信息与软件工程学院上机实验报告

(第 4 次)

一、实验名称

13.1.4 实验四 串口双机通讯

二、实验目的及要求

1. 实验目的
   通过本次实验掌握 Keil uVision3 和 Proteus 开发环境以及它们联机调试单片机汇编语言和C语言源程序方式方法, 熟悉单片机汇编语言指令, 掌握单片机串行口通信方法。
2. 实验要求
   在本次实验前, 在自己的电脑安装 Keil uVision3 和 Proteus7.0 以上版本的软件, 使用 Proteus 软件画参考硬件原理图 (如图13-10)。

三、实验环境

- **仿真实验**：装有 Keil uVision3 以及 Proteus7.0 以上版本软件的笔记本或台式PC机。
- **元件清单**：2个STC89C52、2个74HC245总线收发器、4个7SEG-BCD数码管, 一个SW-SPDT-MOM 双向开关, 2个BUTTON按钮开关, 2个示波器, 一个+5V稳压电源。

四、实验内容

按硬件原理图连线, 编写软件实现:

假定有A、B两机, 以方式1进行串行口通信, 其中A机发送信息, B机接收信息, 双方的晶振频率 $f_osc = 11.0592MHz$, 通信波特率为9600。

通信协议：通信开始时, A机首先发送一个启动信号AA, B机接收到后发送一个应答信号 BB 表示同意接收。

A机收到BB后, 就可以发送数据了。假定该实验发送的数据长度是32个字节, 发送数据缓冲区为片内数据区, 首地址为30H, 该数据缓冲区为bufs, 数据块发送完后要立即发送校验和, 进行发送数据准确性校验。

B机将接收到的数据存储到数据缓冲区bufR, 接收数据存放在片内50H (注：指导书原文), 接收数据块后, 再接收机发来的校验和, 并将其与B机求出的校验和进行比较。若两者相等, 说明接收正确, B机回答00H; 若两者不等, 说明不正确, B机回答 FFH, 请求重发。A机接收到OOH, 停止发送, 若接收到FFH信号重新发送。

设计要求:

1. A、B两机点对点的通信, 该程序可以在双方机中运行, 不同的是在程序运行之前, 要判别 P1.0口的输入, 若P1.0=1, 表示该机是发送方; 若P1.0=0, 表示该机是接收方。
2. 双方的P1.7为校验和控制, 按钮抬起时(P1.7=1), 发送方送出正确校验和, 发送1次数据! 按钮按下时(P1.7=0), 发送方送出错误校验和, 则重发!
3. 发送方将发送的数据送数码管显示, 接收方将接收到数据也送数码管显示。

五、实验设计

1、实验步骤

实验流程图

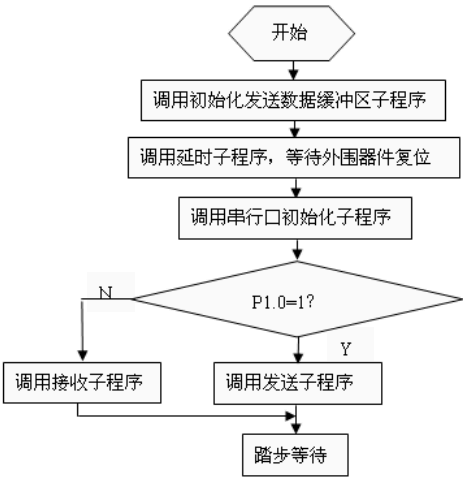


图 13-11 点对点通讯主程序框图

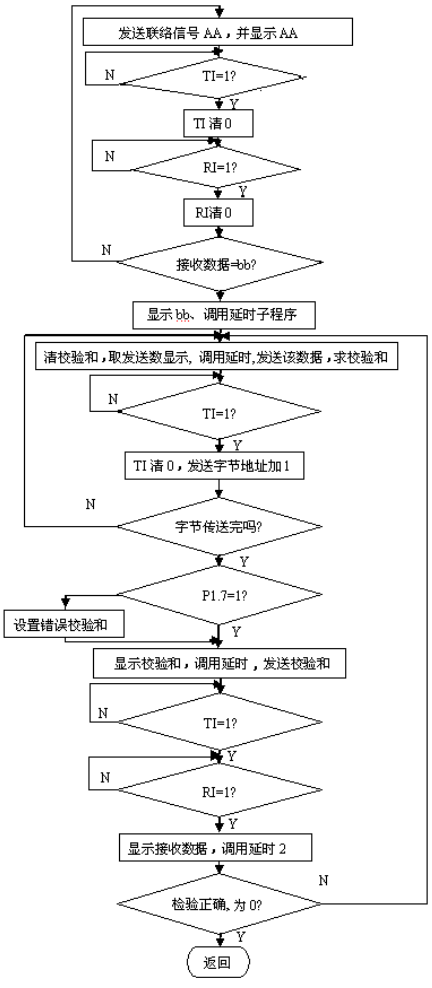


图 13-12 点对点通讯发送子程序框图

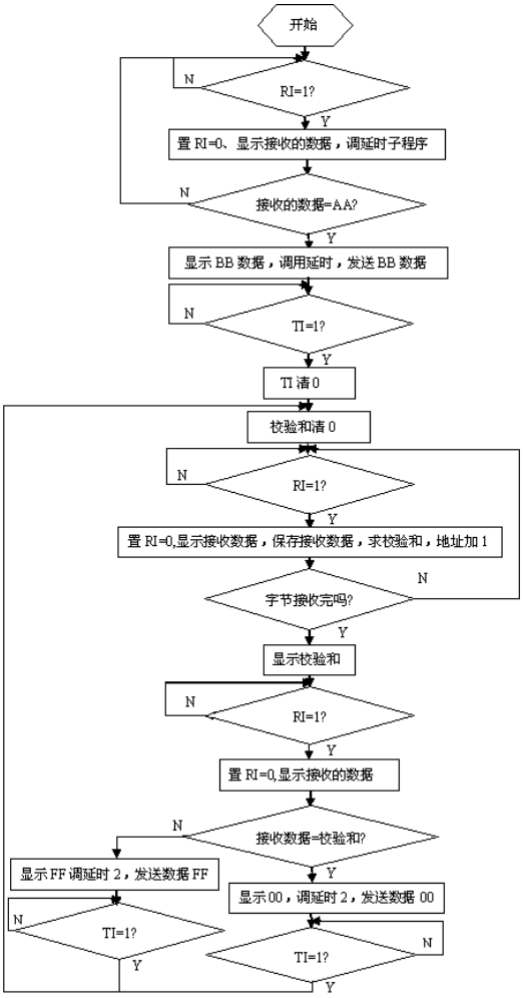


图 13-13 点对点通讯接收程序框图

设计思想

- 本实验构建一个带握手（AAH/BBH）和校验和机制的半双工异步串行通信。
- **初始化**：设置串口为方式1（SCON=50H），使用定时器T1作波特率发生器。
- **波特率设置**：使用11.0592MHz晶振，T1设为方式2（8位自动重装）。MOV TMOD, #20H。MOV TH1, #0FDH (9600波特率, SMOD=0)。SETB TR1 启动T1。
- **角色分配**：程序启动时检查P1.0的状态，JB P1.0, SEND_SUB（跳转到发送子程序）或 JNB P1.0, REC_SUB（跳转到接收子程序）。
- **通信协议**：严格按照实验内容的协议流程，使用TI（发送中断标志）和RI（接收中断标志）来查询发送/接收状态（查询方式），并通过SBUF寄存器收发数据。
- **校验和**：发送方在发送32字节数据时，使用一个寄存器（如B）将所有数据累加，得到校验和。接收方在接收时也同样累加计算。
- **错误模拟**：发送方在发送校验和之前，检查P1.7。若P1.7=0，则对计算出的校验和加1（或取反），以发送一个错误的校验和，用于测试重发机制。
- **状态显示**：收发的数据和关键状态（AA, BB, 校验和等）通过调用数码管显示子程序显示在硬件上。

实现步骤

1. **主程序 (MAIN)**：

- a. 初始化发送数据缓冲区bufs (片内30H-4FH)。
- b. 调用延时。
- c. 调用串行口初始化子程序：MOV SCON, #50H (方式1, 允许接收), MOV TMOD, #20H (T1方式2), MOV TH1, #0FDH (9600波特率), SETB TR1。
- d. 检查P1.0：若P1.0=1，LJMP SEND_SUB；若P1.0=0，LJMP REC_SUB。
- e. HERE: SJMP HERE。

2. **发送子程序 (SEND_SUB)**：(参照图13-12)
   - a. 发送联络信号 MOV SBUF, #0AAH，等待 JNB TI, $，CLR TI。
   - b. 等待接收应答 JNB RI, $，CLR RI，MOV A, SBUF。
   - c. 检查是否为BBH，CJNE A, #0BBH, ...，若不是则重新发送AAH。
   - d. **发送数据循环**：设置R0指向30H，R2为32。清空校验和寄存器B。
   - e. 循环：MOV A, @R0，ADD A, B，MOV B, A (累加校验和)，MOV SBUF, @R0，等待TI，CLR TI，INC R0，DJNZ R2, ...。
   - f. **发送校验和**：检查P1.7，若P1.7=0，则 INC B (制造错误校验和)。MOV SBUF, B，等待TI，CLR TI。
   - g. **等待应答**：等待RI，CLR RI，MOV A, SBUF。
   - h. 检查应答：CJNE A, #00H, ...，若为FFH（错误），则 LJMP SEND_SUB (或跳转到数据发送循环d)。
   - i. 若为00H（正确），返回主程序 LJMP HERE。
3. **接收子程序 (REC_SUB)**：(参照图13-13)
   - a. 等待接收AAH：JNB RI, $，CLR RI，MOV A, SBUF，CJNE A, #0AAH, ... (若不是则继续等待)。
   - b. 发送应答BBH：MOV SBUF, #0BBH，等待TI，CLR TI。
   - c. **接收数据循环**：设置R0指向50H，R2为32。清空校验和寄存器B。
   - d. 循环：等待RI，CLR RI，MOV A, SBUF，MOV @R0, A，ADD A, B，MOV B, A (累加校验和)，INC R0，DJNZ R2, ...。
   - e. **接收校验和**：等待RI，CLR RI，MOV A, SBUF。
   - f. **比较校验和**：CJNE A, B, ERROR (与B中自算的校验和比较)。
   - g. **正确**：MOV SBUF, #00H，等待TI，CLR TI。LJMP HERE。
   - h. **错误 (ERROR)**：MOV SBUF, #0FFH，等待TI，CLR TI。LJMP HERE。

2、调试过程及实验结果

调试过程：

1. **硬件搭建**：在 Proteus 中按照图 13-10 搭建电路。
   - 放置两个 STC89C52 (A机, B机)。
   - **交叉连接**：A机 的 TXD (P3.1) 连接到 B机 的 RXD (P3.0)；A机 的 RXD (P3.0) 连接到 B机 的 TXD (P3.1)。
   - **角色分配**：A机 的 P1.0 接高电平 (+5V)，使其成为发送方。B机 的 P1.0 接地 (GND)，使其成为接收方。
   - **错误模拟**：A机 的 P1.7 连接一个 `BUTTON`，按钮另一端接地。
   - **显示**：为两台单片机连接数码管以显示数据。

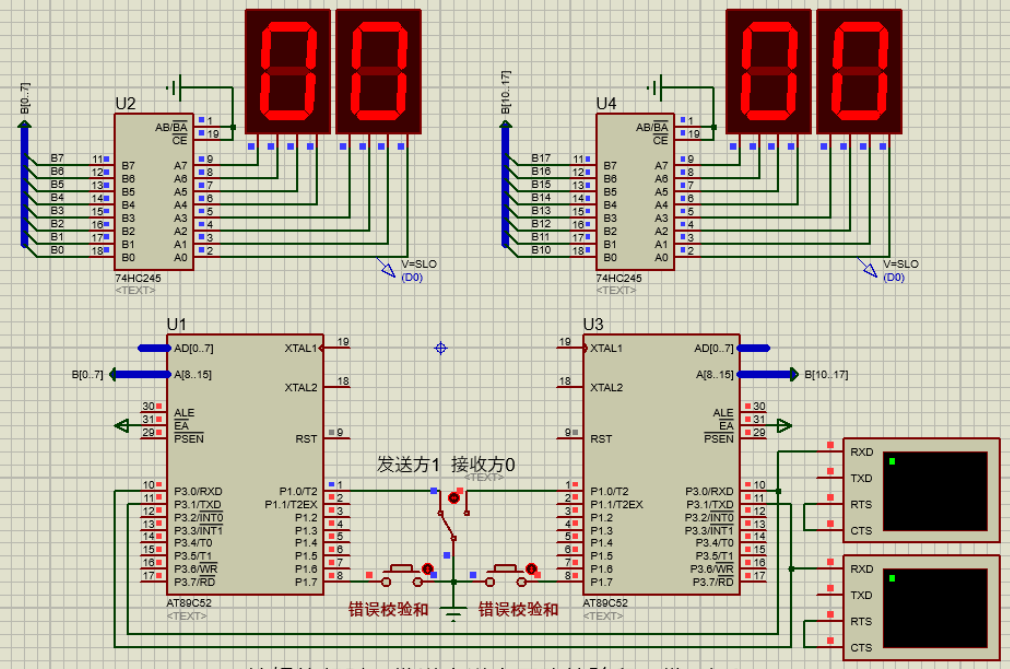
2. **程序加载**：将编译好的 *同一个* .hex 文件加载到 A机 和 B机 中。
3. **波特率设置**：程序中设置 T1 为方式 2 (`TMOD, #20H`)， `TH1` 设为 0FDH。使用 11.0592MHz 晶振，波特率 = $11.0592MHz/(12\times(256-253))/2=9600\ \text{bps}$ (SMOD=0)。
4. **测试一：正常通信 (P1.7 抬起)**
    - 保持 P1.7 按钮不按 (P1.7=1，发送正确校验和)。
    - 运行仿真。
    - **A机 (发送方)**：显示 `AA` ，等待... 显示 `BB`，然后依次显示 32 字节数据，最后显示计算出的校验和，接收到 `00` 后停止。
    - **B机 (接收方)**：显示 `AA` ，然后发送 `BB` (B机显示 `BB`)，然后依次显示接收到的 32 字节数据，最后显示接收到的校验和，比较校验和正确后，发送 `00` (B机显示 `00`)，停止。
5. **测试二：错误重发 (P1.7 按下)**
    - 停止仿真。按住 A机 的 P1.7 按钮 (P1.7=0，发送错误校验和)。
    - 运行仿真。
    - **A机 (发送方)**：显示 `AA`， `BB`，发送 32 字节数据，发送错误校验和 。等待... 接收到 `FF` (A机显示 `FF`)，程序跳转回 `SEND2` ，重新开始发送 32 字节数据。
    - **B机 (接收方)**：显示 `AA`， `BB`，接收 32 字节数据，接收到错误校验和。本地校验和与接收校验和不匹配，发送 `FF` (B机显示 `FF`)。然后B机返回，重新等待接收 `AA` 。

实验结果：

实验结果与预期完全一致。

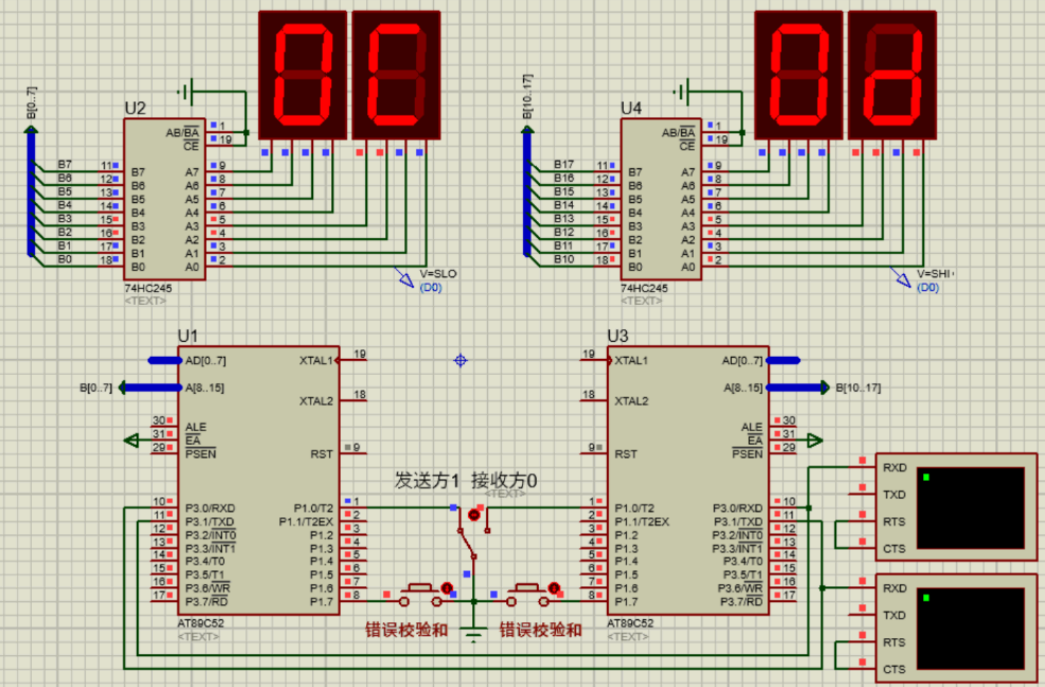
1. **正常通信**：当 P1.7 抬起时，A机发送 32 字节数据和正确校验和，B机校验成功并返回 `00H`，通信正常结束。
2. **错误重发**：当 P1.7 按下时，A机发送了错误校验和，B机校验失败并返回 `FFH`，A机接收到 `FFH` 后自动重新发送数据，完整地实现了差错控制和重发机制。

A、B两机点对点的通信程序示例。
该程序可以在双方机中运行，不同的是在程序运行之前，要判别P1.0口的输入，若P1.0 = 1，表示该机是发送方；若P1.0 = 0，表示该机是接收方。



按钮抬起时，发送方送出正确校验和，发1次！
按钮按下时，发送方送出错误校验和，则重发！

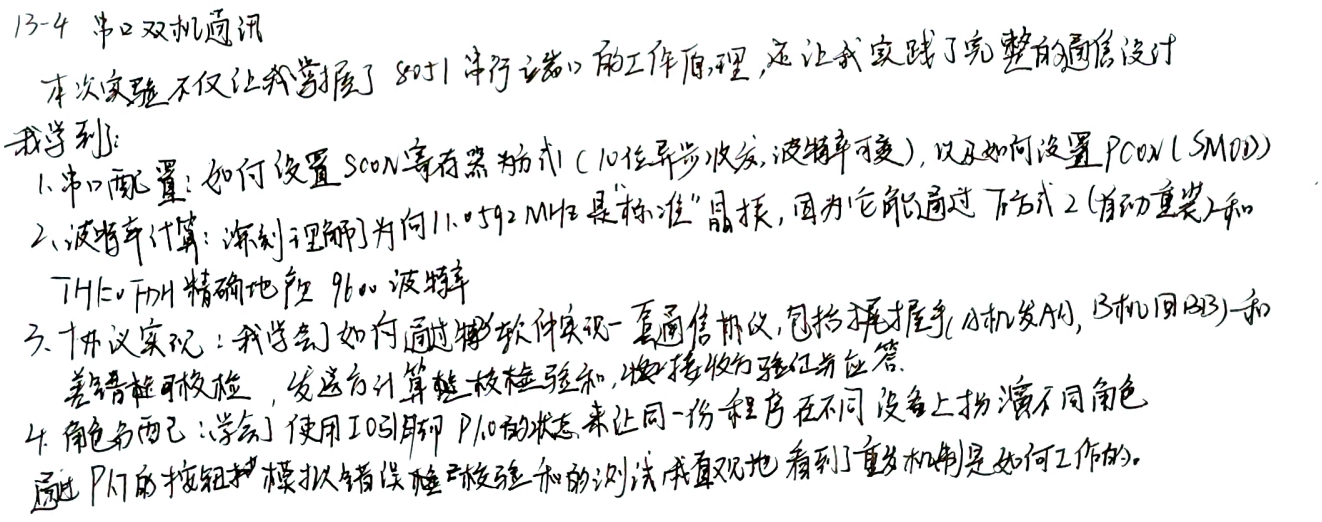
A、B两机点对点的通信程序示例。
该程序可以在双方机中运行，不同的是在程序运行之前，要判别P1.0口的输入，若P1.0 = 1，表示该机是发送方；若P1.0 = 0，表示该机是接收方。



按钮抬起时，发送方送出正确校验和，发1次！
按钮按下时，发送方送出错误校验和，则重发！

3、实验改进（选做）

4、总结



5、附录

```
    ORG     0000H
BUFS            EQU 30H     ;发送数据缓冲区
BUFR            EQU 50H     ;接收数据缓冲区
NUMB            EQU 10H     ;发送/接收数据字节个数
SUMS            EQU 7EH     ;发送检验和单元
SUMR            EQU 7FH     ;接收检验和单元
/******** main主程序*********/
MAIN:   LCALL InitArray
        LCALL   DELAY
        LCALL   InitUart
MAIN2:  JB  P1.0,SEND
        LCALL RECEIVE
HERE:   SJMP HERE
SEND:   LCALL   SEND1
HERE1:  SJMP HERE1
/**********.发送数据单元初始化¯************/
InitArray:  MOV R2,#NUMB
        MOV R0,#BUFS
        MOV A,#00H
LOOP:   MOV @R0,A
        INC R0
        INC A
        DJNZ    R2,LOOP
        RET
/********** 初始化串行口波特率 ************/
InitUart:       ;/* 初始化串行口波特率,使用定时器1*/
            ;/* Setup the serial port for 9600 baud at 11.0592MHz */
    MOV SCON,#50H;  //串口工作方式1
    MOV TMOD,#20H;
```

```
    MOV PCON,#00H;
    MOV TH1,#0FDH
    MOV TCON,#40H;
    RET
/*******发送子程序 **********/
SEND1:
        MOV P2,#0AAH    ;
        MOV SBUF,#0AAH  ;  /*发送联络信号*/
LOOP1:  JNB TI,LOOP1
        CLR TI
LOOP2:  JNB RI,LOOP2
        CLR RI
        MOV A,SBUF
        CJNE A,#0BBH,LOOP2
        MOV P2,#0BBH
        LCALL   DELAY
SEND2:  MOV SUMS,#00H
        MOV R0,#BUFS
        MOV R2,#NUMB
LOOP3:  MOV A,@R0
        MOV P2,A
        LCALL   DELAY
        MOV SBUF,A
LOOP4:  JNB TI,LOOP4
        CLR TI
        ADD A,SUMS
        MOV SUMS,A
        INC R0
        DJNZ R2,LOOP3
LOOP5:  JB P1.7,LOOP6
        ADD A,#01H
        MOV SUMS,A
LOOP6:  MOV P2,A
        LCALL DELAY
        MOV SBUF,A
LOOP7:  JNB TI,LOOP7
        CLR TI
LOOP8:  JNB RI,LOOP8
        CLR RI
        MOV P2,SBUF
        LCALL   DELAY
        MOV A,SBUF
        CJNE A,#00H,LOOP9
ENDS:   RET
LOOP9:  LJMP    SEND2
/*************延时子程序**********************/
DELAY:  MOV     R7,#04H
DL:     MOV     R6,#0FFH
DL1:    MOV     R5,#0FFH
DL5:    DJNZ    R5,DL5
        DJNZ    R6,DL1
        DJNZ    R7,DL
        RET
```

```
END
```