THORLABS

Application Note

DM713 Digital Micrometer: LabVIEW Programming Overview

This application note provides a brief overview for interfacing with the DM713 Digital Micrometer using LabVIEW. A custom program that establishes a serial communications channel with the DM713, requests and receives output data from the micrometer, extracts the displacement value from the received data, and logs selected displacement readings to a text file is created with step-by-step instructions.

This example program is provided as a reference. The user is encouraged to extend or modify the program to fit the needs of the application.

August 12, 2019

1 Preface

This application note provides an introduction to communicating with the DM713 Digital Micrometer by presenting a custom LabVIEW program written for it. Section 2 provides step-by-step instructions for creating this virtual instrument (VI) and describes the purpose of each individual segment. The full VI alone is provided in Section 3.

Section 2 is divided into three parts. Section 2.1 provides instructions for downloading and installing the driver software. Information concerning the required cabling and finding the COM port is also included. In Section 2.2, a new VI is created, serial communications with the DM713 are established, output data from the micrometer is continuously received, and the displacement value and its units of measurement are displayed. The VI is expanded in Section 2.3 to include the option of logging selected displacement values to a user-specified file by clicking a button.

Supported functionality and procedures may differ from those described in this application note if different hardware, firmware, or software versions are used. The versions used to develop this program example were:

- DM173 Hardware / Firmware Version 350-357-30
- GUI and Libraries for the SBC-COMM RS-232 Interface & LabVIEW Driver for the Soleil-Babinet Compensator, Version 2.0.4[1]
- LabVIEW®: Version 17.0 (64 bit)

Note that this programming reference does not include error handling. Please see the National Instruments® website for information on error handling and proper programming practices.

[1] Software for the SBC-COMM can be downloaded from: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=SBC

2 Step-by-Step Instructions for Building the Example VI

Instructions for downloading and installing the driver software, as well as for determining the COM port for the DM713, are provided in Section 2.1. In Section 2.2 a new VI is started, the computer and DM713 are interfaced, the output data from the micrometer is continuously acquired, the displacement values are extracted from the output data, the units of measurement are determined, and the displacement values and their units are displayed. In Section 2.3, the VI is expanded to allow the user to log selected displacement values to a file by clicking a button. The Front Panel of the completed program is shown in Figure 1.

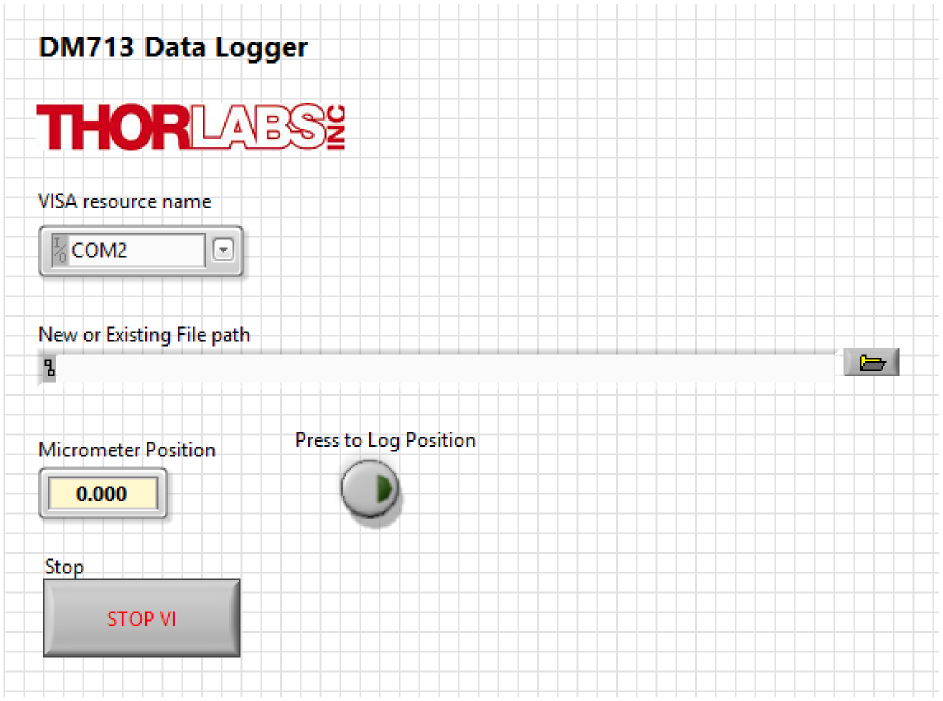


Figure 1 *This screen shot shows the Front Panel of the completed example application.*

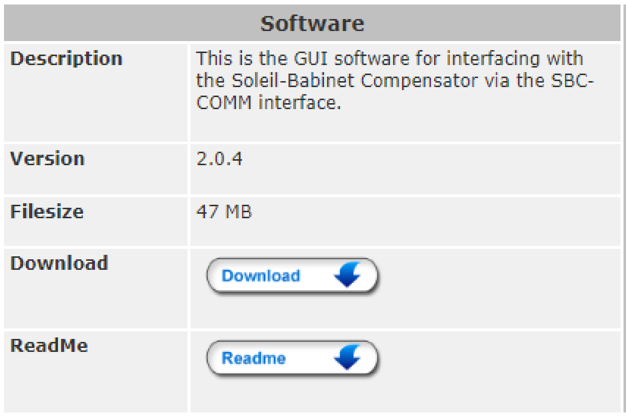
2.1 Download and Install GUI Software, Find COM Port

This section includes instructions for downloading and installing the GUI driver software, as well as determining the COM port that is needed to communicate with the DM713 in Step 10.

1. Download the GUI Software

The GUI software required to communicate with the DM713 is identical to that included on the CD enclosed with the SBC-COMM RS-232 Interface & LabVIEW Driver for the Soleil-Babinet Compensator. Either obtain the software from the CD or download the software from the Thorlabs website, as shown in Figure 2, by navigating to:

https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=SBC



Software	
Description	This is the GUI software for interfacing with the Soleil-Babinet Compensator via the SBC-COMM interface.
Version	2.0.4
Filesize	47 MB
Download	Download
ReadMe	Readme

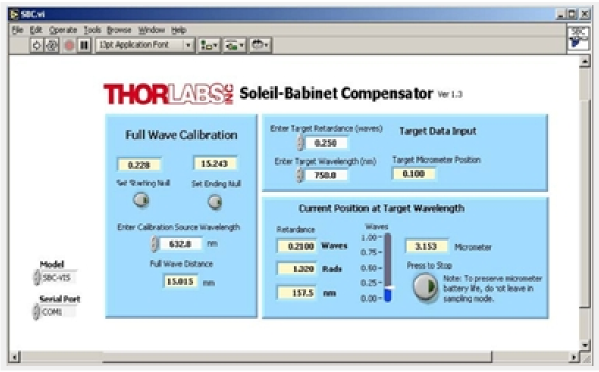
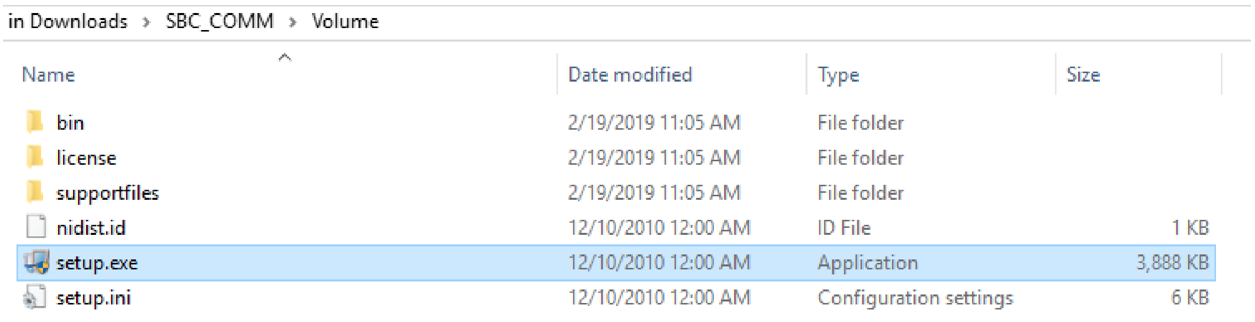

SBC.vi
File Edit Operate Tools Browse Window Help
13pt Application Font
THORLABS Inc Soleil-Babinet Compensator Ver 1.3
Full Wave Calibration
0.228
15.243
Set Starting Null
Set Ending Null
Enter Calibration Source Wavelength
632.8 nm
Full Wave Distance
15.015 nm
Enter Target Retardance (waves)
0.250
Target Data Input
Enter Target Wavelength (nm)
750.0
Target Micrometer Position
0.100
Current Position at Target Wavelength
Retardance
0.2100 Waves
1.320 Rads
157.5 nm
Waves
1.00
0.75
0.50
0.25
0.00
3.153 Micrometer
Press to Stop
Note: To preserve micrometer battery life, do not leave in sampling mode.
Model
SBC-VIS
Serial Port
COM1

Figure 2 *The GUI software for required to create this custom VI for controlling the DM713 is the same as that provided for the SBC-COMM. This software can be downloaded by navigating to the following location: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=SBC.*

2. Install the GUI Software

Unzip the file downloaded in Step 1 and open the Volume folder. Run the setup.exe installer file, which is shown in Figure 3, and follow the prompts to complete the installation of the drivers for the DM713.



in Downloads › SBC_COMM › Volume

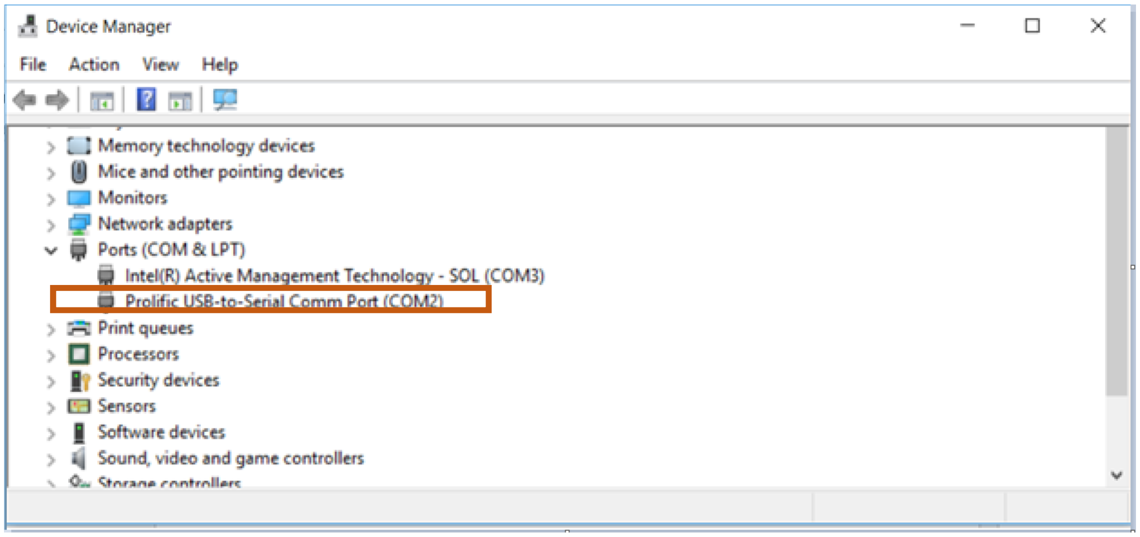
Name	Date modified	Type	Size
bin	2/19/2019 11:05 AM	File folder	
license	2/19/2019 11:05 AM	File folder	
supportfiles	2/19/2019 11:05 AM	File folder	
nidist.id	12/10/2010 12:00 AM	ID File	1 KB
setup.exe	12/10/2010 12:00 AM	Application	3,888 KB
setup.ini	12/10/2010 12:00 AM	Configuration settings	6 KB

Figure 3 *Run the setup.exe file, which is found in the Volume folder of the downloaded GUI software, to install the drivers for the DM713.*

3. Use Cable to Connect Computer and Micrometer and Find COM Port

Connect the DM713 digital micrometer to the computer using an RS-232 cable. If the computer does not have an RS-232 port, an RS-232 to USB converter can be used. Both cables are supplied with the SBC-COMM.

Wait until the computer recognizes the device. If the computer is running a Microsoft® Windows operating system, the DM713 will be listed in the device manager under **Ports (COM and LPT)** with a name that depends on the cable. The cable In this case was a Prolific USB-to-Serial cable, as shown in Figure 4. Make a note of the COM port, which is COM2 in this case.



Memory technology devices
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
Prolific USB-to-Serial Comm Port (COM2)
Print queues
Processors
Security devices
Sensors
Software devices
Sound, video and game controllers
Storage controllers

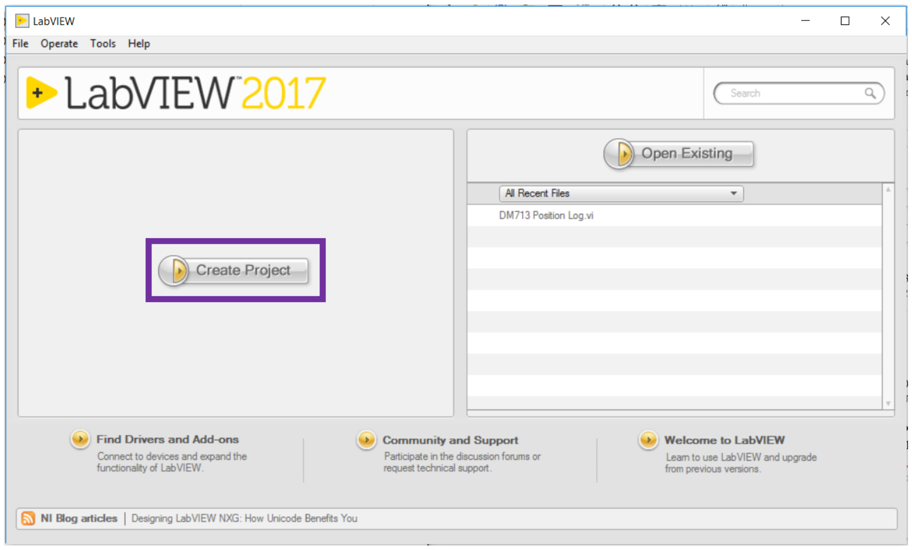
Figure 4 *The serial port used to communicate with the DM713 can be found under the Ports heading in the Windows® Device Manager. In this case, device the port is COM2.*

2.2 Acquiring and Displaying Measurement Data

In this section, a new LabVIEW VI is started, communication between the computer and DM713 micrometer is established, the continuous acquisition of output data from the micrometer is enabled, displacement measurements are extracted from these data, and the displacement values are displayed with their units of measurement.

4. Start a New Blank VI

Open LabVIEW. Click on the **Create Project** link, which is enclosed in the purple box in Figure 5. Choose the new **Blank VI** option.



Connect to devices and expand the functionality of LabVIEW.

Participate in the discussion forums or request technical support.

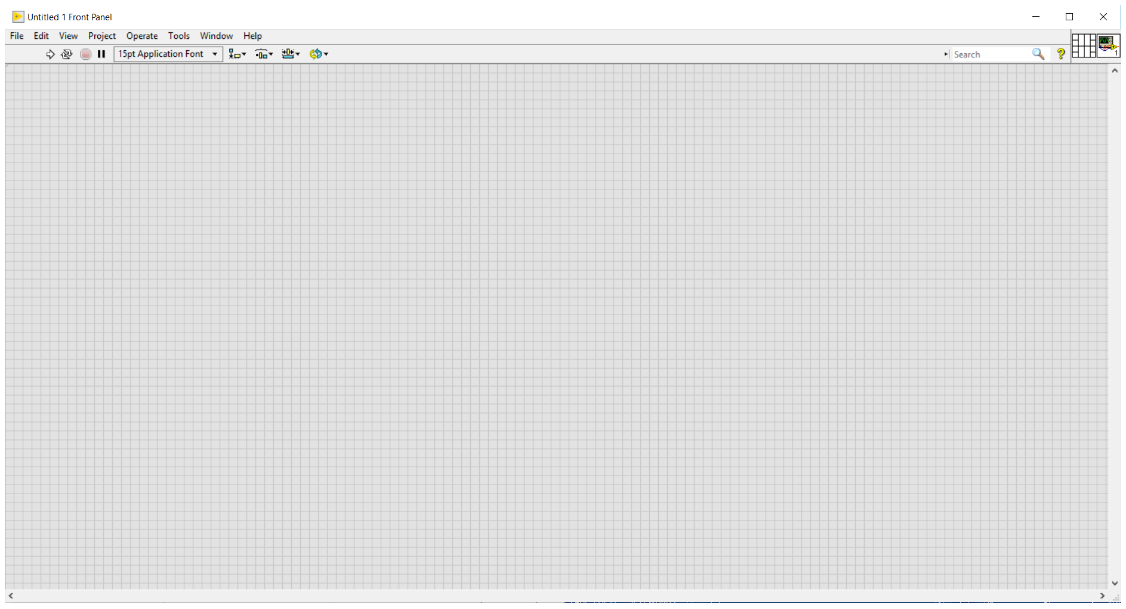
Learn to use LabVIEW and upgrade from previous versions.

Designing LabVIEW NXG: How Unicode Benefits You

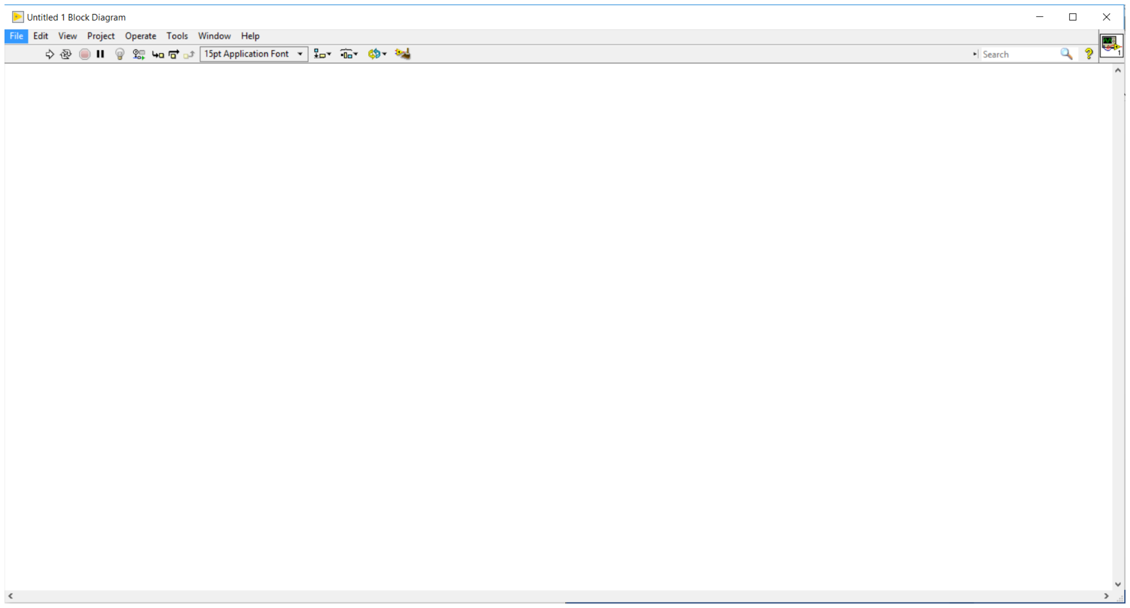
Figure 5 *Open LabVIEW, click the **Create Project** button, and then select **Blank VI**.*

5. *The Front Panel and Block Diagram of a VI*

The new VI consists of two empty windows, which are called the **Front Panel** and the **Block Diagram**. These are shown at the top and bottom of Figure 6, respectively. The **Front Panel** has a grid overlaid on a gray background and is the user interface to the program. The **Block Diagram**, which has a white background, is the window where the graphical program code is placed.



(a) **Front Panel**



(b) **Block Diagram**

Figure 6 *A VI consists of a* ***Front Panel****, which is the user's interface with the program, and a* ***Block Diagram*** *(b), where the code is placed. Both windows are empty, since this is a new Blank VI.*

6. Add the subVI Used to Support Serial Communications

Make the **Block Diagram** the active window, and then press <Ctrl-Space> to open the **Quick Drop** menu. Find the *VISA Configure Serial Port* subVI, as shown in Figure 7. Select this subVI to add it to the **Block Diagram** as shown in Figure 8. A subVI is a VI added to the **Block Diagram** of another VI.

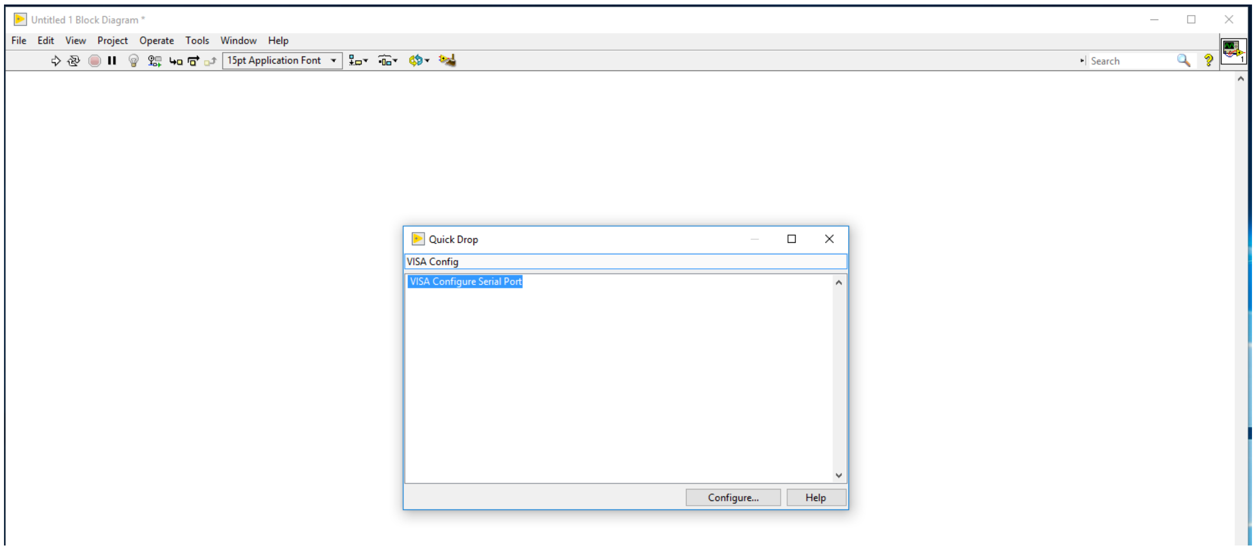

Untitled 1 Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Quick Drop
VISA Config
VISA Configure Serial Port
Configure...
Help

Figure 7 *The Quick Drop menu was opened from the* ***Block Diagram*** *and used to quickly locate the* VISA Configure Serial Port *subVI.*

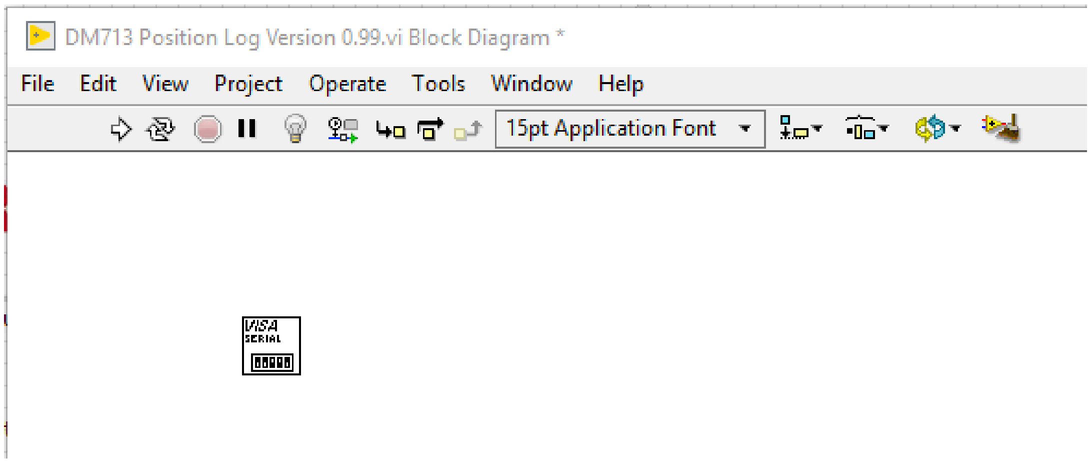

DM713 Position Log Version 0.99.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
VISA SERIAL

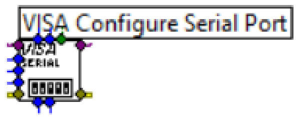
Figure 8 *Selecting the* VISA Configure Serial Port *subVI from the* ***Quick Drop*** *menu adds it to the* ***Block Diagram****.*

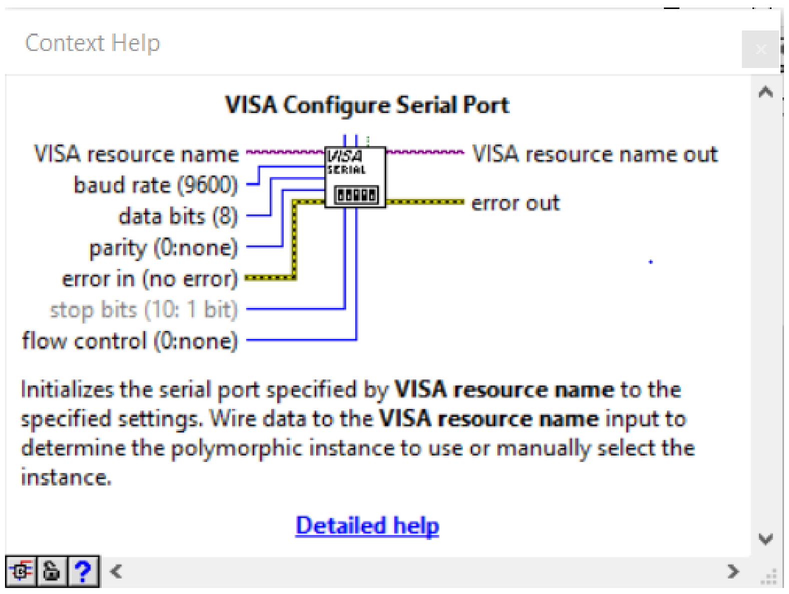
7. The VISA Configure Serial Port VI

Each LabVIEW VI and subVI has one or more terminals, which function as inputs to and outputs from the VI or subVI. The inputs and outputs are called controls and indicators, respectively. The locations of the terminals on a subVI become visible when the cursor hovers over the subVI in the block diagram, as illustrated in Figure 9(a).

Pressing <Ctrl-H> opens a **Context Help** window. When the cursor hovers over an object, such as a subVI on the **Block Diagram**, information about the object is displayed in the **Context Help** window. A **Context Help** window, showing information provided for the *VISA Configure Serial Port* subVI, is shown in Figure 9(b). The information displayed in the window for this subVI includes summary of the subVI's function and the names and descriptions of each terminal.

The National Instruments® website includes additional resources that provide detailed information about serial communications, the *VISA Configure Serial Port* subVI, and the other *VISA VIs* discussed in Step 9.





Initializes the serial port specified by **VISA resource name** to the specified settings. Wire data to the **VISA resource name** input to determine the polymorphic instance to use or manually select the instance.

(a) (b)

Figure 9 *The placement of terminals on the VISA Configure Serial Port subVI are revealed when the cursor hovers over the subVI in the Block Diagram (a). The function of each terminal e*

8. Adding Controls to the VISA Configure Serial Port VI

In order for the *VISA Configure Serial Port* subVI to operate as required, input values (controls and constants) need to be wired to a number of its terminals. To connect a control to a terminal, right click on the terminal to pop up a menu and then navigate through *Create*, and select *Control*, as is illustrated in Figure 10. The control will be visible on the **Front Panel**, and its value can be specified there. Selecting *Constant* from the menu creates an input that is not visible from the **Front Panel** and whose value is specified on the **Block Diagram**.

Right click on the purple terminal at the upper left corner of the *VISA Configure Serial Port* subVI (VISA resource name) and create a control. This control will allow the user to specify the COM port corresponding to the DM713 when the VI is running. The COM port that should be specified was found in Step 3.

Right click on the terminal for *baud rate*, and create a constant. Enter the value, *2400*, indicated in Figure 11. Repeat this procedure for the *data bits*, *parity*, *stop bits*, and *flow control* terminals. Similarly, assign the values *8*, *None*, *1.0*, and *None (0)*, to each respectively. These constants will not be visible on the **Front Panel**.

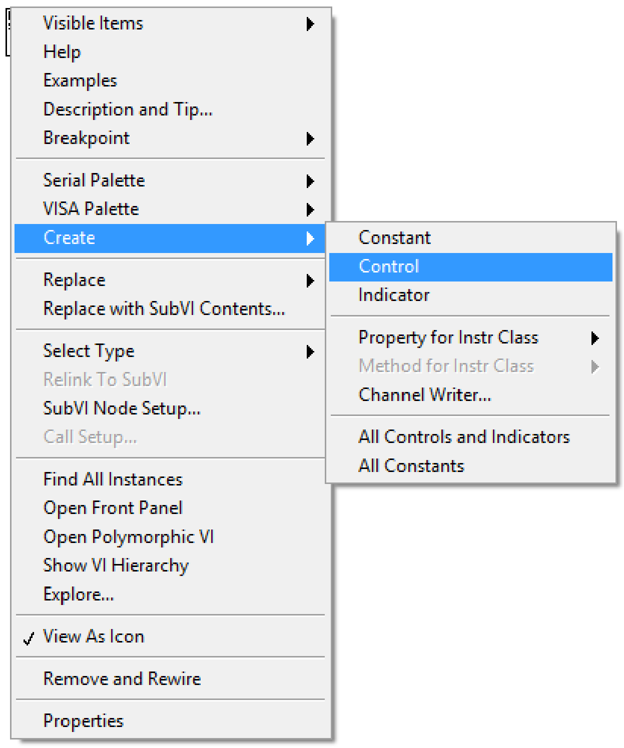


Figure 10 *A control can be created by right clicking on the terminal of a subVI to pop up a menu and then navigating to* Create *and then* Control.

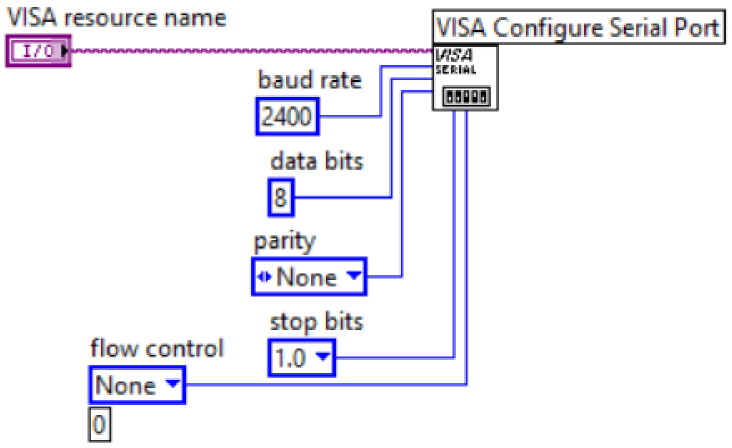


Figure 11 *Right click on the* VISA resource name *terminal and create a control. Right click on and create constants attached to the* baud rate, data bits, parity, stop bits, *and* flow control *terminals. Assign the constants the values shown above. The* VISA resource name *control will be set from the* ***Front Panel***.

9. Add Additional Functions Used to Communicate with the DM713

Additional functions needed to communicate with the micrometer must be added to the VI. Using the Quick Drop feature discussed in Step 6, add the following objects. Arrange and wire them as shown in Figure 12.

VISA Write: The DM713 provides an output data value in response to being sent a single byte, and the *VISA Write* function is used to write (send) that byte.

- After dragging and dropping the *VISA Write* function to the **Block Diagram**, right click the *write buffer* terminal and create a constant. The specific value of the constant does not matter, since the DM713 provides an output value in response to any byte written to it. For this example, the constant 1 is used.
- Wire the *VISA resource name out* terminal from the *VISA Configure Serial Port* subVI to the *VISA resource name* terminal on the *VISA Write* function.

Wait(ms): Since read and write operations do not occur instantaneously, delays are built into the VI to allow the operations to complete. The *Wait(ms)* function sets the delay duration.

- Right click on the left, *milliseconds to wait*, terminal of the *Wait(ms)* function and create a constant. Make this constant 300.

Flat Sequence Structure: These structures are used to control the order in which operations are executed. This structure is used with the *Wait(ms)* function to provide enough time for the bytes from the DM713 to become available before the program attempts to read the bytes.

- Position the *Wait(ms)* function inside of the *Flat Sequence Structure* by clicking the structure and dragging it around the *Wait(ms)* function and its attached constant.
- Wire the *VISA resource name out* terminal on the *VISA Write* function to a tunnel on the left side of the structure's frame. Then connect a wire between this tunnel and another tunnel on the right side of the structure's frame. This will force a 300 ms wait after the *VISA Write* function executes before the *VISA Read* function executes, which provides the DM713 with time to interpret the command and send a return string in response.

VISA Property Node: The *VISA resource name out* contains several properties, but only one is required by the *VISA Read* function in this application. Use the *VISA Property Node* to select the property of interest.

- Connect a wire to the tunnel on the right of the Flat Sequence Structure to access the *VISA resource name out* refnum and wire it to the *VISA Property Nod*e. Left click on the white box at the bottom of the property node. Choose the *Number of Bytes at Serial Port* property, as shown in Figure 13.

VISA Read: This function is used to read the data output from the DM713.

- Wire the *reference out terminal* on the *Property Node* to the *VISA resource name* terminal on the *VISA Read function*. This function reads the bytes corresponding to the data output by the DM713.

- Right click on the *read buffer* terminal on the *VISA Read* function to create an indicator, This creates a display on the **Front Panel** that will show the data output by the DM713.

VISA Close: This function closes the COM port to the DM713, which makes the DM713 available to any other application that may need to communicate with the micrometer.

- Wire the *VISA resource name out* terminal to the *VISA resource name* terminal on the *VISA Close* function.

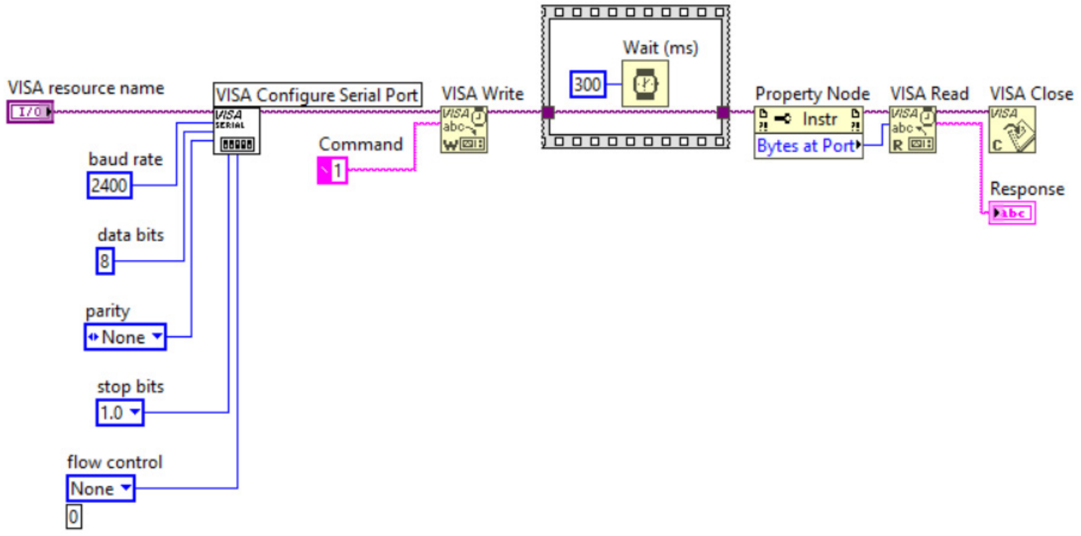


Figure 12 *The subVIs and functions that are used to communicate with the DM713.*

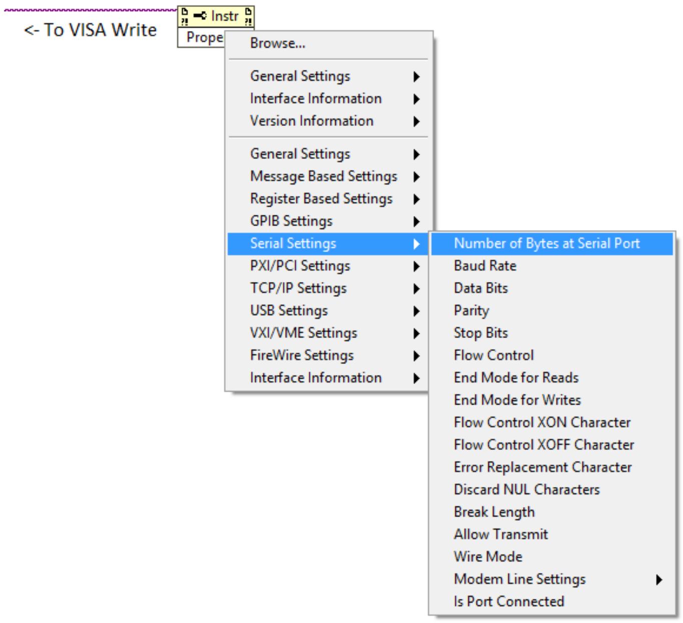


Figure 13 *The* Number of Bytes at Serial Port *property required by the* VISA Read *function is accessed through the Property Node menu as shown.*

10. Test Run to Confirm Successful Communication with the DM713

From the Front Panel, select the COM port found in Step 3 and run the VI. The output string provided by the DM713 should be displayed. The output data from the micrometer includes the displacement value, which was -2.394 in the example shown in Figure 14.

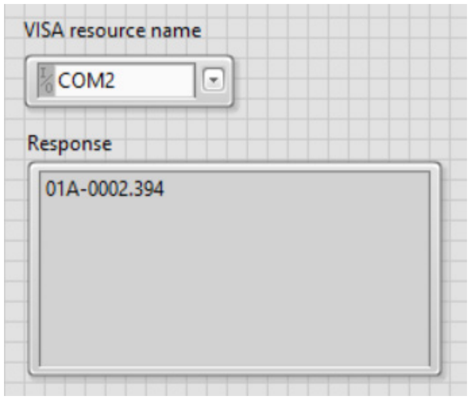


Figure 14 *Running the exiting code displays the data output by the DM713 to the Response indicator. This string includes the displacement value, which is -2.394 in this example.*

11. Extract the Displacement Value from the Output Data

The DM713 outputs a data string in which only the embedded displacement value is of interest to this application. The *Scan From String* function is used to extract the displacement value.

Place the *Scan From String* function close to the *VISA Read* function on the **Block Diagram**. As Right click on the *format string* and *initial scan location* terminals to create constants with values *%f* and *3*, respectively, as shown in Figure 15. Wire the *read buffer* terminal on the *VISA Read* function to the *input string* terminal on the *Scan From String* function. Right click on the *output terminal* and create an indicator.

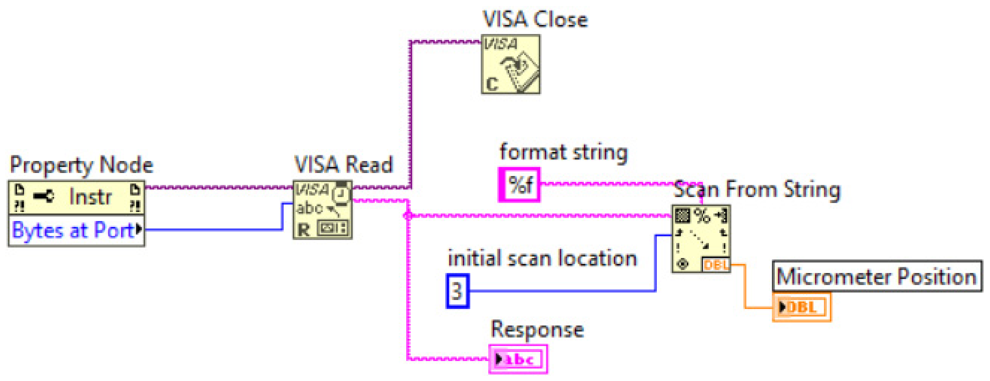


Figure 15 *The* Scan From String *function is used to extract the displacement value embedded in the string output from the DM713 and display it on the **Front Panel**.*

The *Scan From String* function, wired with these controls, ignores the first three bytes (from left as displayed in the *Response* indicator) of the string output by the DM713, converts the remainder of the string into a floating point number, and displays the resulting value on the **Front Panel**. This is shown in Figure 16. The value in the *Micrometer Position* indicator is more easily interpreted as a displacement value than the unprocessed output data shown in the *Response* indicator. The *Response* indicator can be deleted later but is useful now.

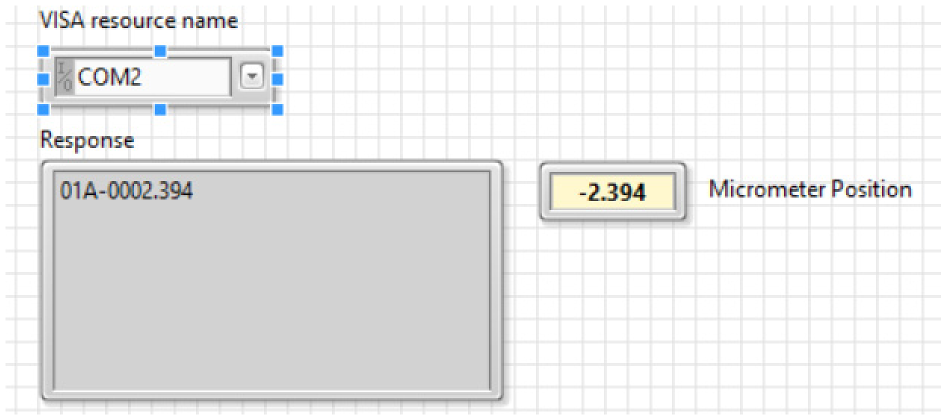


Figure 16 *When running, the program displays the output data in the* Response *indicator and the extracted displacement value in the* Micrometer Position *indicator.*

12. Determine and Display the Measurement Units

The data string from the DM713 does not explicitly specify the units of measurement (inches or millimeters), which need to be displayed by the VI to provide clarity to the end user. However, the number of digits to the right of the decimal place happen to correlate to the units of measurement. There are three digits after the decimal point when the units are millimeters, but there are five digits when the units are inches.

Add a *Search/Split String* function to the VI, and place it below the *Scan from String* function. Add a *String Length* function to the right of the *Search/Split String* function, a *Greater?* function, and then a *Select* function, as shown in Figure 17.

Wire the *read buffer* terminal on the *VISA Read* function to the *string* terminal on the *Search/Split String* function. Right click on the *search string/char* terminal and create a constant consisting of a decimal point. Wire the *match + rest of string* terminal to the input of the *String Length* function. Wire the output of the *String Length* terminal to an input of the *Greater?* function. Right click on the other terminal and create a constant with value *5*. Wire the output of the *Greater?* function to the *s* terminal of the *Select* function. Right click on the *t* and *f* terminals of the *Select* function and create constants with values *in* and *mm*, respectively. Click on the *s?tf* terminal of the *Select* function and create an indicator.

The *Search/Split String* function locates the decimal point in the output string from the DM713, and provides a string consisting of the decimal point and all digits that follow it. The String Length function counts the number of bytes in this string. If the number of bytes is greater than five, the output of Greater? is true. Otherwise it is false. The Boolean output is from Greater? is used to determine whether to display in or mm as the units of measurement.

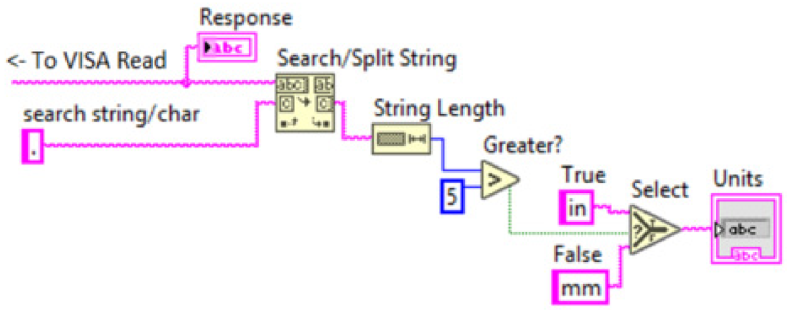


Figure 17 *The* Search/Split String *function and a few other supporting functions are used to determine the length of the sub-string that includes the decimal point and all of the following digits. A length greater than five corresponds to measurement units of inches, otherwise the units are millimeters.*

13. Accommodating the Case when the Output String from the DM713 is Empty

The output string from the DM713 can be empty due to an event such as a USB connection error or latency effects in the network between the computer and micrometer. An empty string has a zero length, and the functions after the VISA Read function will crash the program if an empty string is passed to them. Recovering from the event of an empty string requires restarting LabVIEW. Often, it also requires unplugging and then re-plugging the USB cable from the micrometer before restarting the VI.

In this example, and as shown in Figure 18, Error Handling was implemented to avoid passing an empty string to the *Scan From String* and *Search/Split String* functions. The error handling code:

- A *String Length* function added between the VISA Read function and the *Scan From String* and *Search/Split String* functions. The *read buffer* terminal of the *VISA Read* function was wired to the input terminal of the *String Length* function. Its output terminal was wired to the input of an *Equal To 0?* function.
- A *Case Structure*, with the case set to false, enclosing the *Scan From String* and *Search /Split String* functions and everything following them. The connection to the *read buffer* terminal on the *VISA Read* function was passed into the structure through a tunnel, which is located on the left border of the *Case Structure*.
- No sub-diagram was added to the true case.
- The Boolean output of the *Equal To 0?* function was wired to the selector of the *Case Structure* and used to determine which case would execute.

When the string from the DM713 is empty, the output of the *Equal To 0?* function is true, and the program does not operate further on the string. When the string output from the DM713 is not empty, the output of the *Equal To 0?* function is false, and the program finds the displacement value and its unit of measurement.

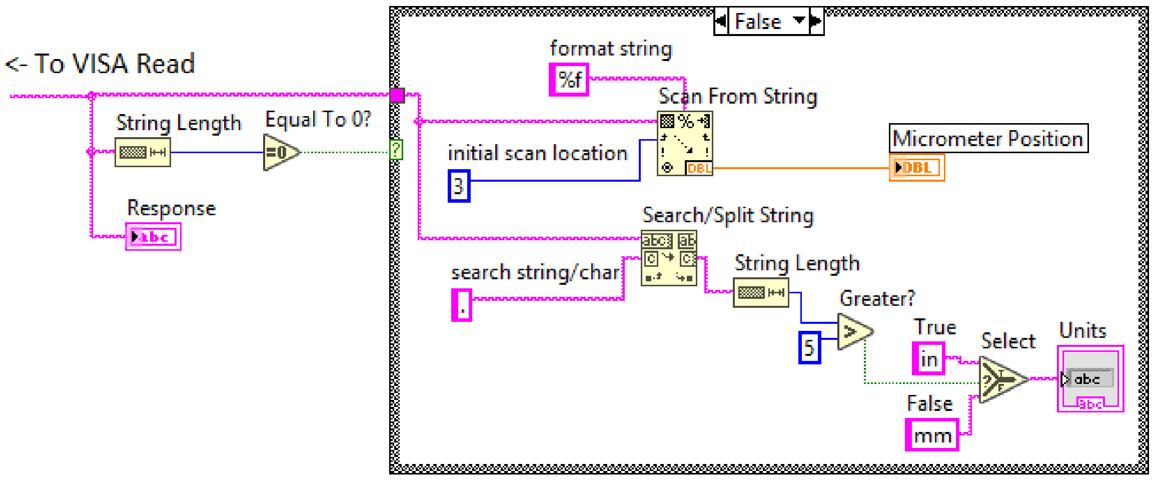


Figure 18 *Error handling was added to address the event of an empty string read by the* VISA Read *function. The code following the* VISA Read *function was enclosed by the false case of a* Case Structure. *The true case was empty. The combination of the* String Length *and* Equal To 0? *functions on the left determine whether the string read by the* VISA Read *function is empty. If it is not, the displayed sub-diagram executes. If the string is empty, the VI does not operate further on the string.*

14. Provide the Option to Continuously Acquire Output Data from the DM713

Displacement values measured by the DM713 can be continuously acquired if the most of the code is enclosed in a *While Loop* structure.

Place everything except the *VISA Configure Serial Port* subVI and the *VISA Close* functions in the *While Loop*, as illustrated in Figure 19.

Add a *Stop* function inside the *While Loop*, and wire it to the conditional terminal located at the bottom right corner of the *While Loop*. The value of the *Stop* function can be toggled by the user from the **Front Panel** while the VI runs.

Add a *Wait(ms)* function inside and near the upper left corner of the *While Loop*. Right click on the *milliseconds to wait* terminal of the *Wait(ms)* function and add a constant. In this example, the value of the constant was set to 200.

Upon running the VI, the serial port will be configured once, and then the code inside the *While Loop* will be repeatedly executed. This will continually update the indicators on the **Front Panel** with the current displacement values read by the micrometer. Changing the value of the constant wired to the *Wait(ms)* function at the upper left corner of the *While Loop* changes the rate at which the output data is read from the DM713. When the *Stop* button on the **Front Panel** is pressed, the VI exits the *While Loop*, *VISA Close* function closes the serial port, and the VI ends.

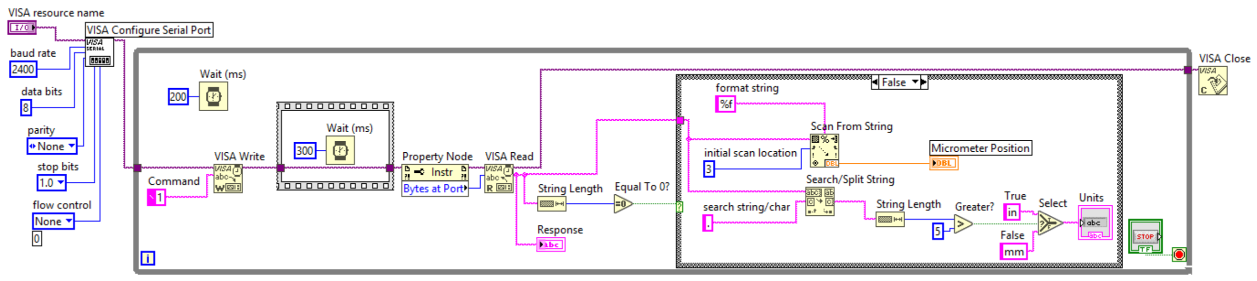


***Figure 19** The displacement value read by the micrometer and displayed on the **Front Panel** can be continuously updated by enclosing all but the* VISA Configure Serial Port *subVI and the* VISA Close *functions in a* While Loop*. Execution can be stopped from the **Front Panel** by adding a* Stop *function control and wiring it to the* While Loop's *conditional terminal. The rate at which the displacement value updates can be controlled by adding a* Wait (ms) *function to the* While Loop*.*

2.3 Logging the Displacement Data to a File

In this section, the VI is expanded to allow the user to save selected displacement values and their timestamps to a file. The code creates the file if it does not exist. Otherwise, the user is presented with the option of overwriting the existing file. A displacement value and its timestamp are appended to the file each time a button is clicked.

15. Add Code to Create a New File or Overwrite an Existing File

Use LabVIEW's Quick Drop feature, which was discussed in Step 6, to add several functions and a Case structure to a location in the Block Diagram to the left of the existing code. As illustrated by Figure 20, the functions and the connections to add are:

Strip Path: This function accepts a path, and makes the last component of the path available at one terminal while the remainder of the path is available at another terminal.

- After placing the *Strip Path* function, right click on the left *path* terminal and create a control. This control will be visible on the **Front Panel** and will allow the user to input a file path. In this example VI, the file must be a text file.

Check if File or Folder Exists: This subVI accepts a path and determines whether this path exists.

- Place the *Check if File or Folder Exists* subVI to the right of the *Strip Path* function.
- Wire the *stripped path* terminal from the *Strip Path* function to the *path* terminal on the *Check if File or Folder Exists* subVI.
- Create a *Case Structure*.
- Wire the *dup path* and the *file or folder exists?* terminals of the *Check if File or Folder Exists* subVI to a tunnel on the left side of the *Case Structure* and to its conditional terminal, respectively.

Create Folder: This function creates the folder specified by the input path.

- Add the *Create Folder* function to the false case of the *Case Structure*.
- Wire the *path* control connected to the tunnel on the left side of the Case Structure to the *path (use dialog)* terminal on the *Create Folder* function. Wire the *created path* terminal on the *Create Folder* function to a tunnel on the right side of the *Case Structure*.
- Change the *Case Structure* condition to true. Wire the left tunnel to the right tunnel.

Build Path: This function appends a name to a path to create a complete new path.

- Add the *Build Path* function to the right of the *Case Structure*.
- Using the tunnel through the right side of the *Case Structure*, wire the *created path* terminal on the *Create Folder* function to the *base path* terminal on the *Build Path* function. Wire the *name* terminal on the Strip Path function directly to the *name or relative path* terminal on the *Build Path* function.

Open/Create/Replace File: This function can be used to open a new file or replace an existing file. Its action can be controlled by the user or by the VI.

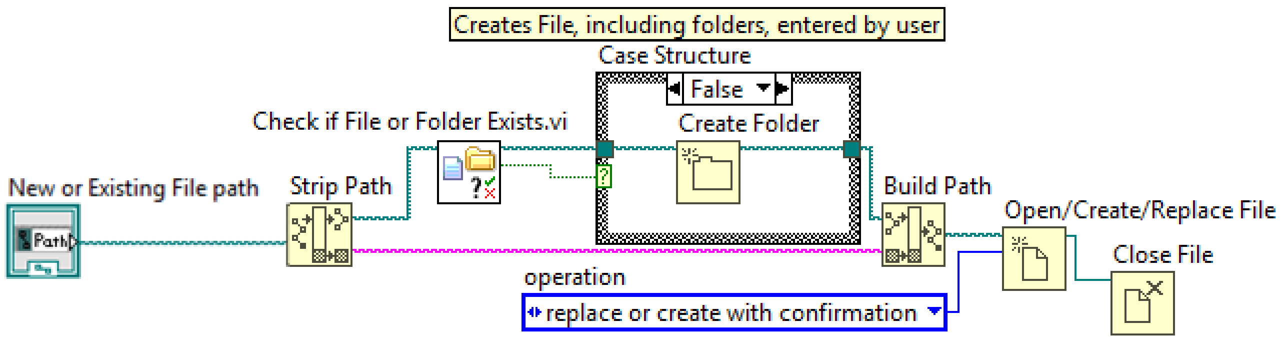
- Add the *Open/Create/Replace File* function to the right of the *Build Path* function.
- Wire the *appended path* terminal of the *Build Path* function to the *file path (use dialog)* terminal on the *Open/Create/Replace File* function.
- Right Click on the operation terminal of the *Open/Create/Replace File* function and create the *replace or create with confirmation* constant.

Close File: This function closes the open file identified by *refnum*.

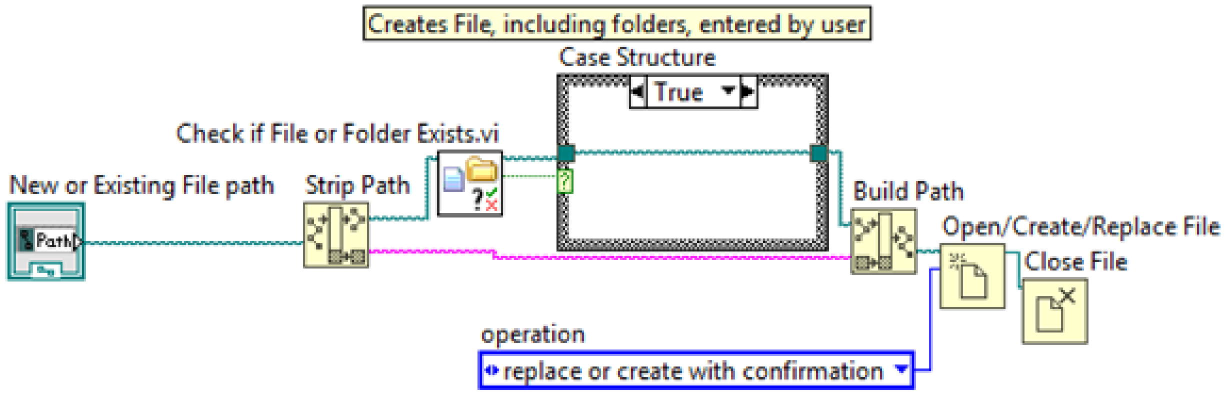
Add the *Close File* function to the right of the *Open/Create/Replace File* function.

Wire the *refnum out* terminal of the *Open/Create/Replace File* to the *refnum* terminal of the *Close File* function.

This code places a control on the **Front Panel** that allows the user to enter the file path corresponding to the file in which user-selected displacement data should be saved. The code creates the file path if it does not already exist. If the specified file path and file exist, the VI provides a prompt asking the user to confirm that the file should be overwritten. This prompt is provided as a result wiring the *replace or create with confirmation* constant to the *Open/Create/Replace File* function. Without the *Close File* function, an error event occurring elsewhere in the VI may result in the loss of data before it can be added to the file, and possibly the loss of the file itself.


Creates File, including folders, entered by user
Case Structure
False
Check if File or Folder Exists.vi
Create Folder
New or Existing File path
Path
Strip Path
Build Path
Open/Create/Replace File
Close File
operation
replace or create with confirmation

(a) False Case of Case Structure


Creates File, including folders, entered by user
Case Structure
True
Check if File or Folder Exists.vi
New or Existing File path
Path
Strip Path
Build Path
Open/Create/Replace File
Close File
operation
replace or create with confirmation

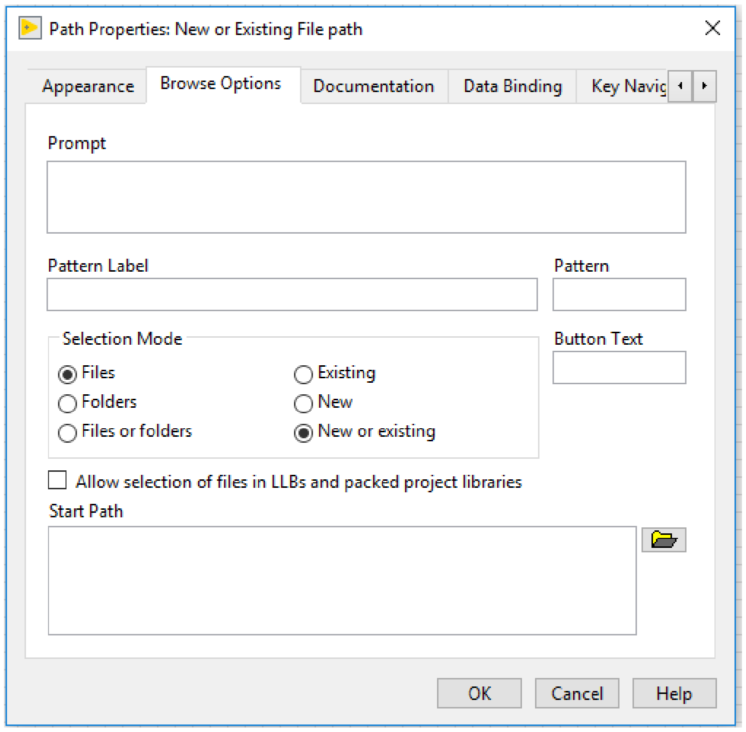
(b) True Case of Case Structure

Figure 20 *The code allows the user to specify a file path from the Front Panel. If the path does not exist, it is created. If the file does not exist, it is created. If the file does exist, the user is asked whether it can be overwritten.*

16. Set Browsing Options for File Path Control

Switch to the **Front Panel** of the VI, which can be quickly done using the <Ctrl-E> shortcut that allows users to toggle between the **Front Panel** and the **Block Diagram**.

Right click on the *New or Existing File path* control and click *Properties*. Under *Browse Options*, make sure *Selection Mode* is set up with *Files* and *New or existing* selected, as shown in Figure 21.


Path Properties: New or Existing File path

Appearance | Browse Options | Documentation | Data Binding | Key Navig

Prompt

Pattern Label

Pattern

Selection Mode

[✓ Files] Folders Files or folders

Existing New [✓ New or existing]

Button Text

☐ Allow selection of files in LLBs and packed project libraries

Start Path

OK | Cancel | Help

Figure 21 *Right click on the* New or Existing File path *control on the* ***Front Panel*** *and click* Properties *to open the window shown above. In the* Selection Mode *section of the* Browse Options *tab,* Files *and* New or existing *should be selected.*

17. Use Flat Sequence to Control Order of Execution

This VI uses a structure called a *Flat Sequence* to force the code described in Step 15 to execute first, before the computer opens communication with the DM713.

Switch back to the Block Diagram using <Ctrl-E>. Add a *Flat Sequence* structure to the VI, then expand the structure to two frames by right clicking on the sequence and select the *Add Frame After* option. Drag all of the code shown in Step 15 into the first frame. Drag all of the code shown in Step 14 into the second frame. The final result is shown in Figure 22.

This constrains the VI to create a user-specified input text file first and then continually read the micrometer position.

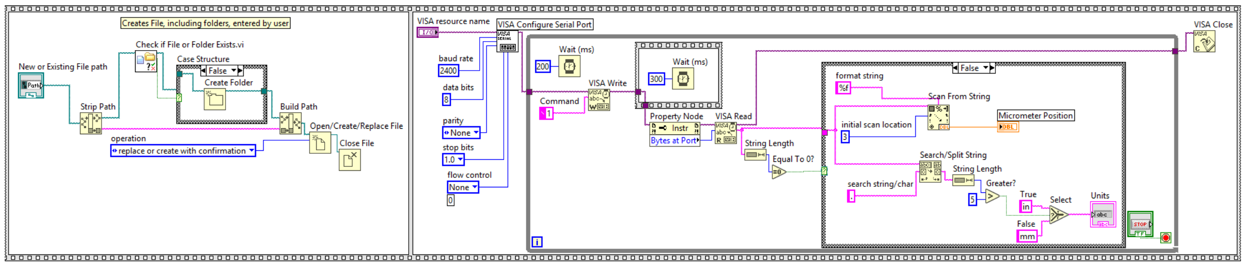


Figure 22 *After adding a two-frame* Flat Sequence *to the VI, drag the code from Step 15 into the first frame and the code from Step 14 into the second. This constrains the VI to create the input text file before continuously reading the micrometer position.*

18. Create Local Variable to be Used to Write Displacement Value

This VI allows the user to write selected displacement measurements to the file opened by the code discussed in Steps 15 and 16. The path for this file, which is specified by the *New or Existing File path* control shown on the left side of Figure 20(a, b), must be referenced when writing data to the file. A *Local Variable* can be used to provide the required reference.

Right click on the *New or Existing File path* control on the **Block Diagram** to open a menu. From *Create*, select *Local Variable*, as shown in Figure 23.

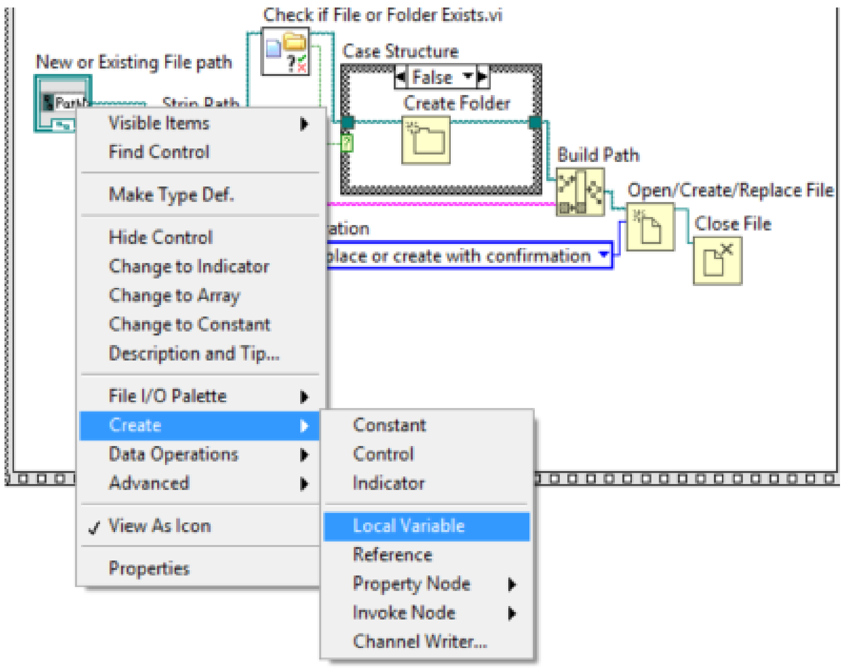


Figure 23 *Create a* Local Variable *that will be used to specify whether to write the displacement value to a file.*

Create a local variable of the *New or Existing File path* control by right clicking the control, then selecting the *Local Variable* option under *Create* in the menu.

Right click the local variable, and then select *Change to Read*. This local variable links to the file path created by the code in the first frame in Figure 22.

19. Open the Text File Before Writing the Displacement Value

This code opens the user's text file and sets the position of the file mark in preparation for writing a displacement value to the file. This code specifies that the new data value be appended to the end of the file.

Use the **Quick Drop** functionality to place a new *Open/Create/Replace File* function and a *Set File Position* function near the existing code segment that includes the *Scan From String* and *Search/Split String* functions. This existing code segment is located in the false *Case Structure* inside the second frame of the *Flat Sequence* structure discussed in Step 17. It may be necessary to enlarge the frames of the *Flat Sequence*, the *While Loop*, and the *Case Structure* to provide the necessary space. As shown in Figure 24,

- Wire the *New or Existing path* local variable, which was created in Step 18, to the *file path (use dialog)* terminal on the *Open/Create/Replace File* function.
- Right click on the *operation* terminal on the *Open/Create/Replace File* function and create an *open* constant. This control directs the VI to open that previously created file.
- Wire the *refnum* out terminal on the *Open/Create/Replace File* function to the *refnum* terminal on the *Set File Position* function.
- Right click on the *offset (in bytes)* terminal on the *Set File Position* function and create a constant with value 0. Then, right click on the *from* terminal on the *Set File Position* function and specify *end*. Together, these 0 and end constants specify that that any addition to the file be appended to end of the file.

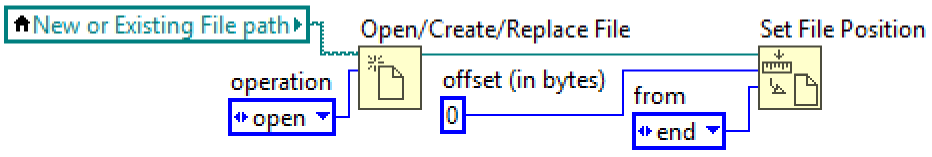


Figure 24 *Use the **Quick Drop** functionality to add a new* Open/Create/Replace File *function and a* Set File Position *function to the second frame of the sequence discussed in Step 17. When wired as shown, the file specified by the* New or Existing File path *local variable is opened, and the preparations are made to append new content to the end of the file.*

20. Write the Displacement Values to the Text File

The code from Step 19 is extended to write each displacement value acquired from the DM713 micrometer to the user-specified text file. This is done by first converting the displacement value from numerical to string representation, appending a return character as a delimiter, and then writing the result to the end of the file. After this string is written to the file, the terminal return character advances the end of the file to the first character on the next row. Due to this, each new displacement value is appended to a new line at the end of the file, so that the data are saved in column format.

To the right of the code shown in Figure 24, add a *Number to Fractional String* function, a *Format Into File* function, and a *Close File* function. As shown in Figure 25,

- Wire the *output* terminal of the *Scan From String* function, which was first discussed in Step 11, to the *number* terminal of the *Number To Fractional String* function.
- Wire the *F-format string* terminal of the *Number To Fractional String* function to the input terminal of the Format Into File function.
- Wire the *refnum out* terminal from the *Set File Position* function to the *input file (use dialog)* terminal on the *Format Into File* function.
- Right click on the *format string* terminal of the *Format Into File* function and create a constant. Enter ***%s\n*** as the string.
- Wire the *output file refnum* terminal of the *Format Into File* function to the *refnum terminal* of the *Close File* function.

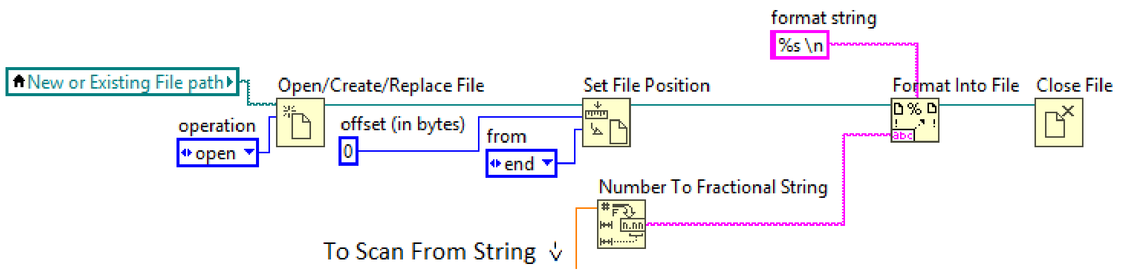


Figure 25 *The code from Figure 24 is extended so that it writes the displacement value from the micrometer to the end of the file. The displacement value is converted to a string, a return character is appended to it as a delimiter, the result is written to the end of the file, and the file is closed.*

21. Add a Timestamp to the Displacement Value Recorded to the Text File

The code from Step 20 is further extended to record a timestamp with each displacement value added to the text file. This is done by concatenating the string representation of the displacement value and the timestamp.

Below the *Number to Fractional String* function shown in Figure 25, add a *Format Date/Time String* function. Add a *Concatenate Strings* function to the right of both. As shown in Figure 26,

- Remove the wire between the *F-format string* terminal on the *Number To Fractional String* function and the *input* terminal on the *Format Into File* function.
- Right click the *time format string* terminal of the *Format Date/Time String* function and create a constant. Enter ***%I:%M:%S %p %A, %b %d, %Y*** as the string. This will record the time as: Hour(12 Hour Clock):Minute:Second am/pm Weekday, Month Day, Year
- Wire the *F-format string* terminal on the *Number To Fractional String* function to one string terminal on the Concatenate Strings function. Wire the date/time string terminal on the Format Date/Time String function to the other terminal on the Concatenate Strings function.

- Wire the *concatenated strings* terminal on the *Concatenated Strings* function to the *input* terminal on the *Format Into File* function.

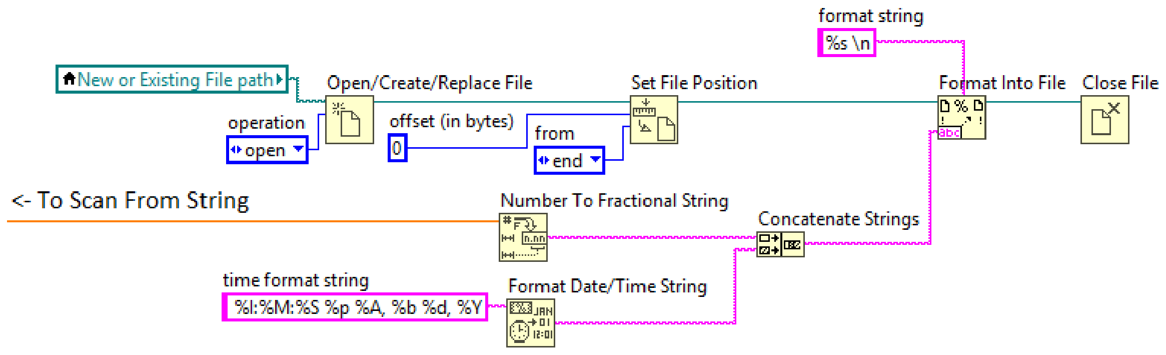


Figure 26 *A button will be used to log individual data points to a file. Drag and drop a button from the toolbox into the form window. Specify the identifier for this button object in the code by entering the identifier into the* ***(Name)*** *field under the* ***Design*** *heading in the* ***Properties*** *window.*

22. Require the User to Specify Which Displacement Values to Record to File

If the VI were run without making further modifications, every displacement value acquired from the DM713 micrometer would be timestamped and appended to the user-specified file. If an application requires only user-selected positions to be logged, an additional *Case Structure* can be used to trigger the addition of a timestamped data value to the file. To obtain the code shown in Figure 27 and Figure 28,

- Within the second frame of the *Flat Sequence* discussed in Step 17, add a new *Case Structure* within the false case of the existing *Case Structure*.

- Move all of the code shown in Figure 26 to the true case of the new *Case Structure*, and move the *Micrometer Position* indicator to the right of the new *Case Structure.*

- Make the necessary adjustments to the wires from the *output* terminal of the *Scan From String* function so that: a wire from that *output* terminal tunnels into and out of the new *Case Structure*, both tunnels are directly connected inside the *Case Structure* by a wire and the *number* terminal on the *Number To Fractional String* function is connected to that wire, and the output tunnel is wired to the *Micrometer Position* indicator.

- Right click the green question mark on the left side of the new *Case Structure*, and select *Create a Control* from the menu.

- Toggle the new *Case Structure* to false, and directly wire the two tunnels.

When the VI is run with this modification in place, data logging will occur when the Boolean control, whose creation was described in the last bullet point, is in the true state. The code added during this step does not affect the display of the micrometer position. That value is continually updated and displayed by the indicator on the **Front Panel**, but the displacement value will be logged to file only when the user toggles the Boolean control to true.

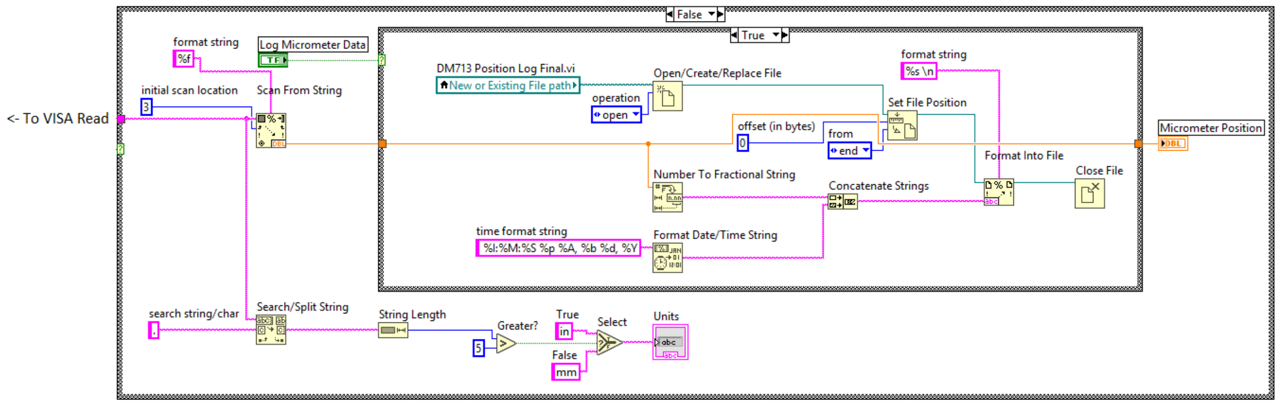


Figure 27 *A* Case Structure *added to the false case of the existing* Case Structure *prevents the enclosed code from executing unless the Boolean control called* Log Micrometer Data *is true. This modification to the VI logs a data value to the user-specified file only if the user toggles a button on the* ***Front Panel****.*

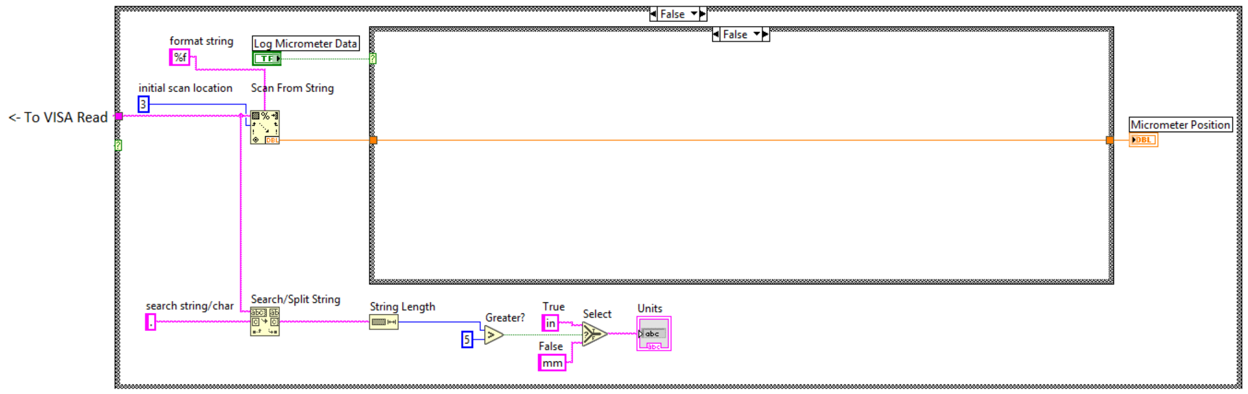
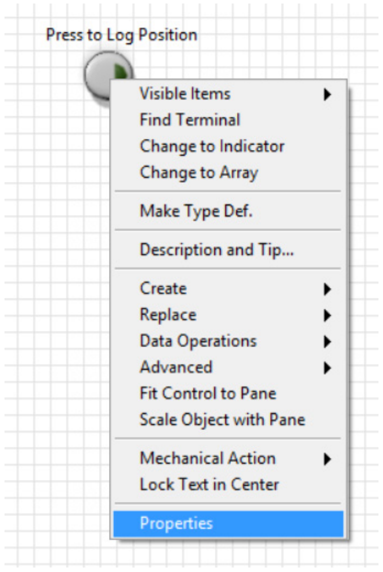


Figure 28 *The false case of the new* Case Structure *is empty, except for the wire connecting the two tunnels. When the Boolean value wired to the* case selector *terminal on the* Case Structure *is false, the displacement value is transferred to the* ***Front Panel*** *display but no other action is performed.*

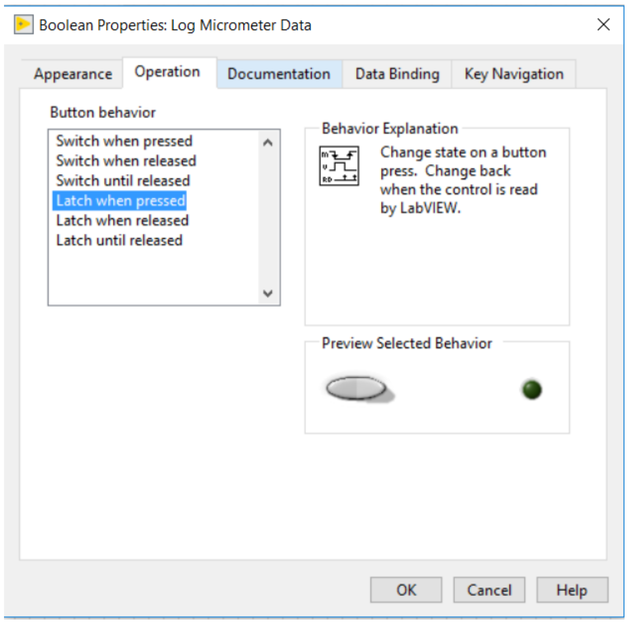
23. Log One Timestamped Position Per Button Press

The requirement for this example application is that only one displacement value be logged each time the button, which was added in Step 22 and is connected to the *case selector* of the new *Case Structure*, is clicked. This behavior can be obtained by specifying that the button latch when clicked. In LabVIEW, the value of the control changes when it latches, and it immediately reverts back to its default value after the VI reads the control.

Switch to the Front Panel and right click on the Boolean control created in Step 22. Select Properties, as shown in Figure 29(a), then specify Latch when pressed on the Operation tab, as shown In Figure 29(b), and click OK.



(a)



Boolean Properties: Log Micrometer Data

Button behavior: Latch when pressed

Behavior Explanation: Change state on a button press. Change back when the control is read by LabVIEW.

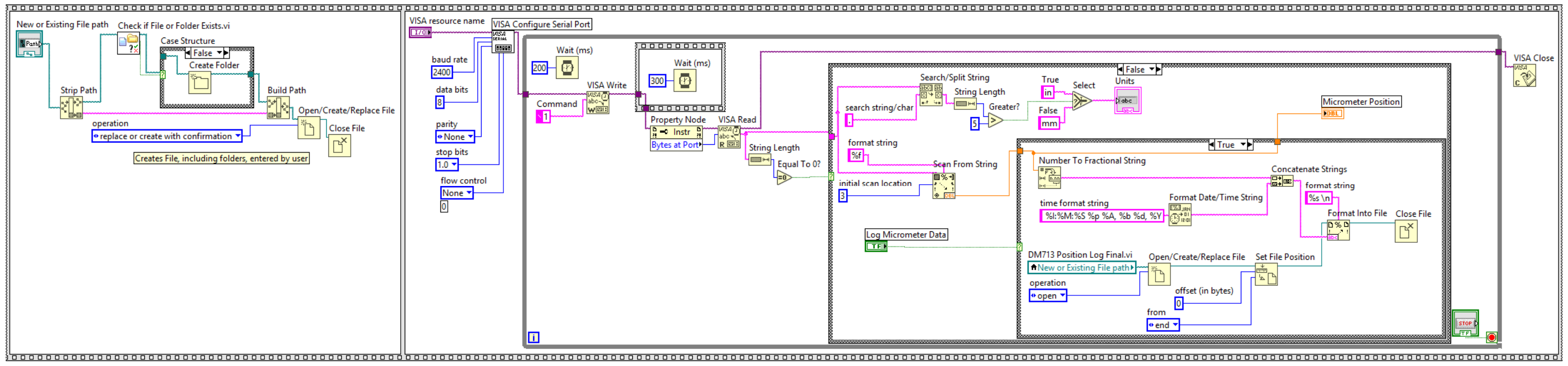
(b)

Figure 29 *Specify the Boolean control created in Step 22 latches when pressed so that each click of the control results in a single displacement value being logged to the user-specified file*

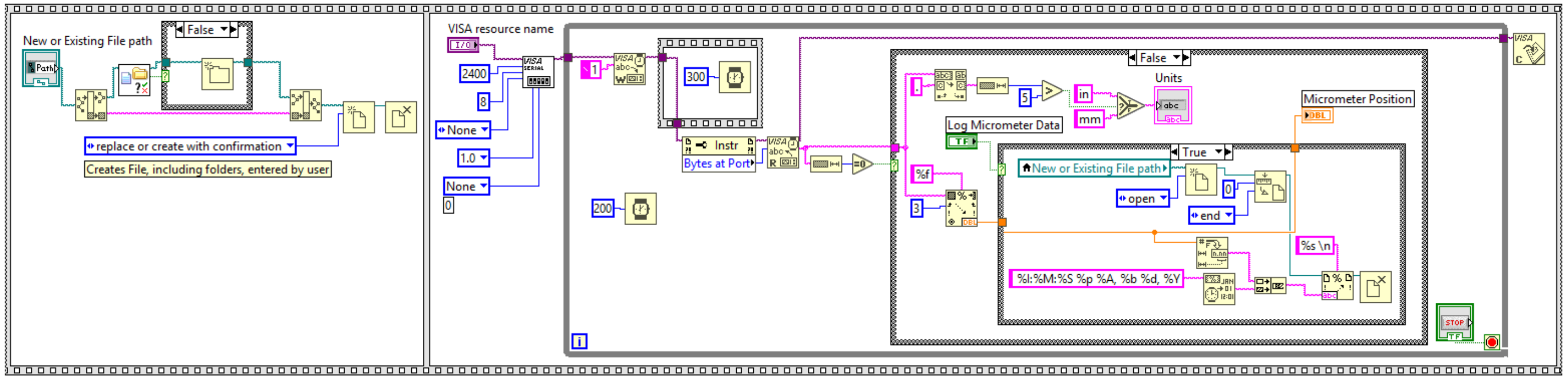
.

3 Complete Program Code

Figure 30 shows the VI created in Section 2 in its entirety. Figure 30(a) includes each function and subVI's labels. The more compact depiction shown in Figure 30(b) omits the labels.



(a)


New or Existing File path
False
replace or create with confirmation
Creates File, including folders, entered by user
VISA resource name
2400
8
None
1.0
None
0
1
300
Instr
Bytes at Port
200
False
Units
5
in
mm
Micrometer Position
Log Micrometer Data
%f
3
True
New or Existing File path
open
0
end
%s \n
%I:%M:%S %p %A, %b %d, %Y
STOP

(b)

Figure 30 *The program created in Section 2 in its entirety shown with labels included (a) and with labels omitted (b).*