Application Note AN112

LabVIEW™ Interface Using Modbus Protocol
MR330 Absolute Fiber Optic Position Sensor

Objective

This application note describes how to interface the Micronor MR330 Absolute Fiber Optic Position Sensor to the National Instruments LabVIEW™ environment. The example described herein utilizes Modbus RTU protocol interface which is standard on the MR330 Controller Module. National Instruments LabVIEW™ supports the Modbus interface with a free downloadable VI (Virtual Instrument). The communications described herein will work equally well for Serial Interface or the USB interface.

Equipment

- PC type computer with either USB or serial Interface capability
- LabVIEW™ 2011 Basic or higher.
- VISA Driver (National Instruments) installed on your computer.
- MR232-1 Serial Interface to RS242 cable. Or a USB Cable.
- Virtual Instrument Demo Software supplied with this application note.

  ***Note**: For the purposes of this Application note it is assumed that you are familiar with the LabVIEW™ environment.*

Installation

Install the demo software to a suitable LabVIEW™ working directory on your computer. The Modbus driver VI (Virtual Instrument) must also be installed. Download from National Instruments web site and follow the installation direction for this component to be installed. For this example we installed version 8.6 into the LabVIEW™ 2011 library:

http://sine.ni.com/devzone/cda/epd/p/id/4756.

Follow the instructions provided in the download file: readme.html.

There is also a useful Modbus primer provided by National Instruments.

If the VISA Driver is not yet installed on your computer then download the free VISA driver software from the National Instruments website: http://www.ni.com/visa/. After downloading proceed to install the VISA driver. There is no specific configuration required for the VISA driver.

When using the serial port interface on the PC make sure you use an appropriate RS232 to RS422/RS485 converter cable. Micronor offers part number RS232-1 Communications Cable. When using the USB cable then the PC will register as a new Com Port and it will show up on the VISA driver as a new comport.

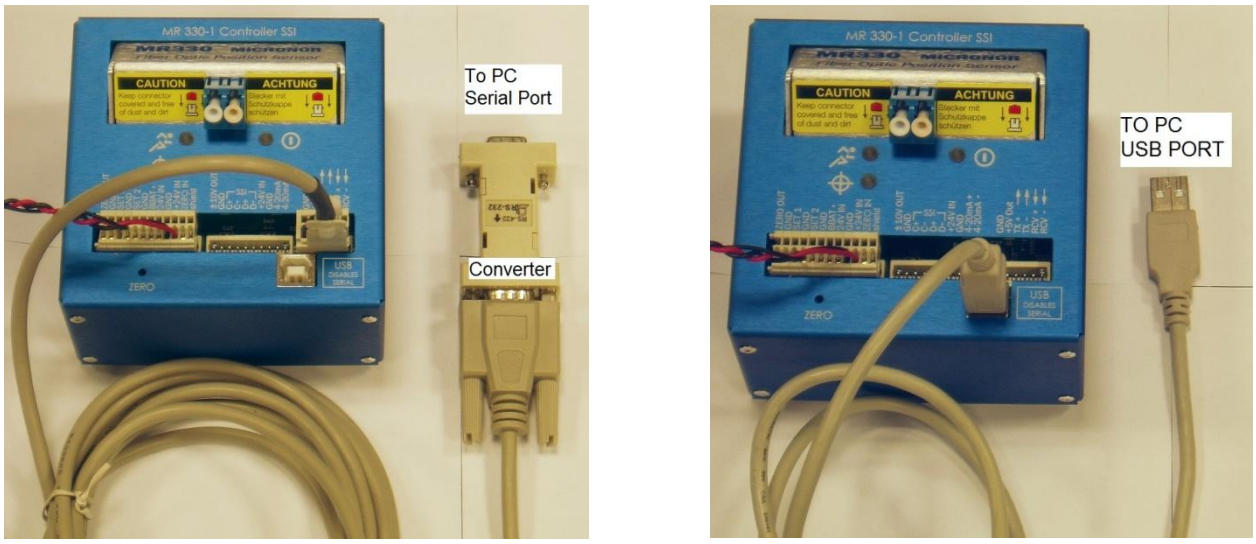

Figure 1: Connecting Controller to PC

Running the Example Software

Start LabVIEW™ and select the MICRONOR example Vi named: "ModBus_Master_Example_MR330.vi"

Before running the program, select the appropriate COMx interface. When using the USB cable, a new COM port (typically COM4 or higher) will be available.

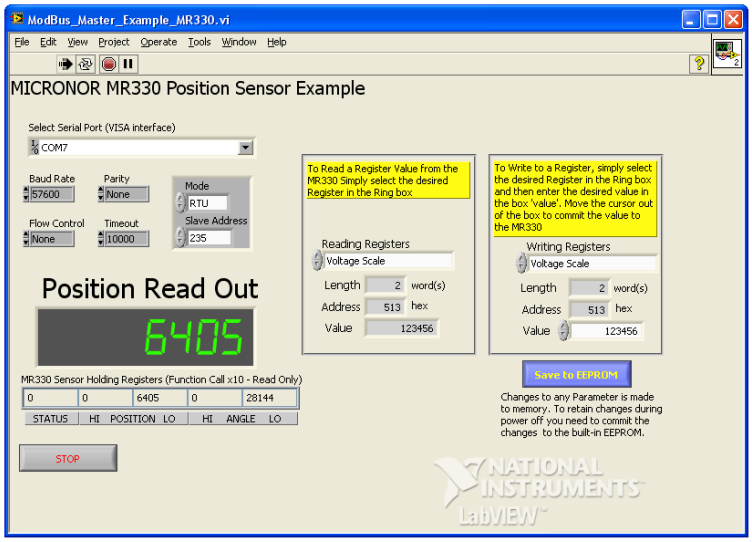

Figure 2: Sample Screen Shot of Software

The MR330 controller is factory preset at 57000 baud and 8 bit with no parity. We recommend you use these settings. To change the baud-rate settings see description later in this application note.

The MR330 also has a Modbus general call address of 235. All units will always respond on that address. This is useful when the actual address setting of the unit is not known.

Run the program and the unit should respond with the position readout being updated every 300ms.

The five numeric windows below the Position read out are the first five Modbus registers of the MR330 and they are continually being updated. The position output covers a full 32 bits and therefore two registers need to be read to get the full multi-turn position. Register 0 indicates the status of the unit, for instance, if you disconnect the fiber cable from the unit the STATUS readout will show 770 indicating a loss of connection to the sensor.

By manipulating the "Reading Registers" Ring Control any register of the MR330 may be inspected. Simply select an appropriate register and the read-out will appear in the field labeled Value. A detailed description for each register is in the appendix of this document.

To write a new value to a register first select the desired register address with the Ring Control labeled "Writing Register" and then enter a value. When you exit the mouse cursor from the window field the new value will be written to the MR330 controller. If the value was outside of a valid range the STATUS field will indicate error 1029 "wrong value". The error will clear once a proper value has been submitted.

If you like to commit the new value to the internal EEPROM click on the button "Save to EEPROM.

Note: This software is intended to demonstrate how to communicate with the MR330 controller. It was not designed to be specifically user friendly.

Explanation of Software

The Block-diagram in *Figure 3* reveals the software behind the front-panel.

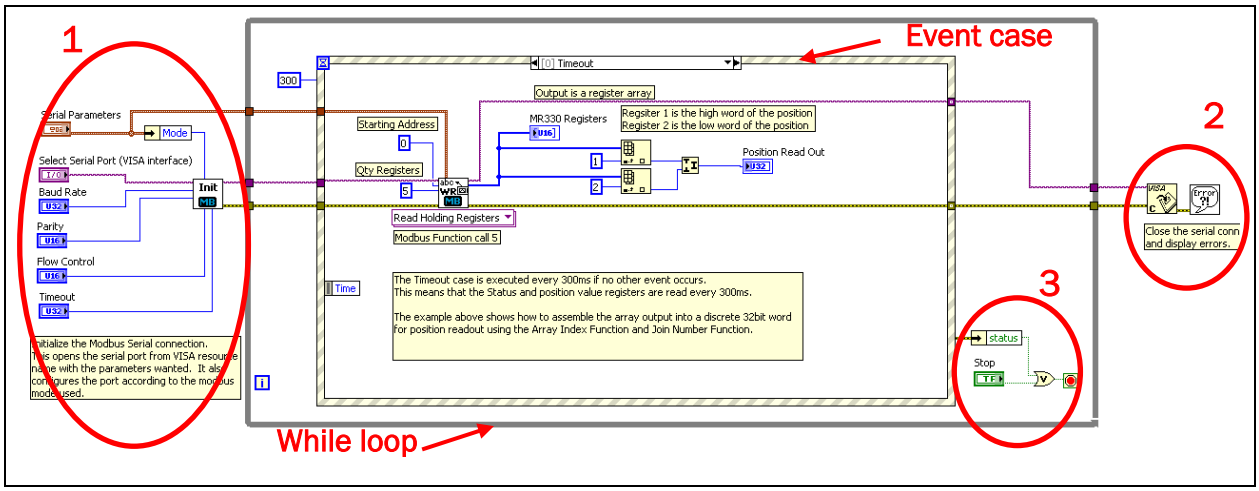

Figure 3: Block Diagram of Software Operation

1.) Serial Interface initialization. The values on the front-panel are propagated to the Modbus initialization VI. The VI "INIT MB" is provided by National Instruments.
2.) Error Reporting. Any error that occurs will be reported with a pop-up dialogue box. The program terminates at that point.
3.) While loop condition. To end the program the STOP button terminates the while loop and any error that occurs will likewise terminate the program.
4.) The while loop frames the program and repeats indefinite until the program is terminated.
5.) There are several event cases.
    a. Event case 0 is triggered every 300ms and reads the first 5 registers of the MR330 controller.
    b. Event case 1 is used to read any register of the unit
    c. Event case 2 is used to write any register of the MR330 unit
    d. Event case 3 is used to handle the EEPROM write (coil write)

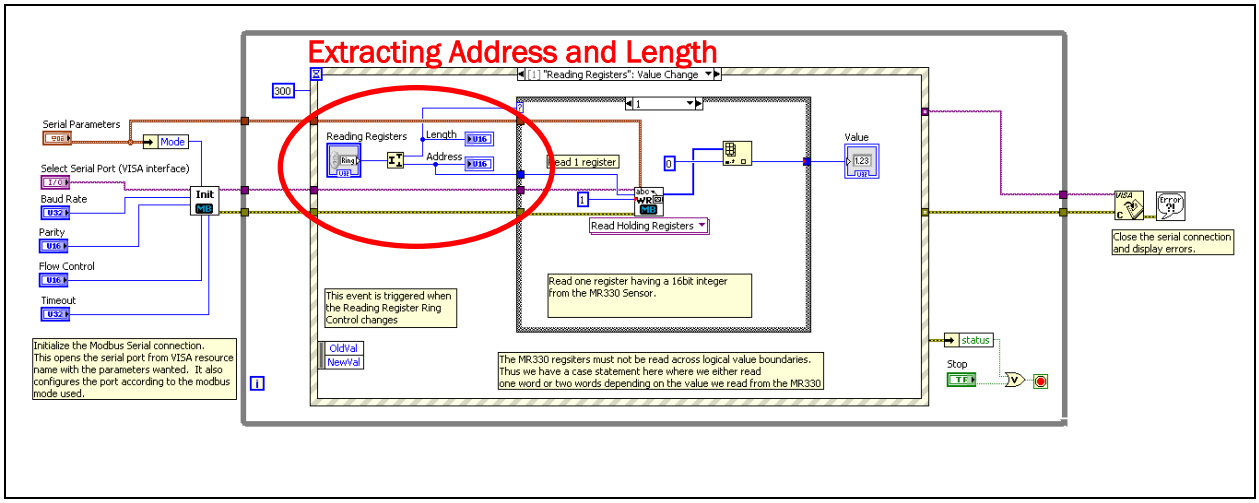

Figure 4: Reading registers using Modbus Function Call 3

Figure 4 shows event case 1 for reading any register using Modbus Function Call 3. The event is triggered whenever the Ring Control "Reading Registers" is being changed. The MR330 has values comprised of single 16bit registers and double 32bit registers. It is not allowed to read across a value boundary. Hence it is important to issue the proper register address for the beginning of a logical value and read the appropriate quantity of registers.

The demo software is organized as follows:
The LabVIEW™ Ring Constant was filled in to hold the information (description, address and register quantity) for each register. The value attached to each entry is a 32bit double word. The high order 16bit word holds the quantity of registers to read while the low order 16bit word holds the register address.

Example: Position, 0x00020001 → 2 is the number of registers, while 1 is the address.

When the event fires the 32bit word is split and the length and address are sent to the Modbus Write block. The black embedded "case of" block is used to distinguish between reading one register or two registers. The reason is because when two 16 bit registers are received they need to be assembled back to the 32 bit readout.

Figure 5 shows the "event case" 1 for reading registers with the case when two registers are returned. The array of two elements is being combined into a full 32bit word.

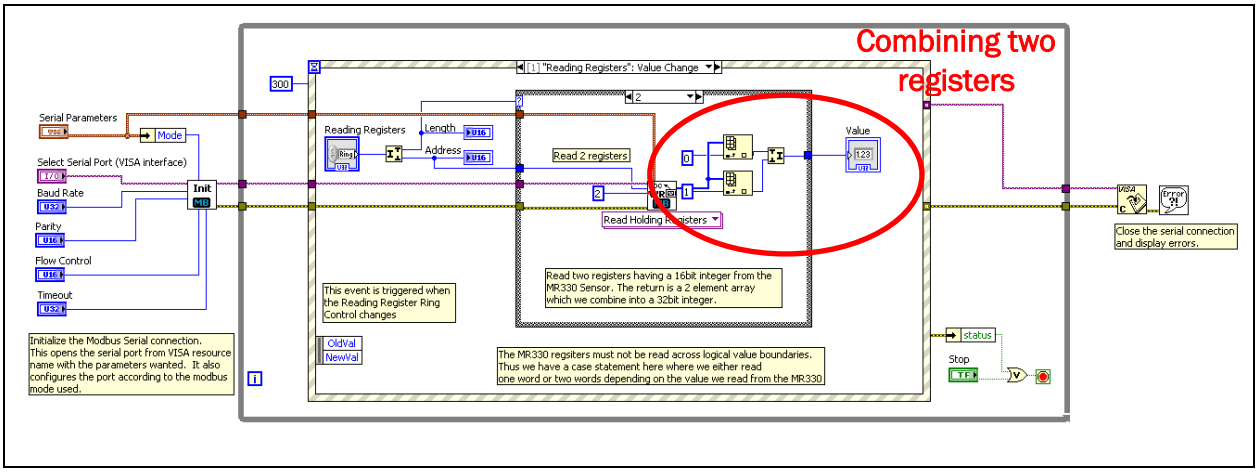

Figure 5: Reading Two Returning Registers

Event case 2 is used to write values (Modbus Function Call hex 10) to the MR330. Similar to the reading mechanism, the Ring Control holds the address and quantity of the registers. There are also two cases to distinguish for writing a single value register or writing a double value register.

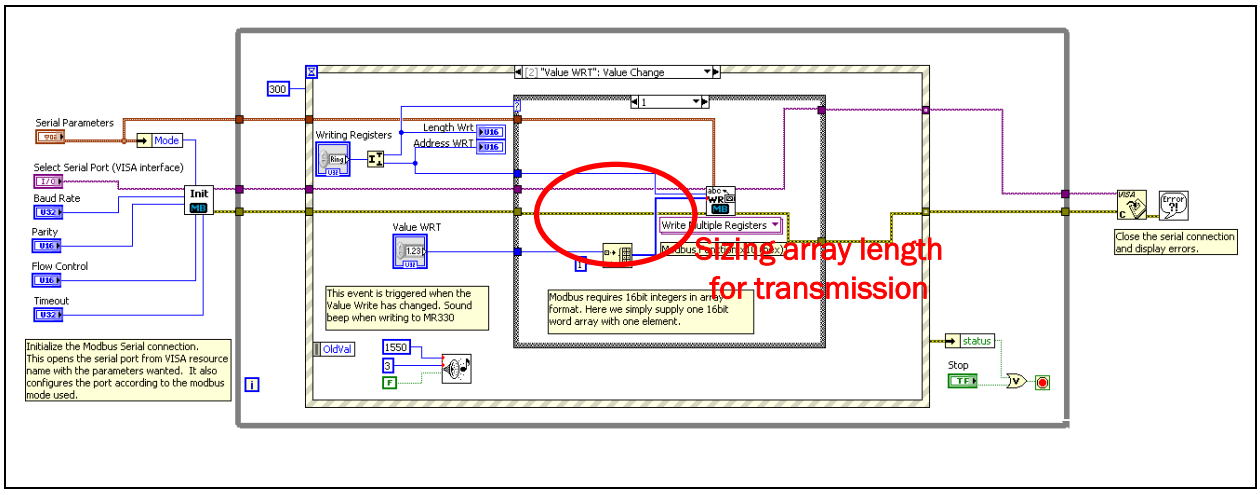

Figure 6: Writing Values to the MR330

The input of the LabVIEW™ Modbus block requires an array of variable length. The block will always transmit the full array length. We therefore select the number of elements of the array to send as indicated in *Figure 6.*

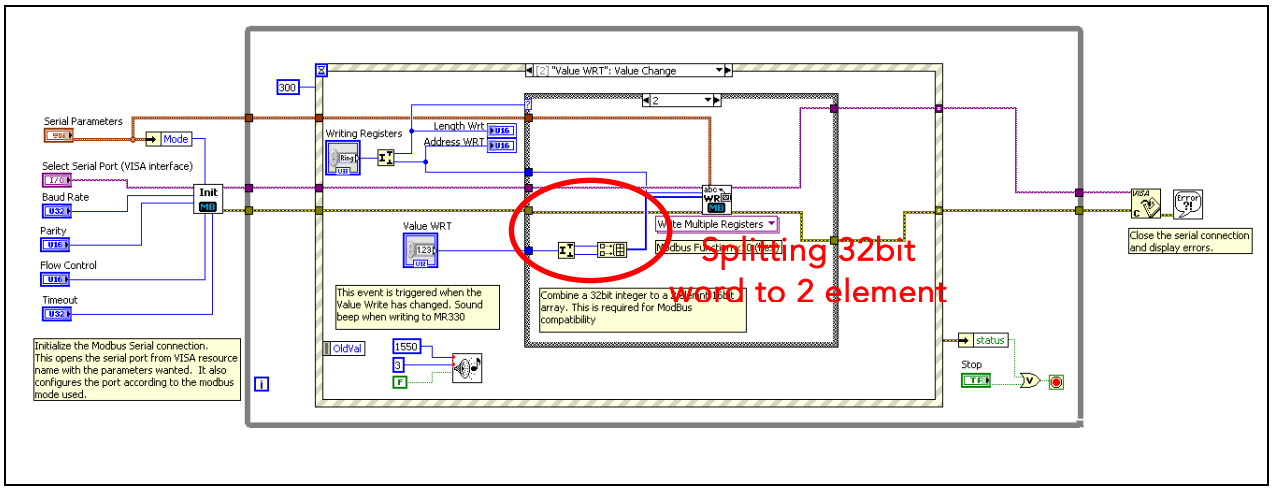

Figure 7: Separating 32 bit Word

When we need to send a 32 bit word we need to present this information as two 16 bit numbers in an array of length 2. This program step is indicated in *Figure 7*. The MR330 Controller supports a few simple coil commands (see appendix for a list). As an example we save the register values to the EEPROM by issuing coil command "ON" at address 0x02.

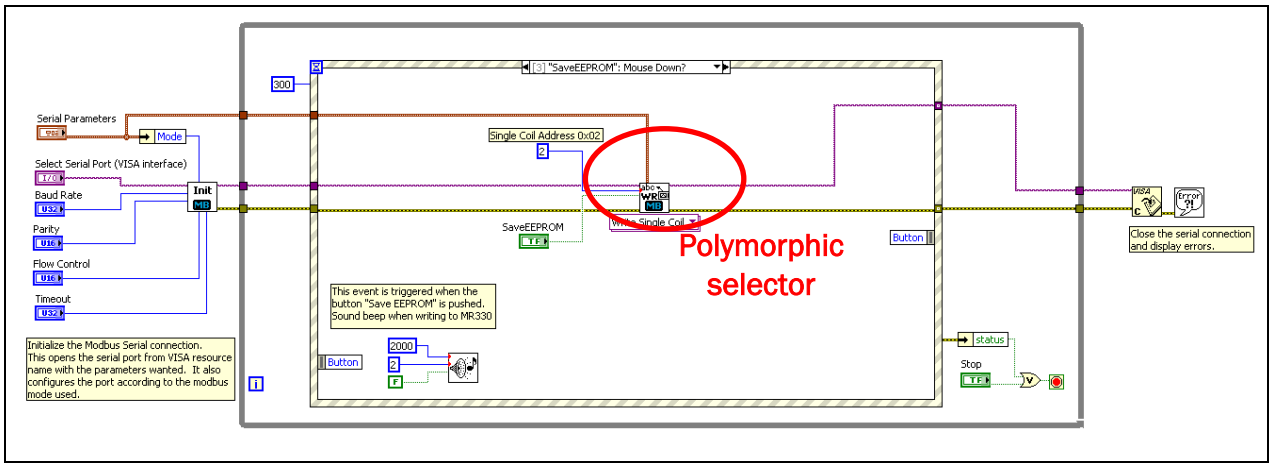

Figure 8: Saving Register Values to EEPROM

Case event 3 is triggered when the button SaveEEPROM is clicked. The coil address 0x02 is fixed by a constant.

Note: Be sure to select the appropriate command using the "polymorphic selector" for the Modbus Write block.

Communications Update Rate

The example program updates the reading approximately every 300ms. The MR330 can be polled faster but we recommend not faster than 25ms. The update rate is also dependent on the speed of the PC and the baud-rate.

With 57,600 baud the round trip for one poll of the first five registers is 5ms.

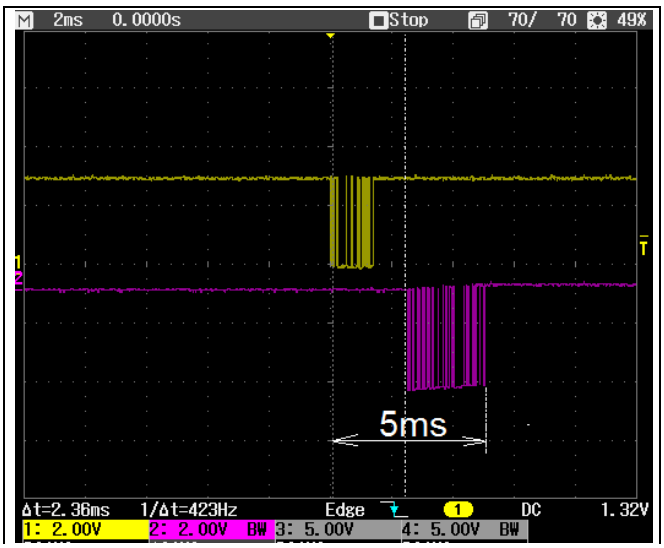

Figure 9: Round Trip of one Poll of the First Five Registers

The scope capture in *Figure 9* shows the poll request in the yellow top trace. The response by the MR330 unit is in purple bottom trace.

The unit responds within 1ms.

The reply in this case was 5 registers.

In a typical application we recommend to poll only the first three registers. This will shorten the transmission time.

Changing the Communications Baud-Rate

Changing the communications baud-rate can be somewhat tricky, because as soon as the baud-rate is being changed the computer and the MR330 may no longer communicate with each other.

To solve this dilemma the MR330 baud-rate will only take effect after a new power up of the unit. Even after changing the baud-rate the controller will continue to communicate with the current baud-rate.

To change to a new baud-rate, select the Write Register "Baudrate Serial" address 0x0193.

Select the baud-rate code as follows:

Code	Baud-Rate
0	9,600
1	19,200
2	38,400
3	57,600
4	115,200

Exit the cursor from the Value Field in order to send the code to the MR330 controller. A short beep will indicate the write command.

Click the Save to EEPROM button in order to commit the new baud-rate.

Stop the LabVIEW™ software and cycle the power on the MR330 unit.

Select the new baud-rate on the software front panel and start the software. It will now communicate with the new baud-rate.

Note: When using the USB connection then VISA will lose the connection as the MR330 controller is powered down. It may be necessary to start the software twice until the USB is registered again with the VISA driver.

Information about Specific Registers

FUNCTION FC03 – Read Holding Registers
FUNCTION FC10 – Write Holding Registers

Holding registers FC03 are used for reading the position and all other parameters.

These Registers can be written using Function FC10 using identical address offset

Register Address	Register Number	Name	# regs	Range	Description
0x000	0x001	System Status	1	n/a	Returns the system status. num register = 1 : reads status only num register = 3 : reads status & position num register = 4 : reads status & position & angle num register = 5 : reads status & position & angle & amplitude
0x001	0x002	Get Position	2	n/a	Returns position count as a 32 bit integer
0x003	0x004	Get Angle	2	n/a	Returns position angle in 1/100 of degrees (0..36000)
0x005	0x006	Get Last Position	2	n/a	Returns the very last position when unit was shut down. Used when determining if the turn counter restore is feasible.
0x040	0x041	Get Error Counts	24	n/a	Returns 24 registers with the total number of errors for each error class.
0x100	0x101	Set New Position	2	0..MaxCount	The value is used as the new position readout. The MR330 automatically calculates a position offset.
0x102	0x103	Get Position Offset	2	n/a	Position offset used to adjust for desired position readout.
0x104	0x105	Device Address	1	1 – 254	Sets the MR310 serial address for commands. Note that the address 4 cannot be used.

					A FC06 command to save EEPROM must be issued following this command.
0x105	0x106	Operating Mode	1	0..3	Used to setting MR330 in calibration, or troubleshooting mode. Normal Operating is 0. Debug mode is 1. Pairing Operation is 2. Do not put unit in any of these modes without first consulting the user manual. Be familiar with what these functions before using.
0x106	0x107	Get Temperature	1	n/a	Reads the temperature in degrees Celsius within the unit.
0x130	0x131	Scan Start	1	1 – 200	Factory use only – do NOT write to it. Determines the start of the disk code reading
0x132	0x133	Voltage Offset	1	-128 – 127	Factory use only – do NOT write to it. Hardware calibration value for voltage output
0x133	0x134	Voltage Gain Pos	1	-128 – 127	Factory use only – do NOT write to it. Hardware calibration value for voltage output
0x134	0x135	Voltage Gain Neg	1	--128 – 127	Factory use only – do NOT write to it. Hardware calibration value for voltage output
0x135	0x136	Current Gain	1	-128 – 127	Factory use only – do NOT write to it. Hardware calibration value for current output
0x136	0x137	Turn Counter	1	0 - 11	Depth of Turn Counter in binary increments (2^n) User sets this value depending how far the turn counter should count until resetting to 0 again Example: n = 3: Maximum Position readout is: 8 revolutions with 8192 per revolution. Therefore maximum position readout will be 65,536.
0x137	0x138	Resolution of analog outputs	1	n/a	Read only for trouble shooting. Provides the maximum resolution of the current and voltage output. This value is internally set whenever the voltage or current scale is being defined.
0x138	0x139	Baudrate SSI	1	25 – 250	Defines the SSI Baudrate. This value should be set by the user and it should match the clock frequency of the SSI master reading the position output.
0x139	0x13A	Baudrate Serial Communications	1	0 – 3	Sets the Baudrate for Serial Communications on the MODBUS. 0 = 9,600 1 = 19,200 2= 38,400 3 = 57,600 4 = 115,200

0x10A	0x10B	Optical Amplitude	1	n/a	Gets the reading for the optical signal amplitude. Should be in the range of 300 to 600. Provides a useful value indicating the quality of the optical link. Maybe read in conjunction of register 0x000 when register length of 5 is specified.
0x10B	0x10C	Pairing Progress	1	n/a	Returns the currently executing calibration step. Read Only, only active during the pairing process.
0x10C	0x10D	Reserved			
0x10D	0x10E	Reserved			
0x10E	0x10F	Reserved			
0x200	0x201	Voltage Mode	1	0 - 3	Defines the output mode for the voltage output. 0 = OFF no Position Output 1 = Single Turn 0 to 10V 2 = Multi Turn 0 to 10V 3 = Multi Turn -10V to +10V
0x201	0x202	Voltage Scale	2	0 – MaxCount	Establishes the scale used for the voltage output. Regardless of Voltage Mode setting 10V refers to the scale value. When the position count reaches the scale value the output is 10V.
0x204	0x205	Current Mode	1	0 – 2	Defines the output mode for the current output. 0 = OFF current is < 300uA. 1 = Single Turn 4 to 20mA 2 = Multi Turn 4 to 20mA
0x205	0x206	Current Scale	2	0 – MaxCount	Establishes the scale used for the isolated current output. Regardless of current Mode setting 16mA refers to the scale value. When position count reaches the scale value then the output is 16mA plus 4mA bias for a total of 20mA.
0x208	0x209	Reset Mode	1	0 – 1	Defines how the hardware input resets the internal counter. 0 = Edge Triggered, resets the counter at the first rising edge 1 = Debounced Trigger when state changes from 0 to 1 after 60ms debounce time. (used for switch or relay input)
0x209	0x20A	Preset Value	2	0 - MaxCount	Counter will be preset to this value when the Zero push button is pressed or when hardware input is activated. (See Reset Mode)
0x20B	0x20C	Turn Direction	1	0 - 1	Defines output results based on turning direction of the sensor

					0 = when CW outputs are positive reading. 1 = when CCW then outputs are positive reading
0x20C	0x20D	Power Up Mode	1	0 – 1	Defines if controller should attempt to restore the turn counter after power-up. 0 = do not restore turn counter. 1 = attempt to restore turn counter. When within the restore range then restore full position, otherwise indicate an error.
0x230	0x231	Set Point 1 On	2	0 - MaxCount	Lower threshold for digital limit switch output 1
0x232	0x233	Set Point 1 Off	2	0 - MaxCount	Upper threshold for digital limit switch output 1
0x234	0x235	Set Point 2 On	2	0 - MaxCount	Lower threshold for digital limit switch output 2
0x236	0x237	Set Point 2 Off	2	0 - MaxCount	Upper threshold for digital limit switch output 2
0x237	0x238	Restore Range	1	0 - 4095	Defines the range within the automatic turn counter restore will be considered valid.
0x300	0x301	Ref Voltage	1	n/a	Internal Reference voltage of 2.5V Updated only at Power ON
0x301	0x302	5 Volt Supply	1	n/a	Internal Supply Voltage 5V Updated only at Power ON
0x302	0x303	12 Volt Supply	1	n/a	Internal Supply Voltage 12V Updated only at Power ON
0x303	0x304	24V Power Supply	1	n/a	External applied Voltage 24V nominal Updated only at Power ON
0x304	0x305	n/a	1	n/a	
0x305	0x306	n/a		n/a	
0x306	0x307	n/a		n/a	
0x307	0x308	n/a		n/a	
0x330	0x331	DAC 1, Chan 1	1		Internal Digital to Analog Converter Value Positive Voltage Output
0x331	0x332	DAC 1, Chan 2	1		Internal Digital to Analog Converter Value Negative Voltage Output
0x332	0x333	DAC 1, Chan 3	1		Internal Digital to Analog Converter Value CCD Bias voltage
0x333	0x334	DAC 1, Chan 4	1		Internal Digital to Analog Converter Value Optical source bias Voltage
0x334	0x335	Reserved			
0x339	0x33A	Optical Pulse Time	1	0..65	Factory use only! Determines the Optical Pulse Strength for Sensor interrogation.

0x400	0x801	Device Name	4	n/a	Returns the ASCII string equivalent as device name (MR330)
0x404	0x805	Version	4	n/a	Returns the ASCII string equivalent of the software version form MM.mm.bb
0x808	0x809	Serial Number	2	n/a	Returns the serial number of the device.

FUNCTION FC05 – Write Single Coil

Single Coil commands are used to trigger an action.

Register Address	Register Number	Name	Description
0x001	0x002	Device Reset	Same as a Power OFF and Power ON cycle.
0x002	0x003	Save to EEPROM	Save current parameters to EEPROM. A time delay of approximately 20ms should be allowed before sending ny other command.
0x003	0x003	Restore From EEPROM	Restore all configuration parameters from EEPROM. Same as a Power Up.
0x004	0x004	Restore Factory Default	Restores Factory Defaults. The MR330 stores a factory default for each user parameter. These values maybe restored using this command. Factory calibration values and pairing data are not affected.
0x004	0x005	Clear Status	Clears the status register. If another error is pending then the status register will reflect that new value in queue.
0x005	0x006	Clear Error Count Table	Resets error table counters to 0. Same as in power up.

Communication Failure Codes

#	Description	S	Remedy	How Cleared	Announced
1025	CMD Unknown Function A non valid or non implemented ModBus function was sent to the controller	1	Check your software for correct function calls.	self clear after one blink	Blink 1x once
1026	CMD Unknown Register A non implemented register address was addressed	1	Check your software for correct register addressing. See user manual with address table.	self clear after one blink	Blink 1x once
1027	CMD Wrong Register Count The register count in your command did not match the length of requested register.	1	Check your software for correct register addressing. See user manual with address table. ***Note:*** This controller does not allow to read across multiple registers.	self clear after one blink	Blink 1x once

1029	CMD Wrong Value The data value was outside the permissible range for this parameter.	1	Consult the user instruction for the permissible parameter values allowed in each register.	self clear after one blink	Blink 1x once
1030	CMD Checksum ModBus Packet Checksum was invalid.	1	Resend the packet.	self clear after one blink	Blink 1x once

MICRONOR INC, 900 Calle Plano, Suite K,
Camarillo, CA 93012 USA
T +1 805 389 6600 F +1 805 389 6605
sales@micronor.com www.micronor.com

MICRONOR AG, Pumpwerkstrasse 32,
CH-8105 Regensdorf, Switzerland
T +41 44 843 4020 F +41 44 843 4039
sales@micronor.ch www.micronor.com