APEX WAVES

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.

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

1-800-915-6216

www.apexwaves.com

sales@apexwaves.com

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

Request a Quote   CLICK HERE

PXIe-7966R

Understanding FlexRIO Modular I/O FPGA Modules

2023-05-10

Contents

Base Clock Resources for FlexRIO Modular I/O FPGA Modules

A base clock is a digital signal existing in hardware that you can use as a clock for an FPGA application.

Your FlexRIO with Modular I/O FPGA Module provides several base clock resources that can be used to run a LabVIEW FPGA VI. Use the table below to determine which base clock resources are available for your FlexRIO FPGA Module.

Table 1. Base Clock Resources by Module

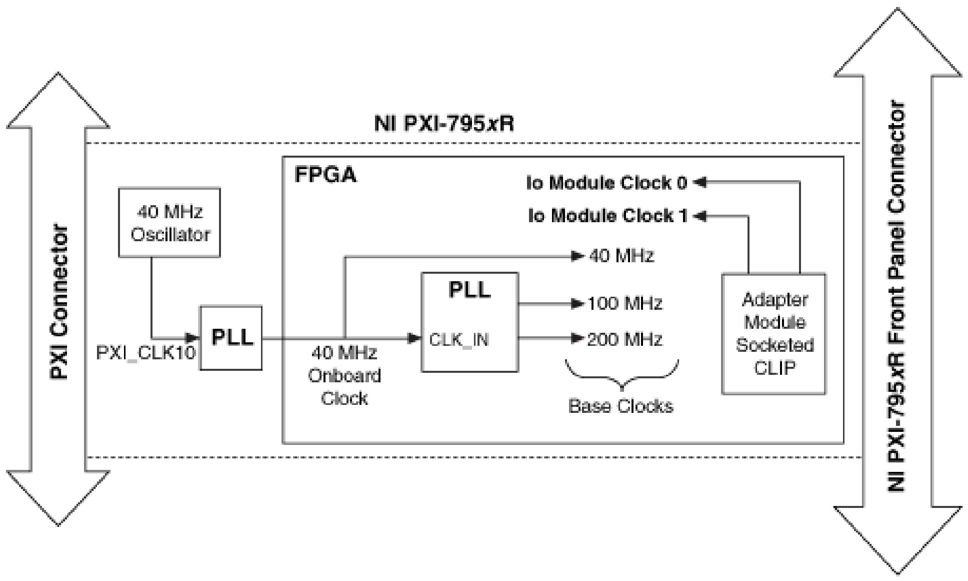
Base Clock	FPGA Module
40 MHz Onboard Clock	all FlexRIO Modular I/O FPGA Modules
PXI_CLK10	all FlexRIO Modular I/O FPGA Modules
100 MHz Clock	all FlexRIO Modular I/O FPGA Modules
200 MHz Clock	all FlexRIO Modular I/O FPGA Modules
IO Module Clock 0/IO Module Clock 1	PXI-795x, PXIe-796x
PXIe_DStarA	PXIe-796x, PXIe-797x
DRAM Clock	PXIe-797x

- **40 MHz Onboard Clock**—The 40 MHz Onboard Clock is the default clock in your LabVIEW FPGA project. This clock can be used as a top-level clock for running your LabVIEW FPGA VI. The top-level clock on an FPGA target determines the execution time of the individual functions and VIs on the FPGA VI block diagram. If you change the frequency of the top-level clock, you also change the execution speed of functions on the block diagram and the execution rate of the FPGA VI.
- **PXI_CLK10**—PXI_CLK10 can be used as a source for running your LabVIEW FPGA VI.
- **100 MHz Clock**—The 100 MHz Clock can be used as a source for running your LabVIEW FPGA VI. The 100 MHz Clock is generated from a PLL in the FlexRIO FPGA module.

- **200 MHz Clock**—The 200 MHz Clock can be used as a source for running your LabVIEW FPGA VI. The 200 MHz Clock is generated from a PLL in the FlexRIO FPGA module.
- **IO Module Clock 0/IO Module Clock 1**—External clock source from your adapter module socketed CLIP. Add this clock to your project from the FPGA Base Clock Properties dialog box. Refer to the CLIP documentation for your specific adapter module for information about the device specific uses of this clock.
- **PXIe_DStarA**—Configure this clock to the frequency of your choice for running your LabVIEW FPGA VI. You can drive the PXIe_DStarA using a timing and synchronization device.
- **DRAM Clock**—DRAM Clock is a 167 MHz clock that drives the DRAM interface.

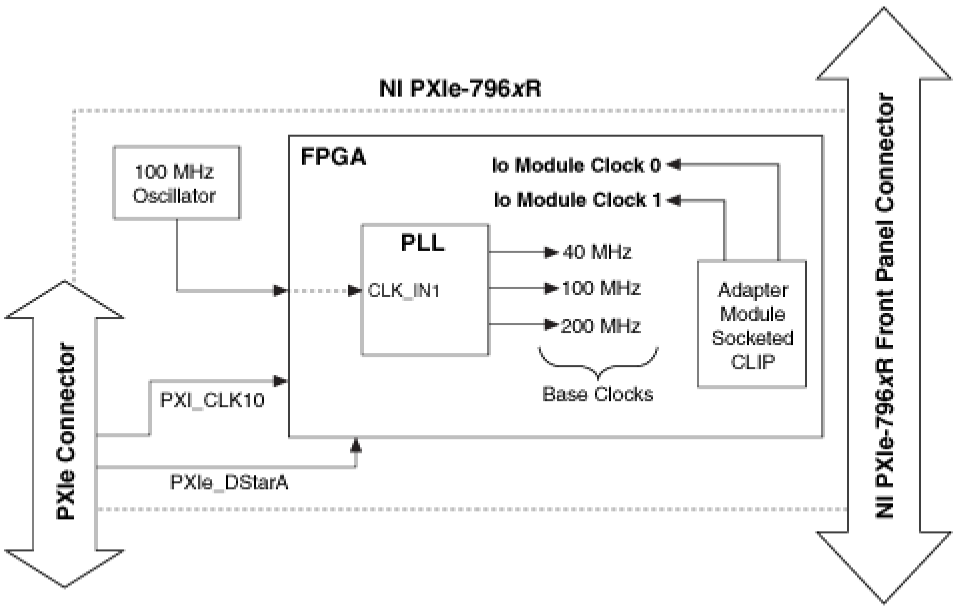
Routing Diagram for PXI-795x Base Clocks

On PXI-795x modules, the 40 MHz onboard clock is generated using a PLL outside of the FPGA. The 100 MHz clock and 200 MHz clock are generated within the device FPGA using a PLL.



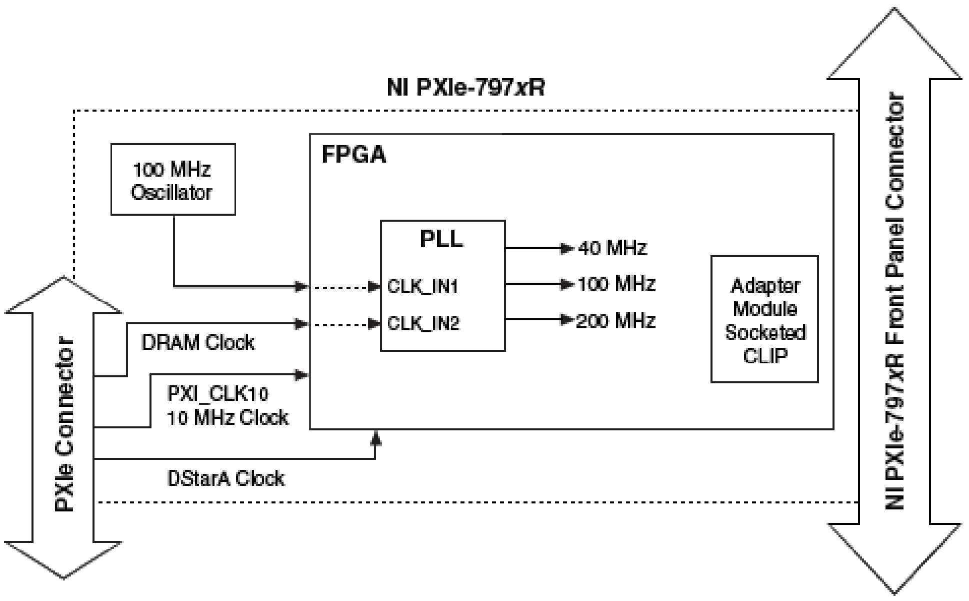
Routing Diagram for PXIe-796x Base Clocks

On PXIe-796x modules, the 40 MHz onboard clock, the 100 MHz clock, and the 200 MHz clock are generated within the device FPGA using a PLL. The PLL source is the onboard 100 MHz oscillator clock. After the VI is downloaded, the FPGA detects whether or not the backplane 100 MHz oscillator clock is present, and if so, it configures the PLL to lock to the backplane 100 MHz oscillator clock.



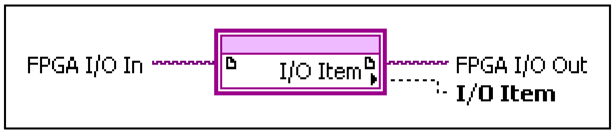
Routing Diagram for PXIe-797x Base Clocks

On PXIe-797x modules, the 40 MHz onboard clock, the 100 MHz clock, and the 200 MHz clock are generated within the device FPGA using a PLL. The PLL source is either the PXI Express Clk100 backplane clock or the onboard 100 MHz oscillator clock. After the VI is downloaded, the FPGA detects whether or not the backplane PXI Express 100 MHz clock is present, and if so, it configures the PLL to lock to the backplane PXI Express 100 MHz clock. If the PXI Express backplane clock is not present (for example, in a CompactPCI Express chassis), then the device FPGA sets the PLL to lock to the onboard 100 MHz oscillator clock.


NI PXIe-797xR
100 MHz Oscillator
FPGA
PLL
CLK_IN1
CLK_IN2
40 MHz
100 MHz
200 MHz
Adapter Module Socketed CLIP
DRAM Clock
PXI_CLK10 10 MHz Clock
DStarA Clock
PXIe Connector
NI PXIe-797xR Front Panel Connector

FlexRIO Modular I/O FPGA Status Items

Figure 1. FPGA I/O Node



You can use an FPGA I/O Node, configured for reading and writing, with the your FlexRIO Modular I/O FPGA Module. The FPGA I/O Node is located on the FPGA I/O Functions palette and performs specific FPGA I/O operations on your FPGA target.

The IO Module Status terminals are synchronous to the 40 MHz clock, and they should be accessed through that clock domain.

PXI-795x Status Items

Board IO FPGA I/O Items

FPGA I/O Item	Description
Device Temperature	Returns the current temperature of the device, in increments of 1 °C. The temperature is measured from an onboard temperature sensor on the module PCB, external to the FPGA. Note The Device Temperature terminal is synchronous to the 40 MHz clock, and it should be accessed through that clock domain. Note The Device Temperature interface requires some time to fully initialize after FPGA download and reset operations. When performing an FPGA download or reset, wait

FPGA I/O Item	Description
	at least 250 ms before reading the Device Temperature terminal.
LEDs	Provides read/write access to the four onboard LEDs (reference designator D<2..5>, located on the module PCB). You can configure these LEDs any way that is useful to you during development and debugging. The four LEDs are accessed through bits <0..3> of the U8 data type.

IO Module Status FPGA I/O Items

FPGA I/O Item	Description
EEPROM Power Enabled	TRUE = Indicates that the adapter module EEPROM power rail is enabled. FALSE = Indicates that the adapter module EEPROM power rail is not enabled.
Expected IO Module ID	Reports the unique 32-bit IO Module ID that the currently downloaded FPGA VI is configured to use. The inserted IO Module ID must match the expected IO Module ID so the adapter module can function properly. The Expected IO Module ID is configured by choosing the appropriate adapter module in the IO Module General Properties dialog box and then recompiling and redownloading the FPGA VI.
Inserted IO Module ID	Reports the unique 32-bit IO module ID of the currently inserted adapter module. The Inserted IO Module ID must match the expected IO Module ID for the adapter module to function properly.
IO Module IO Enabled	TRUE = Indicates that the adapter module connector FPGA I/O pins are enabled.

FPGA I/O Item	Description
	FALSE = Indicates that the adapter module connector FPGA I/O pins are not enabled. **Note** These pins are enabled only when an adapter module is connected, properly powered, and when the FPGA has determined that the connected adapter module is compatible with the FPGA VI currently downloaded to the FPGA.
IO Module Power Enabled	TRUE = Indicates that the adapter module power is fully enabled. FALSE = Indicates that the adapter module power is not enabled. **Note** Adapter module power is enabled only when an adapter module is connected and the FPGA has determined that the connected adapter module is compatible with the program currently downloaded to the FPGA.
IO Module Power Good	Reports the state of the power good signal on the adapter module connector interface. TRUE = Indicates that the connected adapter module power rails have had the proper amount of time to initialize and are operational and stable. FALSE = Indicates that the connected adapter module power rails are not ready.

FPGA I/O Item	Description
IO Module Present	TRUE = Indicates that an adapter module is physically connected to the FPGA module. FALSE = Indicates that an adapter module is not connected to the FPGA module.

Note In software, FlexRIO adapter modules are referred to as IO modules.

PXIe-796x Status Items

IO Module Status FPGA I/O Items

FPGA I/O Item	Description
EEPROM Power Enabled	TRUE = Indicates that the adapter module EEPROM power rail is enabled. FALSE = Indicates that the adapter module EEPROM power rail is not enabled.
Expected IO Module ID	Reports the unique 32-bit IO Module ID that the currently downloaded FPGA VI is configured to use. The inserted IO Module ID must match the expected IO Module ID so the adapter module can function properly. The Expected IO Module ID is configured by choosing the appropriate adapter module in the IO Module General Properties dialog box and then recompiling and redownloading the FPGA VI.
Inserted IO Module ID	Reports the unique 32-bit IO module ID of the currently inserted adapter module. The Inserted IO Module ID must match the expected IO Module ID for the adapter module to function properly.

FPGA I/O Item	Description
IO Module IO Enabled	TRUE = Indicates that the adapter module connector FPGA I/O pins are enabled. FALSE = Indicates that the adapter module connector FPGA I/O pins are not enabled. **Note** These pins are enabled only when an adapter module is connected, properly powered, and when the FPGA has determined that the connected adapter module is compatible with the FPGA VI currently downloaded to the FPGA.
IO Module Power Enabled	TRUE = Indicates that the adapter module power is fully enabled. FALSE = Indicates that the adapter module power is not enabled. **Note** Adapter module power is enabled only when an adapter module is connected and the FPGA has determined that the connected adapter module is compatible with the program currently downloaded to the FPGA.
IO Module Power Good	Reports the state of the power good signal on the adapter module connector interface. TRUE = Indicates that the connected adapter module power rails have had the proper amount of time to initialize and are operational and stable. FALSE = Indicates that the connected adapter module power rails are not ready.
IO Module Present	TRUE = Indicates that an adapter module is physically connected to the FPGA module.

FPGA I/O Item	Description
	FALSE = Indicates that an adapter module is not connected to the FPGA module.

Note In software, FlexRIO adapter modules are referred to as IO modules.

Board IO FPGA I/O Items

FPGA I/O Item	Description
Device Temperature	Returns the current temperature of the device, in increments of 1 °C. The temperature is measured from an onboard temperature sensor on the module PCB, external to the FPGA. **Note** The Device Temperature terminal is synchronous to the 40 MHz clock, and it should be accessed through that clock domain. **Note** The Device Temperature interface requires some time to fully initialize after FPGA download and reset operations. When performing an FPGA download or reset, wait at least 250 ms before reading the Device Temperature terminal.
Clock100 PLL Unlocked	Indicates if the PLL that generates the 40 MHz, 100 MHz, and 200 MHz base clocks has unlocked after the FPGA bitfile download. After the FPGA bitfile downloads, the PLL locks to PXIe_CLK100 by default. If PXIe_CLK100 is not present, the PLL locks to an onboard 100 MHz oscillator. If the PLL unlocks from PXIe_CLK100 (due to a glitch on the clock, for example) the FPGA then attempts to relock the PLL to the 100 MHz oscillator.

FPGA I/O Item	Description
	Note NI recommends that you reload the FPGA bitfile if the Clock100 PLL Unlocked status is TRUE. Xilinx does not guarantee proper PLL operation in these cases, and undesired behavior may occur.

PXIe-797x Status Items

Board IO FPGA I/O Items

FPGA I/O Item	Description
Clock100 PLL Unlocked	Indicates if the PLL that generates the 40 MHz, 100 MHz, and 200 MHz base clocks has unlocked after the FPGA bitfile download. After the FPGA bitfile downloads, the PLL locks to PXIe_CLK100 by default. If PXIe_CLK100 is not present, the PLL locks to an onboard 100 MHz oscillator. If the PLL unlocks from PXIe_CLK100 (due to a glitch on the clock, for example) the FPGA then attempts to relock the PLL to the 100 MHz oscillator. **Note** NI recommends that you reload the FPGA bitfile if the Clock100 PLL Unlocked status is TRUE. Xilinx does not guarantee proper PLL operation in these cases, and undesired behavior may occur.
Device 3.3V Power	Returns the power that the 3.3V power rail is consuming. The power consumption is indicated by the following formula:

FPGA I/O Item	Description
	Power = P x 10 mW, where P is the value returned by LabVIEW. Therefore, a value of 150 indicates a power consumption of 1.5W (150 x 10 mW = 1.5 W). The default shutdown limit for this power rail is 9.7 W; do not exceed this limit. Do not exceed 37.75 W of power consumption between the 3.3V power rail and the 12V power rail. Exceeding this limit causes the FPGA to shut down.
Device 12V Power	Returns the power that the 12V power rail is consuming. The power consumption is indicated by the following formula: Power = P x 10 mW, where P is the value returned by LabVIEW. Therefore, a value of 150 indicates a power consumption of 1.5W (150 x 10 mW = 1.5 W). The default shutdown limit for this power rail is 35.5 W; do not exceed this limit. Do not exceed 37.75 W of power consumption between the 3.3V power rail and the 12V power rail. Exceeding this limit causes the FPGA to shut down.
Device Temperature	Returns the current temperature of the device, in increments of 1 °C. The temperature is measured from an onboard temperature sensor on the PXIe-7972R PCB, external to the FPGA. **Note** The Device Temperature terminal is synchronous to the 40 MHz clock, and it should be accessed through that clock domain. **Note** The Device Temperature interface requires some time to fully initialize after FPGA download and reset operations. When performing an FPGA download or reset, wait

FPGA I/O Item	Description
	at least 250 ms before reading the Device Temperature terminal.

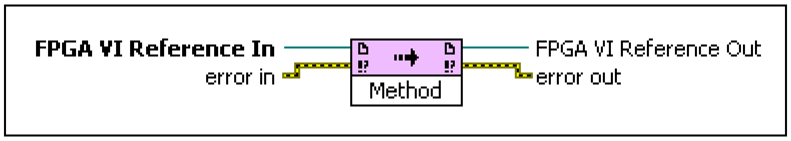
IO Module Status FPGA I/O Items

FPGA I/O Item	Description
EEPROM Power Enabled	TRUE = Indicates that the adapter module EEPROM power rail is enabled. FALSE = Indicates that the adapter module EEPROM power rail is not enabled.
Expected IO Module ID	Reports the unique 32-bit IO Module ID that the currently downloaded FPGA VI is configured to use. The inserted IO Module ID must match the expected IO Module ID so the adapter module can function properly. The Expected IO Module ID is configured by choosing the appropriate adapter module in the IO Module General Properties dialog box and then recompiling and redownloading the FPGA VI.
Inserted IO Module ID	Reports the unique 32-bit IO module ID of the currently inserted adapter module. The Inserted IO Module ID must match the expected IO Module ID for the adapter module to function properly.
IO Module IO Enabled	TRUE = Indicates that the adapter module connector FPGA I/O pins are enabled. FALSE = Indicates that the adapter module connector FPGA I/O pins are not enabled. **Note** These pins are enabled only when an adapter module is connected, properly powered, and

FPGA I/O Item	Description
	when the FPGA has determined that the connected adapter module is compatible with the FPGA VI currently downloaded to the FPGA.
IO Module Power Enabled	TRUE = Indicates that the adapter module power is fully enabled. FALSE = Indicates that the adapter module power is not enabled. **Note** Adapter module power is enabled only when an adapter module is connected and the FPGA has determined that the connected adapter module is compatible with the program currently downloaded to the FPGA.
IO Module Power Good	Reports the state of the power good signal on the adapter module connector interface. TRUE = Indicates that the connected adapter module power rails have had the proper amount of time to initialize and are operational and stable. FALSE = Indicates that the connected adapter module power rails are not ready.
IO Module Present	TRUE = Indicates that an adapter module is physically connected to the FPGA module. FALSE = Indicates that an adapter module is not connected to the FPGA module.

Note In software, FlexRIO adapter modules are referred to as IO modules.

Invoke Method Functions



You can use an Invoke Method function to invoke an action from the host VI to the FPGA VI with this device. The host VI is available from the LabVIEW FPGA host interface. The Invoke Method function is located on the FPGA Interface VIs and Functions palette.

In addition to the standard methods supported by the LabVIEW FPGA Module, NI FlexRIO targets also support the following custom FPGA Invoke methods.

Interface Methods	Description
Control IO Module	Enables or disables power to the connected adapter module. You must physically insert an adapter module into the FPGA module to run this method. **Enabled** TRUE = Enables power to the adapter module. FALSE = Disables power to the adapter module.
IO Module Status	Queries status information about the FlexRIO adapter module interface. You do not need to physically insert an adapter module in the FlexRIO system to run this method. **IO Module Present?** TRUE = Indicates that an adapter module is physically inserted in the FlexRIO system.

Interface Methods	Description
	FALSE = Indicates that no adapter module is inserted. **Power Enabled?** TRUE = Indicates that power to the adapter module is enabled. FALSE = Indicates that power to the adapter module is disabled. **I/O Enabled?** TRUE = Indicates that FPGA GPIO pin buffers on the adapter module interface are enabled. FALSE = Indicates that FPGA GPIO pin buffers are disabled. **IO Module Mismatch?** TRUE = Indicates that the currently downloaded FPGA VI is expecting to use a different adapter module than that which is currently inserted. In this case, power and I/O to the adapter module is disabled. To use the currently inserted adapter module with your FPGA program, configure your LabVIEW project to use the currently inserted adapter module, recompile your VI, and redownload it to the FPGA. FALSE = Indicates that the currently downloaded FPGA VI is compatible with the current adapter module.
Redetect IO Module	Redetects the adapter module that is currently connected to the FlexRIO FPGA module. You must physically insert an adapter module in the FlexRIO FPGA module to run this method.

Interface Methods	Description
	Note This method is only necessary when creating a custom adapter module.

Configuring DRAM with FPGA Memory Items

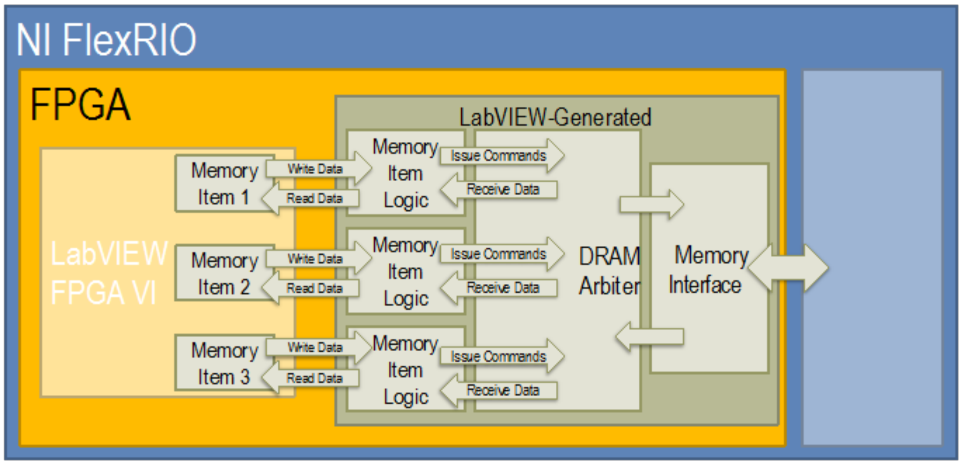
Use the FPGA memory item interface to use DRAM in the same way that you use block memory and look-up tables (LUT). DRAM memory items appear in the **Project Explorer** window under the FPGA target. The FPGA memory item interface allows you to partition the physical DRAM banks into multiple memory items. You can create target-scoped or VI-defined memory items.

Complete the following steps to configure DRAM with FPGA memory items.

1. Determine whether you want to create a target-scoped memory item or a VI-defined memory item.

Option	Description
Create a target-scoped memory item	Right-click the FPGA target in the Project Explorer window and select **New** » **Memory** from the shortcut menu. The Memory Properties dialog box appears.
Create a VI-defined memory item	Place a VI-defined Memory Configuration node on the block diagram, right-click the node, and select **Configure** from the shortcut menu. The Memory Properties dialog box appears.

Note A memory item targets a single DRAM bank. If you only select one memory item, this memory item can be as large as the entire bank. You can use memory items to divide the full DRAM space into smaller memories that you can access independently from different sections of the LabVIEW FPGA code. The figure below shows the logic that LabVIEW generates to provide access to a DRAM bank.



2. Configure the memory item in the Memory Properties dialog box. Click **OK**. The memory item is now populated in the Project Explorer window under the target.

   Note If you use a Memory Method Node in a single-cycle Timed Loop, make sure the corresponding arbitration option is **Arbitrate if Multiple Requestors Only** or **Never Arbitrate**.

3. Use the memory item in an FPGA VI.

Configuring DRAM with Socketed CLIP

Use the socketed CLIP interface to communicate directly with the onboard DRAM. Socketed CLIP lists all memory interfaces that are compatible with the selected DRAM bank item.

Note Use the latest CLIP versions to access the newest features and to automatically disable the synchronization registers.

Complete the following steps to configure DRAM with socketed CLIP.

1. Right-click an FPGA target in the Project Explorer window and select **Properties** from the shortcut menu to display the FPGA Target Properties dialog box.
2. Select **DRAM Properties** to display the DRAM Properties page.
3. In the **Mode** drop-down list, select **Socketed CLIP**.
4. Click **OK**. The new DRAM Bank is populated in the Project Explorer window under the target.
5. Right-click **DRAM Bank x** and select **Properties** to display the DRAM Bank x Properties dialog box.
6. Select **Enable DRAM** if it is not already selected to display the DRAM configuration options.
7. Configure the DRAM bank in the DRAM Bank x Properties dialog box and click **OK** when you are done. The DRAM signals are populated in the Project Explorer window under **DRAM Bank x**.

Correct DRAM Access Size and Frequency

The access size is the amount of information stored in one memory address. You can set up memory to use a variety of data types. To achieve the best performance and to utilize the maximum amount of data, use a data type that matches the access size of the device. The access size is the exact number of bits that are written and read in a given memory access.

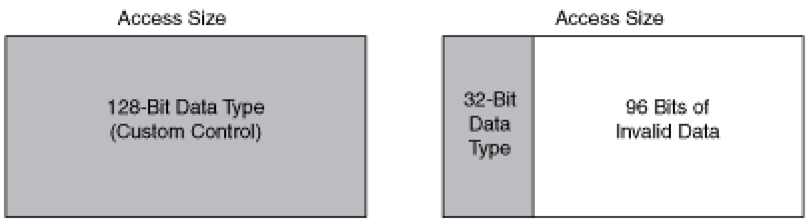
The following table shows the specifications for FlexRIO targets that support FPGA-accessible DRAM, including the optimum access size.

Note The PXI-7951, PXIe-7961, PXIe-7971, and PXIe-791x do not have DRAM.

FlexRIO Target	Number of DRAM Banks	Size per Bank	Bandwidth per Bank	Access Size	Optimal Clock Rate
PXI-795x	2	64 MB	800 MB/s	64 bits	100 MHz
PXI-796x	2	256 MB	1.6 GB/s	128 bits	100 MHz
PXI-797x	1	2 GB	10.5 GB/s	512 bits	166 MHz
NI-793x	1	2 GB	10.5 GB/s	512 bits	166 MHz
PXIe-791x	2	2 GB	10.5 GB/s	512 bits	166 MHz

If you use a data type that is smaller than the access size, the remaining bits receive an unknown and invalid value, but still get written and take up both space and bandwidth. For example, if the access size is 128 bits wide and a 32-bit data type is chosen during configuration of the DRAM, there will be 96 bits of unknown and invalid data.

The following figure shows an optimized memory element and a memory element where the data type is smaller than the access size.



NI recommends using a data type that is the same width as the access size of the DRAM to optimize each access. Memory items accept a cluster as a data type and information can be packaged into clusters to achieve data types larger than those native to LabVIEW.

You can maximize bandwidth by pushing data into the memory item interface at the clock rate. NI recommends that you push data into the memory item interface within the Optimal Clock Rate specified in the table above.

Note Clock sources other than DRAM Clock can be used, although performance will not be optimized.

Disabling Synchronization Registers

The DRAM interface signals are synchronous to the DRAM interface clock. Synchronization registers cause a delay in sending and receiving data or commands to and from the DRAM interface. For proper device operation, you must disable all synchronization registers for all DRAM interface signals and all input signals. Always disable synchronization registers for synchronous interfaces when proper operation depends on no latency.

Note All NI PXI version 1.1 and later CLIP items and all NI PXI Express CLIP items automatically disable all synchronization registers

1. Right-click a DRAM interface signal and select **Properties** from the shortcut menu to open the FPGA I/O Properties dialog box.
2. Select **Advanced Code Generation** in the Category list to open the Advanced Code Generation page.
3. Select **0** in the Number of Synchronizing Registers for Output Data box to disable all synchronization registers for that signal.

DRAM Interfaces

Some FlexRIO devices contain onboard DRAM that is directly accessible from the FPGA VI. LabVIEW supports two types of DRAM interfaces: FPGA memory items and socketed CLIP. You cannot use both FPGA memory items and socketed CLIP to access the same DRAM bank in an FPGA VI. Refer to your FPGA product's **Specifications** document to determine the available amount of onboard DRAM for your device.

DRAM Memory Interface

FlexRIO devices have different memory interfaces that provide access to the external DRAM memory. When you enable the DRAM memory bank in the DRAM General Properties dialog box, any compatible memory interfaces are automatically displayed in the Memory Interface window.

The following table lists all the NI-developed memory interfaces that are compatible with the external DRAM memory banks on FlexRIO devices, and it also gives a basic overview of their functionality.

Memory Interface Name	Memory Interface Description
Random Access - 64 Bit	Provides a high-performance random access interface to external memory. Use this interface for maximum performance or if you require a random access-based memory interface. The data port is 64 bits wide. This memory interface is supported only on FlexRIO PXI devices (PXI-795x).
FIFO - 64 Bit	Provides a FIFO-based interface to external memory. When using this memory interface, the entire memory bank is treated as a large FIFO. Use this interface if your throughput application requires a straightforward, FIFO-based memory interface. The data port is 64 bits wide. This memory interface is supported only on FlexRIO PXI devices (PXI-795x).
Random Access - 128 Bit	Provides a high-performance random access interface to external memory. Use this for maximum performance or if you require a random access-based memory interface. The data port is 128 bits wide. This memory interface is supported only on FlexRIO PXI Express devices (PXIe-796x and PXIe-797x).
FIFO - 128 Bit	Provides a FIFO-based interface to external memory. When using this memory interface, the entire memory bank is treated as a large FIFO. Use this interface if your throughput application

Memory Interface Name	Memory Interface Description
	requires a straightforward, FIFO-based memory interface. The data port is 128 bits wide. This memory interface is supported only on FlexRIO PXI Express devices (PXIe-796x and PXIe-797x).

Memory Items Per Bank

Memory items that are assigned to the same bank cannot access the DRAM simultaneously. Access to the DRAM is controlled by a grant time that dictates when a memory item can read or write to the DRAM bank, and for how long. By default, a DRAM bank assigns grant times of 50 cycles per memory item. You can modify the grant times in the DRAM Properties page of the FPGA target.

Note You must configure grant times at compile time; grant times cannot be modified during run time.

The ideal number of memory items per bank is 1. Having more than one memory item per bank prevents either item from accessing the DRAM with the most efficient bandwidth and latency, since memory items have to share access to the DRAM as dictated by the grant time.

Request Pipelining

There is a long latency between data requests and the execution of the requests due to the pipeline of the DRAM architecture.

Therefore, NI recommends creating your design with a “look ahead” approach. Keep requesting the data you wish to read, without waiting for the retrieve method's data valid strobe. The requests for reads and writes get stored in a queue inside the memory item and then passed on to the memory, which also has an internal queue where it picks out commands in order. Even though each individual request is still subject to latency and some non-determinism, you now get much higher transfer rates because DRAM can access several pieces of data sequentially instead of treating each request separately.

Sequential Data Access

DRAM is optimized for high storage density and high bandwidth. DRAM accesses data sequentially and in large blocks. For example, you have to read the data in address 0x1 after you have read the data in address 0x0 and the processor reads large blocks of memory into cache, even if the program being executed requests a single byte.

To maximize performance, avoid switching between reading and writing, accessing noncontiguous addresses, or writing to memory in decrementing-address fashion. The most efficient access strategy is to perform only one type of access, either reading or writing, on a large number of sequential addresses. Although this is optimal, it is not practical for most applications. A more practical approach is to maximize the amount of sequential data being accessed and minimizing changes in access modes.

The Process VI in the Memory IDL has an arbiter configuration input that allows you to dynamically alter the number of sequential reads and writes (write grant time and read grant time). Use this functionality to dedicate 100% of DRAM bandwidth to reading only, writing only, or equally sharing bandwidth between reads and writes.

FIFO - 64 Bit Memory Interface

This memory interface provides the easiest-to-use FIFO interface to external DRAM. The FIFO is exposed as separate read and write interfaces. Both the write-side and read-side data ports are 64-bit data words.

Signals for Writing to the FIFO

The following table lists the write-side I/O provided by the FIFO - 64 Bit memory interface.

Memory Interface I/O	Data Type	To Memory/From Memory	Description
Full	Bool	From memory	TRUE = Indicates that the FIFO is full and that no data can be written to it. FALSE = Indicates that the FIFO is not full and that data can written to it.
Write_Data_Upper	U32	To memory	Sets the upper 32 bits of the data to write into the FIFO.
Write_Data_Lower	U32	To memory	Sets the lower 32 bits of the data to write into the FIFO.
Write	Bool	To memory	TRUE = Transfers data from the Write_Data_Upper and Write_Data_Lower signals into the FIFO.

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			FALSE = Transfers no data to the FIFO. **Note** Do not set the Write signal to TRUE when the Full signal is TRUE.

Signals for Reading from the FIFO

The following table lists the read-side I/O provided by the FIFO - 64 Bit memory interface.

Memory Interface I/O	Data Type	To Memory/From Memory	Description
Data_Available	Bool	From memory	TRUE = Indicates that the values in the Read_Data_Upper and Read_Data_Lower signals are valid and may be read. FALSE = Indicates that the values in the Read_Data_Upper and Read_Data_Lower signals are invalid and may not be read yet.
Read_Data_Upper	U32	From memory	Displays the upper 32 bits of the data to read

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			from the FIFO. This signal is valid only when the Data_Available signal is TRUE.
Read_Data_Lower	U32	From memory	Displays the lower 32 bits of the data to read from the FIFO. This signal is valid only when the Data_Available signal is TRUE.
Read	Bool	To memory	TRUE = Reads data from the Read_Data_Upper and Read_Data_Lower signals out of the FIFO. FALSE = Does not read any data from the FIFO. **Note** Do not set the Read signal to TRUE when the Data_Available signal is FALSE.

The write-side interface Full signal is TRUE when data cannot be added into the FIFO. If the Full signal is low, you can write data into the FIFO by driving the Write_Data_Upper and Write_Data_Lower signals with your data and setting the Write signal to TRUE. Do not set the Write signal to TRUE when Full is TRUE because this condition leads to undefined behavior.

When read-side data on the Read_Data_Upper and Read_Data_Lower lines is available to be read, the Data_Available signal is TRUE. After this data is read, set the Read signal to TRUE. Do not set the Read signal to TRUE when Data_Available is FALSE because this condition leads to undefined behavior.

The FIFO - 64 Bit memory interface requires you to define the clock domains of the single-cycle Timed Loop from which the read-side and write-side interfaces are accessed. Configure these interfaces using the Clock Selections property page for the corresponding DRAM bank item in the LabVIEW project. Configure the Input_Clock signal to be driven with the LabVIEW FPGA clock resource which you are using to clock the single-cycle Timed Loop that is accessing the write-side interface. Likewise, configure the Output_Clock signal to be driven with the LabVIEW FPGA clock resource that you are using to clock the single-cycle Timed Loop that is accessing the read-side interface.

Note All memory interface signals must be used in a single-cycle Timed Loop. The clocks of this single-cycle Timed Loop must be specified as input clocks in the FPGA Base Clock Properties dialog box.

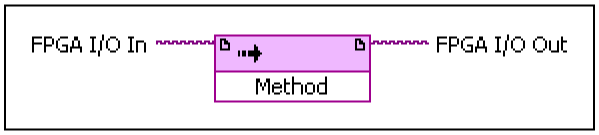
Note You must disable all synchronization registers for all DRAM FPGA I/O signals (by setting the value to 0). The DRAM interface signals are synchronous to the DRAM interface clock. Synchronization registers causes a delay in sending and receiving data/commands to/from the DRAM interface. This delay can prevent proper operation. Always disable synchronization registers for synchronous interfaces where proper operation depends on no latency. All NI PXI version 1.1 and later CLIP items and all NI PXI Express CLIP items automatically disable all synchronization registers. For more information about disabling synchronization registers, refer to the Advanced Code Generation FPGA I/O Properties Page.

Note Depending on clock rates and your application, it may take several clock cycles for data input to the FIFO to be read from the FIFO.

The FIFO - 64 Bit memory interface is designed to simultaneously read and write data at speeds up to 40 MHz. It is possible to run the interface up to 200 MHz. However, at speeds greater than 40 MHz, the Full and Data_Available signals may temporarily become TRUE and FALSE, respectively.

LabVIEW contains example VIs that read and write to the FIFO - 64 Bit memory interface. To access the NI Example Finder, open LabVIEW and select **Help** » **ind Examples**, then select **Hardware Input** and **Output** » **FlexRIO**. You can also access device-specific examples by selecting **Add device** from the **Hardware** pull-down menu in the NI Example Finder.

FPGA I/O Method Node



Use the FPGA I/O Method Node to invoke a method on an I/O item or hardware under an FPGA target. The I/O Method Node is located on the FPGA I/O Functions palette. You can use the following methods with this device.

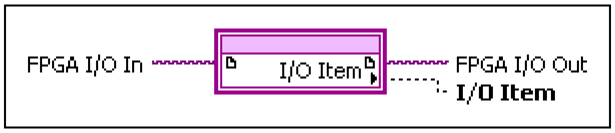
Note The following methods apply only to the fixed I/O that automatically associates with the installed NI PXI-795x target.

Method	Description
Set Output Data	Writes data to the digital line or port without enabling the line or port. You can use the Set Output Data method to optimize performance when performing successive writes to a DIO resource. The data type of the **Data** input depends on the I/O item. For example, if the I/O item is a digital line, Data requires a Boolean data type.
Set Output Enable	Determines whether the digital input and output resource reads external input or writes output. Wiring TRUE to Set Output Enable for a digital line allows the resource to write data. Wiring FALSE to Set Output Enable allows the resource to read external data. The data type of the **Enable** input depends on the I/O item. For example, if the I/O item is a digital line, **Enable** requires a Boolean data type, and if the I/O item is an 8-bit digital port, **Enable** requires a U8 data type. The binary values of the U8 input correspond to the individual lines of the digital

Method	Description
	port. Zeros correspond to FALSE inputs and ones correspond to TRUE inputs.
Wait on Any Edge	Pauses the execution of the I/O Method Node until the next falling or rising edge of the digital signal. The Timeout input on this method specifies the number of FPGA clock cycles that the Wait on Any Edge method waits for the next falling or rising edge. 0 = Causes the method to timeout immediately. Negative value = Causes the method to wait indefinitely. Positive value = Causes the method to wait for that number of clock cycles before timing out.
Wait on Falling Edge	Pauses the execution of the I/O Method Node until the next falling edge of the digital signal. The Timeout input on this method specifies the number of FPGA clock cycles that the Wait on Falling Edge method waits for the next falling edge. 0 = Causes the method to timeout immediately. Negative value = Causes the method to wait indefinitely. Positive value = Causes the method to wait for that number of clock cycles before timing out.
Wait on High Level	Pauses the execution of the I/O Method Node until the digital signal is high. The Timeout input on this method specifies the number of FPGA clock cycles that the Wait on High Level method waits for the next logic high level.

Method	Description
	0 = Causes the method to timeout immediately. Negative value = Causes the method to wait indefinitely. Positive value = Causes the method to wait for that number of clock cycles before timing out.
Wait on Low Level	Pauses the execution of the I/O Method Node until the digital signal is low. The Timeout input on this method specifies the number of FPGA clock cycles that the Wait on Low Level method waits for the next logic low level. 0 = Causes the method to timeout immediately. Negative value = Causes the method to wait indefinitely. Positive value = Causes the method to wait for that number of clock cycles before timing out.
Wait on Rising Edge	Pauses the execution of the I/O Method Node until the next rising edge of the digital signal. The Timeout input on this method specifies the number of FPGA clock cycles that the Wait on Rising Edge method waits for the next rising edge. 0 = Causes the method to timeout immediately. Negative value = Causes the method to wait indefinitely.

PXI FPGA I/O Items

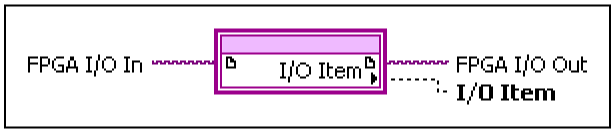


You can use an FPGA I/O Node, configured for reading and writing, with this device. The FPGA I/O Node is located on the FPGA I/O Functions palette and performs specific FPGA I/O operations on your FPGA target. You can select the following PXI FPGA I/O items for this device.

Table 2. PXI FPGA I/O Items

FPGA I/O Item	Description
PXI_CLK10	Controls the 10 MHz clock on the PXI backplane. You can use this clock to synchronize multiple PXI modules. Use an FPGA I/O Node configured for reading to access this channel.
PXI_Star	Controls the PXI star trigger line. Use an FPGA I/O Node configured for reading to access this channel.
PXI_Trigx	Controls PXI trigger channel x, where x is a PXI trigger number <0..7>. Use an FPGA I/O Node configured for reading or writing, or use the Set Output Data or Set Output Enable methods to access this channel. Follow the guidelines for using PXI triggers with the LabVIEW FPGA Module for proper device functionality.

PXI/PXIe FPGA I/O Items



You can use an FPGA I/O Node, configured for reading and writing, with this device. The FPGA I/O Node is located on the FPGA I/O Functions palette and performs specific FPGA I/O operations on your FPGA target. You can select the following PXI/PXI Express FPGA I/O items for this device.

Table 3. PXI/PXIe FPGA I/O Items

FPGA I/O Item	Description
PXI_Clk10	Controls the 10 MHz clock on the PXI backplane. You can use this clock to synchronize multiple PXI modules. Use an FPGA I/O Node configured for reading to access this channel.

Table 4. PXI Express FPGA I/O Items

Terminal	Description
PXIe_DStarB	Controls the differential star trigger that creates connections between a PXI Express system timing module and a peripheral device. PXIe_DStarB distributes trigger signals from the system timing slot to the peripherals (input). Use an FPGA I/O Node configured for reading to access this channel.
PXIe_DStarC	Controls the differential star trigger that creates connections between a PXI Express system timing module and a peripheral device. PXIe_DStarC sends trigger or clock signals from the peripherals to the system timing slot (output). Use an FPGA I/O Node configured for writing to access this channel.
PXIe_Sync100	Controls the reference signal that is synchronous to the rising edge of PXIe_CLK100.

Terminal	Description
	Use PXIe_Sync100 to synchronize multiple modules in a PXI Express system. Use an FPGA I/O Node configured for reading to access this channel.

Supported PXI Express FPGA Module Functionality

- Arbitration—This device supports arbitration to determine which object can access the resource if multiple objects request access at the same time. Configure the arbitration settings for the channels of this device in the FPGA I/O Properties dialog box for the FPGA I/O item you are using.
- FPGA Base Clocks—This device supports FPGA base clocks. For more information about the available base clock resources, refer to the **PXIe-796x Base Clocks** topic.
- Single-Cycle Timed Loop—This device supports the single-cycle Timed Loop for the implementation of multiple clock domains in an FPGA VI.
- Peer-to-Peer Streaming—This device supports peer-to-peer data streaming. Use LabVIEW FPGA to configure your FlexRIO device to set up a peer-to-peer streaming session.
- FlexRIO Peer-to-Peer Resource Availability—FlexRIO PXI Express devices support both Peer-to-Peer Reader and Peer-to-Peer Writer streams. You can configure a total of 16 streams. The following table lists the stream types available for FlexRIO PXI Express devices.

FIFO Type	Description
DMA input	Reads data into the FIFO from the VI.
DMA output	Writes data from the FIFO into the VI.
P2P Writer	Writes data to the peer-to-peer reader device.
P2P Reader	Reads data from the peer-to-peer writer device.

Supported PXI Express FPGA Module Functionality

- Arbitration—This device supports arbitration to determine which object can access the resource if multiple objects request access at the same time. Configure the arbitration settings for the channels of this device in the FPGA I/O Properties dialog box for the FPGA I/O item you are using.
- FPGA Base Clocks—This device supports FPGA base clocks. For more information about the available base clock resources, refer to the **PXIe-797x Base Clocks** topic.
- Single-Cycle Timed Loop—This device supports the single-cycle Timed Loop for the implementation of multiple clock domains in an FPGA VI.
- Peer-to-Peer Streaming—This device supports peer-to-peer data streaming. Use LabVIEW FPGA to configure your FlexRIO device to set up a peer-to-peer streaming session.
- FlexRIO Peer-to-Peer Resource Availability—FlexRIO PXI Express devices support both Peer-to-Peer Reader and Peer-to-Peer Writer streams. The following table lists the stream types available for FlexRIO PXI Express devices.

FIFO Type	Description
DMA input	Reads data into the FIFO from the VI.
DMA output	Writes data from the FIFO into the VI.
P2P Writer	Writes data to the peer-to-peer reader device.
P2P Reader	Reads data from the peer-to-peer writer device.

Random Access - 128 Bit Memory Interface

This memory interface provides the highest performance interface to external DRAM. Both the write-side and read-side data ports are exposed as 128-bit data words. Requests to write to and read from DRAM are satisfied by issuing a write or read command to the Random Access - 128 Bit memory interface.

The following table lists the I/O provided by the Random Access - 128 Bit memory interface.

Memory Interface I/O	Data Type	To Memory/From Memory	Description
Address	U64	To memory	Sets the address in external memory for reading or writing. The physical data bus for external memory is 32 bits wide (4 bytes). Each unique address value represents 4 bytes of data. Therefore, the total number of unique addresses in external memory is equal to (Memory Size in bytes)/4. **Note** The memory interface exposed to LabVIEW FPGA is 128 bits wide. As a result, each memory

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			write or read operation accesses four different address locations in memory. The memory controller latches this signal value only when you issue a new memory write command by asserting the Command_Write_Enable signal.
Command	U8	To memory	0 = Performs a memory write. 1 = Perform a memory read. The memory controller latches this signal value only when you issue a new memory write command by asserting the

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			Command_Write_Enable signal.
Command_Write_Enable	Bool	From memory	TRUE = Performs either a memory write or read command, as dictated by the Command signal setting. FALSE = Does not perform any new memory write or read commands. To write or read a single 128-bit data value, assert this signal for only one clock cycle. Each new command is latched in a single clock cycle.
Command_FIFO_Full	Bool	From memory	TRUE = The memory controller internal command FIFO is full. FALSE = Space is available in the command FIFO, and the memory controller can accept new commands. **Note** Never assert the Command_Write_Enable signal

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			when the Command_ FIFO_Full signal is TRUE.
Read_Data_Upper	U64	From memory	Displays the upper 64 bits of the 128-bit data value that was read from external memory. When you issue a memory read command, this signal contains the data retrieved from external memory. The signal state is valid only on the clock cycles in which Read_Strobe is TRUE.
Read_Data_Lower	U64	From memory	Displays the lower 64 bits of the 128-bit data value that was read from external memory. When you issue a memory read command, this signal contains the data retrieved from external memory. The signal state is valid only on the clock cycles in which Read_Strobe is TRUE.
Read_Strobe	Bool	From memory	TRUE = Indicates that you can latch data onto the Read_Data_Upper

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			and Read_Data_Lower signals. FALSE = Indicates that you cannot latch data onto the Read_Data_Upper and Read_Data_Lower signals yet. When you issue a memory read command, the memory read operation takes more than one clock cycle to complete. When the memory read completes, the Read_Strobe signal asserts TRUE for one clock cycle. **Note** You must latch the Read_Data_Upper and Read_Data_Lower signals on the same clock cycle that Read_Strobe asserts.
Write_Data_Upper	U64	To memory	Sets the upper 64 bits of the 128-bit data value to be written to

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			external memory. The memory controller latches this signal value only when you issue a new memory write command by asserting the Command_Write_Enab le signal.
Write_Data_Lower	U64	To memory	Sets the lower 64 bits of the 128-bit data value to be written to external memory. The memory controller latches this signal value only when you issue a new memory write command by asserting the Command_Write_Enab le signal.
Data Mask Lower			
Data Mask Upper			
Initialization_Done	Bool	From memory	TRUE = Indicates that the memory interface initialization sequence is completed. FALSE = Indicates that the memory interface initialization sequence is not completed. The memory controller performs its initialization sequence each time the FPGA is reprogrammed, and when the Reset Invoke

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			Method function is called in the host VI. **Note** Never assert Command_Write_Enable when the Initialization_Done signal is FALSE.

Each command that is written to the Random Access - 128 Bit memory interface is added to an internal command FIFO. This command FIFO can fill up, which drives the Command_FIFO_Full signal to TRUE. Do not issue any new commands to the Random Access - 128 Bit memory interface when the command FIFO is full.

The Random Access - 128 Bit memory interface automatically performs an initialization sequence after the FPGA is first programmed, and when the Reset Invoke Method function is called in the host VI. Do not issue new commands to the Random Access - 128 Bit memory interface before the initialization sequence is completed. The Initialization_Done signal returns to TRUE when the Random Access - 128 Bit memory interface initialization sequence is completed.

Each individual address value accesses 32 bits of data. Each write and read command writes and reads 128 bits of data, respectively. Because of this scheme, writing and reading data at consecutive addresses requires that you increment the address by 4 for each write and read command. Also, because each address value accesses 32 bits of data, the maximum number of valid addresses is equal to (Memory Size in Bytes)/4.

The Random Access - 128 Bit memory interface requires that you define the clock domain of the single-cycle Timed Loop from which it is being accessed. You can configure the clock domain in the Clock Selections property page for the corresponding DRAM bank item in the LabVIEW project. Configure the User_Clk signal to be driven with the LabVIEW FPGA clock resource that you are using to clock the single-cycle Timed Loop that is accessing the Random Access - 128 Bit memory interface I/O. All DRAM interface I/O added to the LabVIEW FPGA project by the Random Access - 128 Bit memory interface are synchronous to this User_Clk.

Note All memory interface signals must be used in a single-cycle Timed Loop.

Note You must disable all synchronization registers for all DRAM FPGA I/O signals (by setting the value to 0). The DRAM interface signals are synchronous to the DRAM interface clock. Synchronization registers causes a delay in sending and receiving data/commands to/from the DRAM interface. This delay can prevent proper operation. Always disable synchronization registers for synchronous interfaces where proper operation depends on no latency. All NI PXI version 1.1 and later CLIP items and all NI PXI Express CLIP items automatically disable all synchronization registers. For more information about disabling synchronization registers, refer to the Advanced Code Generation FPGA I/O Properties Page.

Random Access - 64 Bit Memory Interface

This memory interface provides the highest performance interface to external DRAM. Both the write-side and read-side data ports are exposed as 64-bit data words. Requests to write to and read from DRAM are satisfied by issuing a write or read command to the Random Access - 64 Bit memory interface.

The following table lists the I/O provided by the Random Access - 64 Bit memory interface.

Memory Interface I/O	Data Type	To Memory/From Memory	Description
Address	U32	To memory	Sets the address in external memory for reading or writing. The physical data bus for external memory is 16 bits wide (2 bytes). Each unique address value represents 2 bytes of data. Therefore, the total number of unique addresses in external memory is equal to (Memory Size in bytes)/2. **Note** The memory interface exposed to LabVIEW FPGA is 64 bits wide. As a result, each memory

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			write or read operation accesses four different address locations in memory. The memory controller latches this signal value only when you issue a new memory write command by asserting the Command_Write_Enable signal.
Command	U8	To memory	0 = Performs a memory write. 1 = Perform a memory read. The memory controller latches this signal value only when you issue a new memory write command by asserting the

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			Command_Write_Enable signal.
Command_Write_Enable	Bool	From memory	TRUE = Performs either a memory write or read command, as dictated by the Command signal setting. FALSE = Does not perform any new memory write or read commands. To write or read a single 64-bit data value, assert this signal for only one clock cycle. Each new command is latched in a single clock cycle.
Command_FIFO_Full	Bool	From memory	TRUE = The memory controller internal command FIFO is full. FALSE = Space is available in the command FIFO, and the memory controller can accept new commands. **Note** Never assert the Command_Write_Enable signal

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			when the Command_ FIFO_Full signal is TRUE.
Read_Data_Upper	U32	From memory	Displays the upper 32 bits of the 64-bit data value that was read from external memory. When you issue a memory read command, this signal contains the data retrieved from external memory. The signal state is valid only on the clock cycles in which Read_Strobe is TRUE.
Read_Data_Lower	U32	From memory	Displays the lower 32 bits of the 64-bit data value that was read from external memory. When you issue a memory read command, this signal contains the data retrieved from external memory. The signal state is valid only on the clock cycles in which Read_Strobe is TRUE.
Read_Strobe	Bool	From memory	TRUE = Indicates that you can latch data onto the Read_Data_Upper

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			and Read_Data_Lower signals. FALSE = Indicates that you cannot latch data onto the Read_Data_Upper and Read_Data_Lower signals yet. When you issue a memory read command, the memory read operation takes more than one clock cycle to complete. When the memory read completes, the Read_Strobe signal asserts TRUE for one clock cycle. **Note** You must latch the Read_Data_Upper and Read_Data_Lower signals on the same clock cycle that Read_Strobe asserts.
Write_Data_Upper	U32	To memory	Sets the upper 32 bits of the 64-bit data value to be written to

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			external memory. The memory controller latches this signal value only when you issue a new memory write command by asserting the Command_Write_Enable signal.
Write_Data_Lower	U32	To memory	Sets the lower 32 bits of the 64-bit data value to be written to external memory. The memory controller latches this signal value only when you issue a new memory write command by asserting the Command_Write_Enable signal.
Data Mask			
Initialization_Done	Bool	From memory	TRUE = Indicates that the memory interface initialization sequence is completed. FALSE = Indicates that the memory interface initialization sequence is not completed. The memory controller performs its initialization sequence each time the FPGA is reprogrammed, and when the Reset Invoke

Memory Interface I/O	Data Type	To Memory/From Memory	Description
			Method function is called in the host VI. **Note** Never assert Command_Write_Enable when the Initialization_Done signal is FALSE.

Each command that is written to the Random Access - 64 Bit memory interface is added to an internal command FIFO. This command FIFO can fill up, which drives the Command_FIFO_Full signal to TRUE. Do not issue any new commands to the Random Access - 64 Bit memory interface when the command FIFO is full.

The Random Access - 64 Bit memory interface automatically performs an initialization sequence after the FPGA is first programmed, and when the Reset Invoke Method function is called in the host VI. Do not issue new commands to the Random Access - 64 Bit memory interface before the initialization sequence is completed. The Initialization_Done signal returns to TRUE when the Random Access - 64 Bit memory interface initialization sequence is completed.

Each individual address value accesses 16 bits of data. Each write and read command writes and reads 64 bits of data, respectively. Because of this scheme, writing and reading data at consecutive addresses requires that you increment the address by 4 for each write and read command. Also, because each address value accesses 16 bits of data, the maximum number of valid addresses is equal to (Memory Size in Bytes)/2.

The Random Access - 64 Bit memory interface requires that you define the clock domain of the single-cycle Timed Loop from which it is being accessed. You can configure the clock domain in the Clock Selections property page for the corresponding DRAM bank item in the LabVIEW project. Configure the User_Clk signal to be driven with the LabVIEW FPGA clock resource that you are using to clock the single-cycle Timed Loop that is accessing the Random Access - 64 Bit memory interface I/O. All DRAM interface I/O added to the LabVIEW FPGA project by the Random Access - 64 Bit memory interface are synchronous to this User_Clk.

Note All memory interface signals must be used in a single-cycle Timed Loop.

Note You must disable all synchronization registers for all DRAM FPGA I/O signals (by setting the value to 0). The DRAM interface signals are synchronous to the DRAM interface clock. Synchronization registers causes a delay in sending and receiving data/commands to/from the DRAM interface. This delay can prevent proper operation. Always disable synchronization registers for synchronous interfaces where proper operation depends on no latency. All NI PXI version 1.1 and later CLIP items and all NI PXI Express CLIP items automatically disable all synchronization registers. For more information about disabling synchronization registers, refer to the Advanced Code Generation FPGA I/O Properties Page.

Reading Data from DRAM

Complete the following steps to read data from DRAM.

LabVIEW contains example VIs that read and write to the Random Access - 128 Bit memory interface. To access the NI Example Finder, open LabVIEW and select **Help** » **Find Examples**, then select **Hardware Input and Output** » **FlexRIO**. You can also access device-specific examples by selecting **Add device** from the **Hardware** pull-down menu in the NI Example Finder.

Note The Random Access - 128 Bit memory interface queues read requests internally. Therefore, you may issue multiple read requests for different addresses before you even receive the data back for your first read request. As a result, if you are queuing up multiple read requests, it is important that you properly track the order of the data that is returned when Read_Strobe asserts. All data returned from the Random Access - 128 Bit memory interface when Read_Strobe asserts is properly ordered with respect to the original read request command ordering.

1. Ensure that the Command_FIFO_Full Boolean is FALSE.
2. Drive the Address signal with the DRAM memory address that you want to read from.
3. Drive the Command signal with 1. This signal communicates to the Random Access - 128 Bit memory interface to perform a memory read command.
4. Drive the Command_Write_Enable signal to TRUE for one clock cycle. You must only drive this signal for a single clock cycle if you want to only issue a single memory read command. Each clock cycle in which this signal is asserted causes a new read command to be issued to the Random Access - 128 Bit memory interface.
5. Continually monitor the state of the Read_Strobe signal. When this signal reads as TRUE, latch the 128-bit data value on the Read_Data_Upper and Read_Data_Lower signals. This contains the data most recently read from the Random Access - 128 Bit memory interface.

Reading Data from DRAM

Complete the following steps to read data from DRAM.

LabVIEW contains example VIs that read and write to the Random Access - 64 Bit memory interface. To access the NI Example Finder, open LabVIEW and select **Help** » **Find Examples**, then select **Hardware Input and Output** » **FlexRIO**. You can also access device-specific examples by selecting **Add device** from the **Hardware** pull-down menu in the NI Example Finder.

Note The Random Access - 64 Bit memory interface queues read requests internally. Therefore, you may issue multiple read requests for different addresses before you even receive the data back for your first read request. As a result, if you are queuing up multiple read requests, it is important that you properly track the order of the data that is returned when Read_Strobe asserts. All data returned from the Random Access - 64 Bit memory interface when Read_Strobe asserts is properly ordered with respect to the original read request command ordering.

1. Ensure that the Command_FIFO_Full Boolean is FALSE.
2. Drive the Address signal with the DRAM memory address that you want to read from.
3. Drive the Command signal with 1. This signal communicates to the Random Access - 64 Bit memory interface to perform a memory read command.
4. Drive the Command_Write_Enable signal to TRUE for one clock cycle. You must only drive this signal for a single clock cycle if you want to only issue a single memory read command. Each clock cycle in which this signal is asserted causes a new read command to be issued to the Random Access - 64 Bit memory interface.
5. Continually monitor the state of the Read_Strobe signal. When this signal reads as TRUE, latch the 64-bit data value on the Read_Data_Upper and Read_Data_Lower signals. This contains the data most recently read from the Random Access - 64 Bit memory interface.

Supported PXI FPGA Module Functionality

- Arbitration—This device supports arbitration to determine which object can access the resource if multiple objects request access at the same time. Configure the arbitration settings for the channels of this device in the FPGA I/O Properties dialog box for the FPGA I/O item you are using.
- FPGA Base Clocks—This device supports FPGA base clocks. For more information about the available base clock resources, refer to the **PXI-795x Base Clocks** topic.
- Single-Cycle Timed Loop—This device supports the single-cycle Timed Loop for the implementation of multiple clock domains in an FPGA VI.

Writing Data into DRAM

Complete the following steps to write data into the DRAM.

LabVIEW contains example VIs that read and write to the Random Access - 64 Bit memory interface. To access the NI Example Finder, open LabVIEW and select **Help** » **Find Examples**, then select **Hardware Input and Output** » **FlexRIO**. You can also access device-specific examples by selecting **Add device** from the **Hardware** pull-down menu in the NI Example Finder.

1. Ensure that the Command_FIFO_Full Boolean is FALSE.
2. Drive the Address signal with the DRAM memory address that you want to write.
3. Drive the Write_Data_Upper with the upper 64 bits of the 128-bit data word.
4. Drive the Write_Data_Lower with the lower 64 bits of the 128-bit data word.
5. Drive the Command signal with 0. This communicates to the Random Access - 128 Bit memory interface to perform a memory write command.
6. Drive the Command_Write_Enable signal to TRUE for one clock cycle. You must only drive this signal for a single clock cycle if you want to only issue a single memory write command. Each clock cycle that this signal asserts causes a new write command to be issued to the Random Access - 128 Bit memory interface.

Writing Data into DRAM

Complete the following steps to write data into the DRAM.

LabVIEW contains example VIs that read and write to the Random Access - 64 Bit memory interface. To access the NI Example Finder, open LabVIEW and select **Help** » **Find Examples**, then select **Hardware Input and Output** » **FlexRIO**. You can also access device-specific examples by selecting **Add device** from the **Hardware** pull-down menu in the NI Example Finder.

1. Ensure that the Command_FIFO_Full Boolean is FALSE.
2. Drive the Address signal with the DRAM memory address that you want to write.
3. Drive the Write_Data_Upper with the upper 32 bits of the 64-bit data word.
4. Drive the Write_Data_Lower with the lower 32 bits of the 64-bit data word.
5. Drive the Command signal with 0. This communicates to the Random Access - 64 Bit memory interface to perform a memory write command.
6. Drive the Command_Write_Enable signal to TRUE for one clock cycle. You must only drive this signal for a single clock cycle if you want to only issue a single memory write command. Each clock cycle that this signal asserts causes a new write command to be issued to the Random Access - 64 Bit memory interface.