Using Logix5000 Controllers as Masters or Slaves on Modbus

Purpose of the Document

This application solution, and the associated RSLogix 5000 project files, help you use Logix5000™ controllers as Modbus™ RTU masters or slaves.

The term Logix5000 controller refers to any controller that is based on the Logix5000 operating system, such as:

- ControlLogix® controllers (1756)
- CompactLogix™ controllers (1769)
- FlexLogix™ controllers (1794)
- PowerFlex® 700S with DriveLogix™ controllers

The sample RSLogix™ 5000 projects provided with this solution use ControlLogix controllers. However, you can use the controller tags, periodic tasks and user-defined data types (described later in this document) with any Logix5000 controller.

Who Should Use This Document

This manual is intended for individuals who program applications that use Logix5000 controllers on Modbus, such as:

- software engineers
- control engineers
- application engineers

You should be familiar with:

- Logix5000 controllers
- RSLogix 5000
- Modbus RTU

IMPORTANT This application solution is intended for experienced Modbus users.

Publication CIG-AP129A-EN-P - February 2004

Before You Begin

Before you use this application solution, consider the following:

- Application Solution Disclaimer
- Controller Serial Port Unvailable as a Programming Connection If Used as a Master on Modbus
- Subset of Modbus Function Codes
- Automatic Conversion of Little Endian Data to Big Endian Data
- Connect Logix5000 Controller to RS485 Network Via 1761-NETAIC
- Sample Files in RSLogix 5000 v13 Format

Application Solution Disclaimer

All information is provided "AS IS" – No warranty or implied merchantability.

Controller Serial Port Unvailable as a Programming Connection If Used as a Master on Modbus

When you use your Logix5000 controller as a master on Modbus, as described beginning on page 5, you must change your controller's mode to User. In this case, your controller's serial port is unavailable as a programming connection.

Subset of Modbus Function Codes

This solution only supports a subset of Modbus function codes. Table 1 describes the function codes this solution supports.

Table 1

Function code:	Name:	Data level:	Description:
01	Read Coils	Bit level	This function code is used to read from 1 to 2000 contiguous status of coils in a remote device. The coils in the response message are packed as one coil per bit of the data field. 0 = OFF 1 = ON
02	Read Discrete Inputs		This function code is used to read from 1 to 2000 contiguous status of discrete inputs in a remote device. The discrete inputs in the response message are packed as one input per bit of the data field. 0 = OFF 1 = ON
05	Write Single Coil		This function code is used to write a single output to either ON or OFF in a remote device. 0 = OFF 1 = ON
15	Write Multiple Coils		This function code is used to force each coil in a sequence of coils to either ON or OFF in a remote device. 0 = OFF 1 = ON
03	Read Holding Registers	Word (16-bit) level	This function code is used to read the contents of a contiguous block of holding registers in a remote device.
04	Read Input Registers		This function code is used to read from 1 up to 120 contiguous input registers in a remote device.
06	Write Single Register		This function code is used to write a single holding register in a remote device.
16	Write Multiple Registers		This function code is used to write from 1 up to 120 contiguous registers in a remote device.

IMPORTANT This solution only supports Modbus RTU transmission mode. Modbus ASCII transmission mode is not supported.

The Modbus protocol specification is available at:

http://www.modbus.org

Automatic Conversion of Little Endian Data to Big Endian Data

The solution automatically converts data from the format used on Modbus format (i.e., little endian) to the format used with Logix5000 controllers (i.e., big endian). You do not need to convert any of the values you receive in your project.

Connect Logix5000 Controller to RS485 Network Via 1761-NETAIC

Use the AIC+ Advanced Interface Converter (1761-NET-AIC) to connect your Logix5000 controller to the RS485 network. For more information on how to use the 1761-NET-AIC converter, see the following publications:

- Advanced Interface Converter (AIC+) Installation Instructions, publication 1761-IN002
- Advanced Interface Converter (AIC+) User Manual, publication 1761-6.4

You can view, download or order these publications from:

http://www.theautomationbookstore.com.

Sample Files in RSLogix 5000 v13 Format

The RSLogix 5000 project files associated with this solution use RSLogix 5000, version 13. To use these files with an RSLogix 5000 version previous to 13, you must use the Import/Export function.

We recommend you use this solution with RSLogix 5000, version 13 or greater. Remember, your Logix5000 controller must use a firmware revision that matches the RSLogix 5000 version (i.e., we recommend you upgrade your Logix5000 controller to firmware revision 13.x or greater to use this solution.

Using this Application Solution

There are two parts to this solution.

This section:	Uses this RSLogix 5000 project:	Begins on:
Using Logix5000 Controllers as Masters on Modbus	ModbusMaster.ACD	page 5
Using Logix5000 Controllers as Slaves on Modbus	ModbusSlave.ACD	page 22

The ACD files give you:

- controller tags
- periodic tasks
- user-defined data types (in the ModbusMaster.ACD file only)

that you can copy into your RSLogix 5000 project to communicate with the Logix5000 controllers via Modbus RTU function codes (e.g. Read Coil Status).

Using Logix5000 Controllers as Masters on Modbus

The ModbusMaster.ACD file helps you use a Logix5000 controller as a master on Modbus. The ACD file contains:

- 11 controller tags
- 1 periodic task
- 2 user-defined data types

You must do the following tasks to use an Logix5000 controller as a master on Modbus:

1. Copy User-Defined Data Types From the ModbusMaster.ACD File to Your RSLogix 5000 Project
2. Copy Controller Tags From the ModbusMaster.ACD File to Your RSLogix 5000 Project
3. Copy ModTask From the ModbusMaster.ACD File to Your RSLogix 5000 Project
4. Configure the Controller's Communication Port
5. Configure the New Controller Tags
6. Enable the New Program
7. Verify that Your RSLogix 5000 Project is Working

Copy User-Defined Data Types From the ModbusMaster.ACD File to Your RSLogix 5000 Project

Do the following steps:

1. Start RSLogix 5000.
2. Open the ModbusMaster.ACD file. The ACD file is available at either of the following locations:
   - Allen-Bradley Channel Extranet - Download the file from the United States Channel Extranet.
   - Vendor Sample Project PDF - Access the PDF via the Help menu in RSLogix 5000.
3. Open your RSLogix 5000 project in a second instance of RSLogix 5000.
4. Copy the following user-defined data types (UDTs):

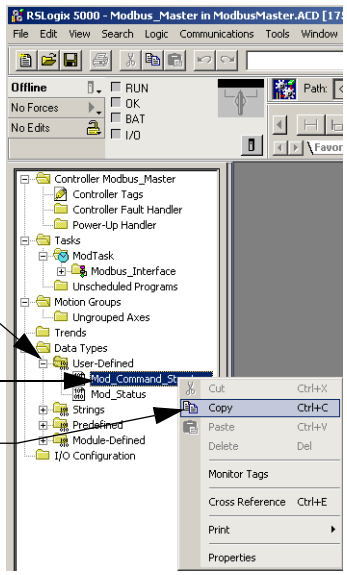
   - Mod_Command_Structure
   - Mod_Status

   from the ModbusMaster.ACD file. You can only copy one UDT at a time, so repeat the process below for the second UDT.

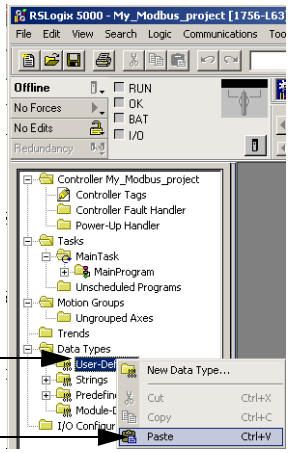
A. Open the User-Defined Data Types folder.

B. Right-click on the first UDT.

C. Click Copy.



5. Paste the UDT into your RSLogix 5000 project.



A. Right-click on the User-Defined Data Types folder.

B. Click Paste.

You can only paste one UDT at a time.

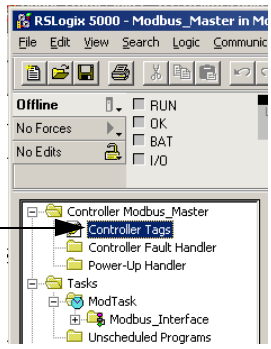
6. Repeat the previous Copy/Paste steps for the second UDT.

Copy Controller Tags From the ModbusMaster.ACD File to Your RSLogix 5000 Project

Do the following steps to copy the controller tags to your Modbus program.

1. Copy the controller tags from the ModbusMaster.ACD file.

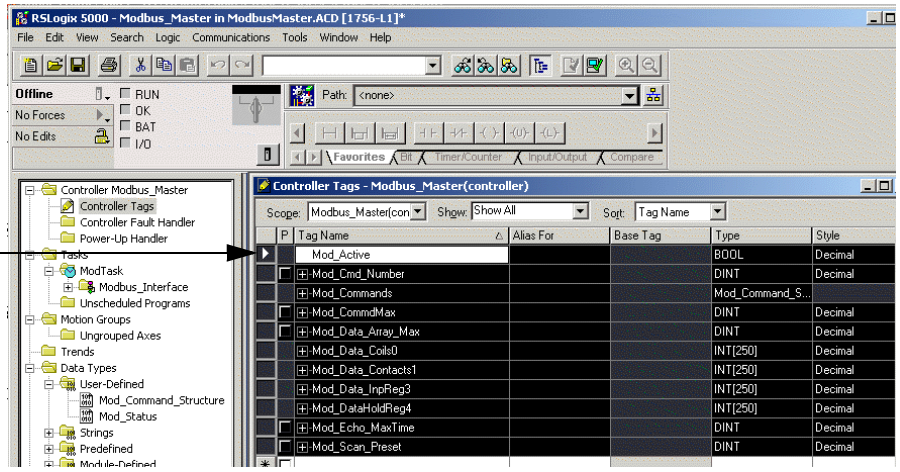
A. Double-click on Controller Tags in the Modbus master file.



B. Make sure you are in the Tag Editor.

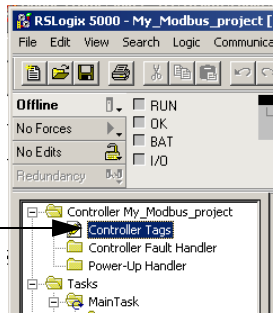
C. Highlight the 11 controller tags.

D. Copy the tags using the CTRL + C keystroke.



2. Paste the controller tags into your RSLogix 5000 project.

A. Double-click on Controller Tags in your project.

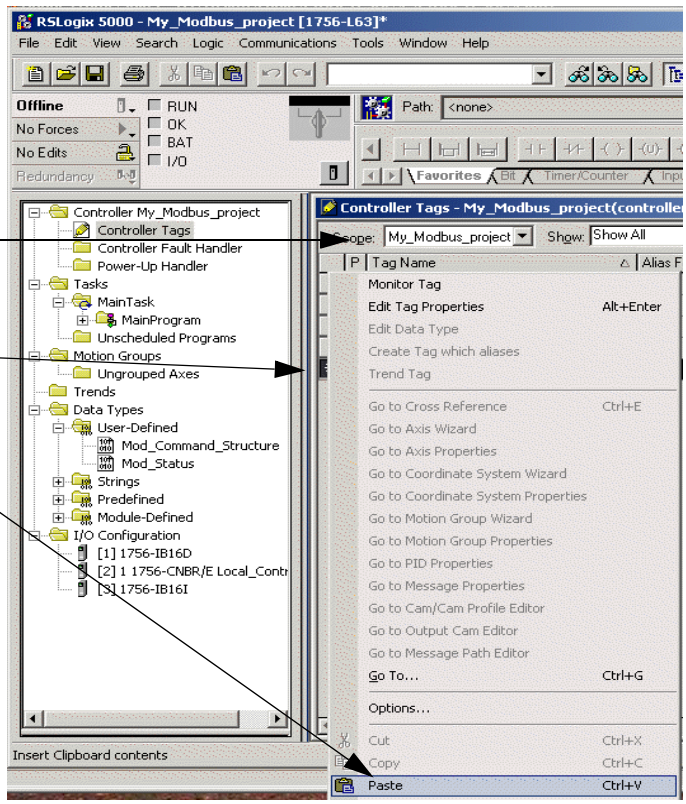


B. Make sure you are in the Tag Editor.

C. Make sure you are pasting the tags into the controller scope.

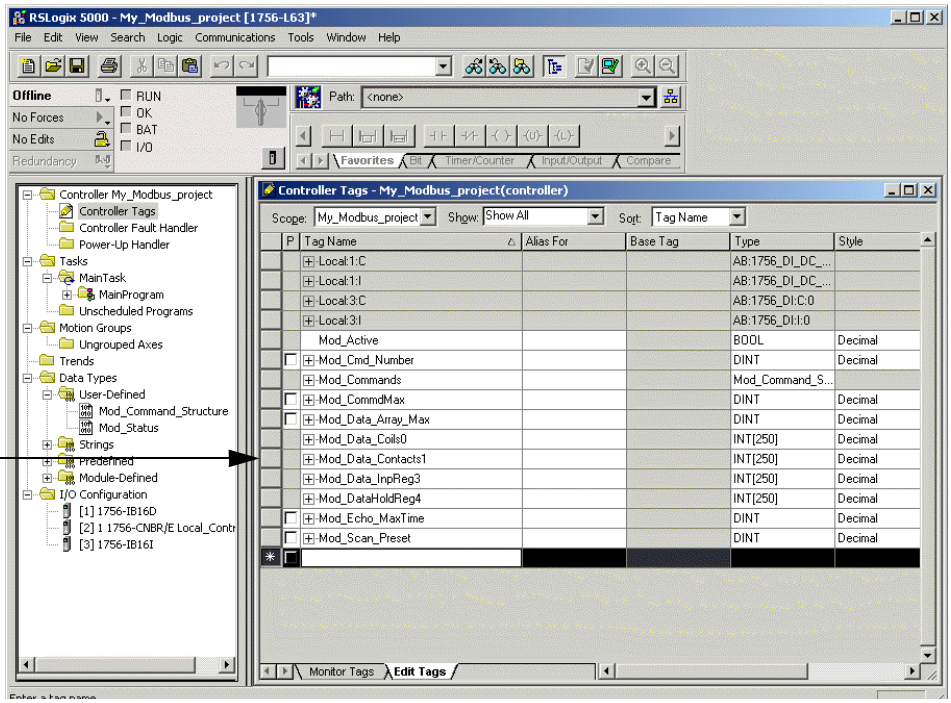
D. Right-click on the bottom tag line.

E. Click Paste in your project.



RSLogix 5000 keep the new tags together in your RSLogix 5000 project because all 11 tags use a ‘Mod_’ prefix, as shown below.

New controller tags in your Modbus file.



New Modbus Master Controller Tags

Table 4 describes the controller tags you copied from the ModbusMaster.ACD file to your RSLogix 5000 project.

Table 2

Tag name:	Tag type:	Description:	Valid values:
Mod_Active	BOOL	This tag determines whether your project runs the new program in a Logix5000 controller. This tag = 0 (i.e., the solution is disabled) by default, and you must change the tag to 1 to run the solution.	0 = Do not run the program (default) 1 = Run the program
Mod_Cmd_Number	DINT	Number of command elements (i.e., Mod_Commands tag) scanned. For example, if you set this tag = 5, the controller only scans command elements 0 through 4 and ignores command elements 5 - 39. This number should match the number of command elements used in your project.	Any value from 1 - 40
Mod_Commands	Command[40]	Command file that contains multiple user-defined configurable parameters described below:	
Mod_Commands[x].Enable	INT	Determines if the command element is enabled.	0 = Command element disabled 1 = Command element enabled always 2 = Command element enabled when controller reaches the scan number listed in the Mod_Commands[x].ScanNumber tag. 3 = Command element enabled only on the 1st scan
Mod_Commands[x].EchoReceived	INT	The master controller writes a number in this field if the command element was successfully executed. The number written matches that used in the Mod_Commands[x].Enable tag. For example, if you set the Mod_Commands[x].Enable tag = 3, the controller writes a 3 in this field if the command element was executed successfully.	0 = Command element was either disabled or did not execute successfully 1, 2 or 3 = Command element was successfully executed; typically this value is 1 because the Mod_Commands[x].Enable tag is typically = 1. Write 0 in this field before enabling the instructions to monitor the change after the Command element executes.

Table 2

Tag name:	Tag type:	Description:	Valid values:
Mod_Commands[x].ScanNumber	INT	Determines at which scan the slave executes the command element. For example, if you type a 3 in this tag, the slave only executes this command element on scan 3 (out of 10). This tag is only used if the Mod_Commands[x].Enable tag = 2. If the Mod_Commands[x].Enable tag = 0 or 1, this tag is ignored.	Any value from 0 - 12 Default = 12
Mod_Commands[x].AddressOffsetin Master	INT	Sets a word-level offset in the controller's data table when a read or write is executed. This value is added to the value of the Mod_Commands[x].Starting Address tag to determine where the master begins reading or writing data in its data table.	0 = No offset Positive nonzero number = Offset
Mod_Commands[x].SlaveAddress	INT	Designates the node number of the slave where the controller writes data to and reads data from.	0 - 255
Mod_Commands[x].FunctionCode	INT	Designates the function code of the service commanded.	**Bit level function codes** 01 = Read Coil Status 02 = Read Input Status 05 = Write Single Coil 15 = Write Multiple Coils **Word level function codes** 03 = Read Holding Registers 04 = Read Input Registers 06 = Write Single Register 16 = Write Multiple Registers For a full description of these function codes, see Table 1 on page 3.

Table 2

Tag name:	Tag type:	Description:	Valid values:
Mod_Commands[x].Starting Address	INT	This value is added to the address offset (i.e., Mod_Commands[x]. AddressOffsetinMaster tag) to determine the starting address when the master reads or writes data in its data table. Depending on the command element's function code, this value may be in bits or words. For example, if this tag = 5 in a command element that uses function code 01 (bit level code - Read Coils), the address offset is increased by 5 bits. However, if the command element uses function code 03 (word level code - read holding registers), the address offset is increased by 5 words.	Application specific
Mod_Commands[x].Numberofpoints	INT	Designates the number of points that the controller should read or write. Depending on the command element's function code, this value may be in bits or words. For example, if this tag = 10 in a command element that uses function code 03 (word level code - Read Holding Registers), the controller reads 10 words.	Application specific
Mod_Commands[x].Spare1	INT	Not used	
Mod_Commands[x].Spare2	INT	Not used	
Mod_CommdMax	DINT	Maximum number of commands you want to use in your program.	40 = default
Mod_Data_Array_Max	DINT	Modbus_Data registers size	Modbus_Data register size
Mod_Data_Coils0	INT[250]	Modbus register - Output bit that the Modbus master sends to the slave.	
Mod_Data_Contacts1	INT[250]	Modbus register - Input bit that the Modbus master receives from the slave.	
Mod_Data_InpReg3	INT[250]	Modbus register - Input registers that the Modbus master receives from the slave.	
Mod_Data_HoldReg4	INT[250]	Modbus register - Output registers that the Modbus master sends to the slave.	

Table 2

Tag name:	Tag type:	Description:	Valid values:
Mod_Echo_MaxTime	DINT	Maximum time (milliseconds) for the master to wait for a echo from a slave before determining the slave node is missing.	We recommend you set this time greater than the time it takes to send the longest MSG in your project.
Mod_Scan_Preset	DINT	Number of scans the master should perform.	0 - 40 We recommend at least 2.

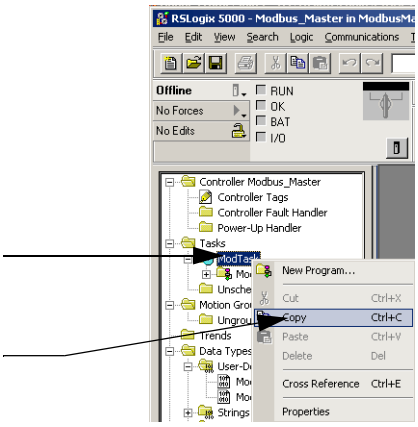
Copy ModTask From the ModbusMaster.ACD File to Your RSLogix 5000 Project

After you have copied the new UDTs and controller tags into your RSLogix 5000 project, you must copy the ModTask task, and its corresponding Modbus_Interface program, to your RSLogix 5000 project.

IMPORTANT The ModTask is configured as a Periodic task with a Period = 9ms. We determined this period to allow for the best performance of the solution. If you would like this solution to use less controller resources, increase the period.

Do these steps to copy the task.

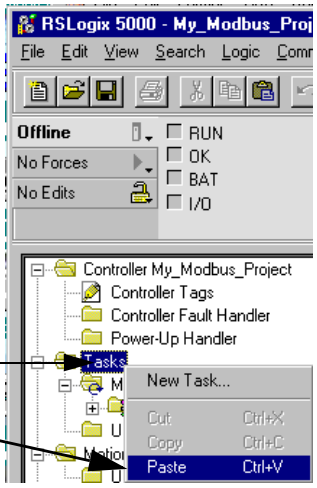
1. Copy the ModTask task from the ModbusMaster.ACD file.



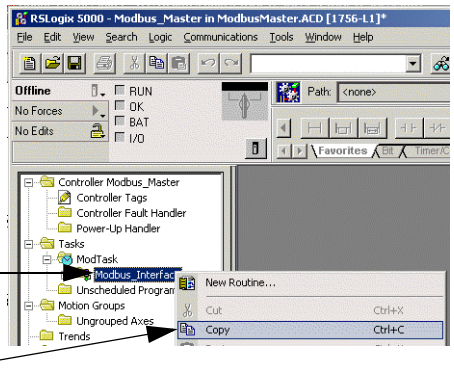
A. Right-click on ModTask.

B. Click Copy.

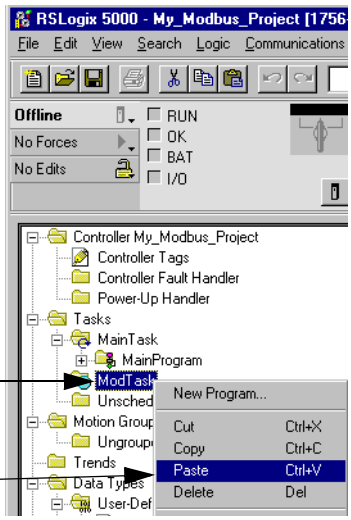
2. Paste the ModTask task into your RSLogix 5000 project.



3. Copy the Modbus_Interface program from the ModbusMaster.ACD file.



4. Paste the Modbus_Interface program into the ModTask task in your RSLogix 5000 project.



Configure the Controller's Communication Port

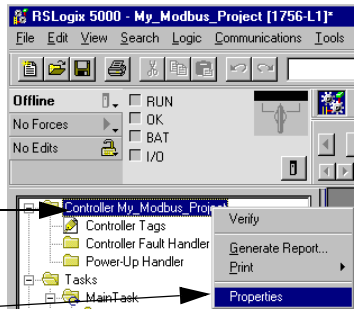
After you copy the new UDTs, controller tags, ModTask task and Modbus_Interface program into your RSLogix 5000 project, configure the communication port for the Logix5000 controller in your RSLogix 5000 project.

Do these steps to configure your controller's communication port.

1. Access the controller's properties.

A. Right-click on your controller.

B. Click Properties.



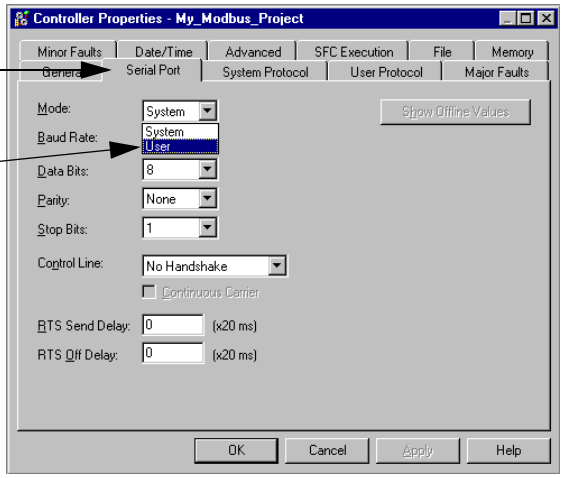
2. Configure the Serial Port tab.

A. Click on the Serial Port tab.

B. Use the pulldown menu to change the mode to User.

C. If you need to change your Read/Write Buffer Size, apply your changes and move to step 3 on page 17.

D. If you do not need to change your Read/Write Buffer Size, click OK and move to page 17.



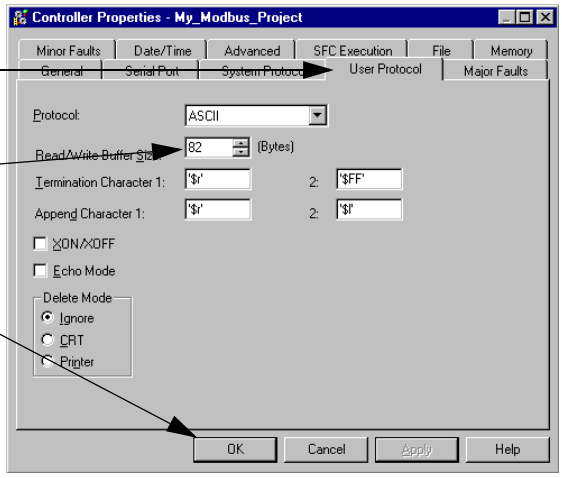
3. If necessary, configure the User Protocol tab.

   Your controller's default Read/Write Buffer Size = 82 bytes. If you expect to read and write more data than that in your RSLogix 5000 project, increase this buffer size to accommodate your project.

A. Click on the User Protocol tab.

B. Change the Read/Write Buffer Size to accommodate your project.

C. Click OK.



Configure the New Controller Tags

Once the new controller tags are copied into your RSLogix 5000 project, you need to configure the tag described in Table 3:

Table 3

This tag:	is configured once per:
Mod_Number_of_Instr	project
Mod_Commands[x].Enable	command
Mod_Commands[x].ScanNumber	
Mod_Commands[x].AddressOffsetinMaster	
Mod_Commands[x].SlaveAddress	
Mod_Commands[x].FunctionCode	
Mod_Commands[x].StartingAddress	
Mod_Commands[x].Numberofpoints	

Before you configure the tags listed above, make sure you understand how a Modbus master uses the tag configuration to read data from, and write data to, its data table.

Tag Arrays in Modbus Master Data Table

There are 4 tag arrays in Modbus master data tables, one for each of the following:

- Coil data (Mod_Data_Coils0 tags) - data is read from or written into this array in bit format
- Contact data (Mod_Data_Contacts1 tags) - data is written to this array in bit format
- Input Register data (Mod_Data_InpReg3 tags) - data is written to this array in word format
- Hold Register data (Mod_DataHoldReg4 tags) - data is read from or written into this array in bit format

Depending on how you configure the Mod_Commands tags in this solution, the Modbus master:

- uses 1 of its 4 arrays at a time (determined by the function code)
- reads data from, or writes data to, a specific location (determined by the address offset and the starting address)
- reads or writes a specific amount of data (determined by the number of points)
- reads the data (to be written to a slave node) or writes data (that was read from a slave node) to a specific location

For more information, see the example on page 19.

EXAMPLE

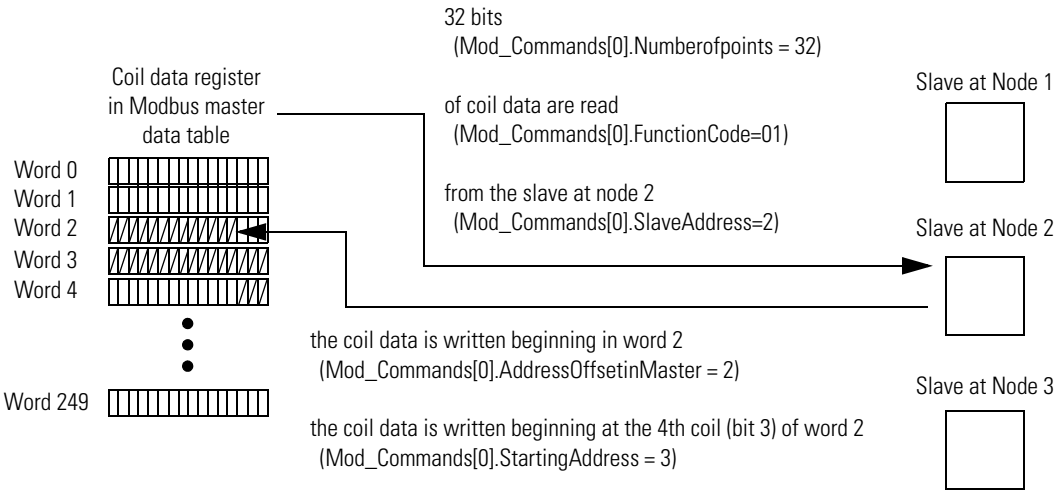
If you want the following to occur when a command is executed:

- the command is always enabled (i.e., occurs with every scan)
- the master writes the coil data beginning at word 2 in the Coil data register
- coil data is read from the Modbus slave at node 2
- the master begins writing at bit 3 of word 2 in the Coil data register
- 32 bits of coil data are read from the slave

Configure your command with these tag values:

- Mod_Commands[0].Enable = 1
- Mod_Commands[0].AddressOffsetinMaster = 2
- Mod_Commands[0].SlaveAddress = 2
- Mod_Commands[0].FunctionCode = 1
- Mod_Commands[0].StartingAddress = 4
- Mod_Commands[0].Numberofpoints = 32

This example command is shown in the graphic below.



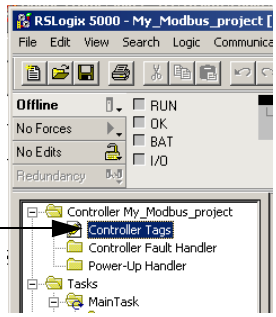
In this example, the Mod_Commands[0].StartingAddress and Mod_Commands[0].Numberofpoints tags represent bit values because the Modbus function code was 01 (Read Coils), a bit level code.

If a word level Modbus function code (e.g. 03 - Read Holding Registers) was used, the Mod_Commands[0].StartingAddress and Mod_Commands[0].Numberofpoints tags would represent word values. In other words, instead of reading 32 bits of data from the slave, the master would read 32 words of data from the slave and write 32 words into its data table, beginning with word 5.

Do these steps to change the tag values:

1. Access the controller tags.

Double-click on Controller Tags in your project.



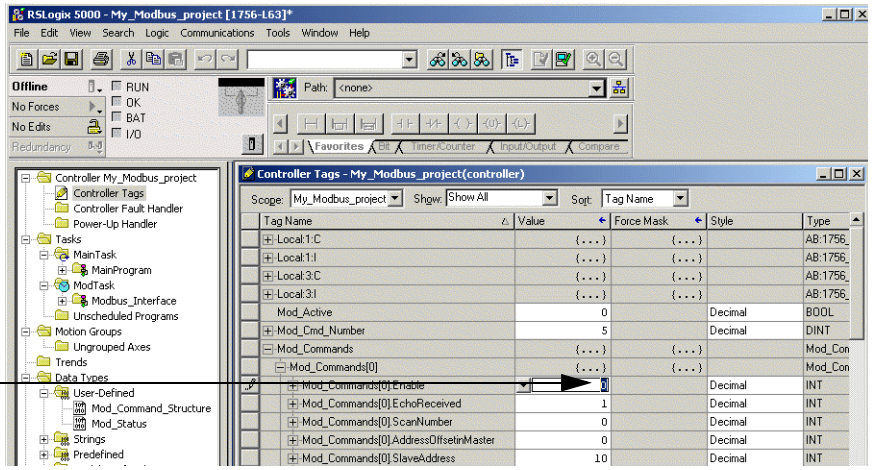
2. Select the Monitor Tags tab.

3. Change the tags as necessary. For example, the screen below shows the Commands[0].Enable tag.

A. Highlight the tag value.

B. Type a new value.

C. Press the *Enter* key.



The sample screen below uses tags that were configured for the example on page 19.

The command is always enabled.
Mod_Commands[0].Enable = 1

The command was successfully executed. (The controller automatically fills in this number when the command is executed successfully.)
Mod_Commands[0].EchoReceived = 1

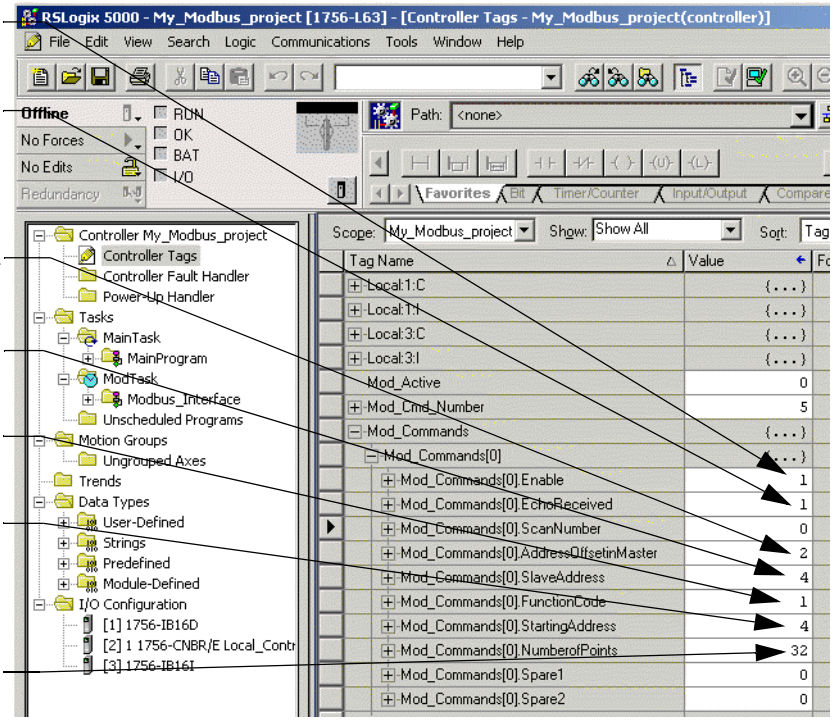
The address offset in the master data table is 2 words.
Mod_Commands[0].AddressOffsetinMaster = 2

The slave node address is 4.
Mod_Commands[0].SlaveAddress = 4

Coil data is read from the slave.
Mod_Commands[0].FunctionCode = 1

4 bits (bits because the function code is a bit level code) are added to the address offset in the master data table.
Mod_Commands[0].StartingAddress = 4

32 bits of data are read from the slave and written to the master data table.
Mod_Commands[0].Enable = 32



Enable the New Program

Finally, you must enable the new program before it will run. To enable the new program, you must change the Mod_Active controller tag to 1.

By default, this tag = 0 so you can load the program (using the steps described in this document) to your RSLogix5000 project without the program starting.

Verify that the Modbus_Interface Program is Working

Once you copy everything from the Modbus Master.ACD file into your RSLogix 5000 project, monitor your controller's Main Task to make sure the Modbus_Interface program is returning data correctly.

To see how to monitor your project to make sure it is operating properly, see page 32.

Using Logix5000 Controllers as Slaves on Modbus

The ModbusSlave.ACD file helps you use a Logix5000 controller as a slave on Modbus. The ACD file contains:

- 10 controller tags
- 1 periodic task

You must do the following tasks to use an Logix5000 controller as a slave on Modbus:

1. Copy Controller Tags From the ModbusSlave.ACD File to Your RSLogix 5000 Project
2. Copy ModTask From the ModbusSlave.ACD File to Your RSLogix 5000 Project
3. Configure New Controller Tag Sizes
4. Enable the New Program
5. Verify that Your RSLogix 5000 Project is Working

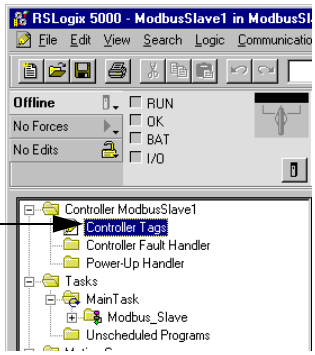
Copy Controller Tags From the ModbusSlave.ACD File to Your RSLogix 5000 Project

Do the following steps to copy the controller tags to your Modbus program.

1. Start RSLogix 5000.
2. Open the ModbusSlave.ACD file. The ACD file is available at either of the following locations:
   - Allen-Bradley Channel Extranet - Download the file from the United States Channel Extranet.
   - Vendor Sample Project PDF - Access the PDF via the Help menu in RSLogix 5000.
3. Open your RSLogix 5000 project in a second instance of RSLogix 5000.

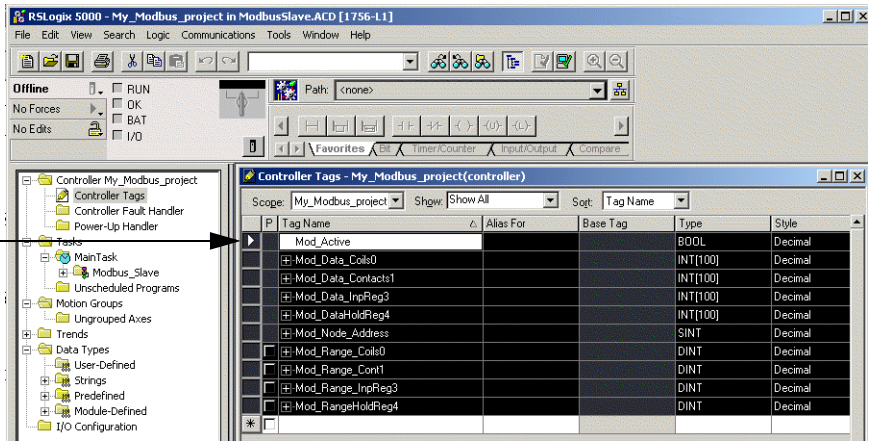
4. Copy the controller tags in the ModbusSlave.ACD file.

A. Double-click on Controller Tags.



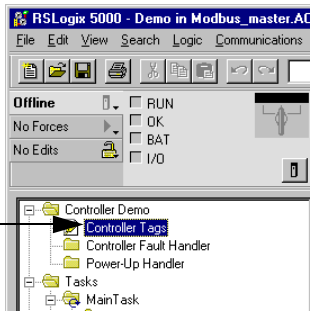
B. Select the 10 controller tags.

C. Copy the tags using the CTRL + C keystroke.



5. Paste the controller tags into your RSLogix 5000 project.

A. Double-click on Controller Tags.

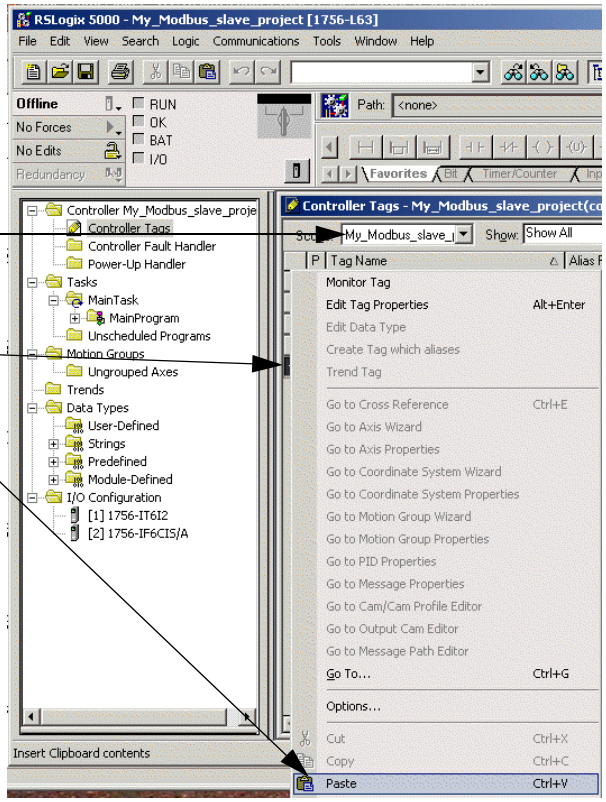


B. Make sure you are in the Tag Editor.

C. Make sure you are pasting the tags into the controller scope.

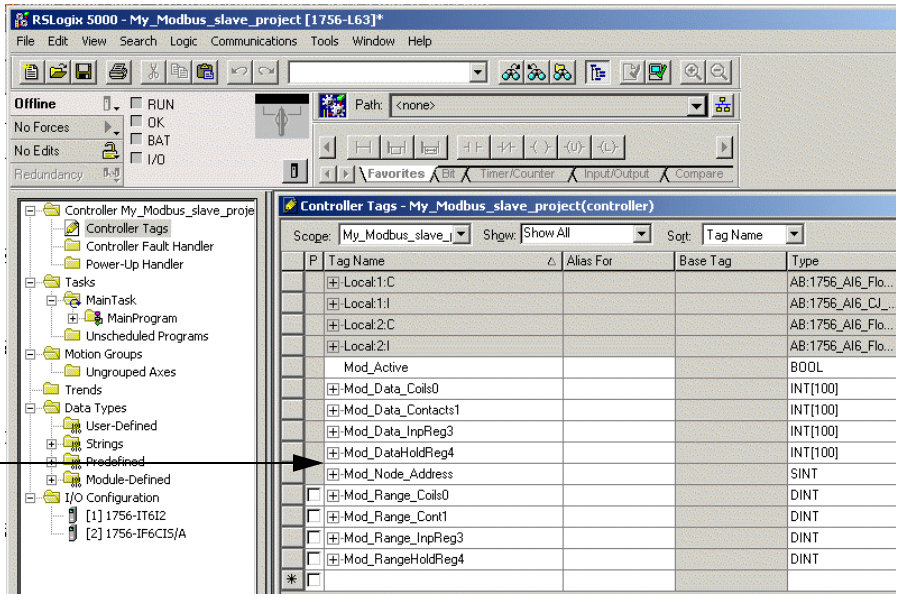
D. Right-click on the bottom tag line.

E. Click Paste.



RSLogix 5000 pastes the new tags into your RSLogix 5000 project and keeps them together because all 10 tags use a 'Mod_' prefix, as shown below.

New controller tags in the Modbus master file.



New Modbus Slave Controller Tags

Table 4 describes the controller tags you copied from the ModbusSlave.ACD file to your RSLogix 5000 project.

Table 4

Tag name:	Tag type:	Description:	Valid values:
Mod_Active	BOOL	This tag determines whether your project runs the new program in a Logix5000 controller. This tag = 0 (i.e., the solution is disabled) by default, and you must change the tag to 1 to run the solution.	0 = Do not run the program (default) 1 = Run the program
Mod_Data_Coils0	INT	Modbus Data Register - Output bits that the Modbus master writes data to.	
Mod_Data_Contacts1	INT	Modbus Data Register - Input bits that the Modbus master reads data from.	
Mod_Data_InpReg3	INT	Modbus Data Register - Input register that Modbus master reads data from.	
Mod_Data_HoldReg4	INT	Modbus Data Register - Output register that Modbus master writes data to.	
Mod_Node_Address	SINT	Modbus Slave (Node) Address - Represents the Modbus slave's node address.	0 - 255
Mod_Range_Coils0	DINT	Modbus RegisterSize - Tag that monitors the array size of the Mod_Data_Coils0 tag. The default size for this tag is 100 words, meaning the Mod_Data_Coils0 tag is limited to 100 words of data. You only need to change the size of this tag if you change the size of the tag it monitors. For example, if you change the Mod_Data_Coils0 tag size to 50 words, you should change this tag size to 50 or less.	

Table 4

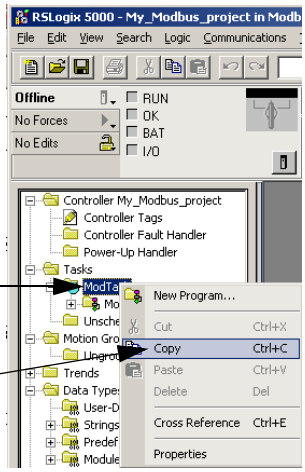
Tag name:	Tag type:	Description:	Valid values:
Mod_Range_Cont1	DINT	Modbus RegisterSize - Tag that monitors the size of data used in the Mod_Data_Contacts1 tag. The default size for this tag is 100 words, meaning the Mod_Data_Contacts1 tag is limited to 100 words of data. This tag's size is purposely 2 words smaller than the tag it monitors to prevent array overflow. You only need to change the size of this tag if you change the size of the tag it monitors.	
Mod_Range_HoldReg4	DINT	Modbus RegisterSize - Tag that monitors the size of data used in the Mod_Data_HoldReg4 tag. The default size for this tag is 100 words, meaning the Mod_Data_HoldReg4 tag is limited to 100 words of data. You only need to change the size of this tag if you change the size of the tag it monitors.	
Mod_Range_InpReg3	DINT	Modbus RegisterSize - Tag that monitors the size of data used in the Mod_Data_InpReg3 tag. The default size for this tag is 100 words, meaning the Mod_Data_InpReg3 tag is limited to 100 words of data. You only need to change the size of this tag if you change the size of the tag it monitors.	

Copy ModTask From the ModbusSlave.ACD File to Your RSLogix 5000 Project

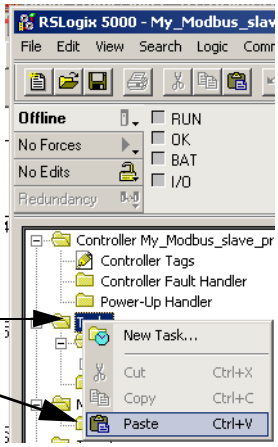
After you copy the controller tags from the ModbusSlave.ACD file into your RSLogix 5000 project, you must copy the ModTask task, and its corresponding program, to your project as well.

Do the following steps to copy the ModTask task to your RSLogix 5000 project.

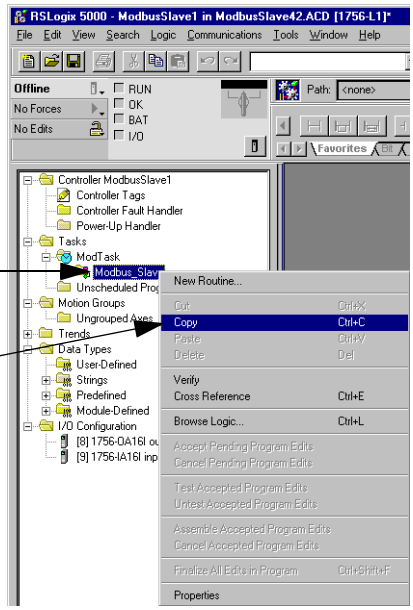
1. Copy the ModTask task from the ModbusSlave.ACD file.



2. Paste the ModTask task into your RSLogix 5000 project.



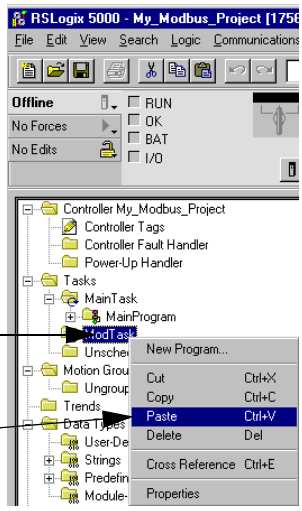
3. Copy the Modbus_Slave program from the ModbusSlave.ACD file.



A. Right-click on Modbus_Interface.

B. Click Copy.

4. Paste the Modbus_Slave program into your RSLogix 5000 project.



A. Right-click on ModTask.

B. Click Paste.

Configure New Controller Tag Sizes

The Modbus Data Register tags (i.e., Mod_Data_Coils0, Mod_Data_Contacts1, Mod_Data_HoldReg4 and Mod_Data_InpReg3) use a default size = 100 words. If you need to increase the tag sizes, you can.

However, if you increase the size of the Modbus Data Register tags, you must also change the size of the corresponding Modbus Register Range tags that monitor them.

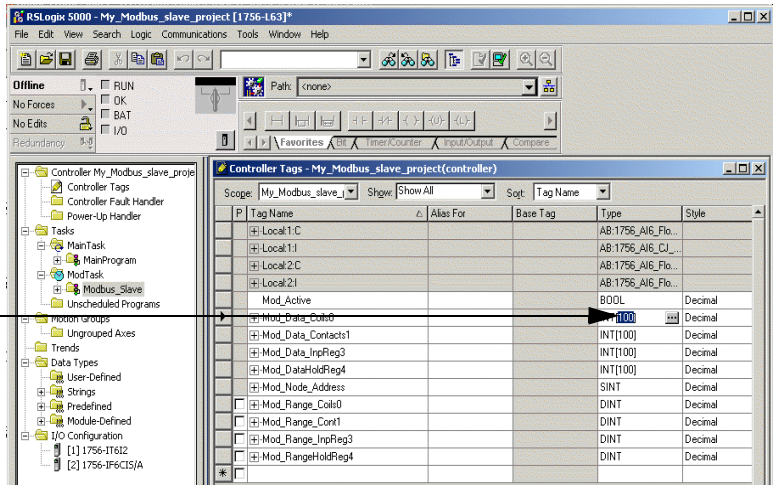
Table 5 explains which Modbus Data Register Range tags must be changed to accommodate new sizes in the Modbus Data Register tags.

Table 5

If you change the size of this tag:	You must change the size of this tag:
Mod_Data_Coils0	Mod_Range_Coils0
Mod_Data_Contacts1	Mod_Range_Contacts1
Mod_Data_InpReg3	Mod_Range_InpReg3
Mod_Data_HoldReg4	Mod_Range_HoldReg4

To change the **Modbus Data Register** tags' size, do the following steps:

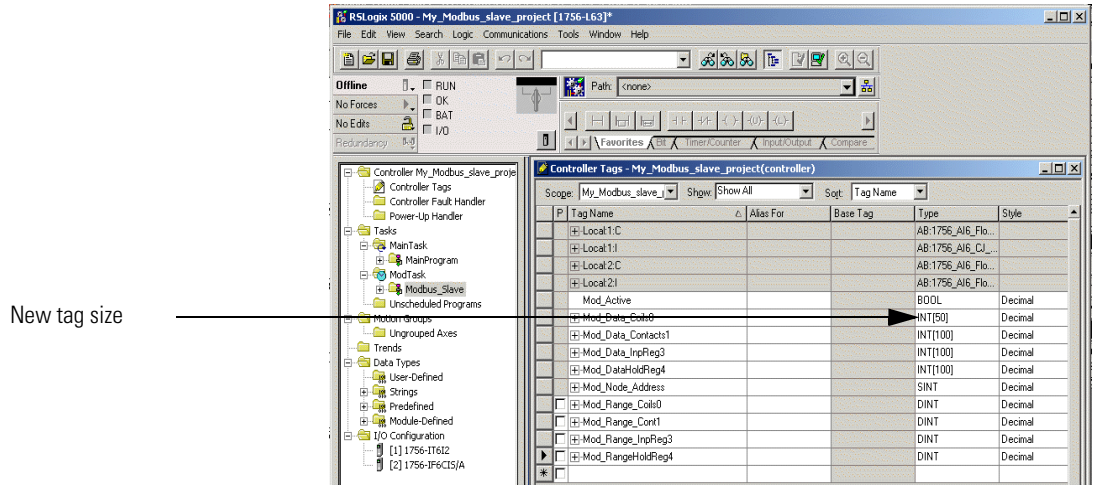
1. On the Edit Tags tab, highlight the number in the tag type field.



Tag size highlighted

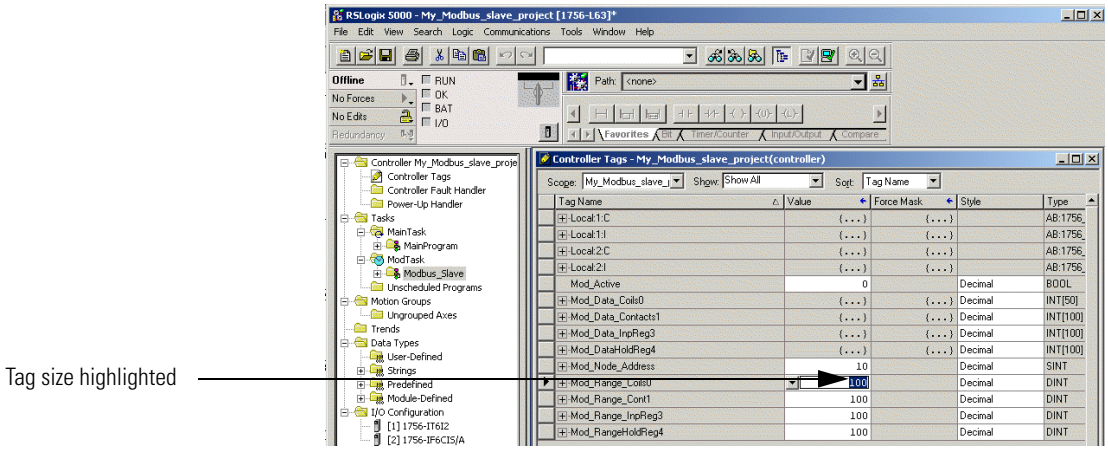
2. Type the new size.

3. Press the Enter key to see the new size reflected in the tag.



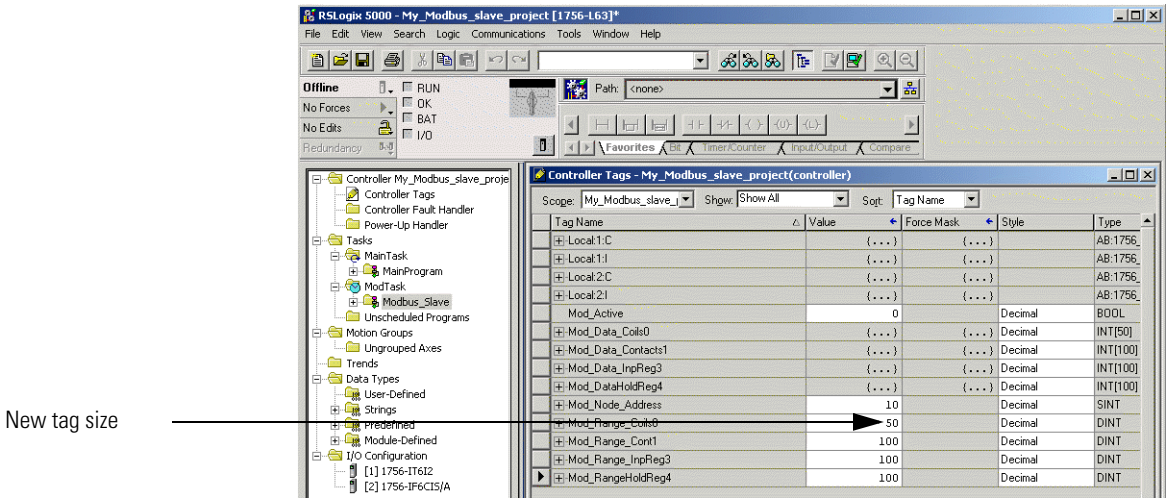
To change the **Modbus Register Range** tags' size, do the following steps:

1. On the Monitor Tags tab, highlight the tag value.



2. Type the new size.

3. Press the Enter key to see the new size reflected in the tag.



Enable the New Program

Finally, you must enable the new program before it will run. To enable the new program, you must change the Mod_Active controller tag to 1.

By default, this tag = 0 so you can load the program (using the steps described in this document) to your RSLogix5000 project without the program starting.

Verify that Your RSLogix 5000 Project is Working

Once you copy everything from the ModbusSlave.ACD file into your RSLogix 5000 project, monitor the controller's Main Task to make sure the Modbus_Slave program is returning data correctly.

To see how to monitor your project to make sure it is operating properly, see page 32.

Monitoring and Troubleshooting Your Modbus Project

Whether you use a Logix5000 controller as a master or slave on Modbus, you may want to monitor your program once you have implemented the solutions offered in this document. In other words, you want to make sure your application is operating correctly.

Both the ModbusMaster and ModbusSlave ACD files provide diagnostic features to verify that:

- a command executed
- a command transmitted
- slaves answer correctly after receiving a master's command
- slaves rejecting a command do so with an error code

Both ACD files used in this application solution provide diagnostics, although the ModbusMaster.ACD file provides significantly more diagnostics than the ModbusSlave.ACD file.

Verify Successful Command Execution

You can verify that a command executed successfully, when your Modbus master sends a command element to a Modbus slave simply by monitoring the Mod_Commands[x].EchoReceived tag. Do these steps to verify successful execution:

1. Before sending a command to a Modbus slave, set the Mod_Commands[x].EchoReceived tag = 0 via the Tag Monitor in RSLogix 5000.

2. Send the command to the Modbus slave.

3. Monitor the value in the Mod_Commands[x].EchoReceived tag.
   - If the tag value changes to match the value used in the command element's Mod_Commands[x].Enable tag (i.e., 1, 2 or 3), the command executed properly and the slave returned a valid response. Note that the execution may continuously repeat, if repeated scans are occurring.
   - If the tag value of the command element's Mod_Commands[x].Enable tag remains = 0, the command was not executed successfully.

Use Program Tags

With both the ModbusMaster and ModbusSlave portions of this solution, you copied periodic tasks to your RSLogix 5000 project. Each task contains Program Tags that allow you to monitor your project's performance. The Program Tags are located in either:

- the Modbus Interface program for the Modbus Master file.

  or

- the Modbus Slave program for the Modbus Slave file.

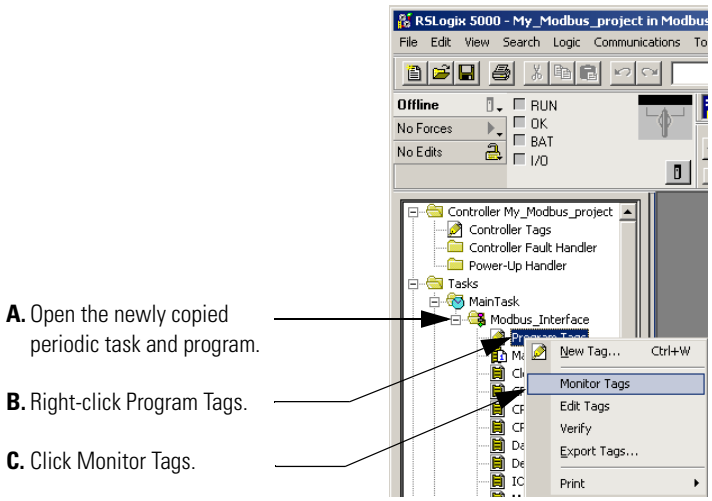
To use the Program Tags, do these steps:

TIP

Before monitoring the data for a particular command element, you may want to set the Mod_Commands[x].Enable tag for that command element = 3. With this value, the command only executes one time instead of repeatedly and you can monitor static.

However, if you want to monitor dynamic data to verify that any changes are occurring correctly, use 1 or 2 in the Mod_Commands[x].Enable tag.

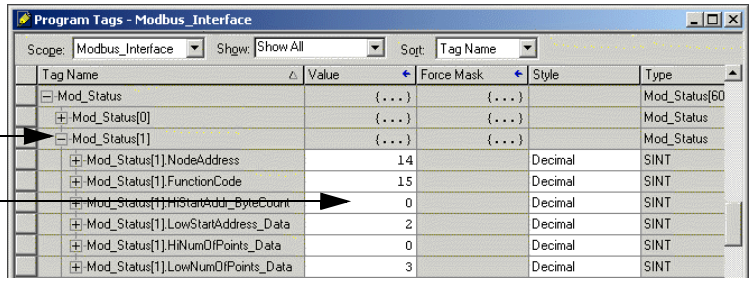
1. Access the Program Tags. This example shows how to access the tags in the ModbusMaster file.



2. Scroll to the tags you need to monitor. The example screen below shows Mod_Status tags used in the ModbusMaster file to monitor the operation of command element 1

A. Expand the tags as needed.

B. Monitor these values.



Some of the Program Tags correlate directly to Mod_Commands elements. Table 6 on describes some of the Program Tags you can use to monitor and troubleshoot your program.

Table 6

This Program Tag:	available in this ACD file:	has this description:
MsgCountRec	ModbusMaster.ACD file ModbusSlave.ACD file	Counts the number of messages received every 10 seconds. • In the ModbusMaster.ACD file, this tag should closely match the MsgCountTrans tag. If there is a considerable difference, then some commands may have been sent to a slave that does not exist or cannot respond. • In the ModbusSlave.ACD file, this tag may not closely match the MsgCountTrans tag because the slave receives all messages but only transmit when addressed.
MsgCountTrans		Counts the number of messages transmitted every 10 seconds. • In the ModbusMaster.ACD file, this tag should closely match the MsgCountRec tag. If there is a considerable difference, then some commands may have been sent to a slave that does not exist or cannot respond. • In the ModbusSlave.ACD file, this tag may not closely match the MsgCountRec tag; the slave receives all messages but only transmit when addressed.
MsgTimeSpent		Measures the time used for a transaction from the moment it is sent by the master until an answer is received from the slave. If the slave does not answer, this tag will match the MsgTimeMax program tag.
Mod_Status[x].NodeAddress	ModbusMaster.ACD file only	This tag corresponds to the Mod_Commands[x].SlaveAddress tag in the command sent from the master to the slave. Shows the node address of the slave to which the master sent the command.

Table 6

This Program Tag:	available in this ACD file:	has this description:
Mod_Status[x].FunctionCode	ModbusMaster.ACD file only	This tag corresponds to the Mod_Commands[x].FunctionCode tag in the command sent from the master to the slave. Shows the function code of the command element sent to the slave if the command was executed successfully. In this case, the tag shows one of the function codes supported in this solution (i.e., 1, 2, 3, 4, 5, 6, 15 or 16). If the command was not executed successfully, this tag shows one of the following: 0 = Command is disabled. In this case, 0 appears in all Mod_Status fields. -1 = Command is sent but the node does not answer. The node may not have answered because: • the slave does not exist • the slave exists but took too long to respond • the master sent a function code this solution doe not support • the master sent a bad CRC -2 = Command is not sent because the number of bits exceeds 320 or array limits in the Master were exceeded. -10 = Command is not sent because an invalid function code is used in that command -1xxx = Command is sent, but the slave cannot execute the command and returns the function code plus Hex80. This then appears as (Function Code -128) in decimal notation. 1xx = Command with Mod_Commands[x].Enable=3 (one time execution) was processed. In this case, xx represents the function code. This display may be old because it shows last time execution data.
Mod_Status[x].HiStartAddr_ByteCount		This tag corresponds to the Mod_Commands[x].Starting Address controller tag in the command. Data shown is dependent on whether the command used a read or write function code: • If the command uses a write function code, the slave merely answers the master with the same data the master is writing. In this case, this tag shows the high starting address of the data written to the slave. • If the command uses a read function code, the slave returns the requested data (i.e., the data the master is reading from the slave). In this case, this tag shows the number of bytes needed to return the data requested in the Mod_Commands[x].Numberof points controller tag in the command.

Table 6

This Program Tag:	available in this ACD file:	has this description:
Mod_Status[x].LowStartAddress_Data	ModbusMaster.ACD file only	This tag corresponds to the Mod_Commands[x].Starting Address controller tag in the command. Data shown is dependent on whether the command used a read or write function code: • If the command uses a write function code, the slave merely answers the master with the same data the master is writing. In this case, this tag shows the low starting address of the data written to the slave. • If the command uses a read function code, the slave returns the requested data (i.e., the data the master is reading from the slave). In this case, this tag shows the data at the low starting address.
Mod_Status[x].HiNumOfPoints_Data		This tag corresponds to the Mod_Commands[x].Numberof points controller tag in the command. Data shown is dependent on whether the command used a read or write function code: • If the command uses a write function code, the slave merely answers the master with the same data the master is writing. In this case, this tag shows the high point number of the data written to the slave. • If the command uses a read function code, the slave returns the requested data (i.e., the data the master is reading from the slave). In this case, this tag shows the 2nd byte of data requested.
Mod_Status[x].LowNumOfPoints_Data		This tag corresponds to the Mod_Commands[x].Numberof points controller tag in the command. Data shown is dependent on whether the command used a read or write function code: • If the command uses a write function code, the slave merely answers the master with the same data the master is writing. In this case, this tag shows the low point number of the data written to the slave. • If the command uses a read function code, the slave returns the requested data (i.e., the data the master is reading from the slave). In this case, this tag shows the 3rd byte of data requested.

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication CIG-AP129A-EN-P - February 2004