LabVIEW Templates and Sample Projects

Overview

LabVIEW 2013 continues to expand the in-product templates and sample projects, which provide recommended starting points designed to ensure the quality and scalability of a system. All of the templates and sample projects are open-source and include extensive documentation designed to clearly indicate how the code works and the best practices for adding or modifying functionality. In addition to demonstrating recommended architectures, these projects also illustrate best practices for documenting and organizing code.

You can create a new LabVIEW project from an existing template or sample project by selecting “Create Project” from the Getting Started window. The list of available options depends on the software modules and drivers that you have installed. This document lists the templates and sample projects that National Instruments provides and supports.

Table of Contents

1. Templates
2. Desktop Sample Projects
3. LabVIEW Real-Time and LabVIEW FPGA Sample Projects
4. Adding Custom Templates and Sample Projects
5. Getting Started with a Template or Sample Project

1. Templates

Templates demonstrate the fundamental building blocks of most LabVIEW applications. One or more templates are often used in combination to build real-world systems. These templates provide common architectures using well-adopted design patterns that you can modify to build a system.

Simple State Machine

This template facilitates defining the execution sequence for sections of code. This particular implementation often is referred to as a Moore machine, which determines the next state based on decisions made in the current state. The design of this template makes it easy to insert new sections of code, remove sections of code, or change the order in which sections execute – all without making major modifications to the structure of the application.

Read Documentation | View Webcast

Queued Message Handler

This template facilitates multiple sections of code running in parallel and sending data between them. Each section of code represents a process, such as acquiring data, and is designed similarly to a state machine. The separation of these components enables a responsive user interface and the ability to continuously log data while other messages are executing.

Read Documentation | View Webcast

Actor Framework

This is a template for creating LabVIEW applications that consist of multiple, independent tasks that need to communicate with each other. This framework was designed to address common development scenarios that can lead to significant duplication of code when extending functionality or adding processes. The Actor Framework is an advanced template that makes extensive use of LabVIEW Classes.

Read Documentation | View Webcast

2. Desktop Sample Projects

Desktop sample projects illustrate the use of one or more templates in an actual application. These projects fulfill the most common requirements of desktop-based measurement applications, including responsive user interfaces, asynchronous analysis, data-logging, user dialogs, error handling, and multiple independent tasks.

Finite Measurement

The Finite Measurement sample project acquires a single measurement and provides options for exporting the measurement to file. This sample project is designed as a state machine that can respond to user requests to configure and run a measurement*, perform analysis, display results, and log data to disk.

Read Documentation

*DAQmx 9.5.5 or later installs a second version of this sample project that uses the DAQmx API to configure and acquire an actual measurement with hardware I/O.

Continuous Measurement and Logging

The Continuous Measurement and Logging sample project acquires measurements continuously and logs them to disk. It executes five loops in parallel to ensure that various tasks can be executed at the same time while responding to and en-queuing user requests. These loops include event handling, UI messaging, acquisition*, logging, and display.

Read Documentation

*DAQmx 9.5.5 or later installs a second version of this sample project that uses the DAQmx API to configure and acquire an actual measurement with hardware I/O.

Feedback Evaporative Cooler

The Feedback Evaporative Cooler sample project is built using the Actor Framework template and implements an evaporative cooler with hot-swappable hardware, controllers, and user interfaces. The sample project is made up of multiple independently-running VIs, called actors, that represent the user interface, the cooler, its fans, and its water level. Each actor has the ability to pass command-like messages to, and receive such messages from, related actors. The architecture supports expansion for statically or dynamically including more actors.

Read Documentation

Supervisory Control and Data Acquisition System - New in LabVIEW 2013

This sample project implements Supervisory Control and Data Acquisition (SCADA) for a simulated NI CompactRIO and programmable logic controller (PLC)-based system. With a server and human machine interface (HMI) client architecture, it makes use of the LabVIEW Datalogging and Supervisory Control (DSC) Module for third-party connectivity and historical data and alarm logging in a server application. For the HMI client application, it implements event-based updates to the user interfaces and dynamic loading of subpanels to demonstrate an efficient and responsive user interface architecture.

Note: This sample project functionality requires the LabVIEW DSC Module.

Read Documentation | Read Detailed Guide on NI Community | View Webcast

3. LabVIEW Real-Time and LabVIEW FPGA Sample Projects

Embedded systems typically require an architecture that is designed for reliability and deterministic performance. As a result, many embedded applications require processes dedicated to system status monitoring, error handling and watchdog timers. The LabVIEW sample projects for CompactRIO and PXI RT DAQ illustrate best practices for addressing these needs by providing recommended software architectures that can be used for a variety of embedded control and monitoring systems. These sample projects also illustrate best practices for data communication, network connectivity, control routines, data logging, and more.
CompactRIO Sample Projects

**LabVIEW FPGA Control on CompactRIO**
This sample project is designed for applications that require high performance control and/or hardware-based safety logic. Rather than running the control algorithm in software, the control is implemented in the FPGA fabric, enabling your control loops to achieve rates faster than 10 kHz with minimal jitter. The FPGA VI also contains safety logic that immediately puts any outputs into a safe state upon a critical error or real-time software failure for maximum reliability.

[Read Documentation] [Read Detailed Guide on Ni Community] [View Webcast]

**LabVIEW Real-Time Control on CompactRIO (RIO Scan Interface)**
This sample project is designed for control applications that require deterministic performance with single-point I/O rates of 100 Hz or less. This sample project does not use the FPGA hardware, but leverages the deterministic, real-time processor for control. It uses the RIO Scan Interface (RSI) to access I/O data as variables within the real-time application.

[Read Documentation] [View Webcast]

**LabVIEW FPGA Waveform Acquisition and Logging on CompactRIO**
This sample project includes customizable high-speed FPGA based analog acquisition and logs the acquired data to disk on the real-time system when a trigger condition is met. This sample project is designed to run headless, or it can connect to the optional user interface that is provided.

[Read Documentation] [View Webcast]

**LabVIEW Real-Time Sequencer on CompactRIO - New in LabVIEW 2013**
This sample project implements a sequence engine that executes user-defined sequences or recipes in LabVIEW Real-Time for control applications. It includes a Windows-based user interface that generates user-defined sequences, deploys them to CompactRIO, and monitors the sequence engine status. The user interface uses dynamic loading of subpanels to demonstrate an efficient and responsive user interface architecture.

[Read Documentation] [Read Detailed Guide on Ni Community] [View Webcast]

Real-Time Sample Projects (with DAQmx)
The LabVIEW Real-Time (NI-DAQmx) sample projects are designed for applications involving real-time control and/or waveform acquisition and logging.

**LabVIEW Real-Time Control (NI-DAQmx)**
Implements deterministic, software-based control of a plant. This sample project uses NI-DAQmx. This sample project is designed for control applications that require deterministic performance of the DAQ control I/O, using NI Real-Time PXI controllers and NI DAQ.

[Read Documentation]

**LabVIEW Real-Time Waveform Acquisition and Logging (NI-DAQmx)**
Acquires continuous waveform data and logs it to disk. This sample project uses NI-DAQmx. This sample project includes analog data acquisition and logs the acquired data to disk on the real-time system when a trigger condition is met. This sample project is designed to run headless, or it can connect to the optional user interface that is provided. It is designed for applications using NI Real-Time PXI controllers with NI DAQ, or the NI Stand-alone CompactDAQ.

[Read Documentation]

4. Adding Custom Templates and Sample Projects
Advanced users can complement the included list of templates and sample projects with their own, making the new ‘create project’ an excellent mechanism to share and distribute recommended templates across a team of developers.

The Create Project dialog for a custom template or sample project can be changed to include additional items that can script the creation of custom code. For more information on how to create and add your own items to this dialog, click here.

5. Getting Started with a Template or Sample Project
Templates and sample projects require the use of a wide variety of LabVIEW skills and programming concepts. The documentation provided in each project enumerates the concepts that users need to be familiar with. In addition, the documentation on the block diagram clearly indicates where to add or modify code in blue colored comments.

You can also reference the LabVIEW Skills Guide to see a list of recommended skills and training resources.