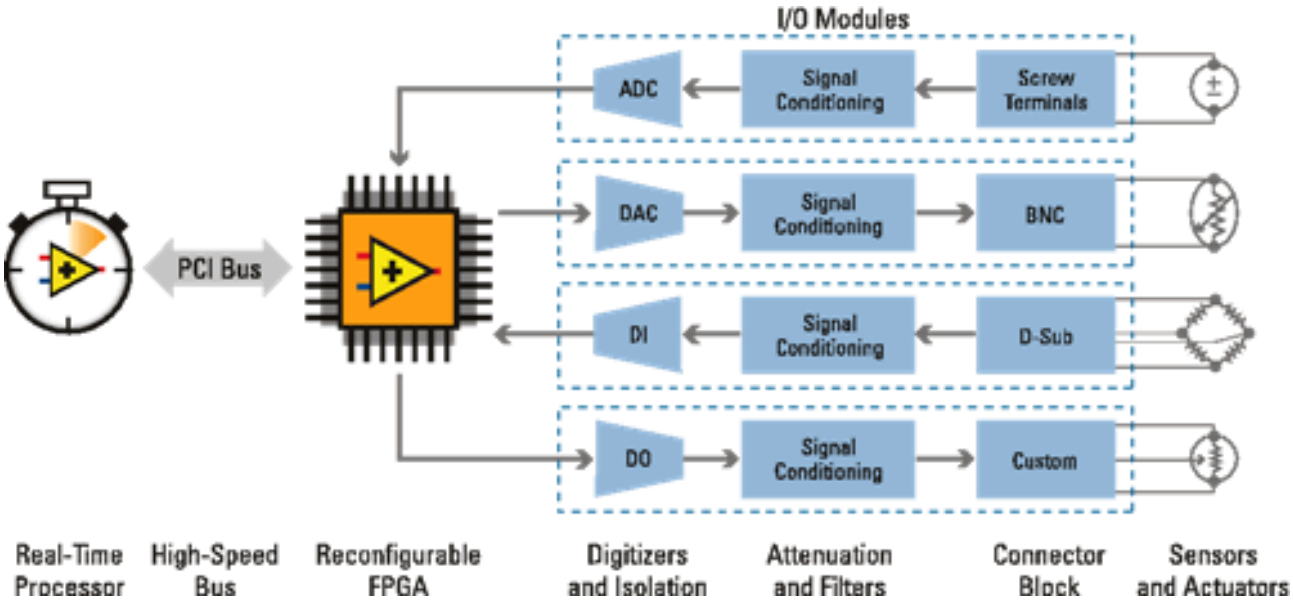
# Getting Started with CompactRIO - Performing Basic Control

Click here to view the list of other articles

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.




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.

### 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.

### 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.

### 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 (7.07 by 3.47 by 3.47 in.) and weighs just 1.58 kg (3.47 lb).

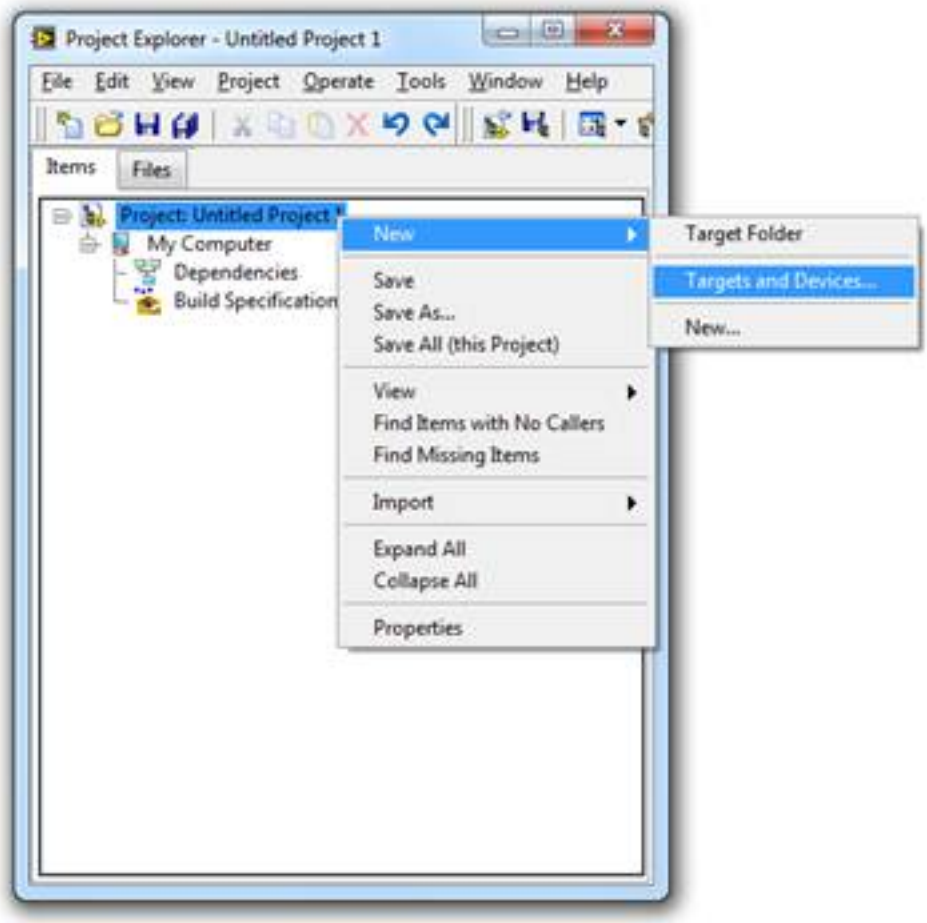
## Starting a New CompactRIO Project in LabVIEW

### Starting a New CompactRIO Project in LabVIEW

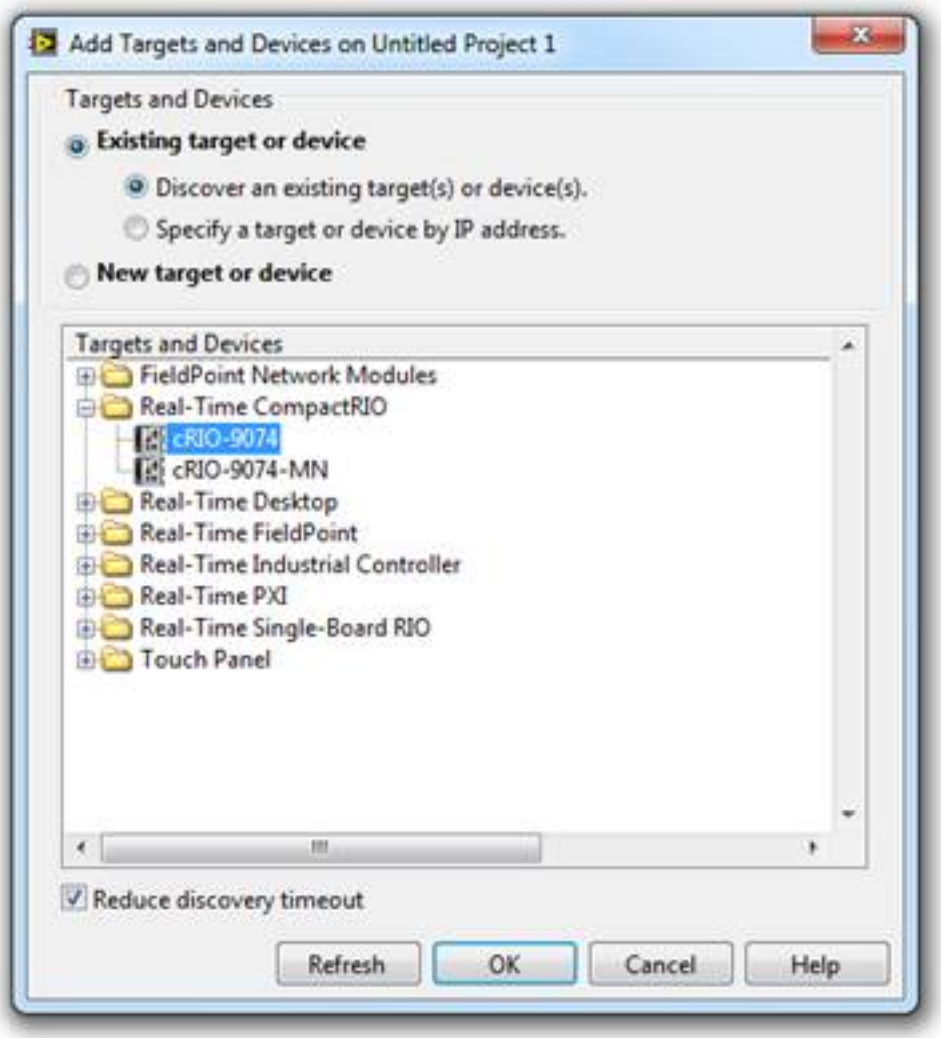
Begin by creating a new project in LabVIEW, where you will manage your code and hardware resources.

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.



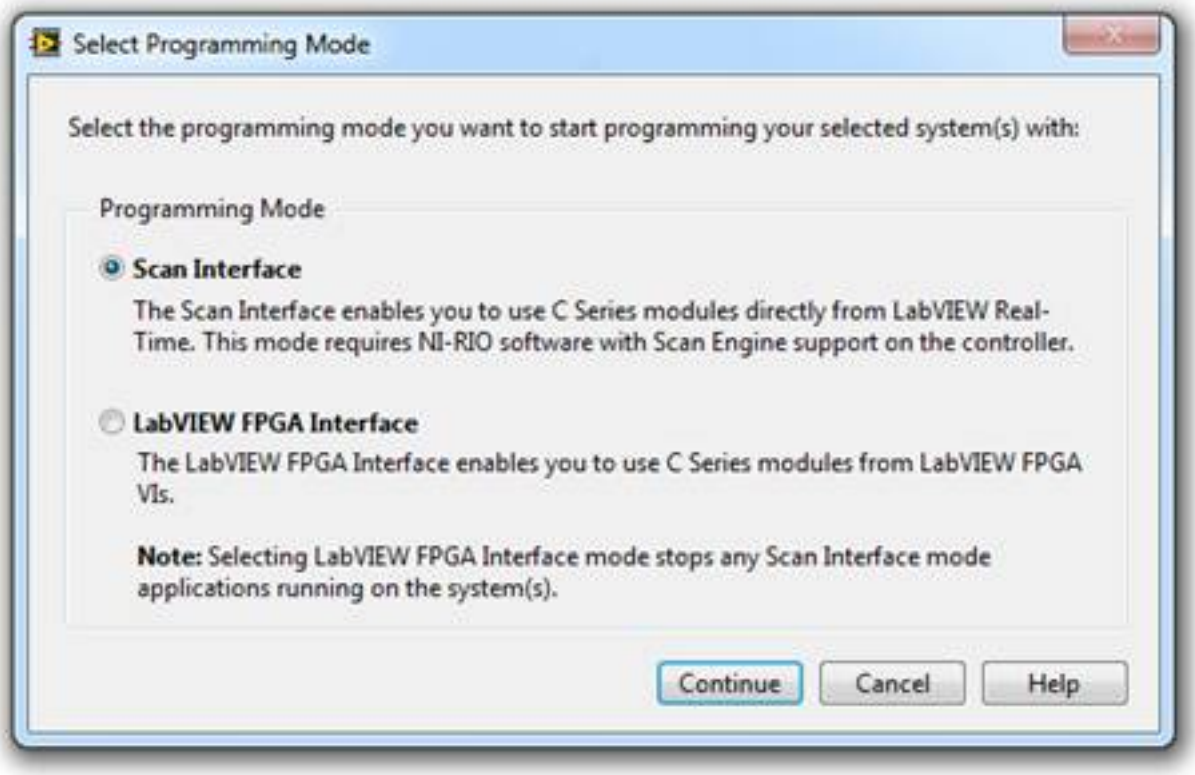

Select the Appropriate Programming Model

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 you 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. The LabVIEW Help online discusses this process for scan mode and FPGA mode.

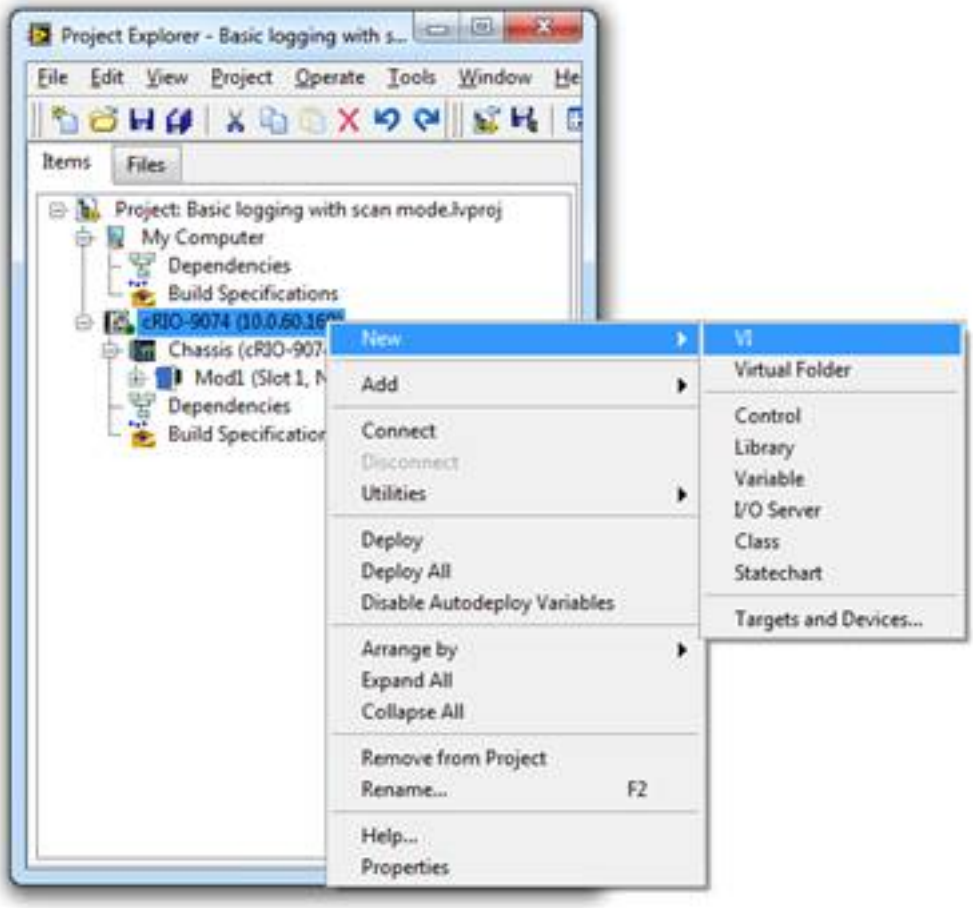
CompactRIO Scan Mode Tutorial

This section will walk you through creating a basic control application on CompactRIO using scan mode. If

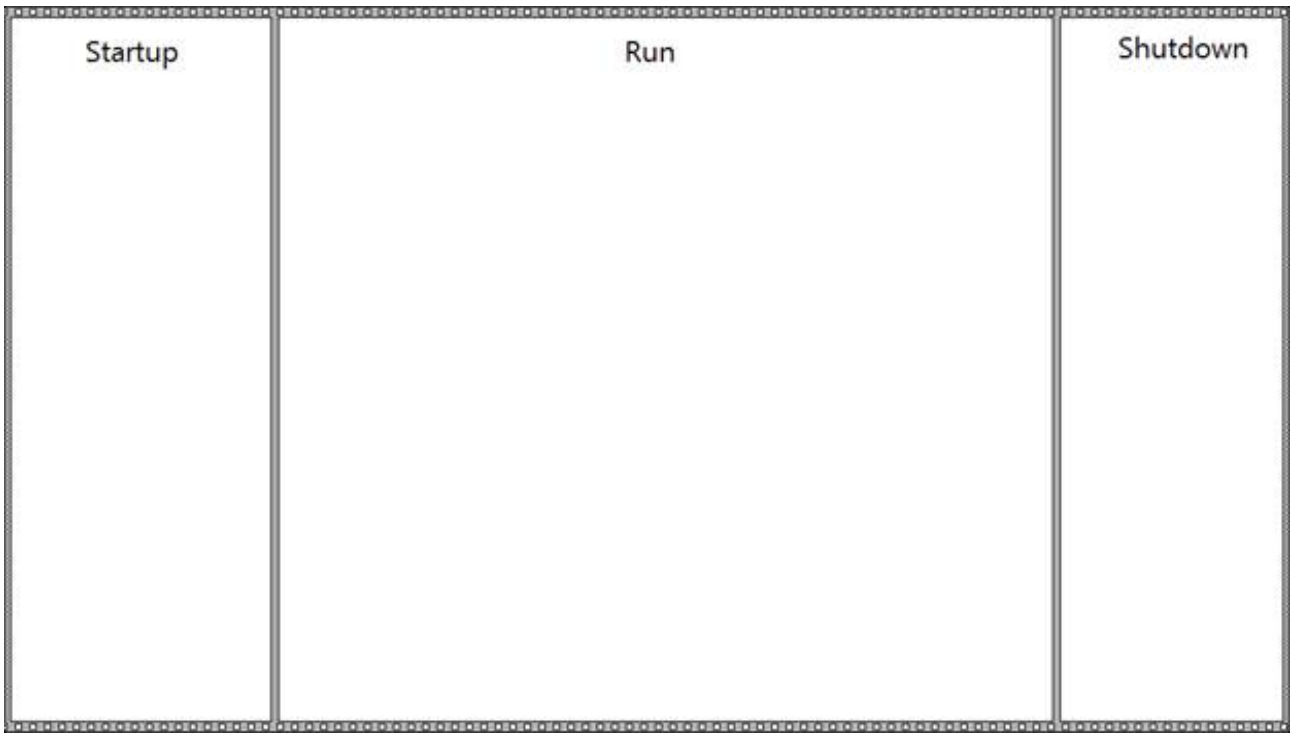
you chose to use the LabVIEW FPGA Interface, see the LabVIEW FPGA Tutorial below. You should now have a new LabVIEW Project that contains your CompactRIO system, including the controller, chassis, and C Series I/O modules. In this tutorial we will be using an NI 9211 Thermocouple input module; however, the process can be followed for any analog input module. You can also download the solution from here.

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

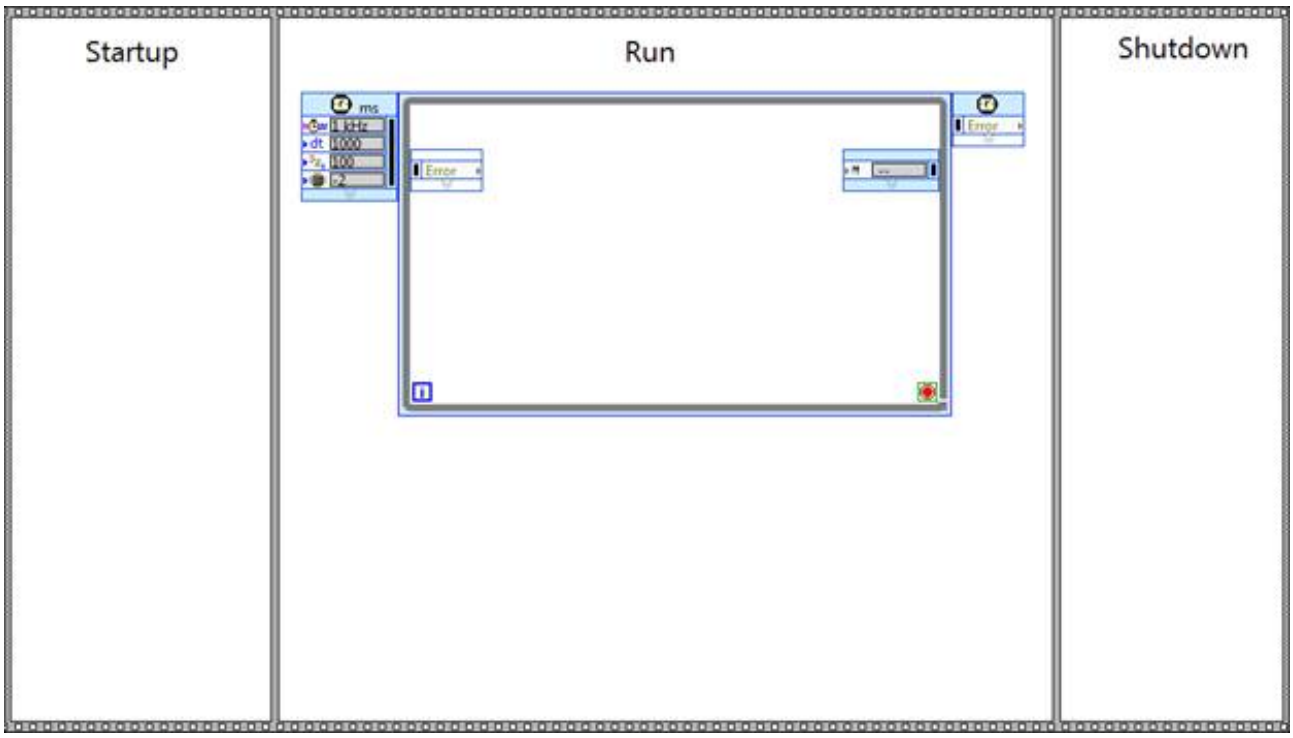
2. This project will only contain one VI, which is the LabVIEW Real-Time application that runs embedded on the CompactRIO controller. Create this the VI by right-clicking on the CompactRIO real-time controller in the project and selecting New»VI. Save the VI as RT.vi.



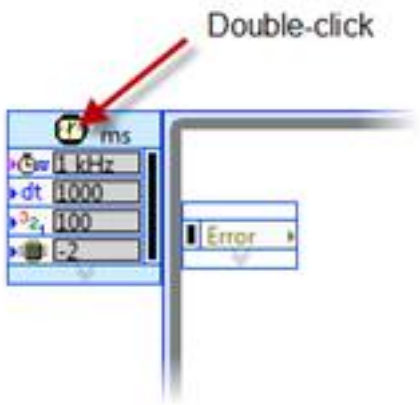

3. The basic operation of this application will include three routines: startup, run, and shutdown. A flat sequence structure is an easy way to enforce this order of operation. Place a flat sequence structure with three frames on your RT.vi block diagram as shown below.



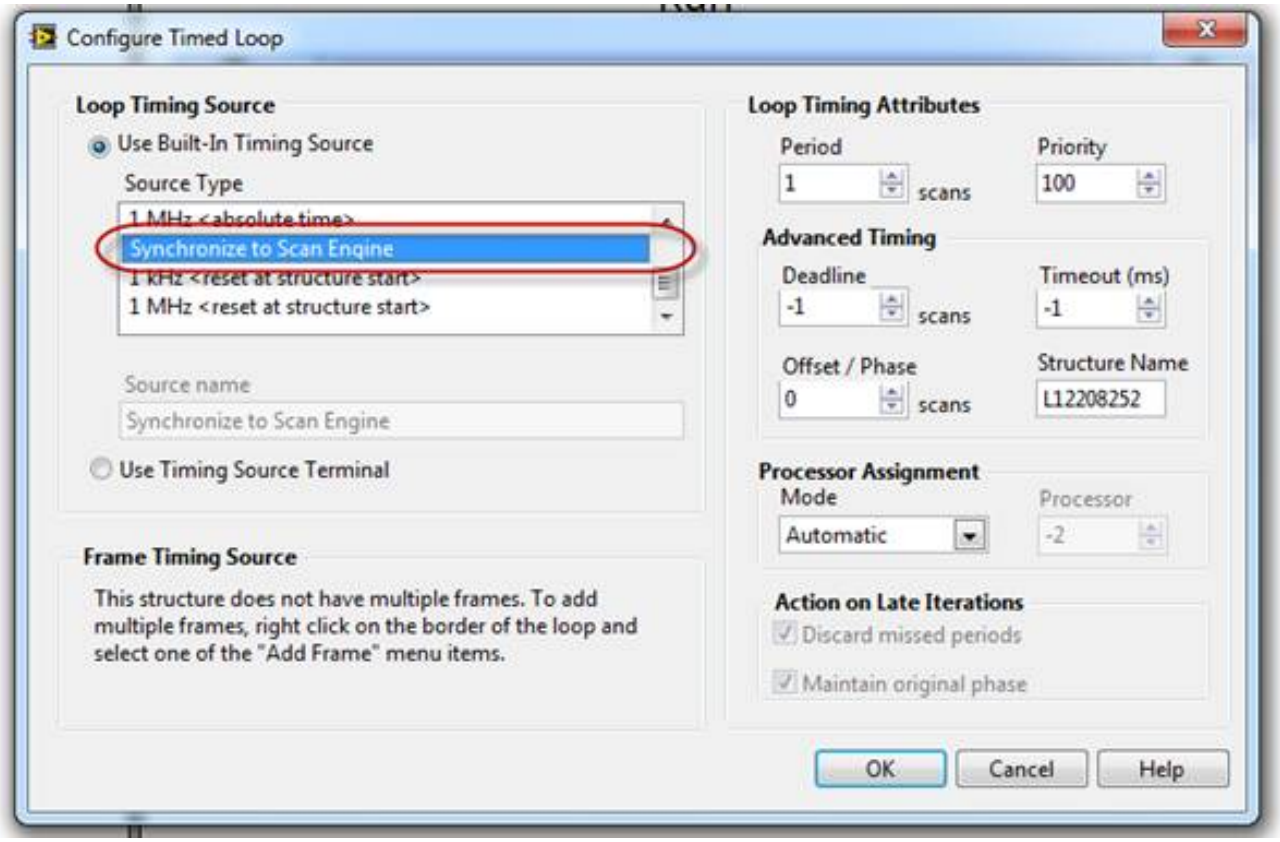

4. Now add a timed loop to the Run frame of the sequence structure. Timed loops provide the ability to synchronize code to various time basis, including the NI Scan Engine that reads and writes scan mode I/O.




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

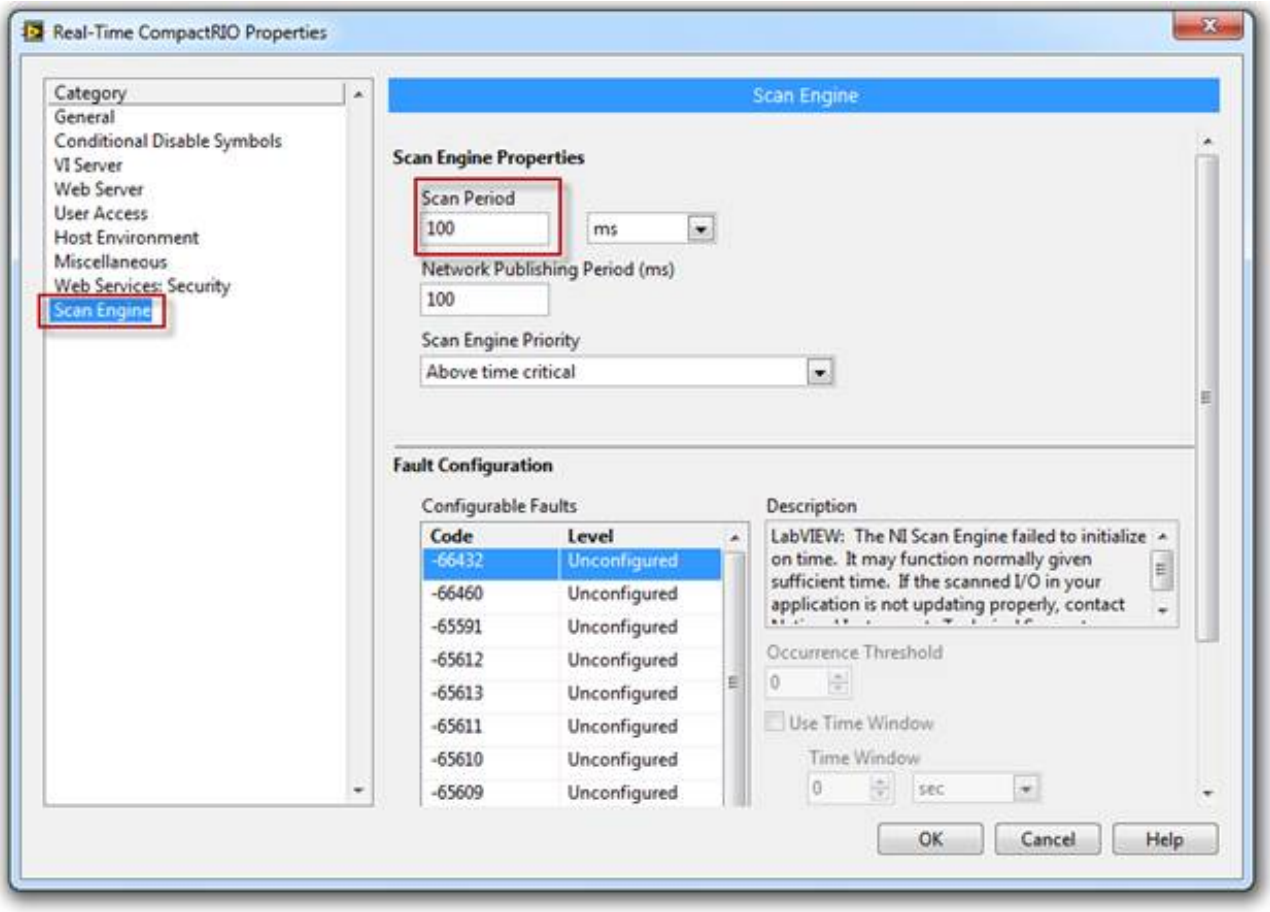



6. Select Synchronize to Scan Engine as the Loop Timing Source. Click OK. This will cause the code in the timed loop to execute once, immediately after each I/O scan, ensuring that any I/O values used in this timed loop are the most recent values.

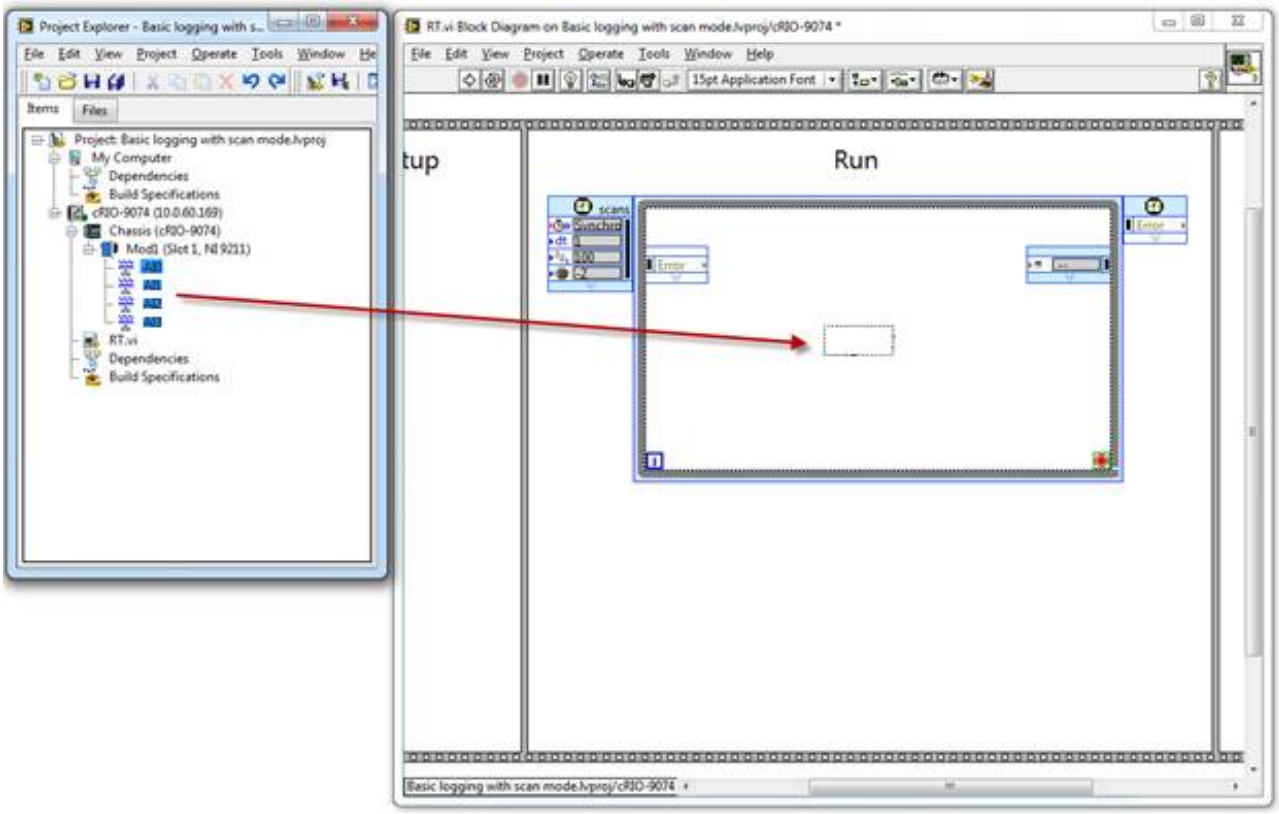



7. The previous step configured the timed loop to run synchronized to the scan engine. Now configure the rate of the scan engine itself by right-clicking on the CompactRIO real-time controller in the LabVIEW Project and selecting Properties.

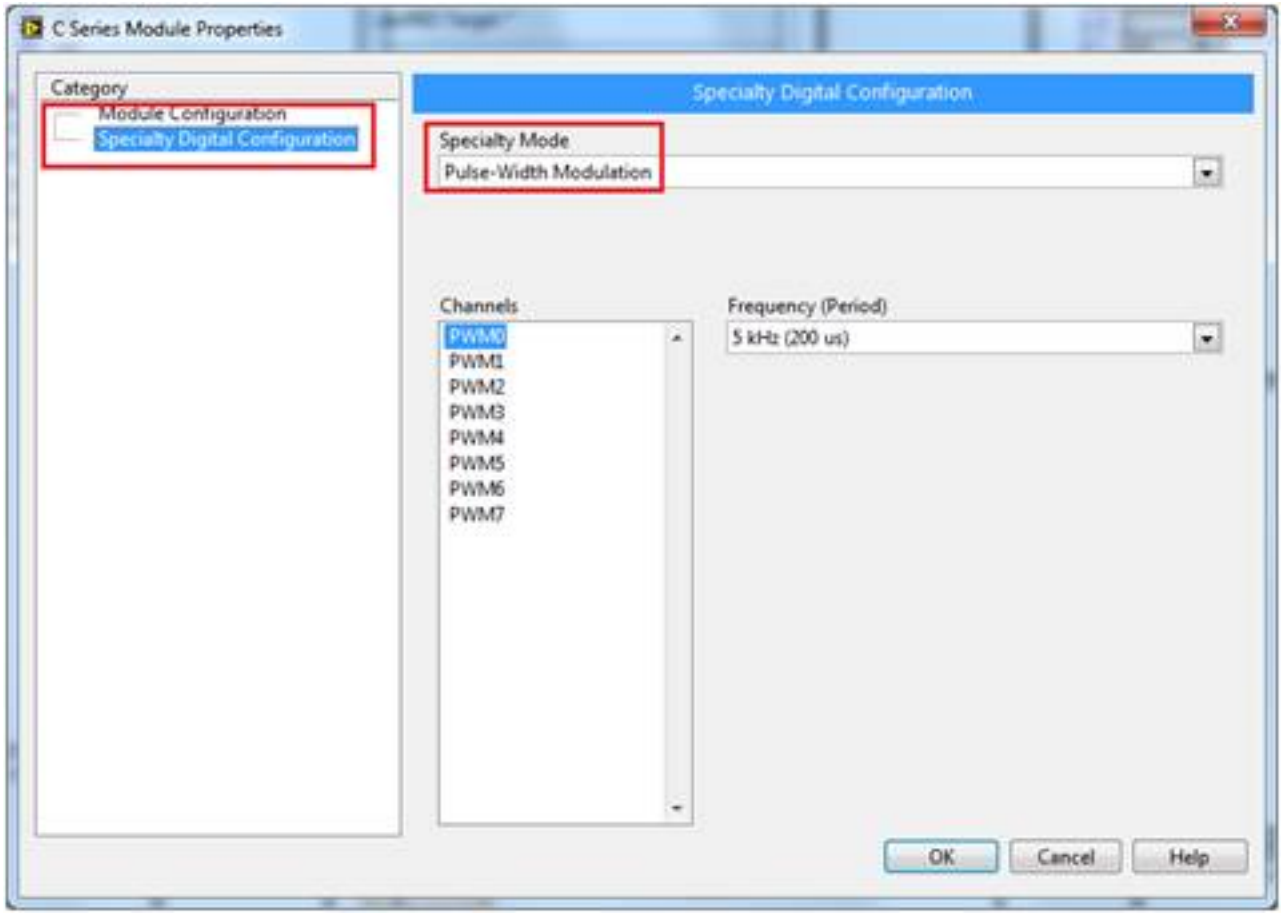
8. Select Scan Engine from the categories on the left and enter 100ms as the Scan Period. This will cause all of the I/O in the CompactRIO system to be updated every 100ms (10Hz). The Network Publishing Period can also be set from this page, which controls how often the I/O values are published to the 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 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 real-time controller, chassis, and the I/O module you would like to log. Select AI0 clicking on it then drag and drop it into the timed loop on your RT.vi diagram as shown below.

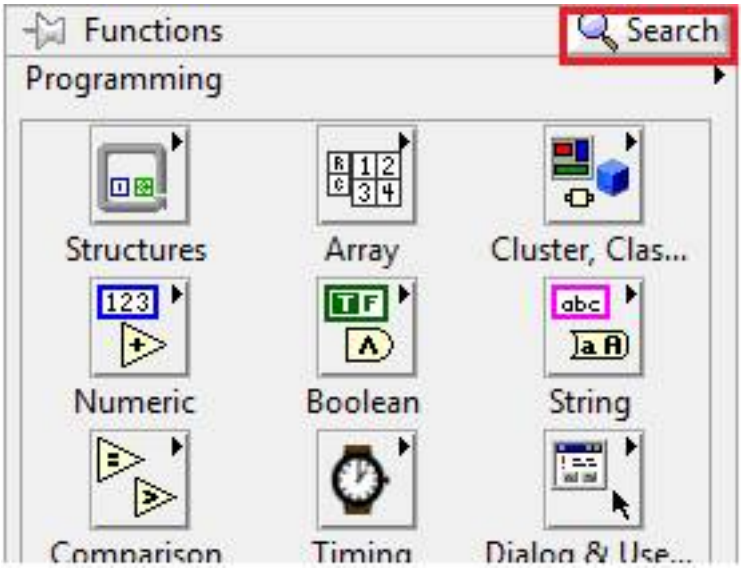



10. Now we will want to configure the digital module in our project for specialty digital Pulse Width Modulated output so we can use a PWM signal to control our imaginary heater unit. To do this right click your digital module in the project and select Properties. In the C Series Module Properties dialog select Specialty Digital Configuration and a Specialty Mode of Pulse-Width Modulation as shown in the image below. Specialty Digital mode allows your module to react to pattern based digital I/O at rates significantly faster than is available with the scan interface. Click OK and your module will now be in PWM mode.

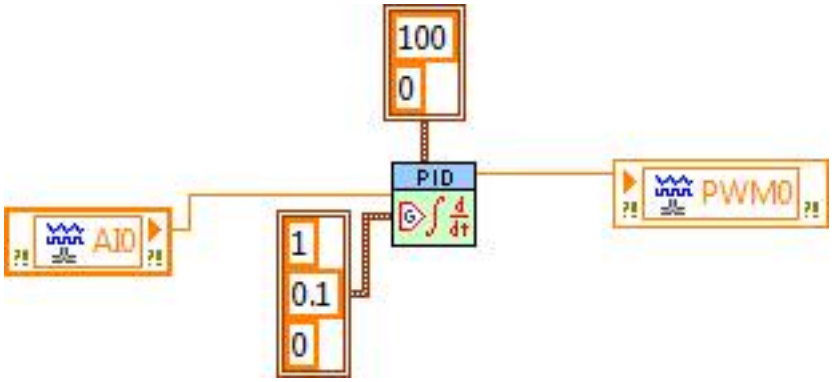



11. Now you are ready to add your PWM output to the block diagram. To do so, expand the Mod2 object in your project and drag and drop the PWM0 item to the block diagram as you did with the AI0 I/O node in the previous step.

12. Next we will want to add the PID control logic to our 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.



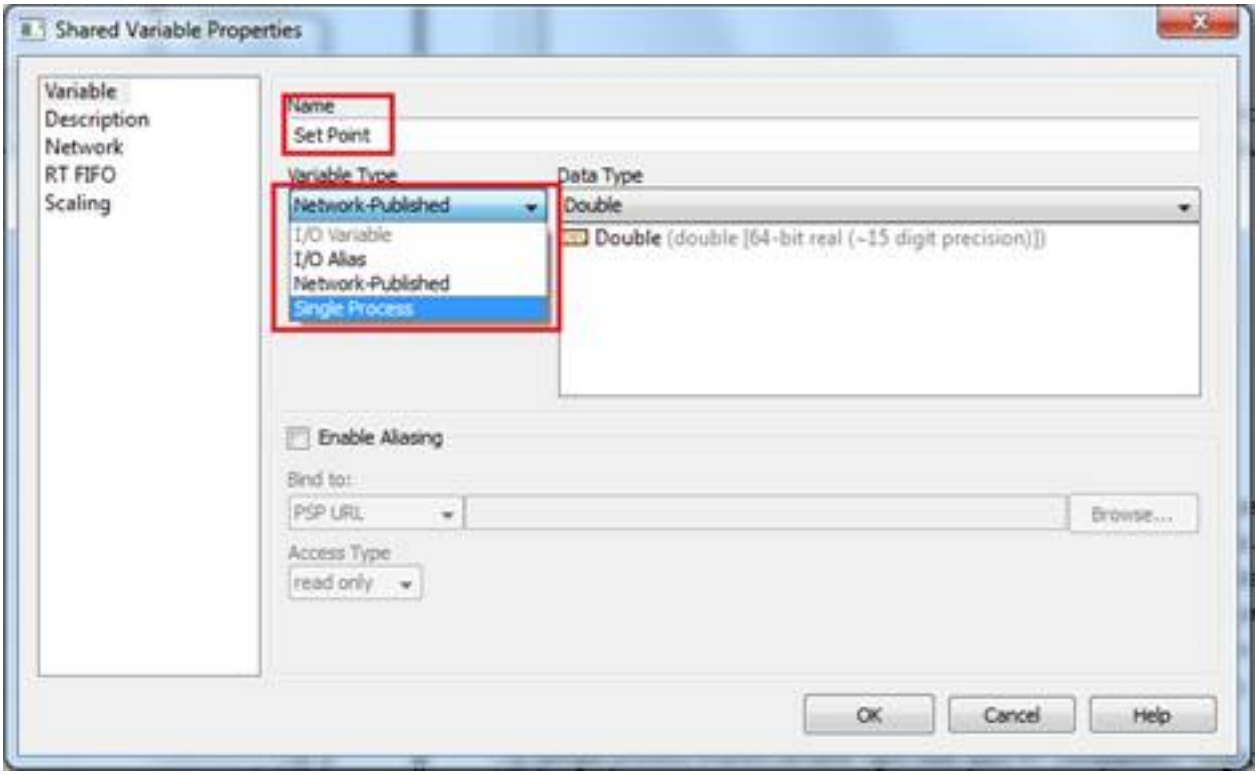

13. Search for PID and select PID.vi in the Control Design and Simulation Palette and drag it to your block diagram in the timed loop and wire the PID VI as shown in the image below.



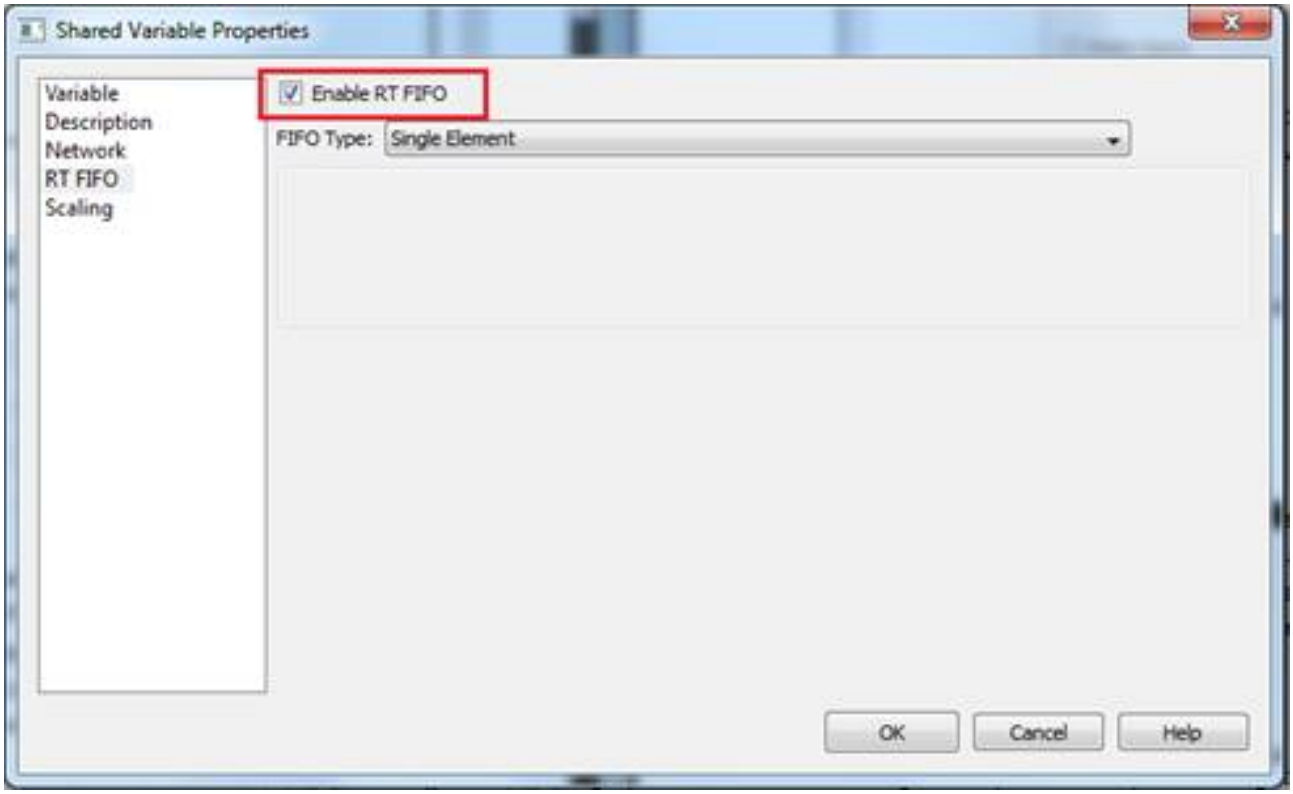

14. Notice that we did not wire the set point input at this time. We did this because it is best practice to keep user interface (UI) objects out of your high priority control loop. Since we may want to interact with and adjust our set point at run time, we will want to create a control we can interact with in our lower priority loop. We will also want to create single process shared variables for I/O in our high priority control loop. Since we need two controls in our application (set point and stop) we need to create two new single process shared variables.

To create a single process shared variable right click your RT CompactRIO Target in the LabVIEW Project and select New >> Library. Rename your library something intuitive like RTComm. Now right click your new library and select New>>Variable. This will open the Shared Variable Properties dialogue. Name your variable Set Point (or whatever you want actually), and select “Single Process” for the variable type in the

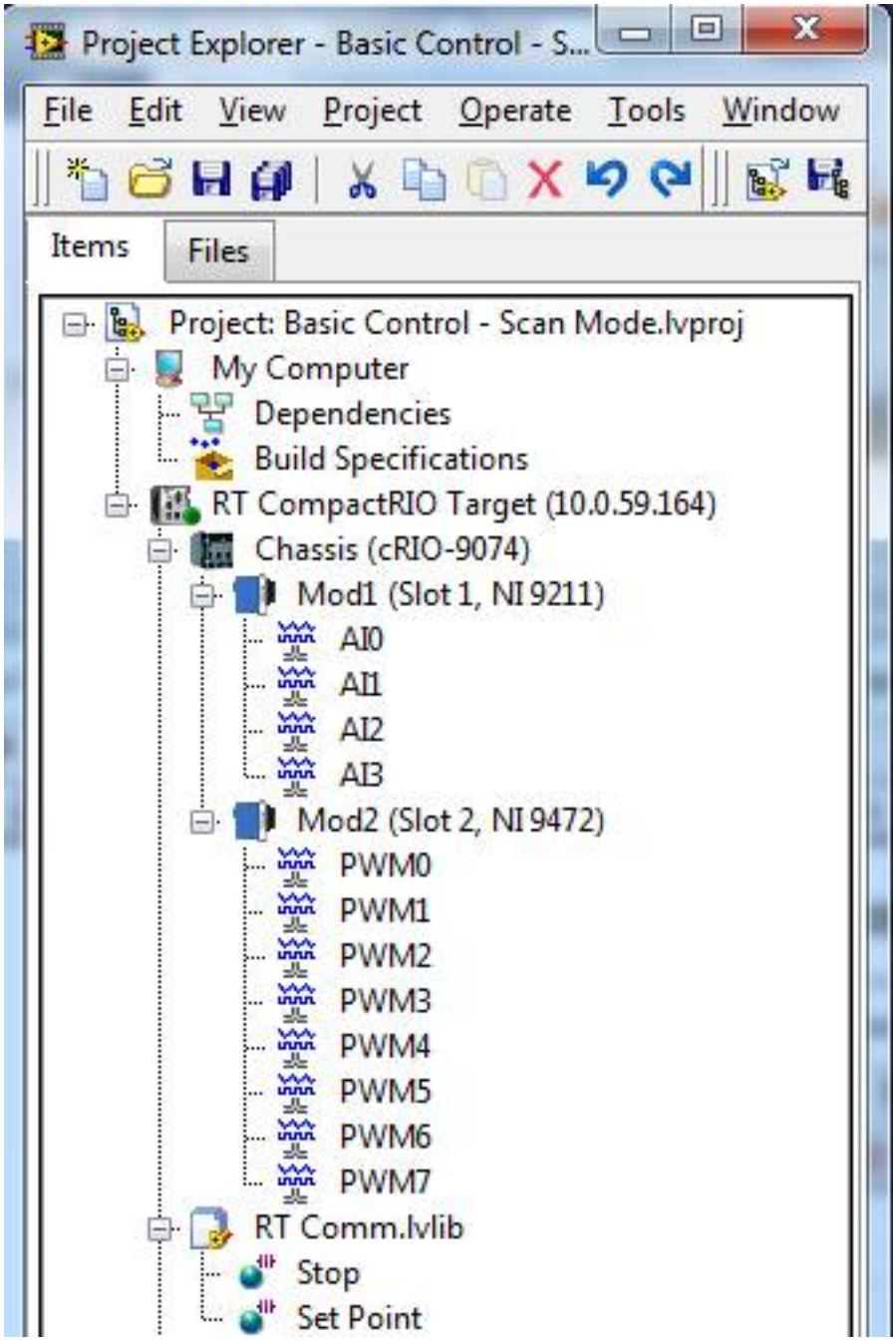
Variable Type drop down box as shown in the image below.  Leave the Data Type as Double.



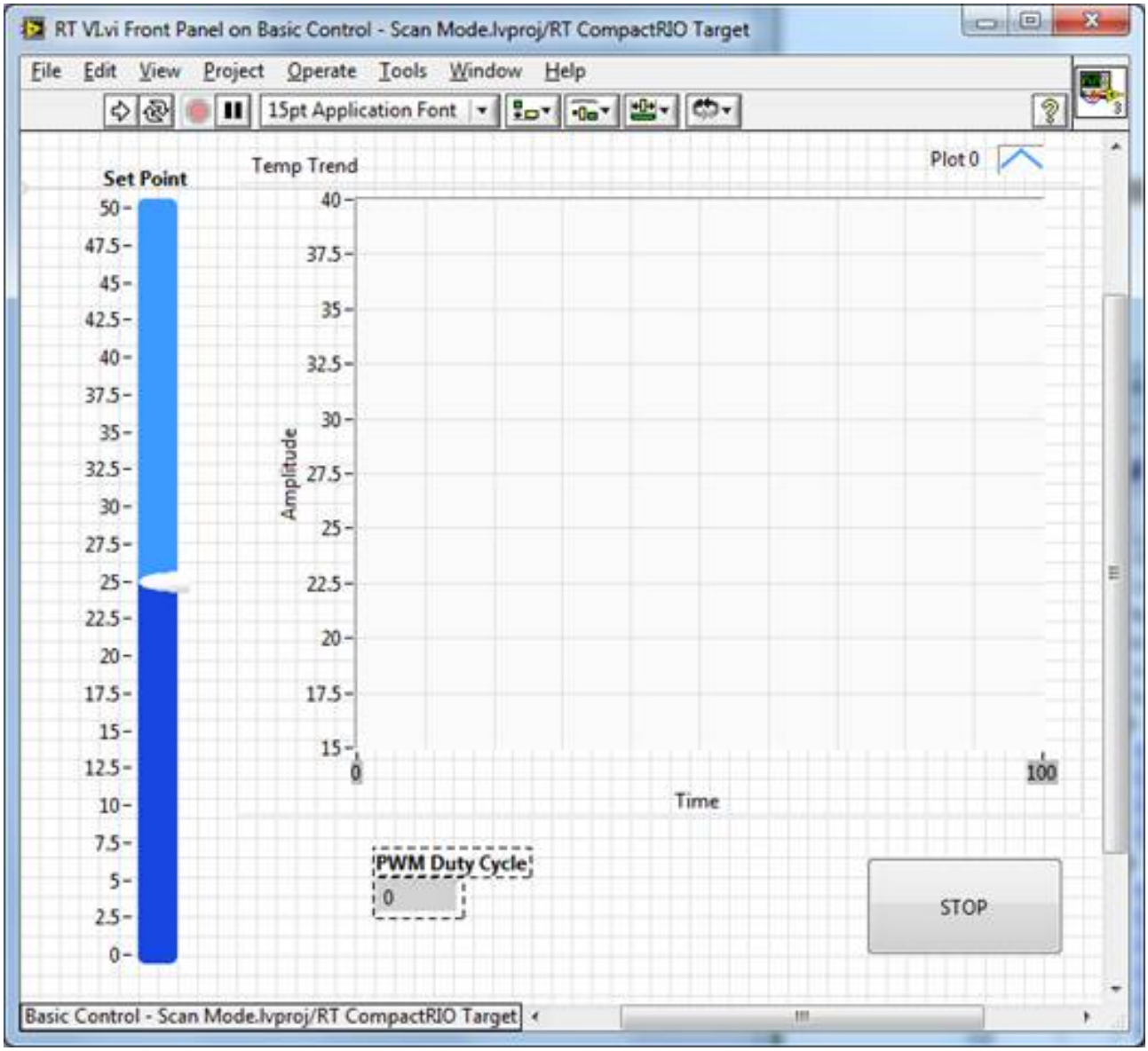

Finally click on the RT FIFO option in the left hand tree and click the Enable RT FIFO check box.



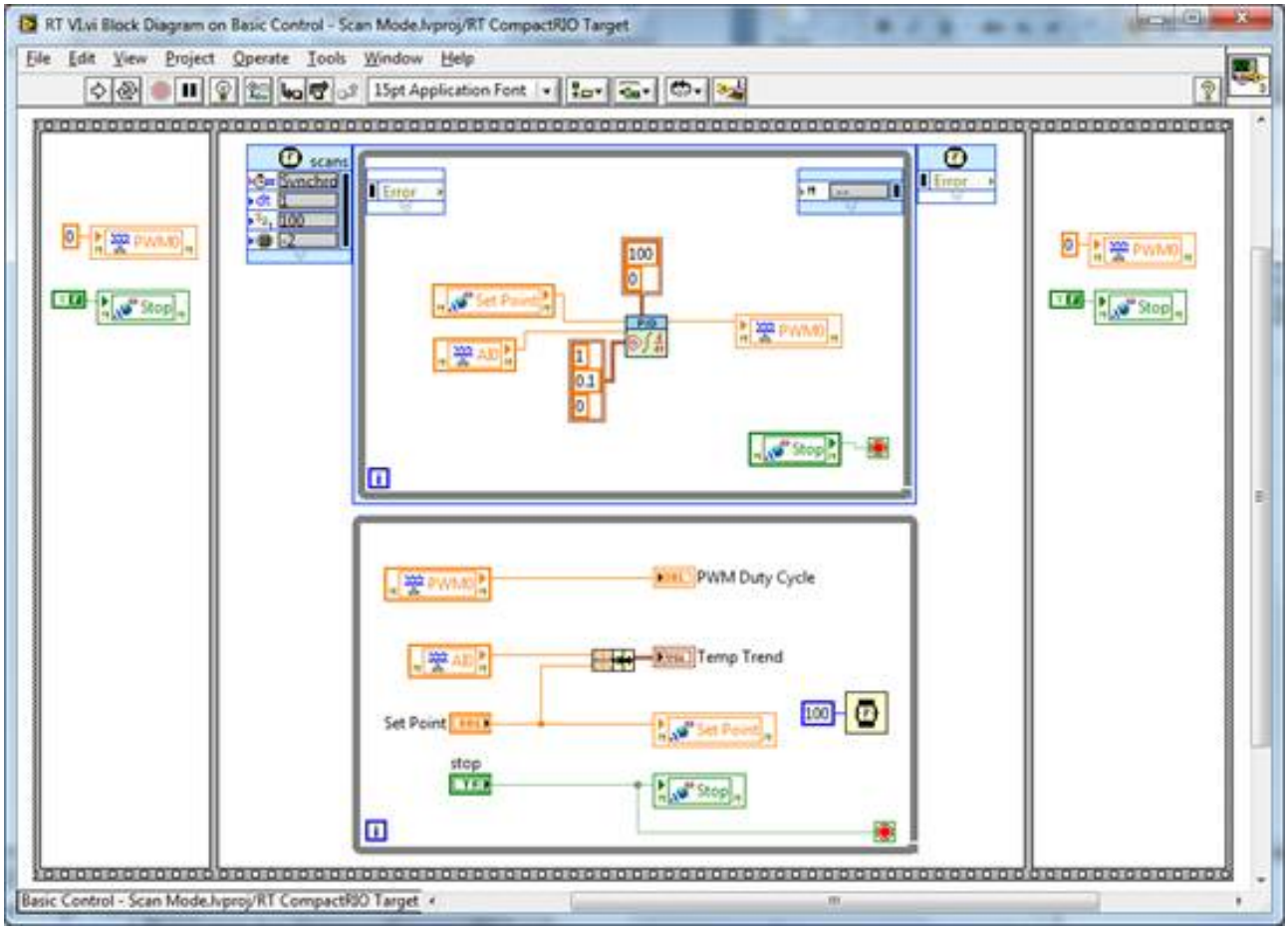

15. Create another single process shared variable in the library you just created. This variable is for the Stop control we are going to create that will stop our program when we are ready. This new variable should have all the same settings as the previous Set Point variable with the exception of the type, which should be Boolean. Once finished, your project should look similar to the image below.




16. Now we will create our user interface. To do so add a Slide control, Waveform Chart, Numeric control, and Stop (Boolean) control as shown in the image below.




17. Now we will finish wiring our program. Create a secondary (non timed) loop for your UI objects and finish wiring your block diagram as shown in the image below.




18. Note the addition of I/O to the configuration and shutdown states to ensure your I/O is in a known state when your program begins and ends.Your basic control application is now ready to run.

## View Additional CompactRIO Development Resources

To continue learning, check out the additional resources on the CompactRIO Setup and Services page.