THINK INDIA (Quarterly Journal)

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Analysis Of Coding Efficiency For Implementation Of Serial Portdata Communication Using Modbus Protocol

Sudipta Acharjee, Saptarshi Naskar,

Guru Nanak Dev Institute Of Technology

Abstract: -

Serial communication is the process of sending data sequentially one bit at a time, over a communication channel or computer bus. 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. Primary aim of the work is the mapping of the magnetic field within a magnet (Magnet having the hollow cylindrical shape). In this scheme, one computer as Master terminal and another as Slave terminal are taken for communication.

Keywords: RS-232, DTE, DCE, Modbus Protocol

Introduction: -

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 paper.

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.

Description of RS 232 Port:

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

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

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.

Study of MODBUS Protocol:

The common language used by the Modicon controllers is the MODBUS protocol [1,7]. This is an open protocol and it defines a massage 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 massage field.

The MODBUS protocol provides the internal standard that the Modicon controllers use for passing massages. During communications, the protocol determines how each controller will know its device address, recognize a massage addressed to it, determine the kind of action to be taken, and extract any data or other information contained in the massage. If a reply is required, the controller will construct the reply massage and send it using MODBUS protocol. In massage transmissions, the MODBUS protocol embedded into each networks 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.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

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 massage 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.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Proposed work:

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 massage framing technique.

Software & Hardware Requirement Specification:

The project is designed completely in Borland C. For this reason, a minimum hardware configuration of "Pentium I" processor is needed. Hardware with better configuration will provide much better platforms to the software to operate. However, the hardware configuration that is to be used to develop the project is as follows:

For Master terminal

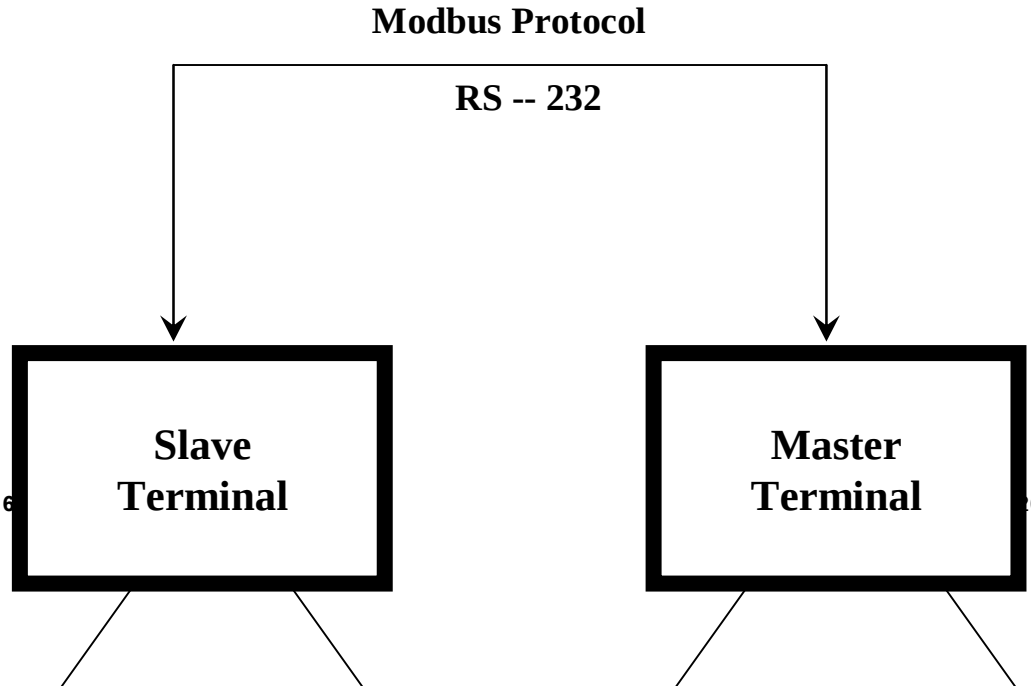
ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Intel Pentium I Processor with 133 MHz (processor speed).
Hard Disk: 20 GB (Seagate).
Samsung 52X CD-ROM drive.
Samsung 1.44MB Floppy drive.
RAM: 128 MB (Hyundai).
Peripherals: Genius PS-2 Mouse, Keyboard: Frontech.
Display: 14" Samsung Monitor.

For Slave terminal
Intel Pentium I Processor with 133 MHz (processor speed).
Hard Disk: 10 GB (Seagate).
Samsung 52X CD-ROM drive.
Samsung 1.44MB Floppy drive.
RAM: 64 MB (Hyundai).
Peripherals: Genius PS-2 Mouse, Keyboard: Frontech.
Display: 14" Samsung Monitor.

After the description of the hardware configuration comes the description of the software configuration. The operating system to be used for the development of the project is Windows 98.
Windows 98 operating system: Features: Innovative, easy-to-use features (like multiple display support, accessibility wizard, web integration, etc.), Improved Reliability (like scan disk, backup, system file checker, etc.), faster operating system (Disk defragmenter, Maintenance wizard, etc.). The language software used for the development of the project is Borland C. For both the Master-Slave terminal the particular software is used.

System design



ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Code Efficiency:

Efficiency of coding determines how simple is the coding. Thus it is an important feature that coding must be less complex. Proper use of loops and control statements helps to increase the efficiency of coding. All these are implemented in the project. Besides this there are also other ways to improve efficiency of coding. Like:

Code efficiency is improved by using appropriate C functions and Key Words. C language is very strong language now a day. We can say C as mid level language, because in can interact directly to the hardware by using defined library functions. We can also build some functions as well. We can directly interrupt the CPU process.

Use of Module file and functions at appropriate places

If we want to get interrupt of CPU process we may use some functions provided by C itself. But it is very slow process to execute. There are several ways to interact the hardware. These are as:

Using standard library functions.

Using ROM-BIOS functions (routines).

Using DOS functions.

Directly programming the hardware.

The programs which use 'standard library functions' to interact with the hardware are no doubt most reliable, but work very slowly.

If we directly program the disk drive controller card, we have to have the knowledge about, the type of the disk drive, in which direction it rotates, at what speed it rotates, etc. Moreover, there is no guarantee that the program would work properly.

So the two approaches 'directly programming the hardware' and 'using standard library functions' are two extremes. The golden mean is to either use 'ROM-BIOS functions' or the 'DOS functions'.

The most important reason why we should use ROM-BIOS and DOS functions is: where we want or we don't, these functions occupy some space in memory. This is because they are always present in memory wherever the computer is on.

Testing

Testing comes after the implementation of a design through use of a code. The main objective of testing is to find out whether the software developed is able to meet with the user requirements as described in the analysis and design phase. Testing also provides the programmers confidence to generate an error free coding.

The various testing strategies used for testing the project are as follows:

Hardware testing / System testing: the hardware support required for the development of the project was tested to see that all the components are free from any defects.

Module testing / Unit testing: each module is tested individually. The tests were conducted to check that all the functions used are working according to their intention, use of loops are functioning properly, data transfer from one terminal to another terminal is working properly, source data file and destination data file are being updated properly etc. most of the syntactical errors are removed under this testing.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Application testing / Integrated testing: After unit or modular testing all the units / modules are combined to form the overall application and was tested. Both logical as syntactical errors were completely removed to produce error – free results. The overall application was tested with junk yet sensible data to check whether the project executes properly meeting its objectives.

TESTING REPORTS:

The overall reports of the various tests conducted with the application are given below:

Sending command to Slave....

:, 3, 4, 200,1(alt+(255-207+1))(LRC code)

Its for me!

Proper transmission.....

Received Data:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103,104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148,149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193,194, 195, 196, 197, 198, 199,

Its for me!

Properly transmitted.....

Function code = 3

Additional Information = 200

Send Data...

3, 4, 200, 1

********** Slave device - response **********

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103,104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148,149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193,194, 195, 196, 197, 198, 199,

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Implementation and Maintenance

Implementation

The term implementation has different meanings, ranging from the conversion of a basic application to a complete replacement of a computer system. The procedure, however, is virtually the same. Implementation is thus the process of converting a new or revised system design into an operational one.

- For the implementation of the software we have to install the software (Borland C).
- We have to connect the two computers by a cable through the serial port.
- We have to sort the two serial port such that read buffer of a terminal must be sort to the other's write buffer and write buffer of the terminal to the read buffer of that terminal.
- We have to check the continuity of the cable carefully.
- We may directly solder the two end of the cable to the serial port or by other loose serial port of female type.

Maintenance

Maintenance is an important feature during the running of the software
Maintenance can be divided into:

- Maintenance starts when we install the software we have tested the application by giving test data but after installation real data are install. During entry if some problem arises then remake the module or debug the error.
- If by adding certain modules the system becomes efficient then we have to make it.
- Recovery of data is one of the major portions of maintenance.
- Whether the cables are working properly we have to check
- While processing by the real data, data failure may occur. We have to check whether it is due to length of the cable or not. Because in this case for a long distance data may not be properly transmitted.

Future Scope of The Project

In the ASCII mode of Modbus protocol the time-out feature and diagnostic feature of the protocol has not been implemented. It can be implemented further. The RTU mode of the Modbus protocol can also be implemented further. Because the RTU has the several advantage over the ASCII mode. The most important advantage of the RTU over the ASCII is the error checking methodology. In the ASCII mode error checking is done by the LRC method but in case of the RTU mode the error checking is done by the CRC method. And over all I want to implement the protocol for the scientific data communication.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

References:

[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.

[8] Ubiquity, Volume 2008 Issue January | BY **SAPTARSHI NASKAR** , **KRISHNENDU BASULI** , **SAMAR SEN SARMA** **SERIAL PORT DATA COMMUNICATION USING MODBUS PROTOCOL**