1.1 基本概念

什么是单片机

ROM - Flash

RAM - SRAM

MCU

MCU - Micro Controller Unit

1.2 发展概况

系列介绍

8位

Intel MCS-51兼容单片机

Atmel AVR单片机 Arduino-UNO

16位

Ti MSP430

32位

ST STM32

Atmel AVR32 Arduino-MEGA2560

乐鑫 ESP32-C3 RISC-V

1.3 特点

1.4 趋势

低电压：3.3V - 1.8V **STM32L4**

高速：32位 高主频 **STM32H7/F7** OpenMV

集成性：A/D D/A **MSP430**

2.1 MCS-51单片机的基本组成

2.1.1 MCS-51单片机的基本组成

1.中央处理器

2.内部程序存储器

内部ROM

存放程序、原始数据、表格内容

3.内部数据存储器

RAM的低128B

存放随机数据及运算结果

4.定时/计数器

2个16位

5.并行IO口

4个8位并行IO

P0：低8位地址/8位数据，GPIO

P1：GPIO

P2：高8位地址，GPIO

P3：第二功能输入、输出，GPIO

6.串行口

全双工异步通信收发

7.中断控制系统

2个外部中断源

2个定时/计数器中断源

1个串行中断源

8.时钟电路

外接石英晶体和微调电容

8051：25MHz

89C51：24MHz

89S51：33MHz

9.位处理器

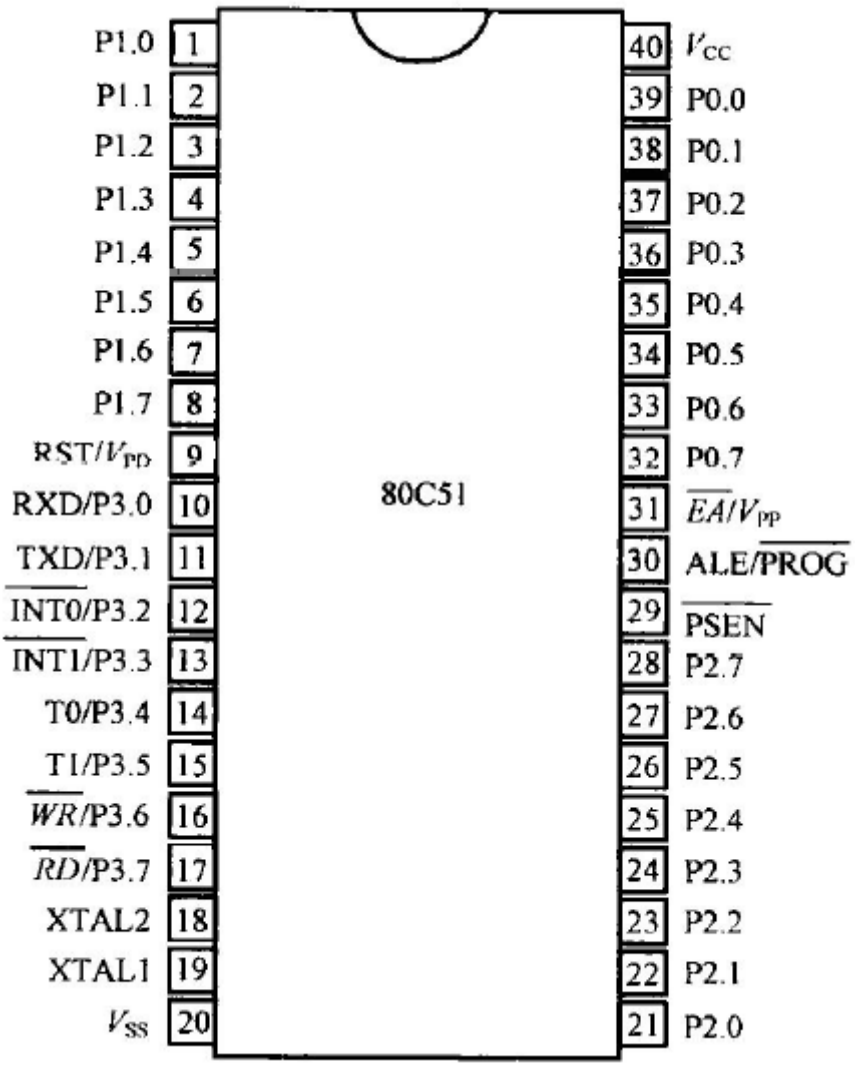
10.内部总线

2.1.2 封装及引脚

一、封装

DIP-40 双列直插

二、引脚



1.输入、输出

P0-P3

2.电源及时钟

V_{CC}：+5V

V_{SS}：GND

XTAL：晶振

3.控制线和复位

ALE：访问外部存储器时锁存地址；不访问时输出振荡器频率1/6的正脉冲信号

$\overline{EA}$：EA为低时，只访问外部程序存储器；EA为高时，先访问内部，超出容量后访问外部

$\overline{PSEN}$：访问外部程序存储器时每周期两次有效

RST：两个周期高电平触发复位

三、第二功能

1.P3

引脚	第二功能	第二功能信号名称
P3.0	RXD	串行口输入端
P3.1	TXD	串行口输出端
P3.2	$\overline{INT0}$	外部中断 0 请求输入端，低电平有效
P3.3	$\overline{INT1}$	外部中断 1 请求输入端，低电平有效
P3.4	T0	定时/计数器 0 的计数脉冲输入端
P3.5	T1	定时/计数器 1 的计数脉冲输入端
P3.6	$\overline{WR}$	外部 RAM 写选通信号输出端，低电平有效
P3.7	$\overline{RD}$	外部 RAM 读选通信号输出端，低电平有效

2.烧录

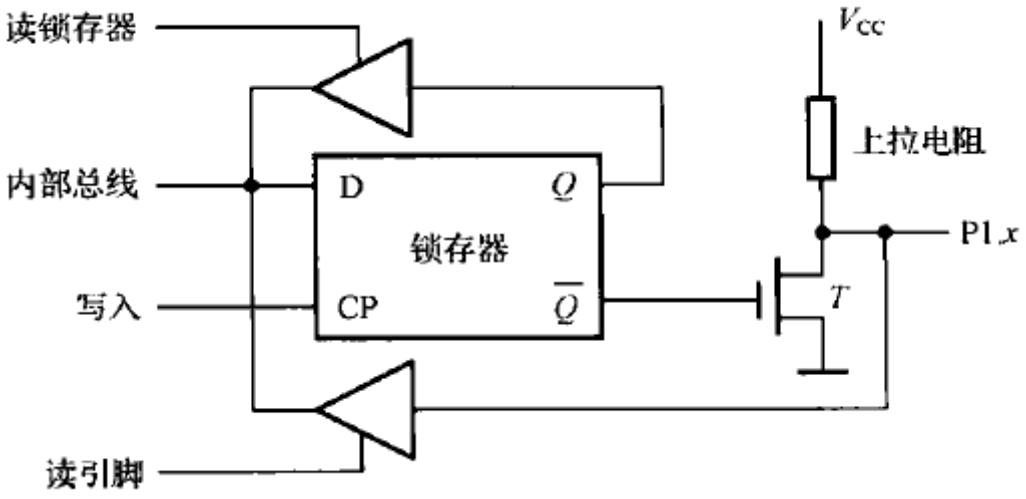
$\overline{PROG}$拉低，VPP接25V

3.备用电源

V_{PD}保护内部RAM不掉电

2.2 MCS-51单片机的并行I/O端口结构

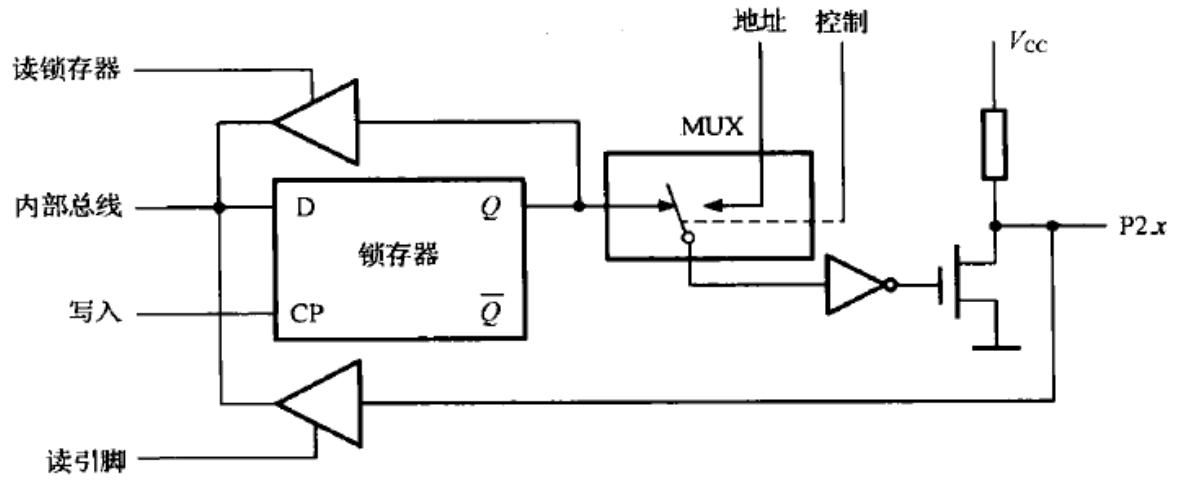
2.2.1 P1口



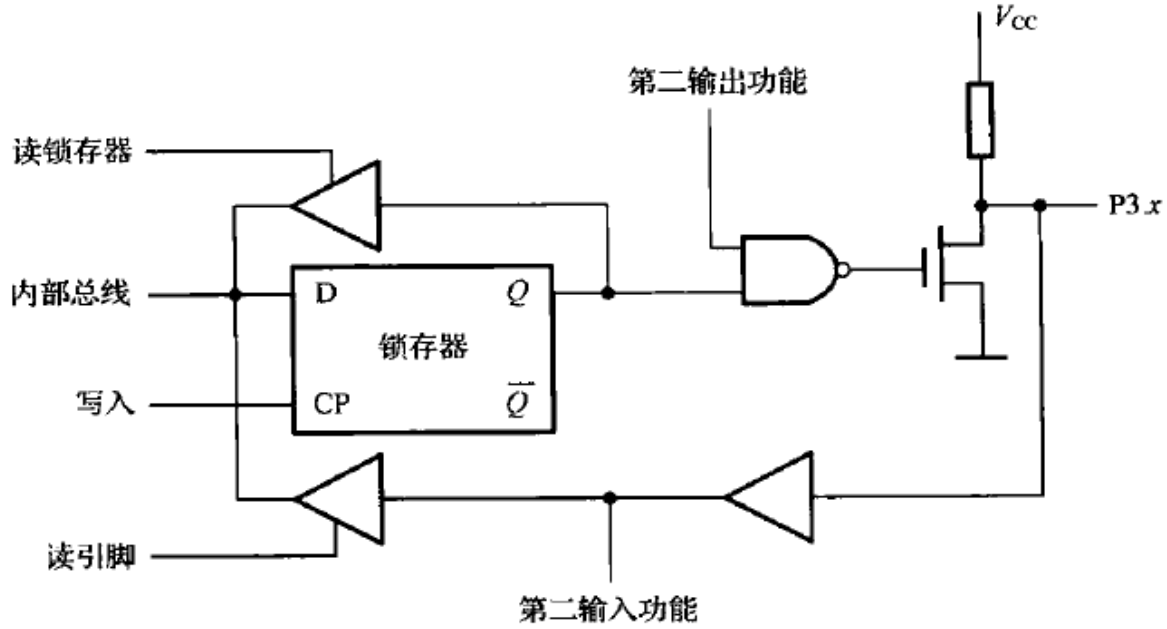
上拉电阻为场效应管等效

输入前应输出1防止短路（准双向I/O口）

2.2.2 P2口



2.2.3 P3口



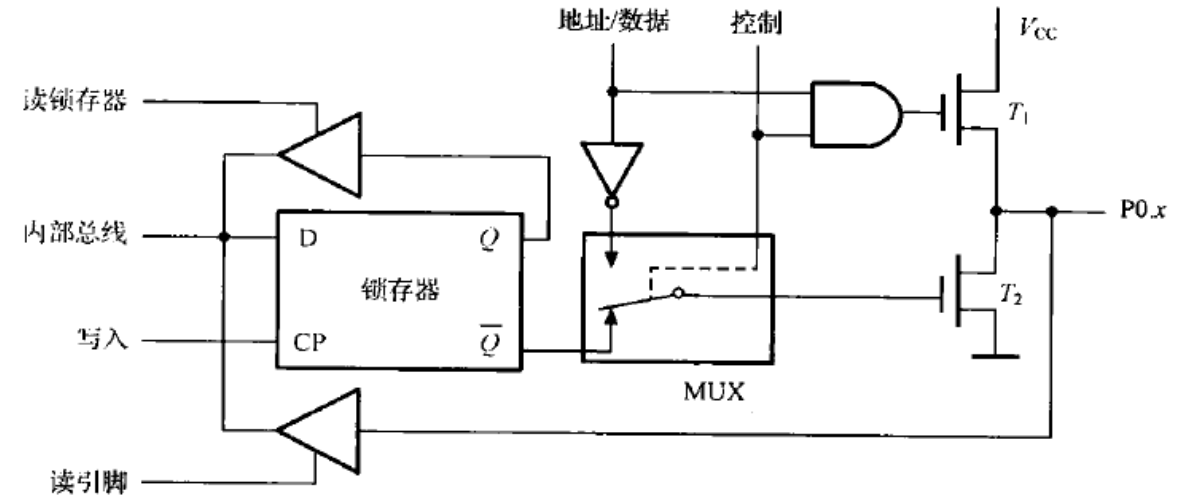
第一输出功能：第二输出功能信号线置1，与非门充当非门

第二输出功能：锁存器置1，与非门充当非门

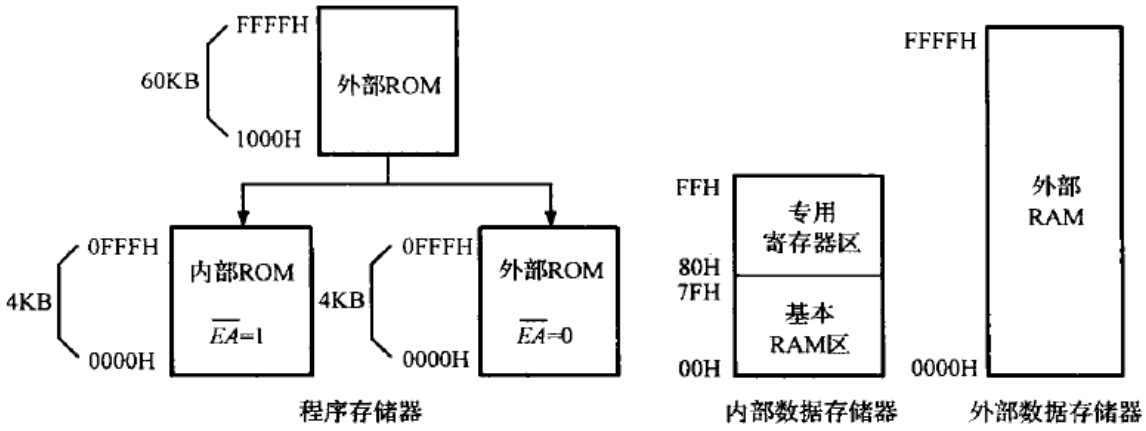
第一输入功能：锁存器置1，第二输出功能信号线置1，场效应管截止

第二输入功能：锁存器置1，第二输出功能信号线置1，场效应管截止

2.2.4 P0口



2.3 MCS-51单片机的存储器结构



2.3.1 程序存储器ROM

$\overline{EA}=1$，单片机先执行片内4KB，地址超出后自动转向片外

$\overline{EA}=0$，单片机执行片外程序存储器内容

0003H-002AH单元均匀分为5段

特殊地址	功　能
0000H	主程序入口
0003H	外部中断 0 入口地址
000BH	定时/计数器 0 溢出中断入口地址
0013H	外部中断 1 入口地址
001BH	定时/计数器 1 溢出中断入口地址
0023H	串行口中断入口地址

主程序从0030H单元之后开始

2.3.2 数据存储器RAM

内部数据存储器：256B，00H-FFH，使用`MOV`访问

外部数据存储器：最大64KB，0000H-FFFFH，使用`MOVX`访问

一、基本RAM区

1.工作寄存器区 00H-1FH

RS1	RS0	工作寄存器组	工作寄存器地址
0	0	工作寄存器组 0	R0～R7 对应的地址为 00～07H
0	1	工作寄存器组 1	R0～R7 对应的地址为 08～0FH
1	0	工作寄存器组 2	R0～R7 对应的地址为 10～17H
1	1	工作寄存器组 3	R0～R7 对应的地址为 18～1FH

2.位寻址区 20H-2FH

16个单元，共16*8=128位，既可以按字节寻址，也可以按位寻址

3.用户RAM区 30H-7FH

80个单元，按字节寻址

二、特殊功能寄存器区

1.程序计数器PC

16位，存放下一条指令地址，自动加1

2.累加器A/ACC

8位，E0H，可存放操作数

3.B寄存器

8位，F0H，主要用于乘除运算，也可作为一般数据寄存器使用

4.数据指针寄存器DPTR

16位

DPH：DPTR高位字节，83H

DPL：DPTR低位字节，82H

访问外部数据存储器时作地址指针使用

5.程序状态字寄存器PSW

8位，D0H

位序	PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
位标志	CY	AC	F0	RS1	RS0	OV	—	P

CY

进位标志位

AC

辅助进位标志位

F0

用户标志位，自行决定

RS1、RS0

选择工作寄存器组

OV

溢出标志位

P

奇偶标志位，累加器A中1为偶数置0，反之置1

6.堆栈指针寄存器SP

8位，81H，存放堆栈栈顶地址

堆栈遵循LIFO，向上生长

进栈操作：`PUSH`，先SP加1，后写入数据

出栈操作：`POP`，先读出数据，后SP减1

SP初值为07H，但一般设置为30H-7FH地址空间的高端

2.4 MCS-51单片机的时钟电路与时序

2.4.1 时钟电路

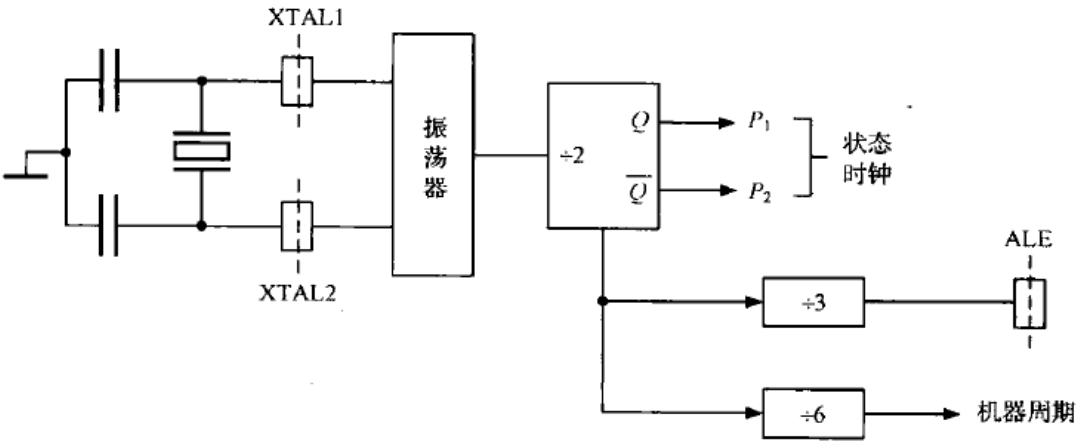
一、内部时钟

XTAL1与XTAL2跨接晶体或陶瓷振荡器

晶体振荡器选择20-40pF电容，陶瓷振荡器选择30-50pF电容

二、外部时钟

针对AT89，XTAL1接外部时钟，XTAL2悬空



2.4.2 时序

一、振荡周期

晶振的振荡周期

二、时钟周期

振荡脉冲二分频后的信号周期

三、机器周期

一个机器周期 - 6个状态 - 12个节拍（振荡周期）

四、指令周期

执行一条指令需要的时间

2.5 MCS-51单片机的工作方式

2.5.1 复位方式

一、复位信号

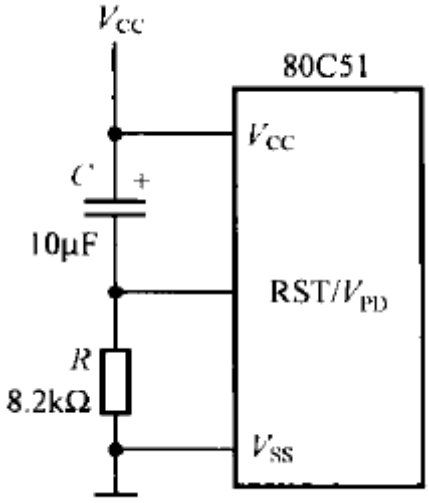
RST保持两个机器周期（24个振荡周期）以上高电平触发复位

二、复位操作

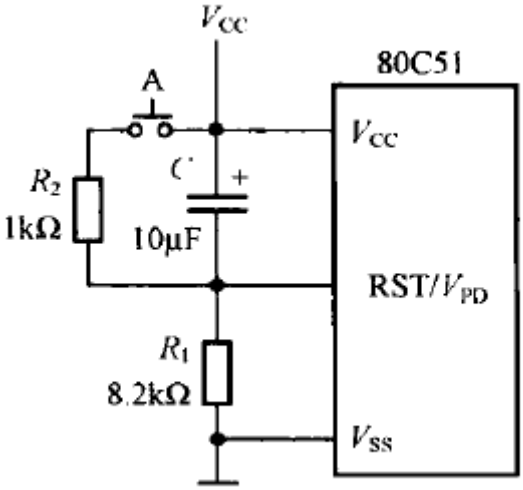
复位使得PC初始化至0000H，ALE与$\overline{PSEN}$无效，不影响RAM

三、复位方式

1.上电自动复位



2.按键手动复位



2.5.2 程序执行方式

复位后系统指针指向0000H，在0000H处放置长转移指令转移至中断服务区后面（0030H）

2.5.3 单步执行方式

一、外部电路

按钮连接$\overline{INT0}$，按钮按下高电平，放开低电平

二、中断程序

```
JNB P3.2,$  ;INT0为0时阻塞
JB  P3.2,$  ;INT0为1时阻塞
RETI        ;返回
```

2.5.4 低功耗方式

一、HMOS单片机掉电方式

主电源电压小于备用电源时，进入掉电中断，将有用数据保存至内部RAM中后由备用电源向RAM供电，时钟与CPU皆停止

主电源恢复后自动恢复现场

二、CHMOS单片机节电运行方式

由电源控制寄存器PCON（87H）控制

位序	PCON. 7	PCON. 6	PCON. 5	PCON. 4	PCON. 3	PCON. 2	PCON. 1	PCON. 0
位符号	SMOD	-	-	-	GF1	GF0	PD	IDL

PD=1，进入掉电方式

IDL=1，进入待机方式

待机方式

振荡器工作，切断时钟到CPU的通路，中断、串口、定时器正常允许

退出：激活任意中断；复位

掉电方式

振荡器停止

退出：复位

2.5.5 EPROM的编程和校验工作方式

3.1 指令格式及其符号说明

3.1.1 指令格式

[标号:] 助记符 [目的操作数],[源操作数] [;注释]

3.1.2 常用符号说明

符 号	含 义
R*n*	表示当前选定寄存器组的工作寄存器 R0～R7，*n*＝0～7
R*i*	表示作为间接寻址的寄存器，只有 R0、R1 两个，即 *i*＝0，1
A	累加器 A，ACC 则表示累加器 A 的地址
#data	表示 8 位立即数，即 00H～FFH
#data16	表示 16 位立即数，即 0000H～FFFFH
addr16	表示 16 位地址，可用于 64KB 范围内寻址，用于 LCALL 和 LJMP 指令中
addr11	表示 11 位地址，可用于 2KB 范围内寻址，用于 ACALL 和 AJMP 指令中
direct	片内 8 位 RAM 单元地址，包括特殊功能寄存器的地址或符号名称
rel	带符号的 8 位地址偏移量(－128～＋127)，用于 SJMP 和条件转移指令中
bit	位寻址区的直接寻址位，即内部 RAM(包括 SFR)中的可寻址位
(×)	表示×地址单元中的内容，或由×所指定的某寄存器的内容
((×))	由×间接寻址的单元中的内容
←	将箭头后面的内容传送到箭头前面去
$	当前指令所在地址
/	加在位地址之前，表示该位状态取反
@	间接寻址寄存器或基址寄存器的前缀

3.1.3 指令的字节

一、单字节指令

二、双字节指令

三、三字节指令

3.2 寻址方式

3.2.1 立即寻址

```
1 MOV A , #25H          ;将8位立即数25H赋值给累加器A
2 MOV DPTR , #1234H     ;将16位立即数赋值给数据指针DPTR
```

3.2.2 直接寻址

```
MOV A , 50H          ;将50H所存的数据赋值给累加器A。只能使用8位地址，因此只能寻址内部RAM
MOV A , P1           ;直接寻址是访问特殊功能寄存器的唯一方法
```

3.2.3 寄存器寻址

```
MOV A , R7           ;将寄存器R7的内容赋值给累加器A
```

3.2.4 寄存器间接寻址

```
MOV A , @R0          ;将寄存器R0的内容作为地址，把该地址内容赋值给累加器A
```

R0，R1，DPTR可用于寄存器寻址

Ri可访问内部RAM低128字节与外部RAM低256字节

DPTR访问全部64KB外部RAM

3.2.5 变址寻址

```
MOVC A , @A+DPTR     ;将累加器A与DPTR内容相加作为地址，将地址内容赋值给累加器A，@作用于A+DPTR
```

3.2.6 相对寻址

```
SJMP 50H             ;将读取当前指令后的PC与操作数相加，得到地址后跳转
```

3.2.7 位寻址

3.3 MCS-51单片机指令系统

3.3.1 数据传送类指令

一、普通传送指令

1.片内数据存储器传送指令

```
MOV A , #data        ;立即寻址
MOV A , direct       ;直接寻址
MOV A , Rn           ;寄存器寻址
MOV A , @Ri          ;寄存器间接寻址

MOV Rn , #data       ;立即寻址
MOV Rn , direct      ;直接寻址
MOV Rn , A           ;寄存器寻址

MOV direct , #data   ;立即寻址
MOV direct , direct  ;直接寻址
MOV direct , A       ;寄存器寻址
MOV direct , Rn      ;寄存器寻址
```

```
MOV direct , @Ri          ;寄存器间接寻址

MOV @Ri , #data           ;立即寻址
MOV @Ri , direct          ;直接寻址
MOV @Ri , A               ;寄存器寻址

MOV DPTR , #data16        ;立即寻址
```

2.片外数据存储器传送指令

只能通过**累加器A**与片外数据存储器数据传送，且只能通过@Ri与@DPTR**间接寻址**

```
MOVX A , @Ri              ;只能寻址低8位
MOVX @Ri , A              ;只能寻址低8位
MOVX A , @DPTR
MOVX @DPTR , A
```

3.程序存储器传送指令

```
MOVC A , @A+DPTR          ;寻址64KB
MOVC A , @A+PC            ;寻址256B
```

二、数据交换指令

1.整字节交换指令

8位数据交换

```
XCH A , Rn
XCH A , direct
XCH A , @Ri
```

2.半字节交换指令

低4位数据交换

```
XCHD A , @Ri
```

3.累加器高低半字节交换指令

```
SWAP A
```

三、堆栈操作指令

```
PUSH direct             ;SP+1,存入direct
POP direct              ;读取direct,SP-1
```

3.3.2 算术运算类指令

一、加法指令

1.不带进位

```
ADD A , #data           ;A=(A)+data
ADD A , direct          ;A=(A)+(direct)
ADD A , Rn              ;A=(A)+(Rn)
ADD A , @Ri             ;A=(A)+((Ri))
```

第7位有进位，CY=1

第3位有进位，AC=1

第6、7位有一个进位，OV=1

2.带进位

```
ADDC A , #data          ;A=(A)+data+(CY)
ADDC A , direct         ;A=(A)+(direct)+(CY)
ADDC A , Rn             ;A=(A)+(Rn)+(CY)
ADDC A , @Ri            ;A=(A)+((Ri))+(CY)
```

3.加1指令

```
INC A
INC direct
INC Rn
INC @Ri
INC DPTR
```

二、减法指令

1.带借位的减法指令

```
SUBB A , #data          ;A=(A)-data-(CY)
SUBB A , direct         ;A=(A)-(direct)-(CY)
SUBB A , Rn             ;A=(A)-(Rn)-(CY)
SUBB A , @Ri            ;A=(A)-((Ri))-(CY)
```

第7位有借位，CY=1

第3位有借位，AC=1

第6、7位有一个借位，OV=1

2.减1指令

```
DEC A
DEC direct
DEC Rn
DEC @Ri
```

三、乘法指令

```
MUL AB
```

将累加器A与寄存器B相乘，得到16位乘积，高位在A，低位在B

四、除法指令

```
DIV AB
```

被除数为A，除数为B；执行后商为A，余数为B

五、十进制调整指令

```
MOV A , #93H
ADD A , #59H
DA A
```

将A调整为BCD码

3.3.3 逻辑运算及移位类指令

一、与

```
ANL A , #data
ANL A , direct
ANL A , Rn
ANL A , @Ri
ANL direct , #data
ANL direct , A
```

将源操作数与目标操作数按位取与

二、或

```
ORL A , #data
ORL A , direct
ORL A , Rn
ORL A , @Ri
ORL direct , #data
ORL direct , A
```

将源操作数与目标操作数按位取或

三、异或

```
XRL A , #data
XRL A , direct
XRL A , Rn
XRL A , @Ri
XRL direct , #data
XRL direct , A
```

将源操作数与目标操作数按位取异或

四、累加器清零与取反

```
CLR A                ;清零
COL A                ;取反
```

五、循环移位

```
RL A                 ;累加器循环左移
RR A                 ;累加器循环右移
RLC A                ;带进位累加器循环左移
RRC A                ;带进位累加器循环左移
```

3.3.4 控制转移类指令

一、无条件转移

1.长转移指令

```
LJMP addr16          ;将16位地址送PC
```

2.绝对转移指令

```
AJMP addr11          ;PC高5位不变，赋值低11位
```

3.短转移指令

```
SJMP rel             ;PC当前值加rel，rel是补码形式有符号数
```

4.变址寻址转移指令

```
JMP @A+DPTR          ;将A与DPTR内容相加得到地址
```

二、条件转移

1.累加器判零转移指令

```
JZ rel               ;若A=0则PC=PC+2+rel
JNZ rel              ;若A≠0则PC=PC+2+rel
```

2.比较转移指令

```
CJNE A , #data , rel          ;不等时转移，若目的操作数大于等于源操作数，CY=0；目的操作数小于源操作数，CY=1
CJNE A , direct , rel         ;可使用该特性实现数值比较
CJNE Rn , #data , rel
CJNE @Ri , direct , rel
```

3.减1非零转移指令

```
DJNZ Rn , rel                ;Rn-1后不为0则跳转
DJNZ direct , rel
```

三、子程序调用及返回

1.长调用指令

```
LCALL addr16                 ;PC+3；分两次将PC压入堆栈，先压低位；将地址送入PC
```

2.绝对调用指令

```
ACALL addr11                 ;PC+2；分两次将PC压入堆栈，先压低位；将11位地址送入PC低位
```

3.子程序返回指令

```
RET                          ;自动从堆栈中取出PC
```

4.中断返回指令

```
RETI                         ;自动从堆栈中取出PC
```

四、空指令

```
NOP                          ;延迟一个机器周期
```

3.3.5 位操作类指令

一、位传送指令

```
MOV C , bit                  ;实现可寻址位与位累加器CY之间的传送
MOV bit , C
```

二、位置位与清零指令

```
SETB C                       ;置1
SETB bit
CLR C                        ;置0
CLR bit
```

三、位运算指令

```
ANL C , bit                  ;CY与bit
ANL C , /bit                 ;CY与bit非
ORL C , bit                  ;CY或bit
ORL C , /bit
CPL C                        ;C非
CPL bit
```

四、位控制转移指令

1.以C为条件

```
JC rel                          ;若CY=1则跳转
JNC rel                         ;若CY=0则跳转
```

2.以位地址内容为条件

```
JB bit , rel                    ;bit=1则跳转
JNB bit , rel                   ;bit=0则跳转
JBC bit , rel                   ;bit=1则跳转且bit清零
```

4.1 汇编语言的概述

4.1.1 汇编语言的特点

4.1.2 汇编语言的伪指令

1.ORG

确定下一条指令的地址

2.DB

保存字节常数或字符或表达式

```
DB 73H,01H,90H
DB "Hello!"
```

3.DW

保存字数据（16位）

```
DB 1234H,90H
```

4.EQU

赋值标号

```
AREA EQU 1000H
STK EQU AREA
```

5.END

源程序结束

6.DATA

指定的数据地址赋予规定的字符名称

7.DS

保留指定表达式的若干字节空间

```
ORG 1000H
DS 07H
DB 20H,25H,33H,46H          ;1007H为20H
```

8.BIT

将位地址赋予字符名

```
FLG BIT 20H                ;标志位FLG定义在位地址20H
AI BIT P1.0                ;把P1.0的位地址赋值给AI
```

4.2 汇编语言源程序的编辑和汇编

4.2.1 手工编程和汇编

4.2.2 机器编辑和交叉汇编

4.3 汇编语言程序设计

4.3.1 简单程序设计

1.BCD转十进制

片内RAM内35H单位中BCD码转换至片内RAM中40H单元中

```
        ORG 0800H       ;代码所在位置
        BCD EQU 35H     ;数据源地址
        BIN EQU 40H     ;数据目标地址
START:
        MOV A , BCD     ;BCD内容赋值A
        ANL A , #F0H    ;取高四位
        SWAP A          ;交换高低四位
        MOV B , #0AH    ;B=10
        MUL AB          ;10*原本高位，得16位乘积，高位在A
        MOV R0 , A      ;R0=A
        MOV A , BCD     ;BCD内容赋值A
        ANL A , #0FH    ;取低四位
        ADD A , R0      ;A=A+R0
        MOV BIN , A     ;BIN地址赋值A
        SJMP $          ;跳转至当前地址（死循环）
        END
```

2.BCD转ASCII

片内RAM内20H单位中BCD码转换至片内RAM中21H、22H单元中

```
        ORG 1000H
        BCD EQU 20H
        ASC EQU 21H
START:
        MOV A , BCD
        ANL A #0FH
        ADD A #30H
```

```
        MOV ASC , A
        MOV A , BCD
        ANL A #F0H
        SWAP A
        ADD A #30H
        MOV ASC+1 , A
        SJMP $
        END
```

4.3.2 分支程序设计

1.单/双分支

在片内RAM中40H处存在X，按规则将Y写入片内RAM中41H处

$$Y=\begin{cases}1, \mathrm{X}>0\\0, \mathrm{X}=0\\-1, \mathrm{X}<0\end{cases}$$

```
        ORG 1000H
START:
        MOV A 40H
        JZ DONE                 ;A=0时跳转
        JNB ACC.7 , POST        ;A[7]=0（正数）时跳转
        MOV A , #FFH            ;A=-1
        SJMP DONE
POST:
        MOV A , #01H
DONE:
        MOV 41H , A
        SJMP $
        END
```

2.多分支

P1口：被操作数、结果低8位

P3口：操作数、结果高8位

R2：0-加 1-减 2-乘 3-除

```
        ORG 1000H
START:
        MOV P1 , #DATA1
        MOV P2 , #DATA2
        MOV DPTR , #TABLE
        MOV A , R2
        RL A
        JMP @A+DPTR
TABLE:
        AJMP PRG0
        AJMP PRG1
        AJMP PRG2
        AJMP PRG3
PRG0:
        MOV A , P1          ;A=P1
        ADD A , P3          ;A=A+P3
```

```
        MOV P1 , A          ;P1=A
        CLR A
        ADDC A , #00H       ;A=A+CY
        MOV P3 , A          ;P3=A
        RET
PRG1:
        MOV A , P1
        CLR C
        SUBB A , P3
        MOV P1 , A
        CLR A
        RLC A
        MOV P3 , A
        RET
PRG2:
        MOV A , P1
        MOV B , P3
        MUL AB
        MOV P1 , A
        MOV P3 , B
        RET
PRG3:
        MOV A , P1
        MOV B , P3
        DIV AB
        MOV P1 , A
        MOV P3 , B
        RET
```

4.3.3 循环程序设计

一、单重循环

1. 片内RAM38H-47H存放16个二进制无符号数，求和存放R4、R5

```
        ORG 0800H
START:  MOV R0,#38H     ;起始地址
        MOV R2,#10H     ;计次循环
        MOV R4,#00H
        MOV R5,#00H
LOOP:   MOV A,R5        ;低位送A
        ADD A,@R0       ;A=A+R0
        MOV R5,A        ;R5=A
        CLR A           ;A=0
        ADDC A,R4       ;A=A+R4+CY
        MOV R4,A        ;R4=A
        INC R0          ;R0++
        DJNZ R2,LOOP    ;if R2!=0 goto LOOP
        SJMP $          ;阻塞
        END
```

2. 片内RAM的30H-4FH传送至片外1800H开始

```
        ORG 1000H
START:  MOV R0,#30H
        MOV DPTR,#1800H      ;访问片外RAM应使用MOVX，并使用DPTR做地址
        MOV R2,#20H
LOOP:   MOV A,@R0
        MOVX @DPTR,A
        INC R0
        INC DPTR
        DJNZ R2,LOOP
        SJMP $
        END
```

二、多重循环

设计50ms延时程序

```
        ORG 0800H
DELAY:  MOV R7,#200      ;外循环次数，1T
DLY1:   MOV R6,#123      ;内循环次数，1T
DLY2:   DJNZ R6,DLY2     ;R6不为1时循环，2T
        NOP              ;1T
        DJNZ R7,DLY1     ;R7不为1时循环，2T
        RET              ;2T
```

$f_{OSC}=12MHz$时，1T为1us

总延迟=(246+2+1+1)T*200+2T+1T=50.003ms

4.3.4 数值转换程序

一、BIN - BCD

入口：16bit无符号数R3、R2

出口：R6、R5、R4

```
BINBCD: CLR A
        MOV R4,A
        MOV R5,A
        MOV R6,A
        MOV R7,#10H          ;计次
LOOP:   CLR C
        MOV A,R2
        RLC A
        MOV R2,A
        MOV A,R3
        RLC A
        MOV R3,A
        MOV A,R4
        ADDC A,R4
        DA A
        MOV R4,A
        MOV A,R5
        ADDC A,R5
        DA A
        MOV R5,A
```

```
        MOV A,R6
        ADDC A,R6
        MOV R6,A
        DJNZ R7,LOOP
        RET
```

二、BCD - BIN

三、ASCII - BIN

```
ASCBIN: MOV A,R1            ;A=R1
        CLR C
        SUBB A,#30H         ;A-30H
        MOV R1,A            ;R1=A
        SUBB A,#0AH         ;A-10
        JC LOOP             ;IF CY=1(A<10) GOTO LOOP
        XCH A,R1            ;A=R1
        SUBB A,#07H         ;A-7
        MOV R1,A            ;R1=7
LOOP:   RET
```

4.3.5 查表程序设计

查表计算$Y = X^2$

```
        ORG 1000H
START:  MOV A,30H
        MOV DPTR,#TABLE
        MOVC A.@A+DPTR
        MOV 31H,A
TABLE:  DB 0,1,4,9,16,25,36,49,64,81
        END
```

```
        ORG 1000H
START:  MOV A,30H
        ADD A,#02H
        MOVC A.@A+PC
        MOV 31H,A
TABLE:  DB 0,1,4,9,16,25,36,49,64,81
        END
```

5.1 C51数据与运算

5.1.1 C51的数据类型

bit：位变量

sbit：可位寻址空间的一个位

sfr：特殊功能寄存器

sfr16：16位特殊功能寄存器

5.1.2 C51数据的存储类型

类型关键字	存储区	描述
data	DATA	单片机内部 RAM 空间的低 128B，可在一个周期内直接寻址。
bdata	BDATA	DATA 区中可以字节、位混合寻址的 16B 位寻址区。
idata	IDATA	RAM 区高 128B，必须采用间接寻址。
xdata	XDATA	外部存储区，地址范围 0000H～FFFFH，使用 DPTR 间接寻址。
pdata	PDATA	外部存储区的 256B，可通过 P0 口的地址对其寻址。
code	CODE	程序存储区，内容只读，使用 DPTR 寻址。

5.1.3 8051特殊功能寄存器的C51定义

1.sfr

```
sfr P0 = 0x80;      //等号后必须是常数
```

2.sfr16

对于8051派生系列单片机，可以什么两个连续地址的特殊功能寄存器

3.sbit

可位寻址的特殊功能寄存器和其他可位寻址目标

```
sfr KEYS = 0x80;
sbit KEY_UP = KEYS ^ 1;          //KEYS的第一位

sbit TF0 = 0x88 ^ 5;             //0x88的第五位

sbit TF0 = 0x8D;                 //绝对地址
```

5.1.4 8051并行接口及位变量的C51定义

```
sfr P0 = 0x80;
sfr P1 = 0x90;

sbit P0_0 = 0x80;
sbit P0_1 = 0x81;
```

5.2 C51运算符、表达式及其规则

略

5.3 C51流程控制语句

略

5.4 C51构造数据类型

5.4.1 数组

在声明时，变量不能作为数组的维数

5.4.2 指针

5.4.3 结构体

5.4.4 共用体

5.4.5 枚举

5.5 函数

5.5.1 函数的定义

5.5.2 函数的调用

5.5.3 函数的嵌套与递归

一、嵌套

每次调用嵌套都将使8051系统把2字节压入堆栈中，而C编译器依靠堆栈来进行参数传递

一个函数内应将嵌套调用的层次限制在四五层以内

二、递归

5.5.4 中断服务函数

一、中断服务函数的定义

```
返回值类型 函数名() interrupt 中断号 using 寄存器组号
{
    函数体语句
}
```

1. 中断函数不能传参
2. 中断函数无返回值
3. 不能直接调用中断函数
4. 中断函数中调用其他函数，要确保使用的寄存器相同

二、修饰符interrupt

中断号取值为0-31

0：外部中断0

1：T0

2：外部中断1

3：T1

4：串口中断

5：T2

其他值预留

三、修饰符using

指定本函数内部使用的工作寄存器组，取值为0-3

5.5.5 指向函数的指针变量

5.5.6 局部变量与全局变量

外部变量与局部变量同名时，在局部变量的作用范围内，外部变量被屏蔽

5.6 C51的库函数

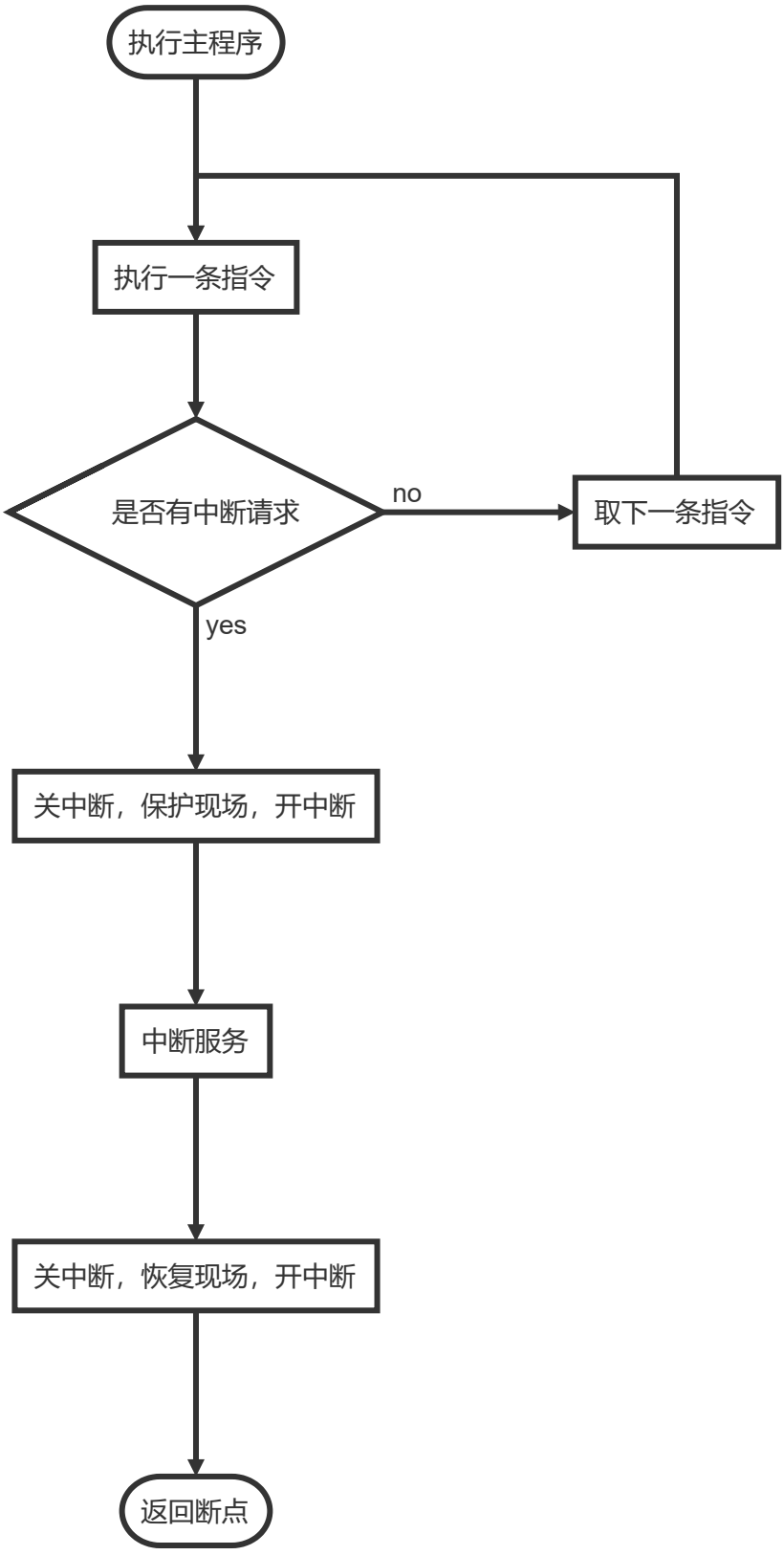
略

5.7 C51模块化程序设计

略

6.1 中断系统

6.1.1 中断概述



6.1.2 中断源

名称	符号	中断引起的原因	中断入口地址
外部中断 0	$\overline{\text{INT0}}$	P3.2 引脚的低电平或负跳变信号	0003H
定时器 0 中断	T0	定时/计数器 0 计数回零溢出	000BH
外部中断 1	$\overline{\text{INT1}}$	P3.3 引脚的低电平或负跳变信号	0013H
定时器 1 中断	T1	定时/计数器 1 计数回零溢出	001BH
串行口中断	TI/RI	串行通信完成一帧数据发送或接收	0023H

一、外部中断源

$\overline{INT0}$与$\overline{INT1}$

电平方式与脉冲方式

二、定时/计数器中断源

计数溢出时产生中断

三、串行口中断

串口完成一帧信息后将发送或接受中断标志位置1

6.1.3 中断控制

一、TCON

位地址	8FH	8DH	8BH	8AH	89H	88H
位符号	TF1	TF0	IE1	IT1	IE0	IT0

1.IE0/IE1

外部中断有效时置1，进入中断服务后自动置0

2.IT0/IT1

为0时外部中断电平触发，为1时外部中断脉冲触发

3.TF0/TF1

定时/计数器溢出时置1，进入中断服务后自动置0

二、SCON

位地址	99H	98H
位符号	TI	RI

1.TI

串口发送完一帧信息后自动置1，进入中断服务后手动置0

2.RI

串口接收完一帧信息后自动置1，进入中断服务后手动置0

三、IE

位地址	0AFH	0AEH	0ADH	0ACH	0ABH	0AAH	0A9H	0A8H
位符号	EA	-	-	ES	ET1	EX1	ET0	EX0

1.EA

中断总开关，高电平有效

2.EX0/EX1

外部中断开关，高电平有效

3.ET0/ET1

定时/计数器中断开关，高电平有效

4.ES

串口中断开关，高电平有效

四、IP

位地址	0BFH	0BEH	0BDH	0BCH	0BBH	0BAH	0B9H	0B8H
位符号	-	-	-	PS	PT1	PX1	PT0	PX0

为0时为低优先级，1时为高优先级

1.PX0/PX1

外部中断优先级设定

2.PT0/PT1

定时/计数器中断优先级设定

3.PS

串口中断优先级设定

响应原则

高优先级可打断低优先级，反之不可

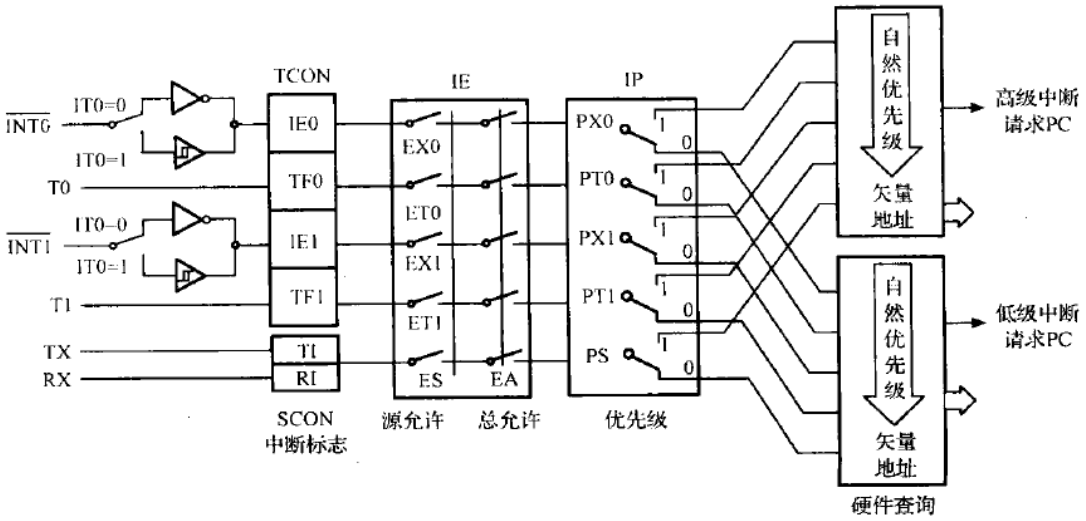
同级不能嵌套

同级同时出现响应顺序：外部0 -> 定时器0 -> 外部1 -> 定时器1 -> 串口

6.1.4 中断响应过程

一、中断采样

二、中断标志位查询



三、中断响应

硬件生成长调用指令，将PC压栈后将中断入口送入PC，中断入口地址放置无条件转移指令

四、中断响应时间

6.1.5 中断请求的撤除

一、定时/计数器中断

响应后硬件自动撤除

二、外部中断

1.脉冲方式

响应后硬件自动撤除

2.电平方式

标志位在响应后硬件自动撤除

需要手动把外部中断输入端置1

三、串行中断

在中断服务内撤除

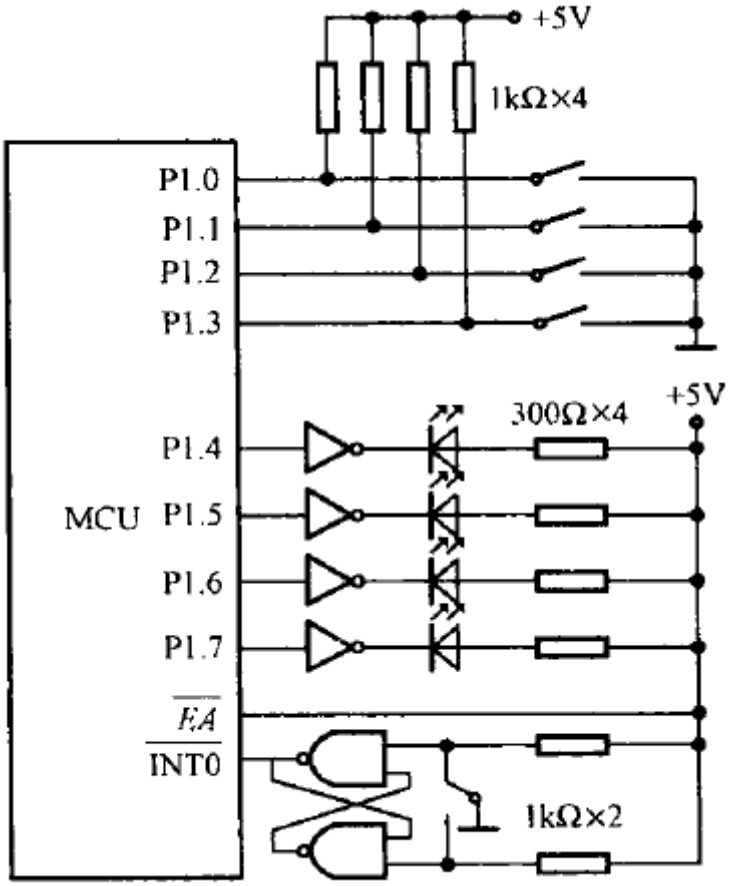
6.1.6 中断程序设计

中断初始化

1. CPU开关中断
2. 设置中断源开关
3. 设置中断优先级
4. 使用外部中断时设置电平/脉冲触发

一、汇编语言

外部中断时将开关状态反映至LED上




```
        ORG 0000H
        AJMP MAIN       ;无条件转移

        ORG 0003H
        AJMP SER        ;指向中断服务函数

        ORG 0030H
MAIN:   MOV P1,#0FH     ;输出高电平保持LED熄灭
        SETB IT0        ;外部中断脉冲触发
        SETB EX0        ;允许外部中断
        SETB EA         ;打开中断总开关
        AJMP $          ;阻塞
SER:    MOV P1,#0FH     ;输出高电平保持LED熄灭，且防止读取IO时大短路
        MOV A,P1        ;读取IO状态
        CPL A           ;取反
        ANL A,#0FH      ;屏蔽A高半字节
        SWAP A          ;交换高低半字节
        MOV P1,A        ;输出
        RETI            ;中断返回
```

二、C语言

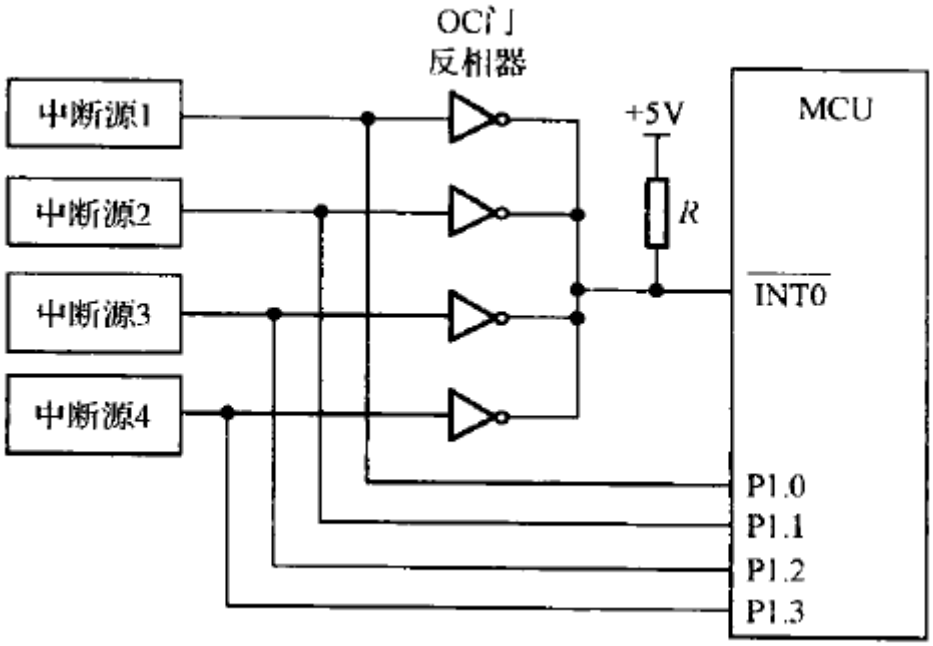
```c
#include<reg51.h>
void int0() interrupt 0{
    P1 = 0x0f;
    P1<<=4;
    ~P1
}
main(){
    EA = 1;
    EX0 = 1;
    IT0 = 1;
    while(1);
}
```

6.1.7 外部中断源的扩展

一、利用定时器扩展外部中断源

将定时器装载值设置为0FFH，外部中断接入技术输入端

二、利用硬件申请软件查询拓展外部中断源

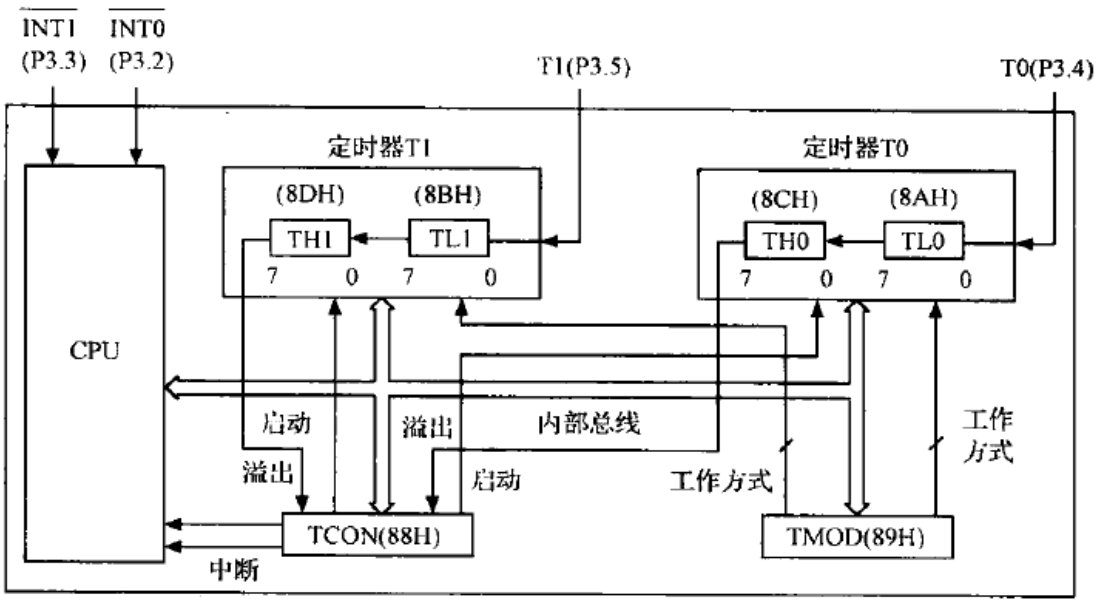


利用线或功能触发中断，读取IO判断中断源

6.2 定时/计数器

6.2.1 定时/计数器的结构及工作原理

一、结构



二、工作原理

1.用作定时器

输入脉冲为内部振荡器输出12分频后（机器周期）的信号

最长定时时间为2^16-1=65535个机器周期

2.用作计数器

最高计数频率为晶振频率的$\frac{1}{24}$

最大计数长度为2^16=65536个脉冲

6.2.2 定时/计数器的控制

一、TCON

1.TF0/TF1

溢出标志位

2.TR0/TR1

运行标志位，为0时停止工作，为1时开始工作

二、TMOD

位序	D7	D6	D5	D4	D3	D2	D1	D0
位符号	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0

低半字节控制定时器0，高半字节控制定时器1

1.GATE

GATE=0时，TR设置为1即可启动定时器

GATE=1时，TR设置为1，同时外部中断引脚为1时才可启动定时器

2.C/$\overline{T}$

为0时选择定时工作

为1时选择计数工作

3.M1和M0

M1M0	工作方式	功能说明
00	方式0	13位定时/计数器
01	方式1	16位定时/计数器
10	方式2	8位自动装载初值定时/计数器
11	方式3	T0分为两个8位定时/计数器，T1停止工作

6.2.3 定时/计数器的工作方式

6.2.4 定时/计数器的初始化

一、初始化

1. 确定工作方式，写入控制寄存器
2. 确定初值，写入寄存器
3. 开启中断，配置优先级
4. 配置寄存器启动定时/计数器
5. 溢出时执行中断服务

二、初值计算

1.计数器初值计算

$$X = 2^M - C$$

2.定时器初值计算

$$t = (2^M - X) \times \text{机器周期}$$

$$X = \frac{2^M - t}{\text{机器周期}} = 2^M - (f_{OSC}/12) \cdot t$$

t单位为us

6.2.5 定时/计数器的应用

一、定时/计数器作定时器

晶振12MHz，使用T0以方式1在P1.0输出周期为4ms的方波

```
        ORG 0000H
        LJMP MAIN
        ORG 000BH
        AJMP SERT0
        ORG 0030H
MAIN:   MOV TMOD,#01H       ;定时模式，16位
        MOV TH0,#0F8H       ;X=65536-12/12*2000=0F830H
        MOV TL0,#30H
        SETB EA             ;开总中断
        SETB ET0            ;开T0中断
        SETB TR0            ;启动T0
        SJMP $
SERT0:  MOV TH0,#0F8H
        MOV TL0,#30H
        CPL P1.0            ;翻转P1.0
        RETI
        END
```

```
#include<reg51.h>
sbit P1_0=P1^0;
void main(void){
    TMOD=0x01;
    P1_0=0;
    TH0=(65536-2000)/256;
    TL0=(65536-2000)%256;
    EA=1;
    ET0=1;
    TR0=1;
    while(1);
}
void timer0(void) interrupt 1 using 1{
    TH0=(65536-2000)/256;
    TL0=(65536-2000)%256;
    P1_0 = !P1_0;
}
```

二、定时/计数器作计数器

使用T0对P3.4脉冲计数，200个脉冲给A加一

```
        ORG 0000H
        LJMP MAIN
        ORG 00BH
        AJMP SERT0
        ORG 0030H
MAIN:   MOV TMOD,#06H
        MOV TH0,#38H
        MOV TL0,#38H
        SETB EA
        SETB ET0
        SETB TR0
```

```
        MOV A,#00H
        SJMP $
SERT0:  INC A
        RETI
        END
```

```
#include<reg51.h>
unsigned char idata *p;
void main(void){
    TMOD = 0X06;
    TH0 = 256-200;
    TL0 = 256-200;
    p=0XE0;
    *p=0;
    EA=1;
    ET0=1;
    TR0=1;
    while(1);
}
void timer0(void) interrupt 1 using 1{
    *p += 1;
}
```

三、门控位的应用

使用T0门控位测量脉宽，将结果存入内部RAM的40H、41H

```
        ORG 0000H
        LJMP MAIN

        ORG 0030H
MAIN:   MOV TMOD,#09H       ;T0计时，方式1，GATE=1
        MOV TH0,#00H
        MOV TL0,#00H
WAIT:   JB P3.2 WAIT        ;等待低电平
        SETB TR0            ;启动T0
WAIT1:  JNB P3.2 WAIT1      ;等待高电平
WAIT2:  JB P3.2 WAIT2       ;等待低电平
        CLR TR0             ;停止计数
        MOV 41H,TH1
        MOV 40H,TL1
        SJMP $
        END
```

```
#include<reg51.h>
unsigned char data *p;
void main(void){
    TMOD=0x09;
    TH0=0;
    TL0=0;
    do{}while(P3.2);
    TR0=1;
    do{}while(!P3.2);
    do{}while(P3.2);
    TR0=0;
```

```
    p=0x40;
    *p=TL0;
    p++;
    *p=TH0;
}
```

6.3 串行通信口

6.3.1 数据通信概述

一、串行与并行

二、同步与异步

三、传输方式

1.单工

单向，单根数据线

2.全双工

双向，两根数据线，可同时收发

3.半双工

双向，可选1/2根，同一时间只能收/发

四、信号传输

1.连接线

近距离：TX+RX

15m左右：RS-232

远程通信：调制解调

2.接口

异步通信：UART

同步通信：USRT

同步/异步通信：USART

6.3.2 单片机的串行通信接口

MCS-51配备可编程全双工UART

一、控制寄存器

1.SCON

位地址	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
位符号	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0/SM1

SM0 SM1	工作方式	功能	波特率
00	方式0	8位同步移位寄存器	$f_{OSC}/12$
01	方式1	10位异步收发	可变
10	方式2	11位异步收发	$f_{OSC}/32$或$f_{OSC}/64$
11	方式3	11位异步收发	可变

SM2

串口工作在方式2/3时：

SM2=1：收到第9位数据为1，前8位送入SBUF，产生中断（地址帧）；收到第9位数据为0，抛弃前8位，不产生中断（数据帧）

SM2=0：前八位送入SBUF，产生中断

REN

为1时允许接收

TB8

串口工作在方式2/3时，存放第9位被发送数据

RB8

串口工作在方式2/3时，存放第9位接收到的数据

串口工作在方式1时，当SM2=0时，存放已接收的停止位

TI

发送中断标志位，软件清0

RI

接收中断标志位，软件清0

2.PCON

位序	D7	D6	D5	D4	D3	D2	D1	D0
位符号	SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD

为1时波特率倍增

GF1/GF0

通用标志位

PD

CHMOS器件低功耗控制位，为0时正常工作，为1时掉电工作

IDL

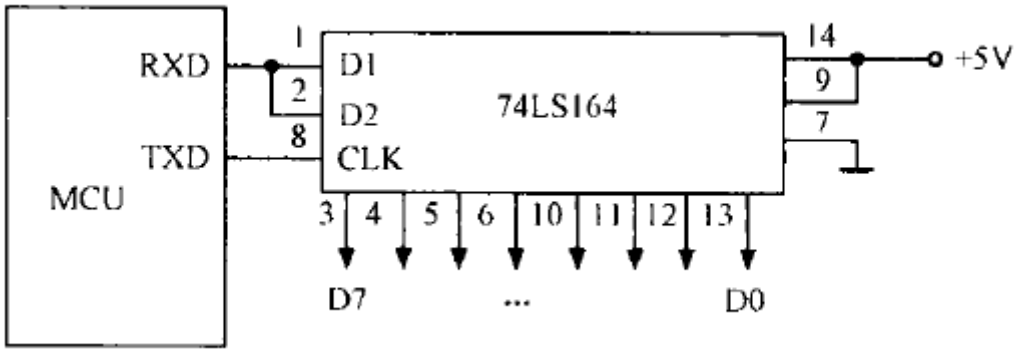
为0时正常工作，为1时空闲工作

6.3.3 串行通信的工作方式及波特率设置

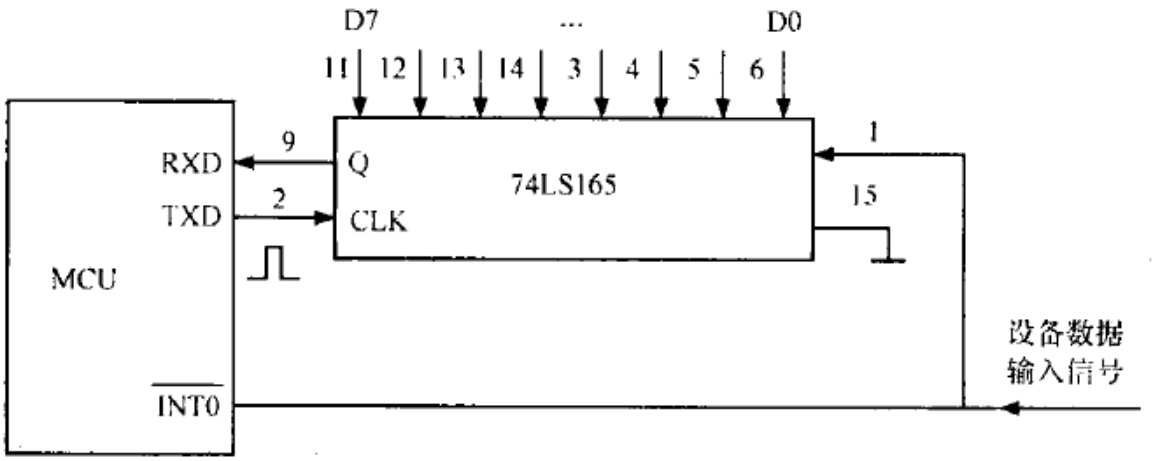
一、工作方式0

RXD作为输入输出，TXD作为时钟

1.输出



2.输入



二、工作方式1

TXD发送，RXD接收

1.输出

硬件自动加入起始位和停止位，一帧10位，8位数据

2.输入

接收器以16倍波特率采样RXD电平

以T1作为波特率发生器，其溢出脉冲作为移位脉冲，更改装载值就可设置波特率

$$f_b = \frac{2^{SMOD}}{32} \times \frac{f_{OSC}}{12\times(256-X)}$$

三、工作方式2/3

TXD发送，RXD接收

1个起始位，1个停止位，1个附加位，8个数据位

1.输出

先设置TB8作为附加位，启动串口发送，完成后置中断，需软件清0

2.输入

接收器以16倍波特率采样RXD电平

方式2：

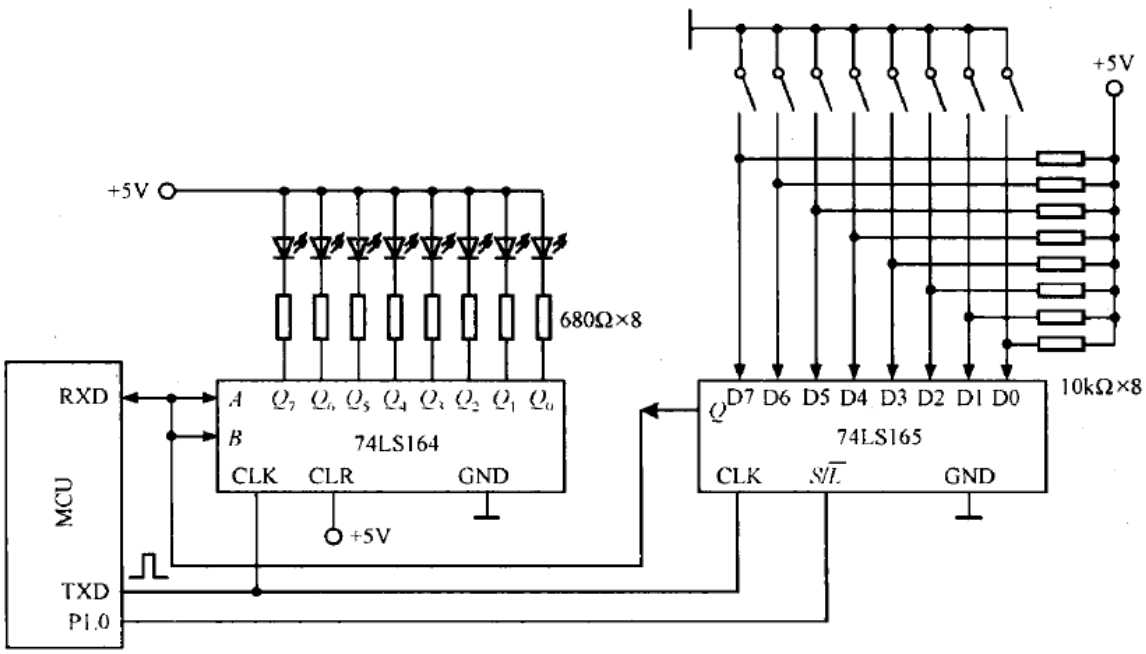
$$f_b = \frac{2^{SMOD}}{64} \times f_{OSC}$$

方式3：

$$f_b = \frac{2^{SMOD}}{32} \times \frac{f_{OSC}}{12\times(256-X)}$$

6.3.4 串行口应用

一、拓展并行IO



```
        ORG 0000H
        LJMP MAIN
        ORG 0030H
MAIN:   MOV SCON #10H       ;REN=1 RI=0 SM0=0 SM1=0
LOOP:   CLR P1.0            ;74165并行读入
        SETB P1.0           ;74165串行位移
        CLR RI              ;中断标志位清零
        JNB RI,$            ;等待中断
        MOV A,SBUF          ;接收完读入A
        CLR TI              ;中断清零
        MOV SBUF,A          ;输出A
        JNB TI,$            ;等待中断
        SJMP LOOP
        END
```

二、双机通信

甲乙以方式1进行串口通信，波特率1200bps，晶振11.0592MHz

甲机：发送外部RAM以ADDRA为首地址共128B单元

乙机：顺序存放在以ADDRB为首地址的外部RAM

```
        ORG 0000H
        LJMP MAINA
        ORG 0023H
        AJMP SERT1A
        ORG 0030H
MAIN:   MOV SP,#60H         ;设置堆栈指针
        MOV SCON,#40H       ;方式1
        MOV TMOD,#20H       ;方式2 自动装载
        MOV TL1,#0E8H       ;初值
        MOV TH1,#0E8H       ;装载值
        MOV PCON,#00H       ;波特率不倍增
        SETB TR1            ;启动定时器
        SSETB EA            ;中断允许
        SETB ES             ;串口中断
        MOV DPTR,#ADDRA     ;设置起始地址
        MOVX A,@DPTR        ;读外部数据
        MOV SBUF,A          ;发送
        SJMP $
SERT1A: CLR T1              ;清中断
        CJNE R0,#7FH,LOOPA  ;没发完128B则进入LOOPA
        CLR ES              ;关闭串口中断
        AJMP ENDA           ;结束
LOOPA:  INC R0              ;计次
        INC DPTR            ;地址后移
        MOVX A,@DPTR        ;读外部数据
        MOV SBUF,A          ;发送
ENDA:   RETI
        END
```

```
        ORG 0000H
        LJMP MAINB
        ORG 0023H
        AJMP SERT1B
        ORG 0030H
```

```
MAIN:   MOV SP,#60H          ;设置堆栈指针
        MOV SCON,#50H        ;方式1
        MOV TMOD,#20H        ;方式2 自动装载
        MOV TL1,#0E8H        ;初值
        MOV TH1,#0E8H        ;装载值
        MOV PCON,#00H        ;波特率不倍增
        SETB TR1             ;启动定时器
        SSETB EA             ;中断允许
        SETB ES              ;串口中断
        MOV DPTR,#ADDRB      ;缓冲区首地址送DPTR
        MOV R0,#00H
        SJMP $
SERT1B: CLR RI               ;清中断
        MOV A,SBUF           ;读缓冲区
        MOVX @DPTR,A         ;写外部数据
        CJNE R0,#7FH,LOOPB   ;判断是否够128B
        CLR ES               ;关中断
        LJMP ENDB
LOOPB:  INC R0
        INC DPTR
ENDB:   RETI
        END
```

三、多机通信