MOIS Overview

The Modular Ocean Instrumentation System (MOIS) software was developed in the LabVIEW programming language. The software runs on a LabVIEW real-time target. The National Instruments produced real-time target MOIS uses is called a CompactRIO (Compact Real-Time Input/Output) or is abbreviated as cRIO. The MOIS software acquires analog measurements as continuous waveform data and serial data as ASCII text data from attached instruments. The software logs both types of data to disk on the real-time controller in a single data file. The data files are an hour in length and the hourly files are archived once a day in a single daily zip file. The MOIS software was created with the LabVIEW FPGA Module, LabVIEW RT Module, and LabVIEW Professional development products.

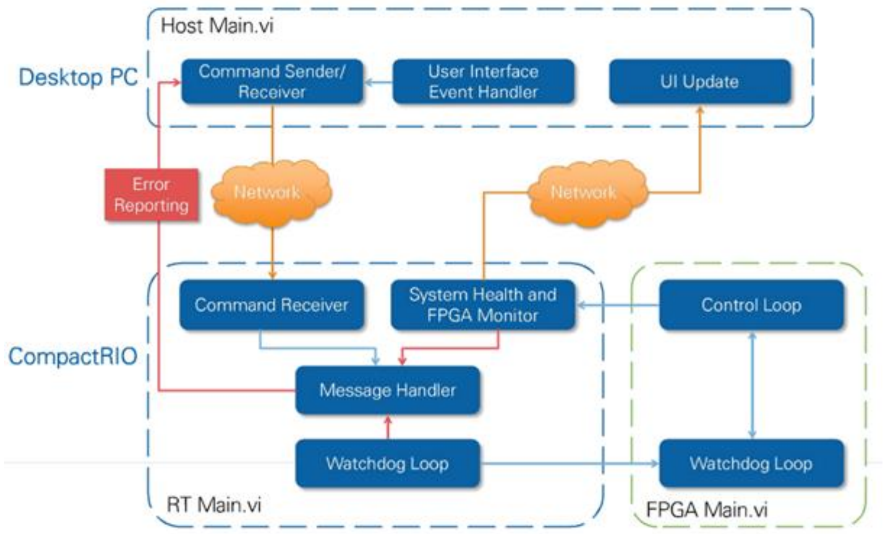
A generic compiled MOIS executable file is installed on the CompactRIO (cRIO-9068) as shipped. For basic data logging no reprogramming is required. There are many instances where it will be desirable for the software to be customized for specific test requirements. For information on the basic operation of MOIS refer to the MOIS User's Manual.

This developer's guide is intended to point the user in the correct direction for enhancing the software to meet specific needs of an individual test. The MOIS software is developed on using an advanced LabVIEW architecture that can be most readily developed further by an experienced LabVIEW developer. New to LabVIEW developers will need more time with the code, and LabVIEW in general, in order to come up to speed with MOIS software development. This guide will attempt to explain the overall architecture where developers can then go to sections of code that may be expanded upon to modify or add functionality. This document is not intended to be a “how to” for programming in LabVIEW.

1. Developer Walkthrough

The MOIS software architecture is adapted from the LabVIEW Waveform Acquisition and Logging on CompactRIO Sample Project that is included with LabVIEW since 2012. This queued message handler (QMH) sample project can serve as a basis of understanding of the MOIS software operation. While the sample project uses a single c-series module microphone module, the MOIS software acquires measurements from five analog input c-series modules and one RS-232 serial c-series module. The software/hardware system diagram is shown below. The software runs on three different computer hardware components: desktop PC, real-time controller, and a FPGA. The Compact RIO contains both the real-time controller and the FPGA. The desktop PC is needed to build a new configuration file and can be used to view data and system status of the cRIO. The analog channel acquisition rate is set in the Configuration.xml file that is generated from the Configuration File Generator.vi. This is the config file that appears in the root directory of the cRIO-9068 controller. The desktop PC is not needed for normal operations where MOIS is recording measurements and serial instrument data.

The diagram in Figure 1 below details the data flow of the in an abstracted manner. The diagram is not specific to the MOIS software but rather CompactRIO systems in general. The three basic pieces are the FPGA, Real-Time (RT) operating system, and an optional user interface (UI) that runs on a PC.



2. Basic Features

Headless operation with optional user interface:

- The user interface VI interacts with the CompactRIO device and displays data. This VI can connect and disconnect from the device at any time without affecting the acquisition and logging loop.

Continuous or triggered (from UI) data logging:

The real-time VI logs acquired data to disk as TDMS files continuously or, optionally, when a trigger condition is met. The data is then written to a data file.

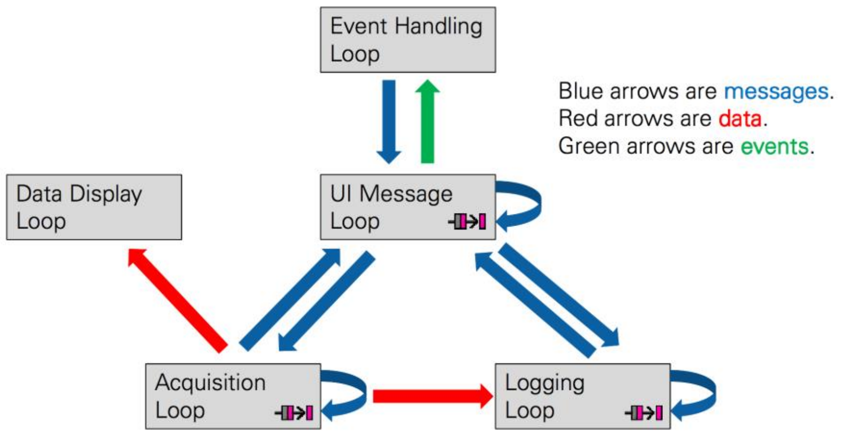
- A new file is created every hour.
- MOIS manages the amount of disk space being used and stops the application if disk space is critically low.

Error handling:

- The application reports and logs all errors from the CompactRIO device, shutting down on any critical error.

3. Overall Architecture

The MOIS LabVIEW software project is based on the Simple State Machine and Queued Message Handler architectures. To learn more about these architectures from within the LabVIEW development environment, refer to the Simple State Machine and Queued Message Handler templates and their documentation, available from the Create Project dialog box, for information about how these templates work or search on ni.com. The diagram below shows the basic architecture of a queued message handler. The architecture allows for a user interface with some control of a real time controller from a desktop PC while being very efficient in terms of processor resources on the real time controller. The architecture lends itself to adding functionality as the software is improved over time. For instance, error handling and logging can be expanded for new error conditions. Additionally, items can be added to the user interface when required.



The MOIS software project is a complete application with based on the QMH architecture. The software has three message handlers (MHL): the primary MHL, an acquisition MHL, and a logging MHL.

Primary Message Handler (MHL)

The primary message handler is handles interaction with the user interface (UI) and logs errors. This MHL runs the MOIS User Interface software.

4. Required Development Software and Hardware

In order to modify and recompile the MOIS software it is necessary to have the LabVIEW development environment installed on a development PC. The following LabVIEW licenses and modules are necessary to program the cRIO-9068 controller. It is possible to download all of the required LabVIEW software for a 30-day evaluation when the expense of software licenses is prohibitive.

Required hardware needed to run the MOIS software:

- NI Compact RIO using ARM processor and RT Linux
- NI C Series modules (in slots as listed below):
  1) NI-9467 GPS Module
  2) NI-9239 +/-10V Analog Input Module

3) NI-9239 +/-10V Analog Input Module
4) NI-9219 Universal Analog Input Module
5) NI-9237 Full / Half Bridge Strain Gage Module
6) NI-9237 Full / Half Bridge Strain Gage Module
7) NI-9870 RS-232 Serial Module
8) NI-9485 Solid State Relay Module

Software Development System (installed on development PC):

- All versions below are 2014 or higher
- LabVIEW Full or Professional Development System
- LabVIEW FPGA Module
- LabVIEW Real-Time Module
- NI-RIO device driver software
- NI CompactRIO Device in FPGA Interface Mode

CompactRIO installed components:

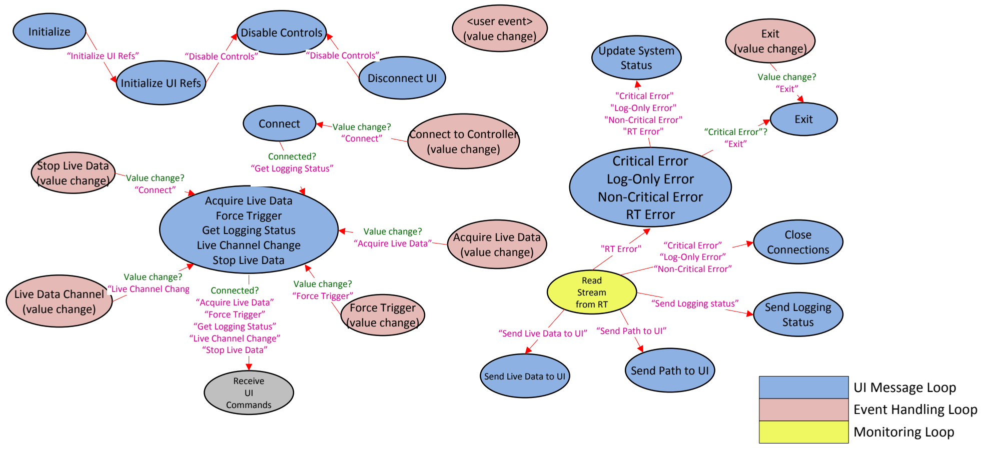
These are installed through National Instruments Measurement and Automation Explorer (MAX) program. If the software modules listed above are installed on the development PC the following software will be available to load on the cRIO using MAX.

- LabVIEW Real-Time
- Network Streams
- Network Variable Engine
- NI System Configuration
- NI-RIO
- NI-Watchdog

5. Overview

This sample project consists of nine parallel loops across three execution targets. The diagram in Figure 1 shows how information is passed between the FPGA, RT, and PC environments. The following loops run in parallel on the desktop computer:

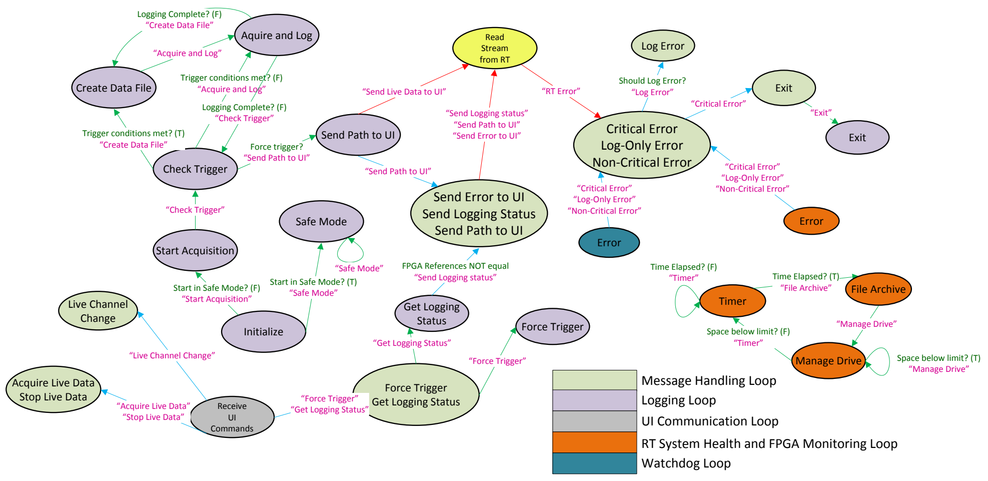
- Handling events from the user interface (UI Main.vi - Event Handling Loop)—Produces messages to the UI Message Loop based on front panel events.
- Handling messages from the user interface and the real-time controller (UI Main.vi - UI Message Loop)—Receives and responds to messages from the Event Handling Loop and, using network streams, the RT Message Handling Loop.
- Displaying messages and data from the CompactRIO device (UI Main.vi - Monitoring Loop)—Displays the latest values of information acquired from RT Loop - System Health and FPGA Monitoring.vi.



The diagram shows message flow and state machine states for the software that runs in the UI. The drawing shows the states that occur during the operation of the software and arrows represent the message queue. The message strings and Boolean operators that imitate the state changes are shown on the arrows. Don't be overly concerned about the complexity of the message passing and flow at this point. The diagram is useful in learning how data is passed and for debugging purposes. The complete diagram is attached at the end of the document.

The following loops run in parallel on the real-time controller:

- Handling commands from the user interface (RT Loop - UI Commands.vi)—Reads commands that are sent from UI Main.vi on the development computer and produces the appropriate messages.
- Handling messages from all loops on the real-time controller (RT Main.vi - Message Handling Loop)—Consumes messages from all loops that run on the real-time controller.
- Ensuring the RT controller remains responsive (RT Loop - Watchdog.vi)—Pets the watchdog, ensuring the RT controller remains responsive.
- Monitoring diagnostic information from the real-time controller and data from the FPGA (RT Loop - System Health and FPGA Monitoring.vi)—Monitors CPU and memory usage of the real-time controller and data from the FPGA VI. This information is written to network-published shared variables and appears on the System Monitoring tab of UI Main.vi.
- Acquiring and logging data from the FPGA (RT Loop - Acquisition and Logging.vi)—Runs FPGA Main.vi, reads data from the DMA FIFO, and logs this data to disk. The analog data is written to the DMA FIFO with each acquisition. When it is determined that a second of data loaded into the FIFO the RT system reads this block of data. The data block contains all channel measurements and has time stamp data added to the FIFO. The time stamp is read and then used as a t0 and the data is converted to a LabVIEW waveform. The delta t for the waveform is determined from the sampling rate. The data is then logged to disk. A new data file is started every hour. This is done by adding an hourly "trigger" to the check trigger state.



- Serial
- Update System Time

The serial port acquisition and the update of the real time clock system time run parallel to rest of the software. They are not integrated into the QMH architecture at the time of writing this guide. The serial acquisition software uses FPGA FIFOs to pass the serial data and time stamp information from the FPGA to the RT system. The serial data is then written to the data file by using the reference to the current data file.

The update system time software is used to update the RT clock to the FPGA GPS disciplined clock. The RT clock is used for the time and date in the naming of the data files. The actual data timestamps are passed up from the FPGA with the data.

The following loop runs on the FPGA:

- Acquiring data (FPGA Main.vi - Acquisition Loop)—Acquires data from the C Series module inputs and writes this data to a DMA FIFO.
- Serial
- FPGA Timekeeper—

The FPGA code does not contain a state machine but has several components that run in parallel on the FPGA. Refer to the block diagram of FPGA9068.vi to see the code. The code is commented to explain the operation of the software.

The analog data is written to the DMA FIFO with each acquisition. When the RT codes determines that a second worth of data from all channels is loaded into the FIFO the RT system reads this block of data

The serial acquisition software uses a separate while loop for each serial port on the NI-9870 RS-232 module. The data flow from the FPGA to the RT is handled via FPGA interrupt requests (IRQ).

MOIS User Interface (UI)

The UI runs on a host PC. The purpose of the user interface UI is only to monitor the system and individual data channels from the RT system. For deployments in the field the UI is normally used for pre deployment verification of channel data and operational state of MOIS. The MOIS UI may be used any time a network connection is available from the MOIS system to a UI host machine. The requested data passed between the RT system and UI are communicated via LabVIEW network streams. LabVIEW network streams are an Ethernet based communication method designed and optimized for lossless, high throughput data communication.

MOIS Real-Time (RT)

The RT code runs on the CompactRIO. The RT code reads measurement and timestamp data from the FPGA and writes it to the data file. The RT code monitors the communication between the FPGA and RT system and logs errors to the log file for supported errors. The RT system periodically receives updates of current time of day from the FPGA system time and updates the RT system clock. The RT system also includes a watchdog function that checks to ensure that the system is RT system is operating at via an inactivity watchdog. The inactivity watchdog uses a continuously incrementing counter. When this counter reaches some terminal count, it executes the expiration action which causes the RT software to log an error to log.txt.

MOIS Field Programmable Gate Array (FPGA)

The FPGA code is run on the FPGA built into the CompactRIO system. The FPGA code looks much like any other LabVIEW code but is compiled by a separate compiler that generates a FPGA bitfile. The FPGA acts as the interface between the I/O modules and the RT system. All of the analog and serial module measurements are read into the FPGA and then passed up to the RT system via FIFO buffers or I/O nodes. The FPGA also reads the pulse per second (PPS) timestamp from the GPS module in slot 1 and runs the FPGA timekeeper to generate a FPGA based system clock that is used for data time stamping.

6. Running MOIS from LabVIEW Development Environment

You can configure, run, and monitor MOIS entirely from executable files as outlined in the MOIS User's Guide, but it's necessary to run from source code in the LabVIEW developer environment to make any changes other than channel scaling and serial port configuration changes. To get started with MOIS in the LabVIEW development environment:

1. Adapt the MOIS software project to your hardware. The MOIS project is already configured to work with the cRIO-9068 controller and specific modules installed. The MOIS User's Manual identifies the installed I/O modules and measurement channels.
2.
3. In the Project Explorer window, open My Computer»Utility - Configuration File Generator.vi.

- Enter the configuration values that are appropriate for your application. Be sure to look at all the values in the TDMS Properties array.
- Sample rate
- Analog channel scaling
- Serial port configuration

4. Run the VI. LabVIEW generates a configuration file, Config.xml in the same directory as the .lvproj file.
5. Transfer this XML file to the root folder of the real-time controller. We use ssh and the client called puTTY that is available to download free online. The pscp.exe file is needed to perform a secure copy from the pc to the cRIO.

Run RT CompactRIO Target»RT Main.vi. This VI begins acquiring data and logging it using the configuration settings in Config.xml.

Open and run My Computer» MOIS UI Main.vi.

Enter the IP address of the MOIS CompactRIO device in the Controller Address text box and click Connect.

After you are connected, display live data in the waveform chart by clicking Acquire Live Data. Change the Live Data Channel control to see live data from different channels.

Click Exit to exit the application.

7. Adapting the MOIS Software Project to other NI Hardware

The FPGA VI in the MOIS LabVIEW project is compiled for specific FPGA and I/O hardware. If you would like to use a different FPGA or different C Series I/O modules, you must adapt the MOIS project to your hardware. The following steps refer to NI CompactRIO devices, but you also can adapt this software project to a NI Single-Board RIO device.

1. Ensure all devices are configured and connected to the same network as the development computer.
2. In the Project Explorer window, add or discover your RT CompactRIO target to the top-level project item.
3. Add or discover your CompactRIO chassis to the RT Compact RIO target you added in the previous step. Ensure the chassis is set to LabVIEW FPGA Interface mode.
4. Add or discover your FPGA target to the CompactRIO chassis you added in the previous step. When prompted to deploy settings, click Deploy Later.
5. Add or discover your C Series input module to the FPGA target you added in the previous step.
6. Drag the following project items from the default RT CompactRIO target to the one you added in step 2:
   - Error Handlers folder

- Globals folder
- Support VIs folder
- RT Loops folder
- Type Definitions folder
- Shared Variables.lvlib
- RT Main.vi

7. Drag the following project items from the default FPGA target to the one you added in step 4:
   - Support VIs folder
   - Type Definitions folder
   - FPGA Main.vi
8. Delete the default RT CompactRIO Target project item that no longer has any VIs associated with it.
9. Re-establish the link between RT Main.vi and the bitfile:
   - Open Start Acquisition and Logging.vi, located in the Support VIs folder on the RT CompactRIO Target.
   - Drag FPGA Main.vi from the Project Explorer window to the Open FPGA VI Reference VI function.
   - Open FPGA Main.vi and ensure the FPGA I/O Node uses the input channels you want. For example, you may want the FPGA I/O Node to read from Mod2/AI5 instead of Mod1/AI2. The actual channels you read from depend on your application.
   - By default, the FPGA I/O Node in this VI reads from Mod1/AI0–3.
10. You now can compile the FPGA VI and run it as part of your application.

Configuring MOIS Project Settings

In the Project Explorer window, open My Computer»Globals»Global - Configuration Options.vi and configure the MOIS project settings.