

Serial Port Data Communication using MODBUS Protocol

Saptarshi Naskar, Krishnendu Basuli and Samar Sen Sarma

Department of Computer Science and Engineering,
University of Calcutta, 92, A. P. C. Road,
Kolkata – 700 009, India

Serial communication is the process of sending data sequentially one bit at a time, over a communication channel or computer bus [5,6,7]. RS-232 is a standard for serial binary data transfer between a data terminal equipment (DTE) and a data circuit-terminating equipment (DCE), commonly used in computer serial ports.

The original RS-232 standard defined only the connection of DTEs with DCEs, i.e., modems. Null modem is a communication method to connect two DTEs (computer, terminal, printer, etc) directly using RS-232 serial cable. The null-modem configuration simplifies the handshaking between computers. In null modem mode, a minimal 3-wire RS-232 port consisting only of transmit data, receive data and ground is commonly used when the full facilities of RS-232 are not required [5,6,7].

The other kind of common connection is a DTE-DTE connection, for instance connection between two PCs, in order to exchange data between them. For that kind of connection, so-called null-modem connection is needed; actually this connection is done in this project.

The third kind of connections is DCE-DCE. Here so-called tail circuit cable is needed, but this is a very uncommon kind of connection.

Other parts of RS-232 specifications [5,6,7]: (a) Signal voltages: –5V to –15V (logical 1), +5V to +15V (logical 0) on the other side –3V to –15V (logical 1), +3V to +15V (logical 0) on the receiver side. Typically on a standard PC +/- 12V is used. (b) Maximal cable length: 50 feet at 19200 bps, 3000 feet at 2400bps (it can be much higher without troubles in most cases). (c) Connectors: The most common RS-232 connectors are the DB-9 and DB-25. There exists each as male and as female versions, in most cases the DTE has a male and the DCE has a female connector (although this can be different in some other cases). (d) Signals in different pins of the serial port are given below in the following table.

DB-9	DB-25	Signal	Direction	Description 200LX
1	8	DCD	Modem → PC	Data Carrier Detect
2	3	RxD	Modem → PC	Received Data (by DTE)
3	2	TxD	PC → Modem	Transmitted Data (by DTE)
4	20	DTR	PC → Modem	Data Terminal Ready
5	7	GND		Signal Ground
6	6	DSR	Modem → PC	Data Set Ready
7	4	RTS	PC → Modem	Request To Send
8	5	CTS	Modem → PC	Clear To Send
9	22	RI	Modem → PC	Ring Indicator
10	1	Port. GND		Not on the DB-9. Protective ground: Shield of plugs and cables are connected.

DCD, DTR, DSR, RTS and CTS are also so-called handshaking lines, which the devices use to exchange information about their status.

The common language used by the Modicon controllers is the MODBUS protocol [1,7]. This is an open protocol and it defines a message structure that controllers will recognize and use,

regardless of the type of networks over which they communicate. It describes the process a controller uses to request access to another device, how it will respond to requests from the other devices, and how error will be detected and reported. It establishes a common format for the layout and contents of message field.

The MODBUS protocol provides the internal standard that the Modicon controllers use for parsing messages. During communications, the protocol determines how each controller will know its device address, recognize a message addressed to it, determine the kind of action to be taken, and extract any data or other information contained in the message. If a reply is required, the controller will construct the reply message and send it using MODBUS protocol. In message transmissions, the MODBUS protocol embedded into each network's packet structure provides the common language by which the device can change data.

Standard MODBUS ports on Modicon controllers use an RS-232C compatible serial interface that defines connector pin-outs, cabling, signal levels, transmission baud rates, and parity checking. Controllers can be networked directly or via modems. Controllers communicate using a master-slave technique, in which only one device (the master) can initiate transactions (queries). The other devices (the slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. Typical master devices include host processors and programming panels. Typical slaves include programmable controllers.

The master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a message (response) to queries that are addressed to them individually. Responses are not returned to broadcast queries from the master.

The MODBUS protocol establishes the format for the master's query by placing into it the device (or broadcast) address, a function code defining the requested action, any data to be sent, and an error-checking field. The slave's response message is also constructed using MODBUS protocol. It contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.

Controllers can be setup to communicate on standard MODBUS networks using either of two transmission modes: ASCII or RTU [1]. Users select the desired mode, along with the serial port communication parameters (baud rate, parity mode, etc), during configuration of each controller. The mode and serial parameters must be the same for all devices on a MODBUS network. Now the message formats for the two modes are discussed.

In ASCII mode, messages start with a *colon* (:) character (ASCII 3A hex), and end with a *carriage return-line feed* (CRLF) pair (ASCII 0x0D and 0x0A). The allowable characters transmitted for all other fields are hexadecimal 0 through 9, and then A through F. Networked devices monitor the network bus continuously for the *colon* character. When one is received, each device decodes the next field (the address field) to find out if it is the addressed device. Intervals of up to one second can elapse between characters within the message. If a greater interval occurs, the receiving device assumes an error has occurred.

In RTU mode, messages start with a silent interval of at least 3.5 character times. This is most easily implemented as a multiple of character times at the baud rate that is being used on the network (shown as T1-T2-T3-T4 in the figure below). The first field then transmitted is the device address. The allowable characters transmitted for all fields are hexadecimal 0 through 9, and then A through F. Networked devices monitor the network bus continuously, including during the *silent* intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device. Following the last transmitted character, a similar interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval. The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes

that the next byte will be the address field of a new message. Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value in the final CRC field will not be valid for the combined messages.

Primary aim of the project is the mapping of the magnetic field within a magnet (Magnet having the hollow cylindrical shape) [2,3,4,7]. In this scheme, one computer as Master terminal and another as Slave terminal are taken for communication. The Master terminal on MS Windows XP platform and contains an operator interface designed using Borland C (Var. 4.5). The Slave terminal in MS Windows 98 platform and it actually controls the mechanical system and acquires data after decoding the command issued by the Master terminal. The Master and Slave terminals are connected with peer-to-peer serial bus through their serial communication port (COM ports). The Slave computer again is connected to a voltage-to-frequency module, a lab made read out circuit and a motor control circuit through a PCL 812 PC add-on card. To map the magnetic field we have used a search coil (10,000 turns) as sensor to the magnetic field. The voltage induced in the search coil moving along the radial axis on the median plane of the magnet is feed to the voltage-to-frequency converter module. This module in turn generates pulses of varying frequencies depending upon the output voltage of the search coil. The pulses generated by the module, are then counted by using the counter of PCL 812. The count gives the relative measure of the magnetic field at a particular point. The direction and the movement of the search coil motor is again controlled by a DA Converter circuit, a power amplifier and a relay, controlled by the same add-on card. The position of the search coil is detected by an optical encoder fitted to the shaft of the driving motor of the search coil. The optical encoder produces quadrature pulses on two channels, which are again counted by the optical encoder reader circuit and hence transferred to the Controller PC by using digital input port of PCL 812. The mechanical arrangement is made in such a way that, optical encoder generates 400 pulses for travel of 53 mm of the search coil. The output pulses of the optical encoder again are used to generate hardware interrupt (IRQ 3) and on each 10th interruption of the encoder read out circuit data and PCL 812 counter data are read and stored on a local buffer in Slave terminal. These data are transmitted to the Master terminal on demand.

In this project only the ASCII mode of the MODBUS protocol is implemented. The RTU mode of the MODBUS protocol can also be implemented further, because the RTU mode has an advantage over the ASCII mode in error checking methodology. In the ASCII mode the error checking is done by the LRC (Liner Redundancy Check) method, but in the case of RTU mode the error checking is done by the CRC (Cyclic Redundancy Check) method. In case of RTU mode the synchronization is needed for data communication between Master-Slave terminals, this is a disadvantage of this mode over the ASCII mode. However, the implementation of the RTU mode of the MODBUS protocol will make the communication more rigid, faster, and secure for its basic message framing technique.

Reference:

- [1] Modicon MODBUS Protocol Reference Guide, June 1996, MODICON, Inc., Industrial Automation Systems One High Street, North Andover, Massachusetts 01845.
- [2] Microprocessor and Interfacing Programming and Hardware, Douglas V. Hall
- [3] Using Assemble Language, Allen L. Wyatt
- [4] Network Programming in C, Barry Nance
- [5] Interfacing the Serial / RS232 Port V5.0
<http://wearcam.org/seatsale/programs/www.beyondlogic.org/parlcd/parlcd.htm>
- [6] Virtual Null Modem, © 2004-2006 AGG Software, Publisher AGG Software Production, © 2004-2006 AGG Software, <http://www.aggsoft.com>
- [7] Naskar S, A. Roy, S. Das Gupta, MODBUS Protocol for Master-Slave Communication in Prototype MFM System, VECC (BDCC), Kolkata, India.