EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Základy práce a programovania CompactRIO systémov pre meracie aplikácie

Učební texty k semináři

Autoři:

Ing. Mgr. Márk Jónás (ANV, s.r.o.)

Datum:

29. 6. 2012

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií CZ.1.07/2.3.00/09.0031

TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

OBSAH

1. Úvod

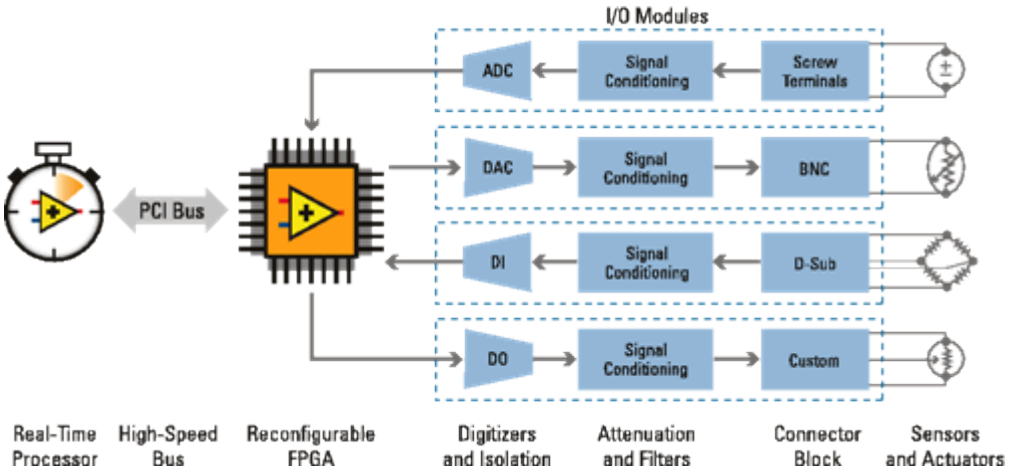
Táto brožúra poskytuje informácie ohľadne základov práce s CompactRIO systémom pre meracie a monitorovacie aplikácie. Zároveň ponúka aj prehľad architektúry a programovacích modelov.

1. The RIO Architecture
2. Starting a New CompactRIO Project in LabVIEW
3. Select the Appropriate Programming Model
4. CompactRIO Scan Mode Tutorial
5. LabVIEW FPGA Tutorial
6. Discussion and Feedback
7. View Additional CompactRIO Development Resources

2. The RIO Architecture

The National Instruments CompactRIO programmable automation controller is an advanced embedded control and data acquisition system designed for applications that require high performance and reliability. With the system's open, embedded architecture, small size, extreme ruggedness, and flexibility, engineers and embedded developers can use COTS hardware to quickly build custom embedded systems. NI CompactRIO is powered by National Instruments LabVIEW FPGA and LabVIEW Real-Time technologies, giving engineers the ability to design, program, and customize the CompactRIO embedded system with easy-to-use graphical programming tools.

CompactRIO combines an embedded real-time processor, a high-performance FPGA, and hot-swappable I/O modules. Each I/O module is connected directly to the FPGA, providing low-level customization of timing and I/O signal processing. The FPGA is connected to the embedded real-time processor via a high-speed PCI bus. This represents a low-cost architecture with open access to low-level hardware resources. LabVIEW contains built-in data transfer mechanisms to pass data from the I/O modules to the FPGA and also from the FPGA to the embedded processor for real-time analysis, postprocessing, data logging, or communication to a networked host computer.



2.1. C Series I/O Modules

A variety of I/O types are available including voltage, current, thermocouple, RTD, accelerometer, and strain gauge inputs; up to ±60 V simultaneous-

sampling analog I/O; 12, 24, and 48 V industrial digital I/O; 5 V/TTL digital I/O; counter/timers; pulse generation; and high voltage/current relays. Because the modules contain built-in signal conditioning for extended voltage ranges or industrial signal types, you can usually connect wires directly from the C Series modules to your sensors and actuators.

2.2. FPGA

The embedded FPGA is a high-performance, reconfigurable chip that engineers can program with LabVIEW FPGA tools. Traditionally, FPGA designers were forced to learn and use complex design languages such as VHDL to program FPGAs. Now, any engineer or scientist can use graphical LabVIEW tools to program and customize FPGAs. Using the FPGA hardware embedded in CompactRIO, you can implement custom timing, triggering, synchronization, control, and signal processing for your analog and digital I/O.

2.3. Real-Time Processor

The CompactRIO embedded system features an industrial 400 MHz Freescale MPC5200 processor that deterministically executes your LabVIEW Real-Time applications on the reliable Wind River VxWorks real-time operating system. LabVIEW has built-in functions for transferring data between the FPGA and the real-time processor within the CompactRIO embedded system. Choose from more than 600 built-in LabVIEW functions to build your multithreaded embedded system for real-time control, analysis, data logging, and communication. You can also integrate existing C/C++ code with LabVIEW Real-Time code to save on development time.

2.4. Size and Weight

Size, weight, and I/O channel density are critical design requirements in many embedded applications. A four-slot reconfigurable embedded system measures 179.6 by 88.1 by 88.1 mm and weighs just 1.58 kg.

2.5. Application Examples

With the low cost and reliability of CompactRIO, as well as its suitability for high-volume embedded measurement and control applications, you can adapt it to solve a wide variety of industry and application challenges. Examples include:

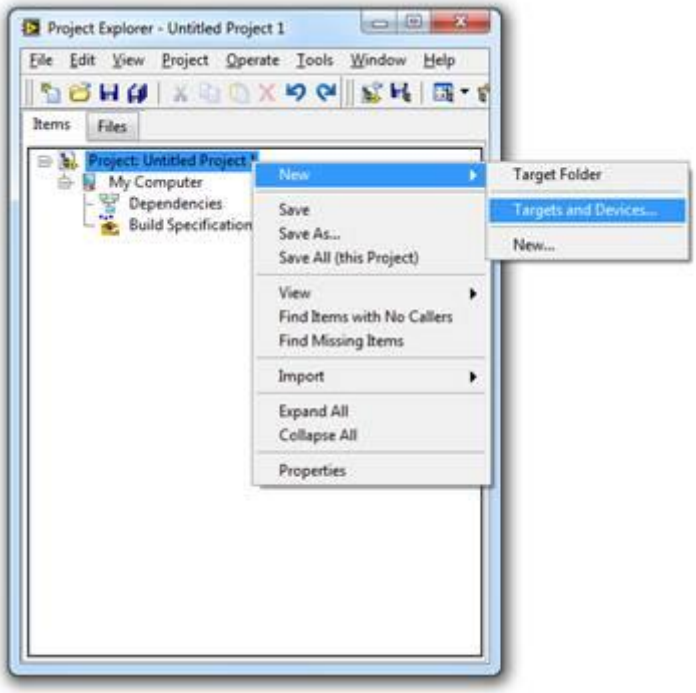
- In-vehicle data acquisition, data logging, and control
- Machine condition monitoring and protection
- Embedded system prototyping
- Remote and distributed monitoring
- Embedded data logging
- Custom multiaxis motion control
- Electrical power monitoring and power electronics control
- Servo-hydraulic and heavy machinery control
- Batch and discrete control
- Mobile/portable noise, vibration, and harshness (NVH) analysis

3. STARTING A NEW COMPACTRIO PROJECT IN LABVIEW

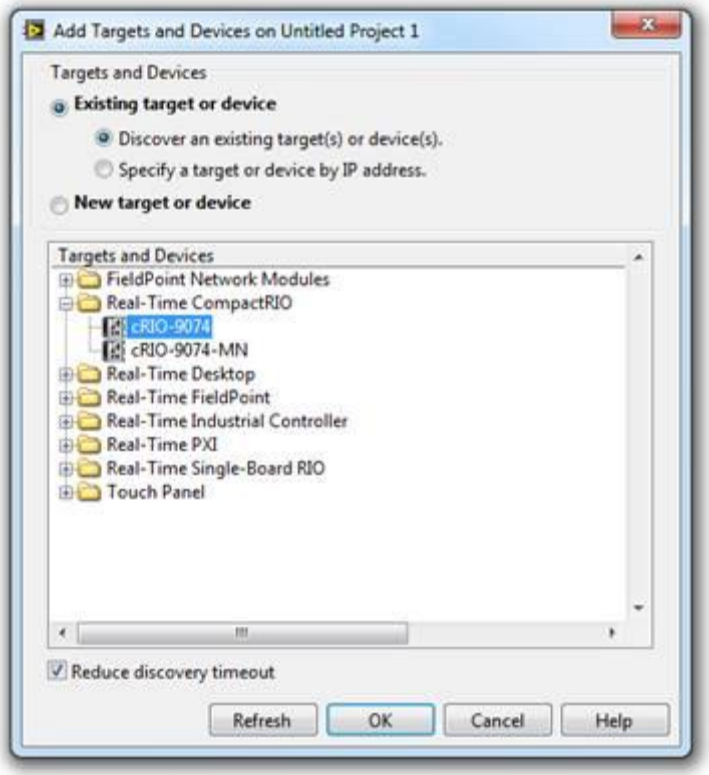
Begin by creating a new project in LabVIEW, where you will manage your code and hardware resources.

1. Create a new project in LabVIEW by selecting **File » New Project**

2. To add your CompactRIO 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 systems on your network or add offline systems. 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 you system is on a remote subnet, you can also select to manually enter the IP address.

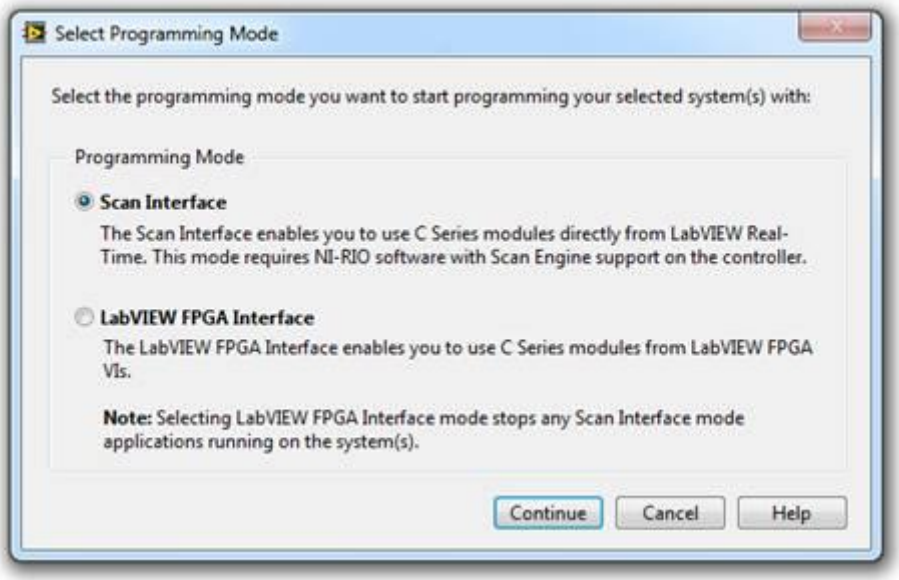

Add Targets and Devices on Untitled Project 1
Targets and Devices
Existing target or device
Discover an existing target(s) or device(s).
Specify a target or device by IP address.
New target or device
Targets and Devices
FieldPoint Network Modules
Real-Time CompactRIO
cRIO-9074
cRIO-9074-MN
Real-Time Desktop
Real-Time FieldPoint
Real-Time Industrial Controller
Real-Time PXI
Real-Time Single-Board RIO
Touch Panel
Reduce discovery timeout
Refresh
OK
Cancel
Help

4. Select the Appropriate Programming Model

LabVIEW provides two programming models for CompactRIO systems. If you have LabVIEW Real-Time and LabVIEW FPGA on your development computer, you will be prompted to select which programming model you would like to use. You can change this setting later in the LabVIEW Project if needed.

Scan Interface (CompactRIO Scan Mode) – this option allows you to program the real-time processor of your CompactRIO system, but not the FPGA. In this mode, NI provides a pre-defined personality for the FPGA that periodically scans the I/O and places it in a memory map, making it available to LabVIEW Real-Time. CompactRIO Scan Mode is sufficient for applications that require single-point access to I/O at rates of a few hundred hertz. To learn more about scan mode, read the Using CompactRIO Scan Mode with NI LabVIEW white paper and view the benchmarks.

LabVIEW FPGA Interface – this option allows you to unlock the real power of CompactRIO by customizing the FPGA personality in addition to programming the real-time processor, achieving performance that would typically require custom hardware. Using LabVIEW FPGA, you can implement custom timing and triggering, off-load signal processing and analysis, create custom protocols, and access I/O at its maximum rate.



Select the appropriate programming model for your application.

LabVIEW will now attempt to detect the chassis and C Series I/O modules present in your system and automatically add them to the LabVIEW Project. Note: If your system was not discovered and you choose to add it offline, you will need to add the chassis and C Series I/O manually.

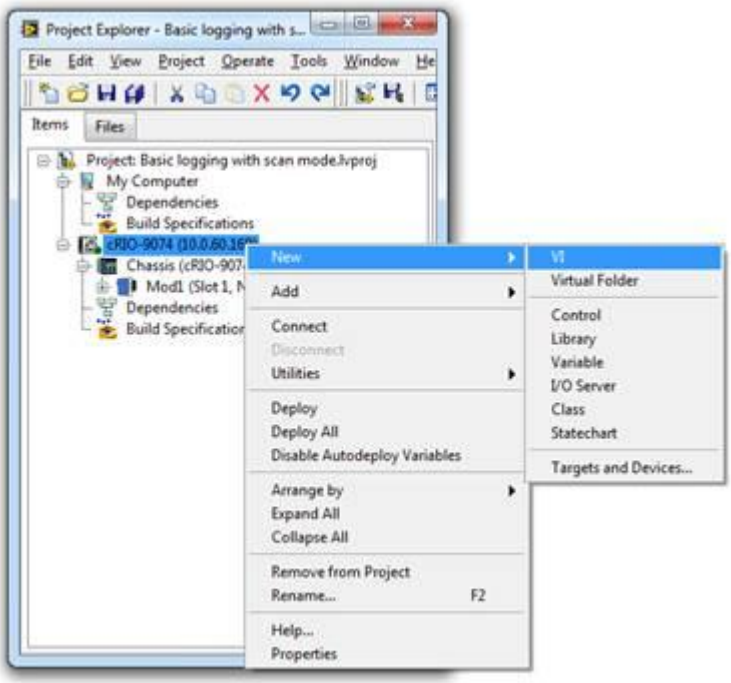
Once your system has been added to the LabVIEW Project, proceed to either the CompactRIO Scan Mode Tutorial or LabVIEW FPGA Tutorial below, depending on which programming model you selected.

4.1. CompactRIO Scan Mode Tutorial

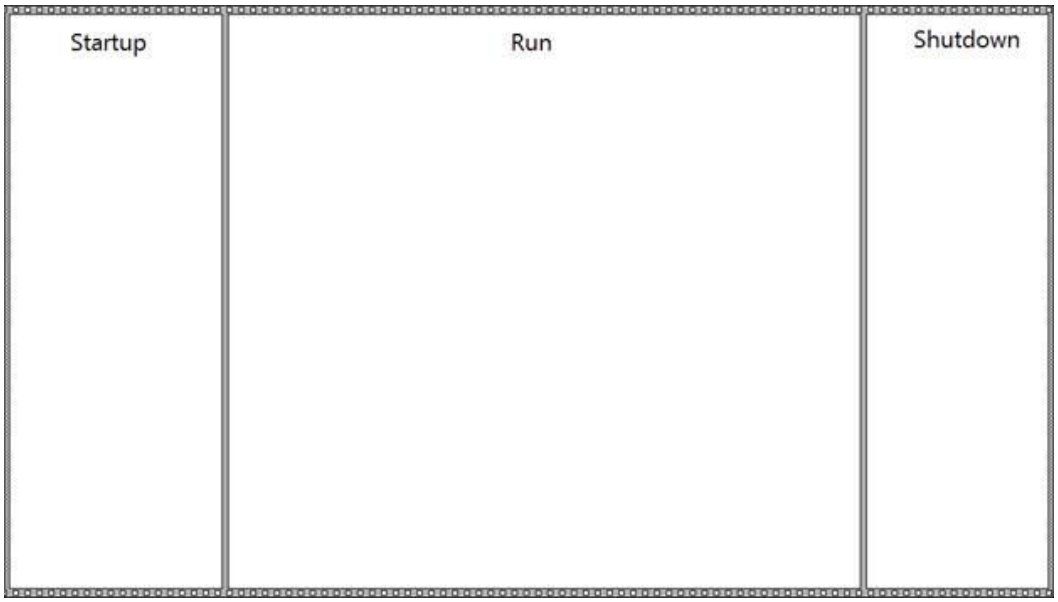
This section will walk you through creating a basic monitoring application on CompactRIO using scan mode. If you chose to use the LabVIEW FPGA Interface, see the LabVIEW FPGA Tutorial below. You should now have a new LabVIEW Project that contains your CompactRIO system, including the controller, chassis, and C Series I/O modules. In this tutorial we will be using an NI 9211 Thermocouple input module; however, the process can be followed for any analog input module.

1. Save the project by selecting **File»Save** and entering Basic logging with scan mode. Click **OK**.

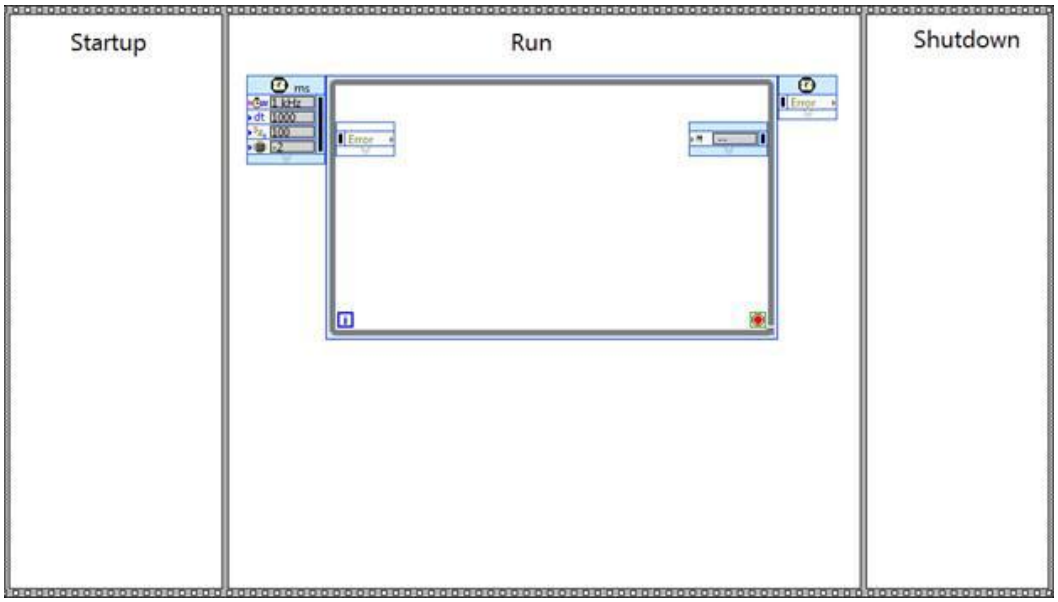
2. This project will only contain two VIs. First, you will create the real-time VI, which is the LabVIEW Real-Time application that runs embedded on the CompactRIO controller. Create this the VI by right-clicking on the CompactRIO real-time controller in the project and selecting New»VI. Save the VI as RT.vi.



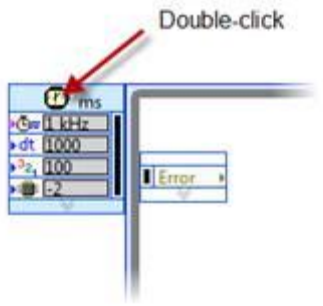
3. The basic operation of this application will include three routines: startup, run, and shutdown. A flat sequence structure is an easy way to enforce this order of operation. Place a **flat sequence structure** with three frames on your RT.vi block diagram as shown below.



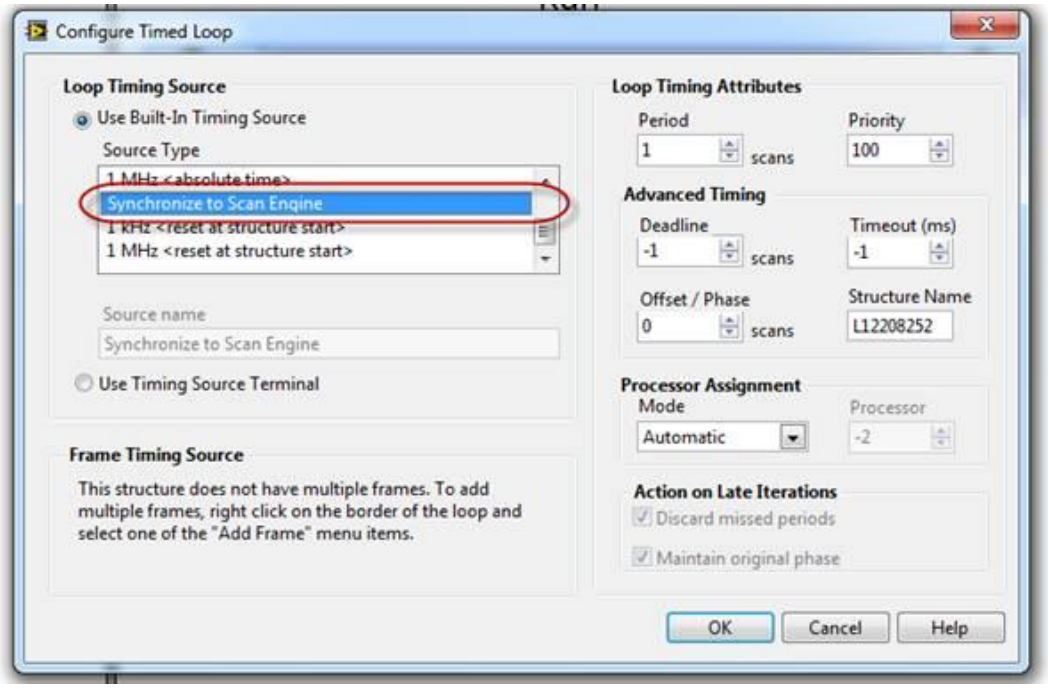
4. Now add a timed loop to the Run frame of the sequence structure. Timed loops provide the ability to synchronize code to various time basis, including the NI Scan Engine that reads and writes scan mode I/O.



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

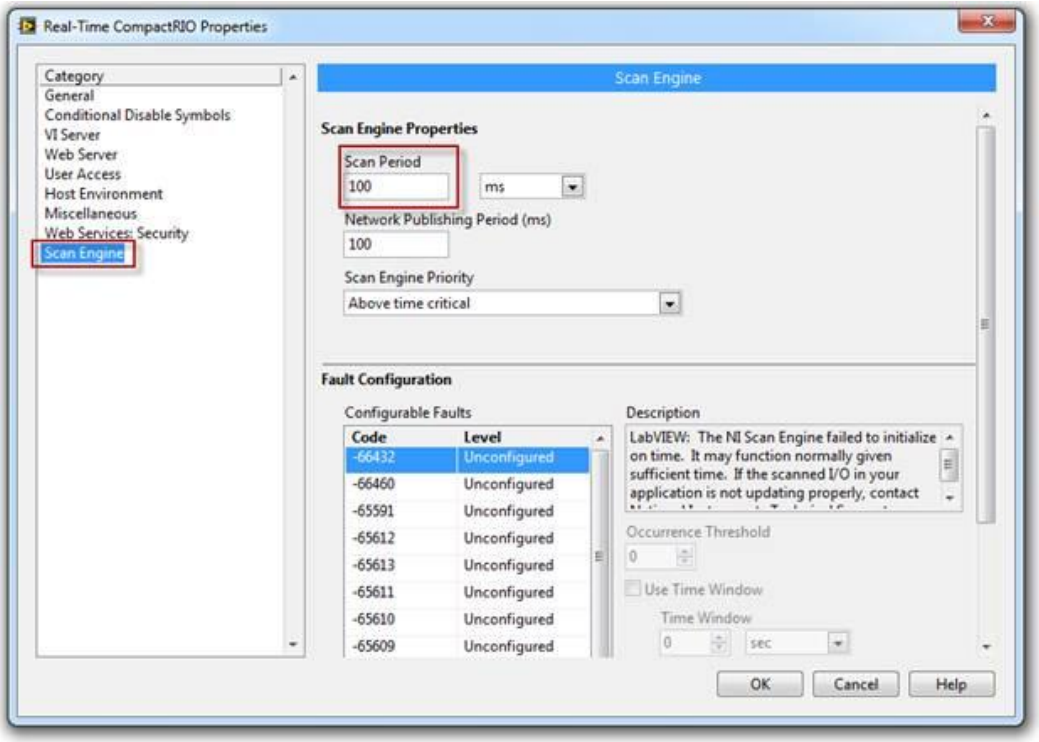


6. Select **Synchronize to Scan Engine** as the Loop Timing Source. Click OK. This will cause the code in the timed loop to execute once, immediately after each I/O scan, ensuring that any I/O values used in this timed loop are the most recent values.

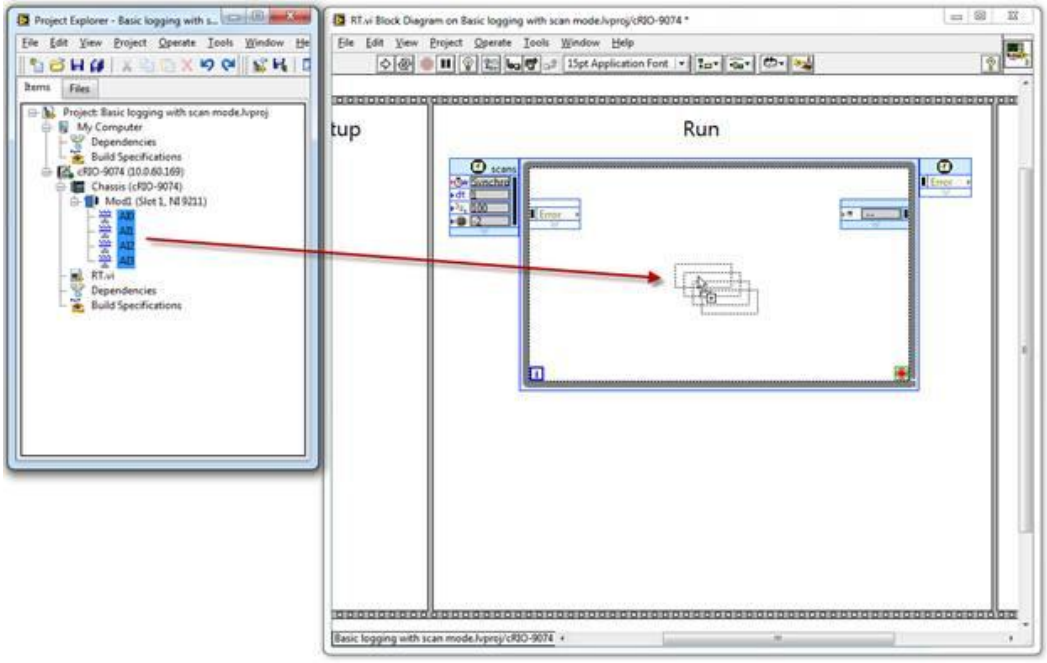


7. The previous step configured the timed loop to run synchronized to the scan engine. Now configure the rate of the scan engine itself by right-clicking on the CompactRIO real-time controller in the LabVIEW Project and selecting **Properties**.

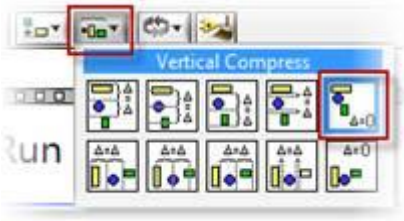
8. Select **Scan Engine** from the categories on the left and enter 100ms as the Scan Period. This will cause all of the I/O in the CompactRIO system to be updated every **100ms** (10Hz). The Network Publishing Period can also be set from this page, which controls how often the I/O values are published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



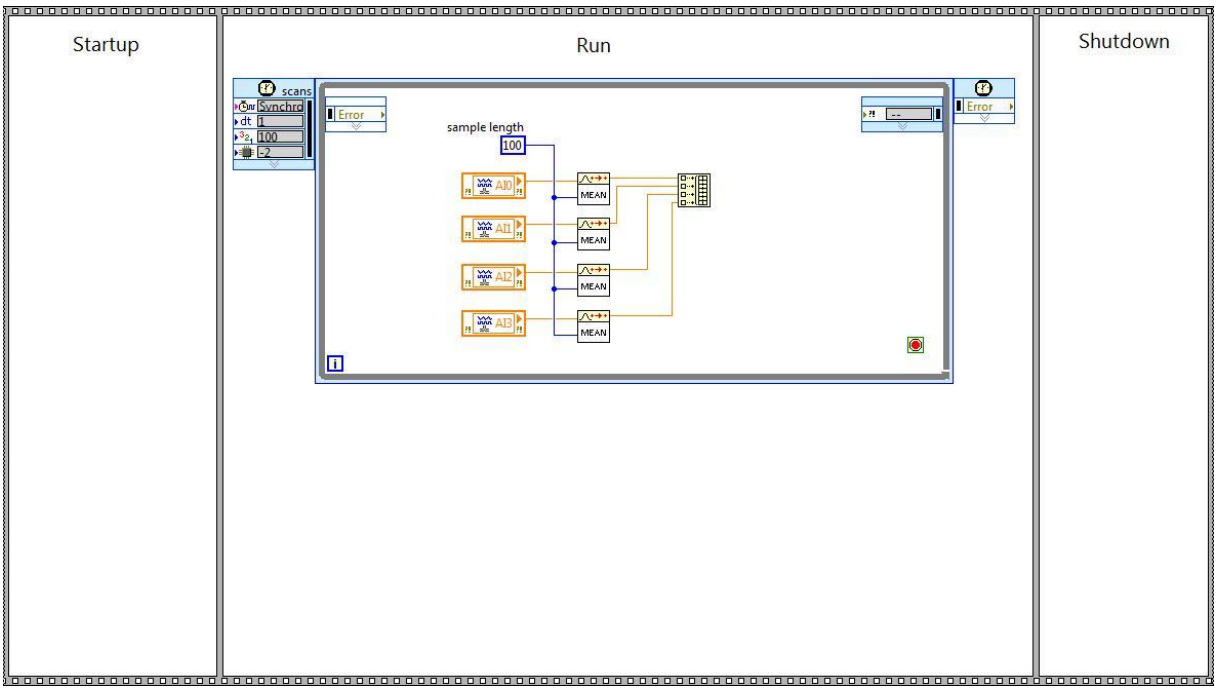
9. Now that you have configured the I/O scan rate, it is time to add the I/O reads to your application for monitoring. When using CompactRIO Scan Mode, you can simply drag and drop the I/O variables from the LabVIEW Project to the block diagram. Expand the CompactRIO real-time controller, chassis, and the I/O module you would like to access. Select all of the channels below the module by clicking on them and using the shift key, then drag and drop them into the timed loop on your RT.vi diagram as shown below.



Tip: Use the Align Objects » Left Edges and Distribute Objects » Vertical Compress items on the LabVIEW toolbar to organize the I/O variables on your diagram.

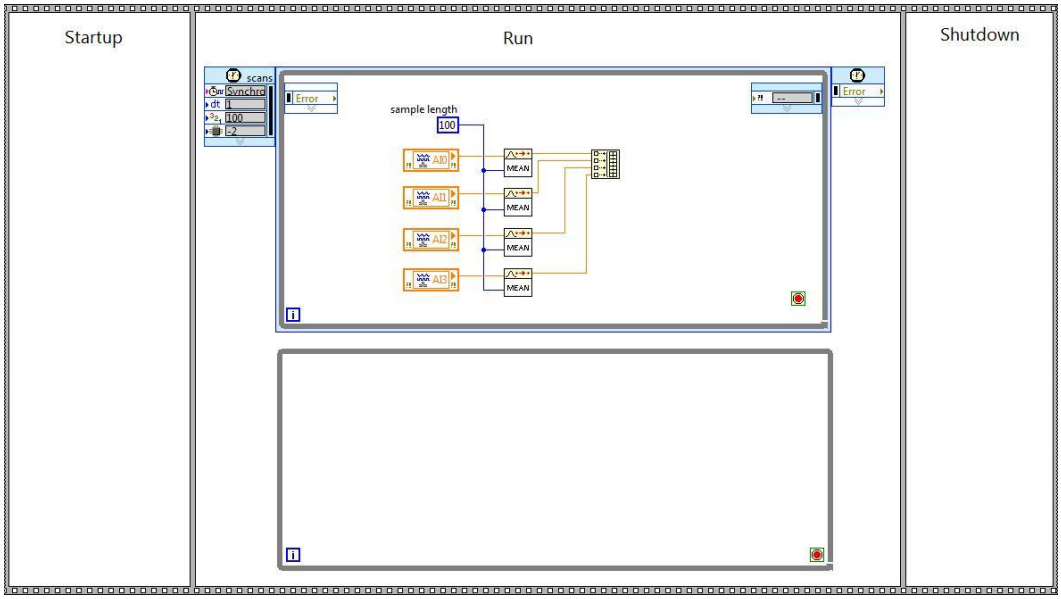


10. In addition to simply monitoring the I/O values remotely, we will add some analysis that runs embedded on the CompactRIO controller. Place four **Mean PtByPt** VIs in the timed loop and wire the I/O variables to their inputs. Create a constant with a value of **100** for the **sample length** input. Wire the results of the mean functions to a **Build Array** function.

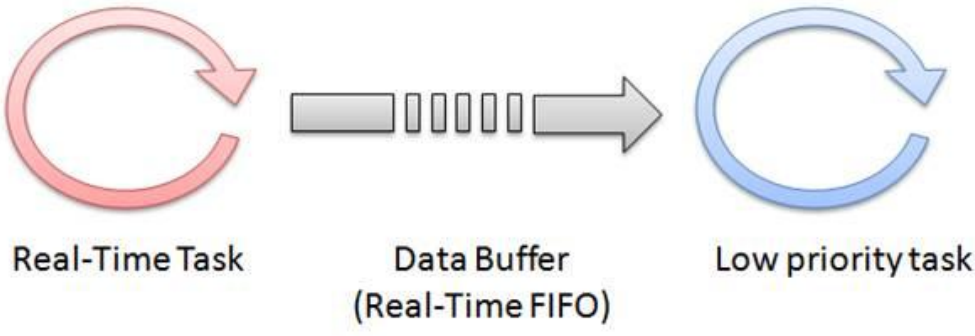


11. The next logical step would be to write the data to the network using a shared variable; however, because network communication takes an undermined amount of time, it is necessary to separate the I/O acquisition task and the network communication. Neglecting this requirement could lead to losing data, since writing a data packet to the network may take longer than the I/O scan, causing a sample to be missed. Place a normal while loop in the

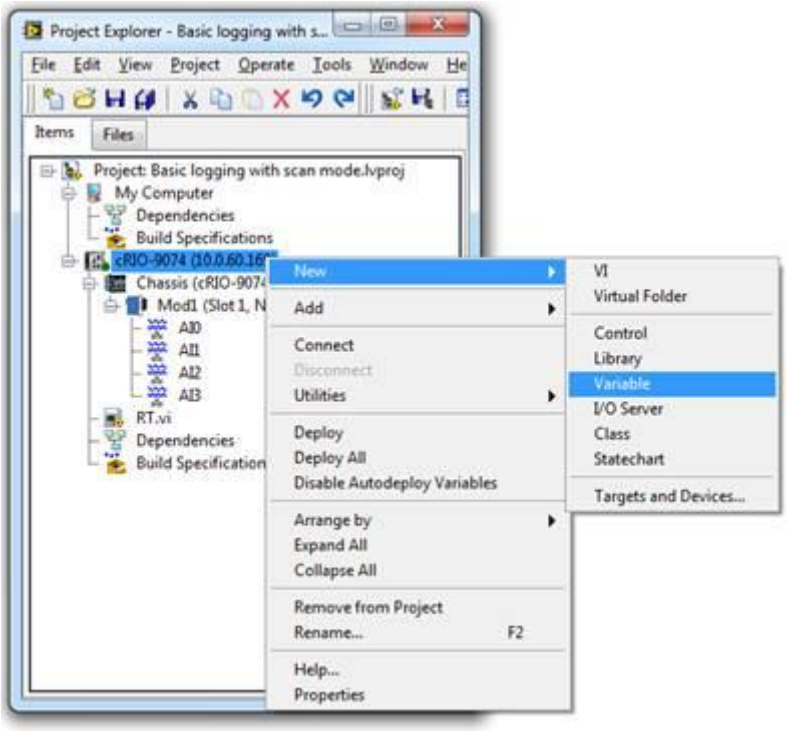
Run frame under the timed loop, which will be used for the network communication.



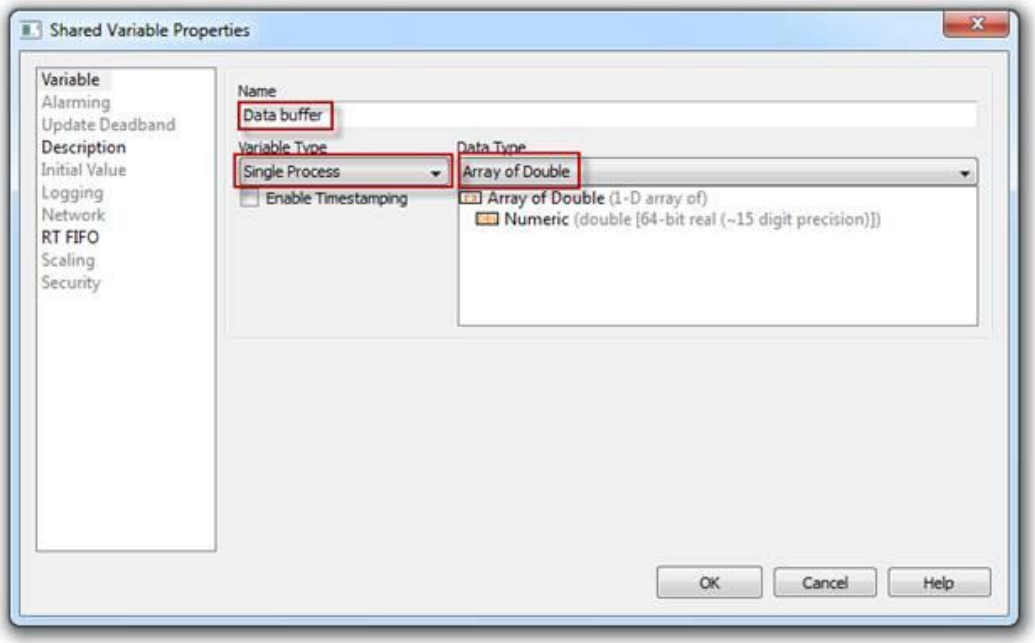
In order write the data to the network in the regular while loop, you need to transfer the latest result of the mean calculation from the timed loop using a real-time FIFO. This will provide a buffer between the two loops. The timed loop will run, synchronized the I/O scan, and write the latest result of the mean calculation to the buffer each time. Then, the regular while loop will read the data out of the buffer and write it to the network. Separating the I/O task and network communication in this way allows your timed loop to run with "real-time" performance, meaning that it will always finish on time.



12. In the LabVIEW Project, right-click on the CompactRIO real-time controller and select **New»Variable**.

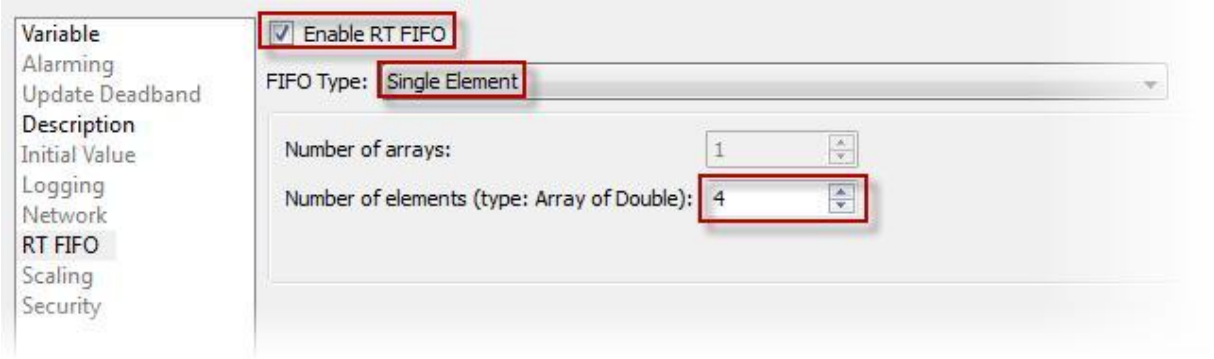


13. Name the variable Data buffer, select **Single Process** as the Variable Type, and **Array of Double** as the Data Type. This will create a locally scoped variable (no network publishing) that contains and array of double precision floating point numbers. Then select **RT FIFO** from the menu on the left.



14. With the RT FIFO category selected, select the **Enable RT FIFO** check box, select **Single Element** for the FIFO Type, and enter 4 for the Number of

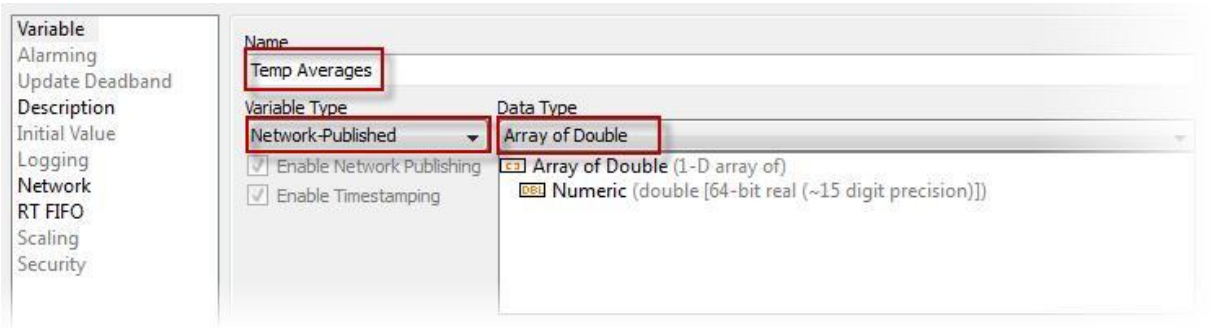
elements (if you are monitoring a number of channels other than 4, enter that instead.) This configures the variable to operate as a real-time-safe global variable, which can serve as the data buffer between our real-time and low priority tasks. The variable will hold one array that contains four double precision numbers. Therefore, we are only keeping track of the latest result of the mean calculation, not every result calculated. Click **OK**.



15. Place a copy of the **Data buffer** variable in the timed loop and one in the regular while loop.

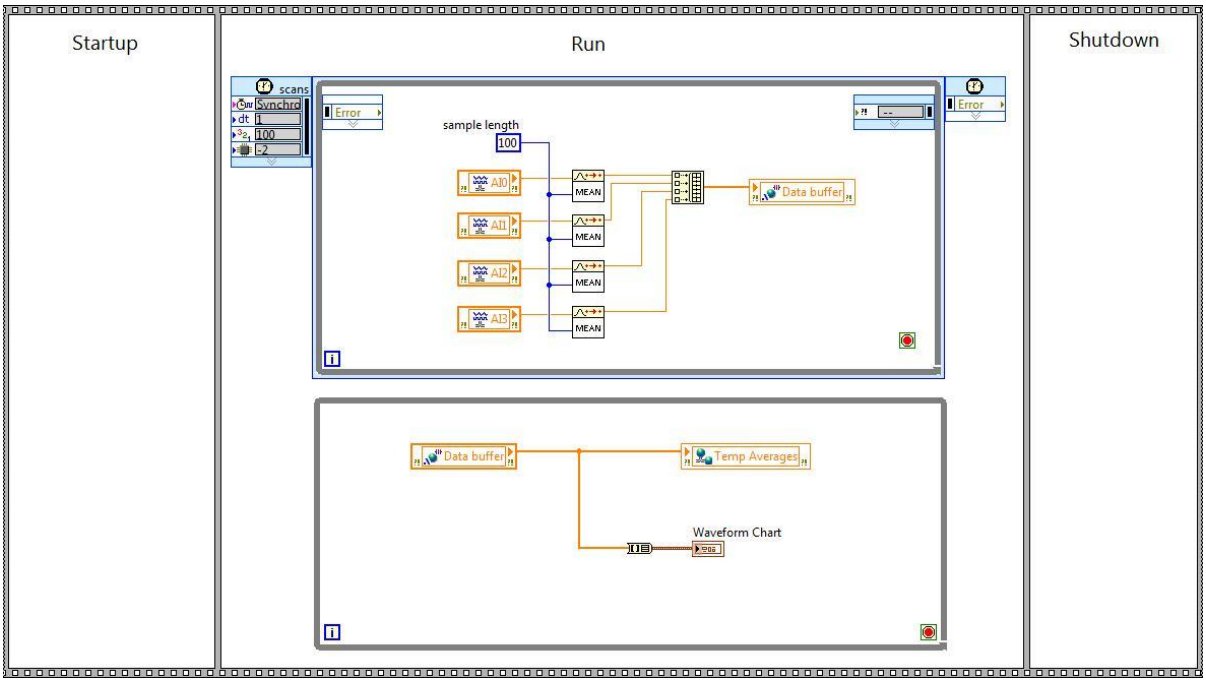
16. Create another variable to perform the network communication. Use the following configuration:

-
    -
        - Name: **Temp Averages**
        - Variable Type: **Network-Published**
        - Data Type: **Array of Double**



17. Place the **Temp Averages** variable in the regular while loop and wire the output of the **Data buffer** variable to its input.

18. Also, place a **Waveform Chart** on the front panel. Use an **Array to Cluster** function to format the data for the chart. Right-click on the Array to Cluster function and select **Cluster Size**... Enter **4** for the size, since there are four elements in the array. Wire the diagram as shown.

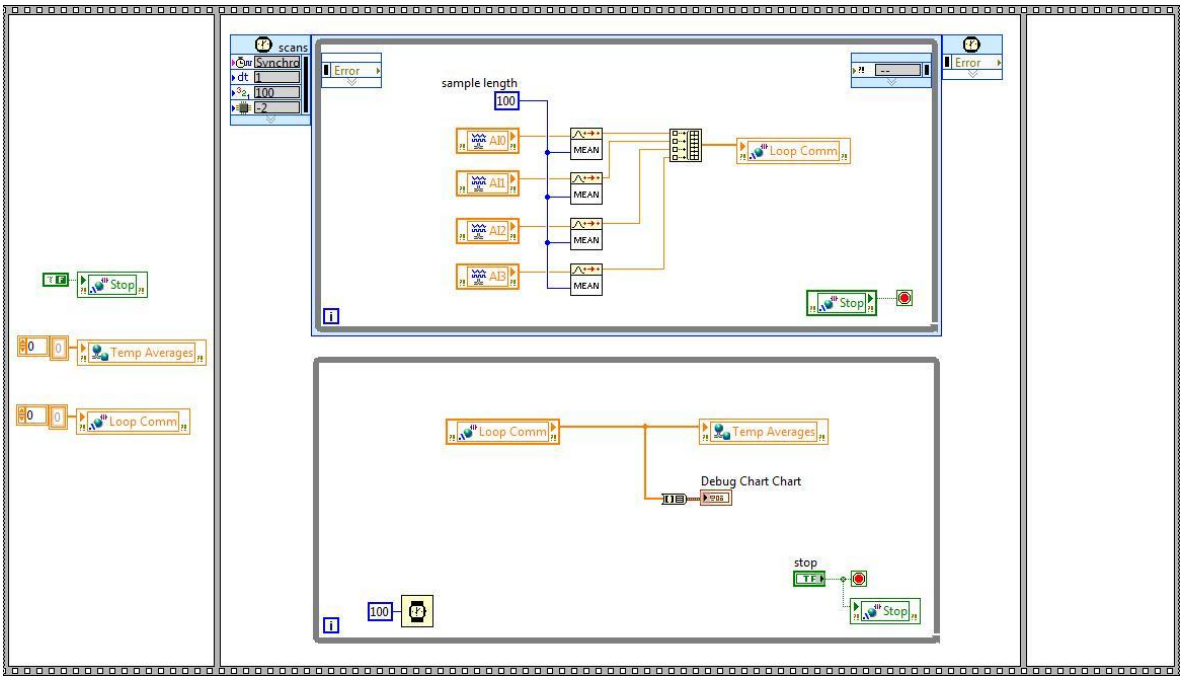


19. To control the stop condition of the while loops, create a shared variable with the following properties:

- Name: **stop**
- Variable Type: **Single Process**
- Data Type: **Boolean**
- RT FIFO: **Single Element**

20. Place the **stop** variable in each loop and create a Boolean stop button on the front panel and place it in the regular while loop.

21. To complete the real-time application, place a copy of each variable in the startup frame with default values wired to them as shown. Also place a **Wait (ms)** function in the regular while loop with a value of **100** to time the network publishing.

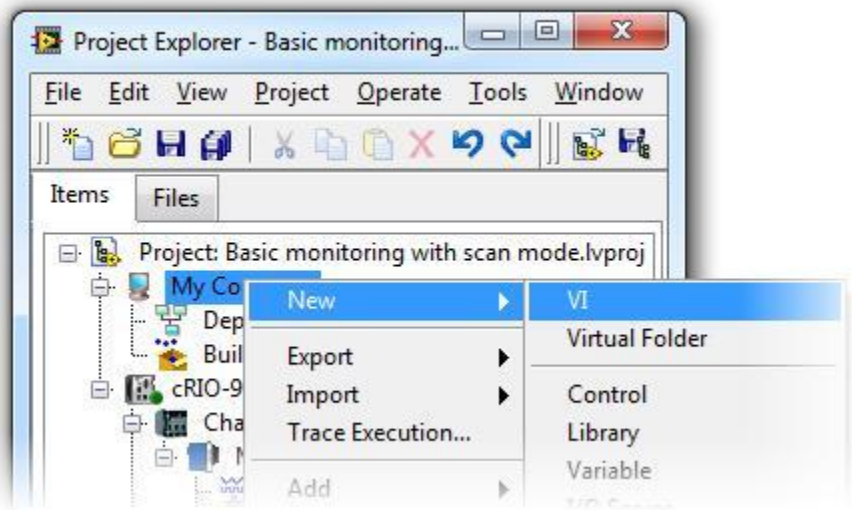


22. Click Run on RT.vi, click Save for any unsaved items, and click OK on any dialogs or warnings about applying changes to the CompactRIO system. LabVIEW will now deploy your VI over Ethernet to run embedded on the CompactRIO system.

23. Once the VI deploys and begins running, view the front panel of your VI to see the current I/O values plotted on the waveform chart. Stop the VI once you verify it is working.

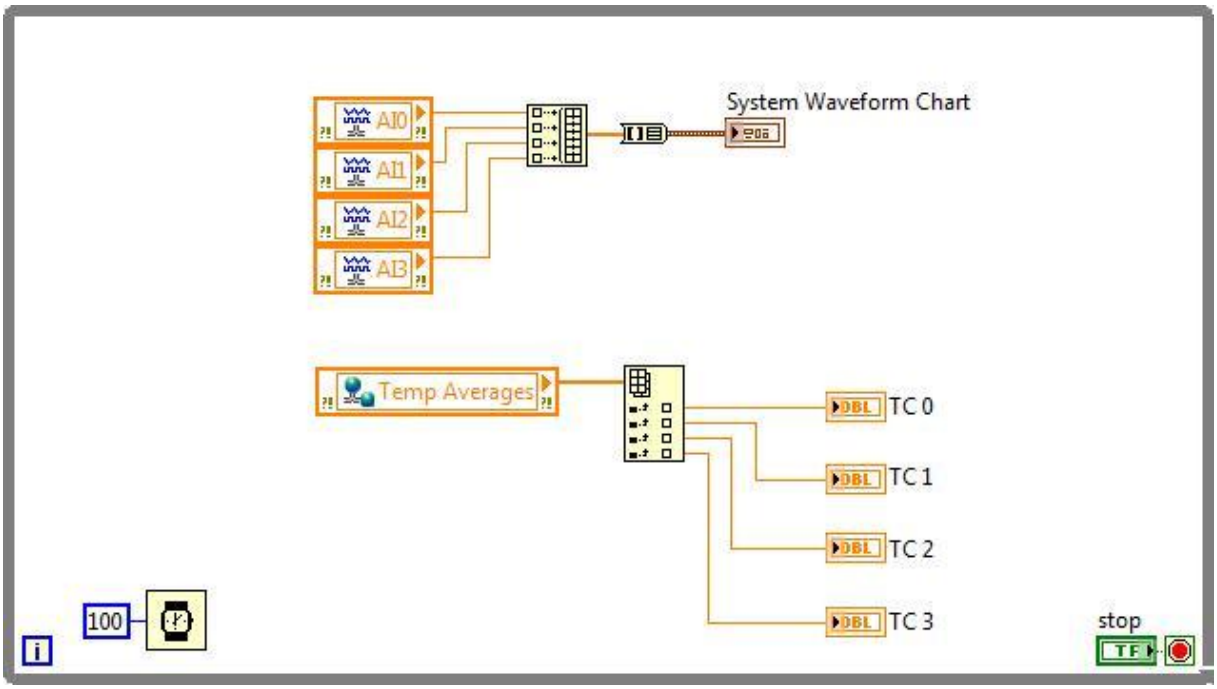
Now, we will create a host VI that can run on a remote machine and communicate with this embedded application.

1. Create a new VI under the My Computer item in the LabVIEW Project. Save the VI as HMI.vi.



2. Create a simple VI that reads from the I/O variables and plots them on a waveform chart, using a Build Array function and an Array to Cluster function, configured with a Cluster Size of 4. Note that I/O variables can be used directly on the Windows host VI, since they are automatically network published by the NI Scan Engine.

3. Use an Index Array function to unpack the Temp Averages and place them on indicators on the front panel. The completed diagram might look like the shown. Also, add timing to the loop with a Wait (ms) function.



4. You have completed this application and can now run it. Run RT.vi first, so that it is running on the CompactRIO controller. Then run HMI.vi, which will connect to the shared variables that are hosted on the CompactRIO controller and display the data.

The host VI in this example could be placed on any computer on the network to monitor the embedded application running on CompactRIO.

At this point you have successfully created an embedded logging application with LabVIEW and CompactRIO.

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz
info@crr.vutbr.cz