Design of Serial Communication in Special Circumstances

Peiyu Zhao
School of Electromechanical and Architectural Engineering, Jianghan University, Wuhan, 430056, China
zpy@21cn.com

Abstract - In recent decades, MCUs employing serial communication have been widely used in instrumentation industry. Many MCU systems need to disable interrupt when it operating some special function with its peripherals. On the other hand, they won't want to lose any communication data with host computer or other MCUs. A method namely "Starting by Interrupt, Receiving in Query mode" (SIRQ) is proposed to make the MCU system fulfil the stringent timing requirements by its peripherals while remaining stable communication with the host computers. The implementation method, flowchart and part of source code are presented. This method has been verified and proved to be simple and robust in some real products.

Index Terms - MCU, Serial Communication, Interrupt

1. Introduction

RS232 Serial ports are widely used in distributed measurement and control systems, instrumentations and computer peripherals to transfer measurement data and calibration parameters. In order to ensure real-time communication and high communication efficient, interruption based UART transceiver are generally used. While with the increasing of MCU peripherals, such as E^{2}PROMs, AD/DA converters and so on, different serial bus interfaces such as SPI, I^{2}C or 1-wire bus could be found in many MCU systems. Generally, these serial bus interfaces need stringent timing to exchange data with the MCUs and should not be interrupted. Otherwise data error or even system halt may occur.

Typically, MCU in the communication is the slave device, and asynchronous serial communication may be initiated by the master device in a random and the master device does not know whether the recipient can receive or not. When the MCU is busy in its own peripherals operating, the interrupt system of the MCU might be disabled. If a UART communication is initiated during this critical period, communication data lost may occur [1]. A communication protocol may be employed to solve this problem [2-4], but this will cause the communication program quite complicated. Hence, to maintain a stable but simple communication between the master and the slave devices using UART is a very important research topic.

The author encountered this problem in developing a high-precision serial AD acquisition board using TI's ADS1234, a 24-bit resolution AD converter, employing the SPI serial interface to communicate with the MCU. Calibration program on host PC is used and connected through the RS232 serial port to the MCU of the AD acquisition board. Interruption based communication program is used in the MCU. During testing, sometimes the calibration procedure failed if the host initiating communication during the process of operating the AD converter using SPI serial bus, when the interrupt system disabled. To solve this problem, a method namely "Starting by Interrupt, Receiving in Query mode" (Hereinafter referred to as SIRQ) is developed and proved to be simple but effective.

2. Principles

As shown in Fig.1, there are 2 independent transmitting and receiving buffers (SBUF) in a MCS-51 UART, sharing the same logic address. When the receiver RXD receives 8-bit serial data from the shift register, it stores the data into the receive buffer SBUF, and generate an interrupt signal RI, informing the CPU to read the data from the receive buffer. It should be noted that the receiver will always receive data from the serial bus and stores the last byte received into the receive SBUF [5, 6], regardless the MCU serial interrupt enabled or disabled. The only different is that, if the interrupt enabled, the MCU will respond immediately and read the received data, while if the interrupt disabled, it will respond after the interrupt re-enabled.

The SIRQ method is based on the following steps: If the host PC want to send data to MCU, it initiate a communication with a single byte header, and waiting a reasonable period of time to ensure the MCU catch the receiver interrupt, and then send the subsequent data. On the other hand, the MCU using an interrupt service routine to check the header and receive the following data from the serial port in inquiry mode. These two steps ensure the MCU will not lose any data from the host computer.

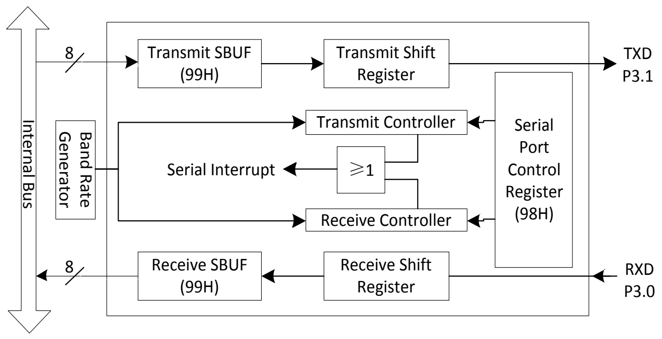


Fig. 1. The UART Diagram of MCS-51 MCU

General monitoring flowchart of the MCU is shown in Fig.2. After system initialization, the MCU disables interrupt, then does sampling or other uninterruptable business, and then enables the interrupt, does calculating, analysis, controlling, displaying and other non-crictial things. After that, the program will be back to “Disable Interrupt” as shown in the flowchart, and then circulates all the processes. The cycle time is T, which is basically unchanged or changed little.

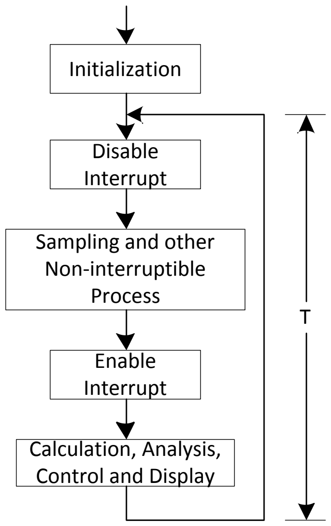

Initialization
Disable Interrupt
Sampling and other Non-interruptible Process
Enable Interrupt
Calculation, Analysis, Control and Display
T

Fig.2 Flowchart of MCU monitoring procedure

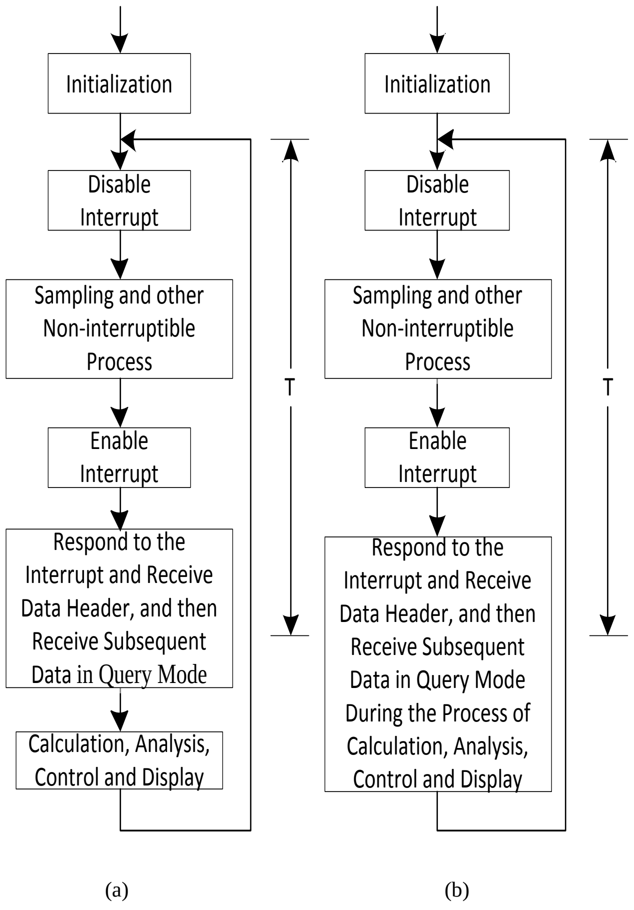

Initialization
Disable Interrupt
Sampling and other Non-interruptible Process
Enable Interrupt
Respond to the Interrupt and Receive Data Header, and then Receive Subsequent Data in Query Mode
Calculation, Analysis, Control and Display
T
(a)
Initialization
Disable Interrupt
Sampling and other Non-interruptible Process
Enable Interrupt
Respond to the Interrupt and Receive Data Header, and then Receive Subsequent Data in Query Mode During the Process of Calculation, Analysis, Control and Display
T
(b)

Fig.3 Flowcharts of the host PC initiate communication at different point time

If the host computer sends data continuously during the interruption disabled, it will cause receiving error. But if the host sends a 1 byte header, and waits a period of time more than T, and then send subsequent data continuously, there will be two cases shown in Fig.3. The first case is that, the first byte of data header received by the MCU during the time when the interruption disabled as shown in Fig.3 (a). In this case, the MCU will respond to the interrupt as soon as the interruption re-enabled, and wait the host to transmit all the remaining data, and then do the non-critical things. The other case is the one byte data header received by the MCU during the time when the interruption enabled. In this case, the MCU will respond to the interrupt immediately, and then waiting the host to transmit all the remaining data, as shown in Fig.3 (b). According to Fig.3, by employing this SIRQ method, the receiver will not lose any transmitting request initiated by the host. It is quite simple but very robust.

3. The Flowchart and Code of Interrupt Service Subroutine

The flowchart and the C51 code of interrupt service subroutine used in the SIRQ method for serial communication are shown in Fig.4 and Fig.5, respectively. As can be seen, the interrupt service subroutine is also very simple.

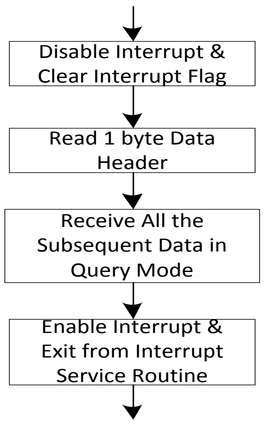
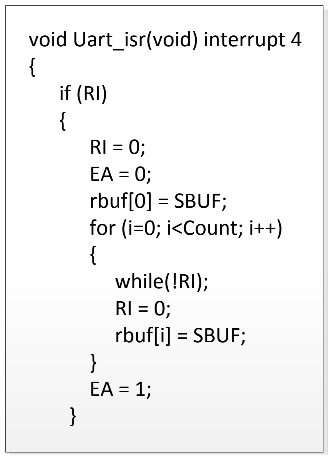

Disable Interrupt & Clear Interrupt Flag
Read 1 byte Data Header
Receive All the Subsequent Data in Query Mode
Enable Interrupt & Exit from Interrupt Service Routine

Fig.4 The flowchart of interrupt service subroutine



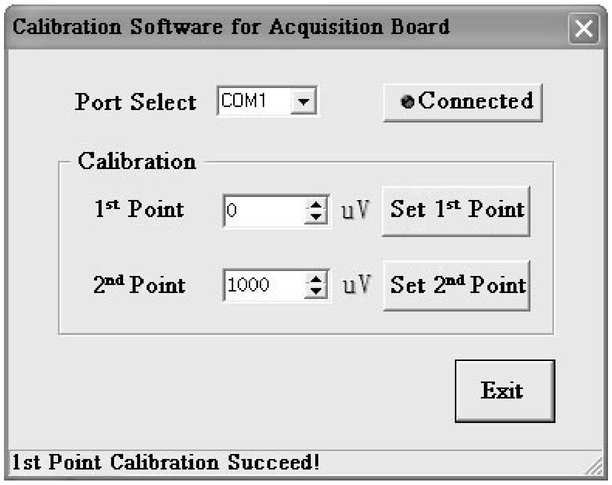
```
void Uart_isr(void) interrupt 4
{
    if (RI)
    {
        RI = 0;
        EA = 0;
        rbuf[0] = SBUF;
        for (i=0; i<Count; i++)
        {
            while(!RI);
            RI = 0;
            rbuf[i] = SBUF;
        }
        EA = 1;
    }
```

Fig.5 The C51 code of the interrupt service routine

4. Verification

The proposed SIRQ method for serial communication is verified with the high-precision serial AD acquisition board discussed in section 1. Simplified 3-wire RS232 is used to connect the host PC and the acquisition board. The calibration program employing MSCOMM control [7, 8] is shown in Fig.6. Filling the voltage values to the first point and the second point according to the actual given voltages to the AD input of the acquisition board, and click the "first point" or "second point" in the program window to calibrate the corresponding point. For each point, the host PC transmits a one byte header and waits a cycle time T, and then transmits the subsequent calibration data.

The receiving program of the MCU using SIRQ method to get the calibration data, it receives the one byte header by interrupt, and then receive the remaining data in query mode in interrupt service routine as shown in Fig.5. It is proved that the method works very well in calibrating the said high-precision serial AD acquisition board.



0

1000

1st Point Calibration Succeed!

Fig.6 The calibration program at host computer

5. Conclusion

"Starting by interrupt, receiving in query mode" (SIRQ) method is proposed for MCU system with UART serial port communicating with host computer or other MCU systems. It resolves the communication unstable problem for those systems with uninterruptable stringent timing peripherals. This method has been adopted in some real products and proved to be practically valuable.

References

[1] Ma Yuchun, Song Hantao, "Serial communication protocol and its application", Application Research of Computers. (4), 2004, pp. 228-232.
[2] Zhao Jiong, Xiong Xiaolei, Zhou Qicai, "Analysis and Study of Serial Data Transfer Protocol", Computer Engineering 30 (9), 2004, pp. 105-108.
[3] Zhang Lei, Liang Jianwu, Chen Ying, "Research and Implementation of Serial Communication Protocol", Modern Computers (9), 2006, pp. 57-59.
[4] Sun Guang, Zhao Zhimin, "High efficiency serial communication protocol design", Instrumentation Technology, (03), 2002, pp. 12-13.
[5] Mei Lifeng, Wang Yanqiu, Wang Yuduo, Zhang Jun, SCM principle and interface technology: Tsinghua University Press, 2007, pp. 121-140.
[6] STC12C5628AD Data Sheet, Nantong Guoxin Microelectronics Co., Ltd., 2011, pp. 205-244.
[7] Zhou Xianhui, Mao Cuili, Wang Changhe, "Development of Temperature Measurement and Control System Based on Serial Communication between STC12C5A and Computer", Journal of Nanyang Institute of Technology (4), 2012, pp. 52-56.
[8] Zhou Yonggang "Serial Communication Between PC and MCU Based on MSComm", Instrumentation Technology, (5), 2013, pp. 12-13.