International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 4, April - 2013

Development Of Master GUI Of "MODBUS Over Serial Line ASCII Mode Protocol" Using Labview^{tm} For Data Acquisition

Nital Patel
Assistant Professor
Nirma University, Ahmedabad

Chintan Desai
M.Tech. (Control and Automation)
Nirma University, Ahmedabad

Abstract

A GUI tool is developed in the LabVIEW^{TM} to support the functionalities like read register, write register and write multiple registers. The GUI tool developed can be useful for testing the health of the MODBUS slave devices by using test mode. The tool is also useful for continuously monitoring the on field parameters by running the tool in auto mode. In auto mode the MODBUS Master GUI tool continuously send query for reading particular registers to the slave devices connected to the serial port selected by the user in the tool. The values of the registers read by the Master Module are displayed in the chart. User can select the Baud rate, parity checking, Number of data bits and time out. The Master GUI is tested and validated by the slave simulator available at [1].

1. Introduction

MODBUS is a serial communications protocol published by Modicon in 1979 to use it with its PLCs. It has now become a *de facto* standard communication protocol. Its development with industrial application kept in mind and being published openly free of royalty, made it very popular. This document describes the MODBUS over Serial Line protocol. MODBUS Serial Line protocol is a Master-Slave protocol. This protocol takes place at level 2 of the OSI model. A master-slave type system has one node (the master node) that issues explicit commands to one of the "slave" nodes and processes responses. Slave nodes will not typically transmit data without a request from the master node, and do not communicate with other slaves.

At the physical level, MODBUS over Serial Line systems may use different physical interfaces (RS485, RS232). TIA/EIA-485 (RS485). Two-Wire interface is the most common. As an add-on option, RS485 Four-Wire interface may also be implemented. A TIA/EIA-232-E (RS232) serial interface may also be used as an interface, when only short point to point communication is required [2].

2. MODBUS ASCII Mode

In case of MODBUS ASCII mode each hex byte of RTU mode frame is converted in two 7 bits ASCII characters. For e.g. 0x30 of the RTU mode is converted into '3' and '0'. Other than that for MODBUS ASCII mode frame begins with ':' and ends with CR/LF. Further the ASCII mode uses LRC checking for the communication error in the received frame. [2]

2.1. MODBUS Master State Transitions

MODBUS master Transition state begins with idle mode. When master needs to send query to the slave it frames the query frame following the MODBUS protocol [2]. The frame is than sent to the lower physical layer and sent via communication channel to the slaves. It should be taken care of that baud rate, parity and line coding must be same on the both the side.

Once the Query is framed and sent to communication channel the master now waits for the response from the slave to which the query has been sent. The master waits for the predefined time interval for the response to be received. Once the timeout occurred the master will assume that communication error has been occurred and will take

necessary action. Master may resend the query or master will go to idle state.

When master receives the response, it processes the response. First it checks the LRC field if LRC is not matched than master will discard the frame and goes to the idle state. If LRC is matched than it checks for the address field. If address field is not matched than master discards the frame and goes to the idle state. If address field is matched than the master processes the other fields and takes necessary action.

3. Master Module Architecture And Functionality

The Master module designed in LabVIEW™ works in either continuous mode or test mode.

3.1. Continuous Mode

In case of continuous mode, the master first sends Query to the slave with the slave ID 0x01 for reading holding registers addressed from 0 to 3. The master now waits for the response till the timeout value defined by the user occurs. If timeout occurs before the response frame is received than "frame not received" message is displayed in the message window of the slave ID 0x01. If frame is received than the frame is processed and LRC and slave ID and function code field is than checked, If any of the three is unexpected than corresponding message is displayed on the message window. If these three fields are same as expected by the master than the master further process the other fields and extracts the values of the register for which the query was sent. The extracted values of the register are first converted in to the hex value as it was received in ASCII character format. The hex value is converted in the decimal value and displayed on the chart. Irrespective of the response received the master sends the query to the slave ID 0x02 for reading the same holding registers as in the case of with the slave id 0x01. The same procedure is followed for the slave ID 0x02 as well. This process continuous till the stop button is pressed or the execution of the program is terminated. The response received by the master is processed.

3.2. Test Mode

Test mode of the master GUI supports 3 MODBUS functions. 1) Read Holding Registers. 2) Write Single Register. 3) Write Multiple Registers.

User is allowed to select the slave ID to which he/she intends to send the query frame. After selecting slave ID, user selects the one of the function code tab. If user selects read register tab, than in that tab user needs to select number of register he wants to read and the starting address of the holding register. After selecting these two parameters user needs to execute the file. The Master GUI will create the MODBUS ASCII over serial frame and send itto the communication channel (COM PORT). The slave after receiving the query frame responds to that query frame. The response is either a normal response or an exception response. When Master receives the normal response, the values of the registers are displayed on the register indicator window. The message window displays the message "normal response received". When master receives exception response the message window displays the exception number and the meaning of that exception [2].

Similarly, for the queries 'Write Single Register' and Write Multiple Register' user selects the address and the values to be written. The message window describes the nature of the response i.e. either normal response or exception response.

In case of timeout occurs before the frame is received the message window displays the "Frame not received".

4. Experimental Setup For Modbus Validation

4.1. Physical layer

A MODBUS slave simulator [1] is used to validate the authenticity of the master designed in the LabVIEW™. MODBUS slave simulator trial version is available for 30 days. The link for the same is given in the *Reference*.

Two PC's are connected through serial COM PORT thus establishing RS232 physical layer. One of the connected PC is used to run the MODBUS Master GUI prepared in LabVIEW™ and the other PC runs the MODBUS slave simulator.

4.2. Results

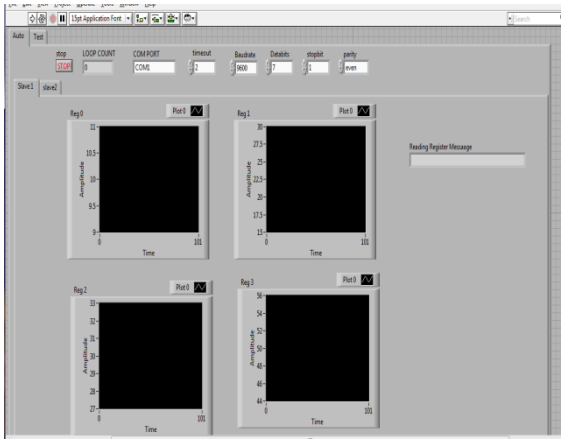

Fig. 1 Master module front panel for auto mode.

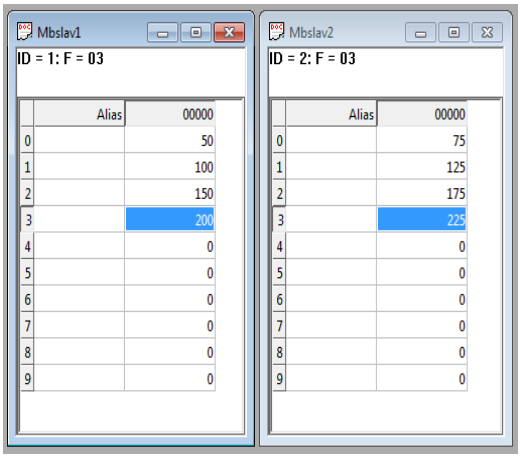

fig. 2 Slave simulator, holding register values

The figure 1 shows the front panel tab for the auto mode. The chart box for reading the holding registers addressing from 0 to3. The provision of the similar tab is provided for the reading the same registers for the slav ID 0x02.

The figure 2 shows the values of the holding registers for the slave ID 0x01 and 0x02. As shown in the figure 3 after the starting of the execution of GUI the values of the registers are displayed on the chart. The figure 3shows the register values for the slave ID 0x01. The same visuals are also available on the tab for slave ID 0x02

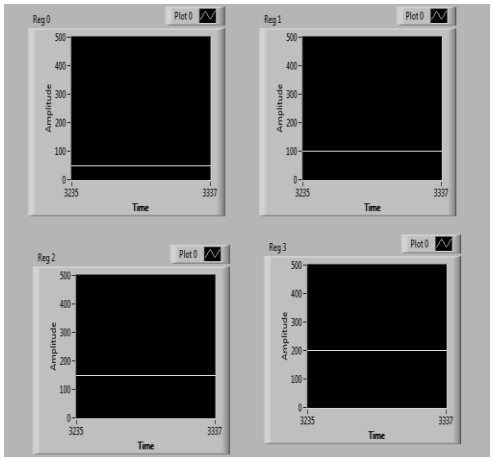

Fig. 3 Display chart for slave ID 0x01 on MODBUS master

The figure 4 shows the front panel for the test mode. As shown in the figure the test mode panel provides facility to query the slaves with different ID's. It has capability to send the query for 3 different functions. It supports function code 0x03(read holding register), 0x06(write single register) and write multiple register(0x10).

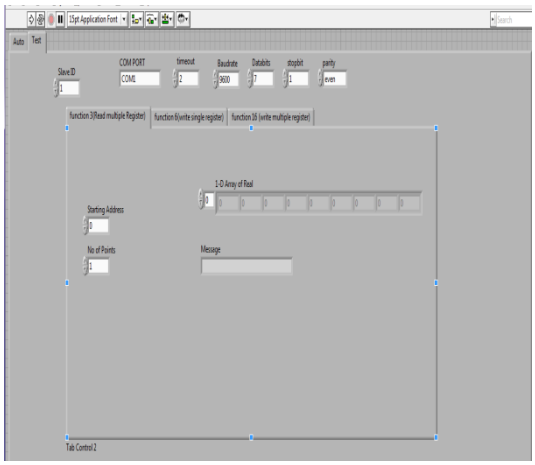

Fig. 4 Front panel for test mode

The figure 5 shows the current register content of the slave ID 0x01. Figure 6 shows the same content value read by the master and shown in the register window. In the figure 7 the query for writing holding registers 0 to 9. The register value changed in the slave simulator. The changed values of the registers are shown in the figure 8.

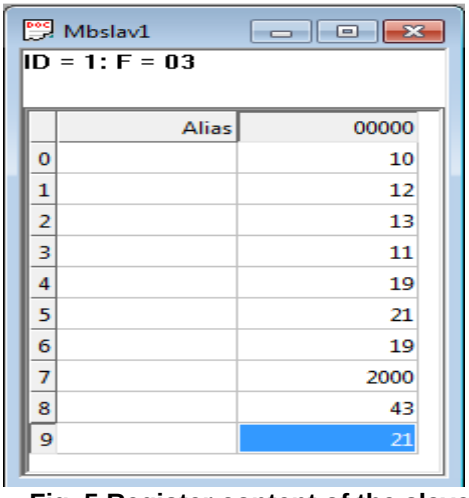

Fig. 5 Register content of the slave ID 0x01

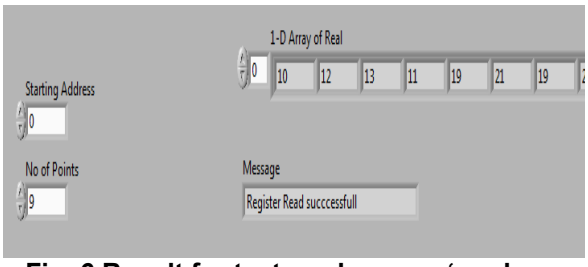

Fig. 6 Result for test mode query 'read register'

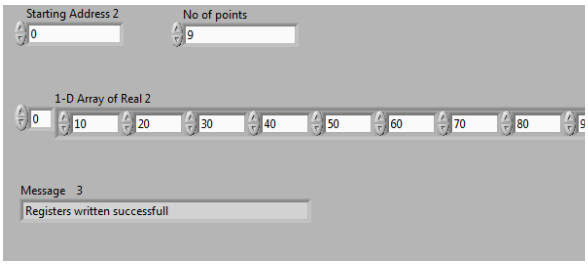

Fig. 7 Result for test mode query 'write multiple register'

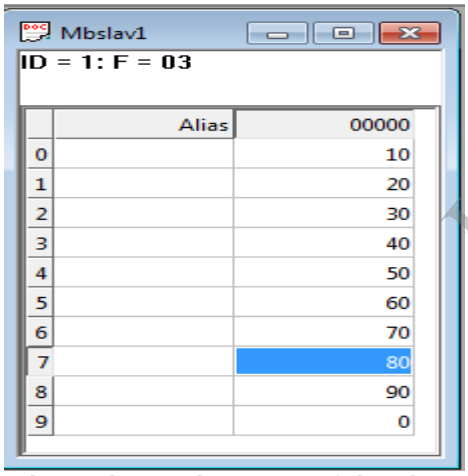

Fig. 8 Changed content of the slave holding registers after query 'write multiple register'

If we try to read or write the register that is not supported by the slave, the slave sends the exception response. The master then displays the message for the exception. In this test example the slave only supports holding registers 0 to 9 so if user tries to read or write other than those registers then slave simulator send exception response. Figure 9 and 10 shows the result of exception response for read register and write register respectively.

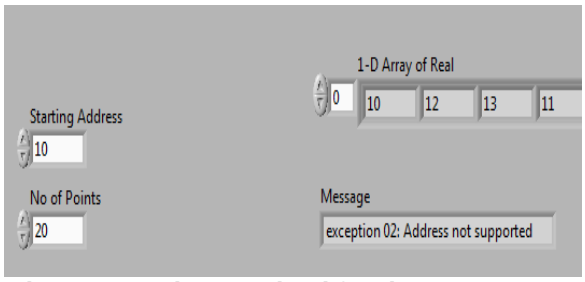

Fig. 9 Exception received for the query 'read registers'

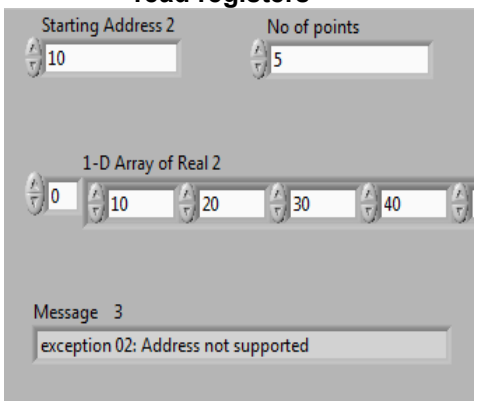

Fig. 10 Exception received for the query 'write registers'

5. Conclusion

The Master module developed for the MODBUS over Serial Line ASCII Mode Protocol is quite useful for the process data monitoring. The module can continuously monitor the process data from the different slave connected on the serial line using MODBUS ASCII protocol. Further development in tool may allow process parameters to be controlled in the close loop rather than just monitoring.

6. References

[1] http://www.modbustools.com/download.asp

[2] MODBUS over Serial Line Specification and Implementation Guide V1.02
http://www.modbus.com/docs/Modbus_over_serial_line_V1_02.pdf