Implementation of Serial Communication between PC and DSP Processor Using Modbus Protocol

Nagendra Sah[1], Gaurav Khurana[2]
[1](Sr. Asstt. Professor, Electronics and Electrical Communication Department, PEC University of Technology, Chandigarh
[2](M.E., Electronics and Electrical Communication Department, PEC University of Technology, Chandigarh

ABSTRACT
This paper described the principle, application and implementation of serial communication between PC and a Digital Signal Processor (DSP). The TMS320F28031 which is a type of DSPs made by Texas Instruments (TI) is used in this implementation. This DSP processor has serial communication interface (SCI) module for serial communication. The SCI is a two‒wire asynchronous serial port, commonly known as a UART (Universal Asynchronous Receiver/Transmitter). The standard of Modbus protocol is implemented for this serial communication. The Modbus protocol provides an industrial standard method that Modbus devices use for parsing messages. PC (Personal Computer) can read/write one or more registers of DSP processor using Modbus communication. This paper highlights the basics of Modbus protocol and also explains the software detail of Modbus implementation.

***Keywords* -** DSP processor, Modbus protocol, UART, serial communication

I. INTRODUCTION

Although all the real time signals are analogue in nature but due to the huge advancement in digital signal processing, all the signals are processed in digital form. As we all know, Digital signal processing has number of advantages over analogue signal processing. To process the real time signal in digital domain, first of all we have to convert that analogue signal into digital signal using ADC (Analog to Digital Converter) and after signal processing; we have to convert digital output back into analogue form by using DAC (Digital to Analog Converter).

Due to huge advancement in the field of VLSI (Very Large Scale Integration) in the past few years, there are number of digital Integrated Chips (IC's) are available in the market for performing the above task more efficiently and accurately. These digital IC's are commonly known as digital signal processor (DSP). DSP has been used widely in auto controller, image process, communication, network, home electrical appliances, and so on. Currently, the most widely used product come from Texas instruments (TI) which takes up almost 60 % of the market [1]. TMS320F28031 is a chip which is a product made by TI. Unlike some other chips, it uses an advanced Havard type architecture that maximizes processing power by maintaining two separate memory bus structures, one memory section for storing program and other for storing data. This will increase the program execution speed. This chip has several integrated peripherals like ADC, SCI, Timer and PWM (Pulse Width Modulator) etc. [2].

The Modbus is one of the most common serial communication protocol used in industrial application for process control. Using Modbus protocol a number of controllers and intelligent devices and communication with each other over any network. Actually Modbus protocol defines a messaging structure which is universally accepted and used. In this paper, we have established a serial communication between PC and DSP processor using Modbus protocol. The master sends a command in hexadecimal and slave responds with its response in hexadecimal too. In order to display the result, software called Modbus tester is used. Same software is also used to configure the communication parameters like communication mode, baud rate, start bits, stop bits and parity bits etc.

II. HARDWARE STRUCTURE

A. SCI

TMS320F28031 DSP processor has an on chip serial communication interface (SCI) as one of its peripheral in its core. SCI is a two‒wire asynchronous serial port and it supports digital communications between the CPU (Central Processing Unit) and other asynchronous peripherals that use the standard non-return-zero (NRZ) format. The SCI receiver and transmitter are double-buffered and each has a 4-level deep FIFO (First in First Out) for reducing servicing overhead. Both have their own separate enable and interrupt bits and both can be operated independently for half-duplex communication, or simultaneously for full duplex communication. To ensure data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to over 65000 different speeds through a 16-bit baud select register [3].

SCI module has two external pins for serial communication that is SCITXD (SCI transmit output

pin) and SCIRXD (SCI receive input pin). Serial Transmission and reception operations can be accomplished through interrupt driven or polled algorithms. Here interrupt driven technique is used for communication in full duplex mode.
Major elements of SCI module is shown in figure 1 and it includes:

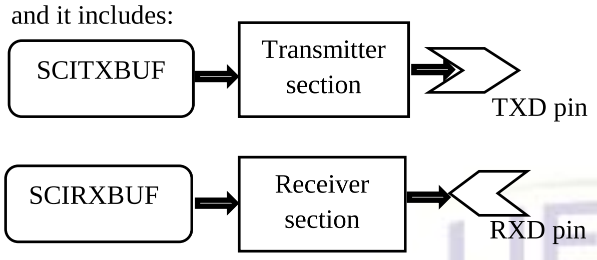


Fig. 1 Internal Device Structure

A transmitter (TX) and its major registers (upper half of Figure 1) are:
SCITXBUF register (SCI transmitter data buffer register) - it Contains data (loaded by the processor) to be transmitted to the remote PC.
TXSHF register (SCI transmitter shift register) - It accepts data from register SCITXBUF and shifts data onto the SCITXD pin, one bit at a time.
A receiver (RX) and its major registers (lower half of Figure 1) include:
RXSHF register (SCI receiver shift register) - It shifts in the data from SCIRXD pin, one bit at a time.
SCIRXBUF register (SCI receiver data buffer register) - It contains data to be read by the DSP processor. Data from a remote PC is loaded into register RXSHF and then into registers SCIRXBUF.

III. MODBUS PROTOCOL

The Modbus is a serial communications protocol published by Modicon in 1979. Modbus protocol defines a standard message structure with universal recognition and usage regardless of the type of networks over which any two devices communicate. It is a master slave communication protocol. It describes the process a master uses to request an access to slave, and how the slave will respond to these requests, and how errors will be detected and reported. Master can initiate transactions (called ‘queries’) and slave respond by supplying the requested data to the master, or by taking the action requested in the query. The master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a message (called a ‘response’) to queries that are addressed to them individually. Responses are not returned to broadcast queries from the master. Figure 2 shows the query response cycle of Modbus communication.

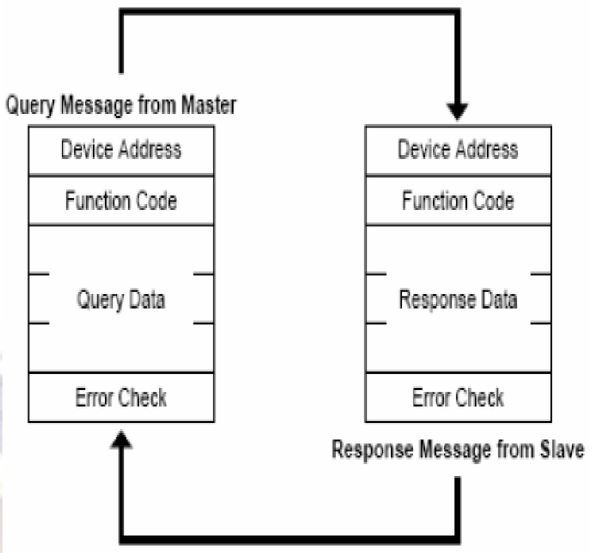


Fig. 2 Modbus master-slave query-response cycle

As shown in figure 2, master’s query consists of slave 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 contains fields confirming the action taken, any data to be returned, and an error–checking field. Slave confirms the action taken by sending the echo of function code sent by the master.
Table I shows the Application Data Unit (ADU) and Protocol Data Unit (PDU) of Modbus protocol. PDU is consisting of function field (1 byte) and data field (variable bytes) and ADU is consisting of address field (1 byte), PDU and error checking field (2 bytes) [4].

Table I Modbus data format

	Protocol data unit (PDU)		
Application Data unit(ADU)			
1 byte	1 byte	Variable	2 bytes
Address field	Function field	Data field	Error checking field

If the slave makes a normal response, the function code in the response is an echo of the function code in the query. 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 by modifying the function code (set the MSB (Most significant Bit) of function code) to indicate that the response is an error response, and the data bytes contain a code that describes the error [5]. Some of the commonly used function codes are shown in table II.

Table II List of commonly used function codes in Modbus protocol

Code	Name
01	Read single coil status
02	Read input status
03	Read multiple holding registers
04	Read multiple input registers
05	Write single coil
07	Read Exception Status
15	Write multiple coils
16	Write multiple registers
23	Read/Write multiple registers

Modbus protocol can be established in two kinds of transmission mode: ASCII (American Standard Code for Information Interchange) mode or RTU (Remote Terminal Unit) mode. In ASCII mode, each 8–bit byte in a message is sent as two ASCII characters. In RTU mode, each 8–bit byte in a message contains two 4–bit hexadecimal characters. The main advantage of RTU mode is that its greater character density allows better data throughput than ASCII for the same baud rate. Modbus protocol has the parity check, besides, the ASCII mode uses the LRC (Longitudinal Redundancy Check) and the RTU mode uses 16 CRC (Cyclical Redundancy Check).

Table III is a comparison between ASCII mode and RTU mode.

Table III Comparison of ASCII and RTU mode

Mode	Beginning marks	Ending marks	Check	Transmission efficiency	Program Processing
ASCII	: (colon)	CR, LF	LRC	low	Direct, easy to debugging
RTU	Non	Non	CRC	High	Indirect, slightly complex

According to Table III, the data transmission rate of ASCII mode is a little lower than RTU mode. So, when need to send large data, user always uses RTU mode. The standard Modbus protocol is to use a RS-232C compatible serial interface, which defines the port pin, cable, digital signal transmission baud rate, parity.

IV. IMPLEMENTATION OF COMMUNICATION

A. *Configuration Setting*

In order to implement the Modbus protocol communication between PC and DSP processor, first of all we should configure both PC and DSP processor for the same communication mode and same baud rate that is 9600 bps (bits per second) [6]. As explained earlier, there are two modes of serial communication in Modbus protocol that is ASCII mode and RTU mode. In this implementation we have used the later one. Configuration used in this implementation is shown in table IV.

Table IV PC and DSP processor communication Configuration

Communication Mode	Remote Terminal Unit (RTU)
Baud Rate	9600 bps
Data bit	8 bits
Stop bit	1 bit
Parity bit	None

Baud rate is nothing but data transfer rate which must be same at both the terminals (PC and DSP Processor). PC baud rate is configured by using Modbus communication interface software named Modbus tester and DSP processor's baud rate is configured by using its control registers. Following formulas are in this process:

Baud rate = Sysclk / (BRR+1) × 8 (1)

BRR = Sysclk / (Baud rate × 8) □ 1 (2)

In "Eq. (1)", the Sysclk stands for system clock frequency and the BRR in "Eq. (2)" is the value of SCIHBAUD and SCILBAUD registers that you should configure.

After setting the proper baud rate, the serial communication is established by using two wire communication method. Modbus protocol is master/slave protocol and communication can be initiated by master only and here PC is working as master and DSP Processor is working as slave. So PC will initiate this communication and sends a command to read DSP Processor's 7 input registers. Each register is of 2 bytes. But data field in Modbus protocol is of 8 bits so each register is represented by two data bytes. First byte represents higher byte and second byte represents lower byte. Hence 7 register is equal to 14 bytes. Slave address is 8 bit long and it is settable. In this project, we have used only one slave and its address is 01.

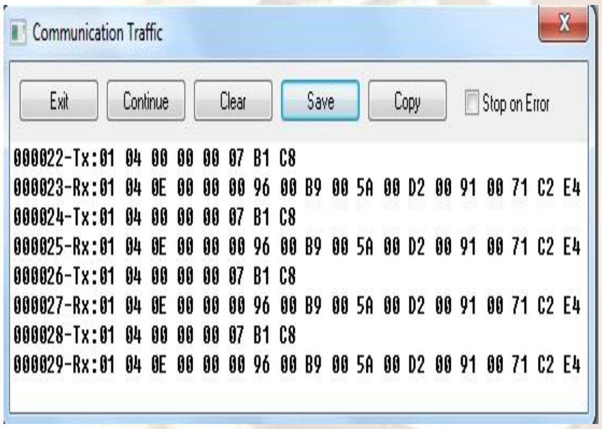
B. *Software Flow*

This communication is implemented in TI tool code compose studio. Code Composer Studio™ (CCStudio) is an integrated development environment (IDE) for Texas Instruments (TI) embedded processor families. CCStudio comprises a suite of tools used to develop and debug embedded applications. Timer 0 interrupt service routine of TMS320F28031 is used in this implementation. Timer 0 is configured to generate an interrupt after every 20 microsecond. In timer 0 ISR (interrupt service routine), we check the flag bit showing the reception of data byte. If this flag bit is set, it shows that byte has been received. Then we process the received the data to extract the useful information form this data [7].

First byte is slave ID (slave address). So first of all, we compare the received byte with the address of that slave. It received byte matches with the slave ID. It means that next whole data frame is addressed to this slave and we will process the whole data frame otherwise we will ignore the whole data frame. Then we go for CRC check. It will tell about the frame validity. If frame is invalid, program will generate a response showing error in received frame using exception code. Next step is extracting the information from function code. List of commonly used function code is shown in table II. Then create the slave response according the function code.

C. *Read Input Registers of DSP Processor*

Here PC is master and will initiate the communication so it is working as transmitter and DSP Processor's response is received by the PC [8]. The communication traffic is shown in figure 3.



```
000022-Tx:01 04 00 00 00 07 B1 C8
000023-Rx:01 04 0E 00 00 00 96 00 B9 00 5A 00 D2 00 91 00 71 C2 E4
000024-Tx:01 04 00 00 00 07 B1 C8
000025-Rx:01 04 0E 00 00 00 96 00 B9 00 5A 00 D2 00 91 00 71 C2 E4
000026-Tx:01 04 00 00 00 07 B1 C8
000027-Rx:01 04 0E 00 00 00 96 00 B9 00 5A 00 D2 00 91 00 71 C2 E4
000028-Tx:01 04 00 00 00 07 B1 C8
000029-Rx:01 04 0E 00 00 00 96 00 B9 00 5A 00 D2 00 91 00 71 C2 E4
```

Fig. 3 Modbus communication traffic

As shown in figure 3, TX (Transmitter) represents the command sent by PC to DSP processor and RX (Receiver) is the response sent by DSP processor to PC. Analysis of TX data is given in table V.

Table V Data sent from PC to DSP Processor

Code (Hex)	Meaning
01	Slave Address.
04	Function code(read multiple input register)
00	Starting address of input registers(higher)
00	Starting address of input register (lower)
00	Number of registers (higher)
07	Number of registers (lower)
B1	CRC (higher)
C8	CRC (lower)

From table V it is clear that, PC wants to read multiple input registers of slave whose address is 01. Starting address (16 bits) of these registers is 0000h is also given by the master in its command and number register is 0007h. Last two bytes are CRC check bits for detecting the error in the transmitted code.

RX is the response sent by DSP Processor to PC and analysis of Rx is given in table VI:

Table VIData sent from DSP Processor to PC

Code (Hex)	Meaning
01	Slave Address.
04	Function code (echo of function code sent by master, as no error is detected by slave in Tx code. If slave detect some error in Tx code then it exception code which is also echo of original function code with its MSB is equal to logic 1)
0E	14 bytes(2*7 registers) in data field
00	Content of first register (higher)
00	Content of first register (lower)
00	Content of second register (higher)
96	Content of second register (lower)
00	Content of third register (higher)
B9	Content of third register (lower)
00	Content of fourth register (higher)
5A	Content of fourth register (lower)
00	Content of fifth register (higher)
D2	Content of fifth register (lower)
00	Content of sixth register (higher)
91	Content of sixth register (lower)
00	Content of seventh register (higher)
71	Content of seventh register (lower)
C2	CRC (higher)
E4	CRC (lower)

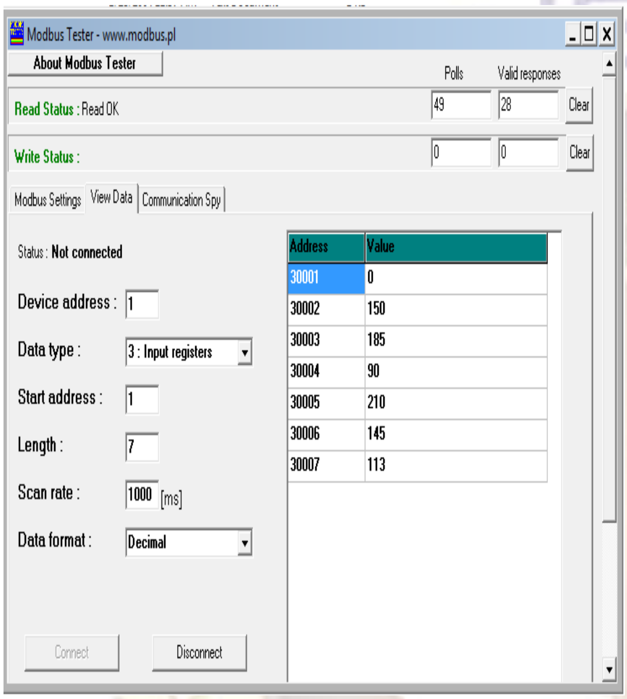
It is clear from table VI that, each 16 bit register is represented by two data fields each of 8 bit long. Hence content each register is shown in hexadecimal and decimal in table VII.

Table VII 16 bit register content in hexadecimal and decimal format

Register number	Content in Hex	Content in Decimal
01	0000h	00
02	0096h	150
03	00B9h	185
04	005Ah	90
05	00D2h	210
06	0091h	145
07	0071h	113

V. RESULTS AND DISCUSSION

Results are shown in figure 4. It is clear from figure 4 that master (PC) send a command to read 7 input registers (14 bytes) of DSP processor and then DSP processor create a response message consist of 19 bytes (1 byte slave ID, 1 byte function code, 1 byte to show number of bytes in data unit (that is 14 in this example), 14 bytes of data and 2 bytes of CRC). Modbus tester software check integrity of slave response if response is found to be valid then it extract 14 byte data unit from 19 byte slave response and display the result in decimal as shown in figure 4.



Modbus Tester - www.modbus.pl

About Modbus Tester

	Polls	Valid responses	
Read Status : Read OK	49	28	Clear
Write Status :	0	0	Clear

Modbus Settings | View Data | Communication Spy

Status : Not connected

Device address : 1

Data type : 3 : Input registers

Start address : 1

Length : 7

Scan rate : 1000 [ms]

Data format : Decimal

Connect | Disconnect

Address	Value
30001	0
30002	150
30003	185
30004	90
30005	210
30006	145
30007	113

Fig. 4 results display in Modbus tester

VI. CONCLUSION

Modbus protocol is implemented and serial communication between PC and DSP Processor is established and results are displayed using Modbus tester software. Thus, we can easily interface a number of digital IC's like DSP processors with our PC using Modbus protocol. So this communication technique can be one of well choice in industrial control applications.

REFERENCES

[1] Zhang Zhi-Qiang, and Zhang Yu-lin, "Realization of Communication Between DSP and PC Based on Modbus Protocol," IEEE International Conference on Multimedia Information Networking and Security, 2009, pp. 258-261.

[2] Peng D.G, Zhang H., Yang Li and Li H. "Design and Realization of Modbus Protocol Based on Embedded Linux System," The 2008 International Conference on Embedded Software and Systems Symposia. July 29-31, 2008, Chengdu, Sichuan, China pp. 275-280.

[3] TI, TMS 320F/28031 DSP Controllers Peripheral Library and Specific Devices, Dallas: Texas Instrument, 2007, pp.56-72.

[4] Zhi-Hua Chen, Min Shi, Qing-Ming Yi,"A Method for DSP asynchronous Serial port Expansion Based on TL16C752B," IEEE, 2011, pp. 844-847

[5] Yue Y., Zhang C.G., Yuang A.J., "Design and implementation of Embedded Man-Machine Interface Based on Modbus". Industrial Control Computer, 2006, 19(1) pp. 8-10.

[6] TI, TMS 320F/28031 DSP Controllers Peripheral Library and Specific Devices, Dallas: Texas Instrument, 2007, pp.96-102.

[7] Song J, QU J P, "Realization of Communication Between PCC and Touch Screen Based on Modbus Protocol," Electric Machine Integration, vol. II, Nov. 2007, pp. 28-73.

[8] TI, TMS 320F/28031 DSP Controllers Peripheral Library and Specific Devices. Dallas: Texas Instrument, 2007, pp 176-185.