微机原理复习资料

summarized by wy.

第一章：微处理器技术简介

1. 8086 微处理器

- **字长 (Word Length):** 8086 CPU 内部的运算器、寄存器都是 16 位的，所以它是一款 16 位微处理器。一个字 (Word) 等于 16 个二进制位 (bit)，即 2 个字节 (Byte)。
- **数据线 (Data Bus):** 16 条 (D0–D15)。这意味着它一次可以传输 16 位数据。
- **地址线 (Address Bus):** 20 条 (A0–A19)。
- D0～D15 与 A0～A15 是分时复用的 ALE（先用作地址线后用作数据线）
- **可寻址空间 (Addressable Space):** 拥有 20 条地址线，其寻址能力为 2 的 20 次方字节 = 1,048,576 字节 = 1MB。

2. 8086 的逻辑组成

8086 CPU 在逻辑上分为两个独立工作的单元，以提高指令执行效率：

- **总线接口单元 (Bus Interface Unit, BIU):** 负责与存储器和 I/O 设备进行数据传输。它包含了段寄存器、指令指针(IP)、地址加法器和指令队列。它的主要任务是取指令、读写操作数。
- **执行单元 (Execution Unit, EU):** 负责执行从 BIU 的指令队列中取出的指令。它包含通用寄存器、算术逻辑单元(ALU)和标志寄存器(FLAGS)。它的主要功能是完成指令的译码和执行工作

3. 分段存储管理

- **基本思想:** 8086 的 1MB 内存空间被划分为若干个逻辑段，每个段最大为 64KB。CPU 通过**段基址**和**偏移地址**共同来寻址。
- **逻辑地址 (Logical Address):** 由“段地址:偏移地址”构成。例如，1000H:0020H。这

是程序员在编程时使用的地址。

- **物理地址 (Physical Address):** CPU 将逻辑地址转换为 20 位的物理地址用于实际寻址。（内存中的唯一地址）
  - **计算公式:** 物理地址 = 段地址 × 16 + 偏移地址。
  - 例如，逻辑地址 1000H:0020H 对应的物理地址是 1000H * 10H + 0020H = 10000H + 0020H = 10020H。

4. 总线周期

一个基本的总线周期是 CPU 与存储器或 I/O 端口完成一次数据交换所需的时间。对于 8086，一个最基本的总线周期由 **4 个时钟周期** 组成，分别称为 T1, T2, T3, T4。

在 8086 的数据、地址复用引脚上，T1 出现的是地址信号，T2～T3 出现的是数据信号。

第二章：单片机基本结构 (以 51 单片机为例)

1. 51 单片机 CPU 的字长

51 单片机是一款 **8 位** 单片机，其数据总线宽度为 8 位，CPU 一次能处理 8 位数据。

2. 周期概念

- **时钟周期 (Clock Cycle):** 也称为振荡周期，是单片机时序的最小单位，等于晶振频率的倒数 (P)。例如，12MHz 晶振的时钟周期是 1/12μs。
- **状态周期 (State Cycle):** 时钟周期经 2 分频后成为内部的时钟信号。每个状态周期包含 2 个时钟周期（2P）（P1，P2）。
- **机器周期 (Machine Cycle):** 完成一个基本操作所需要的时间称为机器周期。一个机器周期包含 6 个状态周期，即 12 个时钟周期 (12P)。（S1P1，S1P2....）
- **指令周期 (Instruction Cycle):** 执行一条指令所需要的所有时间。它由一个到多个机器周期组成，具体取决于指令的复杂程度。

3. I/O 引脚 (P0–P3)

- **P0 口:第一功能：**8 位漏极开路型的双向 I/O 口，可用作用户数据总线

第二功能：在访问外部存储器时，提供低 8 位地址和 8 位双向数据总线（先地址后数据）

- **P1 口:** 内部有上拉电阻的准双向 I/O 口。
- **P2 口: 第一功能：**内部有上拉电阻的 8 位准双向 I/O 口。**第二功能：**在访问外部存储器时，输出高 8 位地址
- **P3 口: 第一功能：**内部有上拉电阻的 8 位准双向 I/O 口。**第二功能：**串口、外部中断、计数脉冲输入端、读写外部数据存储器/IO 口控制端。
- 注：（1）如果单片机内部有程序存贮器，不需要扩展外部存贮器和 I/O 接口，单片机的 4 个口均可作为 I/O 口使用；

  （2）4 个口在作为输入口使用时，均应先对其写“1”，以避免误读；

  （3）P0 口作为 I/O 口使用时应外接 10kΩ的上拉电阻，其他口则可不必；

  （4）P2 口某几根口线作地址使用时，剩下的口线不能作为 I/O 口线使用；

  （5）P3 口的某些口线作第二功能时，剩下的口线可以单独作为 I/O 口线使用。

4. 程序状态字 (PSW) 寄存器（program state word）

PSW 是一个 8 位寄存器，用于存放程序运行时的状态信息。

- **D7：CY (Carry):** 进位标志位。最高位是否有进位或借位
- **D6：AC (Auxiliary Carry):** 辅助进位（半进位）标志位，低四位向高四位产生的进位或借位，用于 BCD 码运算。
- **D5：F0:** 用户可自定义的标志位。
- **D4，D3：RS1, RS0 (Register Select):** 工作寄存器组选择位。
  - 00: 第 0 组 (地址 00H–07H)
  - 01: 第 1 组 (地址 08H–0FH)

    - 10: 第 2 组 (地址 10H–17H)
    - 11: 第 3 组 (地址 18H–1FH)
- **D2：OV (Overflow):** 溢出标志位，用于有符号数运算：判断结果是否正确：1 出错，0 正确。其值为最高位进位和次高位进位的异或运算。
- **D1：F1:** 用户可自定义的标志位。
- **D0：P (Parity):** 奇偶校验位。

该位始终跟踪累加器 A 中含“1”个数的奇偶性，如果 A 中有奇数个“1”，则 P 置“1”，否则置“0”。

用途：串行通讯中的数据校验，判断是否存在传输错误。

5. 复位后的初始值

- **PC (Program Counter):** 0000H。单片机复位后，从程序存储器的 0000H 单元开始执行指令。
- **SP (Stack Pointer):** 07H。堆栈指针指向 07H 单元，这意味着第一个入栈的数据将存放在 08H 单元。

6. 存储器结构

- **哈佛结构 (Harvard Architecture):** 51 单片机采用程序存储器和数据存储器地址空间独立的哈佛结构。
- **程序存储器 (Program Memory):** 片内外统一编址的 64KB 程序存储器（用 16 位地址）特殊地址（8n+3）
- **数据存储器 (Data Memory):**
    - **片内数据存储区:** 共 256 字节。（用 8 位地址）
        - **低 128 字节 (00H~7FH):** RAM 区，包括工作寄存器区、位寻址区和用户 RAM 区。

        - **高 128 字节 (80H~FFH):** 特殊功能寄存器 (SFR) 区。
    - **片外数据存储区:** 最大可扩展 64KB (地址范围 0000H~FFFFH)。（用 16 位地址）

7. 位寻址区

- 片内 RAM 的 **20H 到 2FH** 这 16 个字节，共 128 位，可以进行位寻址。
- 一些特殊功能寄存器 (SFR) 也可以进行位寻址（地址能够被 8 整除）。

8. 最小系统

- **组成部分:** 单片机芯片、时钟电路、复位电路。
- **时钟电路:** 通常由一个晶振和两个电容组成，为单片机提供稳定的工作时钟。
- **复位电路:**
    - **上电复位:** 利用电容充电实现。
    - **按键手动复位:** 提供一个按键，按下时使 RST 引脚出现高电平。
- **复位条件:** RST 引脚上必须维持 **至少 2 个机器周期** 的高电平才能有效复位。

第三章：汇编指令系统

1. 7 种寻址方式

1. **立即寻址:** 操作数直接在指令中。MOV A, #10H　　即 10H→A
2. **直接寻址:** 指令中直接给出操作数的地址。MOV A, 30H　　即(30H)→A
3. **寄存器寻址:** 操作数在寄存器中。MOV A, R0　　即(R0)→A
4. **寄存器间接寻址:** 以寄存器中的内容作为操作数的地址。MOV A, @R0 即((R0))→A
5. **基址变址寻址:** 以 DPTR 或 PC 的内容为基址，A 的内容为位移量，相加后形成地址。MOVC A, @A+DPTR　　即((A+DPTR))→A
6. **相对寻址:** 以 PC 值加上一个 8 位位移量形成跳转地址。用于转移指令。SJMP REL
7. **位寻址:** 直接对可位寻址的位进行操作。SETB 20H

注意：

（1）对程序存储器只能采用立即寻址和基址加变址寻址方式；

（2）对特殊功能寄存器只能采用直接寻址方式，不能采用寄存器间接寻址；

（3）对 8052 等单片机内部 RAM 的高 128 个字节（80H~FFH），只能采用寄存器间接寻址，不能使用直接寻址方式；

（4）对位操作指令只能对位寻址区操作；

（5）外部扩展的数据存储器只能用 MOVX 指令访问；

（6）内部 RAM 的低 128 个字节（00H~7FH）既能用直接寻址，也能用间接寻址。

寻 址 方 式	存储器空间
立即寻址	程序存储器
直接寻址	片内 RAM 低 128 字节、SFR
寄存器寻址	工作寄存器 R0~R7、A、B、DPTR
寄存器间接寻址	片内 RAM：@R0，@R1，SP。片外 RAM：@R0，@R1，@DPTR
基址加变址寻址	程序存储器：@A+PC，@A+DPTR
相对寻址	程序存储器 256 字节范围内：PC+偏移量
位寻址	片内 RAM 的位寻址区（20H~2FH 字节地址）、某些可位寻址的 SFR

2. 关键汇编指令

- **伪指令:**
  - ORG (Originate): 定位指令，告诉汇编程序下一条指令的起始地址。ORG 0100H
  - EQU (Equate): 等值指令，用一个符号名来代替一个数值。COUNT EQU 30H
  - DB (Define Byte): 定义字节数据。 DATA DB 10H, 20H
- **堆栈操作:**
  - PUSH direct: 将指定地址单元的内容压入堆栈，SP+1。（压栈）
  - POP direct: 将栈顶内容弹出到指定地址单元，SP–1。 （出栈）
- **算术运算:** ADD（加法），ADDC（带进位加法），SUBB（带借位减法），INC（加 1），DEC（减 1），MUL（乘法）（高位积存 B，低位积存 A），DIV（除法）（商

- A 余 B)
- **逻辑运算:** ANL (与), ORL (或), XRL (异或), CPL (取反), CLR (清零)
- **移位指令:** RL (左移), RLC (带进位左移), RR (右移), RRC (带进位右移), SWAP A (高低半字节交换)
- **控制转移指令:** LJMP（长转移）, AJMP（绝对转移）, SJMP (短跳转), JZ (结果为零则跳转), JNZ (不为零则跳转), CJNE (比较不相等则跳转), DJNZ (减 1 不为零则跳转), LCALL (长调用), ACALL (绝对调用), RET (子程序返回), RETI (中断返回)

第四章：C51 语言程序设计

1. 数据类型

数据类型 (Data Type)	长度 (字节) (Length (Bytes))	位数 (Bits)	取值范围 (Range)
signed char	1	8	-128 ~ 127
char	1	8	-128 ~ 127
unsigned char	1	8	0 ~ 255
(signed) short	2	16	-32768 ~ 32767
(signed) int	2	16	-32768 ~ 32767
unsigned short int	2	16	0 ~ 65535
unsigned int	2	16	0 ~ 65535
(signed) long	4	32	-2147483648 ~ 2147483647
long	4	32	-2147483648 ~ 2147483647
unsigned long int	4	32	0 ~ 4294967295
unsigned long	4	32	0 ~ 4294967295
float	4	32	3.4e-38 ~ 3.4e38
double	8	64	1.7e-308 ~ 1.7e308
sbit	(1 位)	1	0 或 1，用于定义特殊功能寄存器的位

sfr	1	8	0 ~ 255，用于定义特殊功能寄存器

2. 常用语句

熟悉标准 C 语言的各种语句，如 if-else, for, while, do-while, switch-case 等在 C51 中的应用。

中断函数：void 函数名( ) interrupt n using m

第五章：人机接口技术

1. 独立按键

- **按键消抖:** 由于机械触点的弹性，按键按下和释放时会产生一连串的抖动。程序中通常通过延时来消除抖动。即检测到按键按下后，延时 10-20ms，再次确认按键是否仍处于按下状态。
- **避免重复处理:** 为了防止长按按键导致程序连续执行多次，可以设置一个标志位（如 static 静态变量）。当检测到按键按下并处理后，设置标志位，直到检测到按键释放，才清除标志位，允许下一次处理。

```
uchar key_scan(void)
{
  static kp=0;
  if((P1&0x03)!=0x03)//清除 P1 口高 6 位，防止其对按键值造成影响
    {
       delayms(10);
       if(((P1&0x03)!=0x03)&&(kp==0))
                {
                  kp=1;
                  if((P1&0x03)= =0x02)return 1;//S1 按下
```

```
                if((P1&0x03)= =0X01)return 2;//S2 按下
            }
    }
    else kp=0;
    return  0;    //无键按下，返回无效代码
}
```

2. 矩阵按键

- **扫描方法:**
  - **逐行扫描法:**

    （1）判有无键按下。将列线设置为输出口，输出全 0（所有列线为低电平），然后读行线状态，若行线状态不全为高电平，则可断定有键按下。

    （2）判按下哪个键。先置列线 C0 为低电平，其余列线为高电平，读行线状态，如行线状态不全为“1”，则说明所按键在该列；否则所按键不在该列，再使 C1 列线为低电平，其他列为高电平，判断 C1 列有无按键按下。

    （3）获得相应键号。键号：键号 = 行首号+列号。行首号为列数乘以行号。根据键号就可以进入相应的键功能实现程序。

```
uchar code colcode[4]={0xfe, 0xfd, 0xfb, 0xf7};//逐一激活列线
uchar key_scan(void)
{  //先初步判断有无键按下——>确定行
    uchar temp, row, column, i;
    P1=0XF0;
    temp=P1&0XF0;
    if(temp!=0xf0)
    {
        delayms(10);
```

```
        temp=P1&0XF0;
        if(temp!=0xf0)
        {
            switch(temp)
            {
                case 0x70: row=3; break;
                case 0xb0: row=2; break;
                case 0xd0: row=1; break;
                case 0xe0: row=0; break;
                default: break;
            }
            for(i=0; i<4; i++)
            {       //确定列
                P1=colcode[i];
                temp=P1&0XF0;
                temp=~temp;
                if((temp&0x0f)column=i; //此处原文如此，可能有语法错误
            }
            return row*4+column;     //计算键值
        }
    }
    else P1=0XF0;
    return 16;
}
```

○ **线反转法:** 原理类似，依次将列线置低电平，检测行线。

```
uchar code keyvalue[16]={ 0xee, 0xeb, 0xe7, 0xde, 0xdd, 0xdb, 0xd7, 0xbe, 0xbd, 0xbb,
```

```
0xb7, 0x7e, 0x7d, 0x7b, 0x77}; //键号
uchar key_scan(void)
{
   uchar scan1, scan2, temp, i;
   //读行线状态
   P1=0XF0;
   scan1=P1&0XF0;
   if(scan1!=0xf0)
   {
      delayms(10);
      scan1=P1&0XF0;
      if(scan1!=0xf0)
      {
       //读列线状态
         P1=0x0f;
         scan2=P1&0x0f;
         temp=scan1|scan2;       //拼在一起，找对应的键值
         for(i=0; i<16; i++)
         {
            if(temp==keyvalue[i])
            return i;
         }
         return 16; //返回无效代码
      }
   }
   else return 16; //无键按下，返回无效代码
}
```

- **电路连接:** 行线和列线分别连接到单片机的两个 I/O 口。例如，4x4 矩阵键盘需要 8 个 I/O 口。

3. LED 数码管

- **共阴/共阳:**
  - **共阴极 (Common Cathode):** 所有 LED 的阴极连接在一起接地。段选线送高电平(1)点亮。
  - **共阳极 (Common Anode):** 所有 LED 的阳极连接在一起接电源。段选线送低电平(0)点亮。
- **字形码:** 控制数码管显示特定字符的 8 位二进制代码。例如，显示'0'的共阳字形码通

常是 C0H。

- **显示方式:**
  - **静态显示:** 每个数码管都由一个 I/O 口独立控制，占用 I/O 口多，但显示亮度高，编程简单。
  - **动态显示:** 所有数码管的段选线并联，由一组 I/O 口控制；位选线由另一组 I/O 口控制。通过分时轮流点亮各个数码管，利用人眼视觉暂留效应实现多位显示。优点是节省 I/O 口。

4. 液晶显示 (LCD1602)

- **接口连接:** 需要数据线(DB0–DB7)，控制线(RS, R/W, E)。

表5-2 LCD1602液晶显示模块引脚的功能说明

引脚编号	名称	引脚功能说明
1	Vss	接地引脚（GND）。
2	Vdd	电源引脚（接+5V）。
3	VO	液晶显示驱动电源（0~5V），可接电位器。
4	RS	数据和指令选择控制端，RS=0：命令/状态；RS=1：数据。
5	$R/\overline{W}$	读写控制线，$R/\overline{W}$ =0:写操作；$R/\overline{W}$ =1:读操作
6	E	数据读写操作控制位，E线向LCD模块发送一个脉冲，LCD模块与单片机之间将进行一次数据交换。
7~14	DB0~DB7	数据线，可以用8位连接，也可以只用高4位连接。
15	A	背光控制正电源。
16	K	背光控制地。

- **核心函数:**
  - **写指令 LcdWriteCmd(cmd):** 设置 RS=0, R/W=0，然后将指令码送到数据线，并使能 E 引脚产生一个高脉冲。用于设置显示模式、清屏、设置光标位置等。

```
void w_com(uchar com)
{
   RS=0;
   RW=0;
   E=1;
   P0=com;
   E=0;
   delayms(1);
}
```

此函数形参为写入的命令，无返回值。有两个功能：其一，写入指令代码；其二，确定 LCD 面板上的写入位置，显示命令的最高位为 1 ，所以设置显示地址液晶屏第一行的地址为 0x80 ~ 0x8f，第二行的地址为 0xc0 ~ 0xcf。
例如，执行清屏命令为“w_com （ 0x01 ）;”，定位在第一行第二个字符位显示的语句为 “ w_com （ 0x81 ）;”。

- **写数据 LcdWriteData(dat):** 设置 RS=1, R/W=0，然后将数据送到数据线，并使能 E 引脚产生一个高脉冲。用于在当前光标位置显示一个字符。

```
void w_dat(uchar dat)
{
   RS=1;
   RW=0;
   E=1;
   P0=dat;
   E=0;
   delayms(1);
}
```

dat 代表要显示的字符的 ascii 值 例如要显示 a：void w_dat(‘a’)或 void w_dat(97)

- **初始化 LcdInit():** 一系列写指令操作，用于设置 LCD 的工作模式。

```
void lcd_ini( void )
{
   delayms(10);
   w_com(0x38); //功能设置：8 位口，2 行，5*7 点阵
   delayms(10);
   w_com(0x0c); //显示设置：开显示，关光标，无闪烁
   delayms(10);
   w_com(0x06); //输入模式：右移一格，地址加 1
   delayms(10);
   w_com(0x01); //清显示
   delayms(10);
}
```

- **多位数值显示:** 需要将一个多位数（如 123）分离成单个数字字符 '1', '2', '3'，然后逐个发送给 LCD 显示。例如，123 / 100 = 1, 123 % 100 / 10 = 2, 123 % 10 = 3。

```
w_dat(num/10+0x30);          // 将 num 的十位数转换为 ASCII 码并显示
w_dat(num%10+0x30);          // 将 num 的个位数转换为 ASCII 码并显示
```

5. 蜂鸣器

- **有源 vs 无源:**
  - **有源蜂鸣器:** 内部自带振荡电路。你只需要给它提供直流电源(高电平或低电平，取决于其触发方式)它就能发出固定频率的声音。控制简单，但频率固定不可调。
  - **无源蜂鸣器:** 内部没有振荡源，需要外部提供一定频率的脉冲信号才能发声。可以通过改变脉冲频率来改变音调。

有源蜂鸣器：（直接给引脚设置高低电平就行）

```
void main(void)
{
    while(1)
    {
        BELL=0; //发声
        delayms(1000);
        BELL=1; //不发声
        delayms(1000);
    }
}
```

无源蜂鸣器：（周期性的翻转电平，在代码中产生方波）

```
void main(void)
{
    uint i;
    while(1)
    {
        for(i=0;i<1000;i++) //500Hz 响 1s
        {
            BELL = ~BELL;
            delayms(1);
```

```
        }
        for(i=0;i<1000;i++) //不发声 1s
        {
            BELL=1;
            delayms(1);
        }
    }
}
```

- **驱动电路:** 无源蜂鸣器需要较大的驱动电流，通常用 NPN 三极管来驱动。
- **程序实现:** 通过定时器中断，在中断服务程序中反转控制蜂鸣器的 I/O 口电平，即可产生特定频率的方波。通过改变定时器的重装载值来改变频率（音调），通过控制定时器的开关来控制发声的启停和间歇。

第六章：中断

1. 中断概念

- **定义:** CPU 在执行主程序时，由于内部或外部的紧急事件请求，暂停当前程序，转而去执行一个为该事件准备的服务程序，执行完毕后再返回原程序断点处继续执行的过程。
- **功能:** 实现并行处理、处理突发事件、提高 CPU 效率。

2. 51 单片机中断源

中断源	类型号	查询优先级	入口地址
外部中断 0 (INT0)	0	1	0003H
定时器 0 (T0)	1	2	000BH
外部中断 1 (INT1)	2	3	0013H
定时器 1 (T1)	3	4	001BH

串行口 (UART)	4	5	0023H

3. 外部中断引脚

- **INT0:** P3.2
- **INT1:** P3.3

4. 外部中断触发方式

在 TCON 寄存器中设置：

- **IT0 (TCON.0):** 1=下降沿触发, 0=低电平触发 (INT0)。
- **IT1 (TCON.2):** 1=下降沿触发, 0=低电平触发 (INT1)。

5. 中断响应时间

从中断请求被检测到，到 CPU 开始执行中断服务程序的第一条指令，所需的时间。

- **最短:** 3 个机器周期。
- **最长:** 8 个机器周期。

6. 中断编程

- **初始化:**
  1. **总中断允许:** EA = 1;
  2. **允许各分中断:** 如 EX0 = 1; (允许外部中断 0)EX1,ET0,ET1,ES
  3. **选择触发方式 (对外部中断):** 设置 TCON 寄存器，如 IT0 = 1（边沿触发）
     IT1=1(电平触发)
  4. **设置优先级 (可选):** 设置 IP 寄存器。IP=1：高级；IP=0：低级
- **中断服务函数 (ISR) 定义:**

```
void int0( void ) interrupt n using m {
    // 中断处理代码
}
```

“interrupt”关键字告诉编译器这是一个中断函数，后面的数字是中断类型号。

例：

```
void int0( void ) interrupt 0 using 0
{
   EX0=0;//先禁止外部中断 0 中断，避免重复触发


    P1=0x0f;
    delayms(800);
    P1=0xf0;
    delayms(800);
    EX0=1;//中断返回前,打开外部中断 0 中断
}
```

第七章：定时/计数器

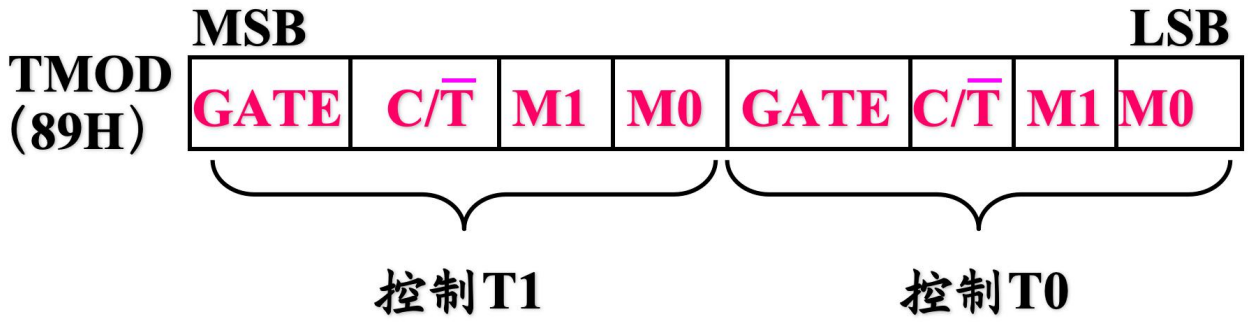
1. 定时 vs 计数

- **联系:** 本质都是对脉冲进行计数。
- **区别:**
  - **定时模式:** 计数脉冲来自单片机内部的系统时钟（通常是 fosc/12）。对机器周期进行计数。用于实现精确定时。
  - **计数模式:** 计数脉冲来自外部输入引脚 **T0 (P3.4)** 或 **T1 (P3.5)**。用于对外部脉冲的计数。
- **外部计数脉冲最高频率:** fosc/24。

2. 工作方式

由 TMOD 寄存器设置，T0 和 T1 各有 4 种方式。

TMOD 不可位寻址



GATE：是否受外部中断的影响

M1 M0

- **00:方式 0:** 13 位定时/计数器。（TH0 是高 8 位 TL0 低 5 位）
- **01:方式 1:** 16 位定时/计数器。
- **10:方式 2:** 自动重装载初值的 8 位计数器。TH0（重装载初值 8 位）、TL0（8 位计数器）
- **11:方式 3:** T0 可拆分为两个独立的 8 位计数器（TL0，TH0）。T1 停止计数

3. 初值计算（以方式 1 为例）

1. 计数器计算初值：X=2^n-N
（X：初值，n 是计数器的位数：与工作方式有关
（方式 0:13，方式 1:16，方式 2:8，方式 3:8）。N 是计满为 0 的所需计数值）
2. 定时器计算初值：T=(2^n-X)*Tp——>X=2^n-T/Tp
（X：初值，T：定时的时间，n：定时器的位数，Tp 是机器周期）
3. THx = (65536 – X) / 256;
4. TLx = (65536 – x) % 256;

4. 定时器中断编程

- **初始化:**
  1. **开总中断:** EA = 1;
  2. **开定时器中断:** ET0 = 1; 或 ET1 = 1;
  3. **确定工作方式:** 配置 TMOD 寄存器。
  4. **装载计数初值:** 设置 THx 和 TLx。

5. **启动定时器:** TR0 = 1; 或 TR1 = 1;

- **中断服务函数:**

```
void Timer0_Routine(void) interrupt 1 {
    // 重装初值 (方式 1 需要手动重装)
    TH0 = ...;
    TL0 = ...;
    // 其他处理代码
}
```

第八章：通信接口设计

1. 异步串行通信

- **特点:** 不需要同步时钟线，收发双方靠约定好的波特率进行通信。以**字符帧**为单位传输，每一帧包含起始位、数据位、可选的校验位和停止位。

2. 波特率 (Baud Rate)

- **概念:** 每秒传输的二进制码元个数，单位是 bps (bits per second)。收发双方必须设置相同的波特率。

3. 串口初始化

- **SCON (Serial Control) 寄存器:** 设置串口工作方式 (SM0, SM1)、允许接收 (REN=1 允许，REN=0 禁止) 等，SM2 允许多机通信
- SM0，SM1，SM2，REN，TB8（发送的第 9 位），RB8（接受的第 9 位），TI（表示一帧数据发送完毕），RI（表示一帧接收完毕）且 TI 和 RI 必须由软件清零
- **PCON (Power Control) 寄存器:** SMOD 位可以使波特率加倍。

5. **串口工作方式（SM0，SM1）**

- **00:方式 0:** 同步移位寄存器方式，用于串并转换。
- **01:方式 1:** 10 位 UART，**波特率可变** (由定时器 1 控制)。
- **10:方式 2:** 11 位 UART，波特率固定 (fosc/32 或 fosc/64)。
- **11:方式 3:** 11 位 UART，**波特率可变** (由定时器 1 控制)。

（1）**工作方式 0:**移位寄存器方式

同步通信，将串行口变成一个 8 位的并行 I/O 口（半双工）

RXD：数据传送 TXD：同步时钟线

（2）工作方式 1:10 位异步收发通信模式（全双工）

1 位起始位——0、8 位数据位——有用信息、1 位停止位——1

发送：TI=0

接收：RI=0，REN=1　　接受有效的条件：（1）RI=0（2）SM2=0 或停止位为 1

（3）工作方式 2:

1 位起始位——0　　8 位数据位——有用信息

1 位校验位——对有用信息的奇偶校验　　1 位停止位——1

接受有效：（1）RI=0（2）SM2=0 或停止位为 1

（4）工作方式 3:波特率设置同方式 1，其他方式同方式 2

5. 波特率计算

方式 0:fosc/12

方式 1 或 3:当使用定时器 1 作为波特率发生器时（通常设为 T1 为工作方式 2：8 位自动重装）：

波特率 = (2^SMOD / 32) * (T1 溢出率)

T1 溢出率 = fosc / [12*(256 – TH1)]

简化公式: 波特率 = (2^SMOD * fosc) / (384 * (256 – TH1))

据此可反推出 TH1 的装载值。

方式 2:fosc/64（SMOD=0） fosc/32（SMOD=1）

6. **串口通信编程方法**

（1）定工作方式——确定 SCON 中的 SM0 和 SM1

（2）其他位：SM2，REN，TI，RI

（3）计算波特率：1，2，3:确定 SMOD。1，3:确定初值 TH1

（4）中断方式的设置

- **查询方式:**
  - **发送:** while(TI == 0); TI = 0; SBUF = dat; (循环查询发送完成标志位 TI)
  - **接收:** while(RI == 0); RI = 0; dat = SBUF; (循环查询接收完成标志位 RI)
- **中断方式:**

1. **初始化:** EA=1; ES=1;
2. **中断服务函数:**

```
void Uart_Routine(void) interrupt 4 {
    if (RI) {
        RI = 0;
        // 接收处理
    }
    if (TI) {
        TI = 0;
        // 发送处理
    }
}
```

第九章：接口扩展

1. 外部总线

- **数据总线 (Data Bus):** P0 口复用。
- **地址总线 (Address Bus):** P2 口 (高 8 位) 和 P0 口 (低 8 位) 复用。
- **控制总线 (Control Bus):** ALE, PSEN, RD, WR 等。

2. 编址方法

- **线选法:** 用高位地址线直接作为片选信号。电路简单，但地址空间浪费严重，容易地址重叠。
- **译码法:** 用地址译码器（如 74LS138）对高位地址线进行译码，产生片选信号。地址空间连续，无浪费。

3. ADC/DAC

- **ADC (Analog-to-Digital Converter):** 模数转换器，将模拟信号转换为数字信号。
- **DAC (Digital-to-Analog Converter):** 数模转换器，将数字信号转换为模拟信号。
- **分辨率 (Resolution):** ADC/DAC 能分辨的最小模拟量变化，通常用位数表示。n 位 ADC 的分辨率为 Vref / 2^n。
- **量化误差:** 由于数字量是离散的，转换过程中产生的误差。

4. SPI 总线

- **特点:** 高速、全双工、同步串行总线。
- **接口定义:**
  - **SCLK (Serial Clock):** 时钟线，由主机产生。
  - **MOSI (Master Out Slave In):** 主机输出，从机输入线。
  - **MISO (Master In Slave Out):** 主机输入，从机输出线。
  - **CS/SS (Chip/Slave Select):** 片选线，低电平有效。

- **TLC549 (ADC), TLC5615 (DAC):** 都是常见的 SPI 接口芯片，需要用 I/O 口模拟 SPI 时序来读写数据。

5. I2C 总线 (IIC)

- **特点:** 双向、两线制 (SDA：数据线, SCL：时钟线)、同步串行总线。通过**软件寻址**来区分总线上的不同设备。空闲时，SDA 和 SCL 两条线都由上拉电阻拉高。
- **寻址字节:**
  - 一个 7 位的设备地址 + 1 位读写方向位 (0: 写, 1: 读)。
- **数据帧格式:**
  - **起始信号:** SCL 为高时，SDA 由高变低。
  - **数据传输:** SCL 为低时，SDA 上的数据变化；SCL 为高时，SDA 上的数据必须稳定。
  - **应答信号 (ACK):** 接收方在第 9 个时钟周期将 SDA 拉低，表示成功接收。
  - **非应答信号 (NACK):** 接收方让 SDA 保持高电平。
  - **终止信号:** SCL 为高时，SDA 由低变高。
- **AT24C02 (EEPROM), PCF8591 (ADC/DAC):** 都是常见的 I2C 接口芯片，需要用 I/O 口模拟 I2C 的起始、终止、收发字节、应答等时序。