visit ni.com/ssp for more information.
- **Declaration of Conformity (DoC)**—A 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.

For information about other technical support options in your area, visit ni.com/services, or contact your local office at ni.com/contact.

You also can 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.

Glossary

C

CLIP	Component-level intellectual property. CLIP provides access to adapter module physical I/O from within the LabVIEW FPGA environment.

D

DDR3	Double data rate. This term usually refers to the communication mechanism used to read and write DRAM.
DRAM	Dynamic random-access memory

F

FPGA	Field-programmable gate array. NI-793xR modules use Xilinx Kintex-7 FPGAs.

G

GPIO	General-purpose input/output

H

HDL	Hardware-description language. Language that describes a circuit's operation, design, and organization.

L

LVFPGA	LabVIEW FPGA

M

MGT	Multi-gigabit transceiver. An MGT is a SerDes capable of operating at serial bits above 1 Gb/s.

P

PFI	Programmable function interface

S

SCTL	Single cycle timed loop
SFP+	Enhanced small form-factor pluggable

V

VHDL	VHSIC Hardware Description Language