Using Touch Screens with Vision Builder AI through Modbus

Introduction

National Instruments Vision Builder for Automated Inspection (Vision Builder AI) can be used as a Modbus slave device for a remote master device when the devices are connected by the serial port. This document provides detailed steps for configuring Vision Builder AI to interact with a Modbus-enabled touch screen, such as the QSI Corporation QTERM-G70 and the AutomationDirect EZTouch Panel EZ-S6C-FST.

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 complement 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 name and register address tables do not overlap in Vision Builder AI.

<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>
Vision Builder AI works with 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

Tip 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
- Vision Builder AI version 2.5 or later
  - Refer to the Vision Builder AI documentation for information about system requirements for the host PC and/or CVS-145x device.
- NI-VISA 3.0.1 installed on the host PC. NI-VISA is the driver software for using 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.
Interfacing with a Modbus Device Using Vision Builder AI

Vision Builder AI includes an inspection step you can use to format and write previous results to internal Modbus registers. Vision Builder AI also includes the Modbus Server, which runs in the background to monitor a serial port for commands issued by a remote master device. When the Modbus Server receives a Modbus request, it performs the specified action and sends a response to the master.

With touch screens that use the Modbus protocol, users can view and input data on the factory floor without the use of a keyboard or mouse. For example, the touch screen can be used to display inspection results, or you can use it to modify inspection parameters on the fly. Additionally, you can use the touch screen to select an inspection.

Complete the following steps to set up a slave device to use Vision Builder AI to communicate with a Modbus master device:

1. Configure the serial port settings to communicate with the master device, which is the touch screen.
2. Use the Modbus Device Manager to set up and run the Modbus Server.
3. In the Vision Builder AI script, add a step to read from or write to the local Modbus registers.
4. Perform product selection using the Modbus device.

Refer to the following sections for more information about these procedures.

Connect and Configure the Modbus Device

Before you can complete the steps in the following sections, you must connect and configure the Modbus device you want to communicate with to the host PC or CVS-145x system.

To determine how to connect and configure the Modbus device, follow the instructions in the device documentation. Note the serial port settings of the Modbus device because you will use them in the Configure the COM Port section.
Configure the COM Port

Follow the steps in the appropriate section below, depending on whether you are using a PC or a CVS-145x system to communicate with the touch screen.

Set Up the COM Port on a PC

2. Expand Devices and Interfaces.
3. Expand Ports (Serial and Parallel).
4. Select the COM port connected to the master device, which is the touch screen.
5. Click the Port Settings tab at the bottom of the MAX window.
6. Configure the settings to match the serial settings of the master device. Refer to the touch screen device documentation for information about the correct COM settings.
7. Click Save at the top of the MAX window.
Set Up the COM Port on a CVS-145x System
1. Launch Vision Builder AI, and select the appropriate CVS-145x in **Execution Target**.
2. Click **Tools»Remote Target Options**.
3. Select the **Serial Port** tab.

4. Configure the settings to match the serial settings of the master device. Refer to the touch screen device documentation for information about the correct COM settings.

5. Click **OK**.

Set Up and Run the Modbus Server
The Modbus server is a background task that waits for commands from the master device on the serial port. After a command is received, the Modbus server processes the Modbus request. Use the Modbus Device Manager in Vision Builder AI to set up and start the Modbus server.

1. In Vision Builder AI, click **Tools»Modbus Device Manager**.
2. Click New Device.

3. In the New Modbus Device dialog box, enter a **Device Name** for the master Modbus device.

4. Select the **COM Port** that the master Modbus device is connected to.

5. Select the **Mode** used by the Modbus device.

   ![New Modbus Device dialog box](image)

   **Note** Select RTU mode for the QTERM-G70 and EZTouch Panel. The RTU (Remote Terminal Unit) transmission mode is used by all Modbus devices.

6. Click **OK**.

7. Assign an address between 1 and 247 in **Vision Builder AI Modbus Slave Address**.

   **Note** Make sure that there is no other slave device with the same address on the serial bus at the same time.

8. If the server is not running, click **Start Server**.

9. Click **OK**.

**Add a Modbus Step to a Vision Builder AI Script**

Modbus steps can be added anywhere in a Vision Builder AI inspection script to read and write to the local Vision Builder AI 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 system 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 Vision Builder AI expects register addresses rather than register names. When you enter the register address to read from in Vision Builder AI, 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 Vision Builder AI Holding Registers table.

QTERM-G70 Holding Register Name = 2
Vision Builder AI Holding Register Address = 1

Some Modbus devices use register tables and addresses as shown in Figure 2.

![Modbus Register Tables and Addresses](image)

Figure 2. Modbus Register Tables and Addresses

Because the register name and address tables do not overlap in Vision Builder AI, ignore the first digit, which indicates the register table. In Vision Builder AI, you select the register table by name from a drop-down list. To determine the start address in Vision Builder AI, 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 Vision Builder AI Holding Registers table.

Modbus Device Holding Register Name = 0x40000
Vision Builder AI Holding Register Address = 0

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 Vision Builder AI Holding Registers table.

EZTouch Panel Holding Register Name = 0x40008
Vision Builder AI Holding Register Address = 7
Add a Command to Read from the Local Modbus Registers

Adding this step creates a new measurement that contains the values read from the register. You can use the data read from the register in subsequent steps in the script.

Complete the following steps to read a command from the local Modbus register:

1. Launch Vision Builder AI and click Configure Inspection.
2. In the Inspection Steps palette, select the Communicate tab.
3. Select the Modbus Slave step.
4. Click Read Registers.

5. In Table, specify which register table to read.
6. In Start Address, select the address in the register table to begin reading.
   Refer to the Register Names and Addresses section for information about register addresses in Vision Builder AI.
7. Select the representation of the data to be read in Type. If a representation larger than a 16-bit register is being read, the additional register(s) are read automatically. For more information about the different data types, refer to the Data Types section of this document.
8. Enter a Name for the measurement information the inspection step is reading in from the register. This name makes reporting and metrics easier to understand.
9. Select the data type of the stored information in Type.
10. Click OK.

Add a Command to Write to the Local Modbus Registers

Complete the following steps to add a command to write data to the local Modbus register:

1. Launch Vision Builder AI and click Configure Inspection.
2. In the Inspection Steps palette, select the Communicate tab.
3. Select the Modbus Slave step.
4. Click Write Registers.
5. In the **Data to Write to Local Register** section, select the type of information to write. You can select one of the constant radio buttons, or you can select a **Measurement** from a previous Vision Builder AI step.

6. Under **Write to Local Register**, select the local register **Table** and **Start Address** to write to. Also specify the type of data you want to use in **Type**.

   Refer to the **Register Names and Addresses** section for information about register addresses in Vision Builder AI.

7. Click **OK**.

8. Click **OK** in the Modbus Slave Configuration dialog box when you are finished adding commands for this step.

### Monitoring the Local Modbus Registers

To view the four register tables in real-time, select **Tools** » **Modbus Local Registers Terminal** or click **View Local Registers** in the Modbus Slave Configuration dialog box. You can use the register update information to help you debug your application if you are not getting the expected results.

To view different register tables, select a table in the View menu.
Production Selection with Modbus

In Vision Builder AI, you can dynamically select different inspections using digital inputs or with a Modbus connection to a master device. Using Modbus product selections, you can use a master device to update the value of a Vision Builder AI register to a value specified on the master device. You can then associate this value with a saved inspection.

Complete the following steps to use Modbus to select inspections:

1. Launch Vision Builder AI and click **Configure Inspection**.
2. Click **Tools»Configure Inputs/Outputs**.
3. In **Product Selection Signal Source**, select **Modbus Slave Product Selection**.
4. Click **Setup**.

![Modbus Slave Product Selection Setup](image)

5. Under **Read Product Select Value from Local Modbus Register**, select the **Register Table** and **Start Address** to read the values from.
6. Indicate the representation of the values in **Type**.
7. Associate saved inspections with the appropriate register values by selecting the inspection from the drop-down menu.

**Note** Because the Coil and Holding registers are the only registers that a master device can write to, these are the only registers available to read from.

8. Click **OK**.
9. Click **Done**.

---

Application Note 209 10 ni.com
Complete the Production Selection Setup

Complete the following steps to complete the Production Selection setup for the host device you are using:

**Vision Builder AI on the Host PC**
1. In Vision Builder AI, click **Tools»Options**.
2. Select the **Inspection** tab.
3. Enable the **Start Inspection when Launched** option.
4. Select the **Inspection** to execute when the application is launched.
5. Click **OK**.

Note  Product selection is not activated until Vision Builder AI has been restarted.

**Vision Builder AI on the CVS-145x System**
1. In Vision Builder AI, click **Tools»Remote Target Options**.
2. Select the **Startup** tab.
3. Enable the **Start Inspection when Launched** option.
4. Specify the **Inspection** that will be executed first when the application is launched.
5. Click **OK**.

Note  Product selection is not activated until the CVS-145x has been rebooted. To reboot the device, click **Tools»Reboot Target**.

**Data Types**

Though Input and Holding registers are 16-bit, other data types, such as floating-point numbers and strings, can be used by using additional registers for each item.

Vision Builder AI works with the 984 and IEC-1131 data types referenced by the Modbus specification.

Indicate the address of the first register. Be sure to avoid accidentally reusing the additional registers used. The table below lists the applicable data types and their bit representation.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>16-Bit Registers Needed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit Unsigned Integer</td>
<td>1</td>
<td>Bit 15-0 of integer = bit 15–0 of register</td>
</tr>
<tr>
<td>16-bit Signed Integer</td>
<td>1</td>
<td>Bit 15-0 of integer = bit 15–0 of register</td>
</tr>
<tr>
<td>ASCII</td>
<td>1 register per 2 letters</td>
<td>First character = Upper byte (bits 15–8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Second character = Lower byte (bits 7-0)</td>
</tr>
<tr>
<td>Data Type</td>
<td>Count</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Floating Point</td>
<td>2</td>
<td>Intel single precision real&lt;br&gt;First register contains bits 15–0 of 32-bit number (bits 15-0 of mantissa)&lt;br&gt;Second register contains bits 31-16 of 32-bit number (exponent and bits 22-16 of mantissa)</td>
</tr>
<tr>
<td>Single Precision Unsigned Decimal</td>
<td>1</td>
<td>Values 0-9999&lt;br&gt;Bit 15-0 of integer = bit 15–0 of register</td>
</tr>
<tr>
<td>Double Precision Unsigned Decimal</td>
<td>2</td>
<td>Rarely used&lt;br&gt;Values 0-99999999&lt;br&gt;First register contains the most significant 4 digits&lt;br&gt;Second register contains the least significant 4 digits (both expressed as binary values in the range 0-9999)</td>
</tr>
<tr>
<td>BYTE</td>
<td>1</td>
<td>8-bit&lt;br&gt;Bits 7-0 of register = 7–0 of BYTE</td>
</tr>
<tr>
<td>DINT</td>
<td>2</td>
<td>32-bit&lt;br&gt;Bits 15-0 of first register = bits 15-0 of DINT&lt;br&gt;Bits 15-0 of second register = bits 31-16 of DINT</td>
</tr>
<tr>
<td>INT</td>
<td>1</td>
<td>Bits 15-0 of register = bits 15-0 of INT</td>
</tr>
<tr>
<td>REAL</td>
<td>2</td>
<td>32-bit&lt;br&gt;Bits 15-0 of first register = bits 15-0 of REAL (bits 15-0 of mantissa)&lt;br&gt;Bits 15-0 of second register = bits 31-16 of REAL (exponent and bits 22-16 of mantissa)</td>
</tr>
</tbody>
</table>
Table 3. Applicable Modbus Data Types (Continued)

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDINT</td>
<td>2</td>
<td>32-bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 15-0 of first register = bits 15-0 of UDINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 15-0 of second register = bits 15-0 of UDINT</td>
</tr>
<tr>
<td>UINT</td>
<td>1</td>
<td>Bits 15-0 of register = bits 15-0 of UINT</td>
</tr>
</tbody>
</table>

© National Instruments Corporation 13 Application Note 209