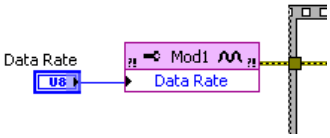
cRIO Waveform Acquisition Reference Examples:
Customizing the FPGA VIs

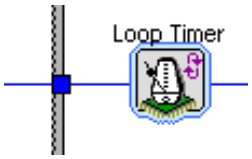
Main FPGA VIs:

Timing the acquisition loop for Delta Sigma hardware modules is very different from timing the acquisition loop for non Delta Sigma modules. This document will discuss key concepts implemented in the FPGA VI and will give a walkthrough on how to expand the code for multiple modules.

Key Concpets:

1. Timing:
   - **[FPGA] DeltaSig Main.vi:** Loop timing with Delta Sigma modules is determined by this Data Rate node. Execution of the acquisition loop will wait at the FPGA IO Node until new samples are ready.
   - **[FPGA] NonDeltaSig Main.vi:** Data Rates with SAR or non Delta Sigma modules is determined by using the FPGA Loop Timer. Execution of the acquisition loop will wait at the Loop Timer until the user specified sampling period has elapsed.





2. Host / FPGA Execution Synchronization:
   - The Real Time host application waits on this interrupt to synchronize itself with the start of the FPGA's acquisition. This synchronization prevents the host application from polling the DMA FIFO before the FPGA is sending data. Interrupt synchronization also prevents the FPGA from sending its data before the host application is ready to receive it.

3. Acquisition Loop Architecture:

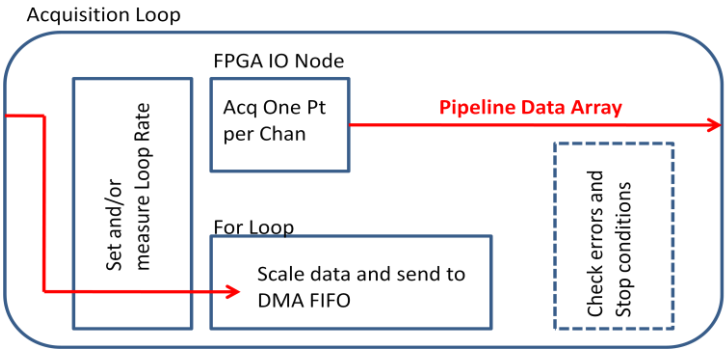


Figure 1: Block Diagram of FPGA Acquisition Loop (Both Delta Sigma and Non Delta Sigma)

- First, the **Loop Rate** set or measured based on the module hardware architecture (see notes above).
- The **FPGA IO Node** returns one point of calibrated data per channel and represents the data using the fixed-point datatype.
- The data is converted to floating point and **pipelined** via a shift register to give the FPGA IO node maximum time to perform its data acquisition. The FPGA IO Nodes have a minimum execution runtime that can take up 75% or more of the sampling period.
- A **For Loop** scales each data point by a user defined factor and sends the data to its DMA FIFO.

- Each iteration of the acquisition loop also checks for FIFO overflow errors (meaning DMA FIFO is full and data has been dropped), module underflow errors (FPGA code takes too long to execute so data is not being returned at specified rate), correct channel count, and the user defined stop conditions.

Expanding the FPGA VI to Support More Channels:

1. First add the new NI C Series module to your LabVIEW project.
    a. Right click the **FPGA Target** and select **New > C Series Modules…**
    b. Chose **Existing target or device** and select "Discover" if given this prompt:

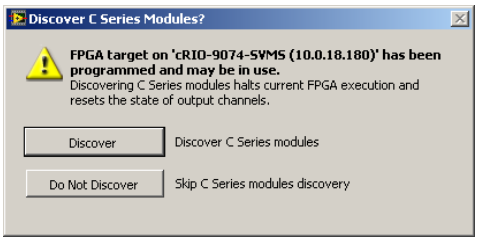


Figure 2: LabVIEW Project Prompt to Discover New C Series Modules

    c. Select the modules you want to add to the project and hit "OK."
    d. <u>If the modules are Delta Sigma based</u> and you want them to be synchronized, you will need to right click the module in the project and configure the **Master Timebase Source**. The *master device* should export its timebase and subsequently the slave devices should refer to that timebase. For example if Mod1 is the master then Mod1 should use **<Onboard Clock>** for its timebase and then enable the radial box named **Export Onboard Clock** (see Figure 3). The slave devices can now select **Mod1** for its **Master Timbebase Source** (see Figure 4). *Note: all of the modules' supported sample rates will now be equal to the master device's supported rates.*

Figure 3: Module Configuration [Timebase Settings] for Master Device Named Mod1

Figure 4: Module Configuration [Timebase Settings] for Slave Device(s)

2. Next expand the **FPGA IO Node** and **Build Array** primitive to add new channels to the pipelined data array.

Before: After:

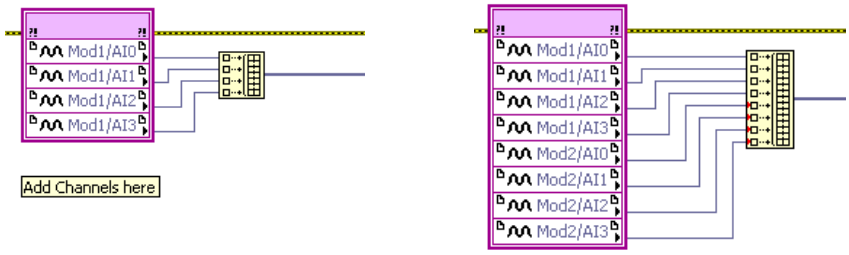


Figure 5: Multiple Synchronized Delta Sigma Based Modules

Note: Delta Sigma based modules and non Delta Sigma based modules should not share the same FPGA IO Node. If you are using a hybrid configuration then you should start with the *Delta Sigma FPGA VI* and add a second FPGA IO node below the original for the non delta sigma channels (see Figure 6 below).

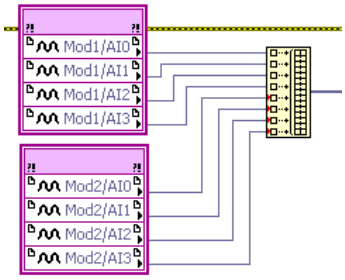


Figure 6: FPGA IO Nodes for Hybrid Applications (Delta Sigma and Non Delta Sigma Based Modules)

3. For *Delta Sigma* based architectures, you must expand the Mod1/Start *and* Mod1/Stop **FPGA I/O Nodes** to programmatically start and stop the modules. Furthermore you must add a Data Rate node for each new Delta Sigma module in the acquisition.

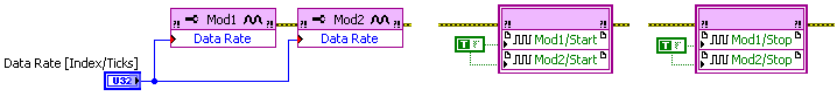


Figure 7: Each Delta Sigma Module Must Have Its Own Data Rate Node and Start/Stop Node