5 Configuration as a Modbus Slave

In This Chapter

❖ Overview 80
❖ Configuration File Settings 81
❖ Slave Configuration 86
❖ Floating-Point Data Handling (Modbus Slave)....................................... 88

5.1 Overview

When configuring the module as a slave, you will be providing a Modbus Memory Map to the person who is programming the Master side of the communications.

Note: If you are using the Sample Ladder Logic, the transfer of data is already done.

Information that is to be read by the Modbus Master device will be placed in the **MCMR.DATA.WRITEDATA** array as this will be pushed out to the module so that values from the ControlLogix processor can be read by the Modbus Master. Information that must be written to the ControlLogix processor from the Modbus Master device will be placed into the **MCMR.DATA.READDATA** array.

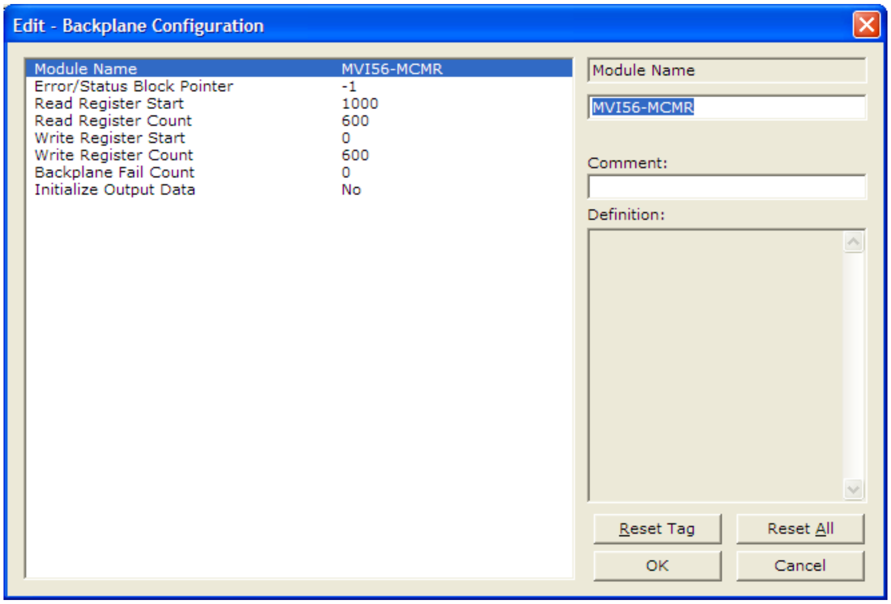
To configure module as a Modbus Slave, you must determine how much data you must transfer to and from the module, to the Modbus Master.

The sample ladder file is configured to transfer 600 16-bit registers in each direction. If more than that is required, please see Applications Requiring More Than 600 Registers of ReadData or WriteData.

5.2 Configuration File Settings

To configure Modbus slave mode, use the **BACKPLANE CONFIGURATION** settings.

This section specifies which of the MVI56-MCMR module's 5000 registers of memory to send from the ControlLogix processor to the MVI56-MCMR module (WriteData) and which registers to send from the MVI56-MCMR module to the ControlLogix processor (ReadData).



Edit - Backplane Configuration

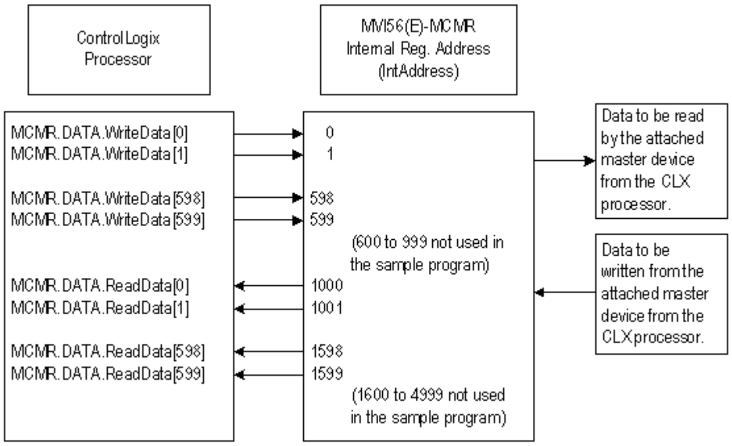
```
Module Name                MVI56-MCMR
Error/Status Block Pointer -1
Read Register Start        1000
Read Register Count        600
Write Register Start       0
Write Register Count       600
Backplane Fail Count       0
Initialize Output Data     No
```

Module Name: MVI56-MCMR

The **WRITE REGISTER START** determines the starting register location for **WRITEDATA [0 TO 599]** and the **WRITE REGISTER COUNT** determines how many of the 5000 registers to use for information to be written out to the module. The sample ladder file will configure 600 registers for Write Data, labeled **MCM.WRITEDATA[0 TO 599].**

Value	Description
Error/Status Block Pointer	This parameter places the STATUS data into the database of the module. This information can be read be the Modbus Master to know the status of the module.
Read Register Start	Determines where in the 5000 register module memory to begin obtaining data to present to the ControlLogix processor in the ReadData tags.
Read Register Count	Sets how many registers of data the MVI56-MCMR module will send to the ControlLogix processor. This value should also be a multiple of 40.
Write Register Start	Determines where in the 5000 register module memory to place the data obtained from the ControlLogix processor from the WriteData tags.
Write Register Count	Sets how many registers of data the MVI56-MCMR module will request from the ControlLogix processor. Because the module pages data in blocks of 40 words, this number must be evenly divisible by 40.
Backplane Fail Count	Sets the consecutive number of backplane failures that will cause the module to stop communications on the Modbus network.

With the sample configuration, the following is the layout of the tags and addressing.


ControlLogix Processor
MVI56(E)-MCMR Internal Reg. Address (IntAddress)
MCMR.DATA.WriteData[0]
MCMR.DATA.WriteData[1]
MCMR.DATA.WriteData[598]
MCMR.DATA.WriteData[599]
MCMR.DATA.ReadData[0]
MCMR.DATA.ReadData[1]
MCMR.DATA.ReadData[598]
MCMR.DATA.ReadData[599]
0
1
598
599
(600 to 999 not used in the sample program)
1000
1001
1598
1599
(1600 to 4999 not used in the sample program)
Data to be read by the attached master device from the CLX processor.
Data to be written from the attached master device from the CLX processor.

The sample configuration values configure the module database for **WRITEDATA[0 TO 599]** to be stored in the module memory at register 0 to 599, and **READDATA[0 TO 599]** to be stored in the module memory at registers 1000 to 1599 as shown above.

5.2.1 Modbus Memory Map

Based on the configuration described above, below is the default Modbus address for the module. Each register within the module can be accessed as a 0x bit address, 1x bit address, 3x register address, or 4x register address.

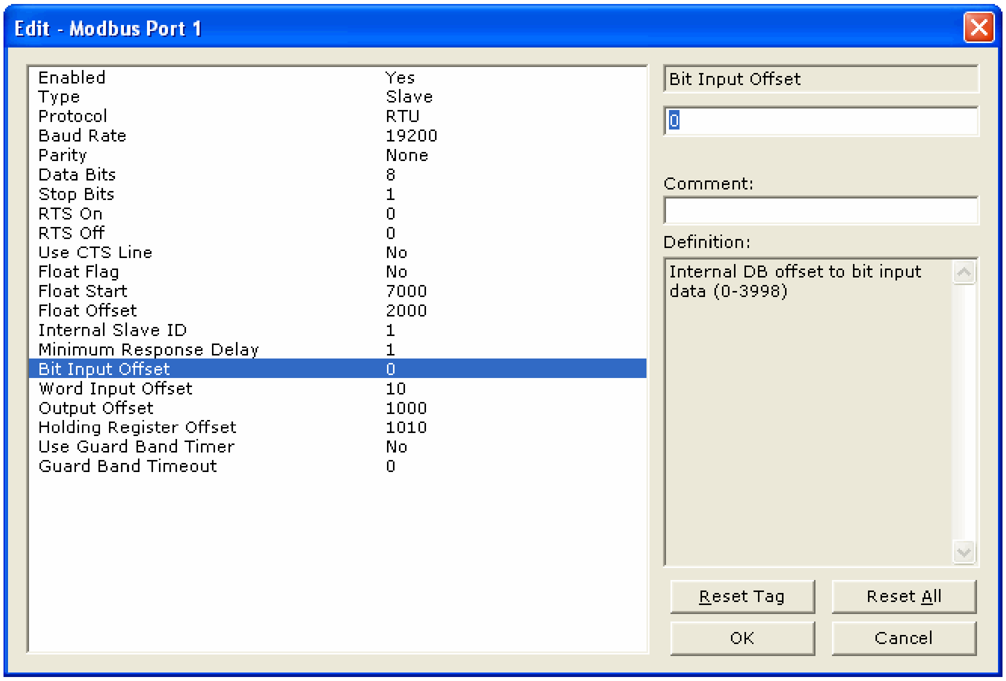
MVI Address	0x	1x	3x	4x	Tag Address
0	0001 to 0016	10001 to 10016	30001	40001	WriteData[0]
1	0017 to 0032	10017 to 10032	30002	40002	WriteData[1]
2	0033 to 0048	10033 to 10048	30003	40003	WriteData[2]
3	0049 to 0064	10049 to 10064	30004	40004	WriteData[3]
4	0065 to 0080	10065 to 10080	30005	40005	WriteData[4]
5	0081 to 0096	10081 to 10096	30006	40006	WriteData[5]
6	0097 to 0112	10097 to 10112	30007	40007	WriteData[6]
7	0113 to 0128	10113 to 10128	30008	40008	WriteData[7]
8	0129 to 0144	10129 to 10144	30009	40009	WriteData[8]
9	0145 to 0160	10145 to 10160	30010	40010	WriteData[9]
10	0161 to 0176	10161 to 10176	30011	40011	WriteData[10]
50	0801 to 0816	10801 to 10816	30051	40051	WriteData[50]
100	1601 to 1616	11601 to 11616	30101	40101	WriteData[100]
200	3201 to 3216	13201 to 13216	30201	40201	WriteData[200]
500	8001 to 8016	18001 to 18016	30501	40501	WriteData[500]
598	9569 to 9584	19569 to 19584	30599	40599	WriteData[598]
599	9585 to 9600	19585 to 19600	30600	40600	WriteData[599]
600 to 999	N/A	N/A	N/A	N/A	Reserved
1000			31001*	41001	ReadData[0]
1001			31002*	41002	ReadData[1]
1002			31003*	41003	ReadData[2]
1003			31004*	41004	ReadData[3]
1004			31005*	41005	ReadData[4]
1005			31006*	41006	ReadData[5]
1006			31007*	41007	ReadData[6]
1007			31008*	41008	ReadData[7]
1008			31009*	41009	ReadData[8]
1009			31010*	41010	ReadData[9]
1010			31011*	41011	ReadData[10]
1050			31051*	41051	ReadData[50]
1100			31101*	41101	ReadData[100]
1200			31201*	41201	ReadData[200]
1500			31501*	41501	ReadData[500]
1598			31599*	41599	ReadData[598]
1599			31600*	41600	ReadData[599]

The above addressing chart will work with many Modbus applications. Values listed in the ReadData array for 31001 to 31600 are shown with an * beside them.

Although these are valid addresses, they will not work in the application. The Master must issue a Write command to the addresses that correspond to the **READDATA** array. For Modbus addresses 3x, these are considered Input registers, and a Modbus Master does not have a function code for this type of data.

5.2.2 Customizing the Memory Map

In some cases, the above memory map will not work for the application. Sometimes a Master must read bits starting at address 0001, and must also read a register starting at 40001. With the memory map in this example (page 83), this is not possible, as **WRITEDATA[0]** is seen as both 0001 to 0016, and 40001. To accommodate this, you can customize the starting location within the module for each device using the parameters shown below.



Edit - Modbus Port 1

Enabled	Yes
Type	Slave
Protocol	RTU
Baud Rate	19200
Parity	None
Data Bits	8
Stop Bits	1
RTS On	0
RTS Off	0
Use CTS Line	No
Float Flag	No
Float Start	7000
Float Offset	2000
Internal Slave ID	1
Minimum Response Delay	1
Bit Input Offset	0
Word Input Offset	10
Output Offset	1000
Holding Register Offset	1010
Use Guard Band Timer	No
Guard Band Timeout	0

Bit Input Offset

0

Comment:

Definition:

Internal DB offset to bit input data (0-3998)

Parameter	Value	Description
Bit Input Offset	0	Defines the starting address within the module for 1x Modbus addressing. A value of 0 sets 10001 to 10016 as address 0 in the MVI56-MCMR module.
Word Input Offset	10	Defines the starting address within the module memory for 3x registers.
Output Offset	1000	Defines the starting address within the module for 0x coils.
Holding Register Offset	1010	Defines the starting address within the module for 4x addressing.

Based on the configuration described above for the ModDef section of the module and the values specified for the offset parameters, below is the Modbus addressing map for the module.

MVI Address	0x	1x	3x	4x	Tag Address
0		10001 to 10016			WriteData[0]
1		10017 to 10032			WriteData[1]
9		10145 to 10160			WriteData[9]
10		10161 to 10176	30001		WriteData[10]
11		10177 to 10192	30002		WriteData[11]
100		11601 to 11616	30091		WriteData[100]
200		13201 to 13216	30191		WriteData[200]
500		18001 to 18016	30491		WriteData[500]
598		19569 to 19584	30489		WriteData[598]
599		19585 to 19600	30490		WriteData[599]
600 to 999	N/A	N/A	N/A	N/A	Reserved
1000	0001 to 0016				ReadData[0]
1001	0017 to 0032				ReadData[1]
1009	0145 to 0160				ReadData[9]
1010	0161 to 0176			40001	ReadData[10]
1011	0177 to 0192			40002	ReadData[11]
1050	0801 to 0816			40041	ReadData[50]
1100	1601 to 1616			40091	ReadData[100]
1200	3201 to 3216			40191	ReadData[200]
1500	8001 to 8016			40491	ReadData[500]
1598	9569 to 9584			40589	ReadData[598]
1599	9585 to 9600			40590	ReadData[599]

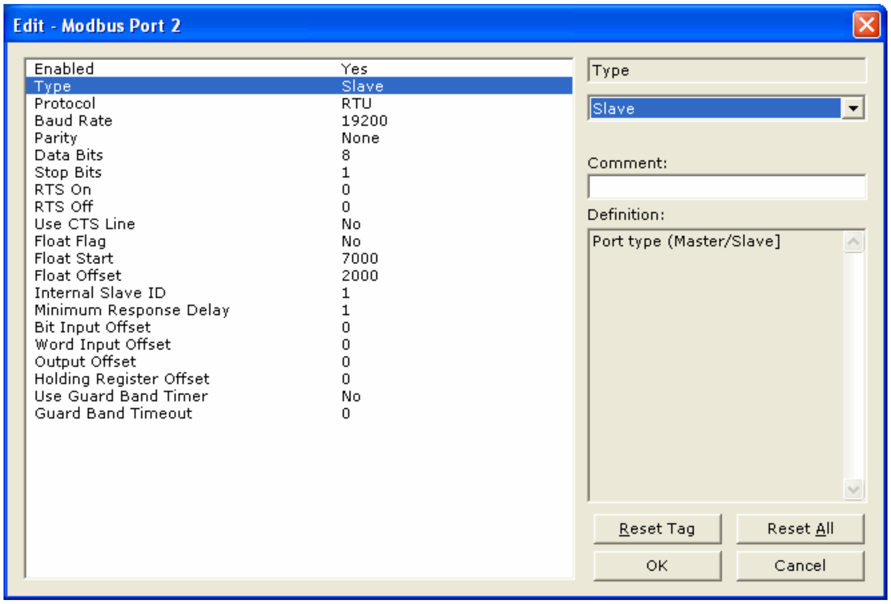
With the offset parameters listed above, the Modbus Master could read from coils 10001 to 10176 using the tags **MCMR.DATA.WRITEDATA[0] TO [9].** The Master could also read from address 30001 to 30490, and the data contained in those Modbus addresses would come from the tags **MCMR.DATA.WRITEDATA[10] TO [499]** within the ControlLogix program.

The Master could then write to coils addressing 0001 to 0160 and this data would reside within the ControlLogix program in tags **MCMR.DATA.READDATA[0] TO [9].** The Master could then write to registers using Modbus addresses 40001 to 40590, and this information would reside in addresses **MCMR.DATA.READDATA[10] TO [599].**

Note: The offset parameter only set the starting location for the data. As shown above, if the Master issues a Write command to address 40001, the data will go into the ControlLogix processor at address **MCMR.DATA.READDATA[10].**

Likewise, a Write To bit address 0161 will also change to address **MCMR.DATA.READDATA[10].0** within the program. Be careful not to overlap your data. You may want leave additional registers/bits unused to allow for future expansion in the program.

5.3 Slave Configuration



Edit - Modbus Port 2

Enabled	Yes
Type	Slave
Protocol	RTU
Baud Rate	19200
Parity	None
Data Bits	8
Stop Bits	1
RTS On	0
RTS Off	0
Use CTS Line	No
Float Flag	No
Float Start	7000
Float Offset	2000
Internal Slave ID	1
Minimum Response Delay	1
Bit Input Offset	0
Word Input Offset	0
Output Offset	0
Holding Register Offset	0
Use Guard Band Timer	No
Guard Band Timeout	0

Type: Slave

Comment:

Definition: Port type (Master/Slave]

Value	Description
Enabled	**1** = enable port, **0** = disable port
Type	**1** = Modbus Slave Port
Protocol	**0** = Modbus RTU mode, **1** = Modbus ASCII mode
Baud Rate	Sets the baud rate for the port. Valid values for this field are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 384 or 3840 (for 38,400 baud), 576 or 5760 (for 57,600 baud) and 115,1152, or 11520 (for 115,200 baud)
Parity	**0** = None, **1** = Odd, **2** = Even
Data Bits	**8** = Modbus RTU mode, **8** or **7** = Modbus ASCII mode
Stop Bits	Valid values are **1** or **2**
RTS On	**0** to **65535** milliseconds to delay after RTS line is asserted on the port before data message transmission begins. This delay can be used to allow for radio keying or modem dialing before data transmission begins.
RTS Off	**0** to **65535** milliseconds to delay after data message is complete before RTS line is dropped on the port.
Use CTS Line	**NO** or **YES** This parameter is used to enable or disable hardware handshaking. The default setting is **NO** hardware handshaking, CTS Line not used. Set to **NO** if the connected devices do not need hardware handshaking. Set to **YES** if the device(s) connected to the port require hardware handshaking (most modern devices do not). If you set this parameter to **YES**, be sure to pay attention to the pinout and wiring requirements to ensure that the hardware handshaking signal lines are properly connected; otherwise communication will fail.
Float Flag	As a Slave, emulates Enron/Daniel style floats. See Floating Point Data Handling for more information (page 88).

Value	Description
Float Start	Register offset in message for floating data point. See Floating Point Data Handling for more information (page 88).
Float offset	Internal address for floats
Internal Slave ID	Valid values are **1** to **247**
Minimum Response Delay	**0** to **65535** milliseconds to delay before response
Bit Input Offset	Defines the starting address within the module for 1x Modbus addressing. A value of 0 sets 10001 to 10016 as address 0 in the MVI56-MCMR module.
Word Input Offset	Defines the starting address within the module memory for 3x registers.
Output Offset	Defines the starting address within the module for 0x coils.
Holding Register Offset	Defines the starting address within the module for 4x addressing.
Use Guard Band Timer	**YES** or **NO** Packet gap timeout for messages
Guard Band Timeout	**0** to **65535** A value of 0 uses the default baud rate, or you can set a timeout value in milliseconds.