华东交通大学信息与软件工程学院

实验报告册

专　　业　计算机科学与技术

班　　级

姓　　名

学　　号

小班序号

课程名称　单片机原理及应用

教　　师　宋凯

学　　期　2025-2026 学年第 1 学期

2025 年　10 月 15 日

信息与软件工程学院上机实验报告

(第 1 次)

一、实验名称

13.1.1 实验一 数据传送

二、实验目的及要求

1. 实验目的
   通过本次实验掌握 Keil uVision3 开发环境以及单片机汇编语言和C语言源程序编辑、编译、调试方式方法, 熟悉单片机汇编语言指令, 掌握单片机片内外存储器间数据传送的方法, 用汇编语言和C语言编程实现单片机片内外存储器存取数据。
2. 实验要求
   在本次实验前, 在自己的电脑安装 Keil uVision3 以上版本的软件, 根据实验内容和程序框图编写汇编语言和C51程序。

三、实验环境

仿真实验: 装有 Keil uVision3 以上版本软件的笔记本或台式PC机。

四、实验内容

编写软件实现: 设置单片机片内存储器存储区首地址为30H、片外存储器存储区首地址为3000H, 存取数据字节个数16个, 将片内存储区内容设置为01H~10H共16个字节, 读取片内首地址为30H单元内容, 将该内容传送到片外数据存储器存储区中保存, 将保存在片外数据区数据依次取出送P1。观察片内、外存储区数据变化、P1口状态变化。

五、实验设计

1、实验步骤

实验流程图

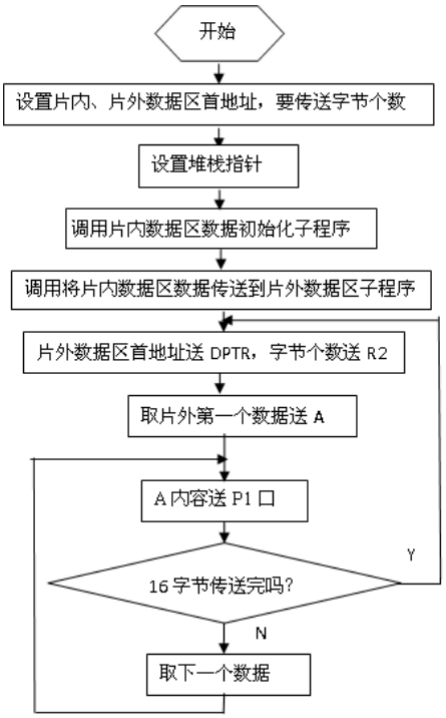


图 13-1 数据传送实验主程序框图

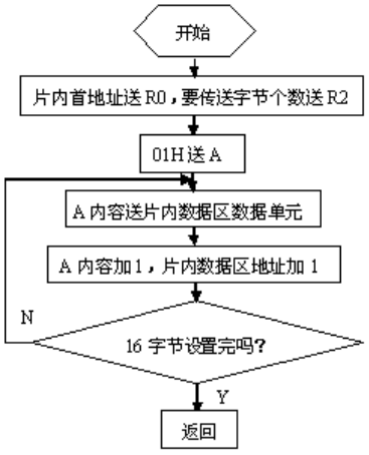


图 13-2 片内数据区数据初始化子程序框图

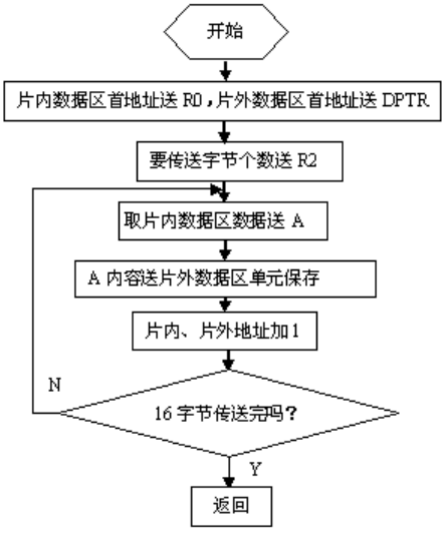


图 13-3 片内数据传送到片外数据区子程序框图

设计思想

本实验利用 8051 单片机的寻址方式和数据传送指令，实现片内 RAM、片外 RAM 和 I/O 端口之间的数据交换。

- **模块化设计**：将程序划分为三个核心模块：片内数据初始化模块、片内到片外数据传送模块、片外数据到 P1 口显示模块。
- **寻址方式**：
  - 片内 RAM 访问：使用寄存器间接寻址方式 (MOV @R0, A 和 MOV A, @R0)，通过 R0 作为地址指针。
  - 片外 RAM 访问：使用数据指针 DPTR 进行寻址 (MOVX A, @DPTR 和 MOVX @DPTR, A)。
  - I/O 端口访问：使用直接寻址方式 (MOV P1, A)。
- **循环控制**：利用 DJNZ 指令来控制16个字节的数据块初始化和传送。
- **数据显示**：设计一个无限循环，将片外 RAM 的数据依次、重复地显示在 P1 端口，以便于在仿真环境中观察结果。

实现步骤

1. **初始化设置**：程序开始后，首先设置堆栈指针 SP 的值（如60H），为后续的子程序调用做准备。
2. **片内数据区初始化**：调用初始化子程序。将片内 RAM 首地址 30H 送入 R0，数据块长度 16 (10H) 送入计数器 R2。将初始数据 01H 送入累加器 A。进入循环：将 A 中的数据存入 R0 指向的地址，然后 A 和 R0 分别加 1，R2 减 1，直到 16 字节（01H~10H）全部写入片内 30H~3FH 区域。
3. **片内到片外数据传送**：调用数据传送子程序。将片内首地址 30H 送入 R0，片外首地址 3000H 送入 DPTR。将数据块长度 16 (10H) 送入计数器 R2。进入循环：先从 R0 指向的片内地址取数到 A，再将 A 中的数据存入 DPTR 指向的片外地址。随后 R0 和 DPTR 分别加 1，R2 减 1，直到 16 字节数据全部复制到片外 3000H~300FH 区域。
4. **数据显示**：执行无限循环。将片外首地址 3000H 送入 DPTR，数据块长度 16 (10H) 送入计数器 R2。进入循环：从 DPTR 指向的片外地址取数到 A，然后将 A 的内容输出到 P1 端口。接着 DPTR 加 1，R2 减 1，循环 16 次。结束后，跳转回数据显示循环的起始位置，重新开始显示。

2、调试过程及实验结果

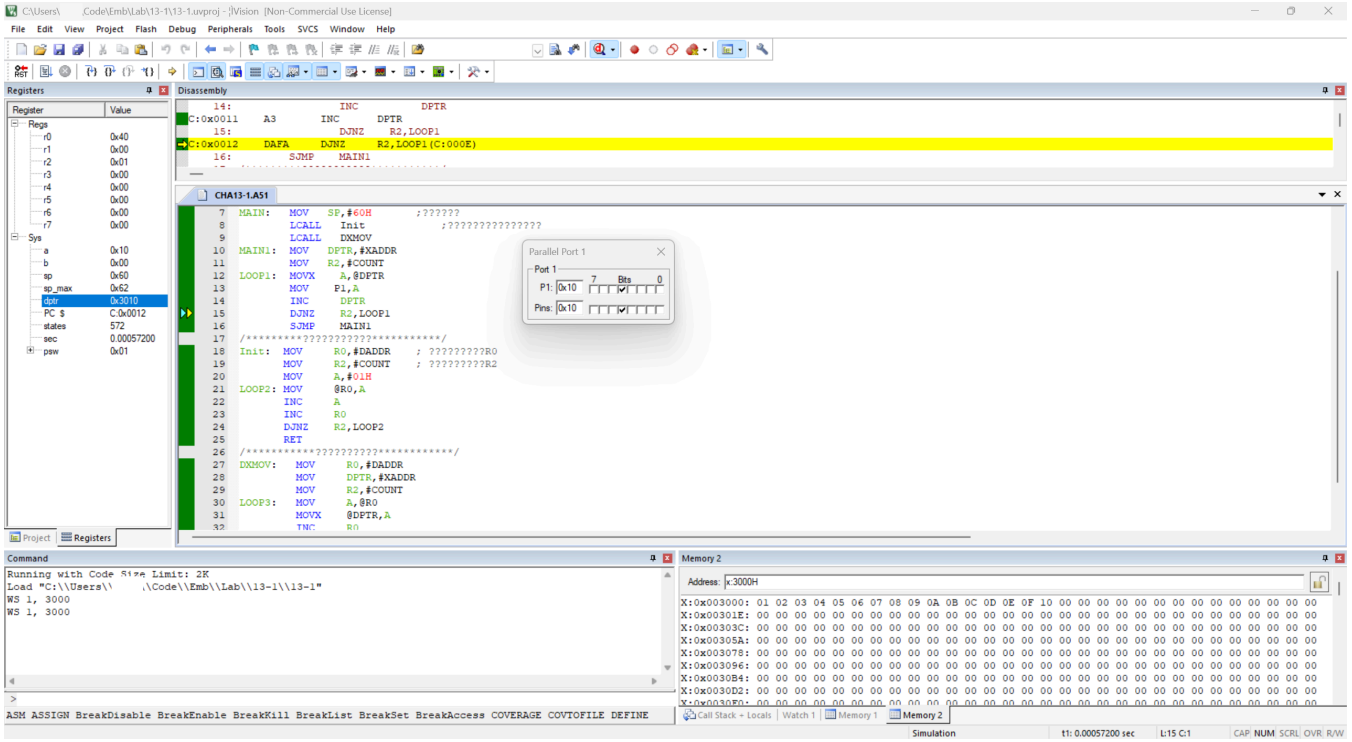
调试过程：

1. 在 Keil uVision3 环境中编译程序，确保没有语法错误。
2. 进入 Debug 调试模式。
3. 打开三个关键窗口：
   - **片内RAM (Internal RAM)**: 打开 `View -> Memory Windows -> Memory 1`。在地址栏输入 `d:30H`。
   - **片外RAM (External RAM)**: 打开 `View -> Memory Windows -> Memory 2`。在地址栏输入 `x:3000H`。
   - **P1 端口**: 打开 `Peripherals -> I/O-Ports -> Port 1`。
4. 使用单步调试 (Step F11) 执行 `LCALL Init` 子程序。观察 Memory 1 (`d:30H`) 窗口，数据从 30H 到 3FH 被依次写入 01H, 02H, ... 10H 。
5. 继续单步调试 (F11) 执行 `LCALL DXMOV` 子程序。观察 Memory 2 (`x:3000H`) 窗口，数据从 3000H 到 300FH 被依次写入 01H, 02H, ... 10H，与片内数据一致。
6. 执行 `MAIN1` 循环，使用连续运行。

实验结果：

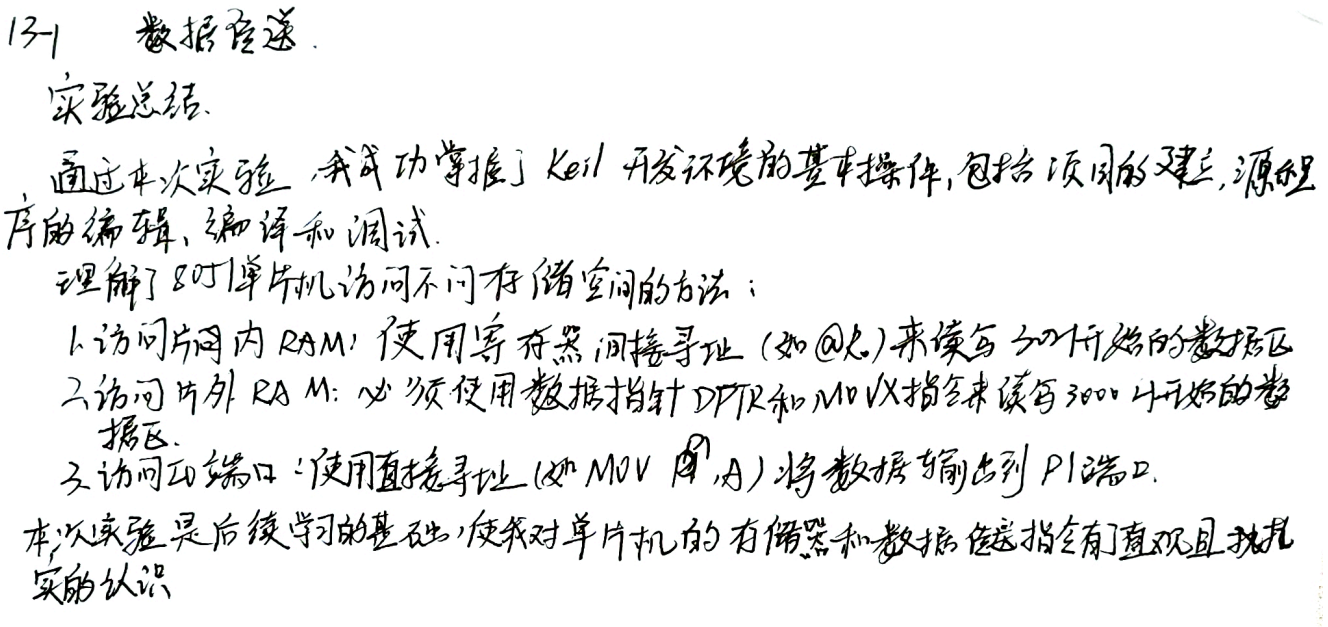
调试运行结果与实验指导书的预期一致。

1. **片内存储器**：`d:30H` 至 `d:3FH` 的 16 个单元内容被成功初始化为 01H 到 10H。
2. **片外存储器**：`x:3000H` 至 `x:300FH` 的 16 个单元内容被成功从片内复制，显示为 01H 到 10H。
3. **P1 端口**：Port 1 窗口的数据循环显示 01H, 02H, 03H... 直到 10H，然后再次从 01H 开始循环。



3、实验改进

4、总结



4、附录

```
/*实验一、文件名：cha13-1.A51*/
        ORG     0000H
DADDR   EQU     30H     ;片内数据区首地址
XADDR   EQU     3000H   ;片外数据区首地址·
COUNT   EQU     10H     ;传送数据大小，共16个字节。

MAIN:   MOV   SP,#60H       ;重置堆栈指针
        LCALL   Init            ;调用片内数据区数据初始化子程序
        LCALL   DXMOV
MAIN1:  MOV   DPTR,#XADDR
        MOV   R2,#COUNT
LOOP1:  MOVX    A,@DPTR
        MOV     P1,A
        INC     DPTR
        DJNZ    R2,LOOP1
        SJMP    MAIN1
/*********片内数据区初始化子程序***********/
Init:    MOV     R0,#DADDR  ; 片内数据区首地址送R0
        MOV     R2,#COUNT    ; 传送数据字节个数送R2
        MOV     A,#01H
LOOP2:  MOV     @R0,A
        INC     A
        INC     R0
        DJNZ    R2,LOOP2
        RET
/***********片内外数据传送子程序************/
DXMOV:   MOV     R0,#DADDR
         MOV     DPTR,#XADDR
         MOV     R2,#COUNT
LOOP3:   MOV     A,@R0
         MOVX    @DPTR,A
         INC     R0
         INC     DPTR
         DJNZ    R2,LOOP3
         RET
```

信息与软件工程学院上机实验报告

(第 2 次)

一、实验名称

13.1.2 实验二 多分支实验

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C51源程序方式方法, 熟悉单片机汇编语言指令, 掌握程序散转的方法, 实现程序的多分支转移。
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13-4)。

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**：1个STC89C52、1个共阳极的七段数码管7SEG-COM-AN-GRN, 一个+5V稳压电源。

四、实验内容

编写软件实现: 根据STC89C52单片机片内30H单元内容(00或01或02或03)进行散转, 1个数码管循环显示对应的数字。(为了便于观察你也可以增加一个循环, 循环取分支变量, 观察显示结果)

五、实验设计

1、实验步骤

实验流程图

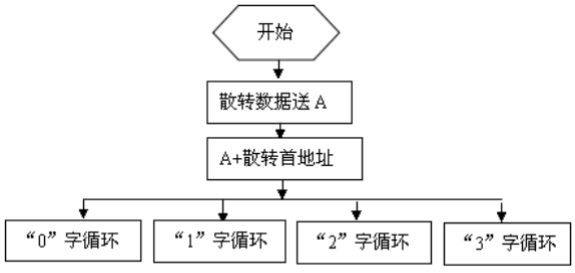


图 13-5 多分支实验框图

设计思想

- 本实验的核心是利用变址寻址指令（如 MOVC A, @A+PC）或跳转指令（JMP @A+DPTR）来实现程序的多分支跳转。
- 首先，在程序存储区（Code Area）定义一个数据表（数码管字型码表）。
- 主程序从片内30H单元读取一个值（00H~03H）到累加器A，这个值将作为查表的偏移量。
- 使用 MOVC 指令，将累加器 A 的内容作为偏移量，与程序计数器 PC（或DPTR，若表基址存入DPTR）相加，

从字型码表中获取对应的共阳极数码管字型码。

- 将获取的字型码输出到 P0 口（或其他连接数码管的端口），点亮数码管显示相应数字。
- 为了实现循环显示，设计一个主循环，不断地从片内30H单元获取数值（或内部循环产生0-3的变量）并执行查表和显示操作。

实现步骤

1. **初始化**：设置堆栈指针 SP。
2. **主循环**：
    - a. (方案一) 从片内RAM 30H单元获取分支数据（0-3）到累加器A。
    - b. (方案二) 或者设置一个循环变量（如R2），使其在0-3之间循环变化，并将R2的值送入A。
    - c. 将累加器 A 的值（偏移量）与后续字型码表的基地址相加，以计算目标字型码的地址。
    - d. 执行查表指令 MOVC A, @A+PC。
    - e. 将查表得到的字型码（如 0C0H, 0F9H, 0A4H, 0B0H 等）输出到 P0 端口。
    - f. 调用一个延时子程序，以便人眼能清晰观察到数码管显示的数字。
    - g. 跳转回主循环的起始处，继续下一个数字的显示。
3. **字型码表**：在主循环之后，定义字型码表（DB 伪指令），按0, 1, 2, 3的顺序存放共阳极字型码。

2、调试过程及实验结果

调试过程：

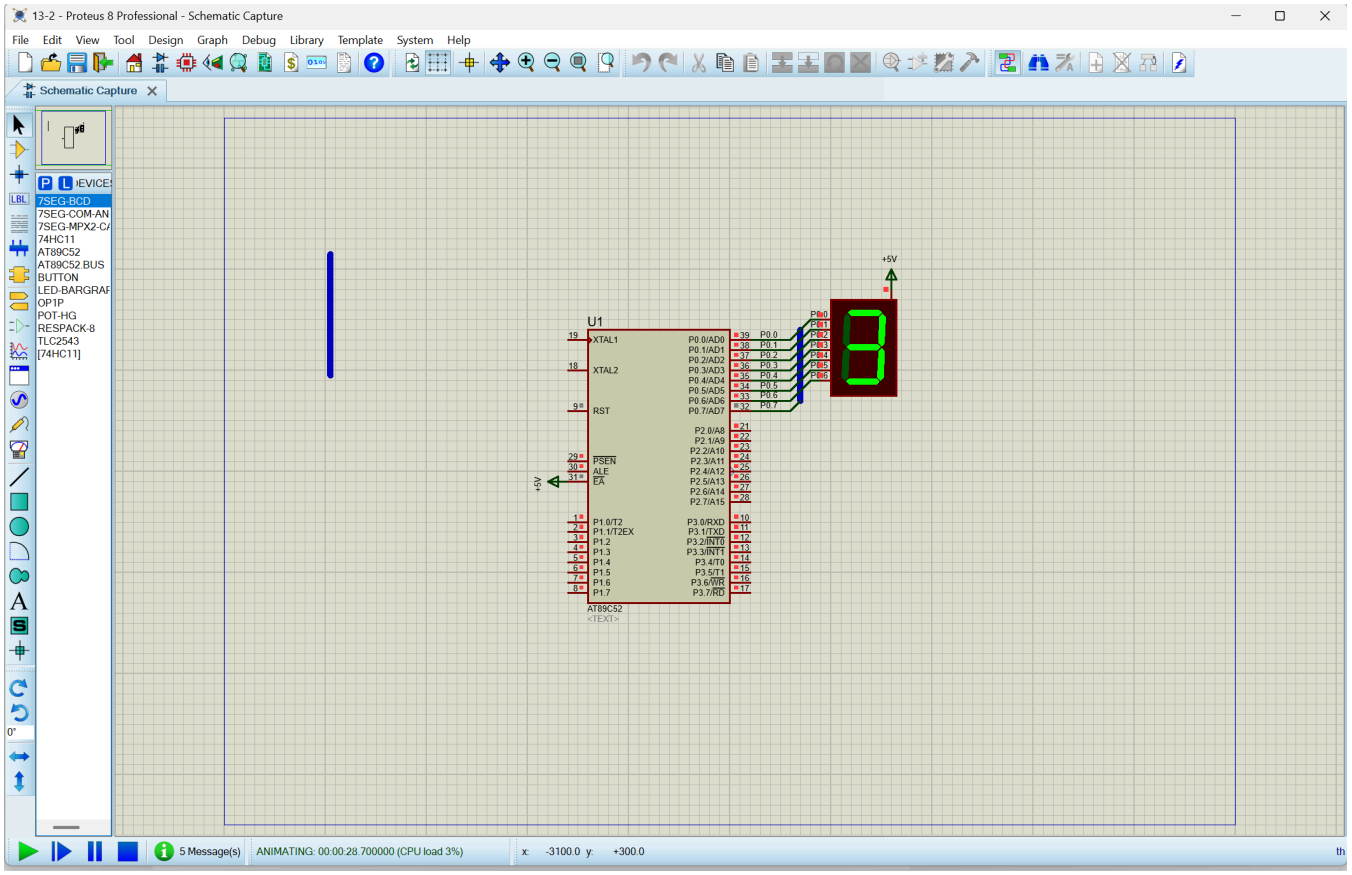
1. **硬件搭建**：在 Proteus 中按照图 13-4 搭建电路。使用一个 STC89C52，将 P0.0 至 P0.7 连接到 `7SEG-COM-AN-GRN` (共阳极七段数码管) 的 a 至 g, dp 引脚
2. **Keil-Proteus 联调设置**：在 Keil 中设置 "Options for Target"，在 "Debug" 标签下选择 "Use: Proteus VSM Simulator"。在 "Output" 标签下确保 "Create HEX File" 已勾选
3. **加载程序**：在 Proteus 的 STC89C52 元件属性中，加载 Keil 生成的 .hex 文件。
4. **程序执行**：
    - 程序（如附录代码）首先初始化 `NUMB` (30H) 单元为 00H。
    - 进入 `START` 循环，将 `NUMB` (00H) 送入 A。
    - 执行 `RL A` (A 变为 00H)，`JMP @A+DPTR`，跳转到 `TABADDR`+0，即 `SJMP DISP0`。
    - `DISP0` 将 0C0H (共阳极'0'的字型码) 存入 `FONT` 单元，并调用 `MJUP1` 显示在 P0 口。
    - 延时后，`INC NUMB` (NUMB 变为 01H)，R2 未减到 0，循环回 `START`。
    - 第二次循环，A=01H，`RL A` =02H，`JMP @A+DPTR` 跳转到 `TABADDR`+2，即 `SJMP DISP1`。
    - `DISP1` 将 0F9H (共阳极'1'的字型码) 显示在 P0 口。
    - 依此类推，直到 R2=0，程序重置 `NUMB` 为 00H，重新开始循环。

实验结果：

启动 Proteus 仿真后，实验现象与预期完全一致

1. **Proteus 仿真**：电路中的七段数码管循环显示数字 "0", "1", "2", "3"，然后又从 "0" 开始，每次显示之间有明显延时，便于观察。

2. **Keil 调试**：若在 Keil 中联调，在 `MOV P0, A` 处设置断点，可以看到 P0 端口的值依次变为 0C0H, 0F9H, 0A4H, 0B0H，这与实验指导书中给出的字型码一致



3、实验改进（选做）

4、总结

13-2 多分支实验.

本次实验让我掌握了程序的多分支转移的方法

我通过实验践理解了两种实现方式.

1. 跳转表：使用 JMP @A+DPTR 指令，搞根据累加器A的值跳转到不同的程序分支。

2. 查表法：使用 MOVC A, @A+PC 或 MOVC A, @A+DPTR 指令，从存储器中查找预设数据

通过调试，我加深了对共阳极数码管字型码的理解，并认识到P0口作为输出口需要上拉电阻

这个实验锻炼了我的硬件~~调试能力~~电路搭建能力和程序逻辑设计能力.

5、附录

```
/*实验二    分支程序，文件名：13-2.A51*/
        ORG 0000H
NUMB    EQU 30H
FONT    EQU 31H
INT:        MOV NUMB,#00H      ;需要显示数字送NUMB单元
        MOV R2,#04H
START:  MOV A,NUMB
        MOV DPTR,#TABADDR
```

```
        RL  A
        JMP @A+DPTR
TABADDR:SJMP    DISP0
        SJMP    DISP1
        SJMP    DISP2
        SJMP    DISP3
DISP0:  MOV FONT,#0C0H
        LJMP    MJUP1
DISP1:  MOV FONT,#0F9H
        LJMP    MJUP1
DISP2:  MOV FONT,#0A4H  ;
        LJMP    MJUP1
DISP3:  MOV FONT,#0B0H
/**************数码管显示子程序*****************/
MJUP1:  MOV     A,FONT
        MOV    P0,A          ;字型送P0口
        ACALL   DELAY
        INC     NUMB
        DJNZ    R2,START
        AJMP    INT
DELAY:  MOV    R7,#0FFH
DL:     MOV    R6,#0FFH
DL6:    DJNZ   R6,DL6
        DJNZ   R7,DL
        RET
        END
```

信息与软件工程学院上机实验报告

(第 3 次)

一、实验名称

13.1.3 实验三 外部中断与定时器/计数器中断实验

二、实验目的及要求

1. 实验目的
   通过本次实验进一步熟悉 Keil uVision3 和 Proteus 开发环境以及它们联机调试单片机汇编语言和C51 源程序方式方法, 掌握外部中断和定时/计数器的工作原理和使用方法。
2. 实验要求
   在本次实验前, 使用Proteus 软件画硬件仿真原理图 (如图13-6) 所示, STC89C52 的P3.2连接按键K, P1.0口连接示波器。

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**：1个STC89C52、1个按键开关 BUTTON、1个+5V 稳压电源、一个示波器。

四、实验内容

若系统时钟频率为6MHz, 编写软件实现: 按一下K键, 产生一次外部中断0中断信号, 启动T1定时, 使P1.0输出周期为1ms的方波。

五、实验设计

1、实验步骤

实验流程图

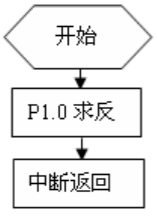


图 13-7　定时/计数器 1 中断服务子程序框图

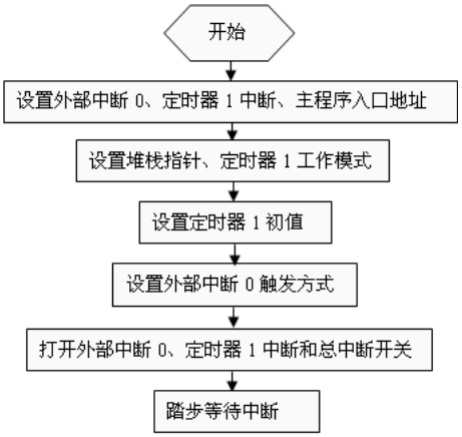


图 13-8　外部中断与定时/计数器中断主程序框图

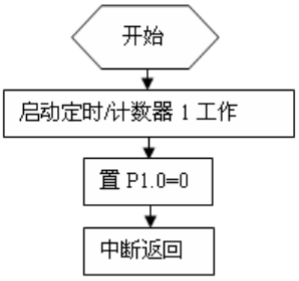


图 13-9　外部中断 0 中断服务子程序框图

设计思想

- 本实验通过结合外部中断0（INT0）和定时器T1中断，实现按键控制方波输出。
- 主程序负责初始化：设置T1为工作方式1（16位定时器），设置INT0为下降沿触发（IT0=1），使能T1中断（ET1=1）、INT0中断（EX0=1）和总中断（EA=1）。
- 定时器初值计算：系统时钟6MHz，机器周期 $T_{cy} = 12/6MHz = 2\mu s$。方波周期1ms，即高电平0.5ms，低电平0.5ms。定时器需定时 $0.5ms = 500\mu s$。计数值 $N = 500\mu s/2\mu s = 250$。T1初值 \= $65536 - 250 = 65286 = \mathrm{FE06H}$。
- 主程序初始化后进入一个空循环（踏步等待中断）。
- 当按键K按下时，P3.2产生下降沿，触发INT0中断。
- 在INT0中断服务程序中：启动定时器T1（SETB TR1），并设置P1.0初始电平（CLR P1.0）。
- 在T1中断服务程序中：对P1.0口取反（CPL P1.0），然后重新装载T1的初值（TH1=FEH, TL1=06H），以维持1ms周期的方波，并实现连续输出。

实现步骤

1. **中断向量设置**：程序起始处设置中断向量，ORG 0000H LJMP MAIN，ORG 0003H LJMP INT0_ISR (外部中断0)，ORG 001BH LJMP T1_ISR (定时器1中断)。
2. **主程序 (MAIN)**:
    - a. 设置堆栈指针SP。
    - b. 设置定时器T1工作模式：MOV TMOD, #01H (T1为方式1)。
    - c. 设置T1初值（用于0.5ms定时）：MOV TH1, #0FEH，MOV TL1, #06H。

    - d. 设置INT0触发方式：SETB IT0 (TCON寄存器中, 设置为下降沿触发)。
    - e. 打开中断：MOV IE, #10000110B (EA=1, EX0=1, ET1=1)。
    - f. 进入等待循环：HERE: SJMP HERE。
3. **外部中断0服务程序 (INT0_ISR)**:
    - a. 启动定时器T1：SETB TR1。
    - b. 设置P1.0初始电平：CLR P1.0。
    - c. 中断返回：RETI。
4. **定时器T1中断服务程序 (T1_ISR)**:
    - a. 重新装载T1初值：MOV TH1, #0FEH，MOV TL1, #06H。
    - b. P1.0引脚电平取反：CPL P1.0。
    - c. 中断返回：RETI。

2、调试过程及实验结果

调试过程：

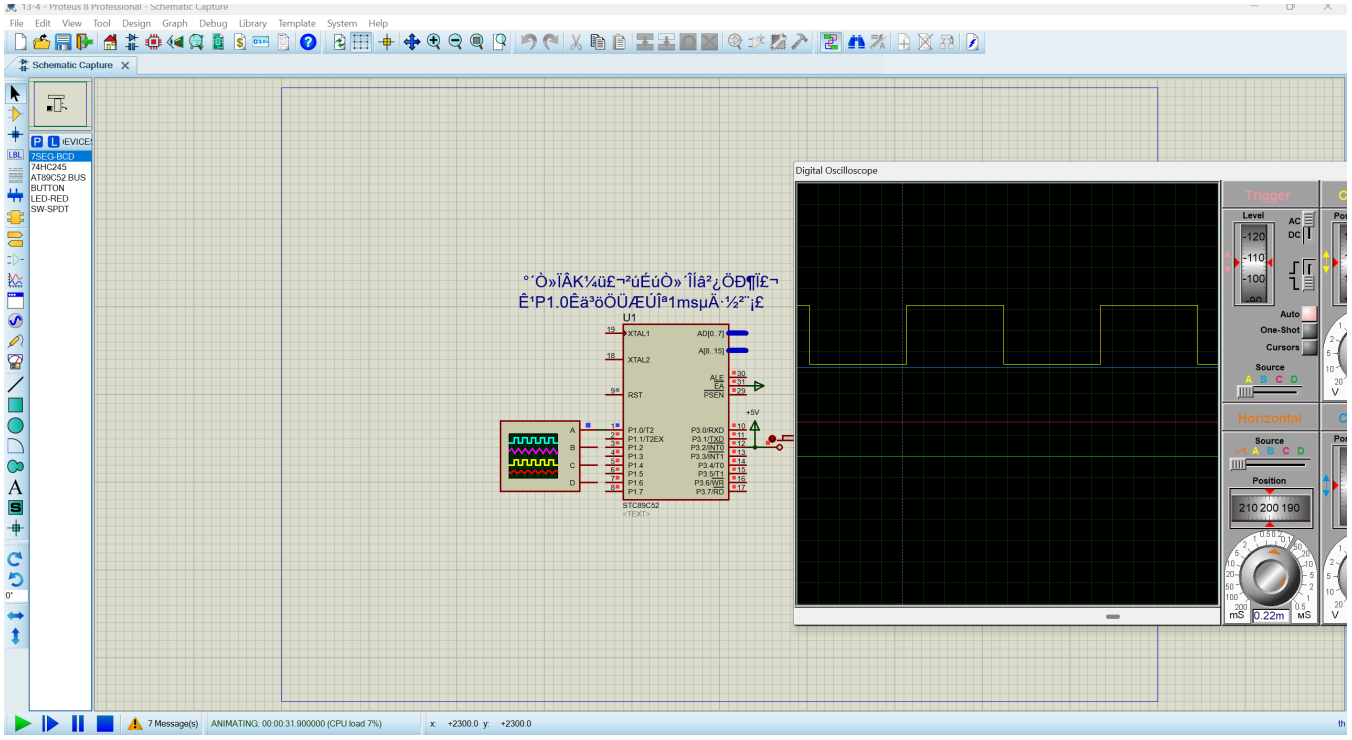
1. **硬件搭建**：在 Proteus 中按照图 13-6 搭建电路。将一个 `BUTTON` 连接到 P3.2 (INT0)，并接一个下拉电阻到地（确保按下时产生下降沿）。将 `OSCILLOSCOPE` (示波器) 的 A 通道连接到 P1.0 。
2. **程序分析**:
    - `MAIN` 函数设置 T1 为方式 2 (8 位自动重装)
    - 设置 `TH1` (重装值) 和 `TL1` (初始值) 均为 06H。
    - 计算定时：时钟 6MHz，机器周期 $T_{cy} = 12/6MHz = 2\mu s$。T1 方式 2 定时时间 = $(256 - \mathrm{TH1}) \times T_{cy} = (256 - 6) \times 2\mu s = 250 \times 2\mu s = 500\mu s$ (即 0.5ms)。
    - 设置 INT0 为下降沿触发 (`SETB IT0`)，并使能 EA, EX0, ET1 中断 。
    - 主程序进入 `SJMP HERE` 等待中断。
3. **运行仿真**:
    - 启动 Proteus 仿真。此时，程序在 `HERE` 循环，P1.0 无输出，示波器显示一条直线（0V）。
    - 点击按键 K。P3.2 产生下降沿，触发 INT0 中断。
    - 程序跳转到 `INT0P`，执行 `SETB TR1` 启动定时器 T1，并 `CLR P1.0` 。示波器电平变为 0V。
    - T1 开始计数，0.5ms 后溢出，触发 T1 中断。
    - 程序跳转到 `T1P`，执行 `CPL P1.0`，P1.0 变为高电平。T1 自动重装 06H 并重新开始计数。
    - 又过了 0.5ms，T1 再次溢出，执行 `CPL P1.0`，P1.0 变回低电平。
    - 此过程不断重复。

实验结果：

实验成功。

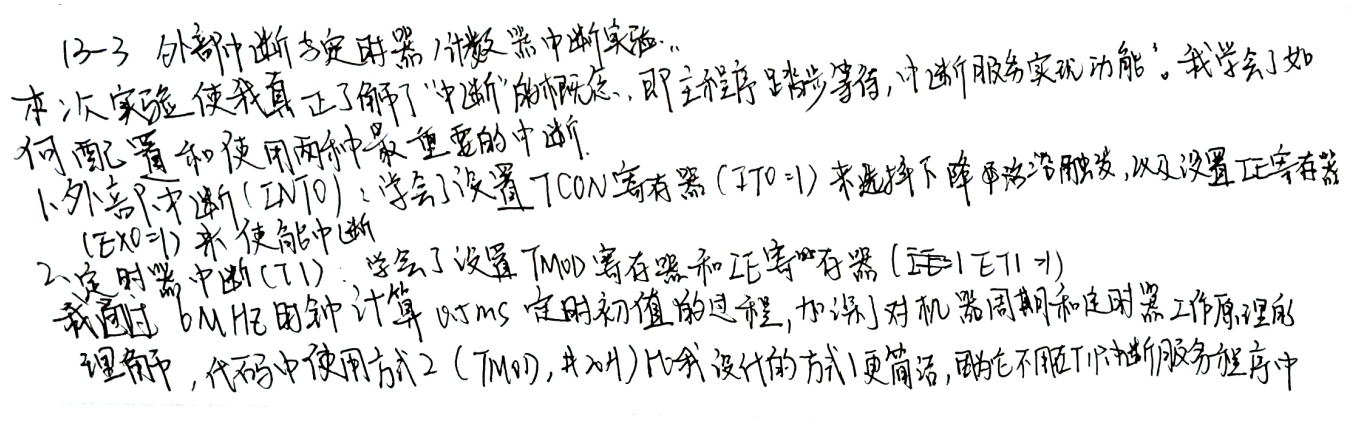
1. **启动前**：P1.0 保持低电平，示波器为一条直线。
2. **按下 K 键后**：示波器立即开始显示一个连续的方波。
3. **波形分析**：在示波器上测量该方波，其高电平持续时间为 0.5ms (500μs)，低电平持续时间也为 0.5ms (500μ

s)。因此，方波的完整周期为 1ms，频率为 1KHz。这与实验内容“使P1.0输出周期为1ms的方波”的要求完全一致 。



3、实验改进（选做）

4、总结



5、附录

```
        ORG     0000H
        LJMP MAIN
        ORG      0003H
        LJMP   INT0P
        ORG    001BH
        LJMP    T1P
        ORG    0100H
MAIN:   MOV   SP,#60H
        MOV   TMOD,#20H
        MOV   TL1,#06H
        MOV    TH1,#06H
        SETB   ET1
```

```
        SETB  IT0
        SETB  EX0
        SETB  EA
HERE:   SJMP  HERE
INT0P:  SETB  TR1
        CLR   P1.0
        RETI
T1P:    CPL   P1.0
        RETI
        END
```

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

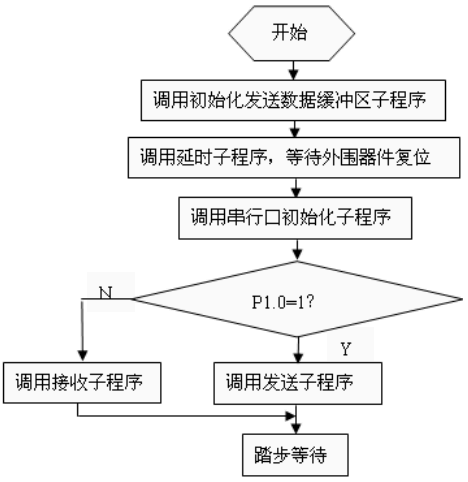


图 13-11 点对点通讯主程序框图

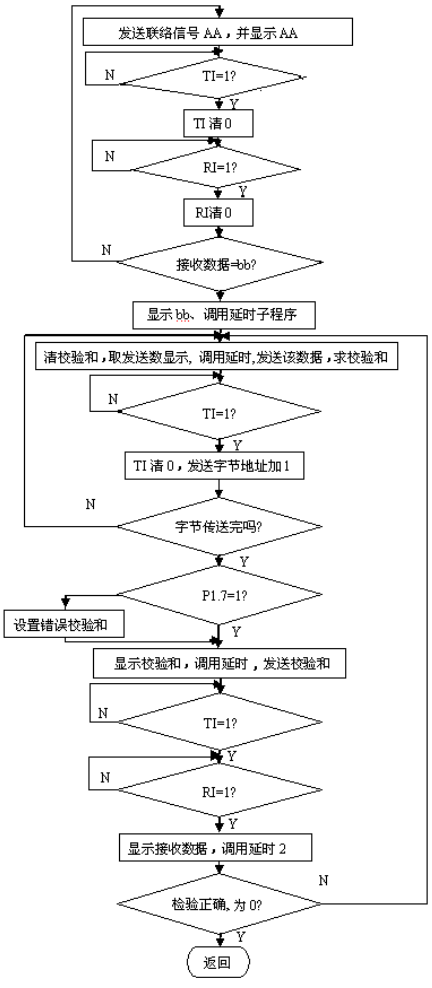


图 13-12 点对点通讯发送子程序框图

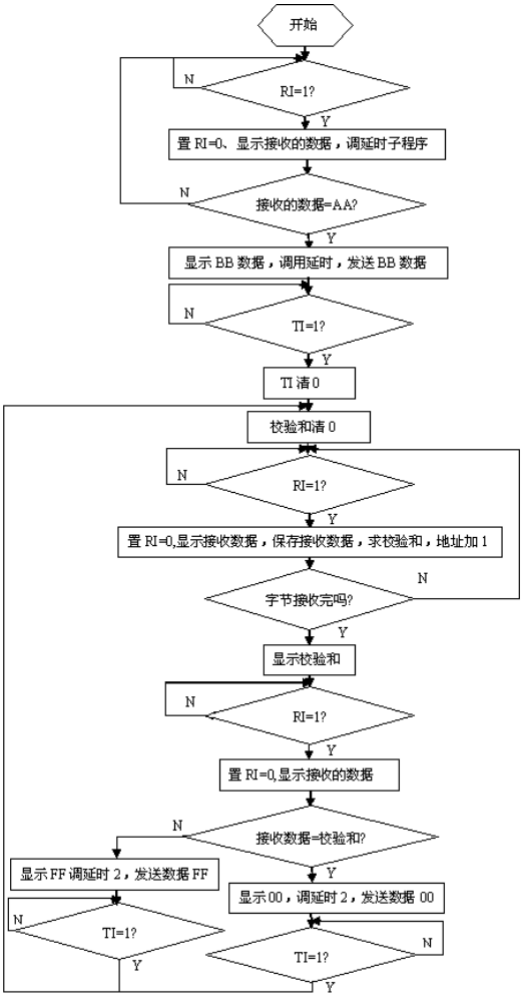


图 13-13 点对点通讯接收程序框图

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。

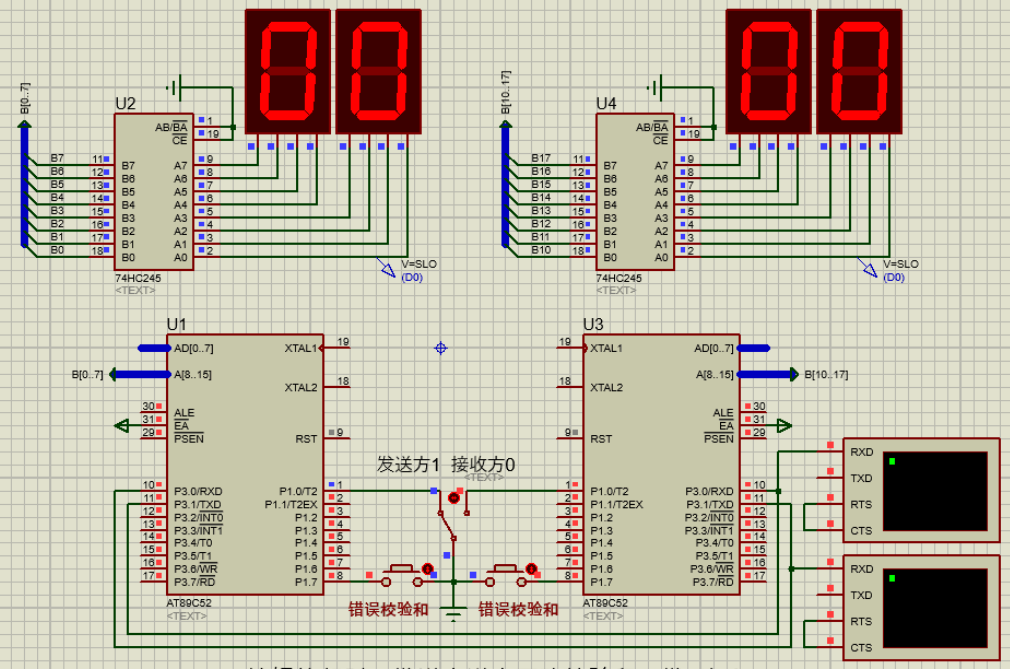
2. **程序加载**：将编译好的 *同一个* .hex 文件加载到 A机 和 B机 中。
3. **波特率设置**：程序中设置 T1 为方式 2 (`TMOD, #20H`)， `TH1` 设为 0FDH。使用 11.0592MHz 晶振，波特率 = $11.0592MHz/(12\times(256-253))/2=9600\ \text{bps}$ (SMOD=0)。
4. **测试一：正常通信 (P1.7 抬起)**
    - 保持 P1.7 按钮不按 (P1.7=1，发送正确校验和)。
    - 运行仿真。
    - **A机 (发送方)**：显示 `AA` ，等待... 显示 `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。

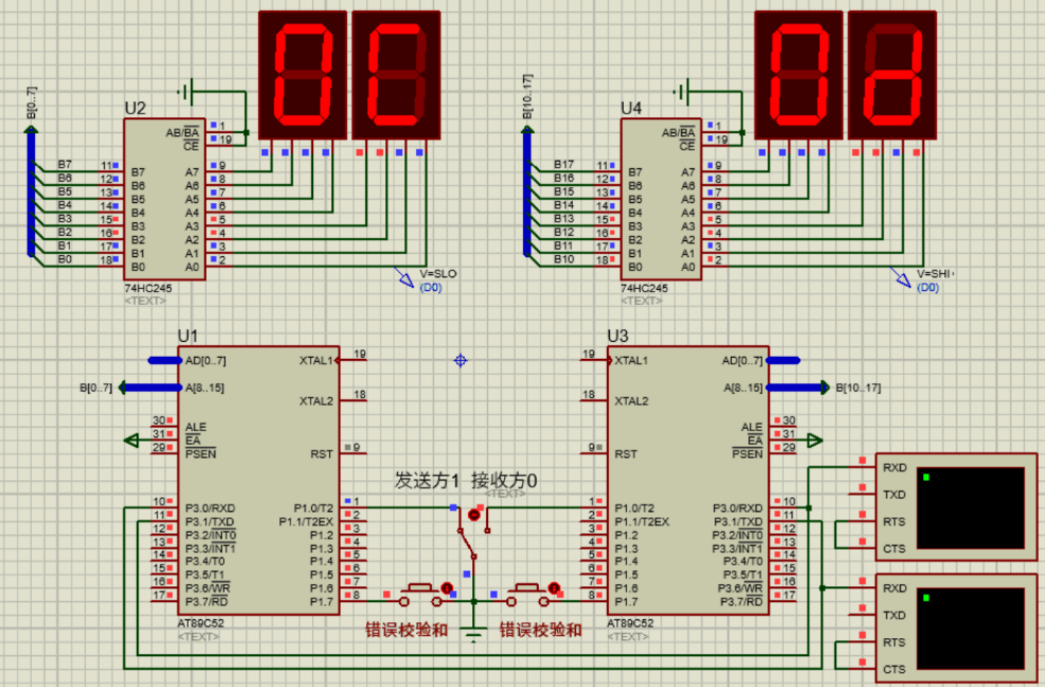
1. **正常通信**：当 P1.7 抬起时，A机发送 32 字节数据和正确校验和，B机校验成功并返回 `00H`，通信正常结束。
2. **错误重发**：当 P1.7 按下时，A机发送了错误校验和，B机校验失败并返回 `FFH`，A机接收到 `FFH` 后自动重新发送数据，完整地实现了差错控制和重发机制。

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！

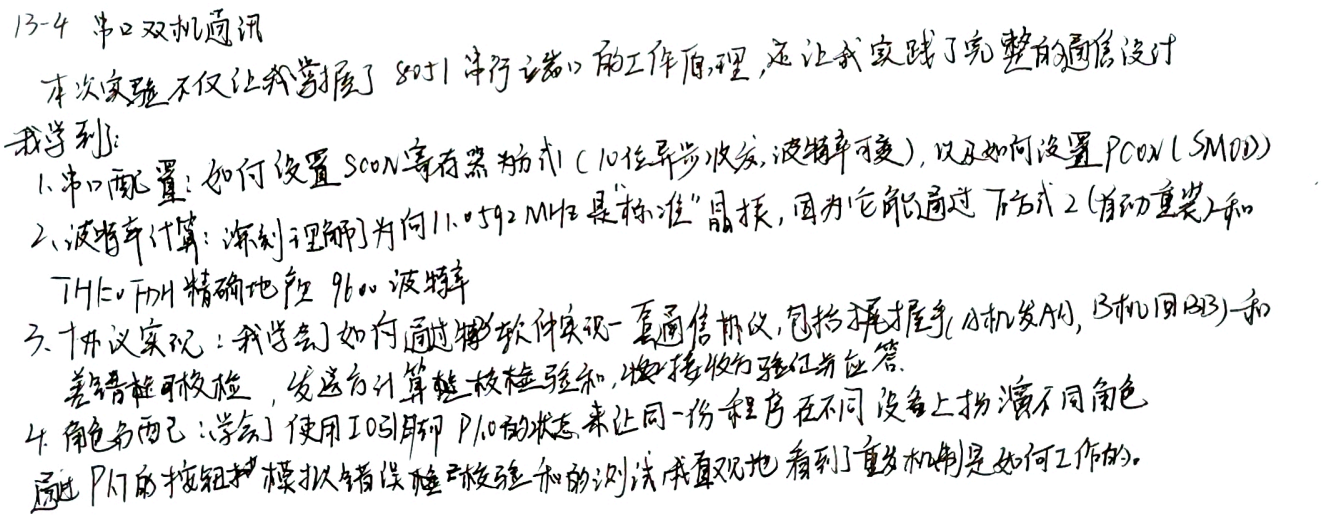
A、B两机点对点的通信程序示例。
该程序可以在双方机中运行，不同的是在程序运行之前，要判别P1.0口的输入，若P1.0 = 1，表示该机是发送方；若P1.0 = 0，表示该机是接收方。



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
```