SYSTEM DEVELOPMENT MANUAL

CompactRIO Single-Board Controller with NI-DAQmx

Use this document to do the following hardware tasks.

- Design a RIO mezzanine card

Use this document to do the following software tasks.

- Configure the single-board controller using LabVIEW
- Program the single-board controller using Real-Time Scan Interface, LabVIEW FPGA, and NI-DAQmx
- Customize BIOS settings
- Read and write external disk drives

Note For information about mounting, grounding, and connecting the single-board controller, refer to the *CompactRIO Single-Board Controller with DAQmx Hardware Installation Manual* on *ni.com/manuals*.

Table 1. CompactRIO Single-Board Controller with DAQmx Models

Digital I/O Models	Multifunction I/O Models
sbRIO-9603	sbRIO-9628
sbRIO-9608	sbRIO-9629
sbRIO-9609	sbRIO-9638

Contents

NATIONAL INSTRUMENTS

Table 2. System Development Manual Definitions

Term	Definition
Connector0	Connector0 refers to the MIO connector on the sbRIO-9628, sbRIO-9629, and sbRIO-9638.
Connector1	Connector1 refers to the DIO connector on the sbRIO-9638.
sbRIO-960x	In this document, sbRIO-960x refers exclusively to the following CompactRIO single-board controllers with NI-DAQmx: sbRIO-9603, sbRIO-9608, and sbRIO-9609.
sbRIO-962x/ sbRIO-963x	In this document, sbRIO-962x/sbRIO-963x refers exclusively to the following CompactRIO single-board controllers with NI-DAQmx: sbRIO-9628, sbRIO-9629, and sbRIO-9638.
sbRIO-96xx	In this document, sbRIO-96xx refers exclusively to the CompactRIO single-board controllers with NI-DAQmx.
C Series Module	C Series Module refers to any board-only C Series module designed for implementation in an sbRIO system.
Onboard I/O Module	Onboard module refers to the distinct onboard modules designed into the sbRIO-96xx.
sbRIO I/O Module	In this document, sbRIO I/O module refers to both onboard modules and C Series modules.

Table 3. RMC Design Concepts

Term	Definition
RMC connector	240-pin, 40 × 6 position, high-density open pin field SEARAY on the sbRIO device.
SEARAY	Connector family used for the RMC connector on the sbRIO device. Manufactured by Samtec.
SoC	System on Chip.
USB Host	USB interface that controls the bus and communicates with connected USB devices.
LVTTL	In compliance with the Low-Voltage Transistor-Transistor Logic (LVTTL) specification.

Table 3. RMC Design Concepts (Continued)

Term	Definition
LVCMOS	In compliance with the Low-Voltage Complementary Metal Oxide Semiconductor (LVCMOS) specification.
LVDS	In compliance with the Low-Voltage Differential Signalling (LVDS) specification.

Table 4. Schematic Conventions

Element	Definition
	Off-page symbol that represents communication to and from the mating connector.
	Off-page symbol that represents communication from the mating connector.
	Off-page symbol that represents communication to the mating connector.
	On-page symbol that represents the signal being driven.
	On-page symbol that represents the signal being received.
	Power supply rail.
	Analog ground.
	Digital ground.
	Chassis ground.
SPARE	Refers to an unpopulated reference designator.

sbRIO I/O Module

This document uses the term sbRIO I/O module to refer to any I/O implemented in an sbRIO system. This includes both Onboard I/O modules and C Series modules. See the descriptions below for an overview of programming and implementation differences.

Onboard I/O Module

The sbRIO-962x and sbRIO-963x feature onboard modules that provide additional analog I/O and digital I/O through Connector0 and Connector1. You can program this I/O in Real-Time (NI-DAQmx) or FPGA modes. You can not program Onboard I/O modules in Real-Time Scan mode.

C Series Module

The sbRIO-960x and sbRIO-962x feature a high-density RMC connector with digital I/O. This connector provides two dedicated C Series interfaces for use with board-only C Series modules. The sbRIO controller can access these C Series modules in three programming modes: Real-Time (NI-DAQmx), Real-Time Scan, and FPGA. You can implement the C Series interfaces with an NI-9697 or NI-9698 RMC breakout board or via a custom RMC daughter board. Refer to the *RMC Connector Design Guide* section of this document for more information and reference schematics.

RMC Connector Design Guide

Use this section to design and implement an RIO mezzanine card that communicates with the sbRIO-96xx. The RMC connector on the sbRIO-96xx is a high-density, high-throughput connector that features up to 96 single-ended DIO lines or up to 45 differential pairs directly connected to the FPGA with the ability to add up to two C Series modules and additional peripherals, including mass storage devices over PCIe or SATA. This section details RMC pins and their functions, reference schematics, PCB layout guidelines, and mechanical considerations.

RMC Connector Overview

RMC Connector Pins

The pins on the RMC connector are divided into the following groups:

- Pins with dedicated functions.
- General purpose digital I/O pins.
- Pins reserved for future use.

Note Leave reserved and unused pins disconnected on RMCs.

Figure 1. RMC Connector Pin Listing

1 VIN_Filtered	2 GND	3 PCIe_Gen2 TX+	4 GND	5 SATA_TX+	6 GND
7 VIN_Filtered	8 GND	9 PCIe_Gen2 TX-	10 GND	11 SATA_TX–	12 GND
13 GND	14 VIN_Filtered	15 GND	16 PCIe_Gen2 RX+	17 GND	18 SATA_RX+
19 GND	20 VIN_Filtered	21 GND	22 PCIe_Gen2 RX-	23 GND	24 SATA_RX–
25 Reserved	26 GND	27 Reserved	28 GND	29 USB_D+	30 GND
31 PCIe_Clk-	32 PCIe_Clk+	33 USB_CPEN	34 Reserved	35 USB_D–	36 GND
37 CLK_REQ#	38 RST#	39 GND	40 ID_SELECT#[1]	41 GND	42 Reserved (3.3 V)
43 SYS_RST#	44 GND	45 SLEEP[1]	46 CVRT#_DIO3[1]	47 GND	48 3.3 V
49 GND	50 DONE#_DIO2[1]	51 SLEEP[2]	52 GND	53 SPIFUNC_DIO4[1]	54 5 V
55 SPICS#_DIO5[1]	56 MOSI_DIO7[1]	57 GND	58 OSCLK_DIO0[1]	59 PROC_LED	60 5 V
61 SPI_CLK[1]	62 GND	63 ID_SELECT#[2]	64 TRIG_DIO1[1]	65 GND	66 5 V
67 GND	68 DONE#_DIO2[2]	69 CVRT#_DIO3[2]	70 GND	71 MISO_DIO6[1]	72 5 V
73 SPICS#_DIO5[2]	74 MOSI_DIO7[2]	75 GND	76 OSCLK_DIO0[2]	77 SPIFUNC_DIO4[2]	78 GND
79 SPI_CLK[2]	80 GND	81 MISO_DIO6[2]	82 TRIG_DIO1[2]	83 GND	84 Reserved
85 GND	86 5V C Series	87 Reserved	88 GND	89 DIO_47 (DIO_47_P)	90 DIO_15
91 5V C Series	92 DIO_63	93 GND	94 DIO_79 (DIO_79_P)	95 DIO_46 (DIO_47_N)	96 GND
97 DIO_95	98 GND	99 DIO_31 (DIO_31_P)	100 DIO_78 (DIO_79_N)	101 GND	102 DIO_14 (DIO_14_P)
103 GND	104 DIO_62 (DIO_62_P)	105 DIO_30 (DIO_31_N)	106 GND	107 DIO_45 (DIO_45_P)	108 DIO_13 (DIO_14_N)
109 DIO_94 (DIO_94_P)	110 DIO_61 (DIO_62_N)	111 GND	112 DIO_77 (DIO_77_P)	113 DIO_44 (DIO_45_N)	114 GND
115 DIO_93 (DIO_94_N)	116 GND	117 DIO_29 (DIO_29_P)	118 DIO_76 (DIO_77_N)	119 GND	120 DIO_12 (DIO_12_P)
121 GND	122 DIO_60 (DIO_60_P)	123 DIO_28 (DIO_29_N)	124 GND	125 DIO_43 (DIO_43_P)	126 DIO_11 (DIO_12_N)
127 DIO_92 (DIO_92_P)	128 DIO_59 (DIO_60_N)	129 GND	130 DIO_75 (DIO_75_P)	131 DIO_42 (DIO_43_N)	132 GND
133 DIO_91 (DIO_92_N)	134 GND	135 DIO_27 (DIO_27_P)	136 DIO_74 (DIO_75_N)	137 GND	138 DIO_10 (DIO_10_P)
139 GND	140 DIO_58 (DIO_58_P)	141 DIO_26 (DIO_27_N)	142 GND	143 DIO_41 (DIO_41_P)	144 DIO_09 (DIO_10_N)
145 DIO_90 (DIO_90_P)	146 DIO_57 (DIO_58_N)	147 GND	148 DIO_73 (DIO_73_P)	149 DIO_40 (DIO_41_N)	150 GND
151 DIO_89 (DIO_90_N)	152 GND	153 DIO_25 (DIO_25_P)	154 DIO_72 (DIO_73_N)	155 GND	156 DIO_08 (DIO_08_P)
157 GND	158 DIO_56 (DIO_56_P)	159 DIO_24 (DIO_25_N)	160 GND	161 DIO_39 (DIO_39_P)	162 DIO_07 (DIO_08_N)
163 DIO_88 (DIO_88_P)	164 DIO_55 (DIO_56_N)	165 GND	166 DIO_71 (DIO_71_P)	167 DIO_38 (DIO_39_N)	168 GND
169 DIO_87 (DIO_88_N)	170 GND	171 DIO_23 (DIO_23_P)	172 DIO_70 (DIO_71_N)	173 GND	174 DIO_06 (DIO_06_P)
175 GND	176 DIO_54 (DIO_54_P)	177 DIO_22 (DIO_23_N)	178 GND	179 DIO_37 (DIO_37_P)	180 DIO_05 (DIO_06_N)
181 DIO_86 (DIO_86_P)	182 DIO_53 (DIO_54_N)	183 GND	184 DIO_69 (DIO_69_P)	185 DIO_36 (DIO_37_N)	186 GND
187 DIO_85 (DIO_86_N)	188 GND	189 DIO_21 (DIO_21_P)	190 DIO_68 (DIO_69_N)	191 GND	192 DIO_04 (DIO_04_P)
193 GND	194 DIO_52 (DIO_52_P)	195 DIO_20 (DIO_21_N)	196 GND	197 DIO_35 (DIO_35_P)	198 DIO_03 (DIO_04_N)
199 DIO_84 (DIO_84_P)	200 DIO_51 (DIO_52_N)	201 GND	202 DIO_67 (DIO_67_P)	203 DIO_34 (DIO_35_N)	204 GND
205 DIO_83 (DIO_84_N)	206 GND	207 DIO_19 (DIO_19_P)	208 DIO_66 (DIO_67_N)	209 GND	210 DIO_02 (DIO_02_P)
211 GND	212 DIO_50 (DIO_50_P)	213 DIO_18 (DIO_19_N)	214 GND	215 DIO_33 (DIO_33_P)	216 DIO_01 (DIO_02_N)
217 DIO_82 (DIO_82_P)	218 DIO_49 (DIO_50_N)	219 GND	220 DIO_65 (DIO_65_P)	221 DIO_32 (DIO_33_N)	222 GND
223 DIO_81 (DIO_82_N)	224 GND	225 DIO_17 (DIO_17_P)	226 DIO_64 (DIO_65_N)	227 GND	228 DIO_00
229 GND	230 DIO_48	231 DIO_16 (DIO_17_N)	232 GND	233 Reserved	234 FPGA_VIO 47..0
235 DIO_80	236 VBAT	237 GND	238 Reserved	239 FPGA_CONF	240 FPGA_VIO 95..48

GND | C Series | FPGA I/O | FPGA Clock Optimized | Processor i/O | Power | Reserved

Use the RMC Connector Feature Set Compatibility table to determine if a previously designed RMC is compatible with the new RMC pinout and as guidance on how to design an RMC for compatibility with future generations of the RMC.

Table 5. RMC Connector Feature Set Compatibility

Feature Set		sbRIO-96x7 (Previous Generation)	sbRIO-96xx (CompactRIO Single-Board Controllers with NI-DAQmx)	Future Design Compatibility
DIO[0..63]		Yes	Yes	Yes
DIO[64..95]		Yes	Yes	Not guaranteed
FPGA_CONF		Yes	Yes	Yes
USB_D+/-		Yes	Yes	Yes
RST#		Yes	Yes	Yes
SYS_RST#		Yes	Yes	Yes
5V		Yes	Yes	Yes
3.3V		Yes	Yes	Yes
FPGA_VIO[1]		Yes	Yes	Yes
VBAT		Yes	Yes	Yes
Processor I/O via DIO[0..95]	CAN	Yes	No	Not guaranteed
	RS-232	Yes	Visit *ni.com/r/sbrio* for more information	Not guaranteed
	RS-485	Yes		Not guaranteed
	SDHC	Yes	No	Not guaranteed
GBE_MDI[0..3+/-]		Yes	No	Not guaranteed
USB_MODE, USB_VBUS		Yes	No	Not guaranteed
USB_CPEN		Yes	Yes	Not guaranteed
Dedicated C Series DIO		Yes	Yes	Not guaranteed
VIN_FILTERED		Yes	Yes	Yes
PCIe		No	Yes	No
SATA		No	Yes	No

[1] FPGA_VIO present when the FPGA is programmed.

Pins with Dedicated Functions

Table 6. RMC Connector Pins with Dedicated Functions

Pin Group	Pin Name	Pin Number	Direction (from Host System)	Description
Power (Output)	GND	2, 4, 6, 8, 10, 12, 13, 15, 17, 19, 21, 23, 26, 28, 30, 36, 39, 41, 44, 47, 49, 52, 57, 62, 65, 67, 70, 75, 78, 80, 83, 85, 88, 93, 96, 98, 101, 103, 106, 111, 114, 116, 119, 121, 124, 129, 132, 134, 137, 139, 142, 147, 150, 157, 160, 165, 168, 170, 173, 175, 178, 183, 186, 188, 191, 193, 196, 201, 204, 206, 209, 211, 214, 219, 222, 224, 227, 229, 232, 237	O	Digital ground from the RMC connector host system.

Table 6. RMC Connector Pins with Dedicated Functions (Continued)

Pin Group	Pin Name	Pin Number	Direction (from Host System)	Description
Power (Output)	3.3 V	48	O	3.3 V from the RMC connector host system. The rail is always on when the main host system is not in sleep mode.
	5 V	54, 60, 66, 72	O	5 V from the RMC connector host system. The rail is always on when the main host system is not in sleep mode.
	FPGA_VIO<47..0>	234	O	I/O voltage for DIO<47..0>. 3.3 V or 2.5 V configurable via the CLIP wizard.
	FPGA_VIO<96..48>	240	O	I/O voltage for DIO<96..48>. 3.3 V or 2.5 V configurable via the CLIP wizard.

Table 6. RMC Connector Pins with Dedicated Functions (Continued)

Pin Group	Pin Name	Pin Number	Direction (from Host System)	Description
Power (Input)	VIN_FILTERED	1, 7, 14, 20	I	9 V to 30 V input to power the device through the RMC connector rather than through the front panel connector.
	VBAT	236	I	This pin may be used to connect a longer life battery to the RTC on the host system. The RTC will track absolute time as long as either the main system battery or RMC battery through VBAT pin contains sufficient charge.

Table 6. RMC Connector Pins with Dedicated Functions (Continued)

Pin Group	Pin Name	Pin Number	Direction (from Host System)	Description
C Series DIO	ID_SELECT#[1], OSCLK_DIO0[1], TRIG_DIO1[1], DONE#_DIO2[1], CVRT#_DIO3[1], SPIFUNC_DIO4[1], SPICS#_DIO5[1], MISO_DIO6[1], MOSI_DIO7[1], SPI_CLK[1] ID_SELECT#[2], OSCLK_DIO0[2], TRIG_DIO1[2], DONE#_DIO2[2], CVRT#_DIO3[2], SPIFUNC_DIO4[2], SPICS#_DIO5[2], MISO_DIO6[2], MOSI_DIO7[2], SPI_CLK[2]	40, 46, 50, 53, 55, 56, 58, 61, 63, 64, 68, 69, 71, 73, 74, 76, 77, 79, 81, 82	I/O	Signal conditioned C Series DIO.
	SLEEP 5V C Series	45, 51 86, 91	O	

Table 6. RMC Connector Pins with Dedicated Functions (Continued)

Pin Group	Pin Name	Pin Number	Direction (from Host System)	Description
Resets	RST#	38	O	Reset that indicates that the main power is not ideal, or that the RMC connector host system has been reset.
	SYS_RST#	43	I	System reset used to reset the RMC connector host system. Asserting this pin causes the RST# pin to also assert. This signal is pulled to 3.3 V with a 4.75 kΩ resistor when in Run Mode, Safe Mode, and Sleep Mode.
High speed USB (after PHY)	USB_D+ USB_D-	29 35	I/O	Port for hi-speed differential USB.
	USB_CPEN	33	O	USB over-current protection enable.

Table 6. RMC Connector Pins with Dedicated Functions (Continued)

Pin Group	Pin Name	Pin Number	Direction (from Host System)	Description
PCIe	PCIe Clk+ PCIe Clk- PCIe_Gen2 TX+ PCIe_Gen2 TX-	32 31 3 9	O	Port for PCIe device.
	PCIe_Gen2 RX- PCIe_Gen2 RX+	22 16	I	
	CLK_REQ#	37	I	Short this signal to GND or to the CLK_REQ# pin of a PCIe connector when the PCIe interface is used. Leave floating otherwise.
SATA	SATA_TX+ SATA_TX-	5 11	O	Port for additional SATA storage. May be routed to an mSATA or M.2 connector.
	SATA_RX+ SATA_RX-	18 24	I	
Additional Processor I/O	PROC_LED	59	O	Indicates a controller state similar to the STATUS LED. Refer to the CompactRIO Single-BoardController with NI-DAQmx Installation Manual for the STATUS LED behavior.

General Purpose Digital I/O Pins

Table 7. RMC Connector General Purpose Digital I/O Pins

Pin Group	Pin Name	Direction (from Host System)	Description
General purpose digital I/O pins	DIO[0..95]	I/O	Pins for connecting directly to the FPGA through a series resistor and for enabling serial peripherals on an RMC.

RMC Connector Electrical Characteristics

RMCs with FPGA I/O pins that require an external pull-up or pull-down should use the values listed in the following table.

Table 8. Recommended External Pull-Up and Pull-Down Values

Requirement	Maximum Value	Minimum Value
External pull-up	14.7 kΩ	1 kΩ
External pull-down	8 kΩ	1 kΩ

RMC Connector Power Requirements

Use the following voltage pins to power the RMC:

- 5 V rail, which provides a primary power source to the RMC
- 3.3 V, which provides an auxiliary power source to the RMC
- FPGA_VIO, which provides I/O power for the FPGA I/O pins

Notice Ensure that your RMC does not source any current onto any of the power pins and can tolerate 5 V, 3.3 V, and FPGA_VIO coming up in any order.

Notice If FPGA_VIO<47..0> and FPGA_VIO<96..48> are configured as different voltages by the LabVIEW FPGA CLIP wizard, then ensure these are separate nets on the RMC design in order to prevent the higher voltage from sourcing current onto the lower voltage.

Fixed Behavior Signals

Power Rails

Note Refer to the model specifications document for your model on *ni.com/manuals* for information about voltage tolerance and current levels.

FPGA_VIO

FPGA_VIO only contains a valid voltage when the FPGA is configured. At other times the voltage rail is floating and will typically settle to 0 V. FPGA_CONF can be monitored to know when FPGA_VIO is valid. If FPGA_VIO is required to be at a known voltage at power up then download your FPGA application to the flash of the sbRIO-96xx.

Both FPGA_VIO<47..0> and FPGA_VIO<96..48>can be configured for either 2.5 V or 3.3 V. A voltage of 2.5 V allows for either 2.5 V single ended IO standards or differential LVDS IO standards. A voltage of 3.3 V allows for 3.3 V single ended IO standards. When configuring the IO through the CLIP both 2.5 V and 3.3 V is supported. When configuring the IO by selecting **New** »**RIO Mezzanine Card** the voltage is fixed at 3.3 V.

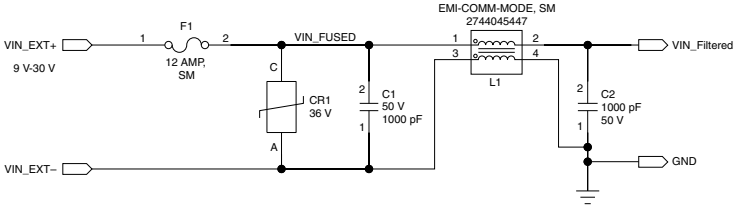
VIN_Filtered

The sbRIO-96xx may alternatively be powered over the RMC connector via the VIN_Filtered pins. These input pins are 9 V to 30 V. These power pins must contain appropriate filtering on them to ensure reliable operation of the sbRIO-96xx.

Note If simultaneously connected to multiple power sources, the sbRIO-96xx draws power from the terminal with the higher voltage. Ensure that the preferred power supply is 500 mV higher than the alternative power supply.

VIN_Filtered Implementation on the RMC

Figure 2. VIN_Filtered Reference Schematic



Connect a well-regulated voltage that falls in the range of 9 V to 30 V to the VIN_Filtered pins to power up the board.

Include a common mode choke in the design before connecting the voltage rails to the RMC connector. Place a transient voltage suppressor before the common choke.

Reference Schematic Design Considerations

The following table lists design considerations for the schematic shown in the previous figure.

Table 9. Power Rails Reference Schematic Design Considerations

Consideration	Notes
TVS Selection	The recommended part is SMDJ36CA from LittleFuse. Any TVS with reverse standoff voltage and breakdown voltage of more than 30 V can be designed in.
Common Mode Choke	The recommended part is 2744045447 from Fair-Rite. Alternatively, use a common mode choke that matches the performance of this part in terms of the DC and AC impedance.
Capacitor	1000 pf is the recommended value of the decoupling input and output capacitor. The recommended part is a ceramic COG.
Fuse	The recommended part is 0451012 from LittleFuse if the only load after the fuse is the VIN_RMC input pin to the sbRIO device. Use a 12 A fuse to provide sufficient margin and prevent false blows due to temperature and process variations. If you choose to connect other loads after the fuse, you must account for the extra current drawn by that load when selecting a fuse.

USB Support

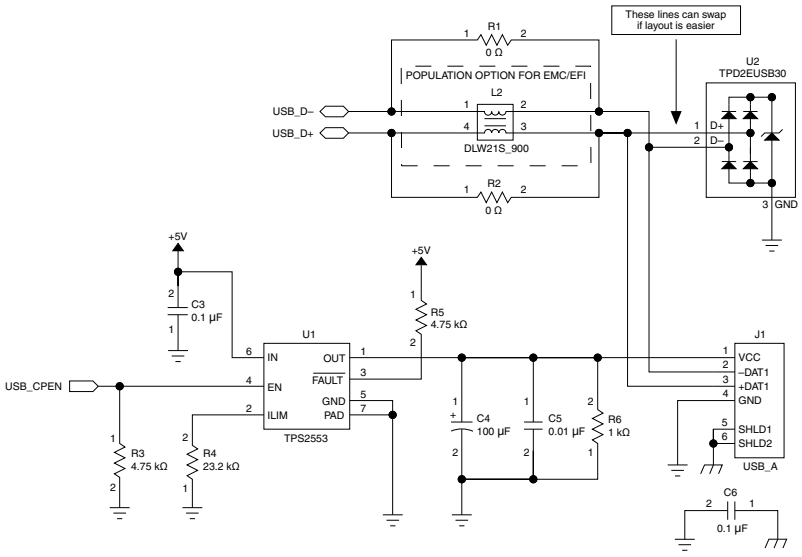
USB support over RMC has the following features.

- Normally used for mass storage or other USB peripherals.
- Supports host mode.
- USB pairs connect to either a USB connector or to a USB device on the RMC board.

Note Your RMC design must provide the 5 V USB_VBUS power to USB Host ports and must limit the current supplied to each host port according to USB specifications.

Note Refer to the *File System* section for more information about configuring USB storage devices over RMC.

Figure 3. USB Host Reference Schematic



There is a pull down resistor on USB_CPEN to guarantee the USB port is not powered until the processor is ready.

Supporting Onboard USB Devices

When you implement a USB device directly on your RMC, you can connect the device to a USB Host port from the sbRIO device. For this case, use the following design guidelines:

- You can connect the USB data pair directly to a USB device on your RMC.
- A current limiter is not required.
- Use the RST# signal to reset the USB device when the sbRIO device is in reset.

Note USB_CPEN has a pulldown to ensure that the USB port is not powered until the processor is ready.

USB Routing Considerations

NI recommends the following design practices for properly routing USB signals on your RMC:

- Route the USB_D+ and USB_D- signals as differential pairs with 90 Ω differential impedance.
- Length-match the positive and negative signal for each USB data pair to within 10 mils.
- Limit the USB_D+ and USB_D- trace lengths on the RMC to 8.0 in. or less, which is the length at which USB compliance was tested.

C Series Interface

- The C Series interface provides up to two slots of C Series support on the RMC.
- All lines can be connected directly to the 15-pin DSUB connector except for the 5 V power. The 5 V power has specific filtering requirements.
- You can program C Series modules with Real-Time (NI-DAQmx) mode, Real-Time Scan mode, and FPGA mode.

C Series Implementation on the RMC

The following figures show schematic designs for the C Series implementation on the RMC.

Figure 4. C Series with Current Limiter Reference Schematic

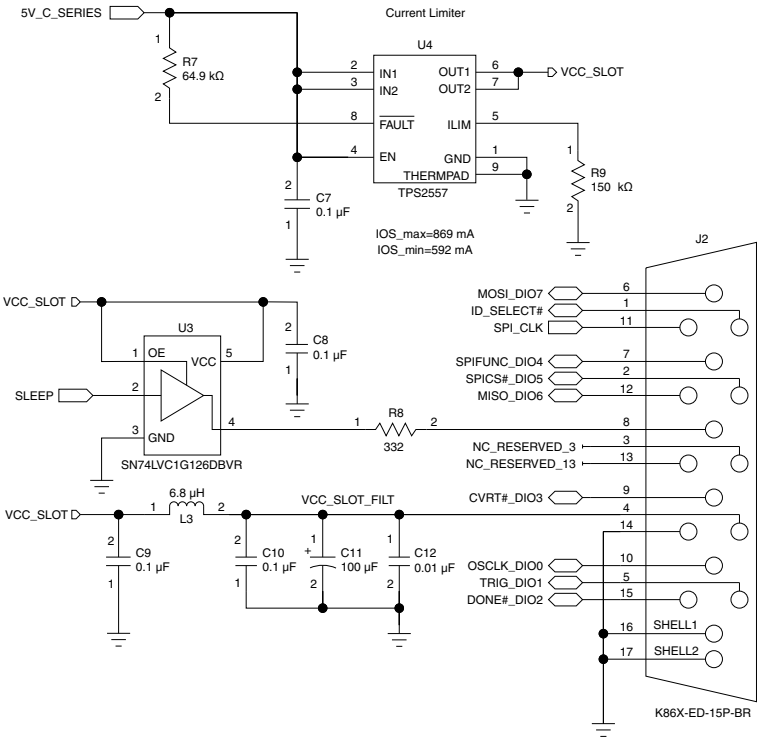
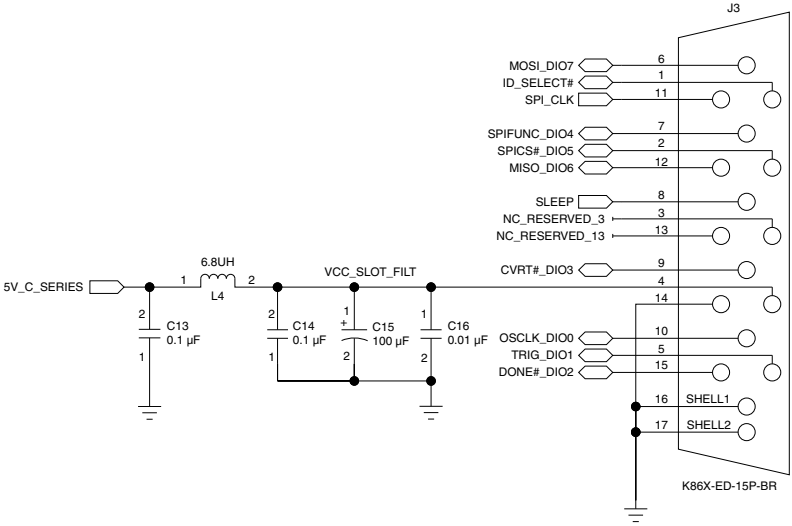


Figure 5. C Series without Current Limiter Reference Schematic



Use a current limiter to protect the PI inductor from overcurrenting in a fault condition and prevent the 5 V pin from accidentally shorting to either GND or CHSY. This protection is beneficial in environments where hot-plugging C Series modules are used.

Reference Schematic Design Considerations

The following table lists design considerations for the schematics shown in the previous figures.

Table 10. C Series Reference Schematic Design Considerations

Consideration	Notes
Current limiter U4	If a current limiter is used, you must re-buffer the sleep signal to the DSUB connector. If a current limiter is not used, you can connect the sleep signal directly to the DSUB connector by removing U15 in the schematic. The buffer prevents the sleep signal from being driven to the C Series module in an overcurrent condition.
Inductor L3 or L4	Power PI Filter specifications: • Value: 6.8 µH ±20% • ESR: <200 mΩ • Rated Current: >=400 mA
Capacitor C11 or C15	Power PI Filter specifications: • Value: 100 µF ±20% • ESR: <100 mΩ

C Series Routing Considerations

NI recommends the following design practices for properly routing C Series signals on your RMC[2]:

- Route the signals with 55 Ω ±10% impedance.
- Length-match each signal to within 250 mils.
- Limit each signal trace length on the RMC to 10.0 in. or less, which is the length from the RMC SEARAY connector to DSUB connector.
- Maintain at minimum a 3 × H line spacing between single-ended traces, where H is the distance in the board stack-up from the trace to its reference plane.

RTC Battery (VBAT)

The NI sbRIO-96xx contains a lithium cell battery that maintains the real-time clock (RTC) on the sbRIO device when the sbRIO device is powered off. A slight drain on the battery occurs when power is not applied to the sbRIO device.

The `CMOS BATTERY IS DEAD` warning appears onscreen during the power-on self test if the battery is dead. The sbRIO-96xx will start, but the system clock will reset to the date and time of the BIOS release.

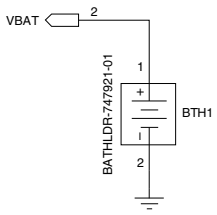
The RMC connector provides a VBAT pin to power the RTC. This allows the customer to select a battery that better aligns with their application if necessary.

[2] SLEEP lines and 5V_C SERIES are exempted from these requirements.

VBAT Implementation on the RMC

The following figure shows a schematic design for the VBAT implementation on the RMC.

Figure 6. VBAT Reference Schematic



Reference Schematic Design Considerations

You can directly connect the battery to VBAT. The sbRIO device already provides a current-limiting resistor and reverse-voltage protection.

Resets

The sbRIO device provides signals for implementing a reset button on an RMC and indicating that the sbRIO device is in reset.

RMC RST#

The RST# pin indicates that 3.3 V and 5 V power provided through the RMC Connector is valid or that the sbRIO-96xx is in reset. The signal goes to 3.3 V if the power is valid when the board powers up or when coming out of reset. The signal asserts to 0 V for at least 1 ms before returning to 3.3 V when going into reset.

SYS_RST#

The SYS_RST# signal is a system reset signal for resetting the sbRIO-96xx processor and FPGA. Asserting this signal causes the RMC RST# signal to also assert. The SYS_RST# signal asserts low. SYS_RST# is pulled to 3.3 V with a 4.75 kΩ resistor when in Run Mode, Safe Mode, and Sleep Mode.

The amount of time for which you assert this signal determines the specific reset behavior.

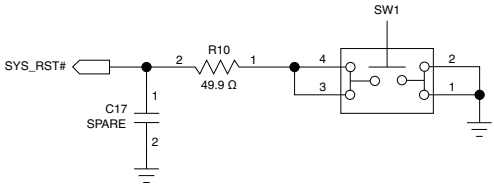
You can assert the SYS_RST# signal before you apply power to the sbRIO-96xx. The sbRIO-96xx remains in reset until the SYS_RST# signal de-asserts. If you assert the SYS_RST# signal before power is applied, then you must de-assert the SYS_RST# signal within five seconds.

Sleep Mode is an advanced feature. You can use the Linux "`shutdown -h now`" command to send the system into Sleep Mode. You can wake up the system by asserting SYS_RST# low.

Reset Implementation on the RMC

The following figure shows a schematic design for the Reset implementation on the RMC.

Figure 7. Reset Reference Schematic



Note No pull up resistor is needed when using a push button as the pull up resistor is included on the sbRIO device.

Reference Schematic Design Considerations

The following table lists design considerations for the schematic shown in the previous figure.

Table 11. Reset Reference Schematic Design Considerations

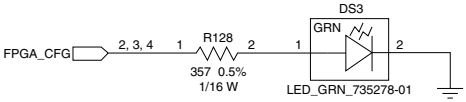
Consideration	Notes
Series termination	When SYS_RST# is driven, you must place a series termination resistor at the driver. When the driver is a mechanical switch, placing series termination is especially important due to the low output impedance of the switch.

FPGA_CONF

The sbRIO device provides an FPGA_CONF signal to indicate when the FPGA is configured, and FPGA_VIO rails have settled to either 2.5 V or 3.3 V as configured by LabVIEW in the CLIP generator. The FPGA_CONF pin drives high when the FPGA has been programmed and drives low when the FGPA has not been programmed.

FPGA_CONF Implementation on the RMC

Figure 8. FPGA_CONF Reference Schematic



User-Defined FPGA Signals

Example applications include, but are not limited to, singled-ended or LVDS differential DAC and ADC interfaces, SPI or I2C connections to sensors, buttons, and relays. In addition to FPGA Digital I/O (DIO), you can use these pins to implement RS-232 and RS-485 devices:

DIO[47..0] and DIO[96..48] can be configured for either 2.5 V or 3.3 V. A voltage of 2.5 V allows for either 2.5 V single-ended I/O standards or differential LVDS I/O standards. A

voltage of 3.3 V allows for 3.3 V single-ended I/O standards. When configuring the I/O through the CLIP both 2.5 V and 3.3 V is supported. When configuring the I/O by selecting **New** »**RIO Mezzanine Card** the voltage is fixed at 3.3 V.

Accessing User-Defined FPGA

Use one of the following methods to access the user-defined FPGA signals in LabVIEW:

- Right-click your **FPGA Target** and select **New** »**RIO Mezzanine Card...** to choose a generic Digital RMC and access all 96 DIO lines with digital I/O nodes.

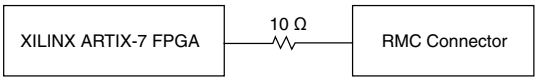
  Note This methodology does not allow you to configure the DIO lines as RS-232 and RS-485 processor peripherals.

- 1. Right click your **FPGA Target** and select **Launch sbRIO CLIP Generator...** to launch the NI Single-Board RIO CLIP Generator application, which you can use to create a socketed component-level IP (CLIP) that defines the FPGA signals or processor peripherals to use in your application.
  2. After you create a CLIP, return to LabVIEW and right-click an **RMC Socket** under the **FPGA Target** and select **Properties**.
  3. In the Socket Properties dialog box, select your CLIP and click OK. The I/O appears under the socket, or the I/O is connected directly to the RT processor.

Note For a given FPGA target, you must use either the digital I/O method or the socketed CLIP method for all 96 DIO lines.

FPGA DIO

Figure 9. Circuitry of One 3.3 V DIO Channel on the RMC Connector



The RMC has 96 2.5 V/3.3 V single-ended I/O or 45 differential pairs. The NI sbRIO-96xx is tested with all DIO channels driving ±3 mA DC loads. FPGA DIO startup states are dependent on the FPGA_VIO power rails, which do not provide valid voltages until after the FPGA is configured. This allows the user to correctly configure the FPGA_VIO voltage levels and prevent accidental damage to connected circuitry.

To ensure startup values, prior to FPGA configuration, place pull-up or pull-down resistors on the RMC DIO channel. When placing pull-up resistors, NI recommends pulling up to FPGA_VIO. Receiving circuitry that interfaces with the RMC DIO should also be powered from FPGA_VIO to ensure stable startup states. The DIO channels on the NI sbRIO-96xx are routed with a 55 Ω characteristic trace impedance. Route all RMCs with a similar impedance to ensure the best signal quality.

Signals ending in a `P` or `N` route loosely differential. If you implement them as differential signals, route with 100 Ω differential impedance on your board. If you implement them as single-ended signals, break them into two seperate traces with 55 Ω impedance each.

FPGA DIO Clock Capabilities

The light blue *FPGA Clock Optimized* designations in the RMC connector pinout indicate either SRCC or MRCC FPGA pins.

- Single-region clock capable (SRCC)—These pins provide a direct connection to the global clock distribution buffers in the FPGA. The pins also connect to the regional buffers on a specific bank of pins.
- Multi-region clock capable (MRCC)—These pins provide a direct connection to the global clock distribution buffers in the FPGA. The pins also connect to the regional and multi-regional buffers on a specific bank of pins.

Tip FPGA DIO pins through the RMC may be used to import or export clocks. Use the CLIP generation wizard to configure DIO lines for this capability. NI recommends that you use FPGA Clock Optimized pins when you import a clock into LabVIEW FPGA.

Additional UART Support

You must connect each of these interfaces to an appropriate RS-232 or RS-485 serial transceiver on your RMC design.

RS-232 and RS-485 processor peripherals created using the CLIP generator are routed from the processor to the FPGA and then to RMC FPGA DIO pins. The FPGA must be configured before the driver is loaded. This can be guaranteed by downloading your FPGA application to the flash of the sbRIO-96xx. You must connect each of these interfaces to an appropriate RS-232 or RS-485 serial transceiver on your RMC design.

Additional RS-232

You can use any FPGA pins, either 3.3 V or 2.5 V single-ended, to implement additional RS-232 ports.

All sbRIO-96xx can implement four additional RS-232 ports:

- sbRIO-960x—ASRL2, ASRL3, ASRL4, ASRL5
- sbRIO-962x/sbRIO-963x—ASRL4, ASRL5, ASRL6, ASRL7

RS-232 Reference Schematic

The following figure shows a schematic design for the RS-232 implementation on the RMC.

Figure 10. RS-232 Reference Schematic

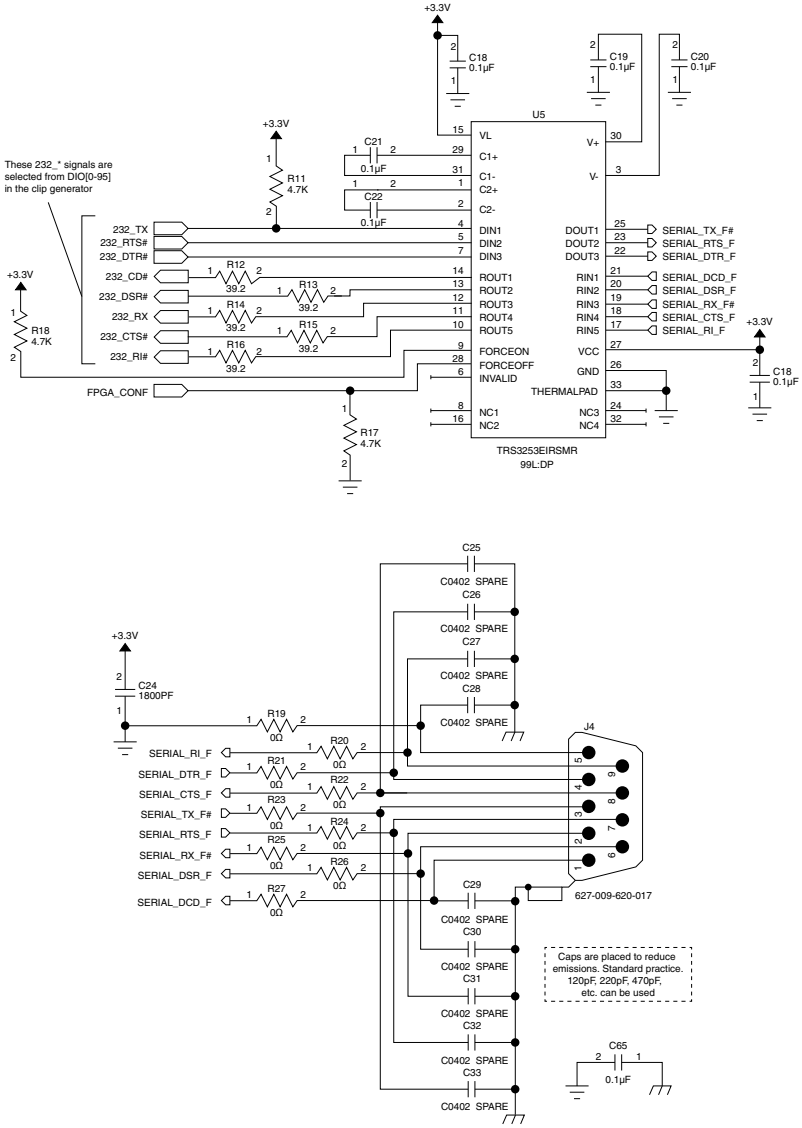


Table 12. RS-232 Reference Schematic Design Considerations

Consideration	Notes
Interface	The RMC reference schematic demonstrates how to use the FPGA DIO pins to implement a null-modem RS-232 serial port.
Serial transceiver	U5 is the RS-232 serial transceiver that converts between between RS-232 and the FPGA signal levels.
Series termination	• R12, R13, R14, R15, and R16 are the series termination for the transceiver. Use series termination at the serial transceiver on all signals being driven to the sbRIO device. • All FPGA DIO signals on the sbRIO device include series termination.
FPGA	All serial port signals pass through the FPGA on the sbRIO device. The FPGA_CONF signal is used to disable the serial transceiver when the FPGA is not configured. Disabling the transceiver in this way prevents any unwanted glitches on the RS-232 port.

Additional RS-485

You can use any FPGA pins, either 3.3 V or 2.5 V single-ended, to implement additional RS-485 ports.

All sbRIO-96xx models can implement two additional RS-485 ports:

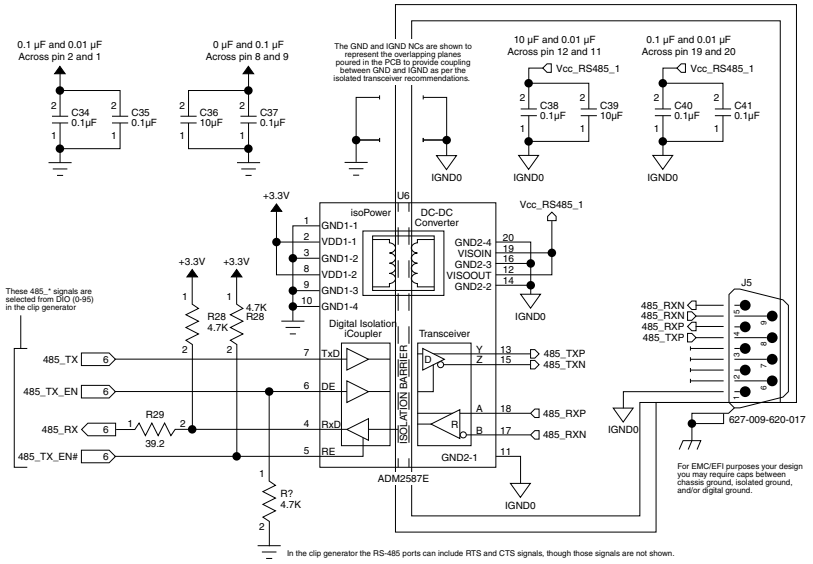
- sbRIO-960x—ASRL6, ASRL7
- sbRIO-962x/sbRIO-963x—ASRL8, ASRL9

Note RS-485 interfaces support RTS and CTS signals in the CLIP generator.

RS-485 Reference Schematic

The following figure shows a schematic design for the RS-485 implementation on the RMC.

Figure 11. RS-485 Reference Schematic



Reference Schematic Design Considerations

Table 13. RS-485 Reference Schematic Design Considerations

Consideration	Notes
Interface	The RMC demonstrates how to use the FPGA DIO pins to implement a null-modem RS-485 serial port.
Serial transceiver	U6 is the RS-485 serial transceiver that converts between RS-485 and the FPGA signal levels.
Series termination	• R29 is the series termination for the transceiver. Use series termination at the serial transceiver on all signals being driven to the sbRIO device. • All FPGA DIO signals on the sbRIO device include series termination.

RS-485 Layout Considerations

Pay close attention to how the ground planes are arranged under the isolated RS-485 transceiver. Isolated and non-isolated ground planes overlap across layers to provide some capacitance between the grounds and help with EMC. Refer to the datasheet for the RS-485 transceiver for more information.

PCIe over RMC

PCIe is routed to the RMC connector to allow for connection to PCIe devices. These devices may require a Linux driver in order to operate properly, which makes this an advanced feature not supported in LabVIEW. Example devices include M.2 SSD, additional Ethernet ports, or WIFI.

Figure 12. PCIe Reference Schematic

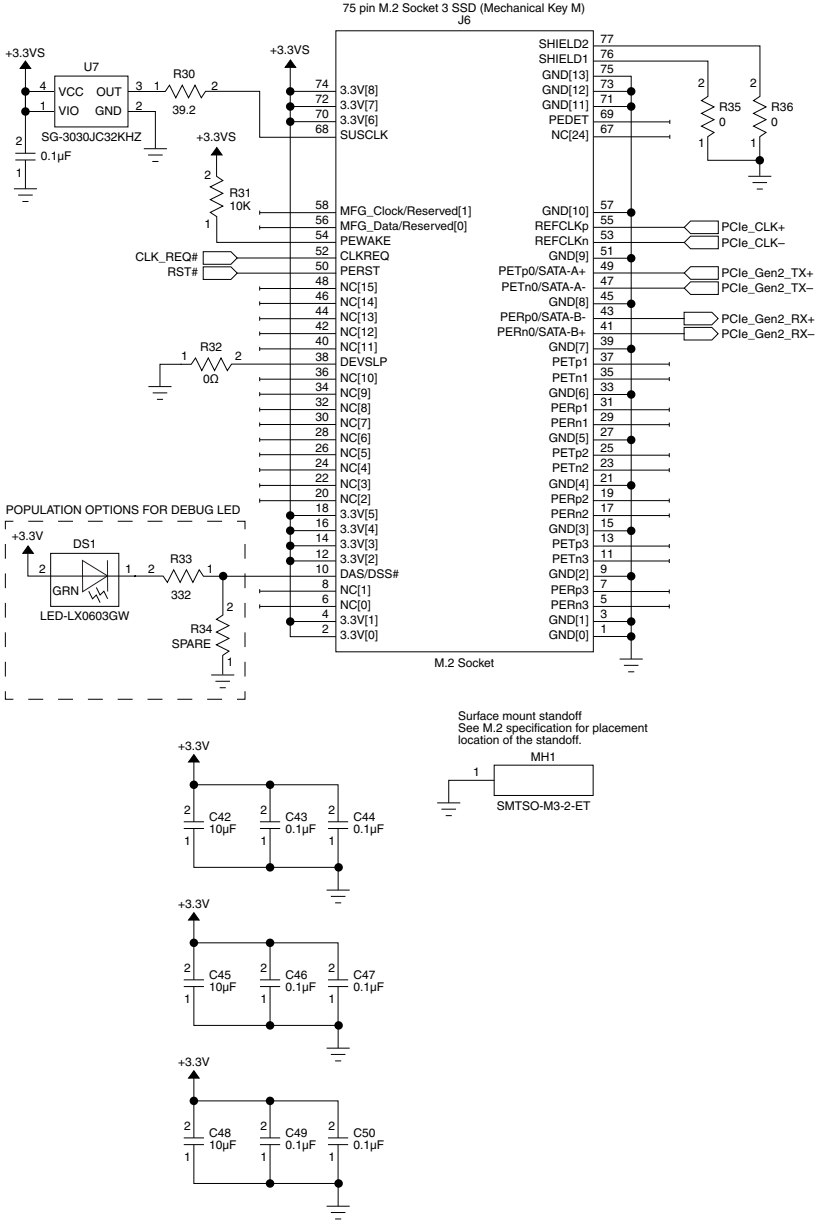


Table 14. PCIe Reference Schematic Design Considerations

Consideration	Notes
Differential signals	85 Ω ±10%
PCIe Gen1	Keep signal length shorter than 203.2 mm (8 in.)
PCIe Gen2	Keep signal length shorter than 152.4 mm (6 in.)

SATA over RMC

SATA is routed to the RMC connector to allow additional SSD storage integrated into the RMC board. These signals can be routed to an onboard SSD, M.2 connector, or mSATA connector. mSATA performs more consistently than other connecters in environments with shock and vibration conditions.

Figure 13. SATA Reference Schematic

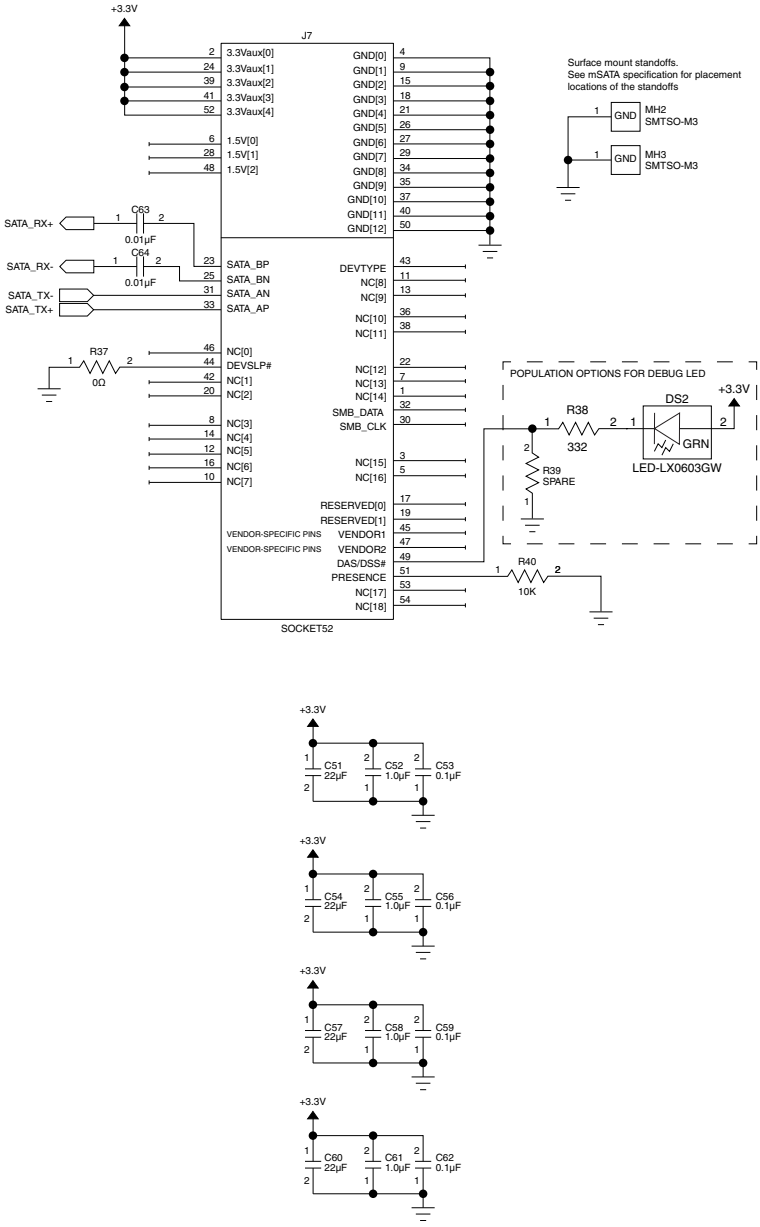


Table 15. SATA Reference Schematic Design Considerations

Consideration	Notes
Differential signals	85 Ω ±10%
Device on same PCB	Keep signal length shorter than 152.4 mm (6 in.)
Device routed to connector with 1.0 m (3.28 ft.) cable	Keep signal length shorter than 76.2 mm (3 in.)

RMC PCB Layout Guidelines

Use the guidelines in this section to help you arrange the I/O signals you implement in your RMC.

Impedance-Controlled Signaling

Use the following guidelines for implementing impedance for all I/O signals:

- All signals connected to the sbRIO device must use impedance-controlled traces.
- Trace geometry to meet impedance requirements vary depending on your specific RMC PCB stack-up. Collaborate with your vendor to match impedance requirements, stack-up, and trace geometry appropriate for your application.
- To properly maintain trace impedance and avoid discontinuities, you cannot route traces over gaps in the reference plane. Use stitching vias and capacitors when appropriate near layer changes to provide a transient return path between reference planes.

Single-Ended Signal Best Practices

Use the following guidelines for implementing single-ended I/O signals:

- Route all single-ended signals that are implemented on your RMC and connected to the sbRIO device with 55 Ω characteristic trace impedance.
- Maintain the following line spacing between single-ended traces, where H is the distance in the board stack-up from the trace to its reference plane:
  - 3 × H for C Series signals
  - 2 × H for all other signals
- Series termination resistors for FPGA DIO signals are included on the sbRIO device. Refer to the *FPGA DIO* section for more information.
- To route single-ended signals to an external connector, NI recommends adding additional series resistance at the RMC connector to account for higher impedance and longer route of external cabling. For reducing overshoot and undershoot, 33 Ω series resistance is typically effective.

Differential Signal Best Practices

NI recommends routing differential pair signals implemented on your RMC and connected to the sbRIO model with the following differential trace impedances.

Table 16. Differential Trace Impedance

Differential Pair Signal	Trace Impendance
FPGA	100 Ω ±10%
USB	90 Ω ±10%
PCIe	85 Ω ±10%
SATA	85 Ω ±10%

Note Maintain at minimum a 3 × H spacing between differential pairs and any other copper features on the same layer, where H is the distance in the board stack-up from the trace to its reference plane.

Ground Plane Recommendations

You must include ground planes on your RMC. All GND pins on the RMC connector of the sbRIO device must connect to the RMC ground planes.

Fanout and Layout Options

Refer to Samtec SEARAY documentation for information about possible fanout and layout options with various layer count RMCs.

Mechanical Considerations

Power dissipated on the RMC will affect and be affected by the power dissipated on the sbRIO device. You must provide serious consideration to the thermal performance of both the RMC and sbRIO device to ensure that your applications meets component specifications. Refer to the *CompactRIO Single-Board Controller with NI-DAQmx Hardware Installation Manual* on *ni.com/manuals* for more information about validating the thermal performance of the sbRIO device. The following recommendations may increase the thermal performance of the system:

- Spread high-power dissipating components across the surface of the printed circuit board rather than placing them in close proximity to each other.
- Place high-power dissipating components on the side of the board opposite the RMC connector.
- Minimize the amount of dissipation by the RMC in the area directly underneath the sbRIO device as this will greatly influence the sbRIO device secondary side local ambient temperature.
- Design and validate a thermal solution for the high-power dissipating components of your RMC.

When deploying in environments that could experience high levels of shock or vibration, the following recommendations may increase the robustness of the system:

- Use a printed circuit board at least 2 mm (0.79 in.) thick.
- Use positive locking connectors with thru-hole technology and the greatest practical amount of gold plating on contacts.
- Design mechanical features for strain relief and retention of connectors and cables.

Selecting an Appropriate Mating Connector

The RMC connector on the sbRIO-96xx is a Samtec SEAF-40-06.5-S-06-2-A-K-TR 240-pin, 6 x 40 position, SEARAY open-pin-field-array connector. To interface with the RMC connector, your RMC design must implement a mating connector that is compatible with the Samtec SEAF series. The following table lists compatible mating connectors.

Table 17. Connector and Compatible Mating Connectors

Connector	Manufacturer, Part Number
RMC connector	Samtec SEAF-40-06.5-S-06-2-A-K-TR
Recommended mating connector[3]	Samtec SEAM-40-03.0-S-06-2-A-K-TR

Selecting Appropriate Standoffs

The Samtec SEAM series connectors are available in multiple heights. The height of the mating connector you select helps determine the height of the standoffs you need.

To prevent over-insertion, the SEARAY connector design requires that standoffs never be less than the stack height. Because standard nominal tolerances might result in a standoff being shorter than the stack height, NI requires that you use standoffs that are 0.15 mm (0.006 in.) taller than the combined height of the J1 connector on the NI sbRIO device and the mating SEARAY connector. Therefore, to determine the required standoff height, you must add the heights of the mated connectors plus an additional 0.15 mm (0.006 in.). Refer to Samtec documentation for more information about SEARAY standoff requirements.

The following table provides an example standoff height calculation using a Samtec SEAM-40-03.0-S-06-2-A-K-TR mating connector.

Table 18. Example Connector Configuration and Calculated Standoff Height

Component	Manufacturer, Part Number	Height
J1 connector	Samtec SEAF-40-06.5-S-06-2-A-K-TR	6.50 mm (0.256 in.)
Mating connector	Samtec SEAM-40-03.0-S-06-2-A-K-TR	3.00 mm (0.118 in.)

[3] Compatible connectors are available in multiple stack height and termination options.

Table 18. Example Connector Configuration and Calculated Standoff Height (Continued)

Component	Manufacturer, Part Number	Height
Required additional standoff height	—	0.15 mm (0.006 in.)
Total calculated standoff height	—	9.65 mm (0.380 in.)

NI Custom Standoffs

NI offers a custom standoff that is an exact fit with the recommended or other compatible 9.5 mm (0.374 in.) stack height mating connectors. This custom M3 × 9.65 mm (0.380 in.) standoff is made from 4.5 mm (0.177 in.) stainless steel hex stock and includes a nylon threadlock patch. The external threads extend 4.78 mm (0.188 in.) and the internal threads are 5 mm (0.197 in.) deep. The standoff is available from NI in quantities of 12 by ordering part number 153166-12.

NI recommends that you use stainless steel fasteners for good corrosion resistance and strength. Tighten M3 fasteners to a torque of 0.76 N · m (6.70 lb · in), unless otherwise noted or required by your specific design constraints.

Discovering the Controller in MAX

1. Launch MAX on the host computer.
2. Expand **Remote Systems** in the configuration tree and locate your system.
3. Select your target.

Tip MAX lists the system under the model number followed by the serial number, such as NI-sbRIO-9628-01CEEDD8 by default.

Setting a System Password

1. In MAX, click the **Log In** button on the toolbar.
2. Enter `admin` in the **User name** field.
3. Leave the **Password** field blank.

Note There is no default password for the sbRIO-96xx, so you must leave the password field blank when logging in until you set a system password.

4. Click the **OK** button.
5. Click the **Set Permissions** button in the toolbar.

   The NI Web-Based Configuration and Monitoring utility opens in your default browser and is where you set the password. If you have not installed Microsoft Silverlight, NI Web-based Configuration & Monitoring prompts you to do so.
6. Click the **Login** button and enter `admin` in the **User name** field.
7. Leave the **Password** field blank if you have not changed the default password, or enter the current password.

8. Double-click **admin** in the list of users under the **Users** tab.
9. Click **Change Password**.
10. Enter and re-enter a new password.
11. Click **OK**.
12. Click **Save**.
13. Click **OK** to confirm you are changing the password.

Notice NI cannot recover lost system passwords. If you forget the password, you must contact NI and reformat the controller.

14. Close the NI Web-Based Configuration and Monitoring utility.

Installing Software on the Controller

1. In MAX, expand your system under Remote Systems.
2. Right-click **Software**.
3. Select **Add/Remove Software** to launch the LabVIEW Real-Time Software Wizard.

Tip You must log in to install software on the sbRIO-96xx. The default username for the sbRIO-96xx is `admin`. There is no default password for the sbRIO-96xx. To set a password for your system, refer to *Setting a System Password*.

4. Select the recommended software set for your LabVIEW and NI-RIO Device Drivers versions.
5. Click **Next**.
6. Select any additional software from the list of software add-ons, if needed.

Tip You can use this wizard at any time to install additional software.

Note Download the NI Device Drivers appropriate for the programming mode deployed in your system design:

- LabVIEW FPGA Module is required to run your modules in the LabVIEW FPGA programming mode.
- NI Scan Engine is required to run your modules in the Real-Time Scan (IO Variables) programming mode.
- NI-DAQmx is required to run your modules in Real-Time (NI-DAQmx) programming mode.

7. Click **Next**.
8. Verify that the summary of software to install is correct.
9. Click **Next** to start the installation.
10. Click **Finish** when the installation is complete.

Testing Your Controller in MAX

Complete the following steps to run a test panel in Measurement & Automation Explorer (MAX) to confirm that your sbRIO-96xx is communicating with your system.

Note MAX test panels are only available for systems running NI-DAQmx driver software with supported sbRIO modules installed and deployed in the Real-Time (NI-DAQmx) programming mode.

1. Launch MAX.
2. Locate and select your sbRIO-96xx system in the **My System** tree.
   - If your controller and the software are functioning correctly, the System Settings for your module will show a status of **`Connected - Running`**.
   - If your sbRIO-96xx is present but the status is **`Connected – Safe Mode (No Software Installed)`**, complete the procedure to install software on the sbRIO-96xx in *Installing Software on the Controller*.
3. Connect the sbRIO controller as indicated in the *Hardware Installation Manual*.
4. Right-click any sbRIO I/O module and select **Test Panels**, or select **Test Panels** from the main configuration window options.

   Note If the sbRIO I/O module you select is not deployed in Real-Time (NI-DAQmx) mode, you will be prompted to switch modes to view the test panel.
5. Configure the measurement settings and click **Start**.

Changing sbRIO I/O Module Programming Modes in MAX

1. Launch MAX on the host computer.
2. Expand **Remote Systems** in the configuration tree and locate your sbRIO-96xx system.
3. Select the sbRIO I/O module.
4. In the **Settings** pane, expand the **Program Mode** pull-down menu.
5. Select the programming mode appropriate for your application.
6. Click **Save**.

Configuring Startup Options

Complete the following steps to configure the sbRIO-96xx startup options in MAX.

1. In MAX, expand your system under Remote Systems.
2. Select the **Startup Settings** tab to configure the startup settings.

sbRIO-96xx Startup Options

You can configure the following sbRIO-96xx startup options.

Table 19. sbRIO-96xx Startup Options

Startup Option	Description
Force Safe Mode	Rebooting the sbRIO-96xx with this setting on starts the sbRIO-96xx without launching LabVIEW Real-Time or any startup applications. In safe mode, the sbRIO-96xx launches only the services necessary for updating configuration and installing software.
Enable Console Out	Rebooting the sbRIO-96xx with this setting on redirects the console output to the RS-232 serial port. You can use a serial-port terminal program to read the IP address and firmware version of the sbRIO-96xx. Use a null-modem cable to connect the RS-232 serial port to a computer. Make sure that the serial-port terminal program is configured to the following settings: • 115,200 bits per second • Eight data bits • No parity • One stop bit • No flow control
Disable RT Startup App	Rebooting the sbRIO-96xx with this setting on prevents any LabVIEW startup applications from running.
Disable FPGA Startup App	Rebooting the sbRIO-96xx with this setting on prevents autoloading of any FPGA application.
Enable Secure Shell (SSH) Logins	Rebooting the sbRIO-96xx with this setting on starts sshd on the sbRIO-96xx. Starting sshd enables logins over SSH, an encrypted communication protocol. **Note** Visit *ni.com/r/openssh* for more information about SSH.
LabVIEW Project Access	Rebooting the sbRIO-96xx with this setting on enables you to add the target to a LabVIEW project.

Configuring FPGA Startup App

Use the RIO Device Setup utility, which you can launch in the following ways, to select an FPGA startup application:

- (Windows 8) Click the **NI Launcher** tile on the Start screen and select **RIO Device Setup**.
- (Windows 7 or earlier) Select **Start»All Programs»National Instruments»RIO Device Setup**.

Using the sbRIO-96xx in LabVIEW

Adding the sbRIO-96xx to a LabVIEW Project

You can use LabVIEW to program in Real-Time (NI-DAQmx) or Real-Time Scan (IO Variables) modes. For applications using advanced functionality that require programming the FPGA, LabVIEW FPGA Module is required to deploy sbRIO I/O modules in the LabVIEW FPGA programming mode.

1. To create a new LabVIEW project, launch LabVIEW and click **Create Project**.
2. Select **Blank Project** from the **Create Project** dialog and click **Finish**.
3. To add your system to the project, right-click the top of the project tree and select **New»Targets and Devices** to launch the **Add Targets and Devices** discovery dialog.
4. In the **Add Targets and Devices** dialog, expand the **Real-Time CompactRIO** folder, select your system, and click **OK**.

   Note If your system is not listed, LabVIEW could not detect it on the network. Ensure that your system is properly configured with a valid IP address in Measurement & Automation Explorer. If your system is on a remote subnet, you can also select to manually enter the IP address. You can locate the system IP address in MAX in the System Settings for the sbRIO-96xx.
5. In the LabVIEW Project Explorer, verify that your system is present in the project tree. Click **File»Save** to save the project.

   The LabVIEW project is created. All sbRIO I/O modules appear under Real-Time Resources by default, which indicates that all modules are deployed in Real-Time (NI-DAQmx) mode.

Deploying Your sbRIO I/O Module in a Programming Mode in LabVIEW

1. Locate your sbRIO I/O module in your LabVIEW project. If you just created the project, all sbRIO I/O modules appear under Real-Time Resources, which indicates that they are in the Real-Time (NI-DAQmx) programming mode.
2. Drag your sbRIO I/O module to **Real-Time Scan Resources**. This indicates you plan to use your module in Real-Time Scan (IO Variables) programming mode.
3. Right-click the sbRIO-96xx in the project and select **Deploy All** to deploy the module in the Real-Time Scan (IO Variables) programming mode.

   Your module is now in Real-Time Scan mode.
4. Drag your sbRIO I/O module back to **Real-Time Resources**. This indicates you plan to use your module in Real-Time (NI-DAQmx) programming mode.
5. Right-click the sbRIO-96xx in the project and select **Deploy All** to deploy the module in the Real-Time (NI-DAQmx) programming mode.

   Your module is now in Real-Time (NI-DAQmx) mode.

Programming Examples

For information on getting started with DAQmx programming mode examples in LabVIEW, go to *ni.com/r/criodaqmx*. You can also access example projects in LabVIEW by selecting **Help»Find Examples**, and navigating to **Hardware Input and Output»sbRIO**.

Adding the sbRIO-96xx to a LabVIEW FPGA Project

LabVIEW FPGA Module is required to program the user-accessible FPGA on the sbRIO-96xx or deploy sbRIO I/O modules in the LabVIEW FPGA program mode. To program in Real-Time (NI-DAQmx) or Real-Time Scan (IO Variables) modes, the LabVIEW FPGA module is not required.

1. To create a new LabVIEW project, launch LabVIEW and click **Create Project**.
2. Select **Templates»LabVIEW FPGA Project**.
3. Select the **CompactRIO Embedded System** project type and click **Next**.

   The **System Setup** window opens, which will allow you to add your system to the project.
4. Under **Discover existing system**, check the box next to **Device is connected to a remote subnet IP address**.
5. Enter the IP address for the sbRIO-96xx and click **Next**.

   Note You can locate the system IP address in MAX in the System Settings for the sbRIO-96xx.
6. In the LabVIEW Project Explorer, verify that your system is present in the project tree and click **Finish**.

   The LabVIEW project is created. All sbRIO I/O modules and corresponding I/O appear under the FPGA Target.

   Note By default, all sbRIO I/O modules appear under the FPGA Target, which indicates that all modules are selected for use in LabVIEW FPGA mode.

Deploying Your sbRIO I/O Module in a Programming Mode in LabVIEW FPGA

Complete the following steps to deploy sbRIO I/O modules in a new programming mode in the LabVIEW project.

1. Locate your sbRIO I/O module in your LabVIEW project. If you just created the project, all sbRIO I/O modules appear under the FPGA Target, which indicates that they are in the LabVIEW FPGA programming mode.
2. Drag your sbRIO I/O module to **Real-Time Scan Resources**. This indicates you plan to use your module in Real-Time Scan (IO Variables) programming mode.
3. Right-click the sbRIO-96xx in the project and select **Deploy All** to deploy the module in the Real-Time Scan (IO Variables) programming mode.

   Your module is now in Real-Time Scan mode.

4. Drag your sbRIO I/O module to **Real-Time Resources**. This indicates you plan to use your module in Real-Time (NI-DAQmx) programming mode.
5. Right-click the sbRIO-96xx in the project and select **Deploy All** to deploy the module in the Real-Time (NI-DAQmx) programming mode.

   Your module is now in Real-Time (NI-DAQmx) mode.

Programming Examples

For information on getting started with DAQmx programming mode examples in LabVIEW FPGA, go to *ni.com/r/daq2comboExample*.

Choosing Your Programming Mode

The sbRIO-96xx supports three programming modes on a per slot basis for both Onboard I/O modules and C Series modules.

	**Real-Time**	Enables you to use sbRIO I/O modules directly from LabVIEW Real-Time, using NI DAQmx. sbRIO I/O modules appear under the Real-Time Resources item in the MAX Project Explorer window and I/O channels appear as I/O variables under the modules. To use I/O variables, you drag and drop them from the Project Explorer window to LabVIEW Real-Time VIs. Use this mode to make the sbRIO I/O module behave like it is in a CompactDAQ controller, using the Real-Time NI-DAQmx and NI-XNET drivers to communicate, and access the four counter/timers and the PFI trigger connector on the controller.
	**Real-Time Scan**	Enables you to use C Series modules directly from LabVIEW Real-Time, using I/O variables. C Series modules that you use in Scan Interface mode appear under the Real-Time Scan Resources item in the MAX Project Explorer window and I/O channels appear as I/O variables under the modules. To use I/O variables, you drag and drop them from the Project Explorer window to LabVIEW Real-Time VIs. In this mode, you do not need to do any LabVIEW FPGA development. LabVIEW programs the FPGA for you with a fixed FPGA bitfile that communicates with all the C Series modules that RT Scan mode supports. LabVIEW also sends C Series module data to the Real-Time host to be displayed in I/O variables. Real-Time Scan mode also enables you to dynamically detect which types of C Series modules are plugged into module slots. You can not program Onboard I/O in Real-Time Scan mode.

	**FPGA**	Enables you to use sbRIO I/O modules from LabVIEW FPGA VIs. sbRIO I/O modules appear directly under the FPGA Target item in the MAX Project Explorer window and I/O channels appear as FPGA I/O items under the FPGA Target. To access the I/O channels, you either configure FPGA I/O Nodes in a LabVIEW FPGA VI or drag and drop the I/O channels from the Project Explorer window to a LabVIEW FPGA VI block diagram. Use this mode to add more flexibility, customization, timing, and synchronization to your applications. To use the sbRIO system in FPGA mode, you must either have the LabVIEW FPGA Module installed on the host computer, or have access to a compiled bitfile that you can download to the FPGA. In either case, you use the Open FPGA VI Reference function in a LabVIEW Real-Time VI to access the FPGA VI or bitfile.

Table 20. Supported Programming Modes for Popular Tasks

Task	Real-Time (NI-DAQmx)	Real-Time Scan	FPGA
Control rates up to 1 kHz	■	■	
Control rates between 1 kHz and 2.5 kHz (application dependent)	■	■	■
Control rates over 2.5 kHz			■
High-speed waveform acquisition	■		■

Note Some sbRIO I/O modules can only be used in certain programming modes. For module-specific software support information, visit *ni.com/r/swsupport*.

Software Configuration in NI-DAQmx Programming Mode

Analog Input with NI-DAQmx

You can perform analog input measurements using one of two methods:

- Install a supported single board analog input C Series module with any sbRIO-96xx controller.
- Use Connector0 to access the sbRIO I/O module on the sbRIO-962x/sbRIO-963x.

Set the programming mode to Real-Time (NI-DAQmx) mode. The measurement specifications, such as number of channels, channel configuration, sample rate, and gain, are determined by the type of sbRIO I/O module used.

The sbRIO controller has eight input timing engines, which means that up to eight hardware-timed analog input tasks can be running at a time on the controller. An analog input task can include channels from multiple analog input modules. However, channels from a single module cannot be used in multiple tasks.

Multiple timing engines allow the sbRIO controller to run up to eight analog input tasks simultaneously, each using independent timing and triggering configurations. The eight timing engines are it0, it1,…it7.

Hardware-Timed Single Point (HWTSP) Mode

In HWTSP mode, samples are acquired or generated continuously using hardware timing and no buffer. You must use the sample clock or change detection timing types. No other timing types are supported.

Use HWTSP mode if you need to know if a loop executes in a given amount of time, such as in a control application. Because there is no buffer, if you use HWTSP mode, ensure that reads or writes execute fast enough to keep up with hardware timing. If a read or write executes late, it returns a warning.

Note DSA modules do not support HWTSP mode.

Analog Input Triggering Signal

A trigger is a signal that causes an action, such as starting or stopping the acquisition of data. When you configure a trigger, you must decide how you want to produce the trigger and the action you want the trigger to cause. The sbRIO controller supports internal software triggering, external digital triggering, analog triggering, and internal time triggering.

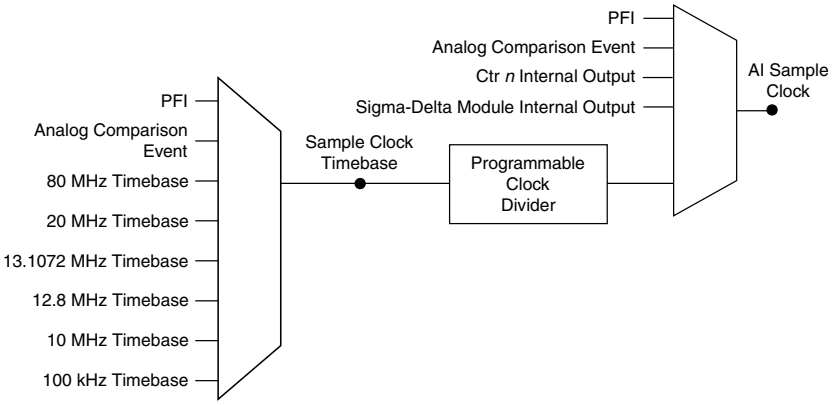
Three triggers are available: Start Trigger, Reference Trigger, and Pause Trigger. An analog or digital signal can initiate these three trigger actions. sbRIO I/O Parallel digital input modules can be used in any controller slot to supply a digital trigger.

Analog Input Timing Signals

AI Sample Clock Signal

A sample consists of one reading from each channel in the AI task. Sample Clock signals the start of a sample of all analog input channels in the task. The sample clock can be generated from external or internal sources as shown in the figure below.

Figure 14. AI Sample Clock Timing Options



Routing the Sample Clock to an Output Terminal

You can route Sample Clock to any output PFI terminal. Sample Clock is an active high pulse by default.

AI Sample Clock Timebase Signal

The AI Sample Clock Timebase signal is divided down to provide a source for Sample Clock. AI Sample Clock Timebase can be generated from external or internal sources. AI Sample Clock Timebase is not available as an output from the controller.

AI Start Trigger Signal

Use the Start Trigger signal to begin a measurement acquisition which consists of one or more samples. Once the acquisition begins, configure the acquisition to stop in one of the following ways:

- When a certain number of points has been sampled (in finite mode)
- After a hardware reference trigger (in finite mode)
- With a software command (in continuous mode)

An acquisition that uses a start trigger (but not a reference trigger) is sometimes referred to as a posttriggered acquisition. That is, samples are measured only after the trigger.

When you are using an internal sample clock, you can specify a default delay from the start trigger to the first sample.

Using a Digital Source

To use the Start Trigger signal with a digital source, specify a source and a rising or falling edge. Use the following signals as the source:

- Any PFI terminal
- Counter n Internal Output

The source also can be one of several other internal signals on your sbRIO controller. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Using an Analog Source

Some sbRIO I/O modules can generate a trigger based on an analog signal. In NI-DAQmx, this is called the Analog Comparison Event. When you use an analog trigger source for Start Trigger, the acquisition begins on the first rising edge of the Analog Comparison Event signal.

Routing AI Start Trigger to an Output Terminal

You can route the Start Trigger signal to any output PFI terminal. The output is an active high pulse.

Using a Time Source

To use the Start Trigger signal with a time source, configure a specific time in NI-DAQmx. Refer to the "Timestamps" and "Time Triggering" topics in the *NI-DAQmx Help* for more information on accessing time-based features in the NI-DAQmx API.

AI Reference Trigger Signal

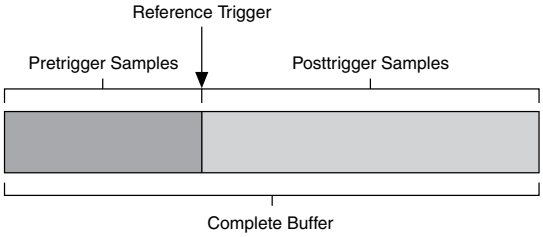
Use a reference trigger to stop a measurement acquisition. To use a reference trigger, specify a buffer of finite size and a number of pretrigger samples (samples that occur before the reference trigger). The number of posttrigger samples (samples that occur after the reference trigger) desired is the buffer size minus the number of pretrigger samples.

Once the acquisition begins, the sbRIO controller writes samples to the buffer. After the sbRIO controller captures the specified number of pretrigger samples, the sbRIO controller begins to look for the reference trigger condition. If the reference trigger condition occurs before the sbRIO controller captures the specified number of pretrigger samples, the controller ignores the condition.

If the buffer becomes full, the sbRIO controller continuously discards the oldest samples in the buffer to make space for the next sample. This data can be accessed (with some limitations) before the sbRIO controller discards it. Refer to the *Can a Pretriggered Acquisition be Continuous?* document for more information.

When the reference trigger occurs, the sbRIO controller continues to write samples to the buffer until the buffer contains the number of posttrigger samples desired. The figure below shows the final buffer.

Figure 15. Reference Trigger Final Buffer



Using a Digital Source

To use a reference trigger with a digital source, specify a source and a rising or falling edge. Either PFI or one of several internal signals on the sbRIO controller can provide the source. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Using an Analog Source

Some C Series modules can generate a trigger based on an analog signal. In NI-DAQmx, this is called the Analog Comparison Event.

When you use an analog trigger source, the acquisition stops on the first rising or falling edge of the Analog Comparison Event signal, depending on the trigger properties.

Routing the Reference Trigger Signal to an Output Terminal

You can route a reference trigger to any output PFI terminal. Reference Trigger is active high by default.

AI Pause Trigger Signal

You can use the Pause Trigger to pause and resume a measurement acquisition. The internal sample clock pauses while the external trigger signal is active and resumes when the signal is inactive. You can program the active level of the pause trigger to be high or low.

Using a Digital Source

To use the Pause Trigger, specify a source and a polarity. The source can be either from PFI or one of several other internal signals on your sbRIO controller. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Using an Analog Source

Some C Series modules can generate a trigger based on an analog signal. In NI-DAQmx, this is called the Analog Comparison Event.

When you use an analog trigger source, the internal sample clock pauses when the Analog Comparison Event signal is low and resumes when the signal goes high (or vice versa).

Note Pause triggers are only sensitive to the level of the source, not the edge.

AI Convert Clock Signal Behavior For Analog Input Modules

Scanned Modules

Scanned sbRIO I/O analog input modules contain a single A/D converter and a multiplexer to select between multiple input channels. When the module interface receives a Sample Clock pulse, it begins generating a Convert Clock for each scanned module in the current task. Each Convert Clock signals the acquisition of a single channel from that module. The Convert Clock rate depends on the module being used, the number of channels used on that module, and the system Sample Clock rate.

The driver chooses the fastest conversion rate possible based on the speed of the A/D converter for each module and adds 10 µs of padding between each channel to allow for adequate settling time. This scheme enables the channels to approximate simultaneous sampling. If the AI Sample Clock rate is too fast to allow for 10 µs of padding, NI-DAQmx selects a conversion rate that spaces the AI Convert Clock pulses evenly throughout the sample. NI-DAQmx uses the same amount of padding for all the modules in the task. To explicitly specify the conversion rate, use the **ActiveDevs** and **AI Convert Clock Rate** properties using the **DAQmx Timing** property node or functions.

Simultaneous Sample-and-Hold Modules

Simultaneous sample-and-hold (SSH) sbRIO I/O analog input modules contain multiple A/D converters or circuitry that allows all the input channels to be sampled at the same time. These modules sample their inputs on every Sample Clock pulse.

Delta-Sigma Modules

Delta-sigma sbRIO I/O analog input modules function much like SSH modules, but use A/D converters that require a high-frequency oversample clock to produce accurate, synchronized data. Some delta-sigma modules in the sbRIO controller automatically share a single oversample clock to synchronize data from all the modules that support an external oversample clock timebase when they all share the same task. (DSA modules are an example).

The oversample clock is used as the AI Sample Clock Timebase. The sbRIO controller supplies 10 MHz, 12.8 MHz, and 13.1072 MHz timebases from which software automatically selects based on the modules in the task. When delta-sigma modules with different oversample clock frequencies are used in an analog input task, the AI Sample Clock Timebase can use any of the available frequencies; by default, the fastest available is used. The sample rate of all modules in the task is an integer divisor of the frequency of the AI Sample Clock Timebase.

When one or more delta-sigma modules are in an analog input task, the delta-sigma modules also provide the signal used as the AI Sample Clock. This signal is used to cause A/D conversion for other modules in the system, just as the AI Sample Clock does when a delta-sigma module is not being used.

When delta-sigma modules are in an AI task, the controller automatically issues a synchronization pulse to each delta-sigma module so that their ADCs are reset at the same time. Because of the filtering used in delta-sigma A/D converters, these modules usually exhibit a fixed input delay relative to non-delta-sigma modules in the system. This input delay is specified in the sbRIO I/O module documentation.

When channels from delta-sigma sbRIO I/O modules are included in a multi-chassis task, please ensure that the first channel in your channel list is from a delta-sigma module.

Note DSA modules do not support HWTSP mode.

Slow Sample Rate Modules

Some sbRIO I/O analog input modules are specifically designed for measuring signals that vary slowly, such as temperature. Because of their slow rate, it is not appropriate for these modules to constrain the AI Sample Clock to operate at or slower than their maximum rate. When using such a module in the sbRIO controller mixed with a non-slow sample module in the same task, exceeding the maximum sampling rate of the slow sample module results in the most recently acquired sample being read multiple times. In this scenario, the first sample of a hardware-timed acquisition with a slow sampled sbRIO I/O module is sampled when the task is committed.

For more information about which sbRIO I/O modules are compatible with the sbRIO controller, go to *ni.com/r/rdcdaq*.

Getting Started with AI Applications in Software

You can use the sbRIO controller in the following analog input applications:

- Single-point acquisition
- Hardware-Timed Single Point acquisition
- Finite acquisition
- Continuous acquisition

For more information about programming analog input applications and triggers in software, refer to the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Analog Output with NI-DAQmx

To generate analog output, install an analog output sbRIO I/O module in any slot on the sbRIO controller. The generation specifications, such as the number of channels, channel configuration, update rate, and output range, are determined by the type of sbRIO I/O module used.

The sbRIO controller has eight output timing engines, which means that up to eight hardware-timed analog output tasks can be running at a time on the controller. On a single analog output sbRIO I/O module, you can assign any number of channels to either a hardware-timed task or a software-timed (single-point) task. However, you cannot assign some channels to a hardware-timed task and other channels (on the same module) to a software-timed task.

Multiple timing engines allow the sbRIO controller to run up to eight analog output tasks simultaneously, each using independent timing and triggering configurations. The eight timing engines are ot0, ot1,… ot7.

Analog Output Data Generation Methods

When performing an analog output operation, you either can perform software-timed generations or hardware-timed generations.

Software-Timed Generations

With a software-timed generation, software controls the rate at which data is generated. Software sends a separate command to the hardware to initiate each DAC conversion. In NI-DAQmx, software-timed generations are referred to as on-demand timing. Software-timed generations are also referred to as immediate or static operations. They are typically used for writing out a single value, such as a constant DC voltage.

The following considerations apply to software-timed generations:

- If any AO channel on a module is used in a hardware-timed (waveform) task, no channels on that module can be used in a software-timed task
- You can configure software-timed generations to simultaneously update
- Only one simultaneous update task can run at a time
- A hardware-timed AO task and a simultaneous update AO task cannot run at the same time

Hardware-Timed Generations

With a hardware-timed generation, a digital hardware signal controls the rate of the generation. This signal can be generated internally on the controller or provided externally.

Hardware-timed generations have several advantages over software-timed acquisitions:

- The time between samples can be much shorter
- The timing between samples is deterministic
- Hardware-timed acquisitions can use hardware triggering

Hardware-Timed Single Point (HWTSP) Mode

In HWTSP mode, samples are acquired or generated continuously using hardware timing and no buffer. You must use the sample clock or change detection timing types. No other timing types are supported.

Use HWTSP mode if you need to know if a loop executes in a given amount of time, such as in a control application. Because there is no buffer, if you use HWTSP mode, ensure that reads

or writes execute fast enough to keep up with hardware timing. If a read or write executes late, it returns a warning.

Note DSA modules do not support HWTSP mode.

Buffered Analog Input

A buffer is a temporary storage in computer memory for generated samples. In a buffered generation, data is moved from a host buffer to the sbRIO controller onboard FIFO before it is written to the sbRIO I/O modules.

One property of buffered I/O operations is sample mode. The sample mode can be either finite or continuous:

- *Finite*—Finite sample mode generation refers to the generation of a specific, predetermined number of data samples. After the specified number of samples is written out, the generation stops.
- *Continuous*—Continuous generation refers to the generation of an unspecified number of samples. Instead of generating a set number of data samples and stopping, a continuous generation continues until you stop the operation. There are three different continuous generation modes that control how the data is written. These modes are regeneration, onboard regeneration, and non-regeneration:
  - In regeneration mode, you define a buffer in host memory. The data from the buffer is continually downloaded to the FIFO to be written out. New data can be written to the host buffer at any time without disrupting the output. There is no limitation on the number of waveform channels supported by regeneration mode.
  - With onboard regeneration, the entire buffer is downloaded to the FIFO and regenerated from there. After the data is downloaded, new data cannot be written to the FIFO. To use onboard regeneration, the entire buffer must fit within the FIFO size. The advantage of using onboard regeneration is that it does not require communication with the main host memory once the operation is started, which prevents problems that may occur due to excessive bus traffic or operating system latency. There is a limit of 16 waveform channels for onboard regeneration.
  - With non-regeneration, old data is not repeated. New data must continually be written to the buffer. If the program does not write new data to the buffer at a fast enough rate to keep up with the generation, the buffer underflows and causes an error. There is no limitation on the number of waveform channels supported by non-regeneration.

Analog Output Triggering Signals

A trigger is a signal that causes an action, such as starting or stopping the acquisition of data. When you configure a trigger, you must decide how you want to produce the trigger and the action you want the trigger to cause. The sbRIO controller supports internal software triggering, external digital triggering, analog triggering, and internal time triggering.

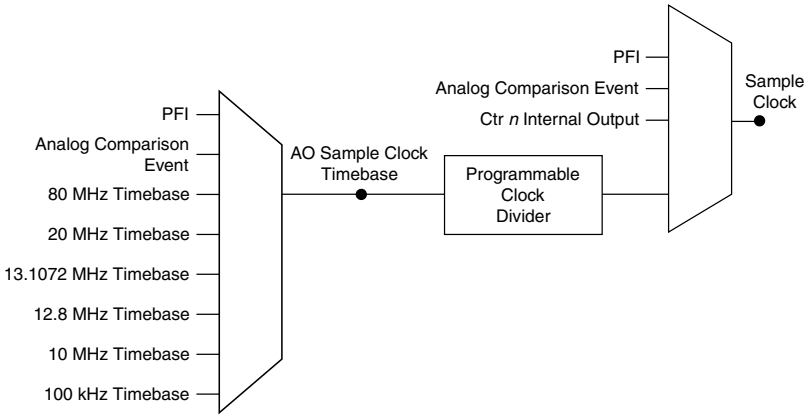
Analog output supports two different triggering actions: AO Start Trigger and AO Pause Trigger. An analog or digital signal can initiate these actions. sbRIO I/O parallel digital input modules can be used in any controller slot to supply a digital trigger. An analog trigger can be supplied by some sbRIO I/O analog modules.

Analog Output Timing Signals

AO Sample Clock Signal

The AO sample clock signals when all the analog output channels in the task update. AO Sample Clock can be generated from external or internal sources as shown in the figure below.

Figure 16. Analog Output Timing Options



Routing AO Sample Clock to an Output Terminal

You can route AO Sample Clock to any output PFI terminal. AO Sample Clock is active high by default.

AO Sample Clock Timebase Signal

The AO Sample Clock Timebase signal is divided down to provide a source for AO Sample Clock. AO Sample Clock Timebase can be generated from external or internal sources, and is not available as an output from the controller.

Delta-Sigma Modules

The oversample clock is used as the AO Sample Clock Timebase. The sbRIO controller supplies 10 MHz, 12.8 MHz, and 13.1072 MHz timebases. When delta-sigma modules with different oversample clock frequencies are used in an analog output task, the AO Sample Clock Timebase can use any of the available frequencies; by default, the fastest available is

used. The update rate of all modules in the task is an integer divisor of the frequency of the AO Sample Clock Timebase.

Note DSA modules do not support HWTSP mode.

AO Start Trigger Signal

Use the AO Start Trigger signal to initiate a waveform generation. If you do not use triggers, you can begin a generation with a software command. If you are using an internal sample clock, you can specify a delay from the start trigger to the first sample. For more information, refer to the *NI-DAQmx Help*.

Using a Digital Source

To use AO Start Trigger, specify a source and a rising or falling edge. The source can be one of the following signals:

- A pulse initiated by host software
- Any PFI terminal
- AI Reference Trigger
- AI Start Trigger

The source also can be one of several internal signals on the sbRIO controller. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

You also can specify whether the waveform generation begins on the rising edge or falling edge of AO Start Trigger.

Routing AO Start Trigger Signal to an Output Terminal

You can route AO Start Trigger to any output PFI terminal. The output is an active high pulse.

Using a Time Source

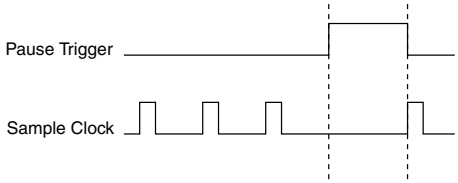
To use the Start Trigger signal with a time source, configure a specific time in NI-DAQmx. Refer to the "Timestamps" and "Time Triggering" topics in the *NI-DAQmx Help* for more information on accessing time-based features in the NI-DAQmx API.

AO Pause Trigger Signal

Use the AO Pause Trigger signal to mask off samples in a DAQ sequence. When AO Pause Trigger is active, no samples occur, but AO Pause Trigger does not stop a sample that is in progress. The pause does not take effect until the beginning of the next sample.

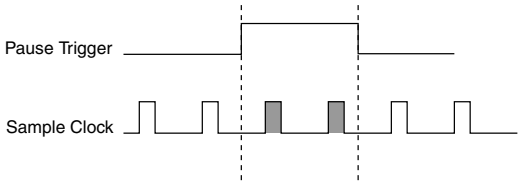
When you generate analog output signals, the generation pauses as soon as the pause trigger is asserted. If the source of the sample clock is the onboard clock, the generation resumes as soon as the pause trigger is deasserted, as shown in the following figure.

Figure 17. AO Pause Trigger with the Onboard Clock Source



If you are using any signal other than the onboard clock as the source of the sample clock, the generation resumes as soon as the pause trigger is deasserted and another edge of the sample clock is received, as shown in the following figure.

Figure 18. AO Pause Trigger with Other Signal Source



Using a Digital Source

To use AO Pause Trigger, specify a source and a polarity. The source can be a PFI signal or one of several other internal signals on the sbRIO controller.

You also can specify whether the samples are paused when AO Pause Trigger is at a logic high or low level. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Minimizing Glitches on the Output Signal

When you use a DAC to generate a waveform, you may observe glitches on the output signal. These glitches are normal; when a DAC switches from one voltage to another, it produces glitches due to released charges. The largest glitches occur when the most significant bit of the DAC code changes. You can build a lowpass deglitching filter to remove some of these glitches, depending on the frequency and nature of the output signal. Go to *ni.com/support* for more information about minimizing glitches.

Getting Started with AO Applications in Software

You can use the sbRIO controller in the following analog output applications:

- Single-point (on-demand) generation
- Hardware-Timed Single Point generation
- Finite generation

- Continuous generation
- Waveform generation

For more information about programming analog output applications and triggers in software, refer to *NI-DAQmx Help* or to the *LabVIEW Help*.

Digital Input/Output with NI-DAQmx

To use digital I/O, install a digital sbRIO I/O module into any slot on the sbRIO controller. The I/O specifications, such as number of lines, logic levels, update rate, and line direction, are determined by the type of sbRIO I/O module used.

Serial DIO versus Parallel DIO Modules

Serial digital modules have more than eight lines of digital input/output. They can be used in any controller slot and can perform the following tasks:

- Software-timed and hardware-timed digital input/output tasks

Parallel digital modules can be used in any controller slot and can perform the following tasks:

- Software-timed and hardware-timed digital input/output tasks
- Counter/timer tasks (can be used in up to two slots)
- Accessing PFI signal tasks (can be used in up to two slots)
- Filter digital input signals

Software-timed and hardware-timed digital input/output tasks have the following restrictions:

- You cannot use parallel and serial modules together on the same hardware-timed task.
- You cannot use serial modules for triggering.
- You cannot do both static and timed tasks at the same time on a single serial module.
- You can only do hardware timing in one direction at a time on a serial bidirectional module.

To determine the capability of digital modules supported by the controller, refer to the *Software Support for CompactRIO, CompactDAQ, Single-Board RIO, R Series, and EtherCAT* document.

Static DIO

Each of the DIO lines can be used as a static DI or DO line. You can use static DIO lines to monitor or control digital signals on some sbRIO I/O modules. Each DIO line can be individually configured as a digital input (DI) or digital output (DO), if the sbRIO I/O module being used allows such configuration.

All samples of static DI lines and updates of static DO lines are software-timed.

Digital Input

You can acquire digital waveforms using either parallel or serial digital modules. The DI waveform acquisition FIFO stores the digital samples. The sbRIO controller samples the DIO lines on each rising or falling edge of the DI Sample Clock signal.

Multiple input timing engines allow the sbRIO controller to run up to eight hardware-timed digital input tasks simultaneously, each using independent timing and triggering configurations. The eight input timing engines are it0, it1,…it7. All eight of the input timing engines are shared between analog input and digital input tasks, allowing up to 8 hardware-timed input tasks.

Digital Input Triggering Signals

A trigger is a signal that causes an action, such as starting or stopping the acquisition of data. When you configure a trigger, you must decide how you want to produce the trigger and the action you want the trigger to cause. The sbRIO controller supports internal software triggering, external digital triggering, analog triggering, and internal time triggering.

Three triggers are available: Start Trigger, Reference Trigger, and Pause Trigger. An analog or digital trigger can initiate these three trigger actions. sbRIO I/O parallel digital input modules can be used in any controller slot to supply a digital trigger.

Digital Input Timing Signals

The sbRIO controller features the following digital input timing signals:

- DI Sample Clock Signal*
- DI Sample Clock Timebase Signal
- DI Start Trigger Signal*
- DI Reference Trigger Signal*
- DI Pause Trigger Signal*

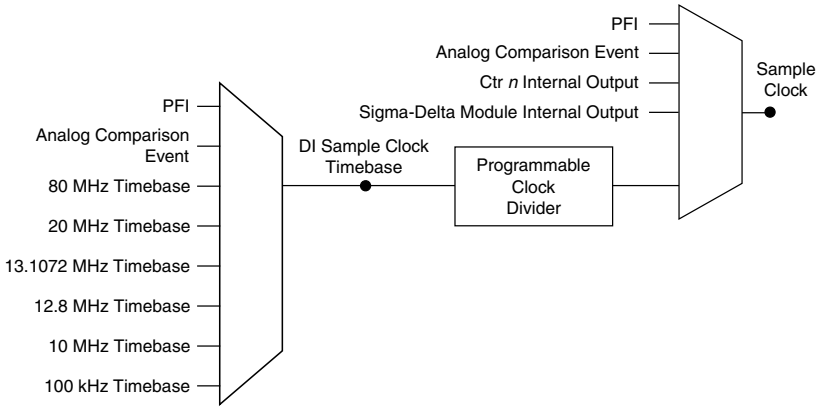
Signals with an * support digital filtering. Refer to the *PFI Filters* section for more information.

DI Sample Clock Signal

Use the DI Sample Clock signal to sample digital I/O on any slot using parallel digital modules, and store the result in the DI waveform acquisition FIFO. If the sbRIO controller receives a DI Sample Clock signal when the FIFO is full, it reports an overflow error to the host software.

A sample consists of one reading from each channel in the DI task. DI Sample Clock signals the start of a sample of all digital input channels in the task. DI Sample Clock can be generated from external or internal sources as shown in the following figure.

Figure 19. DI Sample Clock Timing Options



Routing DI Sample Clock to an Output Terminal

You can route DI Sample Clock to any output PFI terminal.

DI Sample Clock Timebase Signal

The DI Sample Clock Timebase signal is divided down to provide a source for DI Sample Clock. DI Sample Clock Timebase can be generated from external or internal sources. DI Sample Clock Timebase is not available as an output from the controller.

Using an Internal Source

To use DI Sample Clock with an internal source, specify the signal source and the polarity of the signal. Use the following signals as the source:

- it Sample Clock
- ot Sample Clock
- Counter n Internal Output
- Frequency Output
- DI Change Detection Output

Several other internal signals can be routed to DI Sample Clock. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Using an External Source

You can route the following signals as DI Sample Clock:

- Any PFI terminal
- Analog Comparison Event (an analog trigger)

You can sample data on the rising or falling edge of DI Sample Clock.

Routing DI Sample Clock to an Output Terminal

You can route DI Sample Clock to any output PFI terminal. The PFI circuitry inverts the polarity of DI Sample Clock before driving the PFI terminal.

DI Start Trigger Signal

Use the DI Start Trigger signal to begin a measurement acquisition. A measurement acquisition consists of one or more samples. If you do not use triggers, begin a measurement with a software command. Once the acquisition begins, configure the acquisition to stop in one of the following ways:

- When a certain number of points has been sampled (in finite mode)
- After a hardware reference trigger (in finite mode)
- With a software command (in continuous mode)

An acquisition that uses a start trigger (but not a reference trigger) is sometimes referred to as a posttriggered acquisition. That is, samples are measured only after the trigger.

When you are using an internal sample clock, you can specify a delay from the start trigger to the first sample.

Using a Time Source

To use the Start Trigger signal with a time source, configure a specific time in NI-DAQmx. Refer to the "Timestamps" and "Time Triggering" topics in the *NI-DAQmx Help* for more information on accessing time-based features in the NI-DAQmx API.

Using a Digital Source

To use DI Start Trigger with a digital source, specify a source and a rising or falling edge. Use the following signals as the source:

- Any PFI terminal
- Counter n Internal Output

The source also can be one of several other internal signals on the sbRIO controller. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Routing DI Start Trigger to an Output Terminal

You can route DI Start Trigger to any output PFI terminal. The output is an active high pulse.

DI Reference Trigger Signal

Use a reference trigger signal to stop a measurement acquisition. To use a reference trigger, specify a buffer of finite size and a number of pretrigger samples (samples that occur before the reference trigger). The number of posttrigger samples (samples that occur after the reference trigger) desired is the buffer size minus the number of pretrigger samples.

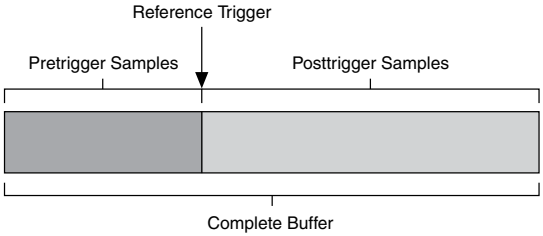
Once the acquisition begins, the sbRIO controller writes samples to the buffer. After the sbRIO controller captures the specified number of pretrigger samples, the controller begins to look for

the reference trigger condition. If the reference trigger condition occurs before the sbRIO controller captures the specified number of pretrigger samples, the controller ignores the condition.

If the buffer becomes full, the sbRIO controller continuously discards the oldest samples in the buffer to make space for the next sample. This data can be accessed (with some limitations) before the sbRIO controller discards it. Refer to the *Can a Pretriggered Acquisition be Continuous?* document for more information.

When the reference trigger occurs, the sbRIO controller continues to write samples to the buffer until the buffer contains the number of posttrigger samples desired. The figure below shows the final buffer.

Figure 20. Reference Trigger Final Buffer



Using a Digital Source

To use DI Reference Trigger with a digital source, specify a source and a rising or falling edge. Either PFI or one of several internal signals on the sbRIO controller can provide the source. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Routing DI Reference Trigger Signal to an Output Terminal

You can route DI Reference Trigger to any output PFI terminal. Reference Trigger is active high by default.

DI Pause Trigger Signal

You can use the DI Pause Trigger signal to pause and resume a measurement acquisition. The internal sample clock pauses while the external trigger signal is active and resumes when the signal is inactive. You can program the active level of the pause trigger to be high or low.

Using a Digital Source

To use DI Pause Trigger, specify a source and a polarity. The source can be either from PFI or one of several other internal signals on your sbRIO controller. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Digital Input Filters

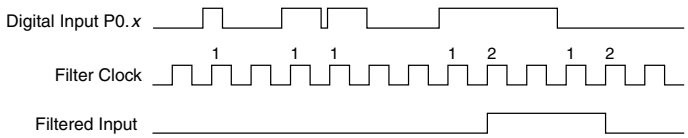
When performing a hardware timed task, you can enable a programmable debouncing filter on the digital input lines of a parallel DIO module. All lines on a module must share the same filter configuration. When the filter is enabled, the controller samples the inputs with a user-configured Filter Clock derived from the controller timebase. This is used to determine whether a pulse is propagated to the rest of the system. However, the filter also introduces jitter onto the input signal.

In NI-DAQmx, the filter is programmed by setting the minimum pulse width, Tp[4], that will pass the filter, and is selectable in 25 ns increments. The appropriate Filter Clock is selected by the driver. Pulses of length less than 1/2 Tp will be rejected, and the filtering behavior of lengths between 1/2 Tp and 1 Tp are not defined because they depend on the phase of the Filter Clock relative to the input signal.

The figure below shows an example of low-to-high transitions of the input signal. High-to-low transitions work similarly.

Assume that an input terminal has been low for a long time. The input terminal then changes from low to high, but glitches several times. When the filter clock has sampled the signal high on consecutive rising edges, the low-to-high transition is propagated to the rest of the circuit.

Figure 21. Filter Example



Getting Started with DI Applications in Software

You can use the sbRIO controller in the following digital input applications:

- Single-point acquisition
- Hardware-Timed Single Point acquisition
- Finite acquisition
- Continuous acquisition

For more information about programming digital input applications and triggers in software, refer to the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Change Detection Event

The Change Detection Event is the signal generated when a change on the rising or falling edge lines is detected by the change detection task.

[4] Tp is a nominal value; the accuracy of the controller timebase and I/O distortion will affect this value.

Routing Change Detection Event to an Output Terminal

You can route ChangeDetectionEvent to any output PFI terminal.

Change Detection Acquisition

You can configure lines on parallel digital modules to detect rising or falling edges. When one or more of these lines sees the edge specified for that line, the sbRIO controller samples all the lines in the task. The rising and falling edge lines do not necessarily have to be in the task.

Change detection acquisitions can only be buffered:

- *Buffered Change Detection Acquisition*—A buffer is a temporary storage in computer memory for acquired samples. In a buffered acquisition, data is stored in the sbRIO controller onboard FIFO then transferred to a PC buffer. Buffered acquisitions typically allow for much faster transfer rates than nonbuffered acquisitions because data accumulates and is transferred in blocks, rather than one sample at a time.

Digital Output

To generate digital output, install a digital output sbRIO I/O module in any slot on the sbRIO controller. The generation specifications, such as the number of channels, channel configuration, update rate, and output range, are determined by the type of sbRIO I/O module used.

With parallel digital output modules (formerly known as hardware-timed modules), you can do multiple software-timed tasks on a single module, as well as mix hardware-timed and software-timed digital output tasks on a single module. On serial digital output modules (formerly known as static digital output modules), you cannot mix hardware-timed and software-timed tasks, but you can run multiple software-timed tasks.

You may have a hardware-timed task or a software-timed task include channels from multiple modules, but a hardware-timed task may not include a mix of channels from both parallel and serial modules. Multiple timing engines allow the sbRIO controller to run up to eight hardware-timed digital output tasks simultaneously, each using independent timing and triggering configurations. The eight output timing engines are ot0, ot1,…ot7. All eight of the output timing engines are shared between analog output and digital output tasks, allowing up to 8 hardware-timed output tasks.

Digital Output Data Generation Methods

When performing a digital output operation, you either can perform software-timed generations or hardware-timed generations.

Software-Timed Generations

With a software-timed generation, software controls the rate at which data is generated. Software sends a separate command to the hardware to initiate each digital generation. In NI-DAQmx, software-timed generations are referred to as on-demand timing. Software-timed generations are also referred to as immediate or static operations. They are typically used for writing out a single value.

For software-timed generations, if any DO channel on a serial digital module is used in a hardware-timed task, no channels on that module can be used in a software-timed task.

Hardware-Timed Generations

With a hardware-timed generation, a digital hardware signal controls the rate of the generation. This signal can be generated internally on the controller or provided externally.

Hardware-timed generations have several advantages over software-timed acquisitions:

- The time between samples can be much shorter.
- The timing between samples is deterministic.
- Hardware-timed acquisitions can use hardware triggering.

Hardware-Timed Single Point (HWTSP) Mode

In HWTSP mode, samples are acquired or generated continuously using hardware timing and no buffer. You must use the sample clock or change detection timing types. No other timing types are supported.

Use HWTSP mode if you need to know if a loop executes in a given amount of time, such as in a control application. Because there is no buffer, if you use HWTSP mode, ensure that reads or writes execute fast enough to keep up with hardware timing. If a read or write executes late, it returns a warning.

Buffered Digital Output

A buffer is a temporary storage in computer memory for generated samples. In a buffered generation, data is moved from a host buffer to the sbRIO controller onboard FIFO before it is written to the sbRIO I/O module(s).

One property of buffered I/O operations is sample mode. The sample mode can be either finite or continuous:

- *Finite*—Finite sample mode generation refers to the generation of a specific, predetermined number of data samples. After the specified number of samples is written out, the generation stops.
- *Continuous*—Continuous generation refers to the generation of an unspecified number of samples. Instead of generating a set number of data samples and stopping, a continuous generation continues until you stop the operation. There are three different continuous generation modes that control how the data is written. These modes are regeneration, onboard regeneration, and non-regeneration:
  - In regeneration mode, you define a buffer in host memory. The data from the buffer is continually downloaded to the FIFO to be written out. New data can be written to the host buffer at any time without disrupting the output.
  - With onboard regeneration, the entire buffer is downloaded to the FIFO and regenerated from there. After the data is downloaded, new data cannot be written to the FIFO. To use onboard regeneration, the entire buffer must fit within the FIFO size. The advantage of using onboard regeneration is that it does not require

communication with the main host memory once the operation is started, which prevents problems that may occur due to excessive bus traffic or operating system latency.

Note Install parallel DO modules in slots 1 through 4 to maximize accessible FIFO size because using a module in slots 5 through 8 will reduce the accessible FIFO size.

– With non-regeneration, old data is not repeated. New data must continually be written to the buffer. If the program does not write new data to the buffer at a fast enough rate to keep up with the generation, the buffer underflows and causes an error.

Digital Output Triggering Signals

A trigger is a signal that causes an action, such as starting or stopping the acquisition of data. When you configure a trigger, you must decide how you want to produce the trigger and the action you want the trigger to cause. The sbRIO controller supports internal software triggering, external digital triggering, analog triggering, and internal time triggering.

Digital output supports two different triggering actions: DO Start Trigger and DO Pause Trigger. A digital or analog trigger can initiate these actions. Any PFI terminal can supply a digital trigger, and some sbRIO I/O analog modules can supply an analog trigger.

Digital Output Timing Signals

The sbRIO controller features the following DO timing signals:

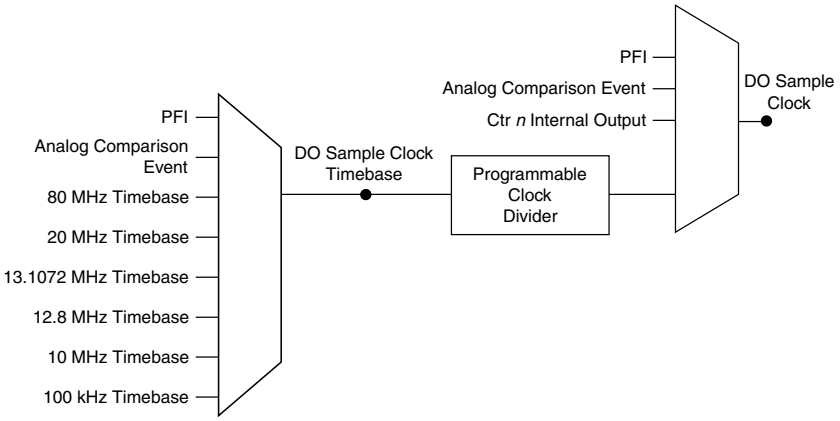
- DO Sample Clock Signal*
- DO Sample Clock Timebase Signal
- DO Start Trigger Signal*
- DO Pause Trigger Signal*

Signals with an * support digital filtering. Refer to the *PFI Filters* section for more information.

DO Sample Clock Signal

The DO Sample Clock signals when all the digital output channels in the task update. DO Sample Clock can be generated from external or internal sources as shown in the image below.

Figure 22. Digital Output Timing Options



Routing DO Sample Clock to an Output Terminal

You can route DO Sample Clock to any output PFI terminal. DO Sample Clock is active high by default.

DO Sample Clock Timebase Signal

The DO Sample Clock Timebase signal is divided down to provide a source for DO Sample Clock. DO Sample Clock Timebase can be generated from external or internal sources and is not available as an output from the controller.

DO Start Trigger Signal

Use the DO Start Trigger signal to initiate a waveform generation. If you do not use triggers, you can begin a generation with a software command. If you are using an internal sample clock, you can specify a delay from the start trigger to the first sample. For more information, refer to the *NI-DAQmx Help*.

Using a Time Source

To use the Start Trigger signal with a time source, configure a specific time in NI-DAQmx. Refer to the "Timestamps" and "Time Triggering" topics in the *NI-DAQmx Help* for more information on accessing time-based features in the NI-DAQmx API.

Using a Digital Source

To use DO Start Trigger, specify a source and a rising or falling edge. The source can be one of the following signals:

- A pulse initiated by host software
- Any PFI terminal

- AI Reference Trigger
- AI Start Trigger

The source also can be one of several internal signals on the sbRIO controller. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

You also can specify whether the waveform generation begins on the rising edge or falling edge of DO Start Trigger.

Routing DO Start Trigger Signal to an Output Terminal

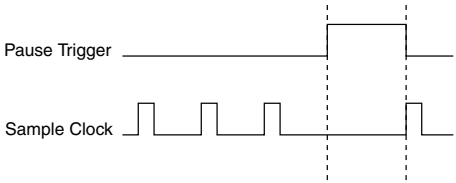
You can route DO Start Trigger to any output PFI terminal. The output is an active high pulse.

DO Pause Trigger Signal

Use the DO Pause Trigger signal to mask off samples in a DAQ sequence. When DO Pause Trigger is active, no samples occur, but DO Pause Trigger does not stop a sample that is in progress. The pause does not take effect until the beginning of the next sample.

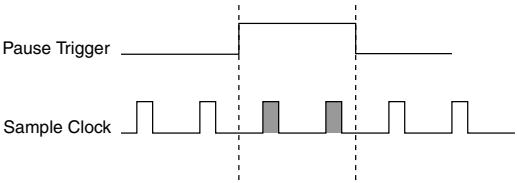
When you generate digital output signals, the generation pauses as soon as the pause trigger is asserted. If the source of the sample clock is the onboard clock, the generation resumes as soon as the pause trigger is deasserted, as shown in the figure below.

Figure 23. DO Pause Trigger with the Onboard Clock Source



If you are using any signal other than the onboard clock as the source of the sample clock, the generation resumes as soon as the pause trigger is deasserted and another edge of the sample clock is received, as shown in the figure below.

Figure 24. DO Pause Trigger with Other Signal Source



Using a Digital Source

To use DO Pause Trigger, specify a source and a polarity. The source can be a PFI signal or one of several other internal signals on the sbRIO controller.

You also can specify whether the samples are paused when DO Pause Trigger is at a logic high or low level. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Getting Started with DO Applications in Software

You can use the sbRIO controller in the following digital output applications:

- Single-point (on-demand) generation
- Hardware-Timed Single Point generation
- Finite generation
- Continuous generation

For more information about programming digital output applications and triggers in software, refer to the *NI-DAQmx Help* or to the *LabVIEW Help*.

Digital Input/Output Configuration for NI 9401

When you change the configuration of lines on a NI 9401 digital module between input and output, NI-DAQmx temporarily reserves all of the lines on the module for communication to send the module a line configuration command. For this reason, you must reserve the task in advance through the DAQmx Control Task before any task has started. If another task or route is actively using the module, to avoid interfering with the other task, NI-DAQmx generates an error instead of sending the line configuration command. During the line configuration command, the output lines are maintained without glitching.

PFI with NI-DAQmx

You can configure channels of a parallel digital module as Programmable Function Interface (PFI) terminals. The sbRIO controller also provides one terminal for PFI. Up to two digital modules can be used to access PFI terminals in a single controller

You can configure each PFI individually as the following:

- Timing input signal for AI, AO, DI, DO, or counter/timer functions
- Timing output signal from AI, AO, DI, DO, or counter/timer functions

PFI Filters

You can enable a programmable debouncing filter on each PFI signal. When the filter is enabled, the controller samples the inputs with a user-configured Filter Clock derived from the controller timebase. This is used to determine whether a pulse is propagated to the rest of the circuit.

However, the filter also introduces jitter onto the PFI signal.

The following is an example of low-to-high transitions of the input signal. High-to-low transitions work similarly.

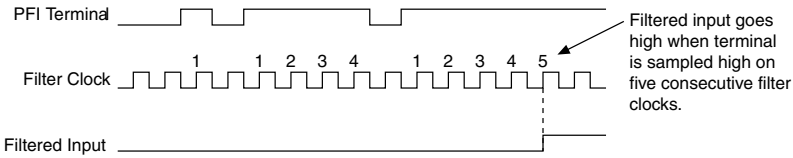
Assume that an input terminal has been low for a long time. The input terminal then changes from low to high, but glitches several times. When the Filter Clock has sampled the signal high on N consecutive edges, the low-to-high transition is propagated to the rest of the circuit. The value of N depends on the filter setting, as shown in the following table.

Table 21. Selectable PFI Filter Settings

Filter Setting	Filter Clock	Jitter	Min Pulse Width* to Pass	Max Pulse Width* to Not Pass
112.5 ns (short)	80 MHz	12.5 ns	112.5 ns	100 ns
6.4 µs (medium)	80 MHz	12.5 ns	6.4 µs	6.3875 µs
2.56 ms (high)	100 kHz	10 µs	2.56 ms	2.55 ms
Custom	User-configurable	1 Filter Clock period	T_{user}	T_{user} - (1 Filter Clock period)
* Pulse widths are nominal values; the accuracy of the controller timebase and I/O distortion will affect these values.				

On power up, the filters are disabled. The figure below shows an example of a low-to-high transition on an input that has a custom filter set to N = 5.

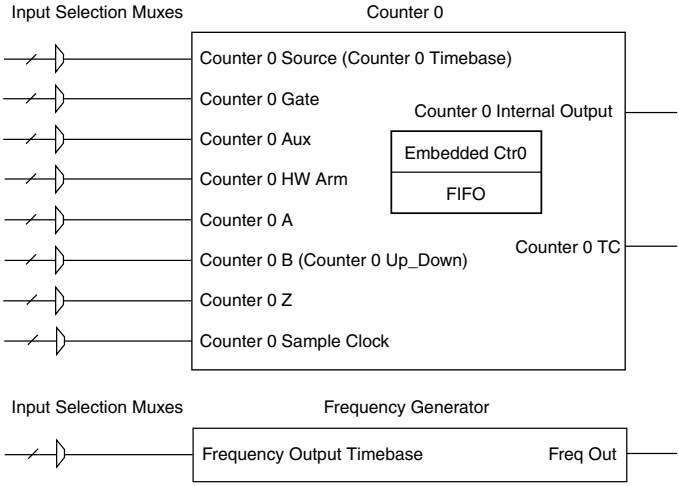
Figure 25. PFI Filter Example



Counters with NI-DAQmx

The sbRIO controller has four general-purpose 32-bit counter/timers and one frequency generator. The general-purpose counter/timers can be used for many measurement and pulse generation applications. The figure below shows the sbRIO controller Counter 0 and the frequency generator. All four counters on the sbRIO controller are identical.

Figure 26. Controller Counter 0 and Frequency Generator



Counters have eight input signals, although in most applications only a few inputs are used.

Each counter has a FIFO that can be used for buffered acquisition and generation. Each counter also contains an embedded counter (Embedded Ctr*n*) for use in what are traditionally two-counter measurements and generations. The embedded counters cannot be programmed independent of the main counter; signals from the embedded counters are not routable.

Counter Timing Engine

Unlike analog input, analog output, digital input, and digital output, the sbRIO controller counters do not have the ability to divide down a timebase to produce an internal counter sample clock. For sample clocked operations, an external signal must be provided to supply a clock source. The source can be any of the following signals:

- AI Sample Clock
- AI Start Trigger
- AI Reference Trigger
- AO Sample Clock
- DI Sample Clock
- DI Start Trigger
- DO Sample Clock
- CTR *n* Internal Output
- Freq Out
- PFI
- Change Detection Event
- Analog Comparison Event

Not all timed counter operations require a sample clock. For example, a simple buffered pulse width measurement latches in data on each edge of a pulse. For this measurement, the measured signal determines when data is latched in. These operations are referred to as implicit timed operations. However, many of the same measurements can be clocked at an interval with a sample clock. These are referred to as sample clocked operations. The following table shows the different options for the different measurements.

Table 22. Counter Timing Measurements

Measurement	Implicit Timing Support	Sample Clocked Timing Support
Buffered Edge Count	No	Yes
Buffered Pulse Width	Yes	Yes
Buffered Pulse	Yes	Yes
Buffered Semi-Period	Yes	No
Buffered Frequency	Yes	Yes
Buffered Period	Yes	Yes
Buffered Position	No	Yes
Buffered Two-Signal Edge Separation	Yes	Yes

Counter Triggering

Counters support three different triggering actions:

- *Arm Start Trigger*—To begin any counter input or output function, you must first enable, or arm, the counter. Software can arm a counter or configure counters to be armed on a hardware signal. Software calls this hardware signal the Arm Start Trigger. Internally, software routes the Arm Start Trigger to the Counter n HW Arm input of the counter.

  For counter output operations, you can use it in addition to the start and pause triggers. For counter input operations, you can use the arm start trigger to have start trigger-like behavior. The arm start trigger can be used for synchronizing multiple counter input and output tasks.

  When using an arm start trigger, the arm start trigger source is routed to the Counter n HW Arm signal.
- *Start Trigger*—You can use the start trigger for counter output functions. You can configure a start trigger to begin a finite or continuous pulse generation. After a continuous generation has triggered, the pulses continue to generate until you stop the operation in software. For finite generations, the specified number of pulses is generated and the generation stops unless you use the retriggerable attribute. When you use this attribute, subsequent start triggers cause the generation to restart.

When using a start trigger, the start trigger source is routed to the Counter n Gate signal input of the counter. Possible triggers for counter output operations are a hardware signal.

For counter input functions, you can use the arm start trigger to have start trigger-like behavior.

- *Pause Trigger*—You can use pause triggers in edge counting and continuous pulse generation applications. For edge counting acquisitions, the counter stops counting edges while the external trigger signal is low and resumes when the signal goes high or vice versa.

  For continuous pulse generations, the counter stops generating pulses while the external trigger signal is low and resumes when the signal goes high or vice versa.

  When using a pause trigger, the pause trigger source is routed to the Counter n Gate signal input of the counter.

Default Counter/Timer Routing

Counter/timer signals are available to parallel digital sbRIO I/O modules. To determine the signal routing options for modules installed in your system, refer to the **Device Routes** tab in MAX.

You can use these defaults or select other sources and destinations for the counter/timer signals in NI-DAQmx. Refer to "Connecting Counter Signals" in the *NI-DAQmx Help* for more information about how to connect your signals for common counter measurements and generations. Refer to "Physical Channels" in the *NI-DAQmx Help* for a list of default PFI lines for counter functions.

Other Counter Features

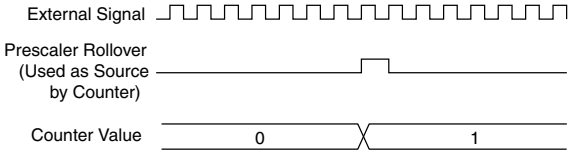
Cascading Counters

You can internally route the Counter *n* Internal Output and Counter *n* TC signals of each counter to the Gate inputs of the other counter. By cascading two counters together, you can effectively create a 64-bit counter. By cascading counters, you also can enable other applications. For example, to improve the accuracy of frequency measurements, use reciprocal frequency measurement, as described in the *Large Range of Frequencies with Two Counters* section.

Prescaling

Prescaling allows the counter to count a signal that is faster than the maximum timebase of the counter. The sbRIO controller offers 8X and 2X prescaling on each counter. Prescaling can be disabled. Each prescaler consists of a small, simple counter that counts to eight (or two) and rolls over. This counter can run faster than the larger counters, which simply count the rollovers of this smaller counter. Thus, the prescaler acts as a frequency divider on the Source and puts out a frequency that is one-eighth (or one-half) of what it is accepting as shown in the figure below.

Figure 27. Prescaling



Prescaling is intended to be used for frequency measurement where the measurement is made on a continuous, repetitive signal. The prescaling counter cannot be read; therefore, you cannot determine how many edges have occurred since the previous rollover. Prescaling can be used for event counting provided it is acceptable to have an error of up to seven (or one) ticks. Prescaling can be used when the counter Source is an external signal. Prescaling is not available if the counter Source is one of the internal timebases (80 MHz Timebase, 20 MHz Timebase, or 100 kHz Timebase).

Synchronization Modes

The 32-bit counter counts up or down synchronously with the Source signal. The Gate signal and other counter inputs are asynchronous to the Source signal, so the sbRIO controller synchronizes these signals before presenting them to the internal counter.

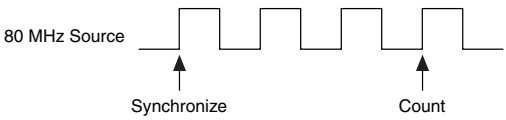
Depending on how you configure your controller, the sbRIO controller uses one of two synchronization methods:

- 80 MHz Source Mode
- External or Internal Source Less than 20 MHz

80 MHz Source Mode

In 80 MHz source mode, the controller synchronizes signals on the rising edge of the source, and counts on the third rising edge of the source. Edges are pipelined so no counts are lost, as shown in the figure below.

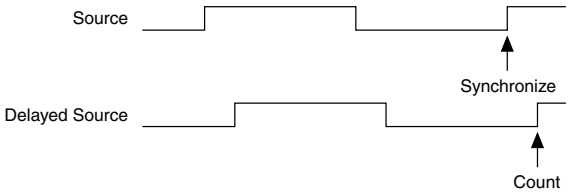
Figure 28. 80 MHz Source Mode



External or Internal Source Less than 20 MHz

With an external or internal source less than 20 MHz, the module generates a delayed Source signal by delaying the Source signal by several nanoseconds. The controller synchronizes signals on the rising edge of the delayed Source signal, and counts on the following rising edge of the source, as shown in the figure below.

Figure 29. External or Internal Source Less than 20 MHz



Counter Input Applications

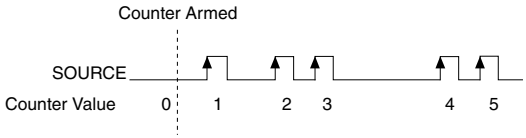
Counting Edges

In edge counting applications, the counter counts edges on its Source after the counter is armed. You can configure the counter to count rising or falling edges on its Source input. You also can control the direction of counting (up or down), as described in the *Controlling the Direction of Counting* section. The counter values can be read on demand or with a sample clock.

Single Point (On-Demand) Edge Counting

With single point (on-demand) edge counting, the counter counts the number of edges on the Source input after the counter is armed. On-demand refers to the fact that software can read the counter contents at any time without disturbing the counting process. The following figure shows an example of single point edge counting.

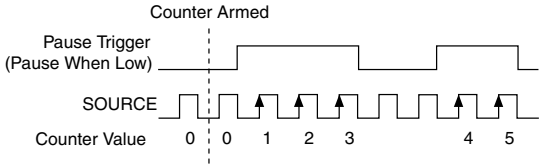
Figure 30. Single Point (On-Demand) Edge Counting



You also can use a pause trigger to pause (or gate) the counter. When the pause trigger is active, the counter ignores edges on its Source input. When the pause trigger is inactive, the counter counts edges normally.

You can route the pause trigger to the Gate input of the counter. You can configure the counter to pause counting when the pause trigger is high or when it is low. The following figure shows an example of on-demand edge counting with a pause trigger.

Figure 31. Single Point (On-Demand) Edge Counting with Pause Trigger



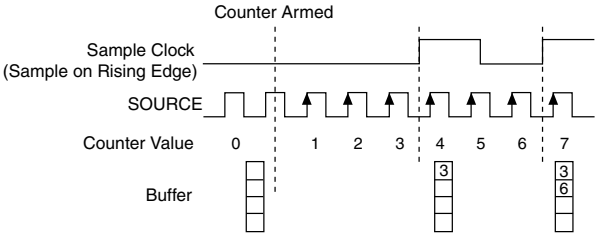
Buffered (Sample Clock) Edge Counting

With buffered edge counting (edge counting using a sample clock), the counter counts the number of edges on the Source input after the counter is armed. The value of the counter is sampled on each active edge of a sample clock and stored in the FIFO. The sampled values will be transferred to host memory using a high-speed data stream.

The count values returned are the cumulative counts since the counter armed event. That is, the sample clock does not reset the counter. You can configure the counter to sample on the rising or falling edge of the sample clock.

The following figure shows an example of buffered edge counting. Notice that counting begins when the counter is armed, which occurs before the first active edge on Sample Clock.

Figure 32. Buffered (Sample Clock) Edge Counting



Controlling the Direction of Counting

In edge counting applications, the counter can count up or down. You can configure the counter to do the following:

- Always count up
- Always count down
- Count up when the Counter 0 B input is high; count down when it is low

Pulse-Width Measurement

In pulse-width measurements, the counter measures the width of a pulse on its Gate input signal. You can configure the counter to measure the width of high pulses or low pulses on the Gate signal.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges on the Source signal while the pulse on the Gate signal is active.

You can calculate the pulse width by multiplying the period of the Source signal by the number of edges returned by the counter.

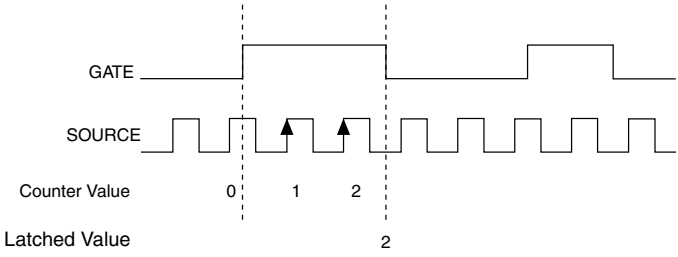
A pulse-width measurement will be accurate even if the counter is armed while a pulse train is in progress. If a counter is armed while the pulse is in the active state, it will wait for the next transition to the active state to begin the measurement.

Single Pulse-Width Measurement

With single pulse-width measurement, the counter counts the number of edges on the Source input while the Gate input remains active. When the Gate input goes inactive, the counter stores the count in the FIFO and ignores other edges on the Gate and Source inputs. Software then reads the stored count.

The following figure shows an example of a single pulse-width measurement.

Figure 33. Single Pulse-Width Measurement



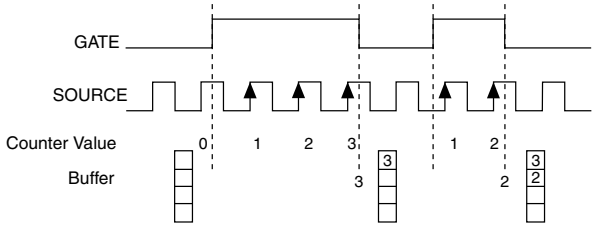
Implicit Buffered Pulse-Width Measurement

An implicit buffered pulse-width measurement is similar to single pulse-width measurement, but buffered pulse-width measurement takes measurements over multiple pulses.

The counter counts the number of edges on the Source input while the Gate input remains active. On each trailing edge of the Gate signal, the counter stores the count in the counter FIFO. The sampled values will be transferred to host memory using a high-speed data stream.

The following figure shows an example of an implicit buffered pulse-width measurement.

Figure 34. Implicit Buffered Pulse-Width Measurement



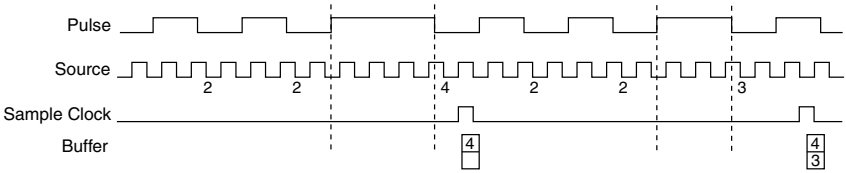
Sample Clocked Buffered Pulse-Width Measurement

A sample clocked buffered pulse-width measurement is similar to single pulse-width measurement, but buffered pulse-width measurement takes measurements over multiple pulses correlated to a sample clock.

The counter counts the number of edges on the Source input while the Gate input remains active. On each sample clock edge, the counter stores the count in the FIFO of the last pulse width to complete. The sampled values will be transferred to host memory using a high-speed data stream.

The following figure shows an example of a sample clocked buffered pulse-width measurement.

Figure 35. Sample Clocked Buffered Pulse-Width Measurement



Note If a pulse does not occur between sample clocks, an overrun error occurs.

Pulse Measurement

In pulse measurements, the counter measures the high and low time of a pulse on its Gate input signal after the counter is armed. A pulse is defined in terms of its high and low time, high and low ticks or frequency and duty cycle. This is similar to the pulse-width measurement, except that the inactive pulse is measured as well.

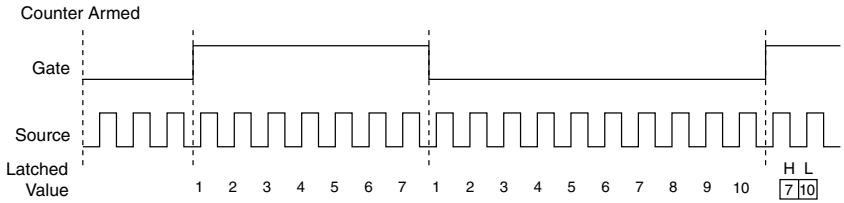
You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges occurring on the Source input between two edges of the Gate signal.

You can calculate the high and low time of the Gate input by multiplying the period of the Source signal by the number of edges returned by the counter.

Single Pulse Measurement

Single (on-demand) pulse measurement is equivalent to two single pulse-width measurements on the high (H) and low (L) ticks of a pulse, as shown in the figure below.

Figure 36. Single (On-Demand) Pulse Measurement



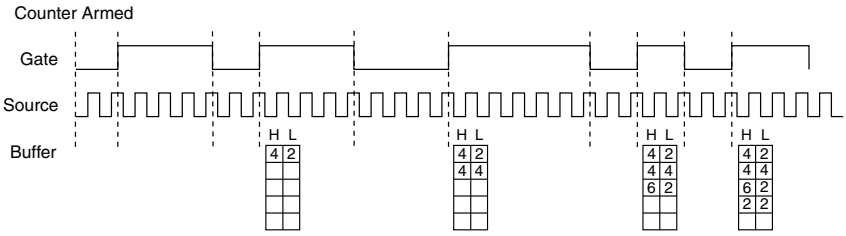
Implicit Buffered Pulse Measurement

In an implicit buffered pulse measurement, on each edge of the Gate signal, the counter stores the count in the FIFO. The sampled values will be transferred to host memory using a high-speed data stream.

The counter begins counting when it is armed. The arm usually occurs between edges on the Gate input but the counting does not start until the desired edge. You can select whether to read the high pulse or low pulse first using the **StartingEdge** property in NI-DAQmx.

The figure below shows an example of an implicit buffered pulse measurement.

Figure 37. Implicit Buffered Pulse Measurement



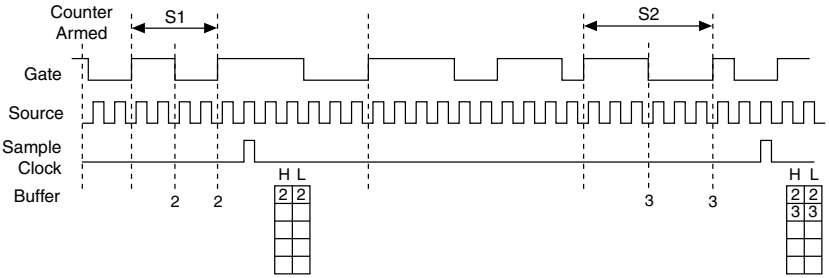
Sample Clocked Buffered Pulse Measurement

A sample clocked buffered pulse measurement is similar to single pulse measurement, but a buffered pulse measurement takes measurements over multiple pulses correlated to a sample clock.

The counter performs a pulse measurement on the Gate. On each sample clock edge, the counter stores the high and low ticks in the FIFO of the last pulse to complete. The sampled values will be transferred to host memory using a high-speed data stream.

The figure below shows an example of a sample clocked buffered pulse measurement.

Figure 38. Sample Clocked Buffered Pulse Measurement



Note If a pulse does not occur between sample clocks, an overrun error occurs.

Semi-Period Measurement

In semi-period measurements, the counter measures a semi-period on its Gate input signal after the counter is armed. A semi-period is the time between any two consecutive edges on the Gate input.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges occurring on the Source input between two edges of the Gate signal.

You can calculate the semi-period of the Gate input by multiplying the period of the Source signal by the number of edges returned by the counter.

Single Semi-Period Measurement

Single semi-period measurement is equivalent to single pulse-width measurement.

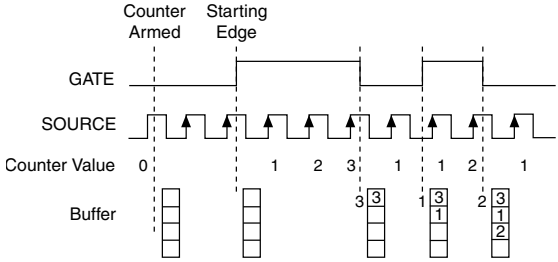
Implicit Buffered Semi-Period Measurement

In implicit buffered semi-period measurements, on each edge of the Gate signal, the counter stores the count in the FIFO. The sampled values will be transferred to host memory using a high-speed data stream.

The counter begins counting when it is armed. The arm usually occurs between edges on the Gate input. You can select whether to read the first active low or active high semi-period using the **CI.SemiPeriod.StartingEdge** property in NI-DAQmx.

The following figure shows an example of an implicit buffered semi-period measurement.

Figure 39. Implicit Buffered Semi-Period Measurement



Pulse versus Semi-Period Measurements

In hardware, pulse measurement and semi-period are the same measurement. Both measure the high and low times of a pulse. The functional difference between the two measurements is how the data is returned. In a semi-period measurement, each high or low time is considered one point of data and returned in units of seconds or ticks. In a pulse measurement, each pair of high and low times is considered one point of data and returned as a paired sample in units of frequency and duty cycle, high and low time or high and low ticks. When reading data, 10 points in a semi-period measurement will get an array of five high times and five low times. When you read 10 points in a pulse measurement, you get an array of 10 pairs of high and low times.

Also, pulse measurements support sample clock timing while semi-period measurements do not.

Frequency Measurement

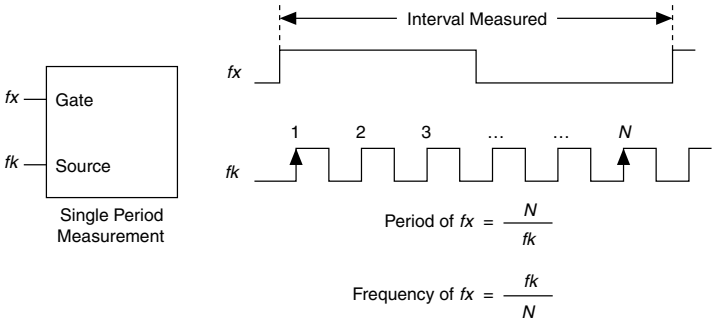
Low Frequency with One Counter

For low frequency measurements with one counter, you measure one period of your signal using a known timebase.

You can route the signal to measure (*fx*) to the Gate of a counter. You can route a known timebase (*fk*) to the Source of the counter. The known timebase can be an onboard timebase, such as 80 MHz Timebase, 20 MHz Timebase, or 100 kHz Timebase, or any other signal with a known rate.

You can configure the counter to measure one period of the gate signal. The frequency of *fx* is the inverse of the period. The following figure illustrates this method.

Figure 40. Low Frequency with One Counter



High Frequency with Two Counters

For high frequency measurements with two counters, you measure one pulse of a known width using your signal and derive the frequency of your signal from the result.

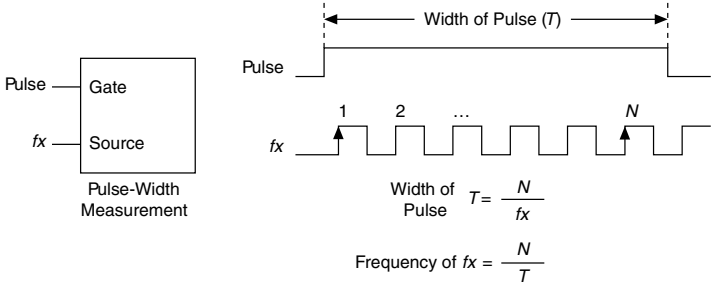
Note Counter 0 is always paired with Counter 1. Counter 2 is always paired with Counter 3.

In this method, you route a pulse of known duration (*T*) to the Gate of a counter. You can generate the pulse using a second counter. You also can generate the pulse externally and connect it to a PFI terminal. You only need to use one counter if you generate the pulse externally.

Route the signal to measure (*fx*) to the Source of the counter. Configure the counter for a single pulse-width measurement. If you measure the width of pulse *T* to be *N* periods of *fx*, the frequency of *fx* is *N*/*T*.

The image below illustrates this method. Another option is to measure the width of a known period instead of a known pulse.

Figure 41. High Frequency with Two Counters



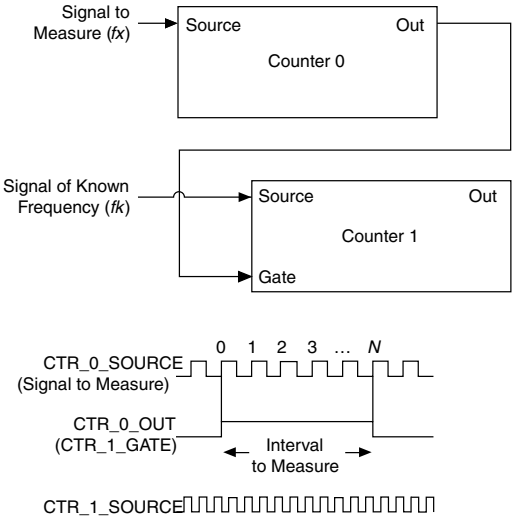
Large Range of Frequencies with Two Counters

By using two counters, you can accurately measure a signal that might be high or low frequency. This technique is called reciprocal frequency measurement. When measuring a large range of frequencies with two counters, you generate a long pulse using the signal to measure. You then measure the long pulse with a known timebase. The sbRIO controller can measure this long pulse more accurately than the faster input signal.

Note Counter 0 is always paired with Counter 1. Counter 2 is always paired with Counter 3.

You can route the signal to measure to the Source input of Counter 0, as shown in the following figure. Assume this signal to measure has frequency *fx*. NI-DAQmx automatically configures Counter 0 to generate a single pulse that is the width of *N* periods of the source input signal.

Figure 42. Large Range of Frequencies with Two Counters



Next, route the Counter 0 Internal Output signal to the Gate input of Counter 1. You can route a signal of known frequency (*fk*) to the Counter 1 Source input. Configure Counter 1 to perform a single pulse-width measurement. Suppose the result is that the pulse width is *J* periods of the *fk* clock.

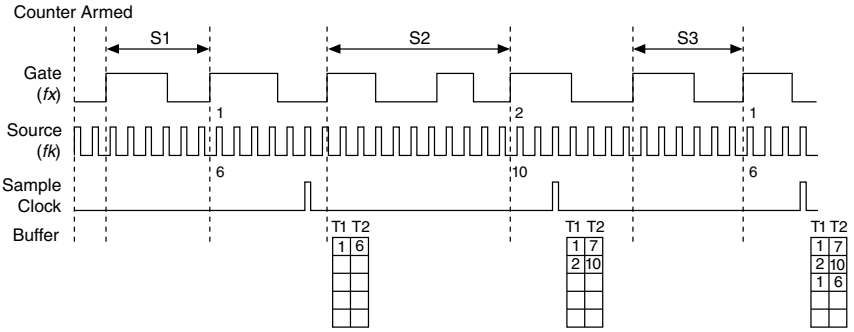
From Counter 0, the length of the pulse is *N*/*fx*. From Counter 1, the length of the same pulse is *J*/*fk*. Therefore, the frequency of *fx* is given by *fx* = *fk* * (*N*/*J*).

Sample Clocked Buffered Frequency Measurement

Sample clocked buffered point frequency measurements can either be a single frequency measurement or an average between sample clocks. Use **CI.Freq.EnableAveraging** to set the behavior. For buffered frequency, the default is True.

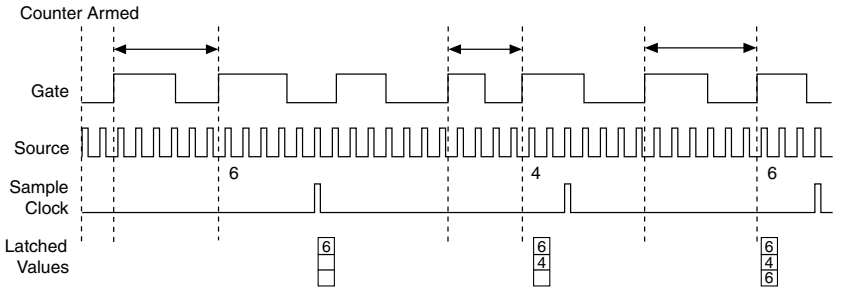
A sample clocked buffered frequency measurement with **CI.Freq.EnableAveraging** set to True uses the embedded counter and a sample clock to perform a frequency measurement. For each sample clock period, the embedded counter counts the signal to measure (*fx*) and the primary counter counts the internal time-base of a known frequency (*fk*). Suppose T1 is the number of ticks of the unknown signal counted between sample clocks and T2 is the number of ticks counted of the known timebase as shown in the following figure. The frequency measured is *fx* = *fk* * (T1/T2).

Figure 43. Sample Clocked Buffered Frequency Measurement (Averaging)



When **CI.Freq.EnableAveraging** is set to False, the frequency measurement returns the frequency of the pulse just before the sample clock. This single measurement is a single frequency measurement and is not an average between clocks as shown in the following figure.

Figure 44. Sample Clocked Buffered Frequency Measurement (Non-Averaging)



With sample clocked frequency measurements, ensure that the frequency to measure is twice as fast as the sample clock to prevent a measurement overflow.

Choosing a Method for Measuring Frequency

The best method to measure frequency depends on several factors including the expected frequency of the signal to measure, the desired accuracy, how many counters are available, and how long the measurement can take. For all frequency measurement methods, assume the following:

fx	is the frequency to be measured if no error
fk	is the known source or gate frequency
Measurement Time (*T*)	is the time it takes to measure a single sample
Divide down (*N*)	is the integer to divide down measured frequency, only used in large range two counters
fs	is the sample clock rate, only used in sample clocked frequency measurements

Here is how these variables apply to each method, summarized in the table below.

- *One counter*—With one counter measurements, a known timebase is used for the source frequency (*fk*). The measurement time is the period of the frequency to be measured, or 1/*fx*.
- *Two counter high frequency*—With the two counter high frequency method, the second counter provides a known measurement time. The gate frequency equals 1/*measurement time*.
- *Two counter large range*—The two counter larger range measurement is the same as a one counter measurement, but now the user has an integer divide down of the signal. An internal timebase is still used for the source frequency (*fk*), but the divide down means that the measurement time is the period of the divided down signal, or *N*/*fx* where *N* is the divide down.
- Sample clocked—For sample clocked frequency measurements, a known timebase is counted for the source frequency (*fk*). The measurement time is the period of the sample clock (*fs*).

Table 23. Frequency Measurement Methods

Variable	Sample Clocked	One Counter	Two Counters	
			High Frequency	Large Range
fk	Known timebase	Known timebase	$\frac{1}{\text{gating period}}$	Known timebase
Measurement time	$\frac{1}{fs}$	$\frac{1}{fx}$	gating period	$\frac{N}{fx}$

Table 23. Frequency Measurement Methods (Continued)

Variable	Sample Clocked	One Counter	Two Counters	
			High Frequency	Large Range
Max. frequency error	$fx \times \frac{fx}{fk \times \left\lceil \frac{fx}{fs} - 1 \right\rceil}$	$fx \times \frac{fx}{fk - fx}$	fk	$fx \times \frac{fx}{N \times fk - fx}$
Max. error %	$\frac{fx}{fk \times \left\lceil \frac{fx}{fs} - 1 \right\rceil}$	$\frac{fx}{fk - fx}$	$\frac{fk}{fx}$	$\frac{fx}{N \times fk - fx}$

Note Accuracy equations do not take clock stability into account. Refer to the device specifications for your model for information about clock stability.

Which Method Is Best?

This depends on the frequency to be measured, the rate at which you want to monitor the frequency and the accuracy you desire. Take for example, measuring a 50 kHz signal. Assuming that the measurement times for the sample clocked (with averaging) and two counter frequency measurements are configured the same, the following table summarizes the results.

Table 24. 50 kHz Frequency Measurement Methods

Variable	Sample Clocked	One Counter	Two Counters	
			High Frequency	Large Range
fx	50,000	50,000	50,000	50,000
fk	80 M	80 M	1,000	80 M
Measurement time (*ms*)	1	.02	1	1
N	—	—	—	—
Max. frequency error (Hz)	.638	31.27	1,000	.625
Max. error %	.00128	.0625	2	.00125

From this, you can see that while the measurement time for one counter is shorter, the accuracy is best in the sample clocked and two counter large range measurements. For another example, the following table shows the results for 5 MHz.

Table 25. 5 MHz Frequency Measurement Methods

Variable	Sample Clocked	One Counter	Two Counters	
			High Frequency	Large Range
fx	5 M	5 M	5 M	5 M
fk	80 M	80 M	1,000	80 M
Measurement time (*ms*)	1	.0002	1	1
N	—	—	—	5,000
Max. frequency error (Hz)	62.51	333 k	1,000	62.50
Max. error %	.00125	6.67	.02	.00125

Again, the measurement time for the one counter measurement is lowest but the accuracy is lower. Note that the accuracy and measurement time of the sample clocked and two counter large range are almost the same. The advantage of the sample clocked method is that even when the frequency to measure changes, the measurement time does not and error percentage varies little. For example, if you configured a large range two-counter measurement to use a divide down of 50 for a 50 k signal, then you would get the measurement time and accuracy listed in the *50 kHz Frequency Measurement Methods* table. But if your signal ramped up to 5 M, then with a divide down of 50, your measurement time is 0.01 ms, but your error is now 0.125%. The error with a sample clocked frequency measurement is not as dependent on the measured frequency so at 50 k and 5 M with a measurement time of 1 ms the error percentage is still close to 0.00125%. One of the disadvantages of a sample clocked frequency measurement is that the frequency to be measured must be at least twice the sample clock rate to ensure that a full period of the frequency to be measured occurs between sample clocks.

- Low frequency measurements with one counter is a good method for many applications. However, the accuracy of the measurement decreases as the frequency increases.
- High frequency measurements with two counters is accurate for high frequency signals. However, the accuracy decreases as the frequency of the signal to measure decreases. At very low frequencies, this method may be too inaccurate for your application. Another disadvantage of this method is that it requires two counters (if you cannot provide an external signal of known width). An advantage of high frequency measurements with two counters is that the measurement completes in a known amount of time.
- Measuring a large range of frequencies with two counters measures high and low frequency signals accurately. However, it requires two counters, and it has a variable sample time and variable error % dependent on the input signal.

The following table summarizes some of the differences in methods of measuring frequency.

Table 26. 5 MHz Frequency Measurement Methods

Method Comparison	Sample Clocked (Averaged)	One Counter	Two Counters	
			High Frequency	Large Range
Number of counters used	1	1	1 or 2	2
Number of measurements returned	1	1	1	1
Measures high frequency signals accurately	Good	Poor	Good	Good
Measures low frequency signals accurately	Good	Good	Good	Poor

Period Measurement

In period measurements, the counter measures a period on its Gate input signal after the counter is armed. You can configure the counter to measure the period between two rising edges or two falling edges of the Gate input signal.

You can route an internal or external periodic clock signal (with a known period) to the Source input of the counter. The counter counts the number of rising (or falling) edges occurring on the Source input between the two active edges of the Gate signal.

You can calculate the period of the Gate input by multiplying the period of the Source signal by the number of edges returned by the counter.

Period measurements return the inverse results of frequency measurements.

Position Measurement

You can use the counters to perform position measurements with quadrature encoders or two-pulse encoders. You can measure angular position with X1, X2, and X4 angular encoders. Linear position can be measured with two-pulse encoders. You can choose to do either a single point (on-demand) position measurement or a buffered (sample clock) position measurement. You must arm a counter to begin position measurements.

Measurements Using Quadrature Encoders

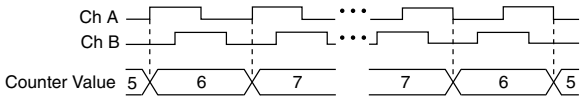
The counters can perform measurements of quadrature encoders that use X1, X2, or X4 encoding. A quadrature encoder can have up to three channels—channels A, B, and Z.

- *X1 Encoding*—When channel A leads channel B in a quadrature cycle, the counter increments. When channel B leads channel A in a quadrature cycle, the counter

decrements. The amount of increments and decrements per cycle depends on the type of encoding—X1, X2, or X4.

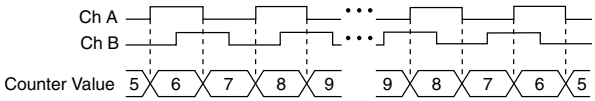
The following figure shows a quadrature cycle and the resulting increments and decrements for X1 encoding. When channel A leads channel B, the increment occurs on the rising edge of channel A. When channel B leads channel A, the decrement occurs on the falling edge of channel A.

Figure 45. X1 Encoding



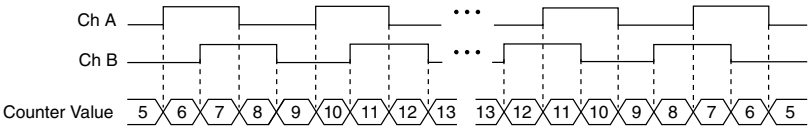
- *X2 Encoding*—The same behavior holds for X2 encoding except the counter increments or decrements on each edge of channel A, depending on which channel leads the other. Each cycle results in two increments or decrements, as shown in the following figure.

Figure 46. X2 Encoding



- *X4 Encoding*—Similarly, the counter increments or decrements on each edge of channels A and B for X4 encoding. Whether the counter increments or decrements depends on which channel leads the other. Each cycle results in four increments or decrements, as shown in the following figure.

Figure 47. X4 Encoding



Channel Z Behavior

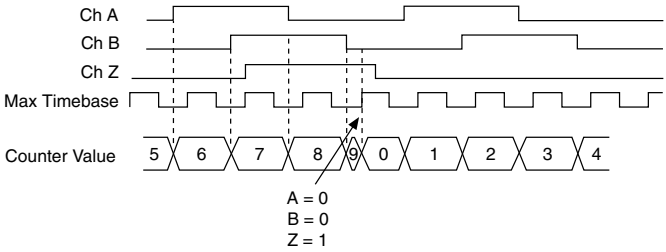
Some quadrature encoders have a third channel, channel Z, which is also referred to as the index channel. A high level on channel Z causes the counter to be reloaded with a specified value in a specified phase of the quadrature cycle. You can program this reload to occur in any one of the four phases in a quadrature cycle.

Channel Z behavior—when it goes high and how long it stays high—differs with quadrature encoder designs. You must refer to the documentation for your quadrature encoder to obtain timing of channel Z with respect to channels A and B. You must then ensure that channel Z is high during at least a portion of the phase you specify for reload. For instance, in the image

below, channel Z is never high when channel A is high and channel B is low. Thus, the reload must occur in some other phase.

In the following figure, the reload phase is when both channel A and channel B are low. The reload occurs when this phase is true and channel Z is high. Incrementing and decrementing takes priority over reloading. Thus, when the channel B goes low to enter the reload phase, the increment occurs first. The reload occurs within one maximum timebase period after the reload phase becomes true. After the reload occurs, the counter continues to count as before. The figure below illustrates channel Z reload with X4 decoding.

Figure 48. Channel Z Reload with X4 Decoding

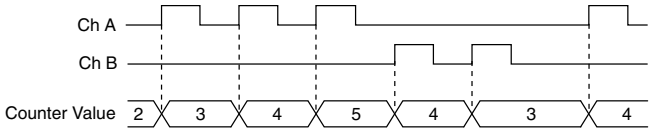


Measurements Using Two Pulse Encoders

The counter supports two pulse encoders that have two channels—channels A and B.

The counter increments on each rising edge of channel A. The counter decrements on each rising edge of channel B, as shown in the following figure.

Figure 49. Measurements Using Two Pulse Encoders

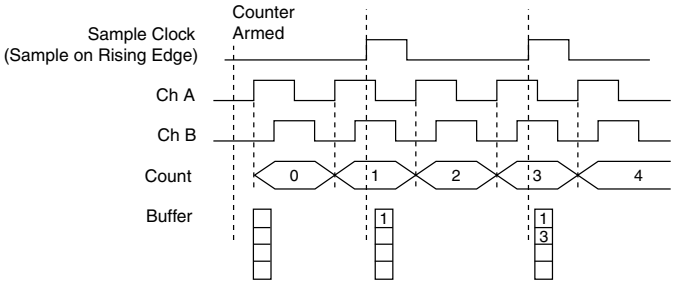


Buffered (Sample Clock) Position Measurement

With buffered position measurement (position measurement using a sample clock), the counter increments based on the encoding used after the counter is armed. The value of the counter is sampled on each active edge of a sample clock. The sampled values will be transferred to host memory using a high-speed data stream. The count values returned are the cumulative counts since the counter armed event; that is, the sample clock does not reset the counter. You can route the counter sample clock to the Gate input of the counter. You can configure the counter to sample on the rising or falling edge of the sample clock.

The figure below shows an example of a buffered X1 position measurement.

Figure 50. Buffered Position Measurement



Two-Signal Edge-Separation Measurement

Two-signal edge-separation measurement is similar to pulse-width measurement, except that there are two measurement signals—Aux and Gate. An active edge on the Aux input starts the counting and an active edge on the Gate input stops the counting. You must arm a counter to begin a two edge separation measurement.

After the counter has been armed and an active edge occurs on the Aux input, the counter counts the number of rising (or falling) edges on the Source. The counter ignores additional edges on the Aux input.

The counter stops counting upon receiving an active edge on the Gate input. The counter stores the count in the FIFO.

You can configure the rising or falling edge of the Aux input to be the active edge. You can configure the rising or falling edge of the Gate input to be the active edge.

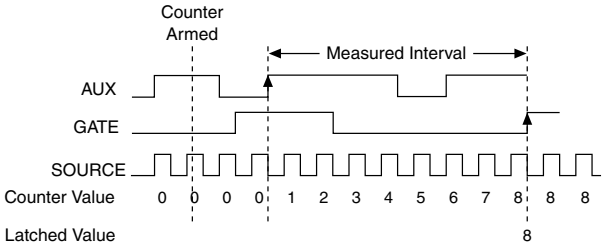
Use this type of measurement to count events or measure the time that occurs between edges on two signals. This type of measurement is sometimes referred to as start/stop trigger measurement, second gate measurement, or A-to-B measurement.

Single Two-Signal Edge-Separation Measurement

With single two-signal edge-separation measurement, the counter counts the number of rising (or falling) edges on the Source input occurring between an active edge of the Gate signal and an active edge of the Aux signal. The counter then stores the count in the FIFO and ignores other edges on its inputs. Software then reads the stored count.

The following figure shows an example of a single two-signal edge-separation measurement.

Figure 51. Single Two-Signal Edge-Separation Measurement



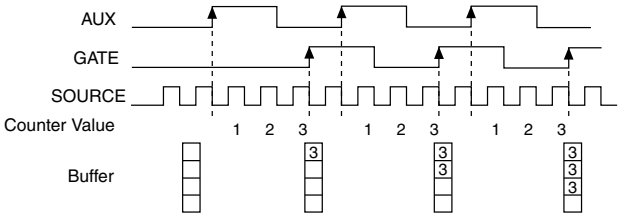
Implicit Buffered Two-Signal Edge-Separation Measurement

Implicit buffered and single two-signal edge-separation measurements are similar, but implicit buffered measurement measures multiple intervals.

The counter counts the number of rising (or falling) edges on the Source input occurring between an active edge of the Gate signal and an active edge of the Aux signal. The counter then stores the count in the FIFO. On the next active edge of the Gate signal, the counter begins another measurement. The sampled values will be transferred to host memory using a high-speed data stream.

The following figure shows an example of an implicit buffered two-signal edge-separation measurement.

Figure 52. Implicit Buffered Two-Signal Edge-Separation Measurement

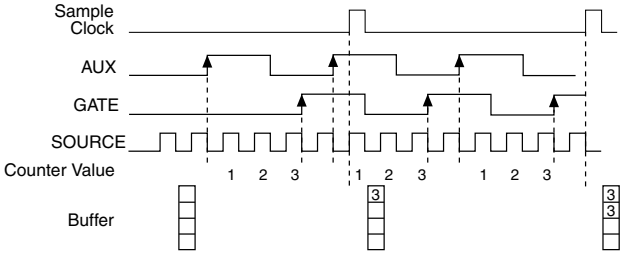


Sample Clocked Buffered Two-Signal Separation Measurement

A sample clocked buffered two-signal separation measurement is similar to single two-signal separation measurement, but buffered two-signal separation measurement takes measurements over multiple intervals correlated to a sample clock. The counter counts the number of rising (or falling) edges on the Source input occurring between an active edge of the Gate signal and an active edge of the Aux signal. The counter then stores the count in the FIFO on a sample clock edge. On the next active edge of the Gate signal, the counter begins another measurement. The sampled values will be transferred to host memory using a high-speed data stream.

The figure below shows an example of a sample clocked buffered two-signal separation measurement.

Figure 53. Sample Clocked Buffered Two-Signal Separation Measurement



Note If an active edge on the Gate and an active edge on the Aux does not occur between sample clocks, an overrun error occurs.

Counter Output Applications

Simple Pulse Generation

Single Pulse Generation

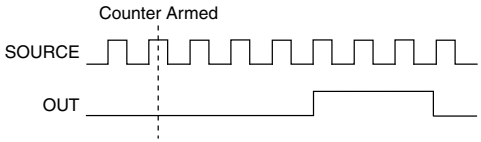
The counter can output a single pulse. The pulse appears on the Counter *n* Internal Output signal of the counter.

You can specify a delay from when the counter is armed to the beginning of the pulse. The delay is measured in terms of a number of active edges of the Source input.

You can specify a pulse width. The pulse width is also measured in terms of a number of active edges of the Source input. You also can specify the active edge of the Source input (rising or falling).

The following figure shows a generation of a pulse with a pulse delay of four and a pulse width of three (using the rising edge of Source).

Figure 54. Single Pulse Generation



Single Pulse Generation with Start Trigger

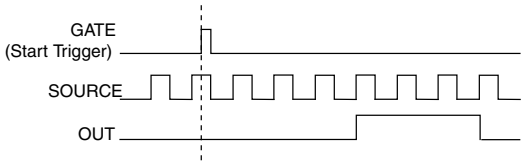
The counter can output a single pulse in response to one pulse on a hardware Start Trigger signal. The pulse appears on the Counter n Internal Output signal of the counter.

You can specify a delay from the Start Trigger to the beginning of the pulse. You also can specify the pulse width. The delay is measured in terms of a number of active edges of the Source input.

You can specify a pulse width. The pulse width is also measured in terms of a number of active edges of the Source input. You can also specify the active edge of the Source input (rising and falling).

The following figure shows a generation of a pulse with a pulse delay of four and a pulse width of three (using the rising edge of Source).

Figure 55. Single Pulse Generation with Start Trigger

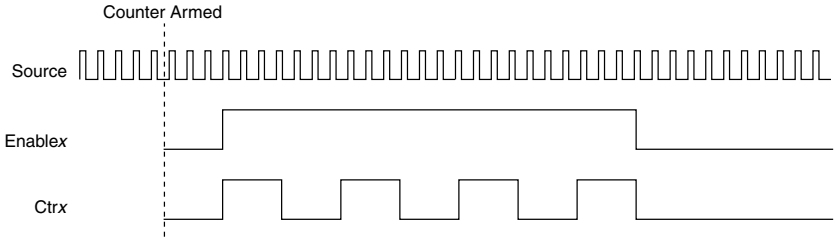


Pulse Train Generation

Finite Pulse Train Generation

This function generates a train of pulses with programmable frequency and duty cycle for a predetermined number of pulses. With sbRIO controller counters, the primary counter generates the specified pulse train and the embedded counter counts the pulses generated by the primary counter. When the embedded counter reaches the specified tick count, it generates a trigger that stops the primary counter generation.

Figure 56. Finite Pulse Train Generation: Four Ticks Initial Delay, Four Pulses



Retriggerable Pulse or Pulse Train Generation

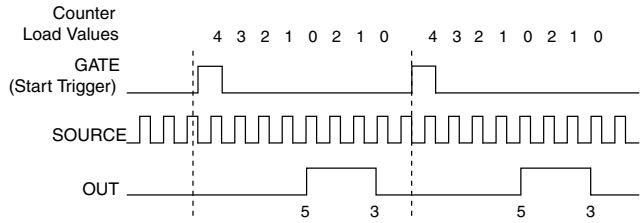
The counter can output a single pulse or multiple pulses in response to each pulse on a hardware Start Trigger signal. The generated pulses appear on the Counter *n* Internal Output signal of the counter.

You can route the Start Trigger signal to the Gate input of the counter. You can specify a delay from the Start Trigger to the beginning of each pulse. You also can specify the pulse width. The delay and pulse width are measured in terms of a number of active edges of the Source input. The initial delay can be applied to only the first trigger or to all triggers using the **CO.EnableInitalDelayOnRetrigger** property. The default for a single pulse is True, while the default for finite pulse trains is False.

The counter ignores the Gate input while a pulse generation is in progress. After the pulse generation is finished, the counter waits for another Start Trigger signal to begin another pulse generation. For retriggered pulse generation, pause triggers are not allowed since the pause trigger also uses the gate input.

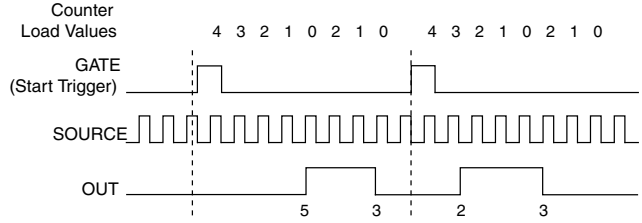
The figure below shows a generation of two pulses with a pulse delay of five and a pulse width of three (using the rising edge of Source) with **CO.EnableInitalDelayOnRetrigger** set to the default True.

Figure 57. Retriggerable Single Pulse Generation with Initial Delay on Retrigger



The figure below shows the same pulse train with **CO.EnableInitalDelayOnRetrigger** set to the default False.

Figure 58. Retriggerable Single Pulse Generation False



Note The minimum time between the trigger and the first active edge is two ticks of the source.

Continuous Pulse Train Generation

This function generates a train of pulses with programmable frequency and duty cycle. The pulses appear on the Counter *n* Internal Output signal of the counter.

You can specify a delay from when the counter is armed to the beginning of the pulse train. The delay is measured in terms of a number of active edges of the Source input.

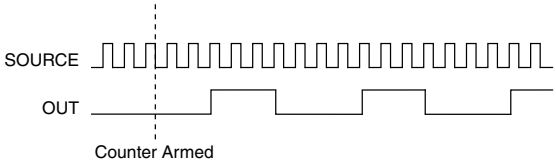
You specify the high and low pulse widths of the output signal. The pulse widths are also measured in terms of a number of active edges of the Source input. You also can specify the active edge of the Source input (rising or falling).

The counter can begin the pulse train generation as soon as the counter is armed, or in response to a hardware Start Trigger. You can route the Start Trigger to the Gate input of the counter.

You also can use the Gate input of the counter as a Pause Trigger (if it is not used as a Start Trigger). The counter pauses pulse generation when the Pause Trigger is active.

The figure below shows a continuous pulse train generation (using the rising edge of Source).

Figure 59. Continuous Pulse Train Generation



Continuous pulse train generation is sometimes called frequency division. If the high and low pulse widths of the output signal are *M* and *N* periods, then the frequency of the Counter *n* Internal Output signal is equal to the frequency of the Source input divided by *M* + *N*.

Buffered Pulse Train Generation

The sbRIO controller counters can use the FIFO to perform a buffered pulse train generation. This pulse train can use implicit timing or sample clock timing. When using implicit timing, the pulse idle time and active time changes with each sample you write. With sample clocked timing, each sample you write updates the idle time and active time of your generation on each sample clock edge. Idle time and active time can also be defined in terms of frequency and duty cycle or idle ticks and active ticks.

Note On buffered implicit pulse trains, the pulse specifications in the DAQmx Create Counter Output Channel are ignored so that you generate the number of pulses defined in the multipoint write. On buffered sample clock pulse trains, the pulse specifications in the DAQmx Create Counter Output Channel are generated after the counters starts and before the first sample clock so that you generate the number of updates defined in the multipoint write.

Finite Implicit Buffered Pulse Train Generation

This function generates a predetermined number of pulses with variable idle and active times. Each point you write generates a single pulse. The number of pairs of idle and active times

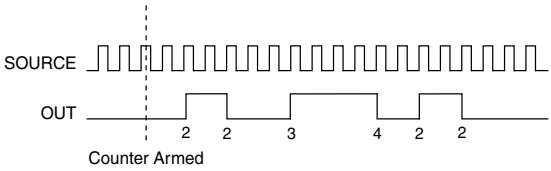
(pulse specifications) you write determines the number of pulses generated. All points are generated back to back to create a user defined pulse train.

The following table and figure detail a finite implicit generation of three samples.

Table 27. Finite Implicit Buffered Pulse Train Generation

Sample	Idle Ticks	Active Ticks
1	2	2
2	3	4
3	2	2

Figure 60. Finite Implicit Buffered Pulse Train Generation



Continuous Buffered Implicit Pulse Train Generation

This function generates a continuous train of pulses with variable idle and active times. Instead of generating a set number of data samples and stopping, a continuous generation continues until you stop the operation. Each point you write generates a single pulse. All points are generated back to back to create a user defined pulse train.

Finite Buffered Sample Clocked Pulse Train Generation

This function generates a predetermined number of pulse train updates. Each point you write defines pulse specifications that are updated with each sample clock. When a sample clock occurs, the current pulse (idle followed by active) finishes generation and the next pulse updates with the next sample specifications.

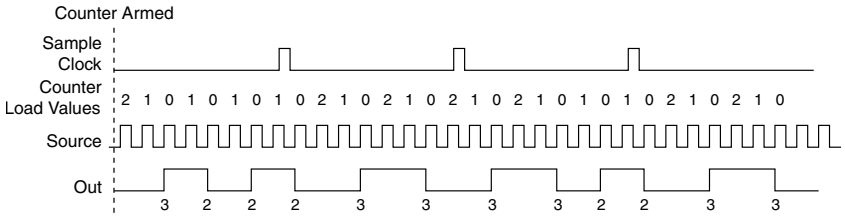
Note When the last sample is generated, the pulse train continues to generate with these specifications until the task is stopped.

The following table and figure detail a finite sample clocked generation of three samples where the pulse specifications from the create channel are two ticks idle, two ticks active, and three ticks initial delay.

Table 28. Finite Buffered Sample Clocked Pulse Train Generation

Sample	Idle Ticks	Active Ticks
1	3	3
2	2	2
3	3	3

Figure 61. Finite Buffered Sample Clocked Pulse Train Generation



There are several different methods of continuous generation that control what data is written. These methods are regeneration, FIFO regeneration, and non-regeneration modes.

Regeneration is the repetition of the data that is already in the buffer.

Standard regeneration is when data from the PC buffer is continually downloaded to the FIFO to be written out. New data can be written to the PC buffer at any time without disrupting the output. With FIFO regeneration, the entire buffer is downloaded to the FIFO and regenerated from there. Once the data is downloaded, new data cannot be written to the FIFO. To use FIFO regeneration, the entire buffer must fit within the FIFO size. The advantage of using FIFO regeneration is that it does not require communication with the main host memory once the operation is started, thereby preventing any problems that may occur due to excessive bus traffic.

With non-regeneration, old data is not repeated. New data must be continually written to the buffer. If the program does not write new data to the buffer at a fast enough rate to keep up with the generation, the buffer underflows and causes an error.

Continuous Buffered Sample Clocked Pulse Train Generation

This function generates a continuous train of pulses with variable idle and active times. Instead of generating a set number of data samples and stopping, a continuous generation continues until you stop the operation. Each point you write specifies pulse specifications that are updated with each sample clock. When a sample clock occurs, the current pulse finishes generation and the next pulse uses the next sample specifications.

Frequency Generation

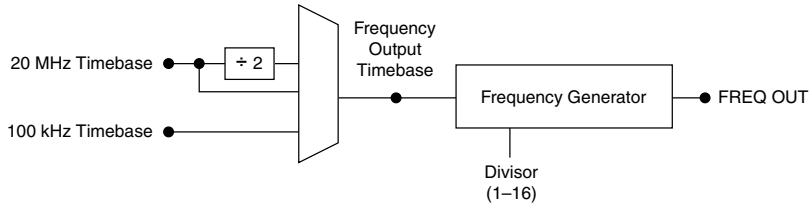
You can generate a frequency by using a counter in pulse train generation mode or by using the frequency generator circuit, as described in the *Using the Frequency Generator* section.

Using the Frequency Generator

The frequency generator can output a square wave at many different frequencies. The frequency generator is independent of the four general-purpose 32-bit counter/timer modules on the cRIO controller.

The following figure shows a block diagram of the frequency generator.

Figure 62. Frequency Generator Block Diagram

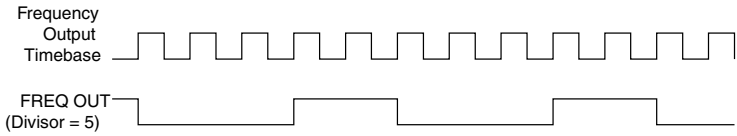


The frequency generator generates the Frequency Output signal. The Frequency Output signal is the Frequency Output Timebase divided by a number you select from 1 to 16. The Frequency Output Timebase can be either the 20 MHz Timebase, the 20 MHz Timebase divided by 2, or the 100 kHz Timebase.

The duty cycle of Frequency Output is 50% if the divider is either 1 or an even number. For an odd divider, suppose the divider is set to D. In this case, Frequency Output is low for (D + 1)/2 cycles and high for (D - 1)/2 cycles of the Frequency Output Timebase.

The following figure shows the output waveform of the frequency generator when the divider is set to 5.

Figure 63. Frequency Generator Output Waveform



Frequency Output can be routed out to any PFI terminal. All PFI terminals are set to high-impedance at startup. The FREQ OUT signal also can be routed to many internal timing signals.

In software, program the frequency generator as you would program one of the counters for pulse train generation.

Frequency Division

The counters can generate a signal with a frequency that is a fraction of an input signal. This function is equivalent to continuous pulse train generation.

Pulse Generation for ETS

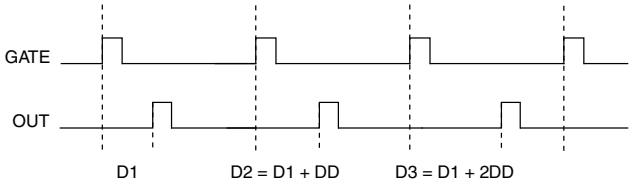
In the equivalent time sampling (ETS) application, the counter produces a pulse on the output a specified delay after an active edge on Gate. After each active edge on Gate, the counter cumulatively increments the delay between the Gate and the pulse on the output by a specified amount. Thus, the delay between the Gate and the pulse produced successively increases.

The increase in the delay value can be between 0 and 255. For instance, if you specify the increment to be 10, the delay between the active Gate edge and the pulse on the output increases by 10 every time a new pulse is generated.

Suppose you program your counter to generate pulses with a delay of 100 and pulse width of 200 each time it receives a trigger. Furthermore, suppose you specify the delay increment to be 10. On the first trigger, your pulse delay will be 100, on the second it will be 110, on the third it will be 120; the process will repeat in this manner until the counter is disarmed. The counter ignores any Gate edge that is received while the pulse triggered by the previous Gate edge is in progress.

The waveform thus produced at the counter's output can be used to provide timing for undersampling applications where a digitizing system can sample repetitive waveforms that are higher in frequency than the Nyquist frequency of the system. The following figure shows an example of pulse generation for ETS; the delay from the trigger to the pulse increases after each subsequent Gate active edge.

Figure 64. Pulse Generation for ETS



Counter Timing Signals

In this section, *n* refers to the sbRIO controller Counter 0, 1, 2, or 3. For example, Counter *n* Source refers to four signals—Counter 0 Source (the source input to Counter 0), Counter 1 Source (the source input to Counter 1), Counter 2 Source (the source input to Counter 2), or Counter 3 Source (the source input to Counter 3).

Note Note All counter timing signals can be filtered. Refer to the *PFI Filters* section for more information.

Counter *n* Source Signal

The selected edge of the Counter *n* Source signal increments and decrements the counter value depending on the application the counter is performing. The following table lists how this terminal is used in various applications.

Table 29. Counter Applications and Counter *n* Source

Application	Purpose of Source Terminal
Pulse Generation	Counter Timebase
One Counter Time Measurements	Counter Timebase
Two Counter Time Measurements	Input Terminal
Non-Buffered Edge Counting	Input Terminal
Buffered Edge Counting	Input Terminal
Two-Edge Separation	Counter Timebase

Routing a Signal to Counter *n* Source

Each counter has independent input selectors for the Counter *n* Source signal. Any of the following signals can be routed to the Counter n Source input:

- 80 MHz Timebase
- 20 MHz Timebase
- 13.1072 MHz Timebase
- 12.8 MHz Timebase
- 10 MHz Timebase
- 100 kHz Timebase
- Any PFI terminal
- Analog Comparison Event
- Change Detection Event

In addition, TC or Gate from a counter can be routed to a different counter source.

Some of these options may not be available in some driver software. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information about available routing options.

Routing Counter *n* Source to an Output Terminal

You can route Counter *n* Source out to any PFI terminal.

Counter *n* Gate Signal

The Counter *n* Gate signal can perform many different operations depending on the application including starting and stopping the counter, and saving the counter contents.

Routing a Signal to Counter *n* Gate

Each counter has independent input selectors for the Counter *n* Gate signal. Any of the following signals can be routed to the Counter *n* Gate input:

- Any PFI terminal
- AI Reference Trigger
- AI Start Trigger
- AO Sample Clock
- DI Sample Clock
- DI Reference Trigger
- DO Sample Clock
- Change Detection Event
- Analog Comparison Event

In addition, a counter's Internal Output or Source can be routed to a different counter's gate.

Some of these options may not be available in some driver software. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information about available routing options.

Routing Counter *n* Gate to an Output Terminal

You can route Counter *n* Gate out to any PFI terminal.

Counter *n* Aux Signal

The Counter *n* Aux signal indicates the first edge in a two-signal edge-separation measurement.

Routing a Signal to Counter *n* Aux

Each counter has independent input selectors for the Counter *n* Aux signal. Any of the following signals can be routed to the Counter *n* Aux input:

- Any PFI terminal
- AI Reference Trigger
- AI Start Trigger
- Analog Comparison Event
- Change Detection Event

In addition, a counter's Internal Output, Gate or Source can be routed to a different counter's Aux. A counter's own gate can also be routed to its Aux input.

Some of these options may not be available in some driver software. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information about available routing options.

Counter *n* A, Counter *n* B, and Counter *n* Z Signals

Counter *n* B can control the direction of counting in edge counting applications. Use the A, B, and Z inputs to each counter when measuring quadrature encoders or measuring two pulse encoders.

Routing Signals to A, B, and Z Counter Inputs

Each counter has independent input selectors for each of the A, B, and Z inputs. Any of the following signals can be routed to each input:

- Any PFI terminal
- Analog Comparison Event

Routing Counter *n* Z Signal to an Output Terminal

You can route Counter *n* Z out to any PFI terminal.

Counter *n* Up_Down Signal

Counter *n* Up_Down is another name for the Counter *n* B signal.

Counter *n* HW Arm Signal

The Counter *n* HW Arm signal enables a counter to begin an input or output function.

To begin any counter input or output function, you must first enable, or arm, the counter. In some applications, such as a buffered edge count, the counter begins counting when it is armed. In other applications, such as single pulse-width measurement, the counter begins waiting for the Gate signal when it is armed. Counter output operations can use the arm signal in addition to a start trigger.

Software can arm a counter or configure counters to be armed on a hardware signal. Software calls this hardware signal the Arm Start Trigger. Internally, software routes the Arm Start Trigger to the Counter *n* HW Arm input of the counter.

Routing Signals to Counter *n* HW Arm Input

Any of the following signals can be routed to the Counter n HW Arm input:

- Any PFI terminal
- AI Reference Trigger
- AI Start Trigger
- Analog Comparison Event
- Change Detection Event

A counter’s Internal Output can be routed to a different counter’s HW Arm.

Some of these options may not be available in some driver software. Refer to the "Device Routing in MAX" topic in the *NI-DAQmx Help* or the *LabVIEW Help* for more information about available routing options.

Counter *n* Sample Clock Signal

Use the Counter *n* Sample Clock (Ctr*n*SampleClock) signal to perform sample clocked acquisitions and generations.

You can specify an internal or external source for Counter *n* Sample Clock. You also can specify whether the measurement sample begins on the rising edge or falling edge of Counter *n* Sample Clock.

If the sbRIO controller receives a Counter *n* Sample Clock when the FIFO is full, it reports an overflow error to the host software.

Using an Internal Source

To use Counter *n* Sample Clock with an internal source, specify the signal source and the polarity of the signal. The source can be any of the following signals:

- DI Sample Clock
- DO Sample Clock
- AI Sample Clock
- AI Convert Clock
- AO Sample Clock
- DI Change Detection output

Several other internal signals can be routed to Counter *n* Sample Clock through internal routes. Refer to "Device Routing in MAX" in the *NI-DAQmx Help* or the *LabVIEW Help* for more information.

Using an External Source

You can route any of the following signals as Counter *n* Sample Clock:

- Any PFI terminal
- Analog Comparison Event

You can sample data on the rising or falling edge of Counter *n* Sample Clock.

Routing Counter *n* Sample Clock to an Output Terminal

You can route Counter *n* Sample Clock out to any PFI terminal. The PFI circuitry inverts the polarity of Counter *n* Sample Clock before driving the PFI terminal.

Counter *n* Internal Output and Counter *n* TC Signals

The Counter *n* Internal Output signal changes in response to Counter *n* TC.

The two software-selectable output options are pulse output on TC and toggle output on TC. The output polarity is software-selectable for both options.

With pulse or pulse train generation tasks, the counter drives the pulse(s) on the Counter *n* Internal Output signal. The Counter *n* Internal Output signal can be internally routed to be a counter/timer input or an “external” source for AI, AO, DI, or DO timing signals.

Routing Counter *n* Internal Output to an Output Terminal

You can route Counter *n* Internal Output to any PFI terminal.

Frequency Output Signal

The Frequency Output (FREQ OUT) signal is the output of the frequency output generator.

Routing Frequency Output to a Terminal

You can route Frequency Output to any PFI terminal.

Digital Routing

The digital routing circuitry of the sbRIO-96xx manages the flow of data between the bus interface and the acquisition and generation sub-systems when programming sbRIO I/O modules in Real-Time (NI-DAQmx) mode. The subsystems include analog input, analog output, digital I/O, and counters. The digital routing circuitry uses FIFOs (if present) in each sub-system to ensure efficient data movement.

Note When programming sbRIO I/O modules in FPGA mode, the flow of data between the modules and the bus interface is programmed using LabVIEW FPGA.

The digital routing circuitry also routes timing and control signals. The acquisition and generation sub-systems use these signals to manage and synchronize acquisitions and generations. These signals can come from the following sources:

- sbRIO I/O modules programmed in Real-Time (NI-DAQmx) mode
- FPGA or DAQ ASIC using the sbRIO trigger bus to share hardware triggers and signals between the LabVIEW FPGA and DAQmx applications

Synchronization Across a Network

Internal Timebase

The onboard 100 MHz oscillator automatically synchronizes to other network-synchronized devices that are part of the local IEEE 802.1AS or IEEE 1588-2008 subnet, depending on the active time reference that is being used on the controller.

The 80 MHz, 40 MHz, 20 MHz, 100 kHz, 13.1072 MHz, 12.8 MHz, and 10 MHz timebases are derived from the onboard oscillator and are synchronized to it. Therefore, the timebases are also synchronized to other network-synchronized timebases on the IEEE 802.1AS or IEEE 1588-2008 subnet. This enables analog input, analog output, digital I/O, and counter/timers to be synchronized to other chassis across a distributed network.

When programming sbRIO I/O modules in FPGA mode, the Time Synchronization IO Nodes can be used to synchronize the LabVIEW FPGA application to other network-synchronized devices.

Network-based Synchronization

IEEE 1588, also known as the precision time protocol (PTP), is an Ethernet-based synchronization method designed for cabled, local networks. The PTP protocol provides a fault tolerant method of synchronizing all participating clocks to the highest quality clock in the network. This method of synchronization between networked devices uses packet-based communication and is possible over the long distances allowed for each Ethernet link, without signal propagation impact. IEEE 1588 has many different profiles, such as IEEE 802.1AS-2011, each of which use different features. Because the profiles are not interoperable with each other, make sure it is known which profile is implemented on the device. For devices on the network to synchronize with each other using IEEE 1588, all devices must be compatible with the desired IEEE 1588 profile and must all be connected within the selected IEEE 1588 profile-compliant network infrastructure.

The sbRIO-96xx controllers are compatible with both the IEEE 802.1AS-2011 profile and the IEEE 1588-2008 (1588v2) Delay Request-Response profile. However, each network port must be configured individually to the specific profile required for the network.

Differences Between IEEE 802.1AS-2011 and IEEE 1588-2008

IEEE 802.1AS-2011, also known as the generalized precision time protocol (gPTP), is a profile of IEEE 1588. A sbRIO-96xx controller can be configured to use either the IEEE 802.1AS-2011 profile or the IEEE 1588-2008 profile by configuring the port's time reference. If a user does not explicitly specify which time reference to use a sbRIO-96xx controller will default to use the IEEE 802.1AS-2011 profile. There are some differences between the IEEE 802.1AS-2011 profile and the IEEE 1588-2008 profile which are called out below:

- IEEE 802.1AS-2011 assumes all communication between devices is done on the OSI layer 2, while IEEE 1588-2008 can support various layer 2 and layer 3-4 communication methods. The IEEE 1588-2008 profile National Instruments implements on the sbRIO-96xx only supports layer 3-4 communication methods. Operating on the layer 2 yields better performance for the IEEE 802.1AS-2011.
- IEEE 802.1AS-2011 only communicates gPTP information directly with other IEEE 802.1AS devices within a system. Therefore, there must be IEEE 802.1AS-2011 support along the entire path from one IEEE 802.1AS-2011 device to another. With IEEE 1588-2008, it is possible to use non-IEEE 1588-2008 switches between two IEEE 1588-2008 devices. The benefit of having IEEE 802.1AS-2011 support along the entire path is a faster performance and lower jitter compared to IEEE 1588-2008.
- With IEEE 802.1AS-2011 there are only two types of time-aware systems: time-aware end stations and time-aware bridges. Whereas with IEEE 1588-2008, there are the following: ordinary clock, boundary clock, end-to-end transparent clock and a time-aware bridges. Based on these factors, IEEE 802.1AS-2011 can reduce complexity and configuration challenges compared to IEEE 1588-2008. A sbRIO-96xx controller acts as a time-aware end station for both protocols.

IEEE 1588 External Switch Requirements

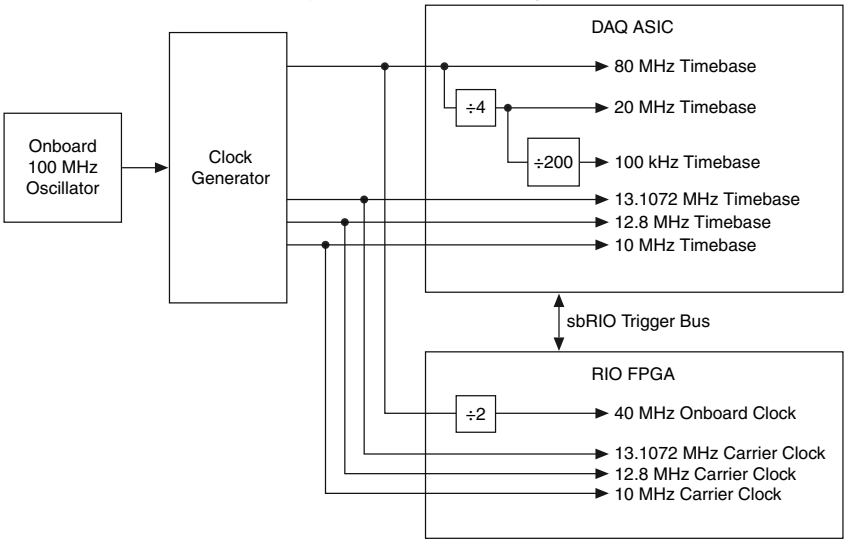
To take advantage of the network synchronization features of the sbRIO-96xx controllers, ensure that your network infrastructure meets certain requirements depending on which IEEE 1588 profile is implemented for your application:

- IEEE 802.1AS-2011 support—Automatically enables timebase synchronization and enables the use of time-based triggers and timestamping between devices across the network. Synchronization performance will meet NI product specifications.
- IEEE 1588-2008 support—Enables timebase synchronization and enables the use of time-based triggers and timestamping between devices across the network. Synchronization performance will vary and may not meet NI product specifications. As a default configuration for IEEE 1588-2008, NI supports the IEEE 1588 Delay Request-Response profile, using the UDP over IP transport (layer 3-4).

Clock Routing

Onboard Clock Routing

Figure 65. Clock Routing Circuitry



Note When switching between programming modes, you may notice the terms timebase and clock used interchangeably. This is due to the DAQ ASIC and the RIO FPGA using different terminology for timing and clock mechanisms. The documentation will use the term based on the programming mode discussed.

80 MHz Timebase

When programming sbRIO I/O modules in Real-Time (NI-DAQmx) mode, the 80 MHz timebase can function as the source input to the 32-bit general-purpose counter/timers. The 80 MHz timebase is generated from the onboard oscillator.

20 MHz and 100 kHz Timebases

When programming sbRIO I/O modules in Real-Time (NI-DAQmx) mode, the 20 MHz and 100 kHz timebases can be used to generate many of the analog input and analog output timing signals. These timebases can also function as the source input to the 32-bit general-purpose counter/timers. The 20 MHz and 100 kHz timebases are generated by dividing down the 80 MHz timebase, as shown in the previous figure.

40 MHz Onboard Clock

When programming sbRIO I/O modules in FPGA mode, the 40 MHz onboard clock is used as the top-level clock for your LabVIEW FPGA application and sbRIO I/O module IO nodes. The 40 MHz onboard clock can be used to clock single-cycle timed loops. Derived clocks of varying frequency can be generated from the 40 MHz onboard clock. The 40 MHz onboard clock is phase aligned with the incoming 80 MHz clock.

13.1072 MHz, 12.8 MHz, and 10 MHz Timebases and Carrier Clocks

When programming sbRIO I/O modules in Real-Time (NI-DAQmx) mode, the 13.1072 MHz, 12.8 MHz, and 10 MHz timebases can be used to generate many of the analog input and analog output timing signals. These timebases can also function as the source input to the 32-bit general-purpose counter/timers. The 13.1072 MHz, 12.8 MHz, and 10 MHz timebases are generated directly from the onboard clock generator.

When programming sbRIO I/O modules in FPGA mode, the 13.1072 MHz, 12.8 MHz, and 10 MHz carrier clocks can be used as the master clock for sbRIO I/O analog input and analog output modules. The 13.1072 MHz, 12.8 MHz, and 10 MHz carrier clocks are available as IO Nodes in LabVIEW FPGA applications and can be used to correlate the onboard clocks with self-timed sbRIO I/O modules containing free-running clocks.

BIOS Configuration

Resetting the System CMOS and BIOS Settings

The sbRIO-96xx BIOS configuration information is stored in a nonvolatile memory location that does not require a battery to preserve the settings. Additionally, the BIOS optimizes boot time by saving specific system information to memory backed up by a battery (CMOS).

Complete the following steps to reset the CMOS and reset the BIOS settings to factory default values.

1. Disconnect power from the sbRIO-96xx.

2. Remove the CMOS battery and wait for longer than 1 second.
3. Replace the CMOS battery.
4. Reconnect power to the sbRIO-96xx.

The `BIOS Reset Detected` warning message appears onscreen.

Power-On Self Test Warning Messages

The sbRIO-96xx POST displays warning messages for specific issues onscreen. You can use MAX to enable Console Out to send these warning messages through the RS-232 serial port.

The POST can display the following warning messages:

- BIOS Reset Detected—This warning is displayed when the CMOS Reset button has been pushed. This warning indicates that the BIOS settings have the default values.
- CMOS Battery Is Dead—This warning is displayed when the CMOS battery is dead and must be replaced. The BIOS settings are preserved even when the CMOS battery is dead, but the system will boot very slowly because the BIOS cannot optimize boot time by saving specific system information to CMOS.

BIOS Setup Utility

Use the BIOS setup utility to change configuration settings and to enable special functions. The sbRIO-96xx ships with configuration settings that work for most applications, but you can use the BIOS setup utility to change configuration settings to meet the needs of your application.

Changing BIOS settings can cause incorrect behavior, including failure to boot. In general, do not change a setting unless you are sure what the setting does. Reset the BIOS settings to restoring the default configuration settings.

Launching the BIOS Setup Utility for BIOS with DisplayPort

Complete the following steps to launch the BIOS setup utility.

1. Connect a video monitor to the USB 3.1 Type-C DisplayPort connector on the sbRIO-96xx.

   Note In order to connect the monitor to the USB 3.1 Type-C DisplayPort connector, you may need to use a VGA-to-Type-C or DVI-to-Type-C adapter.

2. Connect a USB keyboard to the USB 2.0 host port on the sbRIO-96xx.
3. Power on or reboot the sbRIO-96xx.
4. Hold down either the <F10> key or the <Del> key until `Please select boot device:` appears onscreen.
5. Use the Down Arrow key to select **Enter Setup** and press <Enter>. The setup utility loads after a short delay.

The Main setup menu is displayed when you first enter the BIOS setup utility.

Launching the BIOS Setup Utility for BIOS with Console Out

Complete the following steps to launch the BIOS setup utility.

1. Connect a video monitor to the USB 3.1 Type-C DisplayPort connector on the sbRIO-96xx.

   Note In order to connect the monitor to the USB 3.1 Type-C DisplayPort connector, you may need to use a VGA-to-Type-C or DVI-to-Type-C adapter.

2. Connect a USB keyboard to the USB 2.0 host port on the sbRIO-96xx.
3. Power on or reboot the sbRIO-96xx.
4. Hold down either the <F10> key or the <Del> key until `Please select boot device:` appears onscreen.
5. Use the Down Arrow key to select **Enter Setup** and press <Enter>. The setup utility loads after a short delay.

The Main setup menu is displayed when you first enter the BIOS setup utility.

BIOS Setup Utility Keyboard Navigation

Use the following keys to navigate through the BIOS setup utility:

Table 30. Navigation Keys

Key(s)	Function(s)
Left Arrow, Right Arrow	Move between the different setup menus. If you are in a submenu, these keys have no effect, and you must press <Esc> to leave the submenu first.
Up Arrow, Down Arrow	Move between the options within a setup menu.
<Enter>	Enter a submenu or display all available settings for a highlighted configuration option.
<Esc>	Return to the parent menu of a submenu. At the top-level menus, this key serves as a shortcut to the Exit menu.
<+>, <->	Cycle between all available settings for a selected configuration option.
<F8>	Load the previous values for all BIOS configuration settings.
<F9>	Load the optimal default values for all BIOS configuration settings. The optimal default values are the same as the shipping configuration default values.
<F10>	Save settings and exits the BIOS setup utility.

Main Setup Menu

The Main setup menu includes the following settings:

- BIOS Version— This value indicates the version of the controller BIOS.
- Build Date—This value indicates the date and time on which the BIOS was built.

- Embedded Firmware Version—This value identifies the built-in hardware capabilities.
- Processor Type—This value indicates the type of processor used in the controller.
- Base Processor Frequency—This value indicates the the speed of the processor.
- Active Processor Cores—This value indicates the number of active processor cores.
- Total Memory—This value indicates the size of system RAM detected by the BIOS.
- System Date—This setting controls the date, which is stored in a battery-backed real-time clock. Most operating systems also include a way to change this setting. Use <+> and <-> in conjunction with <Enter> and <Tab> to change these values.
- System Time—This setting controls the time of day, which is stored in a battery-backed real-time clock. Most operating systems also include a way to change this setting. Use <+> and <-> in conjunction with <Enter> and <Tab> to change these values.
- Access Level— This value indicates the level of access the current user has on the controller BIOS.

Advanced Setup Menu

The Advanced setup menu contains BIOS settings that do not require modification for normal operation of the sbRIO-96xx. If you have specific problems, such as unbootable disks or resource conflicts, you may need to examine the settings in this menu.

Notice Changing settings in the Advanced setup menu may result in an unstable or unbootable controller. If this happens, restore BIOS settings to the factory defaults.

Power/Wake Configuration Submenu

The Power/Wake configuration submenu contains the power and wake settings for the chipset and sbRIO-96xx. The factory default settings provide the most compatible and optimal configuration.

- Restore after Power Loss—This setting specifies what state to go to when power is re-applied after a power failure.
- Ring Indicator Wake—This setting enables or disables the ability to wake a powered-off system using the Ring Indicator pin of the RS-232 serial port. The default value is **Disabled**.

PCI Subsystem Configuration Submenu

The PCI Subsystem Configuration submenu contains the PCI and PCI Express settings for the chipset and sbRIO-96xx. The factory default settings provide the most compatible and optimal configuration.

PCI Settings

- 64-Bit Memory-Mapped—This setting allows some PCI memory-mapped I/O to be above the 32-bit boundary. This can be useful when using a 64-bit OS and a large number of PCI devices.
- PCI System Error Report—This setting enables the reporting of PCI system errors. For more information on PCI system errors visit *ni.com/r/T1ASPC*.

PCI Express Settings

- PCI Express Port 3—This setting allows you to enable or disable the PCI Express Port 3 in the chipset.
- Speed—This setting allows you to select **Auto**, **Gen 2**, or **Gen 1** as the PCIe port speed.

SATA Drives Submenu

The SATA drives submenu contains the hard disk drive (HDD) interfaces settings. The factory default settings provide the most compatible and optimal configuration.

- SATA Controller(s)—This setting enables or disables the chipset SATA controller.
- SATA Speed Support—This setting allows you to select **Gen1** or **Gen2** for SATA speed support.
- SATA Mode—This setting allows you to select **IDE Mode** or **AHCI** Mode.
- Serial-ATA Port 0—This setting enables or disables the SATA Port 0.
- Onboard Storage—This item displays the onboard drive detected in the system.

SDIO Configuration Submenu

The SDIO configuration submenu contains the SD storage settings for the chipset and sbRIO-96xx. The factory default settings provide the most compatible and optimal configuration.

SDIO Access Mode—This setting allows you to set the access mode for your SD device.

- Auto—Access the SD device in DMA mode if the controller supports it. Otherwise, this will access the SD device in PIO mode.
- ADMA—Access the SD device in ADMA mode.
- SDMA—Access the SD device in SDMA mode.
- PIO—Access the SD device in PIO mode.

USB Configuration Submenu

The USB configuration submenu contains the USB host ports settings. The factory default settings provide the most compatible and optimal configuration.

- Display Port Type-C FW Ver— This items shows the firmware version for the DisplayPort USB type-C port.
- USB Devices—This item lists the total number of devices detected in the system, categorized by device type.
- Legacy USB Support—This setting specifies whether legacy USB support is enabled. Legacy USB support refers to the ability to use a USB keyboard and mouse during system boot or in a legacy operating system such as DOS. Valid options are **Enabled**, **Disabled**, and **Auto**. The default value is **Disabled**.
- Overcurrent Reporting—This setting allows the BIOS to notify the operating system about any USB ports that source too much current. The default value is **Enabled**. Hardware overcurrent protection is always active and cannot be disabled.

- USB Transfer Timeout—This setting specifies the timeout value for Control, Bulk, and Interrupt USB transfers. The default value is **20** seconds.
- Device Reset Timeout—This setting specifies the number of seconds the POST waits for a USB mass storage device to start. The default value is **20** seconds.
- Device Power-Up Delay—This setting specifies the maximum time a device takes before enumerating. Valid options are **Auto** and **Manual**. The default value is **Auto**. When set to **Auto**, a root port is granted 100 ms, and the delay value of a hub port is assigned from the hub descriptor.

Security Setup Menu

The Security menu allows you set the Administrator and User passwords. Passwords must be between three and twenty characters in length.

- Administrator Password—This allows you to set the Administrator password. If you only set the Administrator password, the password only limits access when entering Setup.
- User Password—This allows you to set the User password. If you only set the User password, the password will have Administrator rights and you can use this password to boot or enter Setup.

Boot Setup Menu

The Boot Setup menu contains setting related to the boot process and the boot device priority.

- Boot Settings Configuration Submenu—This item accesses the Boot Settings Configuration submenu.
- Boot Option Priorities—These settings specify the order in which the BIOS checks for bootable devices, including the local hard disk drive, removable devices such as USB flash disk drives or USB CD-ROM drives, or the PXE network boot agent. The BIOS first attempts to boot from the device associated with 1st Boot Device, followed by 2nd Boot Device and 3rd Boot Device. If multiple boot devices are not present, the BIOS setup utility does not display all of these configuration options. To select a boot device, press <Enter> on the desired configuration option and select a boot device from the resulting menu. You can also disable certain boot devices by selecting **Disabled**.

Note Only one device of a given type is shown in this list. If more than one device of the same type exists, use the appropriate device BBS priorities submenu to re-order the priority of devices of the same type.

Boot Settings Configuration Submenu

The Boot Settings Configuration submenu applies alternate configurations to boot settings. The factory default settings provide the most compatible and optimal configuration.

- Setup Prompt Timeout—This setting specifies the number of seconds the system waits for a BIOS Setup menu keypress (the <Delete> key). The default value is **10** seconds.
- Bootup NumLock State—This setting specifies the power-on state of the keyboard NumLock setting. The default value is **On**.

Save & Exit Menu

The Save & Exit setup menu contains all available options for exiting, saving, and loading the BIOS default configuration. As an alternative to this menu, press <F9> to load optimal BIOS default settings and <F10> to save changes and exit setup.

The Save & Exit setup menu includes the following settings:

- Save Changes and Reset—This setup utility then exits and reboots the sbRIO-96xx. Any changes made to BIOS settings are stored in NVRAM. The <F10> key also selects this option.
- Discard Changes and Reset—Any changes made to BIOS settings during this session of the BIOS setup utility are discarded. The setup utility then exits and reboots the controller. The <Esc> key can also be used to select this option.
- Save Changes—Changes made to BIOS settings during this session are committed to NVRAM. The setup utility remains active, allowing further changes.
- Discard Changes—Any changes made to BIOS settings during this session of the BIOS setup utility are discarded. The BIOS setup continues to be active.
- Restore Defaults—This option restores all BIOS settings to the factory default. This option is useful if the controller exhibits unpredictable behavior due to an incorrect or inappropriate BIOS setting. Notice that any nondefault settings such as boot order, passwords, and so on are also restored to their factory defaults. The <F9> key also selects this option.
- Save As User Defaults—This option saves a copy of the current BIOS settings as the User Defaults. This option is useful for preserving custom BIOS setup configurations.
- Restore User Defaults—This option restores all BIOS settings to the user defaults. This option is useful for restoring previously preserved custom BIOS setup configurations.
- Boot Override—This option lists all possible bootable devices and allows the user to override the Boot Option Priorities list for the current boot. If no changes have been made to the BIOS setup options, the system will continue booting to the selected device without first rebooting. If BIOS setup options have been changed and saved, a reboot will be required and the boot override selection will not be valid.

File System

LabVIEW mounts USB devices to the directory and creates symbolic links `/u`, `/v`, `/w`, or `/x` to the media mount point, starting with `/u` if it is available. To prevent any file corruption to external storage devices, verify that any file I/O operations with the specific drive finish before removing the device. Refer to the *LabVIEW Help* for more information.

The file system of the sbRIO-96xx follows conventions established for UNIX-style operating systems. Other LabVIEW Real-Time targets follow Microsoft Windows-style conventions. In order to facilitate the porting of applications from those targets, this target supports the

Windows-style `/C` home directory. This path is bound to the UNIX-style directory `/home/lvuser`.

Various LabVIEW Real-Time system files which would be accessible from `C:` (or `/C`) on other LabVIEW Real-Time targets are found in different locations on this target.

UNIX-style file systems support the concept of a symbolic link, which allows access to a file using an alternative file path. For example, it is possible to link `/C/ni-rt/system`, where dynamic libraries are deployed on other LabVIEW Real-Time targets, to `/usr/local/lib`, where they are stored on the sbRIO-96xx, if the application requires this.

For more information, visit *ni.com/r/RT_Paths*.

Planned Software Support

The following signals are not supported in NI CompactRIO 19.5. Refer to the following table for future release support information.

Interface	Software Release
RS-232 through the RMC connector	NI CompactRIO 19.6
RS-485 through the RMC connector	NI CompactRIO 19.6
CAN and CAN FD through the dedicated CAN port	TBD

Worldwide Support and Services

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

Visit *ni.com/services* for information about the services NI offers.

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

NI corporate headquarters is located at 11500 North Mopac Expressway, Austin, Texas, 78759-3504, USA. For up-to-date contact information for your location, visit *ni.com/contact*.

377905B-01 May 1, 2020