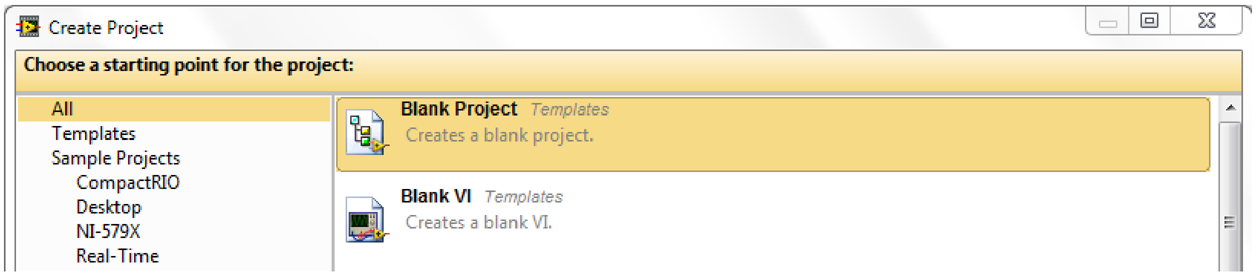
LabVIEW Evaluation Series: Acquire and Analyze Signals with LabVIEW Real-Time

Learn how to add a real-time target to your project and then create an application to deterministically acquire data and analyze it with ready-to-run libraries built into LabVIEW.

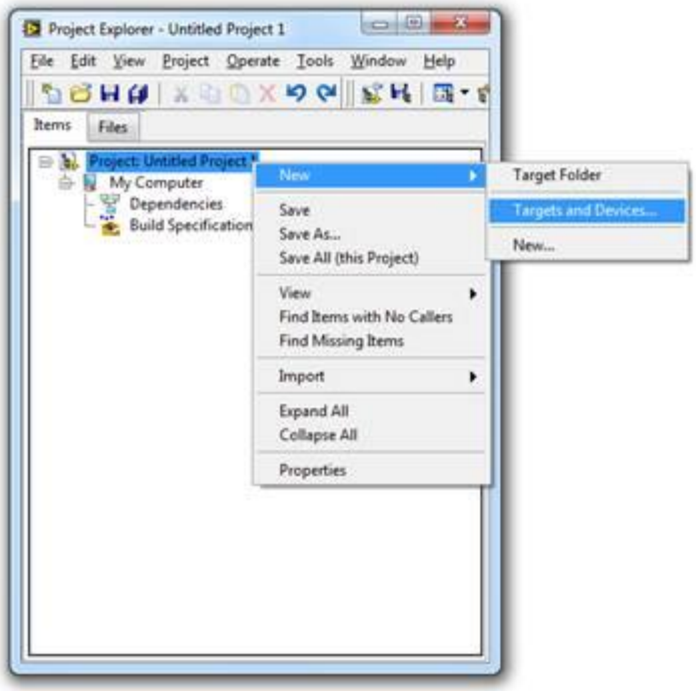
Starting a New Project in LabVIEW and Adding a Real-Time Target

Begin by creating a new project in LabVIEW, where you will manage your code and hardware resources.

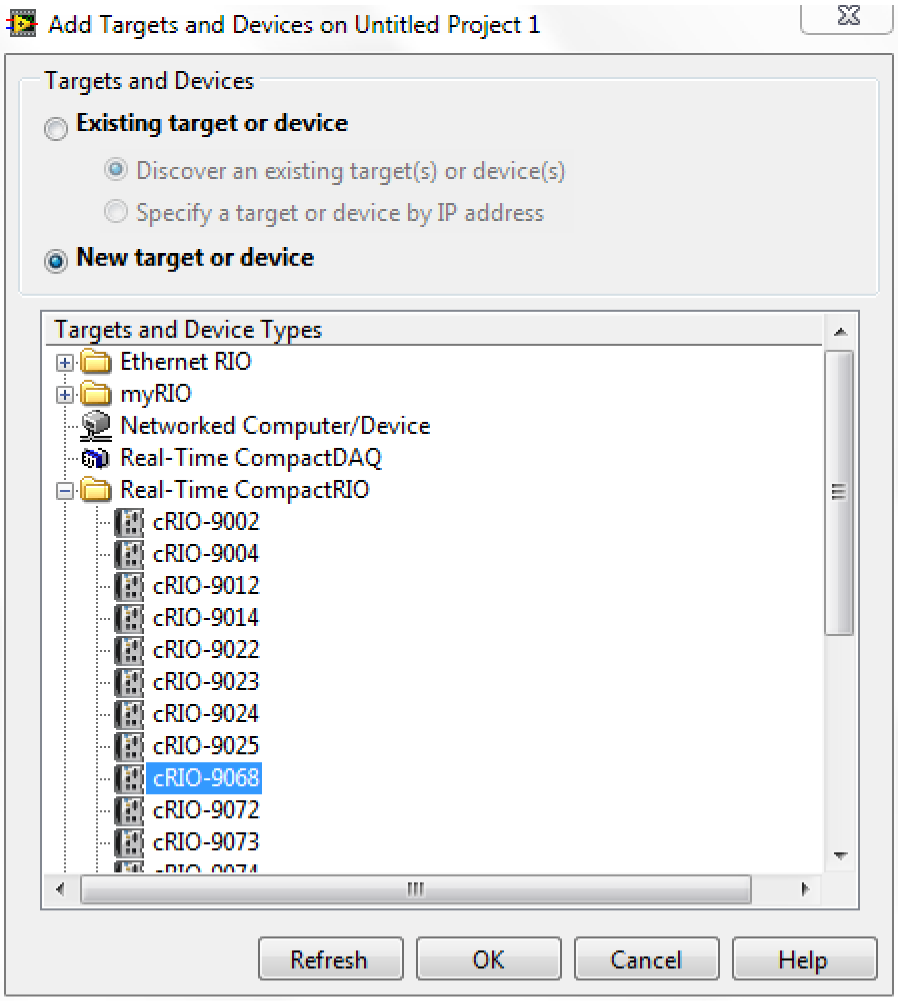
1. Create a new project in LabVIEW by selecting **File » Create Project**. Then select **Blank Project.**



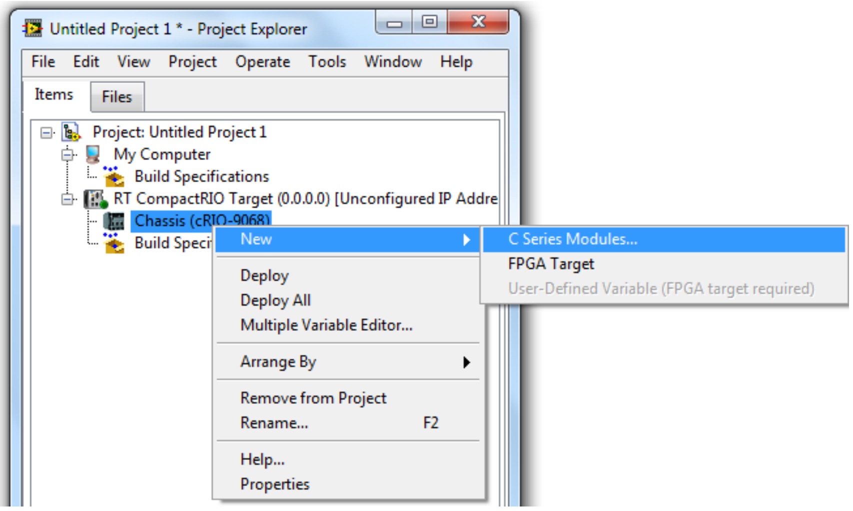
2. In the default project tree you have **My Computer**, which is where you can write code that will run on the Windows machine you're currently developing on. Remember that a Real-Time target has a processor running a real-time operating system, so effectively, it's another computer. To write code that will run on that computer, you need to add another target to your project. To add a real-time system to the project, right-click on the Project item at the top of the tree and select **New » Targets and Devices…**



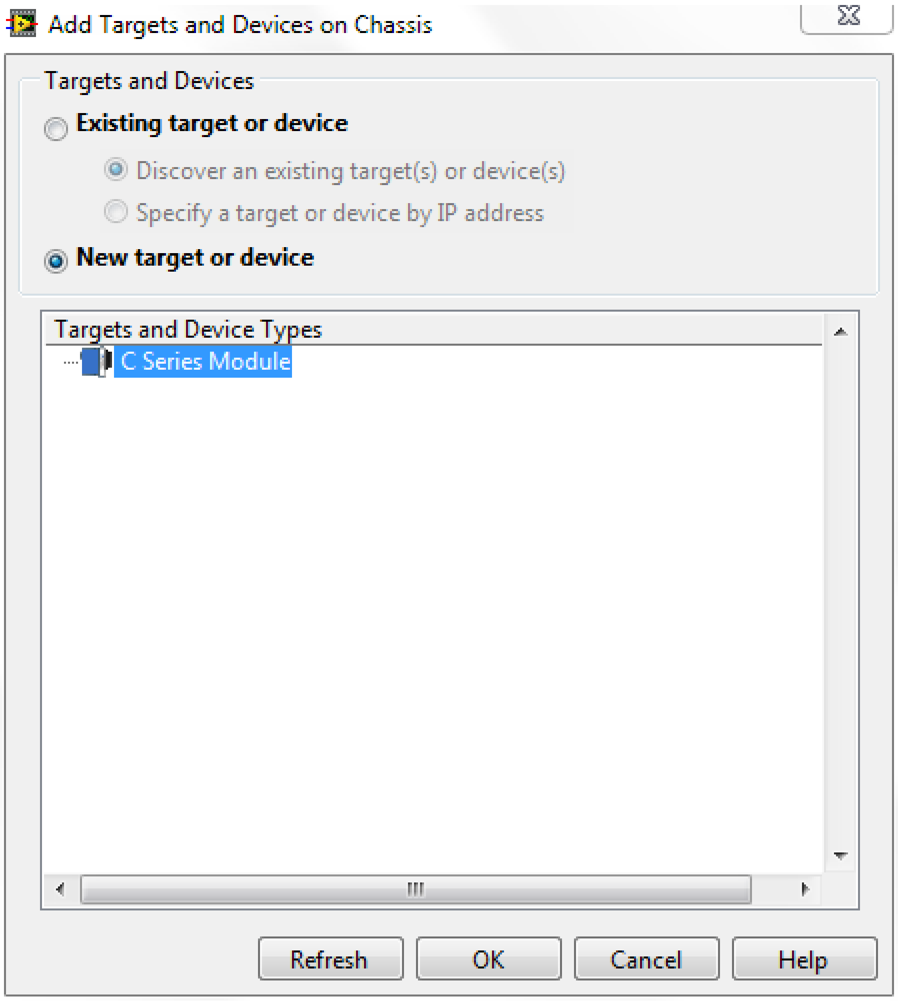
3. This dialog allows you to discover existing systems on your network or add a new system. Since you are just evaluating the software, select **New target or device.** LabVIEW lists available hardware corresponding with the drivers you have installed. Since you installed the NI-RIO driver with your evaluation software, select and expand **Real-Time CompactRIO**, then select **cRIO-9068**.



4. If you had an existing system, LabVIEW would now attempt to detect the I/O present in your system and automatically add them to the LabVIEW Project. Since we are adding a new system, we'll need to manually add a C Series I/O module. To do this, right-click on the Chassis project item, named for the backplane of a CompactRIO, and select **New » C Series Modules…**


Untitled Project 1 * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Untitled Project 1
My Computer
Build Specifications
RT CompactRIO Target (0.0.0.0) [Unconfigured IP Addre
Chassis (cRIO-9068)
Build Speci
New
Deploy
Deploy All
Multiple Variable Editor...
Arrange By
Remove from Project
Rename... F2
Help...
Properties
C Series Modules...
FPGA Target
User-Defined Variable (FPGA target required)

5. In the same manner as the target, since you do not have physical hardware, select **New target or device**, select **C Series Module**, then click **OK.**


Add Targets and Devices on Chassis

Targets and Devices

[Existing target or device]

[Discover an existing target(s) or device(s)]

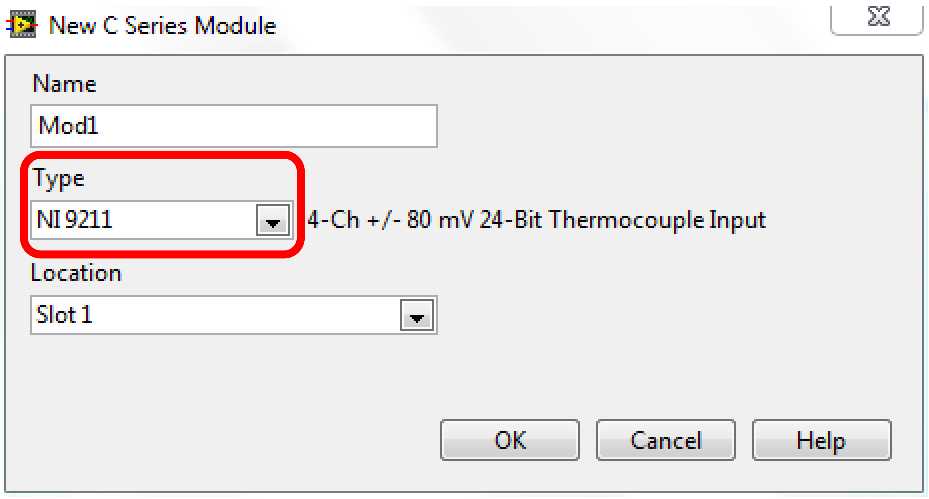
[Specify a target or device by IP address]

[✓ New target or device]

Targets and Device Types

C Series Module

6. In the dialog box that appears, select an NI 9211 Thermocouple module for Mod 1 then click **OK.**

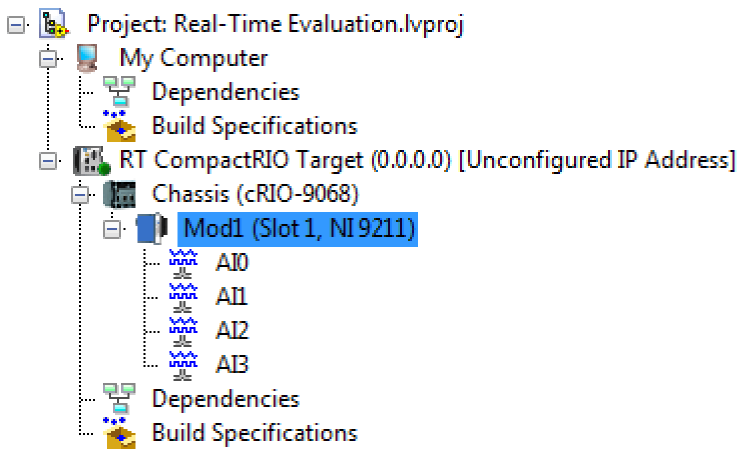

New C Series Module

Name: Mod1

Type: NI 9211 — 4-Ch +/- 80 mV 24-Bit Thermocouple Input

Location: Slot 1

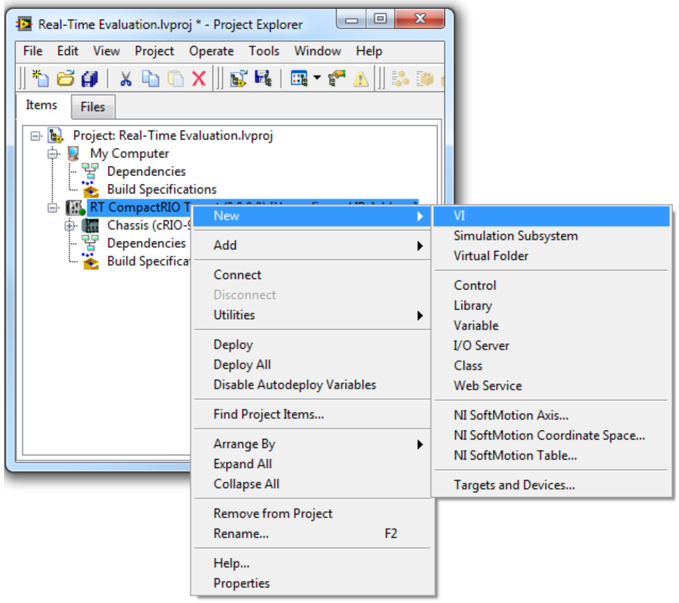
7. You are now ready to start programming and accessing I/O on your real-time target. Navigate to **File » Save** to save your project as **Real-Time Evaluation.lvproj.** Click **OK.** Note that you would communicate to your target by entering its IP Address once it's properly configured on the same network as your development computer.



Developing a LabVIEW Real-Time VI

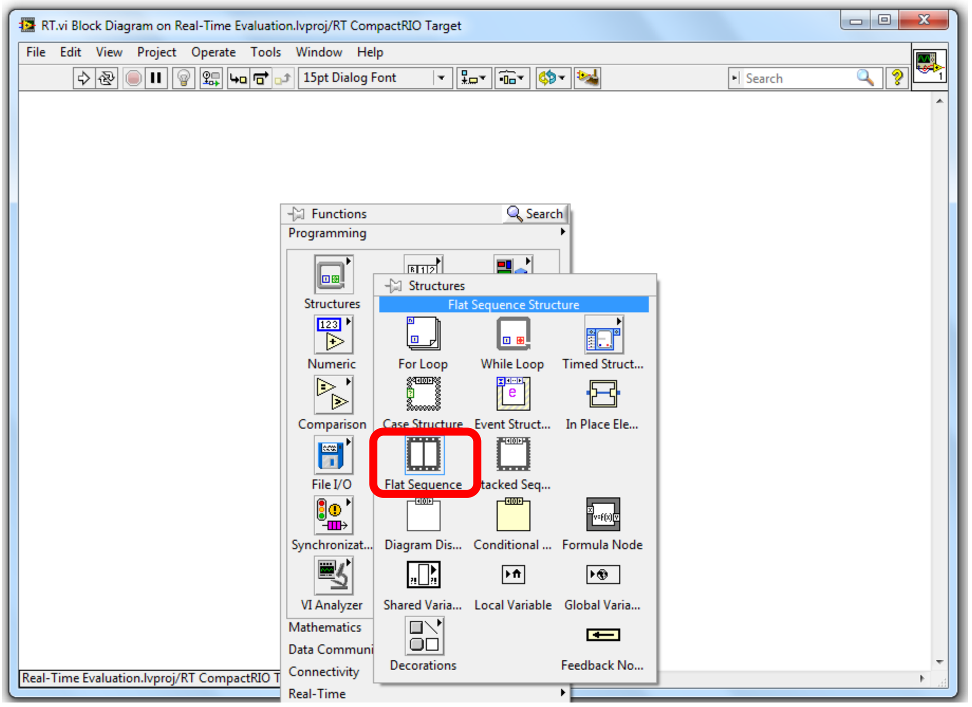
This section will walk you through creating a basic real-time control application on CompactRIO. You should now have a new LabVIEW Project that contains your CompactRIO system and C Series I/O module. In this tutorial we will be using an NI 9211 Thermocouple input module; however, the process can be followed for almost any analog input module.

1. This project will only contain one VI, which is the LabVIEW Real-Time application that runs embedded on the real-time processor. Create this the VI by right-clicking on the CompactRIO real-time controller in the project and selecting **New » VI.** Save the VI as *RT.vi*.

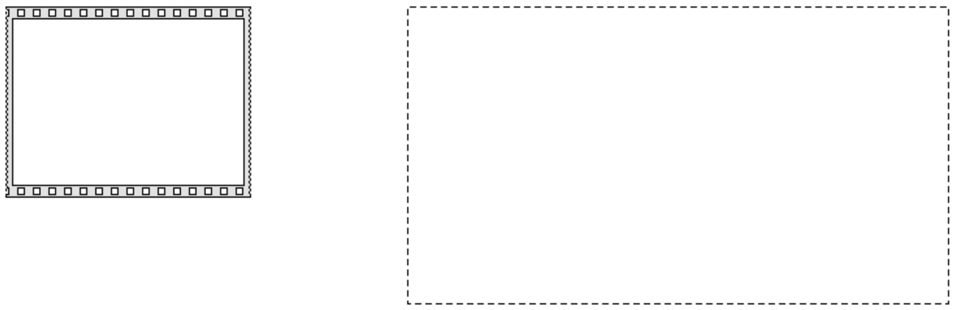


2. Double-click on **RT.vi** to open it. A VI is composed of two windows, the gray **Front Panel**, which serves as the user interface, and the white **Block Diagram**, where code is developed.

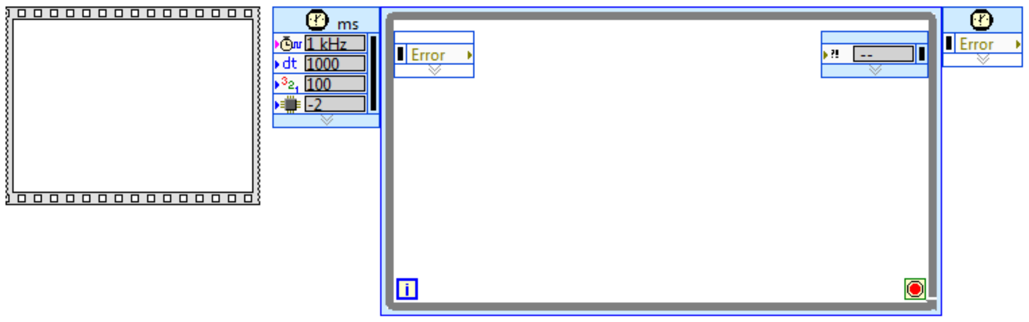
3. The basic operation of this application will include initialization and then two parallel tasks. A flat sequence structure is an easy way to enforce that initialization happens first in dataflow. In the white Block Diagram window, right-click in the blank space to bring up the Functions palette and then select **Programming » Structures » Flat Sequence Structure.**



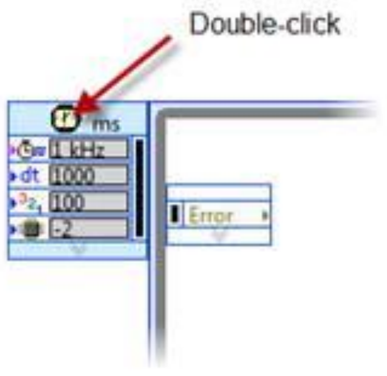
4. Now add a **Timed Loop** (right-click and select **Programming » Structures » Timed Structures » Timed Loop**) to the right of the sequence structure. Timed loops provide the ability to deterministically synchronize the code inside of them to various time bases, running continuously until a stop condition is met.



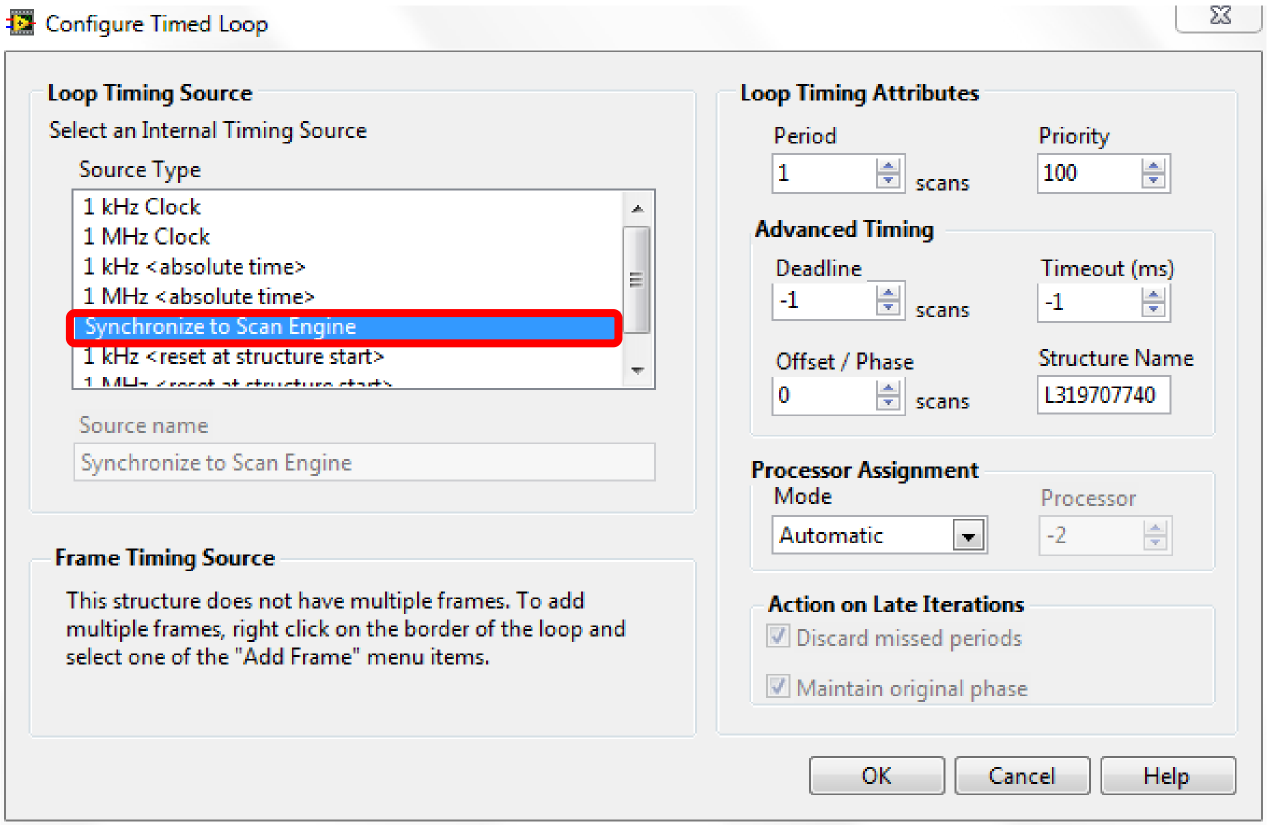
Drag with your mouse the rectangle where you want the loop to appear, then release.



5. To configure the timed loop, double-click on the clock icon on the left input node.



6. Select **Synchronize to Scan Engine** as the Loop Timing Source. Click **OK**. This will cause the code in the timed loop to execute once, immediately after each I/O scan, ensuring that any I/O values used in this timed loop are the most recent values. The NI Scan Engine allows you to access I/O modules directly in LabVIEW Real-Time on a CompactRIO without programming LabVIEW FPGA.



Configure Timed Loop

Loop Timing Source

Select an Internal Timing Source

Source Type

- 1 kHz Clock
- 1 MHz Clock
- 1 kHz <absolute time>
- 1 MHz <absolute time>
- Synchronize to Scan Engine
- 1 kHz <reset at structure start>
- 1 MHz <reset at structure start>

Source name: Synchronize to Scan Engine

Frame Timing Source

This structure does not have multiple frames. To add multiple frames, right click on the border of the loop and select one of the "Add Frame" menu items.

Loop Timing Attributes

Period: 1 scans

Priority: 100

Advanced Timing

Deadline: -1 scans

Timeout (ms): -1

Offset / Phase: 0 scans

Structure Name: L319707740

Processor Assignment

Mode: Automatic

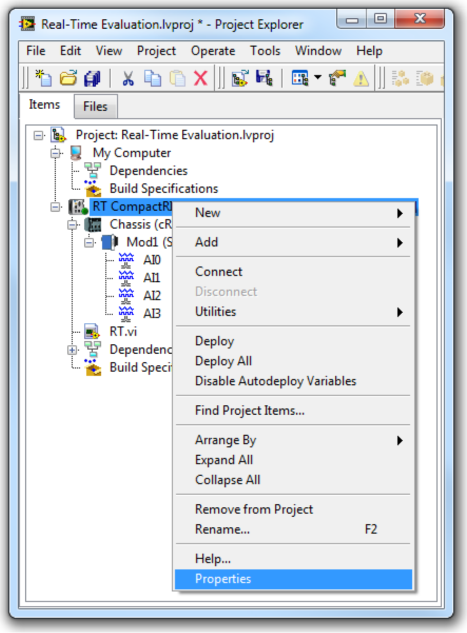
Processor: -2

Action on Late Iterations

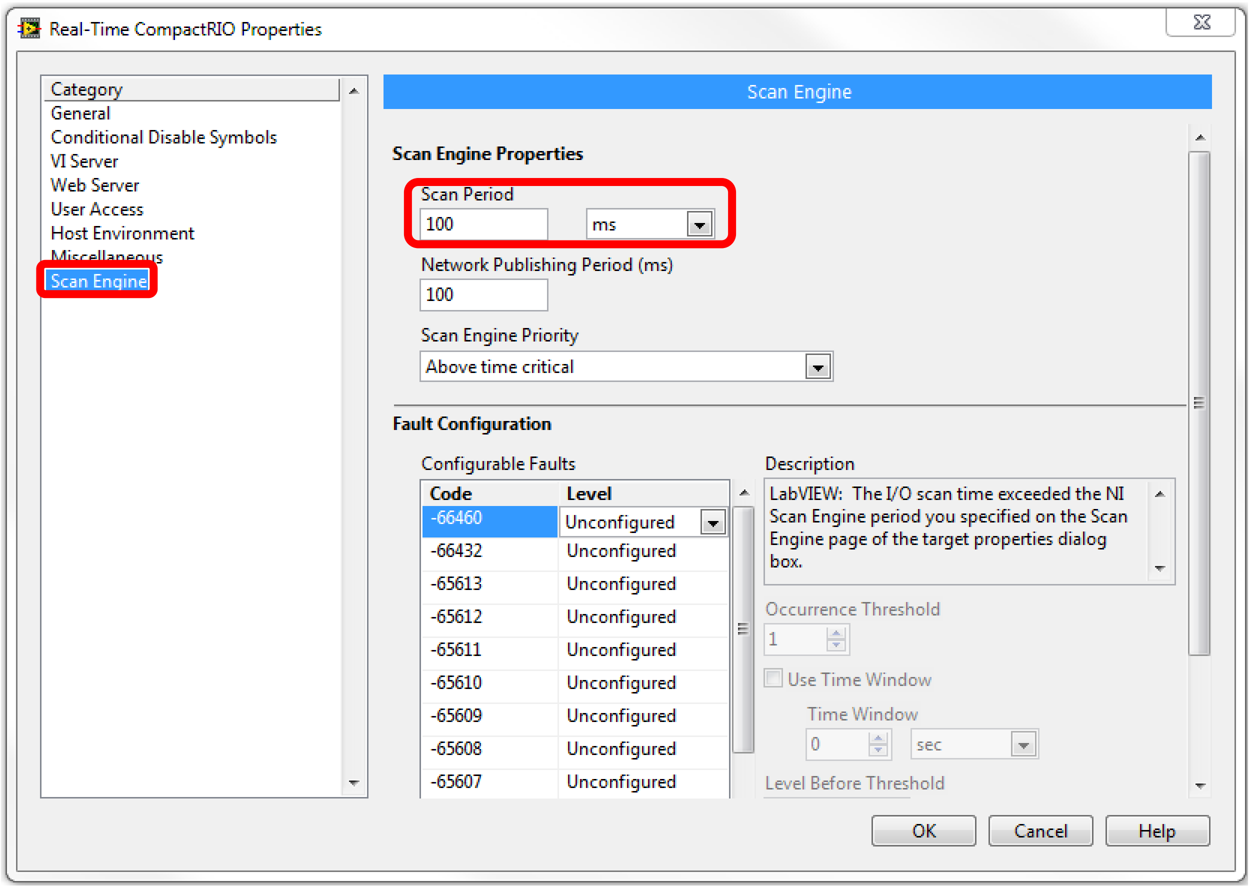
☑ Discard missed periods

☑ Maintain original phase

7. The previous step configured the timed loop to run synchronized to the scan engine. Now configure the rate of the scan engine itself by right-clicking on the top-level *RT CompactRIOTarget* in the LabVIEW Project and selecting **Properties.**



8. Select **Scan Engine** from the categories on the left and enter **100ms** as the *Scan Period.* This will cause all of the I/O in the CompactRIO system to be updated every 100ms (10Hz). The Network Publishing Period can also be set from this page, which controls how often the I/O values are published to the network for remote monitoring and debugging. Click **OK**.


Real-Time CompactRIO Properties

Category
General
Conditional Disable Symbols
VI Server
Web Server
User Access
Host Environment
Miscellaneous
Scan Engine

Scan Engine

Scan Engine Properties

Scan Period: 100 ms

Network Publishing Period (ms): 100

Scan Engine Priority: Above time critical

Fault Configuration

Configurable Faults

Code	Level
-66460	Unconfigured
-66432	Unconfigured
-65613	Unconfigured
-65612	Unconfigured
-65611	Unconfigured
-65610	Unconfigured
-65609	Unconfigured
-65608	Unconfigured
-65607	Unconfigured

Description: LabVIEW: The I/O scan time exceeded the NI Scan Engine period you specified on the Scan Engine page of the target properties dialog box.

Occurrence Threshold: 1

Use Time Window

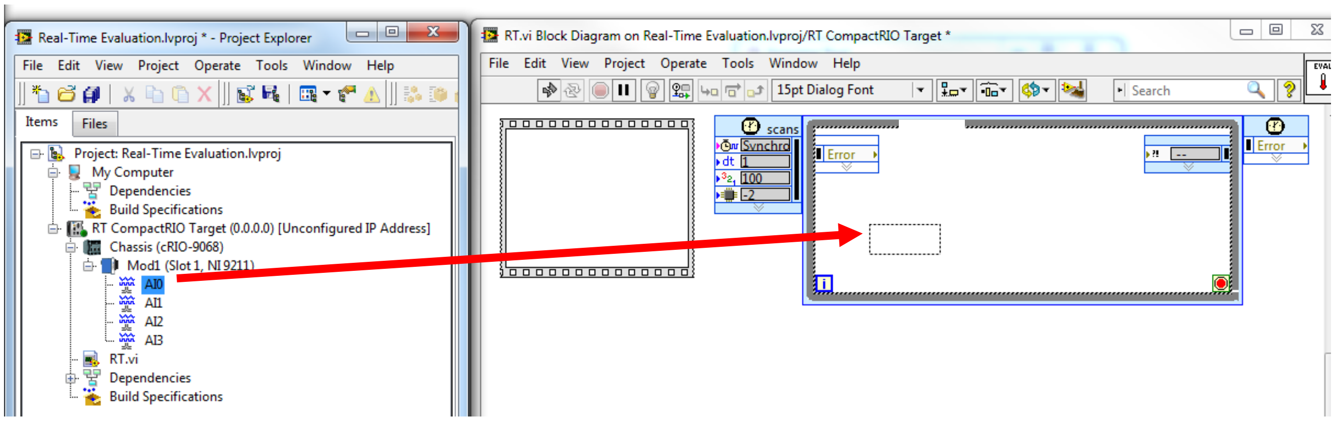
Time Window: 0 sec

Level Before Threshold

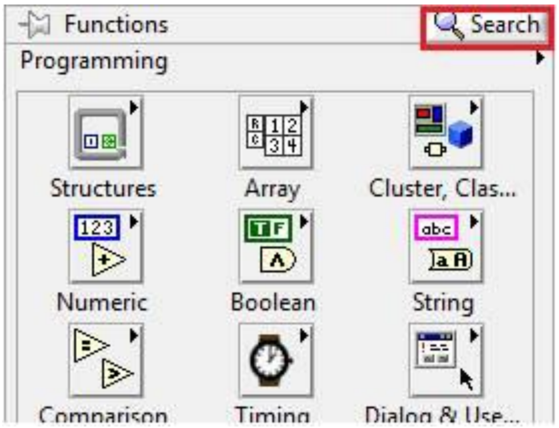
OK Cancel Help

9. Now that you have configured the I/O scan rate, it is time to add the I/O reads to your application for control. When using CompactRIO Scan Mode, you can simply drag and drop the I/O variables from the LabVIEW Project to the RT block diagram. Expand the CompactRIO chassis and I/O module. Select **AI0**,

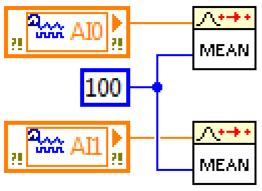
click on it, then drag and drop it into the timed loop on your RT.vi diagram as shown below. Also drag **AI1** into the timed loop.


Real-Time Evaluation.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Real-Time Evaluation.lvproj
My Computer
Dependencies
Build Specifications
RT CompactRIO Target (0.0.0.0) [Unconfigured IP Address]
Chassis (cRIO-9068)
Mod1 (Slot 1, NI 9211)
AI0
AI1
AI2
AI3
RT.vi
Dependencies
Build Specifications
RT.vi Block Diagram on Real-Time Evaluation.lvproj/RT CompactRIO Target *
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
scans
Synchro
dt 1
100
-2
Error
Error

10. Next we will add analysis that runs on the target to the program. To do so right click the block diagram to open the Functions palette and click on the Search button in the top right of the palette.

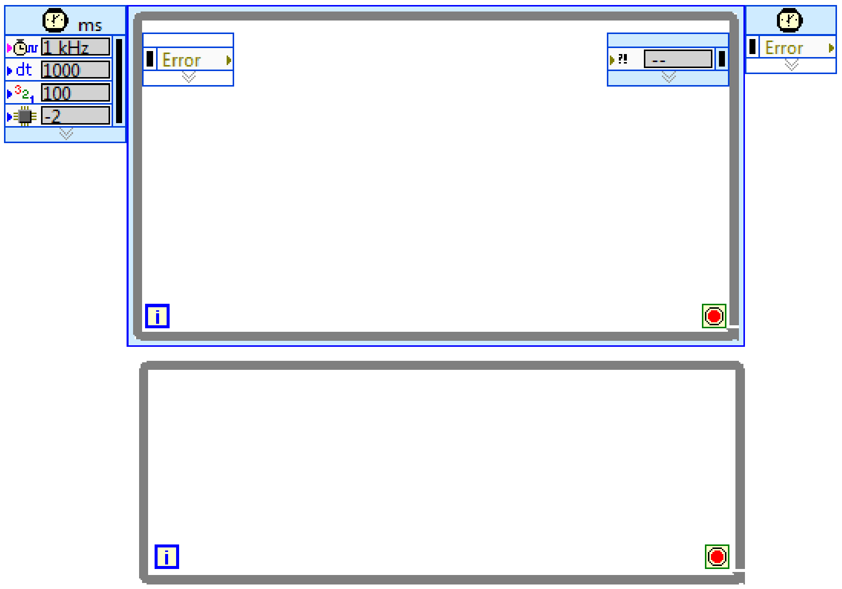


11. Search for “Mean” and select **Mean PtByPt.vi** in the Signal Processing Palette and drag two instances to your block diagram in the timed loop and wire your I/O as shown in the image below. Create a constant by right-clicking on the **sample length** input terminal and type in a value of **100**.

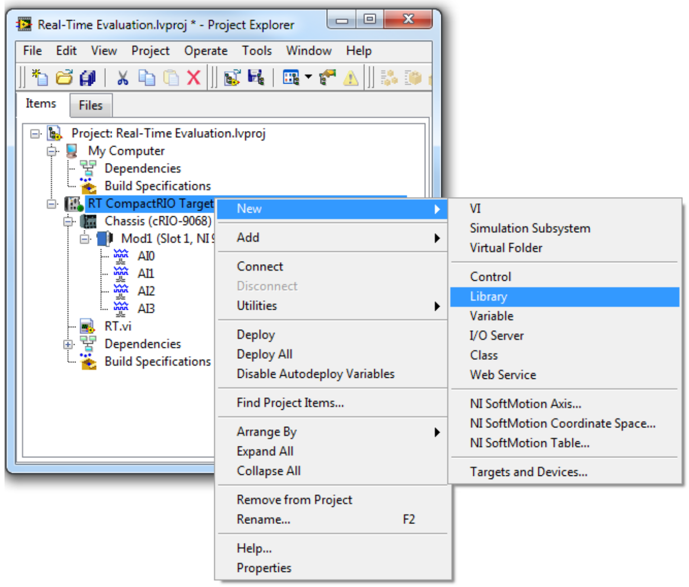


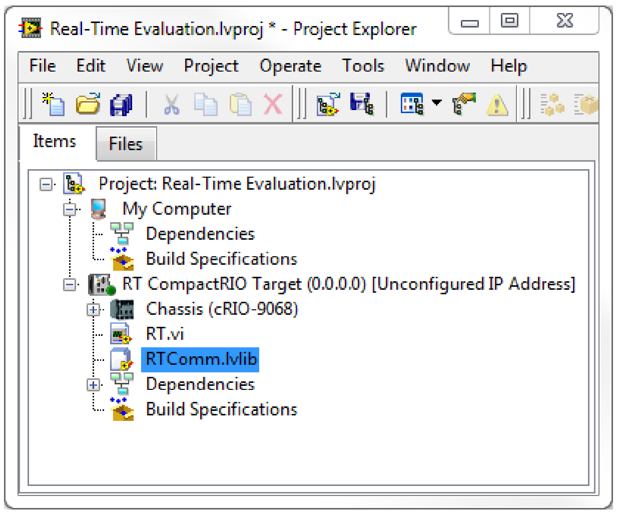
12. The next logical step would be to write the data to the user interface; however, it’s a best practice to separate high priority acquisition from presenting or logging the data. This minimizes jitter and allows your timed loop to run with “real-time” performance, meaning that it will always finish on time. By running tasks in separate loops at distinct rates, you also maximize CPU efficiency by executing each task only as often as necessary.

    You will transfer the latest result of the mean calculation from the high priority timed loop to a while loop using a real-time FIFO. This will provide a buffer between the two loops. The timed loop will run, synchronized to the I/O scan, and write the latest result of the mean calculation to the buffer each time. The while loop will read the data out of the buffer and write it to the user interface to display temperature data at run time. Place down a **While Loop** beneath the timed loop (**Structures » While Loop**).

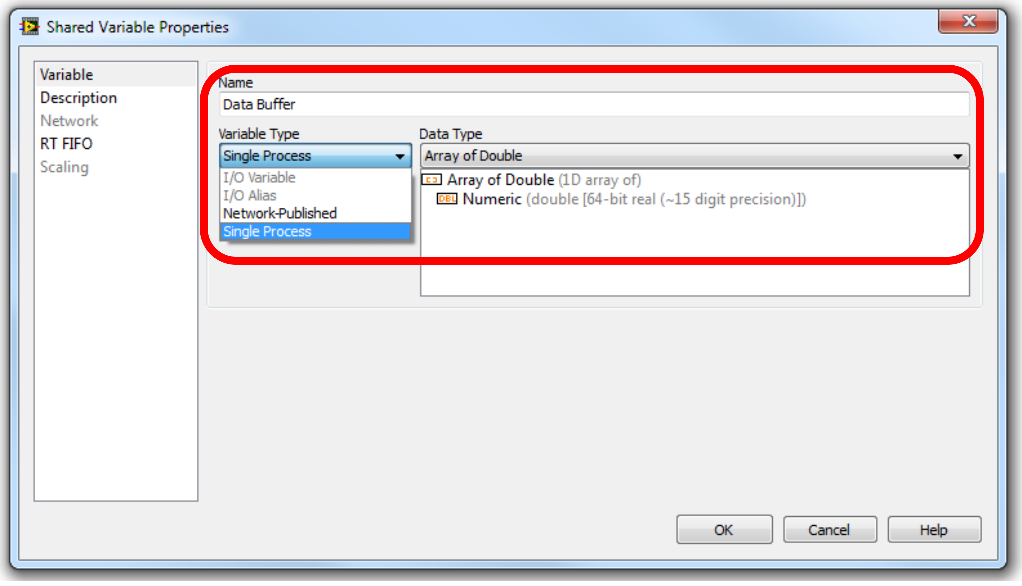


13. Next, create the shared variables to transfer values between the high priority timed loop and low priority while loop. To create a shared variable right click your RT CompactRIO Target in the LabVIEW Project and select **New » Library**. Right-click on the new library and select **Save» Save.** Rename your library something intuitive like “RTComm” to be saved on disk and click **OK**.



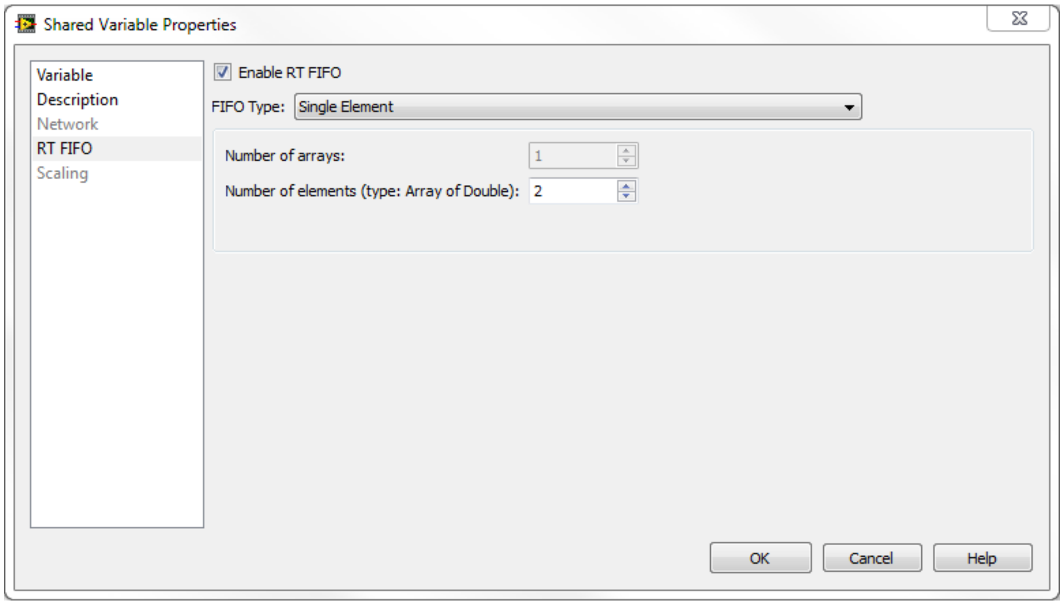

Real-Time Evaluation.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Real-Time Evaluation.lvproj
My Computer
Dependencies
Build Specifications
RT CompactRIO Target (0.0.0.0) [Unconfigured IP Address]
Chassis (cRIO-9068)
RT.vi
RTComm.lvlib
Dependencies
Build Specifications

14. Now, right-click your new library and select **New» Variable**. This will open the Shared Variable Properties window. Name your variable **Data Buffer**, and select **Single Process** for the variable type in the Variable Type drop down box as shown in the image below. Choose **Array of Double** as the Data Type. This will create a locally scoped variable (no network publishing) that contains an array of double precision floating point numbers.



Name: Data Buffer
Variable Type: Single Process
Data Type: Array of Double
Array of Double (1D array of)
Numeric (double [64-bit real (~15 digit precision)])

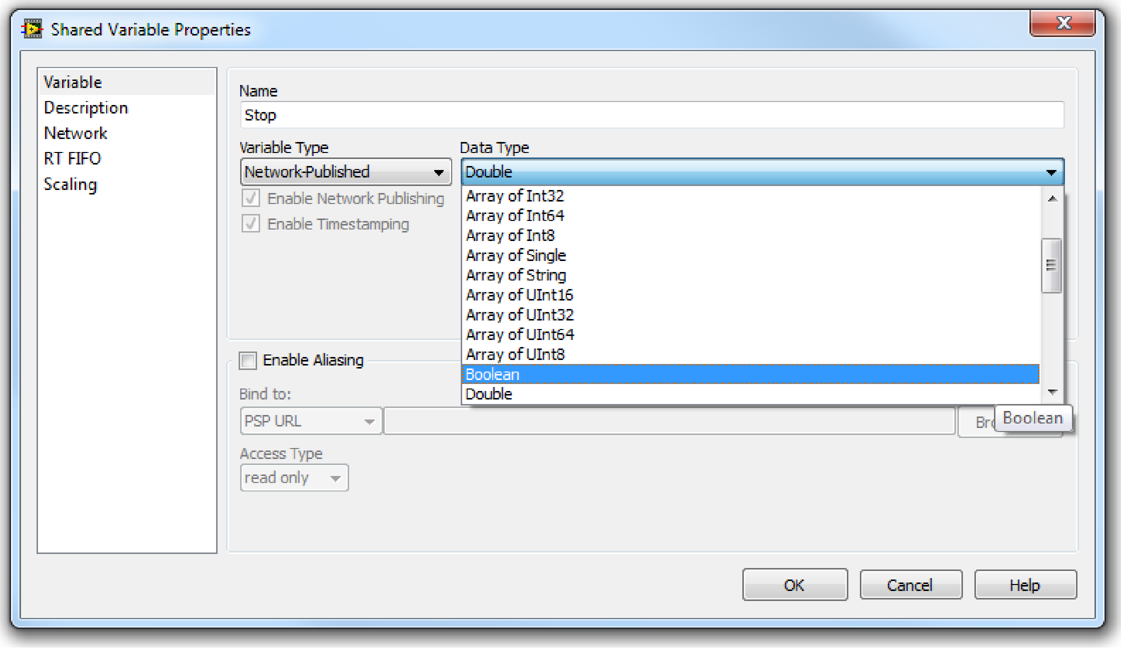
15. Select the **RT FIFO** option from the menu on the left and click the **Enable RT FIFO** check box. Then select **Single Element** for the FIFO Type, and enter **2** for the Number of elements (this would vary depending on the number of I/O channels monitored). This configures the variable to operate as global variable that can maintain the timing requirements, serving as a data buffer between the high priority and low priority tasks. Click **OK**.



☑ Enable RT FIFO
FIFO Type: Single Element
Number of arrays: 1
Number of elements (type: Array of Double): 2

16. Create another single process shared variable in the library you just created. This variable is for the Stop control we are going to create that will stop both loops of the program when directed. This new variable should have all the following settings:

- Name: **Stop**
- Variable Type: **Single Process**
- Data Type: **Boolean**
- RT FIFO: **Enabled** and **Single Element** FIFO Type


Shared Variable Properties

Variable
Description
Network
RT FIFO
Scaling

Name: Stop

Variable Type: Network-Published

Data Type: Double

☑ Enable Network Publishing
☑ Enable Timestamping

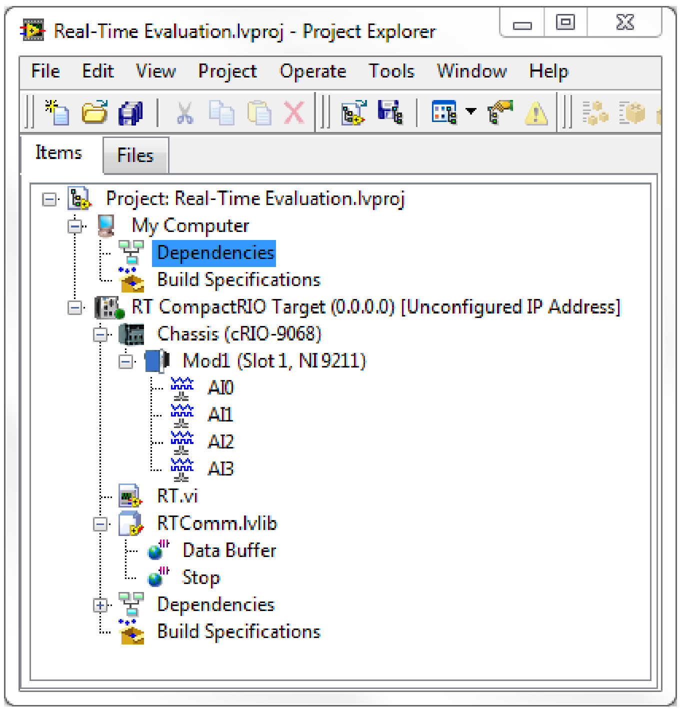
Array of Int32
Array of Int64
Array of Int8
Array of Single
Array of String
Array of UInt16
Array of UInt32
Array of UInt64
Array of UInt8
Boolean
Double

☐ Enable Aliasing

Bind to: PSP URL

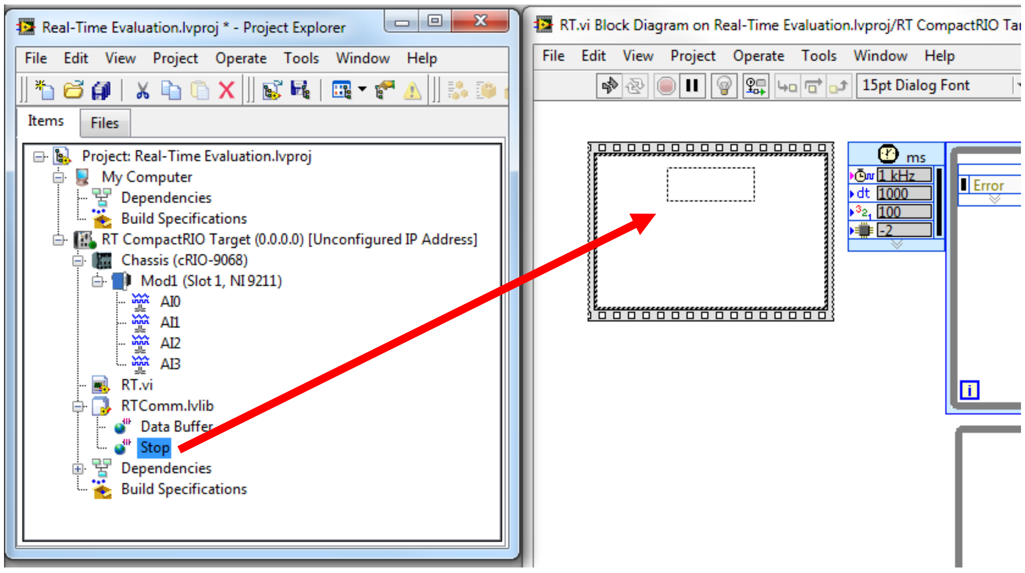
Access Type: read only

17. Once finished, your project should look similar to the image below.

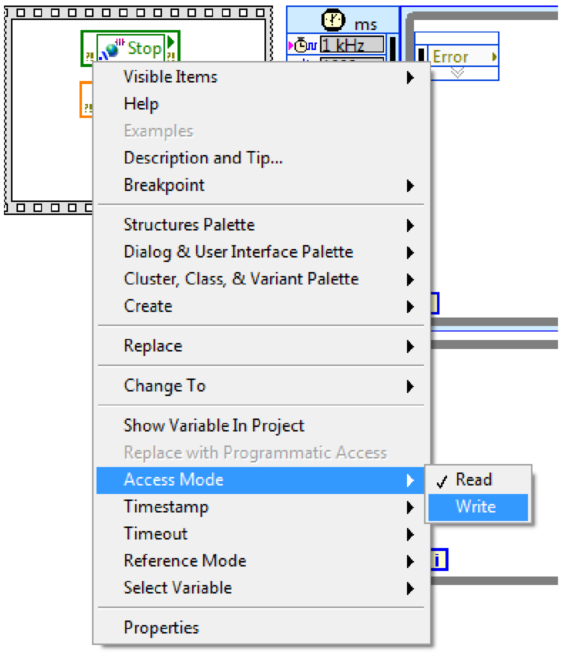

Real-Time Evaluation.lvproj - Project Explorer

- Project: Real-Time Evaluation.lvproj
  - My Computer
    - Dependencies
    - Build Specifications
  - RT CompactRIO Target (0.0.0.0) [Unconfigured IP Address]
    - Chassis (cRIO-9068)
      - Mod1 (Slot 1, NI 9211)
        - AI0
        - AI1
        - AI2
        - AI3
    - RT.vi
    - RTComm.lvlib
      - Data Buffer
      - Stop
    - Dependencies
    - Build Specifications

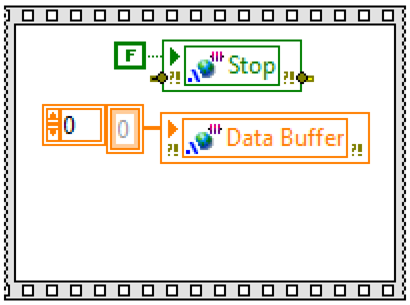
18. Now return to the sequence structure to initialize these variables and an array for the data. Drag an instance of both variables from the project into the structure.


Real-Time Evaluation.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Project: Real-Time Evaluation.lvproj
My Computer
Dependencies
Build Specifications
RT CompactRIO Target (0.0.0.0) [Unconfigured IP Address]
Chassis (cRIO-9068)
Mod1 (Slot 1, NI 9211)
AI0
AI1
AI2
AI3
RT.vi
RTComm.lvlib
Data Buffer
Stop
Dependencies
Build Specifications
RT.vi Block Diagram on Real-Time Evaluation.lvproj/RT CompactRIO Tar
15pt Dialog Font
ms
1 kHz
dt 1000
100
-2
Error

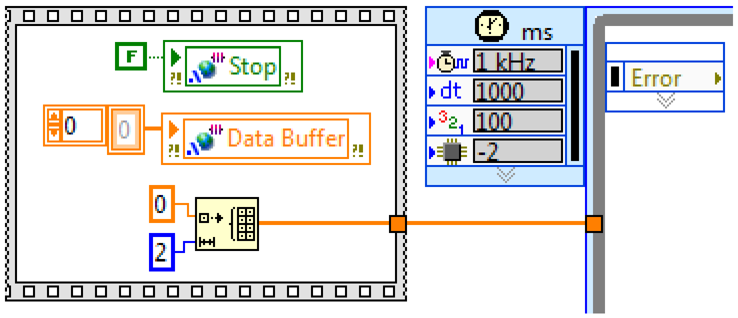
19. Right-click on each variable and select **Access Mode » Write**.



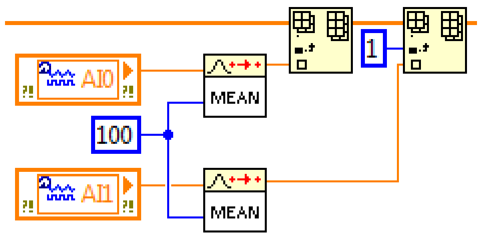
20. Right-click on each variable's input terminal and select **Create » Constant**. Leave the default False and array of zeros values.



21. Place an **Initialize Array.vi** in the structure (Search the Functions palette or navigate to **Programming » Array » Initialize Array**). Right-click the **element** terminal and **Create » Constant**. Set the value to **0**. Do the same for the **dimension size** input terminal and set the constant value to **2**, as you will write two channels of temperature data to the array. Then wire the **initialized array** output through the sequence structure to the border of the timed loop, as shown below.

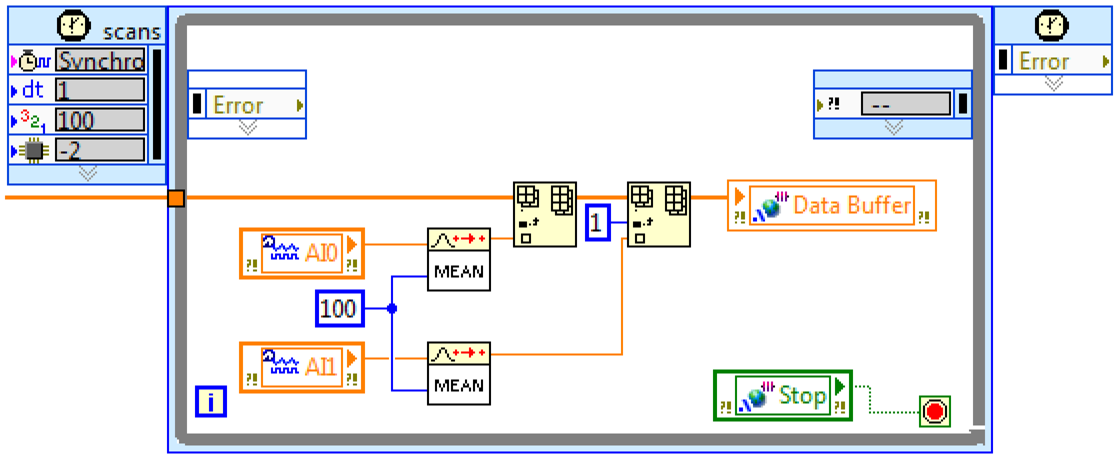


22. Now insert the mean values into the array. Place two instances of the **Insert Into Array.vi** (**Programming » Array » Insert Into Array**) into the timed loop and wire the **mean** output into the **new element** terminal as shown below.

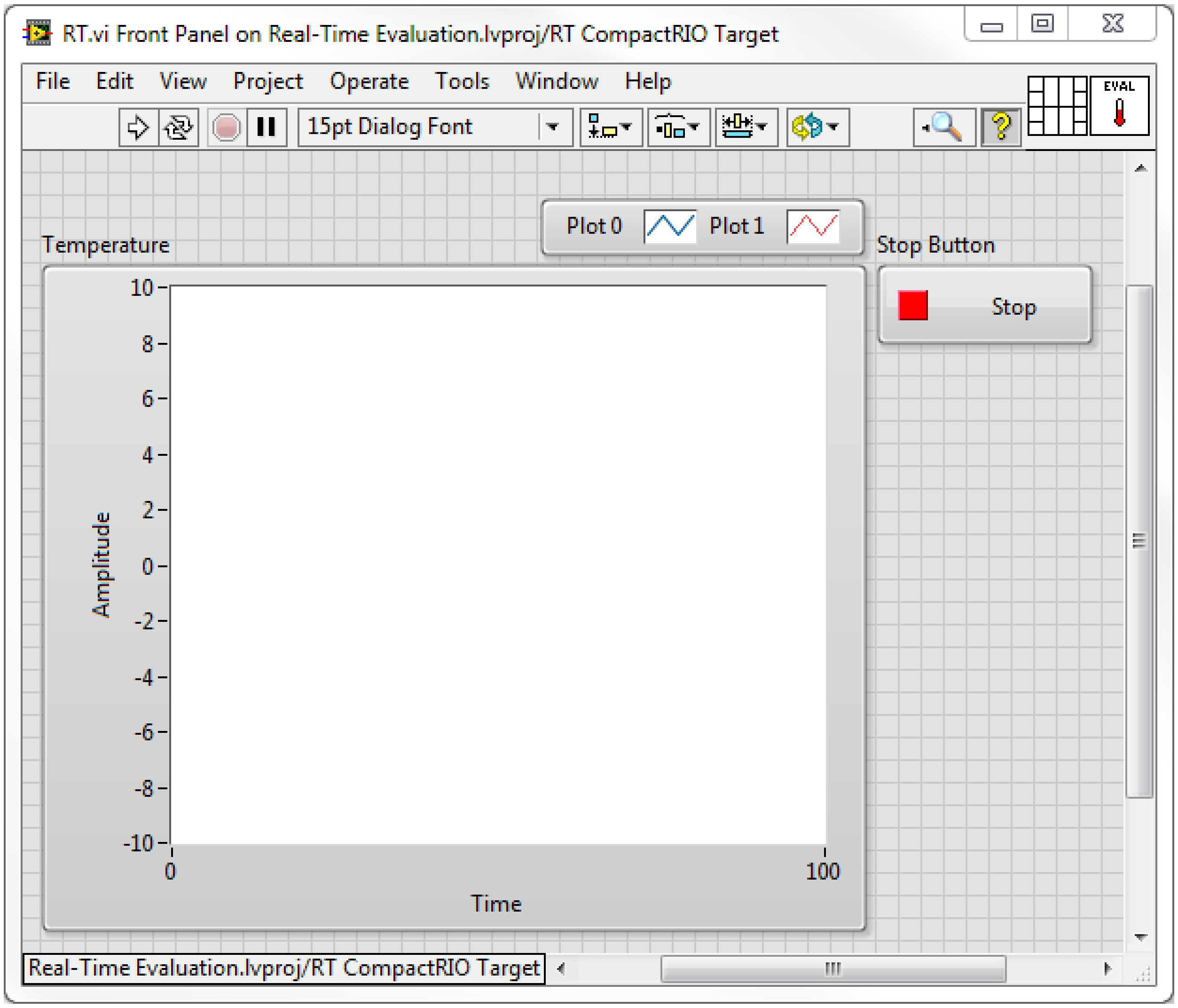


Place one Insert Into Array.vi after the other to build up the array, wiring the **output array** into the **array** input. The first AI0 mean data point is placed into index **0** by default. Create a constant of **1** for the **index** input of the second Insert Into Array.vi to place the AI1 mean data point into the next index.

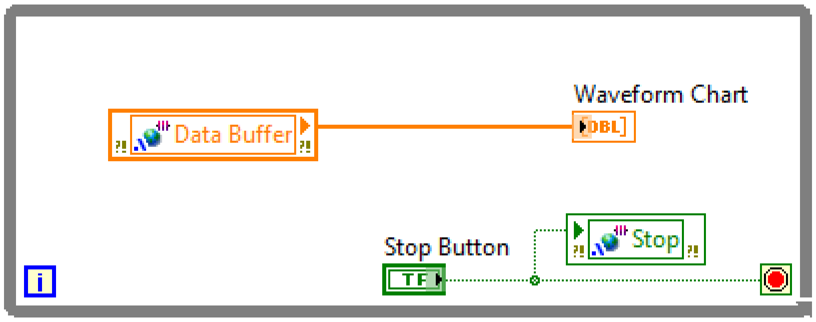
23. Drag an instance of both the **Data Buffer** and **Stop** variables from the project into the timed loop. Wire the **output array** into the **Data Buffer** FIFO variable to transfer data to the while loop. Then wire the **Stop** variable to the **Loop Condition** terminal, as shown below.



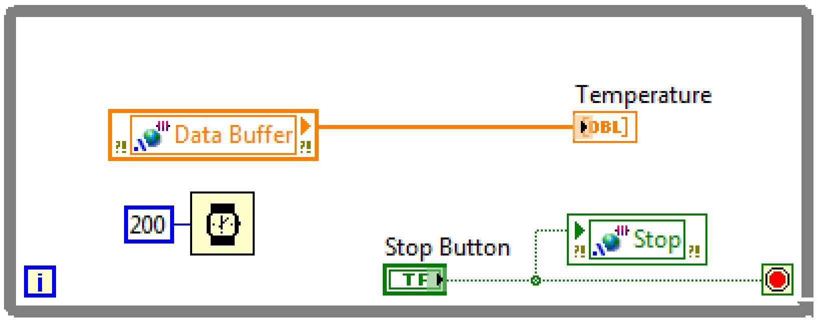
24. Now we will create our user interface to see temperature data and stop the application at run time. Right-click on the Front Panel to bring up the Controls palette and select **Silver » Graph » Waveform Chart**. Then select **Silver » Boolean » Stop Button**. Arrange the user interface as shown below.



25. Now finish wiring the low priority while loop. Drag an instance of both the **Data Buffer** and **Stop** variables into the loop. Wire the **Data Buffer** output into the Waveform Chart, and the Stop Button control value into the **Stop** variable to ensure both loops stop at the same time. Also wire this stop value to the **Loop Condition**.



26. Place down a **Wait (ms).vi** to enforce timing for this task. Right-click on the input terminal and **Create » Constant**, entering a value of 200, to run slower than the high-priority while loop. Finally, Double-click on the **Waveform Chart** label to select it and rename it something meaningful, like **Temperature**.



27. Your basic acquisition and analysis application is now complete. Once you connect CompactRIO hardware you can run the code on the target.

Additional Resources

[Do I Need a Real-Time System?]

[Using NI CompactRIO Scan Mode with NI LabVIEW Software]