Using Touch Screens with LabVIEW through Modbus Serial Devices

You can use LabVIEW VIs on your host PC or NI CVS-145x device to create a Modbus serial slave device for a remote master device or a Modbus serial master device for remote slave devices. This document provides detailed steps on how to build a slave device in LabVIEW to interact with a Modbus enabled touch screen, such as the QSI Corporation QTERM-G70 and the AutomationDirect EZTouch Panel EZ-S6C-FST. This document also includes steps for developing a LabVIEW VI to send out messages as a master device.

Modbus Overview

Modbus is an application layer messaging protocol for client/server communication between devices connected on a bus or network. Modbus can be implemented using asynchronous serial transmission to communicate with touch screens, PLCs, and gateways to support other types of industrial buses.

The Modbus Serial Line protocol is based on a master/slave architecture. An address, ranging from 1 to 247, is assigned to each slave device. Only one master is connected to the bus at any given time. Slave devices do not transmit information unless a request is made by the master device, and slave devices cannot communicate to other slave devices.

Information is passed between master and slave devices by reading and writing to registers located on the slave device. The Modbus specification distinguishes the use of four register tables, each capable of 65,536 items and differentiated by register type and read-write access. Register address tables do not overlap in this implementation of LabVIEW.

<table>
<thead>
<tr>
<th>Tables</th>
<th>Object Type</th>
<th>Type of Access</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Inputs</td>
<td>Single Bit</td>
<td>Read Only</td>
<td>Master can read only. Only the slave device can change its register values.</td>
</tr>
<tr>
<td>Coils</td>
<td>Single Bit</td>
<td>Read-Write</td>
<td>Both master and slave can read and write to these registers.</td>
</tr>
<tr>
<td>Input Registers</td>
<td>16-Bit Word</td>
<td>Read Only</td>
<td>Master can read only. Only the slave device can change its register values.</td>
</tr>
<tr>
<td>Holding Registers</td>
<td>16-Bit Word</td>
<td>Read-Write</td>
<td>Both master and slave can read and write to these registers.</td>
</tr>
</tbody>
</table>
LabVIEW includes the Class 0 and Class 1 functions of the Modbus Specification. A master device can use these requests to access the Modbus register tables.

Class 0:
- Read Multiple Registers
- Write Multiple Registers

Class 1:
- Read Coils
- Read Input Discretes
- Read Input Registers
- Write Coil
- Write Single Register
- Read Exception Status

For more information about the Modbus specification, visit www.modbus.org.

**Required Hardware and Software**

You must have the following hardware and software installed to perform the procedures outlined in this document:

- Host PC and/or NI CVS-145x device
- LabVIEW version 7.1 and later
- Refer to the LabVIEW documentation for information about system requirements for the host PC and/or CVS-145x device.
- LabVIEW Real-Time 7.1 installed on the host PC if you are using a CVS-145x device to communicate with the Modbus device.
- NI-VISA 3.0.1 installed on the host PC. NI-VISA is the driver software that enables you to use the serial port on the PC to configure instrumentation systems with serial interfaces. Visit ni.com/visa to download the NI-VISA driver software.
- VISA Server 3.0.0 installed on the host PC. The VISA server acts as a communication server between the PC and the Modbus device. Visit ni.com/visa to download the VISA Server
- Measurement & Automation Explorer (MAX) version 3.1.1 or later.

**Using LabVIEW to Interface with a Modbus Device**

National Instruments provides LabVIEW VIs you can use to format and write values to an internal Modbus register. The VIs you use to interface with a Modbus device are included in the NIModbus.llb. This library includes the Modbus Serial Slave Demon VI, which you use to run in the background to monitor the serial port for commands issued by a remote master device. When the Modbus Server receives a Modbus request, it performs the specified action, which is to read or write to the Modbus registers, and sends a response to the master.
LabVIEW as a Modbus Slave Device

Modbus Slave Device Setup

Complete the following steps to set up a slave device, such as an NI CVS-145x device or PC, to use LabVIEW to communicate with a Modbus master device, such as a touch screen:

1. Initialize the local Modbus registers.
2. Set up and initialize the COM port.
3. Start the Modbus serial slave server.
4. Read and write to the local Modbus registers.
5. Stop the Modbus Serial Slave Demon and close VISA Sessions.

Initialize the Local Modbus Registers

At the start of a Modbus application, the four local register tables must be created in memory by LabVIEW. Initialize these registers using the MB Registers Manager VI. This VI is a multipurpose VI you can use to initialize, read, and write to any of the registers.

Complete the following steps to configure the MB Registers Manager VI:

1. Go to All Functions » Select a VI.
2. In the dialog box, select NIModbus.llb » MB Registers Manager.vi.
3. Right-click the Function terminal and select Create » Constant.
4. Select Init in the constant.

Tip The other inputs are not necessary for initializing the registers.
Set Up and Initialize the COM Port

Use the MB Serial Init VI to specify which COM port the master device is connected to and to set the serial communication settings.

Complete the following steps to configure the MB Serial Init VI:

1. Go to All Functions » Select a VI.
2. In the dialog box, select NIModbus.llb » MB Serial Init.vi in the LabVIEW block diagram.
3. Create controls or constants for all of the inputs to match the serial settings of the master device.
   - VISA resource name – Select the COM port that the master device is connected to.
   - Baud Rate – Default is 9600
   - Parity – Default is None
   - Flow Control – Default is None
   - Timeout – Default is 10000 ms
   - Mode – Default is RTU

**Note** These serial settings must match the serial settings of the master device.

**Note** Select RTU mode for the QTERM-G70 and EZTouch Panel. The RTU (remote terminal unit) transmission mode is available with all Modbus devices.

Set Up the Modbus Serial Slave Server

The Modbus Serial Slave Demon VI works as the server between the master device and the local Modbus registers. The VI waits for incoming commands from the master device on the COM port and processes them. This VI can be launched dynamically from the main VI to run in the background.

Load the Slave Server Dynamically

Complete the following steps to configure the Modbus Serial Slave VI:

1. Drop the All Functions » Application Control » Open VI Reference VI in the block diagram.
2. Right-click the vi path terminal and select Create » Constant. Copy and paste the path to the MB Serial Slave Demon.vi, which is included in the NIModbus.llb, into the constant.

**Note** When targeting a CVS-145x device, upload the NIModbus.llb, which contains the MB Serial Slave Demon VI, to the CVS-145x device flash. In the Open VI Reference, specify the path of the MB Serial Slave Demon VI on the CVS-145x device.

**Note** When building a Modbus slave application into an executable, you must add the MB Serial Slave Demon.vi as a dynamic VI in the Source Files tab in the LabVIEW Application Builder. National Instruments suggests you specify a relative path for the MB Serial Slave Demon VI where the path is relative to the calling VI.
Set the Control Values for the Slave Server

Use invoke nodes to set values for the MB Serial Slave Demon VI with the action Set Control Value [Variant].

Complete the following steps to set values for MB Serial Slave Demon VI:

1. Drop the All Functions»Application Control»Invoke Node function on the block diagram.
2. Wire the reference output of the Open VI Reference function to the appropriate input on the Invoke Node function.
3. Right-click the top portion of the Invoke Node and select Methods»Set Control Value [Variant].
4. Set up the Modes:
   a. Right-click Control Name and select Create»Constant.
   b. Enter Modes in the string constant input.
   c. Drop an All Functions»Numeric»Enum Constant on the block diagram.
   d. Enter RTU as the first option.
   e. Right-click the Enum Constant and select Add Item After.
   f. Enter ASCII as the second option.
   g. Select the RTU item in the Enum Constant.
   h. Drop an All Functions»Array»Build Array function on the block diagram, and wire the Enum Constant into it.
   i. Wire the Build Array function to the Value property on the Invoke Node function.
5. Repeat steps 1 through 4 for each of the items below, ignoring the Modes item because you already added it.

Tip Refer to the All Functions»Select a VI»NIModbus.llb»MB Serial Example Slave.vi for an example of how these Invoke Node functions are configured.

- VISA resource names – Array of strings. The port is the same as the COM port specified with MB Serial Init, which is the COM port that the slave is connected to.
- Modes – Array of enums, RTU, or ASCII. The mode is the same as the mode specified with MB Serial Init for each slave device.
- Timeouts – Array of U32 values. These are the timeout values for reading serial commands, in milliseconds, for each slave device.
- Slave Address – Numeric U8. Assign a slave address that is between 1 and 247 to identify LabVIEW to the master device.

Tip When you repeat step 2 after you add the first Invoke Node function, wire the reference output and error out outputs from the initial Invoke Node to the subsequent Invoke Nodes.

Tip You do not need to use the Build Array function for the Slave Address item because it is not an array of values.
Set Up the Global Variable

Add the global variable, Slave Demon Running, to the block diagram and initialize the global variable to True. The global variable is used later to stop the MB Serial Slave Demon VI remotely.

Complete the following steps to add the global variable:
1. Go to All Functions » Select a VI.
2. In the dialog box, select NIModbus.llb » MB Globals.vi.
3. Click the MB Globals variable, and select Slave Demon Running.
4. Right-click the Slave Demon Running global variable, and select Create » Constant. Set the constant to True.

Run the Modbus Serial Slave Server VI

1. Drop the All Functions » Application Control » Invoke Node function on the block diagram.
2. Wire the reference output of the previously dropped Invoke Node function or Open VI Reference function to the reference input of the Invoke Node function you just dropped.
3. Right-click the bottom portion of the Invoke Node and select Methods » Run VI.
4. In the Invoke Node function, right-click Wait until done, and then select Create » Constant. Set the constant to False.
5. In the Invoke Node function, right-click Auto Dispose Ref, and then select Create » Constant. Set the constant to False.

Note   You must initialize the local Modbus registers before you run the slave demon.

Reading and Writing to Local Modbus Registers

Use the MB Registers Manager VI to read from and write to the local Modbus registers. Only U16 words can be written to and read from the holding and input registers. If data types other than U16 are necessary, you must programmatically format the other data types to fit the U16 register structure. Refer to the LabVIEW documentation for information about formatting data types.

You can write and read only Boolean values from the discrete inputs and coils registers.

Register Names and Addresses

In most Modbus device configuration software, you must enter a name for the register you want to use. Per Modbus convention, the register address of the slave device, which is the PC or CVS-145x device in this case, is calculated by subtracting 1 from the register name that you specify in the master device configuration software. When you are reading from or writing to a register on a Modbus device, be aware that LabVIEW expects register addresses rather than register names. When you enter the register address to read from in LabVIEW, the value you enter must be the register name you defined in the Modbus device configuration software minus 1.

The QTERM-G70 uses this naming convention. For example, a holding register name defined as 2 for the QTERM-G70 translates to register address 1 in the LabVIEW Holding Registers table.

<table>
<thead>
<tr>
<th>QTERM-G70</th>
<th>Holding Register Name = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabVIEW</td>
<td>Holding Register Address = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QTERM-G70</th>
<th>Holding Register Name = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabVIEW</td>
<td>Holding Register Address = 1</td>
</tr>
</tbody>
</table>
Some Modbus devices use register tables and addresses as shown in the following illustration.

Because the register name and address tables do not overlap in LabVIEW, ignore the first digit, which indicates the register table. In LabVIEW, you select the register table by name from a drop-down list. To determine the start address in LabVIEW, enter the number after the register table number. For example, a register name defined as 0x40000 in a Modbus configuration device translates to register address 0 in the LabVIEW Holding Registers table.

Some devices, such as the EZTouch Panel, require both of these concepts. You must subtract 1 from the register name that you specify in the master device configuration software and ignore the first digit of the start address to ensure proper register addressing.

For example, a register name defined as 0x40008 in the EZTouch Panel translates to register address 7 in the LabVIEW Holding Registers table.

<table>
<thead>
<tr>
<th>Modbus Device</th>
<th>Holding Register Name = 0x40000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabVIEW</td>
<td>Holding Register Address = 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EZTouch Panel</th>
<th>Holding Register Name = 0x40008</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabVIEW</td>
<td>Holding Register Address = 7</td>
</tr>
</tbody>
</table>

### Reading from the Local Modbus Registers

Complete the following steps to set up the MB Registers Manager VI to read from the local Modbus registers:

1. Drop the All Functions » Select a VI » NI Modbus.llb » MB Registers Manager.vi on the block diagram.
2. Right-click the Function terminal, and select Create » Constant. In the constant, select Read to read a specific register, or select Read All to read all 65536 registers.
3. Right-click the Table terminal, and select Create » Constant. In the constant, select the register table you are reading from.
4. Right-click the Start Address terminal, and select Create » Constant. Enter the local register address at which to start reading data.

   The Start Address terminal is necessary only when you have set the Function terminal to Read rather than Read All.
5. Right-click the **Length** terminal, and select **Create»Constant**. Specify the number of registers to read. The **Length** terminal is necessary only when you have set the **Function** terminal to **Read** rather than **Read All**.
6. Right-click the **Words Out** terminal, and select **Create»Indicator**. This indicator displays the U16 data read in from the holding and/or input registers.
   Use the **Words Out** terminal only when you select either **Input Registers** or **Holding Registers** in the **Table** terminal.
7. Right-click the **Bits Out** terminal, and select **Create»Indicator**. This indicator displays the Boolean data read in from the discrete input and/or coils registers.
   Use the **Bits In** terminal only when you select either **Discrete Inputs** or **Coils** in the **Table** terminal.

### Writing to the Local Modbus Registers

Complete the following steps to set up the MB Registers Manager VI to write to the local Modbus registers:

1. Drop the **All Functions»Select a VI»NIModbus.llb»MB Registers Manager.vi** on the block diagram.
2. Right-click the **Function** terminal, and select **Create»Constant**. In the constant, select **Write**.
3. Right-click the **Table** terminal, and select **Create»Constant**. In the constant, select the register table you are writing to.
4. Right-click the **Start Address** terminal, and select **Create»Constant**. Enter the local register address at which to start writing data.
5. Right-click the **Length** terminal, and select **Create»Constant**. Specify the number of elements from the array to write to the local registers.
6. Wire the appropriate U16 data to the **Words In** terminal. This is the data you send to the holding and/or input registers.
   Use the **Words In** terminal only when you select either **Input Registers** or **Holding Registers** in the **Table** terminal.
7. Wire the appropriate Boolean data to the **Bits In** terminal. This is the data you send to the discrete inputs or coils registers.
   Use the **Bits In** terminal only when you select either **Discrete Inputs** or **Coils** in the **Table** terminal.

### Stopping the Modbus Serial Slave Demon and Closing VISA Sessions

The final steps of your Modbus application must properly close all resources that were opened.

Complete the following steps to close resources:

1. Go to **All Functions»Select a VI**.
2. In the dialog box, select **NIModbus.llb»MB Globals.vi**.
3. Click the MB Globals variable, and select **Slave Demon Running**.
4. Right-click the **Slave Demon Running** global variable, and select **Create»Constant**. Set the constant to False.
5. Drop the **All Functions»Application Control»Close Reference VI** on the block diagram.
6. Wire the **vi reference** output from the Open VI Reference VI that you added in the **Set Up the Modbus Serial Slave Server** section to the **reference input** on the Close Reference VI.
7. Drop the **All Functions»Instrument I/O»Serial»VISA Close VI** on the block diagram.
8. Wire the **VISA resource out** terminal of the MB Serial Init VI to the **VISA resource name** terminal of the VISA Close VI.
LabVIEW as a Modbus Master Device

Modbus Master Device Setup

The master device in a Modbus system does not contain any Modbus registers. The Modbus device sends read and write commands to the slave devices on the bus.

Complete the following steps to set up a master device, such as a CVS-145x device or PC, to use LabVIEW to communicate with Modbus slave devices on the same serial bus:

1. Set up and initialize the COM port.
   a. Drop the All Functions » Select a VI » NI Modbus.llb » MB Serial Init.vi in the LabVIEW block diagram.
   b. Use the Create » Constant item in the right-click menu to create controls or constants for all of the inputs, and modify the serial settings to ensure that the master and slave devices match:
      - VISA resource name – Select the COM port that the slave devices are connected to.
      - Baud Rate – Default is 9600
      - Parity – Default is None
      - Flow Control – Default is None
      - Timeout – Default is 10000 ms
      - Mode – Default is RTU

  📝 Note These serial settings must match the slave device you are connected to.

Sending Read and Write Commands to Modbus Slave Devices

Use the MB Serial Master Query VI to send both read and write commands to slave devices over the serial bus.

Complete the following steps to set up the MB Serial Master Query VI to read commands from the Modbus device:

1. Go to All Functions » Select a VI.
2. In the dialog box, select NI Modbus.llb » MB Serial Master Query.vi.
3. Set up the Modbus Command Cluster Input elements as appropriate:
   - Function Code – Select the action you want to perform.
   - Starting Address - Indicate the register address to start reading/writing data.
   - Quantity – Specify the number of registers to be read or written to. This element is not used when either Write Single Coil or Write Single Register is selected in Function Code.
   - Data – This is the U16 array of data to be written to the slave holding registers. This element is used only when either Write Single Register or Write Multiple Registers is selected in Function Code.
   - Discrete – This is the Boolean array of data to be written to the slave coil registers. This element is used only when either Write Single Coil or Write Multiple Coils is selected in Function Code.
4. Set up the Serial Parameters Cluster Input elements as appropriate:
   - Mode – Select RTU or ASCII.
   - Slave Address – Specify which slave device to read from or write to.
5. Read the Registers Cluster Output elements as appropriate:
   - Registers – Outputs an array of U16 values when Read Holding Registers or Read Input Registers is selected in Function Code.
   - Discrete – Outputs an array of Boolean values when Read Coils or Read Discrete Inputs is selected in Function Code.

Release the COM Port
Complete the following steps to close the VISA session opened by the MB Serial Init VI:
1. Drop the All Functions>Instrument I/O>Serial>VISA Close VI on the block diagram.
2. Wire the VISA resource out terminal of the MB Serial Init VI to the Visa resource name terminal of the VISA Close VI.

LabVIEW Modbus Examples
The NI Modbus.lib includes two example programs: MB Serial Example Master and MB Serial Example Slave. If two COM ports are available on your PC, you can run both VIs to implement a Modbus master device and Modbus slave device simultaneously on the same computer. Use a null modem serial cable to connect the two COM ports.

LabVIEW As a Slave
The MB Serial Example Slave continuously reads the first four registers of its local coil and holding register table. The VI writes Boolean and U16 values to the first four registers of its local discrete register table and input register table, respectively.

The MB Serial Example Slave VI contains functions to open the MB Serial Example Slave VI and open a VI reference, which are wired into a series of sequence structures, and a while loop that are used to achieve its functionality.

The outer sequence structure handles the following functions:
   - Initializing the modbus registers
   - Setting up the serial slave
   - Using a while loop to read and write from registers
   - Write to discrete input registers
   - Write to input registers
   - Read from coils and holding registers
   - Include a delay
   - Closing the serial slave demon
   - Closing the VISA resources and the VI references

LabVIEW As a Master
The MB Serial Example Master VI writes Boolean and U16 values to the first four registers of the slave coil table and holding table, respectively. The VI continuously reads the first four registers of the slave discrete register table and input register table.

The MB Serial Example Master VI contains a series of sequence structures and a while loop that are used to achieve its functionality.
The outer sequence structure handles the following functions:

- Initializing the VISA resource
- Using a while loop to read and write from registers
- Write to coils input registers
- Write to holding registers
- Read discrete input registers
- Read input registers
- Include a delay
- Closing the VISA resources and the VI references

### Running the Examples

The MB Serial Example Slave VI reads and writes to local registers, and the MB Serial Example Master VI polls the local registers on the slave device. When the MB Serial Example Master VI is reading registers, it polls the registers on the slave device and displays the data on the master device. The MB Serial Example Master VI also writes to registers on the slave device.

Complete the following steps to run the MB Serial Example Slave VI and MB Serial Example Slave VI:

1. Ensure that the COM ports you want to use are not currently in use.
   - National Instruments suggests you exit Vision Builder for Automated Inspection (Vision Builder AI) before running these examples to ensure that the COM port you want to use is not already being used by a Vision Builder AI script.
2. Launch the example VIs:
   a. Launch LabVIEW.
   b. Click **Open** and navigate to NIModbus.llb»MB Serial Example Slave VI.
   c. Click **Open** again and navigate to NIModbus.llb»MB Serial Example Master VI.
3. Select the appropriate COM port in the **VISA resource name** control for each of the VIs.
4. Click **Run** in each of the VIs.
5. Flip the switches in the Discrete Inputs section of the MB Serial Example Slave VI, or flip the switches in the Coils to Write section of the MB Serial Example Master VI.
   - The corresponding LEDs on each example VI light to indicate which registers are set to true.
6. Enter values in the Input Registers section of the MB Serial Example Slave VI, or enter values in the Registers to Write section of the MB Serial Example Master VI.
   - These changes are displayed in the Holding Registers section of the slave example VI and the Input Registers section of the master example VI.