单片机中断流程分析与示例

笔记本：	中断		
创建时间：	2023/11/28 18:13	**更新时间：**	2023/12/4 22:48
作者：	d095yku4		
URL：	https://eqstu.tfswufe.edu.cn/web/index.html#/student/stuOnlineWork/Onli...		

1.单片机中断基本理念
中断是单片机中的一个重要概念，它可以在CPU执行一条指令时，暂停当前的任务，转而执行一个优先级更高的任务。这个优先级更高的任务通常是一些紧急事件的处理，例如外部设备的输入、定时器的计时等。通过中断机制，单片机可以在不同的任务之间切换，从而实现多任务处理。中断机制的实现需要对硬件进行配置和编程，因此需要深入理解中断的基本原理和操作流程。
2.中断的优势
（1）实时响应：中断可以使单片机快速响应外部事件，提高系统的实时性。

（2）高效处理：中断可以让单片机在执行某个任务的同时，处理其他任务，提高处理效率。

（3）简化程序：通过中断，可以将复杂的任务分解为简单的程序和中断处理程序，使程序结构更清晰。

3. 中断函数与普通函数异同：

A.相同点：都是用于完成特定功能的代码块。

B.不同点：

(1)中断函数：响应中断请求，处理中断事件，执行完毕后返回主程序。

(2)普通函数：按照程序顺序执行，无中断响应和处理。

4. 中断函数编码注意事项：

(1)声明中断向量：每个中断事件都需要声明一个中断向量，用于指向对应的中断处理函数。

(2) 编写中断处理函数：中断处理函数需要根据具体中断事件的需求编写相应的代码。

(3)设置中断优先级：根据中断事件的紧急程度，设置合适的中断优先级，避免低优先级中断抢占高优先级中断。

(4)避免死锁：在编写中断处理程序时，要注意避免死锁现象，确保中断处理程序能够顺利完成。

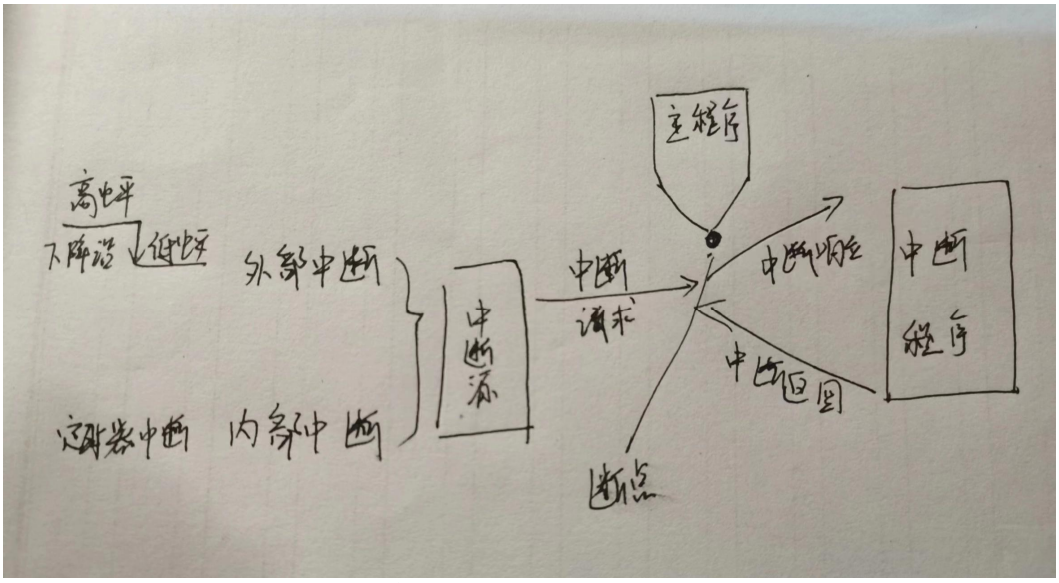
5. 中断使用注意事项：

(1)正确配置中断：确保每个中断事件都有对应的中断向量和处理函数。

(2)避免中断优先级冲突：合理设置中断优先级，避免不同中断事件之间的优先级冲突。

(3)合理使用中断：根据实际需求使用中断，避免过度使用中断导致系统性能下降。

(4)保护现场：在中断处理程序中，要注意保护现场数据，避免数据丢失



中断源向CPU提出中断请求，Cpu暂时中断原来正在处理的事件A，转去处理事件B,处理完毕后再返回原来被中断的地方。

Keil程序代码

```
#include <reg51.h>

// 数码管连接的IO口定义

sbit D0 = P1 ^ 0;

sbit D1 = P1 ^ 1;

sbit D2 = P1 ^ 2;

unsigned char num[3] = {3, 4, 0};

void delay(unsigned int ms) {

    unsigned int i, j;

    for (i = 0; i < ms; i++) {

        for (j = 0; j < 123; j++);

    }

}

void displayNumber(unsigned char *num) {

    D0 = num[0];

    D1 = num[1];

    D2 = num[2];

}

void main() {

    EA = 1;     // 全局中断允许

    IT0 = 1;    // 外部中断0触发方式为边沿触发

    while (1) {

        displayNumber(num);
```

```
    }
}
void externalInterrupt0() interrupt 0 {
    displayNumber(num);
    delay(2000);
    D0 = 1;    // 清除数码管显示
    D1 = 1;
    D2 = 1;

}
```