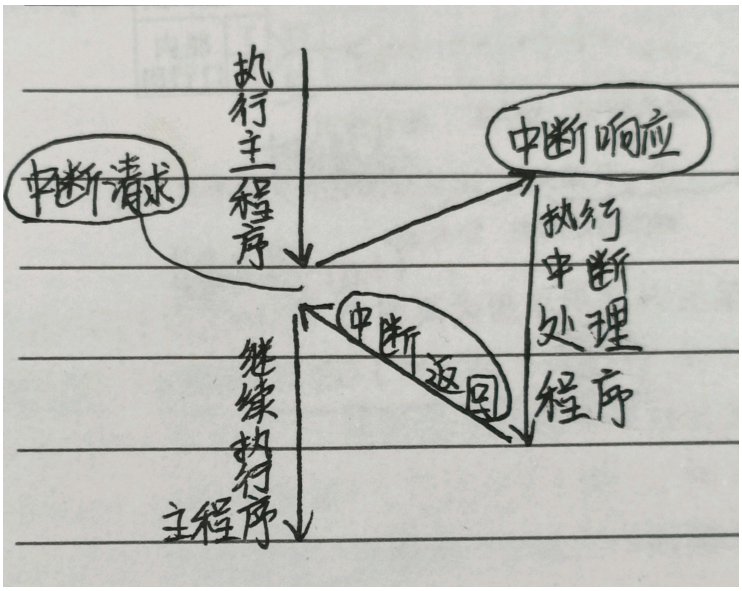
51单片机中断系统

笔记本： 我的第一个笔记本

创建时间： 2023/11/22 17:26 **更新时间：** 2023/12/3 11:27

1.基本概念：CPU在处理某一事件A时，另一事 件B发出请求，(中断请求)CPU暂时中断当前的工作，转去处理事件B(中断响应和中断服务)待CPU将事件B处理完毕后，再回到原来事件A被中断的地方继续处理事件A(中断返回)，这一过程称为中断

2.中断过程示意图



（2）CPU响应中断请求条件：

①中断源有中断请求(中断标置位置1)

②此中断源的中断允许位为1

③CPU开中断(即EA=1)

（3）一次中断过程的完整步骤：

①中断请求：中断事件一旦发生， 中断源就提交中断请求(将中断标志位置1放下开关)

②中断使能：虽然中断源提交了中断请求，但是，能否得到CPU的响应，还要取决于该中断请求能否通过若干关卡送达CPU (中断使能位等于1，关卡放行)，这些关卡有以下2类。

a. 此中断源的中断允许位(1开0关)

b.全局中断允许位(1开0关)

注：EA位总开关不接通，各单路接通也没用

③中断响应：如果一路放行，则CPU响应该中断请求，记录断点，跳转到中断服务程序。对于INT和TMR中断，中断响应时中断标志位会被硬件自动清零。

④中断处理。对中断源进行有针对性的服务。

⑤中断返回。返回到主程序断点处，继续执行主程序。

3.中断系统优点

①分时操作：CPU可以分时为多个外设服务，提高了计算机的利用率

② 实时响应：CPU能够及时处理应用系统的随机事件，系统的实时性大大增强

③可靠性高：CPU具有处理设备故障及掉电等突发性事件能力，从而使系统可靠性提高

4.中断系统特点：

①提高CPU效率

②解决速度矛盾
③实现并行工作
④应付突发事件

5.中断优先级控制（以80c51为例）：

每个中断源的中断优先级都是由中断优先级寄存器IP中的相应位的状态来规定的，IP中断优先级寄存器地址为B8H

表 9-1　中断优先级寄存器 IP

位	7	6	5	4	3	2	1	0
功能			PT2	PS	PT1	PX1	PT0	PX0

PX0（IP.0),外部中断0优先级设定位
PT0(IP.1),定时/计数器T0优先级设定位
PX1 (IP.2),外部中断1优先级设定位
PT1 (IP.3), 定时/计数器T1优先级设定位
PS(IP.4)，串口优先级设定位
PT2 (IP.5)， 定时/计数器T2优先级设定位
同一优先级中的中断请求不止一个时，因此有中断优先权排队问题。同一优先级的中断优先权排队，由中断系统硬件确定的自然优先级形成

表 9-2　各中断源响应优先级及中断服务程序入口

中 断 源	中 断 标 志	中断服务程序入口	优先级顺序
外部中断 0（$\overline{INT0}$）	IE0	0003H	高
定时 / 计数器 T0	TF0	000BH	↓
外部中断 1（$\overline{INT1}$）	IE1	0013H	↓
定时 / 计数器 T1	TF1	001BH	↓
串口	RI 或 TI	0023H	低

设置51单片机的4个中断源，使他们的优先顺序为

T1，INT1, INTO, TO
IPH= 0x08，PT1=1, IP= 0X40, PX1= 1

80C51中断优先级有如下3条原则：
①CPU同时接收到几个中断请求时，首先响应优先级别最高的中断请求
②正在进行的中断过程不能被新的同级或低优先级的中断请求所中断
③正在进行的低优先级中断服务，能被高优先级中断请求所中断
为了实现上述后2条原则，中断系统内部设有2个用户不能寻址的优先级状态触发器。其中一个置1，表示正在响应高优先级的中断，它将阻断后来所有的中断请求;另一个置1表示正在响应低优先级中断，它将阻断后来所有的低优先级中断请求

6.中断允许控制：

CPU对中断系统所有中断以及某个中断源的开放和屏蔽是由中断允许寄存器IE控制的IE中断允许寄存器的地址为A8H

表 9-3　中断允许寄存器 IE

位	7	6	5	4	3	2	1	0
功能	EA			ES	ET1	EX1	ET0	EX0

EXO (IE.0)，外部中断0允许位
ETO(IE.1)，定时1计数器T0中断允许位
EX1 (IE.2),外部中断0允许位
ET1 (IE.3)， 定时/计数器T1中断允许位
ES(IE.4)， 串口中断允许位
EA(IE.7), CPU中断允许(总允许)位

定时器/计数器控制寄存器TCON相应位

表 9-4　定时器 / 计数器控制寄存器 TCON

位	D7	D6	D5	D4	D3	D2	D1	D0
功能	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT0, IT1，设置外部中断的触发方式
为0时，低电平触发方式
为1时，负跳变触发方式
IE0, IE1, 外部中断标志位
其他的是定时/计数器的控制
TF0, TF1,定时器的中断标志
TR1, TR0,打开相应的定时器

7.中断服务函数

- 写法void 函数名（）interrupt n //函数名自己命名，n为中断编号

注意：interrupt必须加，表示定义成中断服务函数
中断号0：外部中断0（INT0）
中断号1：定时/计数器0（T0）
中断号2：外部中断1（INT1）
中断号3：定时/计数器1（T1）
中断号4：串行口中（RI/TI）

- 中断函数与普通函数的异同

(1)相同点是：函数的形式非常类似，中断响应过程和普通函数调用过程也非常相似
(2)不同点有如下几方面：
①中断服务函数不需要声明，普通函数一般需要声明
②普通函数的执行是可预测的;中断服务函数的执行是不可预测的
③)普通函数的跳转是软件(函数调用语句)完成的;中断服务的跳转(中断响应)是由硬件完成的，只要发生了中断事件，并且中断被允许，硬件自动完成中断服务的跳转(中断响应)
④普通函数通过函数名找到被调用函数;中断服务函数通过中断号找到中断服务函数。由此可知，中断服务函数中的函数名其实并没有什么作用
8.中断服务函数遵循规则
(1)中断服务函数不能进行参数转递
(2)在任何情祝下，都不能直接调用中断服务函数
(3)函数内代码越少越好

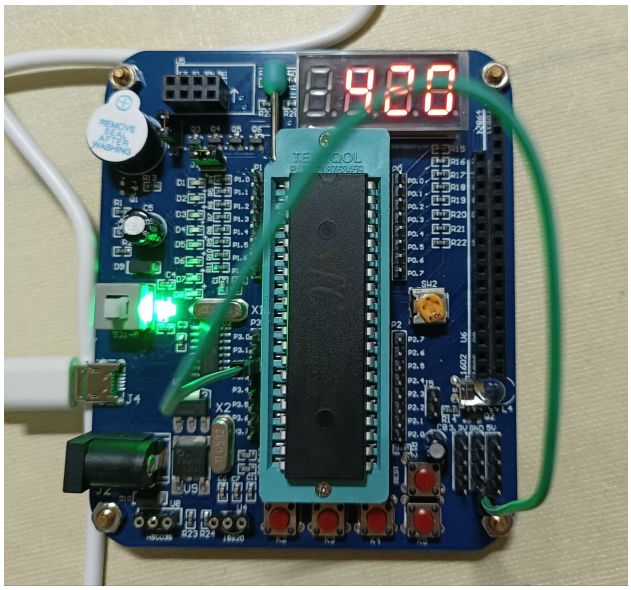
8.利用·中断服务函数实现LED数码管显示学号后三位

- 代码

```c
#include <reg52.h>
#define duan P0
#define uchar unsigned char
//定义第一，二，三，四位LED显示器
sbit weil = P2^4;
sbit wei2 = P2^5;
sbit wei3 = P2^6;
sbit wei4 = P2^7;
//四个数码管引脚
uchar code sz[17] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x39,0x5e,0x79,0x71,0x00};
//实现延时功能，防抖
void delay(unsigned int xms)
{
    unsigned int i,j;
    for(i = xms;i>0;i--)
    for(j = 112;j>0;j--);
}
void low() interrupt 0  //中断函数
{
    delay(10);
}
void main()
{
    while(1)
    {
        duan = sz[4];
        weil = 0;
        wei2 = 1;
        wei3 = 0;
        wei4 = 0;
}
```

```c
void low() interrupt 0  //中断函数
{
    delay(10);
}
void main()
{
    while(1)
    {
        duan = sz[4];
        weil = 0;
        wei2 = 1;
        wei3 = 0;
        wei4 = 0;
        delay(1);
        duan = sz[2];
        weil = 0;
        wei2 = 0;
        wei3 = 1;
        wei4 = 0;
        delay(1);
        duan = sz[0];
        weil = 0;
        wei2 = 0;
        wei3 = 0;
        wei4 = 1;
        delay(1);
        //LED后三位显示420
    }
}
```

- 效果


REMOVE SELF AFTER MASKING