# Getting Started with LabVIEW

*A guide to using LabVIEW with RTD Windows Drivers*

## User's Manual

SWM-640070002 Rev. A

**RTD Embedded Technologies, Inc.**
*AS9100 and ISO 9001 Certified*

**RTD Embedded Technologies, Inc.**
103 Innovation Boulevard
State College, PA 16803 USA
Telephone: 814-234-8087
Fax: 814-234-5218

www.rtd.com

sales@rtd.com
techsupport@rtd.com



# Revision History

| | |
|---|---|
| Rev A | Initial Release |

# Table of Contents


**1 Introduction** **5**

1.1 RTD and National Instruments™ LabVIEW ..... 5

1.2 RTD Software Overview ..... 5

**2 Packaging and Organization** **6**

2.1 File Hierarchy ..... 6
- 2.1.1 LabVIEW Example Programs 6
- 2.1.2 Sub-VIs 7
- 2.1.3 RTD's Dataflow outline 7

**3 Utilizing RTD Libraries in LabVIEW** **8**

3.1 RTD Library Layout ..... 8
- 3.1.1 Wrapper Functions 8

3.2 Accessing the RTD Library with the Call Library Function Node ..... 8
- 3.2.1 Configuring DLL Node: Function Inputs 9
- 3.2.2 Configuring DLL Node: Parameters Tab 9

**4 Resources** **11**

4.1 National Instruments™ LabVIEW ..... 11

**5 Limited Warranty** **12**


# Table of Figures


Figure 1: DM35218 Digital I/O Example Program ..... 5
Figure 2: LabVIEW files layout ..... 6
Figure 3: Several RTD Sub-VIs to increase readability ..... 7
Figure 4: Dataflow being controlled by DLL Path ..... 7
Figure 5: Wrapper Function to Open a DM35956 ..... 8
Figure 6: Example of a configured Call Library Function Node ..... 8
Figure 7: DLL Node Functions Tab ..... 9
Figure 8: Configuring array example ..... 10
Figure 9: Table of LabVIEW parameters ..... 10

# 1 Introduction

## 1.1 RTD and National Instruments™ LabVIEW

RTD is rolling out support for National Instruments™ LabVIEW to our embedded Windows drivers with the goal of giving our customers the ability to quickly and easily create graphical applications that can interact with our embedded hardware. For most RTD example programs in Windows, a corresponding LabVIEW virtual instrument (VI) will be provided to demonstrate how the fundamental board features can be controlled in LabVIEW.

Since LabVIEW can utilize the same RTD libraries as Windows executables, we are making some small additions to our Windows deliverables which will enable LabVIEW to fully access RTD's DLL. This leveraging of the RTD libraries ensures that all LabVIEW code is as simple as possible.

This document is a guide to creating an application in LabVIEW which utilizes the RTD libraries.

## 1.2 RTD Software Overview

LabVIEW Version: RTD uses LabVIEW 2018 to develop its Windows programs. Legacy versions may work as well depending on how recent they are.
Supported Windows Versions: LabVIEW 2018 is compatible with Windows 10 and Windows. To run LabVIEW on Windows XP or VIsta LabVIEW 2015 or older is required.
Supported Bitness: RTD creates all LabVIEW programs in 32-bit and 64-bit, meaning both are supported.

Generally, RTD board software is constructed in three layers:

1. The lowest level of software is the Driver. It interacts directly with the hardware of the board and must be installed to access the board.

2. Then, there is a library, consisting of different compartmentalized functions that interact with the driver, this source code is much more readable than the driver, and its functions cover every feature of the library's respective board.

3. The highest level of code RTD ships with its boards are Example Programs. These are programs that demonstrate how to use the library, and generally showcase the main functions of a board. The Example Programs are traditionally written in C, but in this case, the Example programs are created in LabVIEW. The Example Programs are meant to be a template that can be expanded upon to fully match the customer's needs.

The purpose of this guide is to explain how RTD sets up LabVIEW within the software package, and how LabVIEW can be used to interact with RTD Dynamic Link Libraries. Below is an image that displays the general layout of RTD's LabVIEW programs. The architecture of which will be broken down later in this document.

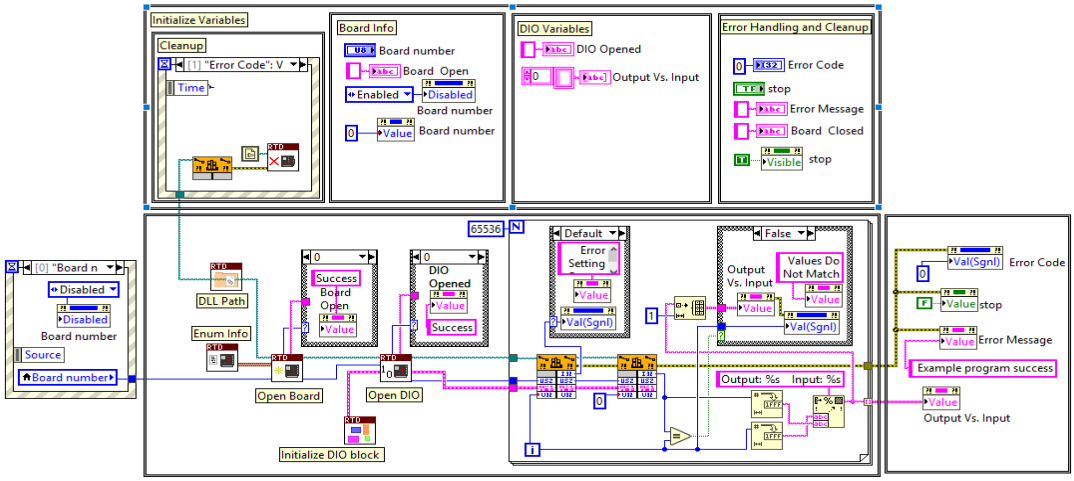


***Figure 1: DM35218 Digital I/O Example Program***

# 2 Packaging and Organization

## 2.1 File Hierarchy

### 2.1.1 LabVIEW Example Programs

RTD's software packages generally come with 4 sub-folders: Examples, Install/Driver, Include, and Library.

Within the Examples folder is where most LabVIEW support files are found. Within the LabVIEW folder, you can typically find four things. The first thing is a documents folder including a readme for the respective board, and this guide. Beyond that, there is a VI_x32 folder, containing the Example VIs for LabVIEW 32-bit, and a VI_x64 designed for LabVIEW 64-bit. These two folders also contain the DLLs for both Example Programs. Besides the necessary differences to make everything work on 32-bit and 64-bit respectively, all files in these two folders are the same. The final folder is named "Sub-VI's", and it holds any Sub-VI's that are used in the Main VI's. These are mostly used to condense code or to perform functions that may be needed multiple times.

Within the Main VI folders are precompiled DLL's for their respective programs. The source code for the DLL's can be found under the Library and Include folders. The DLL's were compiled using Visual Studio 2008, but later versions and other compilers should be able to create the DLL's as well.

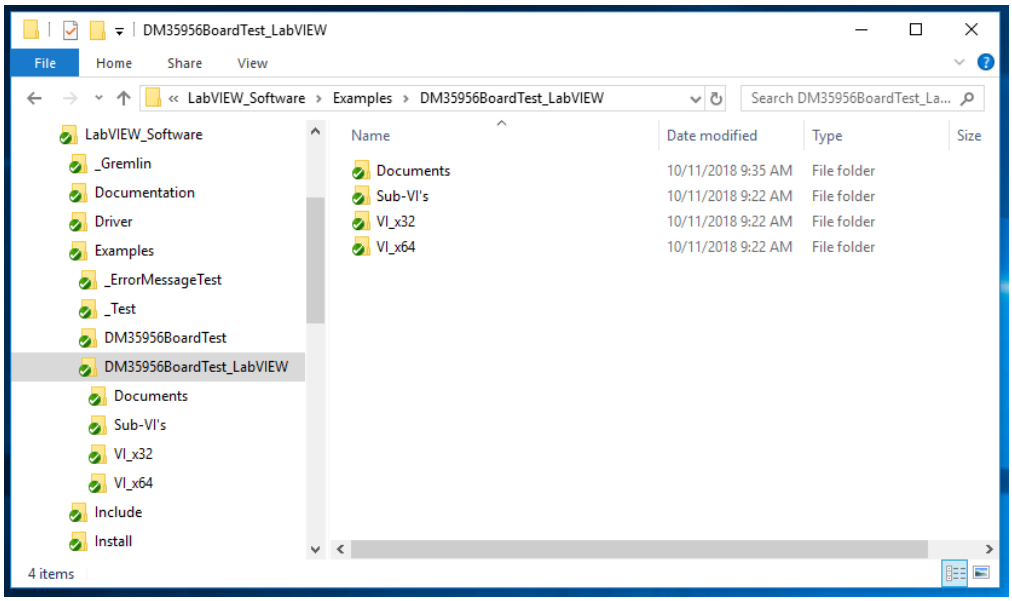


***Figure 2: LabVIEW files layout***

### 2.1.2 SUB-VIS

Each of RTD's example programs uses different Sub-VI's, which all follow a similar template. In general, RTD's Sub-VI's are used to perform tasks that will be used in any Example program, or to condense bits of code that will be used multiple times. Some examples of these include "Open_Board", "DLL_Path", and "Close_Program". Descriptions of each of these Sub-VI's can be found in the readme for their respective boards. Within the Example Program, the Sub-VI's can be identified by an RTD logo on top of a Sub-VI. The inputs and outputs in each Sub-VI are labeled within its front panel.

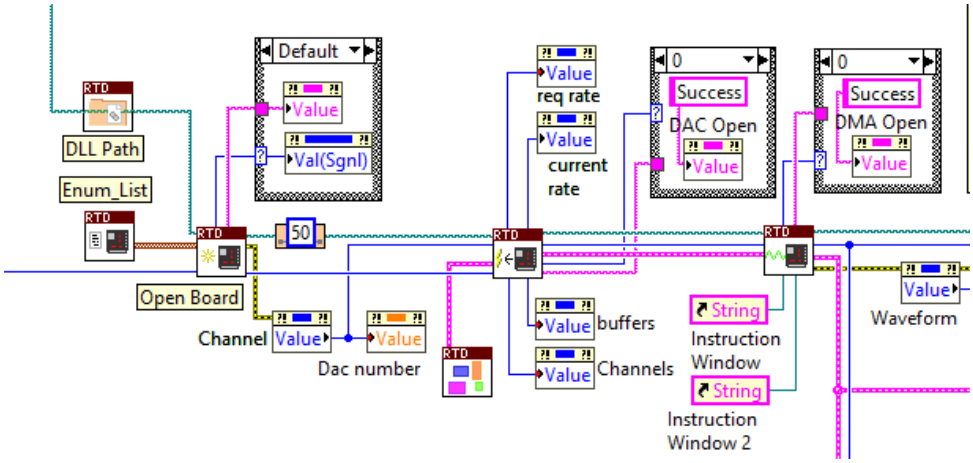


***Figure 3: Increased Readability through RTD Sub-VI's***

### 2.1.3 RTD'S DATAFLOW OUTLINE

Within each VI and Sub-VI, RTD controls dataflow in a similar way. The program will typically run like a regular text document, where dataflow goes from left to right, until it reaches the end of the screen, and then "returns" to one "line down". This is done so that each program can be read like any other document, and this direction of dataflow also gives access to the entire program using only the mouse wheel. Dataflow itself is typically controlled with the DLL Path wire because every library function that is called takes a DLL path as an input. Neither of these concepts need be followed when creating your own program, as they are just RTD's preference for controlling dataflow.

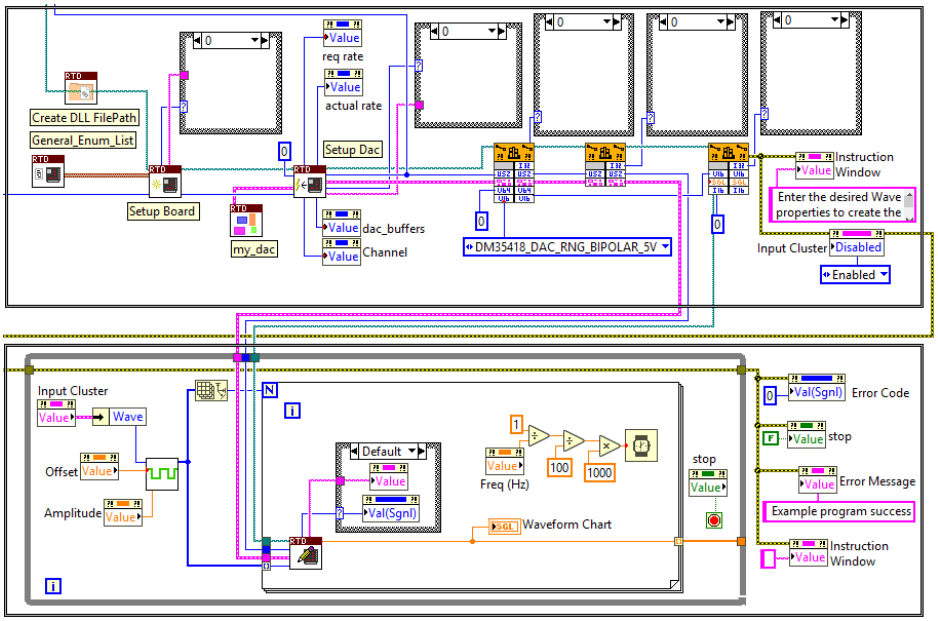


***Figure 4: Dataflow controlled by DLL Path***

# 3 Utilizing RTD Libraries in LabVIEW

## 3.1 RTD Library Layout

The purpose of each RTD Library is to compartmentalize the capabilities of each board to make it easier and faster to utilize the functionality of the board. Each Library contains its own number of source and header files, usually one for each feature (i.e. DMA, Temperature Sensor, Digital I/O, etc). With a few exceptions, all these functions can be called directly from LabVIEW.

### 3.1.1 Wrapper Functions

Certain functions have parameters that cannot be replicated in LabVIEW, and therefore a "wrapper function" must be created in the LabVIEW source file. An example of this is the Board_Open function, which takes the board number and a structure that contains handles. Handles cannot be created within LabVIEW, so to get around this, a LabVIEW_Board_Open function is created. This function only takes the board number as a parameter, but not the structure containing handles. The structure is instead created in the library source file, and then the wrapper function calls the original function with the newly created parameter. The LabVIEW_Open_Board function can be seen below.

```
DM35956_Board_Descriptor *Card;

DM35956LIB_API
DM35956_Error WINAPI DM35956_LabVIEW_Board_Open(uint8 dev_num, struct DM35956_Board_Descriptor **board_pointer)
{

    int result = DM35956_General_Open_Board(0,&Card);

    if (Card != NULL)
    {
        *board_pointer = Card;
    }
    return result;
}
```

***Figure 5: Wrapper Function to Open a DM35956***

The undefinable variable in this case is "Card". LabVIEW sends "board_pointer" into the wrapper function, and once "Card" has been filled, LabVIEW directs "board_pointer" to it, so that later functions can be called with "Card".

## 3.2 Accessing the RTD Library with the Call Library Function Node

On LabVIEW's block diagram, under "Connectivity->Libraries and Executables->" There is a "Call Library Function Node" (DLL Node). This node can be connected to a DLL which contains all functions within the RTD library. The DLL node can be configured extensively by double clicking on it.

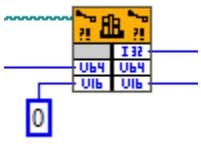


***Figure 6: Example of a configured Call Library Function Node***

### 3.2.1 CONFIGURING DLL NODE: FUNCTION INPUTS

The DLL Node has three types of input, the first two of which are configured under the "Function" Tab of the DLL Node.

1. The first type of input is an error wire. This can be used to control dataflow, and to prevent the called function from running if the error wire contains an error.

2. The second type of input is the DLL file-path, which has two ways to be configured:

    a. The first way is type the file-path to the DLL into the respective field or browse for the DLL.

    b. The second method is to check "Specify path on diagram", and then to run a path wire into the node. RTD uses the second method, and the DLL path wire contains a relative path that is initialized in the DLL_Path Sub_VI. The default DLL path generated by RTD's Sub-VI is the same file-path as the Example VI.

3. The third type of input into the DLL node are the function parameters. These can be configured under the parameters tab and will be explained in the next portion of this document.

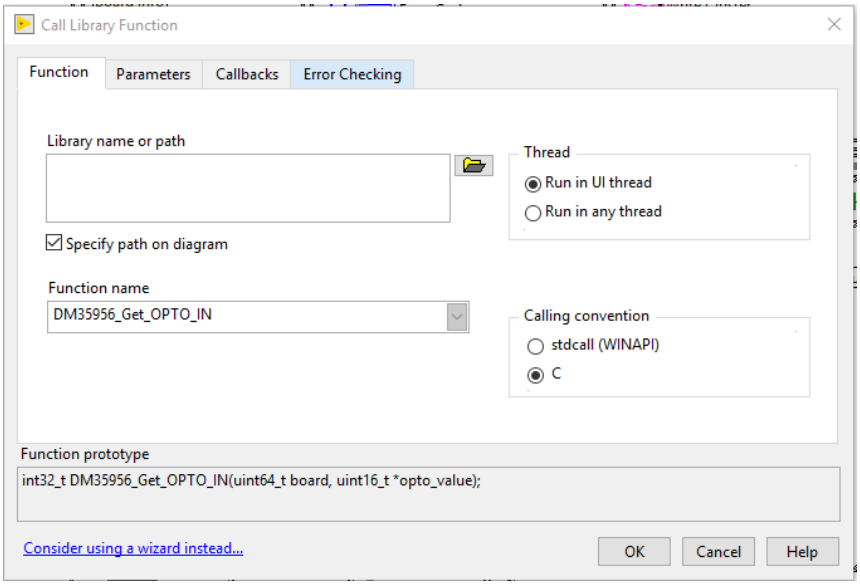


***Figure 7: DLL Node Functions Tab***

Entering functions: If the DLL file-path is typed into the "Library name or path", the "Function name" pull down will display all functions within that DLL. If the file-path is specified on the block diagram, then the function used will need to be manually entered. It may be faster to browse for the DLL, select the desired function, and then switch to "Specify path of diagram". If running 32-bit LabVIEW, the "Calling convention" radio button should be changed to "stdcall (WINAPI)", and no input parameters can contain 64-bit types (i.e. uint64 etc).

### 3.2.2 CONFIGURING DLL NODE: PARAMETERS TAB

This tab is dedicated to setting up the inputs and outputs of the function selected based on the steps in the previous section. Each parameter can be given a name, type, and different option depending on the type. The first parameter listed under this tab is always the value being returned by this function. RTD always returns an error value from its functions, so the outputs of DLL functions should always be a signed 32-bit integer. The input parameters need to be manually added to the DLL node. Hit the blue plus to add a new parameter, and then configure the new parameter to match the source code from the library. A guide for more complex input parameters can be found below.

This table displays input types in C that you would like to send into a library function on the left, and instruction to do that on the right. For example, if the function takes an array as an input, you must select the type “adapt to type”, and then select “array data pointer”. If you are trying to add a type of input that is not listed here, and is not a something simple like an integer, try going through every option the DLL has and selecting whichever option seems to match your input the best.

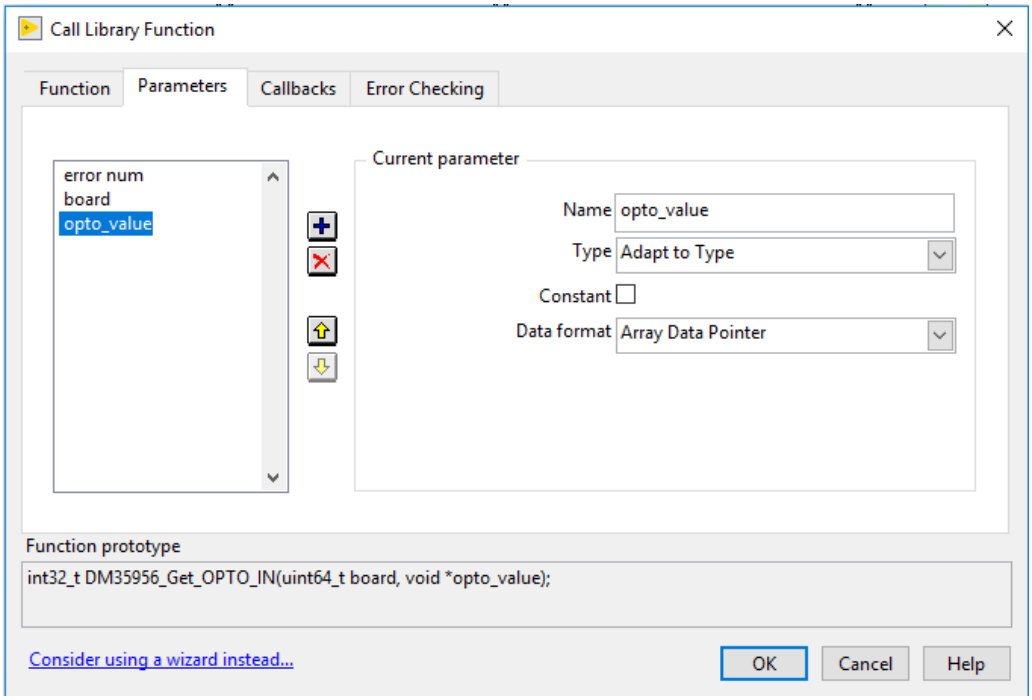


***Figure 8: Configuring array example***

Here are more example inputs that can be configured similarly to how the array was previously configured.

| Desired Type / Parameter | LabVIEW DLL Node Input |
|---|---|
| Pointer to type | Pass: Pointer to Value |
| Array | Adapt to Type: Array Data Pointer |
| Any Struct (Cluster) | Adapt to Type: Pointer to Handles or Handles by Values |
| Boolean Logic | Integer: uint8 |
| Enum or Ring | Integer: Match the type of the data in the Enum or Ring |
| Board Pointer (RTD Specific) | Integer: Pointer Sized Integer (This is only used for this parameter, and it automatically transfers between 64 and 32 bit for their respective programs) |
| Float (Any Analog Value) | Integer: 4 byte single or 8 byte double, depending on size |

***Figure 9: Table of LabVIEW parameters***

# 4 Resources

## 4.1 National Instruments™ LabVIEW

For more information on LabVIEW, refer to the website of National Instruments:

http://www.ni.com/

LabVIEW forum, where most facets of LabVIEW have been discussed at length:

https://forums.ni.com/t5/LabVIEW/bd-p/170

# 5 Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, Inc. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for a Return Material Authorization (RMA) number.

This limited warranty does not extend to any products which have been damaged as a result of accident, misuse, abuse (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, “acts of God” or other contingencies beyond the control of RTD Embedded Technologies), or as a result of service or modification by anyone other than RTD Embedded Technologies. Except as expressly set forth above, no other warranties are expressed or implied, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose, and RTD Embedded Technologies expressly disclaims all warranties not stated herein. All implied warranties, including implied warranties for merchantability and fitness for a particular purpose, are limited to the duration of this warranty. In the event the product is not free from defects as warranted above, the purchaser's sole remedy shall be repair or replacement as provided above. Under no circumstances will RTD Embedded Technologies be liable to the purchaser or any user for any damages, including any incidental or consequential damages, expenses, lost profits, lost savings, or other damages arising out of the use or inability to use the product.

Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, and some states do not allow limitations on how long an implied warranty lasts, so the above limitations or exclusions may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

**RTD Embedded Technologies, Inc.**
103 Innovation Boulevard
State College, PA 16803 USA
Telephone: 814-234-8087
Fax: 814-234-5218

www.rtd.com

sales@rtd.com
techsupport@rtd.com