南京立超电子科技有限公司

N79A8211 之中断应用笔记

2009 年 2 月 26

中国南京市和燕路 251 号金港大厦 A 幢 2406 室

Room 2406,Tower A,Jingang mansion,251 Heyan Road,Nanjing 210028,P。R。China

Tel: 0086-25-83306839/83310926 Fax: 0086-25-83737785

Http://www。dycmcu。com

版权申明

修改记录

版本	日期	作者	修订内容	对应页码
V1.0	2/26/2009	宋娴	初始版本	

目 录

1、文件概要

该项内容主要针对N79A8211的中断系统的基本原理和应用技术作相应的应用说明。

N79A8211 系列的中断分 4 个优先级 10 个中断源。 每个中断源都有相应的优先级设置位，标志位中断向量及使能位。另外系统可以关闭或打开所有中断。

下文将用图文分别说明。

注：以下文件中Fosc是指晶振、RC或外部输入的时钟，F_{SYS}为系统时钟。

2、中断综述

N79A8211系列 的中断分4个优先级 10 个中断源。每个中断源都有相应的优先级设置位，标志位中断向量及使能位。另外系统可以关闭或打开所有中断。

2.1、中断源

N79A8211系列 的中断有10 个中断源（见表2-1）。

中断源	向量地址	中断源	向量地址
外部中断0	0003h	Timer0溢出	000Bh
外部中断1	0013h	Timer1溢出	001Bh
-	0023h	欠压中断	002Bh
-	0033h	边沿触发中断	003Bh
-	0043h	-	004Bh
WDT	0053h	ADC中断	005Bh
-	0063h	PWM钳制中断	0073h
PWM中断	006Bh	-	007Bh

表 2-1：中断向量地址

注：中断向量表不是连续的空间，保留的中断向量可用于今后产品的扩充时使用。

(1)、外部中断 $\overline{INT\ 0}$ 和 $\overline{INT\ 1}$

按照IT0和IT1的设置可以是边沿触发或是电平触发。TCON中的IE0和IE1位是外部中断的标志位，检测这2位的状况可以知道是否产生了外部中断。在边沿触发模式中，系统在每个机器周期都要采样INTx脚。如果在一个周期里采样到高电平在下一个周期里采样到低电平，那么系统就检测到了一个高电平到低电平的跳变，此时相应的IEx位置位，同时向系统申请中断服务。由于系统在每个机器周期都要对外部中断进行采样，因此外部中断输入脚上的高电平或低电平至少要维持一个机器周期。

当系统响应中断执行中断服务程序时，IEx位被自动清除。如果选择电平触发方式，那么中断请求源的低电平信号必须保持到系统响应该中断。在进入中断服务程序时，IEx位不会被硬件清零。如果外部中断输入脚上的电平在中断服务程序完成后依然保持，系统会立即识别该中断再次进入同样的中断服务程序。

(2)、TIMER中断

当TF0、TF1 标志位置位时会产生定时器0和定时器1中断。当定时器溢出时这些标志位会置位。当执行定时器中断服务程序时，这些标志位会被硬件自动清零。

(3)、WDT

看门狗定时器可以用作系统监控器或是一个简单的定时器。无论以何种方式工作，当定时器超时后。看门狗定时器中断标志WDIF（WDCON。3）会置位，如果EIE。4=1，那么这时会产生一个中断。

(4)、串口中断

当串口的两个中断源接受或发送，发生中断时特殊功能寄存器SCON的RI和TI被值‘1’；该位不能自动清‘0’，

用户必须软件清‘0’。

(5)、PWM中断

PWM功能可以产生中断，标志位是BKF，在外部钳制脚发生钳制时会产生中断请求。该位不能自动清‘0’，用户必须软件清‘0’。

(6)、ADC中断

ADC有一个中断源，当ADC转换结束后会产生ADC中断，标志位是特殊功能寄存器ADCCON中的ADCI位；该位不能自动清‘0’，用户必须软件清‘0’。

(7)、边沿触发中断

使能P1。0-P1。2引脚上的边沿触发中断。 该标志位由软件清除。

(8)、欠压中断

欠压检测启动BOD (AUXR。6)，BOF (PCON。5) 标志置位可以触发欠压中断，BOF将由软件清除，如果 BOI (AUXR1。5)置‘1’，欠压检测触发中断EA (IE。7)并EBO (IE。5) 位置‘1’。为了确保正确的检测欠压，VDD下降时间必须慢于50mV/us，上升时间慢于2mV/us。

2.2、中断优先级

N79A8211系列有4个优中断先级结构。这样使N79A8211系列控制更多的中断源有极大的灵活性，N79A8211系列支持多达10个中断源，中断不会打断同等优先级的中断和较高优先级的中断服务程序；最高优先权中断不会被任何中断打断；故若同时有2个中断请求，较高优先级的中断先执行服务程序。

若具有同等优先级中断同时请求中断，由内部有一个监测顺序来决定执行中断服务程序的顺序。

优先位		中断优先级
IPxH	IPx	IPxH
0	0	0
0	1	0
1	0	1
1	1	1

表2-2：四级中断优先级

如下表所示的内容是：中断源、标志位、向量地址、允许位、优先权位、仲裁序列，并且描述了那一个唤醒CPU的掉电模式。

中断源	标置位	向量地址	中断使能位	中断优先级	清标置位	仲裁序列	唤醒掉电
External Interrupt0	IE0	0003H	EX0(IE0.0)	IP0H.0,IP0.0	Hardware,Follow theinverse of pin	1 (最高)	Yes
Brownout Detect	BOF	002BH	EBO(IE.5)	IP0H.5,IP0.5	Software	2	Yes
Watchdog Timer	WDIF	0053H	EWDI(EIE.4)	IP1H.4,IP1.4	Software	3	Yes(1)
Timer0Interrupt	TF0	000BH	ET0(IE.1)	IP0H.1,IP0.1	Hardware,software	4	No
ADC Converter	ADCI	005BH	EAD(IE.6)	IP0H.6,IP0.6	Hardware	5	Yes(1)
External Interrupt1	IE1	0013H	EX1(IE.2)	IP0H.2,IP0.2	Hardware,Follow theinverse of pin	6	Yes
Edge Detect Interrupt	EDF	003BH	EED(EIE.7)	IP1H.7,IP1.7	Software	7	No
Timer1 Interrupt	TF1	001BH	ET1(IE.3)	IP0H.3,IP0.3	Hardware,software	8	No
PWM Period Interrupt	PWMF	006BH	EPWMUF (EIE.6)	IP1H.6,IP1.6	Software	9	No
PWM Brake Interrupt	BKF	0073H	EPWM(EIE.5)	IP1H.5,IP1.5	Software	10 (最低)	No

Note: 1. 当使用内部RC作为主振时，ADC可以将芯片由掉电模式唤醒..

表 2-3: 唤醒掉电模式的中断向量地址

2.3、中断响应

所有中断产生标志均可由硬件置位/复位，同样若软件将这些位置位也可以引发中断。各个中断可以由IE寄存器中的相应位来打开或关闭。IE中有一个中断总控制位，可以打开或关闭所有的中断。

每个机器周期都检测中断标志和中断优先权。如果满足特定条件硬件将执行内部产生的LCALL 指令，目标地址是中断向量地址。产生LCALL的条件是：

（1）、 较低优先级的中断不会打断同等优先级的中断和较高优先级的中断服务程序。

（2）、 在正在执行指令的最后一个周期检测中断标志。

（3）、 正在执行的指令不包括写IE, EIE, IP0, IP0H, IP1 or IPH1寄存器的指令并且不是RETI。

如果上述的任何一个条件不满足，LCALL就不会发生。在每一个指令周期都会检测中断标志。如果上述条件有一个不满足，虽然标志位置‘1’，也不能响应中断。当所有的条件都满足了，中断标志已经消失，该中断也不能再被回应。

处理器响应一个有效的中断是通过执行一个LCALL指令将程序转移到中断入口地址。引起中断的中断标志可能被清除也有可能不被清除。当进入中断服务程序定时器中断的TF0、TF1标志会被硬件清除。外部中断INT0和INT1只有在它们的触发条件发生时他们的标志被清除。看门狗定时器中断标志 WDIF必须有软件清除。硬件执行一个长调指令。该指令保存程序计数器的内容到堆栈，但是不保存程序状态字PSW。当中断发生时PC被装入中断向量地址。中断源的向量地址见上表2-1。

程序从向量地址连续执行到RETI 指令。执行RETI指令处理器将从栈顶弹出数据并装载到PC指针。用户必须注意在进入中断后堆栈存放的内容，如果执行中断返回操作，栈顶的内容已经改变但CPU不会知道堆栈的内容已经改变掉，而是按正常情况将栈顶的数据装入PC指针，这样将会引起错误发生。

2.4、中断响应时间

每一个中断源的响应时间取决于几个方面，如中断自身特点和指令的执行。外部中断 $\overline{INT\ 0}$ 和RI+TI在机器周期的C3采样并且他们相应的中断标志IEx自动的置位或清除。定时器0和1溢出标志在机器周期的C3置位，在下一个机器周期检测中断标志。如果有1个中断请求满足3个条件，硬件将自动产生长跳指令，该指令需要4个机器周期。这样从中断标志置位到执行中断服务程序最少只需要5个机器周期。

如果很长的响应时间可以预知的是三个条件有一个不满足，或者有较高或同等优先级的中断正在执行中断服务程序，很明显中断等待时间与正在执行的中断服务程序的长短有关。如果检测机器周期正在执行指令，需等待指令执行完毕，最大的响应时间(如果不在其它中断的服务程序)发生在W79E4051/2051系列执行写IE, IP, IP1或IPH和MUL、DIV 指令。中断中断源的最长响应时间是12个机器周期，其中包括检测中断1机器周期，完成IE, EIE, IP0, IP0H, IP1 or IP1H访问2机器周期，完成MUL或DIV 指令5机器周期和完成硬件LCALL中断向量位置4机器周期。

也就是说一个简单中断系统中断响应时间总是大于5机器周期并且不大于12机器周期。最大的等待时间是12机器周期即是48时钟周期。注：标准 8051最小等待时间为8机器周期即是96时钟周期。 这可以减少50%时钟周期。

2.5、中断输入

N79A8211系列有10个中断源和两个独立的中断源输入。

把N79A8211系列设置为掉电或空闲模式，如果外部中断为允许，中断产生将唤醒CPU，并继续执行程序。

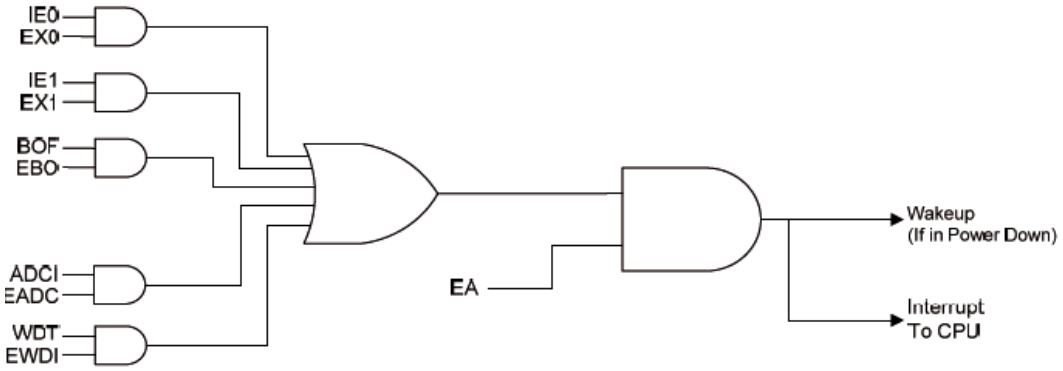


图 2-1: 中断可以唤醒掉电模式

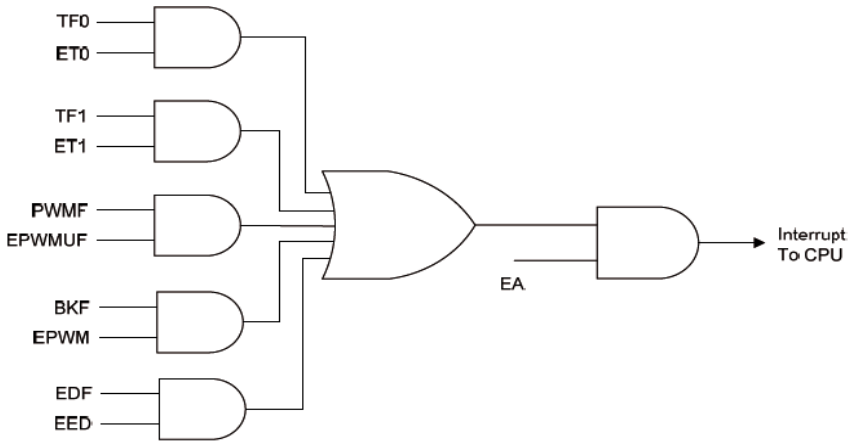


图 12-2：中断不能唤醒掉电模式

2.6、中断的响应过程

（1）、现场保护。将当前地址、ACC、状态寄存器保持到堆栈中。

（2）、切换PC指针，根据不同的中断源所产生的中断，切换到相应的入口地址。

（3）、执行中断处理程序。

（4）、现场恢复。将保存在堆栈中的主程序地址、ACC、状态寄存器恢复。

（5）、中断返回。

3、中断的相关寄存器

3.1、中断允许

助记符:IE 地址: A8h 上电复位值：00H

bit	7	6	5	4	3	2	1	0
名称	EA	EADC	EBO	ES	ET1	EX1	ET0	EX0

位	名称	功能
7	EA	全局中断允许。允许/禁止所有的中断。
6	EADC	允许ADC中断。
5	EBO	允许欠压中断。
4	ES	允许串行端口中断。
3	ET1	允许定时器1中断。
2	EX1	允许外部中断1。
1	ET0	允许定时器0中断。
0	EX0	允许外部中断0。

3.2、中断高级优先

助记符: IP0H 地址: B7h

Bit	7	6	5	4	3	2	1	0
名称	-	PADCH	PB0H	-	PT1H	PX1H	PT0H	PX0H

位	名称	功能
7	-	保留位，如果读它结果是高。
6	PADCH	1:设置ADC中断为高级优先。
5	PB0H	1:设置欠压监测器中断为高级优先。
4	-	保留
3	PT1H	1:设置定时器1中断为高级优先。
2	PX1H	1:设置外部中断1中断为高级优先。
1	PT0H	1:设置定时器0中断为高级优先。
0	PX0H	1:设置外部中断0中断为高级优先。

3.3、中断优先权 0

助记符: IP 地址: B8h

Bit	7	6	5	4	3	2	1	0
名称	-	PADC	PBO	-	PT1	PX1	PT0	PX0

位	名称	功能
7	-	该位是没有使用，读的结果是‘1’.
6	PADC	1:设置中断ADC的优先权是较高优先级.
5	PBO	1:设置中断欠压监测器优先权是较高优先级.
4	-	保留.
3	PT1	1:设置中断定时器1的优先权是较高优先级.
2	PX1	1:设置中断外部中断1的优先权是较高优先级.
1	PT0	1:设置中断定时器0的优先权是较高优先级.
0	PX0	1:设置中断外部中断 0 的优先权是较高优先级.

3.4、中断允许寄存器 1

助记符：EIE 地址：E8h

Bit	7	6	5	4	3	2	1	0
名称	EED	EPWMUF	EPWM	EWDI	-	-	-	-

位	名称	功能
7	EED	0：禁止边沿触发中断. 1：使能边沿触发中断.
6	EPWMUF	0：禁止PWM下溢中断. 1：使能 PWM 下溢中断.
5	EPWM	0：禁止PWM下溢中断（外部钳制）. 1：使能 PWM 下溢中断（外部钳制）.
4	EWDI	0：禁止看门狗中断. 1：使能看门狗中断.
3-0	-	保留.

3.5、中断优先级 1

助记符：IP1H 地址：F7h

Bit	7	6	5	4	3	2	1	0
名称	PEDH	PPWMH	PBKH	PWDIH	-	-	-	-

位	名称	功能
7	PEDH	1:设定边沿触发中断最高.
6	PPWMH	1:设定PWM中断最高.
5	PBKH	1:设置PWM钳制中断级优先级最高.
4	PWDIH	1:设置看门狗中断先级最高.
3-0	-	保留.

3.6、中断优先权

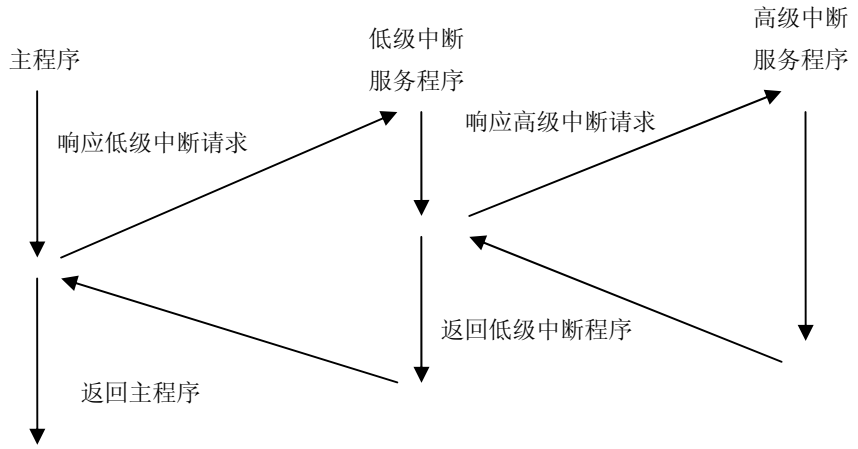
助记符: IP1 地址: F8h

Bit	7	6	5	4	3	2	1	0
名称	PED	PPWM	PBK	PWDI	-	-	-	-

位	名称	功能
7	PED	1:设定边沿触发中断优先级较高.
6	PPWM	1:设定PWM中断优先级较高.
5	PBK	1:设置PWM钳制中断级优先级较高.
4	PWDI	1:设置看门狗中断优先级较高.
3-0	-	保留.

4、中断的使用

4.1、中断嵌套处理示意图



4.2、中断服务程序

```
/*********************************************************************************************
** 函数名称: INT0_ISP
** 函数描述: 外部中断0中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void INT0_ISP(void) interrupt 0
{
   EA=0;
   IE0=0;                                  //清中断0标志
   //在此加入您要处理的代码
   EA=1;
}

/*********************************************************************************************
** 函数名称: Brownout_ISP
** 函数描述: 掉电检测中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Brownout_ISP(void) interrupt 5
```

```
{
   EA=0;
   BOF_CLR;                              //清掉电检测中断标志
   //在此加入您要处理的代码
   EA=1;
}
/*********************************************************************************************
** 函数名称: WDT_ISP
** 函数描述: 看门狗中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void WDT_ISP(void) interrupt 10
{
   EA=0;
   WDIF=0;                               //清看门狗中断标志
   //在此加入您要处理的代码
   EA=1;
}
/*********************************************************************************************
** 函数名称: Timer0_ISP
** 函数描述: TIMER0中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Timer0_ISP(void) interrupt 1
{
   EA=0;
   TF0=0;                                //清T0中断标志
   TL0=T0_5MS_CNT&0x00ff;
   TH0=(T0_5MS_CNT>>8)&0x00ff;
   //在此加入您要处理的代码
   EA=1;
}
```

```
/*********************************************************************************************
** 函数名称: ADC_ISP
** 函数描述: AD中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void ADC_ISP(void) interrupt 11
{
   EA=0;
   ADCI_CLR;                          //清ADCIF
   //在此加入您要处理的代码
   EA=1;
}
/*********************************************************************************************
** 函数名称: INT1_ISP
** 函数描述: 外部中断1中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void INT1_ISP(void) interrupt 2
{
   EA=0;
   IE1=0;                             //清外部中断标志1
   //在此加入您要处理的代码
   EA=1;
}
/*********************************************************************************************
** 函数名称: Edge_ISP
** 函数描述: 边沿检测中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Edge_ISP(void) interrupt 7
{
```

```
    EA=0;
    EDF_CLR;                          //清边沿中断标志
    //在此加入您要处理的代码
    EA=1;
}
/*********************************************************************************************
** 函数名称: Timer1_ISP
** 函数描述: TIMER1中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Timer1_ISP(void) interrupt 3
{
    EA=0;
    TF1=0;                            //清T1中断标志
  TL1=T1_5MS_CNT&0x00ff;
  TH1=(T1_5MS_CNT>>8)&0x00ff;
 //在此加入您要处理的代码
     EA=1;
}
/*********************************************************************************************
** 函数名称: SPI_ISP
** 函数描述: SPI中断服务程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void PWMP_ISP(void) interrupt 13
{
    EA=0;
    PWMF_CLR;                         //清PWM周期中断标志
    //在此加入您要处理的代码
    EA=1;
}
/*********************************************************************************************
```

```
** 函数名称: INT2_ISP
** 函数描述: 外部中断2中断服务程序
** 输入参数: 无
** 输出参数: 无
*****************************************************************************************/
void PWMB_ISP(void) interrupt 14
{
   EA=0;
   BKF_CLR;                                  //精钳制中断位
   //在此加入您要处理的代码
   EA=1;
}
```

4.3、注意事项

（1）、进入中断前，要中断允许。若程序中有几个中断，如有特殊需求需要设定优先级。

（2）、进入中断后，中断标志要清除。

（3）、主程序地址、ACC、状态寄存器等需要恢复，TIMER每进一次中断要恢复初值。

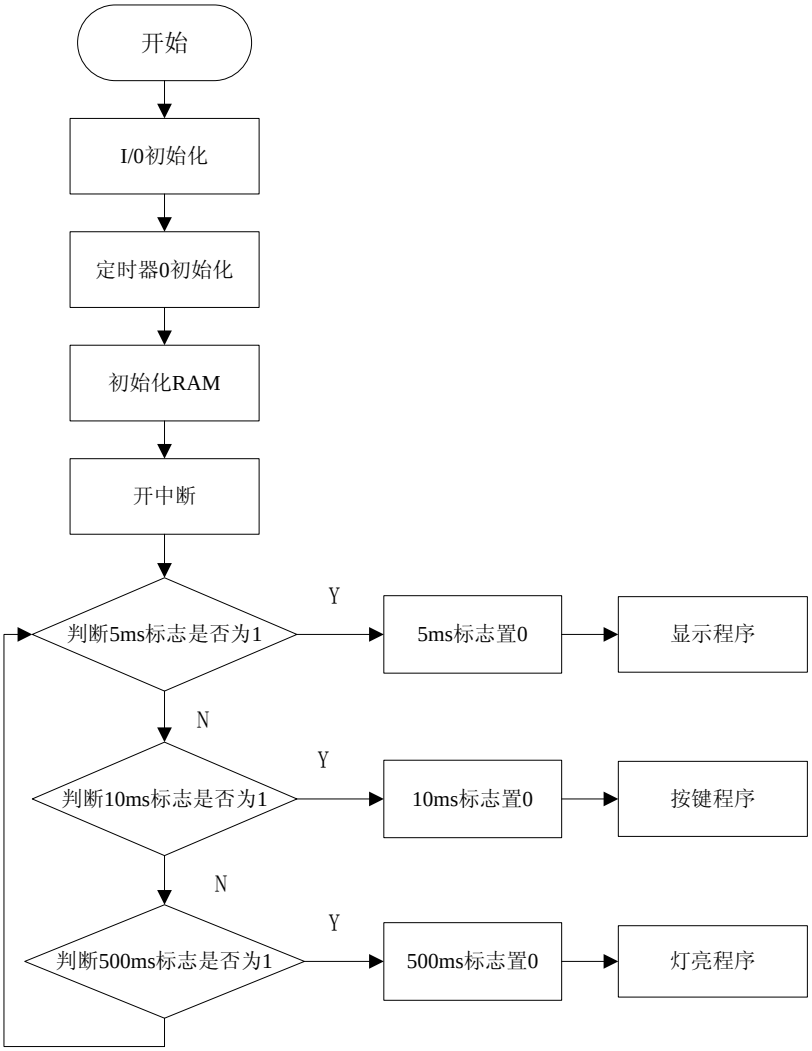
5、案例及说明

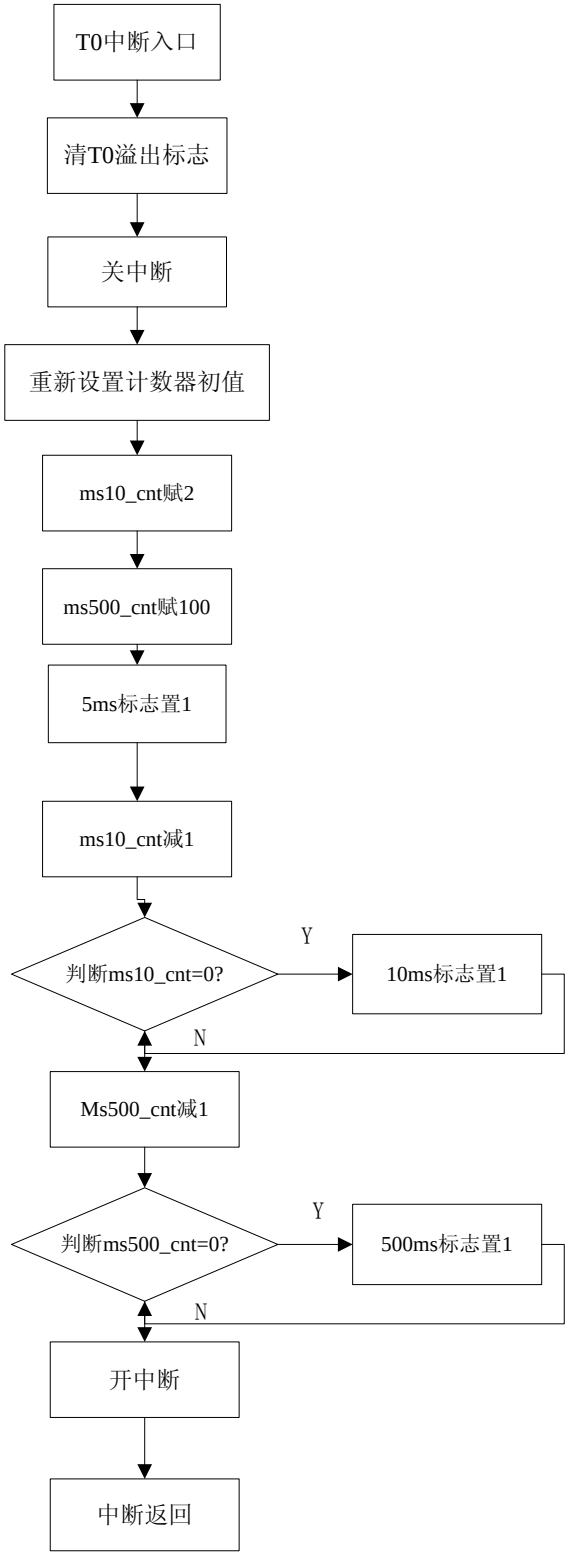
下面是以N79A8211 为例编写的程序通过按键实现数码管0-4的显示和八个LED循环显示。采用TIMER0中断。

功能：（1）、通过按键实现数码管0-4的显示。

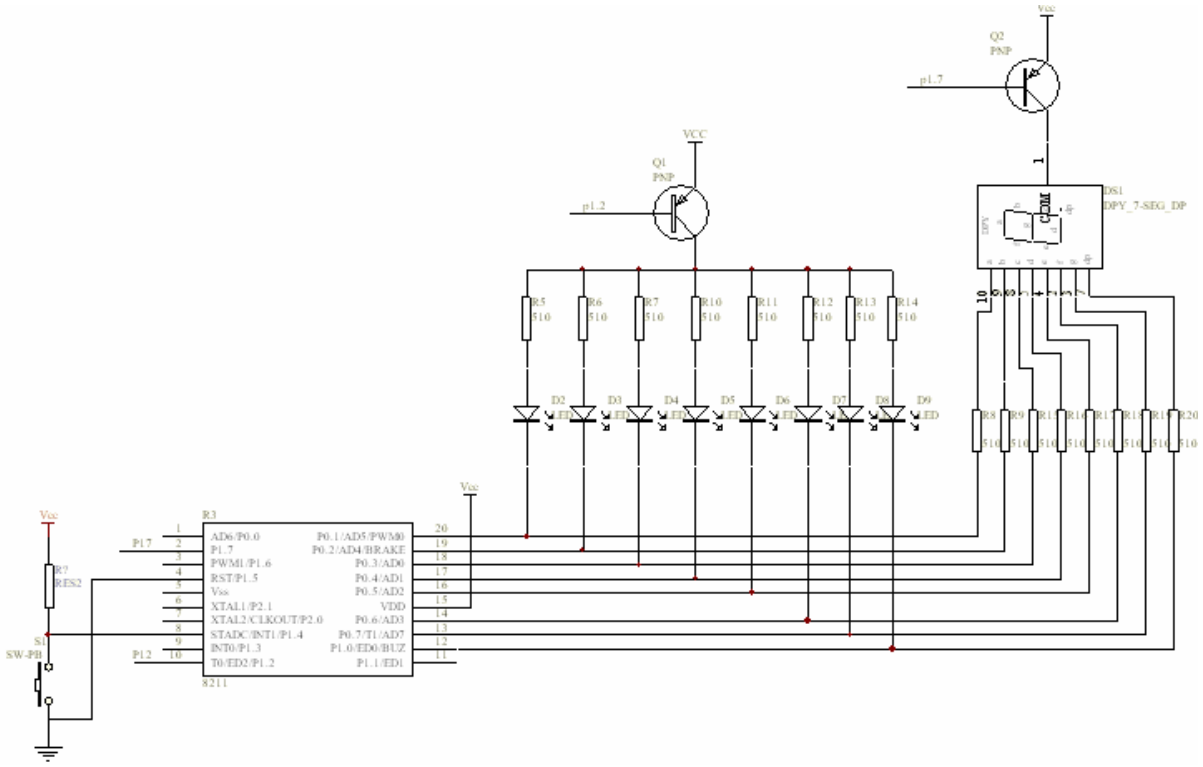
（2）、当数码管显示1时，实现八个LED循环显示。

5.1、程序流程图




T0中断入口
清T0溢出标志
关中断
重新设置计数器初值
ms10_cnt赋2
ms500_cnt赋100
5ms标志置1
ms10_cnt减1
判断ms10_cnt=0?
Y
10ms标志置1
N
Ms500_cnt减1
判断ms500_cnt=0?
Y
500ms标志置1
N
开中断
中断返回

5.2、电路图



5.3、程序

```
#include "config.h"
uint8 code Eight_TAB[]={0xc0,0xf9,0xa4,0xb0,0x99};                    //数码显示字型码
uint8 code Led_TAB[]={0xff,0xfe,0xfd,0xfc,0xf7,0xef,0xdf,0xcf,0x7f,0x00};      //LED码
uint8 bdata  gBTFlag;
sbit  gbT_2ms_Flag   =  gBTFlag^0;     //2ms标志
sbit  gbT_10ms_Flag  =  gBTFlag^1;     //10ms标志
sbit  gbT_200ms_Flag =  gBTFlag^2;     //200ms标志
uint8 bdata  gKEYFlag;
sbit  gbKST    =  gKEYFlag^0;     //键稳定状态
sbit  gbKCD    =  gKEYFlag^1;     //键连续状态
uint8 fun_no;                     //功能号
uint8 eight_val;
uint8 led_val;
/*******************************************************************************************
** 函数名称: Init_IO
** 函数描述: 初始化IO
```

```
** 输入参数: 无
** 输出参数: 无
*******************************************************************************************/
void Init_IO(void)
{
   P0M1=Bin(00000000);
   P0M2=Bin(11111111);             //设置为推挽输出
   P1M1=Bin(00011000);
   P1M2=Bin(11100111);            //P1.3,P1.4设置为输入状态。
   P2M1=Bin(00000000);
   P2M2=Bin(00000011);
   P0=Bin(11111111);
   P1=Bin(11111111);
   P2=Bin(11111111);
}
/*******************************************************************************************
** 函数名称: Init_Timer
** 函数描述: 初始化TIMER
** 输入参数: 无
** 输出参数: 无
*******************************************************************************************/
void Init_Timer(void)
{
   TR0=0;                                                    //关闭TIMER0
   TF0=0;
   TMOD=Bin(00010001);                  //Timer0和Timer1都做为16位手动重装定时器
   CKCON=0x08;              //时钟选择为1/4系统时钟
   TL0=T0_2MS_CNT&0x00ff;
   TH0=(T0_2MS_CNT>>8)&0x00ff;
   TR0=1;                                                      //启动TIMER0
   ET0=1;
   ET1=0;
}
/*******************************************************************************************
```

```
** 函数名称: Init_RAM
** 函数描述: 初始化RAM
** 输入参数: 无
** 输出参数: 无
*******************************************************************************************/
void Init_RAM(void)
{
   gBTFlag=0x00;
   gKEYFlag=0x00;
   fun_no=0;
   eight_val=0;
   led_val=0;
}
/*******************************************************************************************
** 函数名称: Timer0_ISP
** 函数描述: TIMER0中断服务程序          2ms中断一次
** 输入参数: 无
** 输出参数: 无
*******************************************************************************************/
void Timer0_ISP(void) interrupt 1
{
   static uint8 ms10_cnt=5;
   static uint8 ms200_cnt=100;
   EA=0;
   TF0=0;                              //清T0中断标志
   TL0=T0_2MS_CNT&0x00ff;
   TH0=(T0_2MS_CNT>>8)&0x00ff;
   //在此加入您要处理的代码
   gbT_2ms_Flag=1;
   if(--ms10_cnt==0)
   {
      ms10_cnt=5;
      gbT_10ms_Flag=1;
   }
```

```
    if(--ms200_cnt==0)
    {
        ms200_cnt=100;
        gbT_200ms_Flag=1;
    }
    EA=1;
}
/*********************************************************************************************
** 函数名称: Key_Sub
** 函数描述: 判断键状态
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Key_Sub()
{
    if((KEY1==0)||(KEY2==0))
    {
        if(gbKST)
        {
            if(gbKCD)
            {
                //连续键处理
             /*  if(++fun_no==5)
                {
                    fun_no=1;
                }
                eight_val=fun_no;
                */
            }
            else
            {
                gbKCD=1;
                //单次键处理
                if(++fun_no==5)
```

```
            {
               fun_no=0;
            }
            eight_val=fun_no;
            switch(fun_no)
            {
               case 0:
                  break;
               case 1:
                  Led_Sub();
                  break;
               case 2:
                  break;
               case 3:
                  break;
               case 4:
                  break;
               default:
                  break;
            }
         }
      }
      else
      {
         gbKST=1;
      }
   }
   else
   {
      gbKST=0;
      gbKCD=0;
   }

}
```

```
/*********************************************************************************************
** 函数名称: Fresh_SEG(uint8 val)
** 函数描述: 更新SEG段
** 输入参数: val : 所需更新的值
** 输出参数: 无
*********************************************************************************************/
void Fresh_SEG(uint8 val)
{
   SEG1=0;
   SEG2=0;
   SEG3=0;
   SEG4=0;
   SEG5=0;
   SEG6=0;
   SEG7=0;
   SEG8=0;
   if(val&0x01) SEG1=1;
   if(val&0x02) SEG2=1;
   if(val&0x04) SEG3=1;
   if(val&0x08) SEG4=1;
   if(val&0x10) SEG5=1;
   if(val&0x20) SEG6=1;
   if(val&0x40) SEG7=1;
   if(val&0x80) SEG8=1;
}

/*********************************************************************************************
** 函数名称: Dis_Sub
** 函数描述: 显示子程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Dis_Sub()
```

```
{
   static uint8 comsel=0;

   switch(comsel)
   {
      case 0:
         comsel=1;
         COM1=1;
         COM2=1;
         Fresh_SEG(Eight_TAB[eight_val]);
         COM1=0;
         break;
      case 1:
         comsel=0;
         COM1=1;
         COM2=1;
         Fresh_SEG(Led_TAB[led_val]);
         COM2=0;
         break;
      default:
         break;
   }
}

/*********************************************************************************************
** 函数名称: Led_Sub
** 函数描述: Led_Sub子程序
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void Led_Sub()
{
  if(fun_no)
  {
```

```
      if(++led_val>7)
      {
         led_val=0;
      }
  }
  else led_val=0;
}
/*********************************************************************************************
** 函数名称: main
** 函数描述: 程序主函数
** 输入参数: 无
** 输出参数: 无
*********************************************************************************************/
void main(void)
{
   Init_IO();
   Init_Timer();
   Init_RAM();
   EA=1;
   while(1)
   {
      if(gbT_2ms_Flag)
      {
         gbT_2ms_Flag=0;
         Dis_Sub();
      }
      if(gbT_10ms_Flag)
      {
         gbT_10ms_Flag=0;
         Key_Sub();
      }
      if(gbT_200ms_Flag)
      {
         gbT_200ms_Flag=0;
```

```
            Led_Sub();
        }
    }
}
```

编后说明

NA79A8211 系列单片机中断使用指南，是立超电子为帮助用户迅速掌握NA79A8211单片机而编写的，限于水平，难免有错误和不妥之处，恳请广大读者批评指正。请将您的建议和批评发至E-Mail:xian.song@dycmcu.com，我们深表感谢！

继本指南之后，我们已经或即将编写以下使用指南、应用设计和范例程序等，欢迎及时访问我们的网页：www.dycmcu.com，或通过电话、E-Mail 索取更多的应用指南及内部资料等。

使用指南：

NA79A8211 系列单片机A/D 转换器使用指南

NA79A8211 系列单片机PWM 使用指南

NA79A8211 系列单片机内置WDT 使用指南

NA79A8211 系列单片机中断使用指南

NA79A8211 系列单片机的低功耗设计方法

。。。。。。

应用设计：

SH69PXX 系列单片机与E2PROM 接口及程序设计

SH69PXX 系列单片机I2C 总线模拟程序包

。。。。。。

其它：

NA79A8211 系列单片机特殊功能寄存器定义库文件

NA79A8211 系列单片机特殊功能寄存器定义库文件

NA79A8211 系列单片机特殊功能寄存器定义库文件

本应用指南欢迎各相关电子网站转载，为了尊重我们付出的劳动，请您注名出处来自站点：www.dycmcu.com

立超电子 技术支持部

2009 年 2 月 26 日 星期四

编写单位

南京立超电子科技有限公司

参考资料

1、N79A8211 规格书 单片机规格书。