ZigBee 无线SOC 片上系统

CC2530

基础试验手册

声明：*此手册结合在实际开发过程中对芯片使用以及网络上各工程师对英文手册的翻译整理而成，文中难免有错误及误差，如果你有兴趣完善本手册，请把你的建议发给我们，我们及各使用此芯片工程师将非常感谢你的努力，谢谢!*

目 录

CC2530 基础实验

1.1 输入输出 I/O 控制实验

1.1.1 CC2530 基础实验 1：自动闪烁

1、实验介绍

本次实验的目的是让用户学会使用CC2530 的I/0 来控制外设，本例以LED 灯为外设，用CC2530 控制简单外设时，应将I/O 设置为输出。实验现象LED 闪烁。

实验设备：仿真器1 台，电池板(或液晶板)1 块，ZigBee 模块1 块，USB 连接线1 根。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，没有设置而是取默认值的寄存器有：P1SEL，P1INP。

P1 （P1 口寄存器）

位号	位名	复位值	操作性	功能描述
7:0	P1[7:0]	0x00	读/写	P1端口普通功能寄存器，可位寻址

P1DIR （P1 方向寄存器）

位号	位名	复位值	操作性	功能描述
7	DIRP1_7	0	读/写	P1_7方向 0输入，1输出
6	DIRP1_6	0	读/写	P1_6方向 0输入，1输出
5	DIRP1_5	0	读/写	P1_5方向 0输入，1输出
4	DIRP1_4	0	读/写	P1_4方向 0输入，1输出
3	DIRP1_3	0	读/写	P1_3方向 0输入，1输出
2	DIRP1_2	0	读/写	P1_2方向 0输入，1输出
1	DIRP1_1	0	读/写	P1_1方向 0输入，1输出
0	DIRP1_0	0	读/写	P1_0方向 0输入，1输出

P1SEL （P1 功能选择寄存器）

位号	位名	复位值	操作性	功能描述

7	SELP1_7	0	读/写	P1_7 功能0 普通I/O，1 外设功能
6	SELP1_6	0	读/写	P1_6 功能0 普通I/O，1 外设功能
5	SELP1_5	0	读/写	P1_5 功能0 普通I/O，1 外设功能
4	SELP1_4	0	读/写	P1_4 功能0 普通I/O，1 外设功能
3	SELP1_3	0	读/写	P1_3 功能0 普通I/O，1 外设功能
2	SELP1_2	0	读/写	P1_2 功能0 普通I/O，1 外设功能
1	SELP1_1	0	读/写	P1_1 功能0 普通I/O，1 外设功能
0	SELP1_0	0	读/写	P1_0 功能0 普通I/O，1 外设功能

3、实验相关函数

void Delay(uint n)；函数原型是

```
void Delay(uint n)
  {
    uint tt;
    for(tt = 0;tt<n;tt++);
    for(tt = 0;tt<n;tt++);
    for(tt = 0;tt<n;tt++);
    for(tt = 0;tt<n;tt++);
    for(tt = 0;tt<n;tt++);
  }
```

函数功能是软件延时，执行5 次0 到n 的空循环来实现软件延时。延时时间约为5*n/32 μs。

void Initial(void)；函数原型是：

```
void Initial(void)
{
  P1DIR |= 0x03; //P10、P11 定义为输出
  RLED = 1;
  YLED = 1; //LED 灭
}
```

函数功能是把连接LED 的两个I/0 设置为输出，同时将它们设为高电平（此时LED 灭）。

1.1.2 CC2530 基础实验 2：按键控制开关

1、实验介绍

实验设备：仿真器1 台，液晶板1 块，ZigBee 模块1 块，USB 连接线1 根。

让用户掌握按键应用这一常用人机交互方法，本次实用两个分别控制两个LED 灯。按下“OK”键切换ZigBee 模块左边LED 灯开关，按下“CANCEL”键切换ZigBee 模块右边LED 灯开关。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL，P1INP。前面三个寄存器在实验1 已经有详述，这里不再重复介绍。

P1　　参见1.1.1。

P1SEL　参见1.1.1。

P1DIR　参见1.1.1。

P2INP （P1 输入模式寄存器）

位号	位名	复位值	操作性	功能描述
7	PDUP2	0	读/写	P2 口上/下拉选择 0 上拉，1 下拉
6	PDUP1	0	读/写	P1 口上/下拉选择 0 上拉，1 下拉
5	PDUP0	0	读/写	P0 口上/下拉选择 0 上拉，1 下拉
4	MDP2_4	0	读/写	P2_4 输入模式 0 上拉/下拉，1三态
3	MDP2_3	0	读/写	P2_3 输入模式 0 上拉/下拉，1三态
2	MDP2_2	0	读/写	P2_2 输入模式 0 上拉/下拉，1三态
1	MDP2_1	0	读/写	P2_1 输入模式 0 上拉/下拉，1三态
0	MDP2_0	0	读/写	P2_0 输入模式 0 上拉/下拉，1三态

3、实验相关函数

void Delay(uint n)；参见CC2530 基础实验1。

void Initial(void)；参见CC2530 基础实验1。

void InitKey(void);函数原型：

```
void InitKey(void)
  {
    P0SEL &= ~0X30; //P04 P05输入
    P0DIR &= ~0X30; //
```

```
    P0INP |= 0x30; //三态
  }
```

函数功能是将I/O P0_4,P0_5 设为输入（且为三态）以读取按键的状态。

unsigned char KeyScan(void);函数原型：

```
uchar KeyScan(void)
{
  if(K1 == 0) //低电平有效
  {
    Delay(100); //检测到按键
    if(K1 == 0) //前面定义了 #define K1 P04
    {
      while(!K1); //直到松开按键
      return(1);
    }
  }
  if(K2 == 0)
  {
    Delay(100);
    if(K2 == 0)
    {
        while(!K2);
        return(2);
    }
  }
  return(0);
}
```

函数功能是检测按键是否按下，若有键按下，则返回相应的值，如P0_4 对应的按键按下则返回1，P0_5 对应的按键按下返回2。

1.1.3 CC2530 基础实验 3：按键控制闪烁

1、实验介绍

实验设备：仿真器1 台，液晶板1 块，ZigBee 模块1 块，USB 连接线1 根。

本实验的控制比实验2 的控制稍显复杂，这个实验中使用按键控制LED 闪烁。按下“OK”键切换ZigBee 模块左边LED 灯闪烁，按下“CANCEL”键切换ZigBee 模块右边LED灯闪烁。

2、实验相关寄存器

实验中操作了的寄存器有P0，P0DIR，P0SEL，P1INP。前面三个寄存器在1.1.1 已经有详述，这里不再重复介绍。

P1　　参见1.1.1
P1SEL　参见1.1.1
P1DIR　参见1.1.1
P1INP　参见1.1.2

3、实验相关函数

void Delay(uint n)； 参见基础1.1.1。

void Initial(void)； 参见基础1.1.1。

void InitKey(void); 参见基础1.1.1。

unsigned char KeyScan(void); 参见基础1.1.2。

1.2 定时/计数器实验

1.2.1 CC2530 基础实验 4：T1 使用

1、实验介绍

用定时器1 来改变小灯的状态，T1 每溢出两次，两个小灯闪烁一次，并且在停止闪烁后成闪烁前相反的状态。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL，T1CTL 前面三个寄存器在实验1 已经有详述，这里不再重复介绍。

P1 见实验1 说明文档。

P1DIR 见实验1 说明文档。

P1SEL 见实验1 说明文档。

T1CTL（T1 控制&状态寄存器）

位号	位名	复位值	操作性	功能描述
7	CH2IF	0	读/写	定时器1 通道2 中断标志位
6	CH1IF	0	读/写	定时器1 通道1 中断标志位
5	CH0IF	0	读/写	定时器1 通道0 中断标志位
4	OVFIF	0	读/写	定时器溢出中断标志，在在计数器达到计数终值的时候置位
3:2	DIV[1:0]	00	读/写	定时器1 计数时钟分步选择 00：不分频 01：8 分频 10：32 分频 11：128 分频
1:0	MODE[1:0]	00	读/写	定时器1 模式选择 00：暂停 01：自动重装0x0000-0xffff 10：比较计数0x0000-T1CC0 11：PWM 方式

3、实验相关函数

void Delay(uint n)；参见CC2530 基础实验1。

void Initial(void)；函数原型：

```
void Initial(void)
```

```
{
    //初始化P1
    P1DIR = 0x03; //P10 P11 为输出
    RLED = 1;
    YLED = 1; //灭LED
    //用T1 来做实验
    T1CTL = 0x0d; // 通道0，128 分频;自动重载模式(0x0000->0xffff);
}
```

函数功能是将P10，P11 设为输出，并将定时器1 设为自动重装模式，计数时钟为0.25M。

1.2.2 CC2530 基础实验 5：T2 使用

1、实验介绍

用定时器2 来改变小灯的状态，T2 每发生一次中断小灯改变状态一次。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1SEL，P1DIR，T2CTRL，T2M0，T2IRQM 等寄存器。

P1 参见CC2530 实验1

P1SEL 参见CC2530 实验1

P1DIR 参见CC2530 实验1

T2CTRL（T2 配置寄存器）

位号	位名	复位值	操作性	功能描述
7：4	-	0	读	保留，读 0
3	LATCH_MODE	0	读/写	0：当T2MSEL.T2MSEL = 000 读T2M0，T2M1，T2MSEL.T2MOVFSEL=000。读T2MOVF0， T2MOVF1 T2MOVF2 。 1：当T2MSEL.T2MSEL = 000 读T2M0, T2M1，T2MOVF0 , T2MOVF1, aT2MOVF2
2	STATE	0	读	0 计数器空闲模式，1 计数器正常运行。
1	SYNC	1	读/写	同步使能 0：T2 立即起、停。 1：T2 起、停和32.768kHz 时钟及计数新值同步
0	RUN	0	读/写	启动T2，通过读出该位可以知道T2 的状态。 0：停止T2（IDLE）， 1：启动T2（RUN）

T2MOVF2 （T2 多路复用溢出计数器2 寄存器）

位号	位名	复位值	操作性	功能描述
7：0	CMPIM	0	读/写	T2MSEL.T2MOVFSEL=000，

				T2CTRL.LATCH_MODE =0 时，计数值被锁存。

T2M0 (（T2 多路复用寄存器）

位号	位名	复位值	操作性	功能描述
7:0	CMPIM	0	读/写	T2MSEL.T2MSEL=000 和 T2CTRL.LATCH_MODE=0 时，计数值被锁存。T2MSEL.T2MSEL=000和 T2CTRL.LATCH_MODE=1 时，计数值和溢出值被锁存。

T2IRQF（中断标志）

位号	位名	复位值	操作性	功能描述
7：6	-	0	读	保留
5	TIMER2_OVF_COMPARE2F	0	读/写	当溢出计数器计数达到 t2ovf_cmp2 的值时置位
4	TIMER2_OVF_COMPARE1F	0	读/写	当溢出计数器计数达到 t2ovf_cmp1 的值时置位
3	TIMER2_OVF_PERF	0	读/写	当溢出计数器计数等于 t2ovf_per 的值时置位
2	TIMER2_COMPARE2F	0	读/写	当计数器计数达到 t2_cmp2 的值时置位
1	TIMER2_COMPARE1F	0	读/写	当计数器计数达到 t2_cmp1 的值时置位
0	TIMER2_PERF	0	读/写	当计数器计数等于 t2_per 的值时置位

T2IRQM（中断屏蔽）

位号	位名	复位值	操作性	功能描述
7：6	-	0	读	保留
5	TIMER2_OVF_COMPARE2MF	0	读/写	TIMER2_OVF_COMPARE2M 中断使能
4	TIMER2_OVF_COMPARE1M	0	读/写	TIMER2_OVF_COMPARE1M 中断使能
3	TIMER2_OVF_PERM	0	读/写	TIMER2_OVF_PERM 中断使能
2	TIMER2_COMPARE2M	0	读/写	TIMER2_COMPARE2M 中断使能
1	TIMER2_COMPARE1M	0	读/写	TIMER2_COMPARE1M 中断使能

0	TIMER2_PERM	0	读/写	TIMER2_PERM 中断使能

3、实验相关函数

void Delay(uint n)；参见CC2530 基础实验1。

void Initial(void)；函数原型：

```
void Initial(void)
{
    LED_ENALBLE(); //启用LED
    //用T2 来做实验
    SET_TIMER2_OF_INT();    //开溢出中断
    SET_TIMER2_CAP_COUNTER(0X00ff);  //设置溢出值
}
```

函数功能是启用LED，使用LED 可控，开T2 比较中断

重要的宏定义

开启溢出中断

```
#define SET_TIMER2_CAP_INT() \
do{ \
    T2IRQM = 0x04; \
    EA = 1; \
    T2IE = 1; \
    T2MSEL |= 0xf4; \
}while(0)
```

设定溢出周期

```
#define SET_TIMER2_CAP_COUNTER(val) SET_WORD(T2M1,T2M0,val)
```

功能：将无符号整形数val 的高8 位写入T2CAPLPL，低8 位写入T2CAPHPH。

启动T2

```
#define TIMER2_RUN() T2CTRL|=0X01
```

停止T2

```
#define TIMER2_STOP() do{T2CTRL&=0XFE;}while(0)
```

1.2.3 CC2530 基础实验 6：T3 使用

1、实验介绍

用定时器3 来改变小灯的状态，T3 每发生200 次中断小灯改变状态一次。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1SEL，P1DIR，T3CTL，T3CCTL0，T3CC0，T3CCTL1，T3CC1，等寄存器。

P1 参见CC2530 实验1

P1SEL 参见CC2530 实验1

P1DIR 参见CC2530 实验1

T3CTL（T3 控制寄存器）

位号	位名	复位值	操作性	功能描述
7:5	DIV[2:0]	000	读/写	定时器时钟再分频数（对CLKCONCMD.TICKSPD分频后再次分频） 000 ：不再分频 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频
4	START	0	读/写	T3 起停位 0 暂停计数，1 正常运行
3	OVFIM	1	读/写	溢出中断掩码0 关溢出中断，1 开溢出中断
2	CLR	0	读/写	清计数值，写1 使T3CNT=0x00
1:0	MODE[1:0]	00	读/写	T3 模式选择00：自动重装 01：DOWN （从T3CC0 到0x00 计数一次） 10：模计数（反复从0x00 到T3CC0 计数） 11：UP/DOWN （反复从0x00 到T3CC0 再到0x00）

T3CCTL0 （T3 通道0 捕获/比较控制寄存器）

位号	位名	复位值	操作性	功能描述
7	—	0	读	预留
6	IM	1	读/写	通道0 中断掩码0 关中断，1 开中断
5:3	CMP[7:0]	000	读/写	通道0 比较输出模式选择，指定计数值过T3CC0 时的发生事件 000 输出置1（发生比较时），001 输出清0（发生比较时），010 输出翻转， 011 输出置1（发生上比较时） 输出清0 （计数值为0 或UP/DOWN 模式下发生下比较）， 100 输出清0 （发生上比较时）输出置1（计数值为0或UP/DOWN 模式下发生下比较）， 101 输出置1 （发生比较时）输出清0（计数值为0xff时）， 110 输出清0 （发生比较时）输出置1 （ 计数值为0x00 时），111 预留。
2	MODE-	0	读/写	T3 通道0 模式选择0 捕获，1 比较
1:0	CAP	00	读/写	T3 通道0 捕获模式选择 00 没有捕获

				01 上升沿捕获 10 下降沿捕获 11 边沿捕获

T3CC0 （T3 通道0 捕获/比较值寄存器）

位号	位名	复位值	操作性	功能描述
7:0	VAL[7:0]	0x00	读/写	T3 通道0 比较/捕获值

T3CCTL1 （T3 通道1 捕获/比较控制寄存器）

位号	位名	复位值	操作性	功能描述
7	—	0	读	预留
6	IM	1	读/写	通道1 中断掩码 0 关中断，1 开中断
5:3	CMP[7:0]	0	读/写	通道1 比较输出模式选择，指定计数值过T3CC0 时的发生事件 000 输出置1（发生比较时） 001 输出清0（发生比较时） 010 输出翻转 011 输出置1（发生上比较时）输出清0（计数值为0 或UP/DOWN 模式下发生下比较） 100 输出清0（发生上比较时）输出置1（计数值为0 或UP/DOWN 模式下发生下比较） 101 输出置1(发生比较时)输出清0（计数值为0xff 时） 110 输出清0(发生比较时)输出置1(计数值为0x00 时） 111 预留
2	MODE-	0	读/写	T3 通道1 模式选择 0 捕获，1 比较
1:0	CAP	0000	读/写	T3 通道1 捕获模式选择 00 没有捕获 01 上升沿捕获 10 下降沿捕获 11 边沿捕获

T3CC1 （T3 通道1 捕获/比较值寄存器）

位号	位名	复位值	操作性	功能描述
7:0	VAL[7:0]	0x00	读/写	T3 通道1 比较/捕获值

3、实验相关函数

void Init_T3_AND_LED(void)；函数原型：

```
void Init_T3_AND_LED(void)
{
    P1DIR = 0X03;
    RLED = 1;
    YLED = 1;
    TIMER34_INIT(3); //初始化T3
    TIMER34_ENABLE_OVERFLOW_INT(3,1); //开T3 中断
    //时钟32 分频101
    TIMER3_SET_CLOCK_DIVIDE(16);
    TIMER3_SET_MODE(T3_MODE_FREE); //自动重装00－>0xff
    TIMER3_START(1); //启动
}
```

函数功能：将I/0 P10,P11 设置为输出去控制LED，将T3 设置为自动重装模式，定时器时钟16 分频，并启动T3。

void T3_ISR(void)；函数原型：

```
#pragma vector = T3_VECTOR
__interrupt void T3_ISR(void)
{
  //IRCON = 0x00; //清中断标志,硬件自动完成
  if(counter<200)  counter++; //10 次中断LED 闪烁一轮
  else
  {
    counter = 0; //计数清零
    RLED = !RLED; //改变小灯的状态
  }
}
```

函数功能：这是一个中断服务程序，每200 次中断改变一次红色LED 的状态。

重要的宏定义

开启溢出中断

```
#define TIMER34_ENABLE_OVERFLOW_INT(timer,val)                          \
do{T##timer##CTL = (val) ? T##timer##CTL | 0x08 : T##timer##CTL & ~0x08;    \
  EA = 1;                                                               \
  T3IE = 1;                                                             \
}while(0)
```

功能：打开T3 的溢出中断。

复位T3 相关寄存器

```
#define TIMER34_INIT(timer) \
do { \
  T##timer##CTL = 0x06;          \//比较模式
  T##timer##CCTL0 = 0x00;      \
  T##timer##CC0 = 0x00;          \
  T##timer##CCTL1 = 0x00;      \
  T##timer##CC1 = 0x00;          \
} while (0)
```

功能：将T3 相关的寄存器复位到0

控制T3 起停

```
#define TIMER3_START(val) \
    (T3CTL = (val) ? T3CTL | 0X10 : T3CTL&~0X10)
```

功能：val 为1，T3 正常运行，val 为0，T3 停止计数

设置T3 工作方式

```
#define TIMER3_SET_MODE(val) \
do{                                          \
  T3CTL &= ~0X03;                            \
  (val==1)?(T3CTL|=0X01): /*DOWN */          \
  (val==2)?(T3CTL|=0X02): /*Modulo */        \
  (val==3)?(T3CTL|=0X03): /*UP / DOWN */     \
  (T3CTL|=0X00); /*free runing */            \
}while(0)
#define T3_MODE_FREE 0X00
#define T3_MODE_DOWN 0X01
#define T3_MODE_MODULO 0X02
#define T3_MODE_UP_DOWN 0X03
```

功能：根据val 的值将T3 设置为不同模式，一共4 种模式。

1.2.4 CC2530 基础实验 7：T4 使用

1、实验介绍

用定时器4 来改变小灯的状态，T4 每发生200 次中断小灯改变状态一次。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1SEL，P1DIR，T4CTL，T4CCTL0，T4CC0，T4CCTL1，T4CC1等寄存器。

P1 参见CC2530 实验1。

P1SEL 参见CC2530 实验1。

P1DIR 参见CC2530 实验1。

T4CTL（T4 控制寄存器）

位号	位名	复位值	操作性	功能描述
7:5	DIV[2:0]	000	读/写	定时器时钟再分频数（对CLKCONCMD.TICKSPD 分频后再次分频） 000 不再分频 001 2 分频 010 4 分频 011 8 分频 100 16 分频 101 32 分频

				110 64 分频 111 128 分频
4	START	0	读/写	T4 起停位0 暂停计数，1 正常运行
3	OVFIM	1	读/写0	溢出中断掩码0 关溢出中断，1 开溢出中断
2	CLR	0	R0/W1	清计数值，写1 使T4CNT=0x00
1:0	MODE[1:0]	00	读/写	T4 模式选择 00 自动重装 01 DOWN （从T4CC0 到0x00 计数一次） 10 模计数（反复从0x00 到T4CC0 计数） 11 UP/DOWN （反复从0x00 到T4CC0 再到0x00）

T4CCTL0 （T4 通道0 捕获/比较控制寄存器）

位号	位名	复位值	操作性	功能描述
7	—	0	读	预留
6	IM	1	读/写	通道0 中断掩码 0 关中断，1 开中断
5:3	CMP[7:0]	000	读/写	通道0 比较输出模式选择，指定计数值过T4CC0 时的发生事件 000 输出置1（发生比较时） 001 输出清0（发生比较时） 010 输出翻转 011 输出置1（发生上比较时）输出清0（计数值为0 或UP/DOWN 模式下发生下比较） 100 输出清0（发生上比较时）输出置1（计数值为0 或UP/DOWN 模式下发生下比较） 101 输出置1（发生比较时）输出清0（计数值为0xff 时） 110 输出清0（发生比较时）输出置1（计数值为0x00 时） 111 预留
2	MODE-	0	读/写	T4 通道0 模式选择 0 捕获，1 比较
1:0	CAP	00	读/写	T4 通道0 捕获模式选择 00 没有捕获 01 上升沿捕获 10 下降沿捕获 11 边沿捕获

T4CC0 （T4 通道0 捕获/比较值寄存器）

位号	位名	复位值	操作性	功能描述
7:0	VAL[7:0]	0x00	读/写	T4 通道0 比较/捕获值

T4CCTL1 （T4 通道1 捕获/比较控制寄存器）

位号	位名	复位值	操作性	功能描述
7	—	0	读	预留
6	IM	1	读/写	通道1 中断掩码 0 关中断，1 开中断
5:3	CMP[7:0]	0	读/写	通道1 比较输出模式选择，指定计数值过T4CC0 时的发生事件 000 输出置1（发生比较时） 001 输出清0（发生比较时） 010 输出翻转 011 输出置1（发生上比较时）输出清0（计数值为0 或UP/DOWN 模式下发生下比较） 100 输出清0（发生上比较时）输出置1（计数值为0 或UP/DOWN 模式下发生下比较） 101 输出置1（发生比较时）输出清0（计数值为0xff 时） 110 输出清0（发生比较时）输出置1（计数值为0x00 时） 111 预留
2	MODE-	0	读/写	T4 通道1 模式选择 0 捕获，1 比较
1:0	CAP	0000	读/写	T4 通道1 捕获模式选择

T4CC1 （T4 通道1 捕获/比较值寄存器）

位号	位名	复位值	操作性	功能描述
7:0	VAL[7:0]	0x00	读/写	T4 通道1 比较/捕获值

3、实验相关函数

void Init_T4_AND_LED(void)函数原型：

```
void Init_T4_AND_LED(void)
{
    P1DIR = 0X03;
    led1 = 1;
    led2 = 1;
    TIMER34_INIT(4); //初始化T4
    TIMER34_ENABLE_OVERFLOW_INT(4,1); //开T4 中断
    TIMER34_SET_CLOCK_DIVIDE(4,128);
    TIMER34_SET_MODE(4,0); //自动重装00－>0xff
    TIMER34_START(4,1); //启动
}
```

函数功能：将I/0 P10,P11 设置为输出去控制LED，将T4 设置为自动重装模式，定时器时钟16 分频，并启动T4。

void T4_ISR(void)；函数原型：

```
#pragma vector = T4_VECTOR
__interrupt void T4_ISR(void)
{
  //IRCON = 0x00; //清中断标志,硬件自动完成
  if(counter<200)  counter++; //10 次中断LED 闪烁一轮
  else
  {
    counter = 0; //计数清零
    led1 = !led1; //改变小灯的状态
  }
}
```

函数功能：这是一个中断服务程序，每200 次中断改变一次红色LED 的状态。

重要的宏定义

开启溢出中断

```
#define TIMER34_ENABLE_OVERFLOW_INT(timer,val)                    \
do{T##timer##CTL = (val) ? T##timer##CTL | 0x08 : T##timer##CTL & ~0x08;   \
  EA = 1;                                                         \
  T4IE = 1;                                                       \
}while(0)
```

功能：打开T4 的溢出中断。

复位T4 相关寄存器

```
#define TIMER34_INIT(timer) \
do {                                    \
  T##timer##CTL = 0x06;                 \
  T##timer##CCTL0 = 0x00;               \
  T##timer##CC0 = 0x00;                 \
  T##timer##CCTL1 = 0x00;               \
  T##timer##CC1 = 0x00;                 \
} while (0)
```

功能：将T4 相关的寄存器复位到0

控制T4 起停

```
#define TIMER34_START(timer,val) \
    (T##timer##CTL = (val) ? T##timer##CTL | 0X10 : T##timer##CTL&~0X10)
```

功能：timer 为定时器序号，只能取3 或4。val 为1，定时器正常运行，val 为0，定时器停止计数

设置T4 工作方式

```
#define TIMER3_SET_MODE(val)                  \
do{                                           \
  T4CTL &= ~0X03;                             \
  (val==1)?(T4CTL|=0X01): /*DOWN */           \
  (val==2)?(T4CTL|=0X02): /*Modulo */         \
```

```
  (val==3)?(T4CTL|=0X03): /*UP / DOWN */          \
  (T4CTL|=0X00); /*free runing */                 \
}while(0)
#define T4_MODE_FREE 0X00
#define T4_MODE_DOWN 0X01
#define T4_MODE_MODULO 0X02
#define T4_MODE_UP_DOWN 0X03
```

功能：根据val 的值将T4 设置为不同模式，一共4 种模式。

1.3 中断实验

1.3.1 CC2530 基础实验 8：定时器中断

1、实验介绍

用定时器4 来改变小灯的状态，T4 每2000 次中断小灯闪烁一轮，闪烁的时间长度为1000 次中断所耗时间。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1SEL，P1DIR，T4CTL，T4CCTL0，T4CC0，T4CCTL1，T4CC1，IEN0，IEN1 等寄存器。

P1 参见CC2530 实验1

P1SEL 参见CC2530 实验1

P1DIR 参见CC2530 实验1

T4CTL 参见CC2530 实验7

T4CCTL0 参见CC2530 实验7

T4CC0 参见CC2530 实验7

T4CCTL1 参见CC2530 实验7

T4CC1 参见CC2530 实验7

3、实验相关函数

void Init_T4_AND_LED(void)；函数原型：

```
void Init_T4_AND_LED(void)
{
  P1DIR = 0X03;
  led1 = 1;
  led2 = 1;
  TIMER34_INIT(4); //初始化T4
  TIMER34_ENABLE_OVERFLOW_INT(4,1); //开T4 中断
  EA = 1;
  T4IE = 1;
  TIMER34_SET_CLOCK_DIVIDE(4,128);
  TIMER34_SET_MODE(4,0); //自动重装00－>0xff
  T4CC0 = 0Xf0;
  TIMER34_START(4,1); //启动
}
```

函数功能：将I/0 P10,P11 设置为输出去控制LED，将T4 设置为自动重装模式，定时器时钟16 分频，并启动T4。

void T4_ISR(void)；函数原型：

```
#pragma vector = T4_VECTOR
__interrupt void T4_ISR(void)
{
    IRCON = 0x00; //可不清中断标志,硬件自动完成
    if(counter<1000)          counter++; //1000 次中断LED 闪烁一轮
    else
    {
        counter = 0; //计数清零
        GlintFlag = !GlintFlag; //GlintFalg = 1，LED 闪烁
    }
}
```

函数功能：这是一个中断服务程序，每1000 次中断改变一次红色LED 的状态。

重要的宏定义

开启溢出中断

```
#define TIMER34_ENABLE_OVERFLOW_INT(timer,val) \
do{T##timer##CTL = (val) ? T##timer##CTL | 0x08 : T##timer##CTL & ~0x08; \
}while(0)
```

功能：打开T4 的溢出中断。

复位T4 相关寄存器

```
#define TIMER34_INIT(timer) \
do { \
  T##timer##CTL = 0x06; \
  T##timer##CCTL0 = 0x00; \
  T##timer##CC0 = 0x00; \
  T##timer##CCTL1 = 0x00; \
  T##timer##CC1 = 0x00; \
} while (0)
```

功能：将T4 相关的寄存器复位到0

控制T4 起停

```
#define TIMER#define TIMER34_START(timer,val) \
    (T##timer##CTL = (val) ? T##timer##CTL | 0X10 : T##timer##CTL&~0X10)
```

功能：timer 为定时器序号，只能取3 或4。val 为1，定时器正常运行，val 为0，定时器停止计数。

设置T4 工作方式

```
#define TIMER3_SET_MODE(val)                    \
do{                                             \
  T4CTL &= ~0X03;                               \
  (val==1)?(T4CTL|=0X01): /*DOWN */             \
  (val==2)?(T4CTL|=0X02): /*Modulo */           \
  (val==3)?(T4CTL|=0X03): /*UP / DOWN */        \
  (T4CTL|=0X00); /*free runing */               \
```

```
}while(0)
#define T4_MODE_FREE 0X00
#define T4_MODE_DOWN 0X01
#define T4_MODE_MODULO 0X02
#define T4_MODE_UP_DOWN 0X03
```

功能：根据val 的值将T4 设置为不同模式，一共4 种模式。

1.3.2 CC2530 基础实验 9：外部中断

1、实验介绍

使用两个按键来翻转LED的状态，但这里两个按键不是做键盘用，而是产生中断触发信号。按下CC2000系列板上“OK”键，CC2530 模块上1个LED 灯改变当前状态。

2、实验相关寄存器

实验中操作了的寄存器有P0，P0SEL，P0DIR，P0INP，P0IEN，P0CTL，IEN2，P0IFG等寄存器。

P1 参见1.1.1

P1SEL 参见1.1.1

P1DIR 参见1.1.1

P1INP 参见1.1.1

P0IEN （P01 口中断掩码）

位号	位名	复位值	可操作性	功能描述
7	P0_7IEN	0	读/写	P07 中断掩码 0 关中断，1 开中断
6	P0_6IEN	0	读/写	P06 中断掩码 0 关中断，1 开中断
5	P0_5IEN	0	读/写	P05 中断掩码 0 关中断，1 开中断
4	P0_4IEN	0	读/写	P04 中断掩码 0 关中断，1 开中断
3	P0_3IEN	0	读/写	P03 中断掩码 0 关中断，1 开中断
2	P0_2IEN	0	读/写	P02 中断掩码 0 关中断，1 开中断
1	P0_1IEN	0	读/写	P01 中断掩码 0 关中断，1 开中断
0	P0_0IEN	0	读/写	P00 中断掩码 0 关中断，1 开中断

PICTL （P 口中断控制寄存器）

位号	位名	复位值	可操作性	功能描述
7	—	0	读	预留

6	PADSC	0	读/写	输出驱动能力选择0 最小驱动能力，1 最大驱动能力
5	P2IEN	0	读/写	P2（0-4）中断使能位0 关中断，1 开中断
4	P0IENH	0	读/写	P0（4-7）中断使能位0 关中断，1 开中断
3	P0IENL	0	读/写	P0（0-3）中断使能位0 关中断，1 开中断
2	P2ICON	0	读/写	P2 （0-4） 中断配置0 上升沿触发，1 下降沿触发
1	P1ICON	0	读/写	P1 （0-7） 中断配置0 上升沿触发，1 下降沿触发
0	P0ICON	0	读/写	P0 （0-7） 中断配置0 上升沿触发，1 下降沿触发

P0IFG（P0 口中断标志寄存器）

位号	位名	复位值	可操作性	功能描述
7：0	P0IF[7:0]	0x00	读/写	P0（0-7） 中断标志位，在中断条件发生，相应位自动置1

IEN2 （中断使能寄存器2）

位号	位名	复位值	可操作性	功能描述
7:6	—	00	读	没有，读出为0
5	WDTIE	0	读/写	看门狗定时器中断使能 0 关中断，1 开中断
4	P1IE	0	读/写	P1 中断使能使能 0 关中断，1 开中断
3	UTX1IE	0	读/写	串口1 发送中断使能 0 关中断，1 开中断
2	UTX0IE	0	读/写	串口0 发送中断使能 0 关中断，1 开中断
1	P2IE	0	读/写	P2 口中断使能 0 关中断，1 开中断
0	RFIE	0	读/写	普通射频中断使能 0 关中断，1 开中断

3、实验相关函数

void Init_IO_AND_LED(void)；函数原型：

```
void Init_IO_AND_LED(void)
{
    P1SEL &= ~0x03;//P1.0,P1.1为IO口
```

```
    P1DIR |= 0x03; //P1.0,P1.1为输出
    RLED = 0;
    GLED = 0;

    P0SEL &= ~0x30;//P0.4,P0.5为IO口
    P0DIR &= ~0x30;//P0.4,P0.5为输入方式
    P0INP &= ~0x30;
    PICTL |= 0x11; //P0.4,P0.5为下降沿

    P0IFG = 0;//P04 P05中断标志清0
    P0IEN |= 0x30;//端口0(P0.4,P0.5)中断使能
    P0IF = 0; //IRCON中端口0中断标志,0为中断未挂起,1为中断挂起
    EA = 1;
    P0IE = 1;        //端口0中断使能
}
```

函数功能：将I/0 P04，P05 设置为输出去控制LED，使能P0 中断 且配置为下降沿触发。

void P1_ISR(void)； 函数原型：

```
#pragma vector = P0INT_VECTOR
__interrupt void P0_ISR(void)
{
  if(P0IFG>0) //按键中断
  {
    P0IFG = 0;
    RLED = !RLED;
  }
  P0IF = 0; //清中断标志
}
```

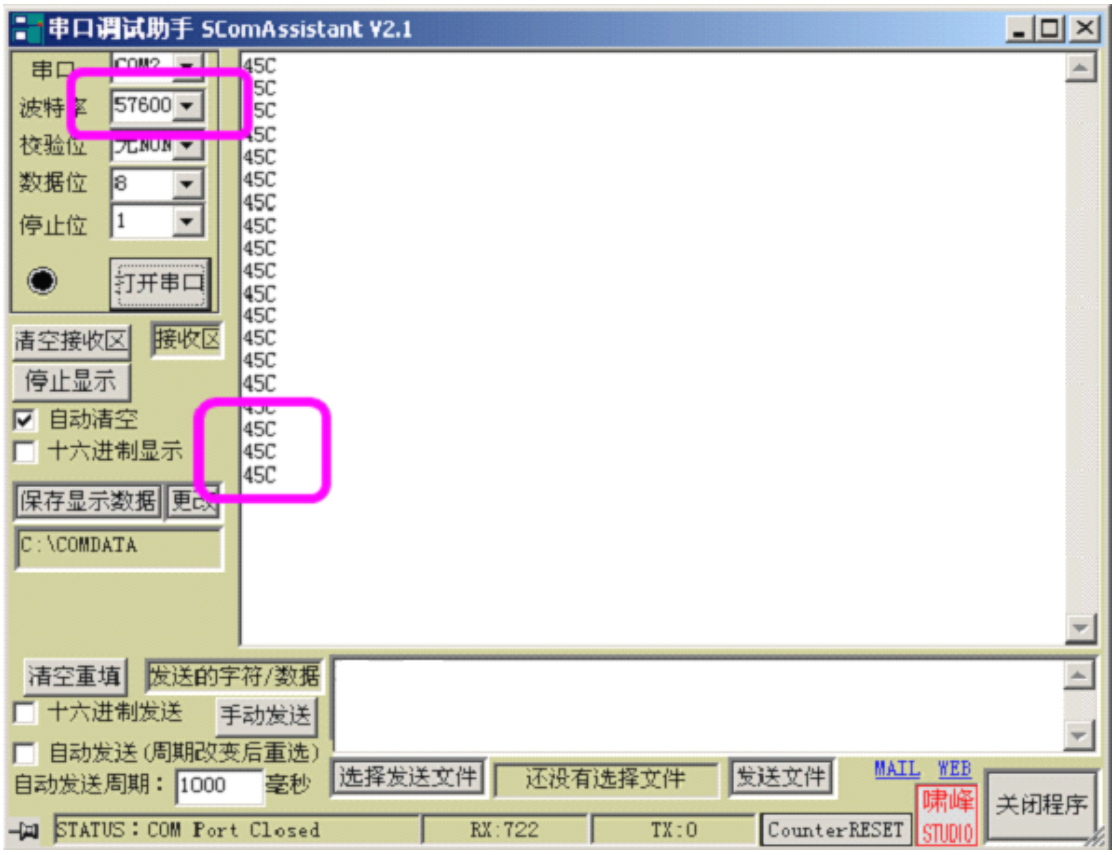
函数功能：在P04，P05 触发中断的时候将绿色LED 的状态翻转。

1.4 AD 实验

1.4.1 CC2530 基础实验 10：片内温度

1、实验介绍

取片内温度传感器为AD 源，并将转换得到温度通过串口送至电脑。



2、实验相关寄存器

实验中操作了的寄存器有CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，CLKCONSTA，IEN0，U0DUB，ADCCON1，ADCCON3，ADCH，ADCL 等寄存器。

IEN0 参见实验5。

CLKCONCMD （时钟控制寄存器）

位号	位名	复位值	可操作性	功能描述
7	OSC32K	1	写	32kHz 时钟源选择 0：32K 晶振，1：32K RC 振荡
6	OSC	1	写	主时钟源选择 0：32M 晶振，1：16M RC 振荡
5:3	TICKSPD[2:0]	001	写	定时器计数时钟分频（该时钟频不大于 OSC 决定频率） 000 32M 001 16M 010 8M 011 4M 100 2M 101 1M 110 0.5M 111 0.25M
2:0	CLKSPD	001	写	时钟速率，不能高于系统时钟

				000: 32 MHz 001: 16 MHz 010: 8 MHz 011: 4 MHz 100: 2 MHz 101: 1 MHz 110: 500 kH 111: 250 kH

CLKCONSTA （时钟状态寄存器）

位号	位名	复位值	可操作性	功能描述
7	OSC32K	1	读	32kHz 时钟源选择 0：32K 晶振，1：32K RC 振荡
6	OSC	1	读	主时钟源选择 0：32M 晶振， 1：16M RC 振荡
5:3	TICKSPD[2:0]	001	读	定时器计数时钟分频（该时钟频不大于OSC决定频率） 000 32M 001 16M 010 8M 011 4M 100 2M 101 1M 110 0.5M 111 0.25M
2:0	CLKSPD	001	读	时钟速率，不能高于系统时钟 000: 32 MHz 001: 16 MHz 010: 8 MHz 011: 4 MHz 100: 2 MHz 101: 1 MHz 110: 500 kH 111: 250 kH

SLEEPCMD （睡眠模式控制寄存器）

位号	位名	复位值	可操作性	功能描述
7	—	0	读	预留
6	XOSC_STB	0	写	低速时钟状态0 没有打开或者不稳定1 打开且稳定

5	HFRC_STB	0	写	主时钟状态0 没有打开或者不稳定1 打开且稳定
4:3	RST[1:0]	XX	写	最后一次复位指示00 上电复位01 外部复位10 看门狗复位
2	OSC_PD	0	写	节能控制，OSC 状态改变的时候硬件清0。0 不关闭无用时钟1 关闭无用时钟
1:0	MODE[1:0]	0	写	功能模式选择00 PM0 01 PM1 10 PM2 11 PM3

PERCFG（外设控制寄存器）

位号	位名	复位值	可操作性	功能描述
7	—	0	读	预留
6	T1CFG	0	读/写	T1 I/O 位置选择 0 位置1，1 位置2
5	T3CFG	0	读/写	T3 I/O 位置选择 0 位置1，1 位置2
4	T4CFG	0	读/写	T4 I/O 位置选择0 位置1，1 位置2
3:2	—	00	R0	预留
1	U1CFG	0	读/写	串口1 位置选择0 位置1，1 位置2
0	U0CFG	0	读/写	串口0 位置选择0 位置1，1 位置2

U0CSR（串口0 控制&状态寄存器）

位号	位名	复位值	可操作性	功能描述
7	MODE	0	读/写	串口模式选择 0 SPI 模式，1 UART 模式
6	RE	0	读/写	接收使能 0 关闭接收，1 允许接收
5	SLAVE	0	读/写	SPI 主从选择 0 SPI 主，1 SPI 从

4	FE	0	读/写	串口帧错误状态 0 没有帧错误，1 出现帧错误
3	ERR	0	读/写	串口校验结果 0 没有校验错误，1 字节校验出错
2	RX_BYTE	0	读/写	接收状态 0 没有接收到数据，1 接收到一字节数据
1	TX_BYTE	0	读/写	发送状态 0 没有发送，1 最后一次写入U0BUF的数据已经发送
0	ACTIVE	0	读	串口忙标志 0 串口闲，1 串口忙

U0GCR （串口0 常规控制寄存器）

位号	位名	复位值	可操作性	功能描述
7	CPOL	0	读/写	SPI 时钟极性 0 低电平空闲，1 高电平空闲
6	CPHA	0	读/写	SPI 时钟相位 0 由CPOL 跳向非CPOL 时采样，由非CPOL 跳向CPOL 时输出 1 由非CPOL 跳向CPOL 时采样，由CPOL 跳向非CPOL 时输出
5	ORDER	0	读/写	传输位序 0 低位在先，1 高位在先
4:0	BAUD_E[4:0]	0x00	读/写	波特率指数值，BAUD_M 决定波特率

U0BAUD （串口0 波特率控制寄存器）

位号	位名	复位值	可操作性	功能描述
7:0	BAUD_M[7:0]	0X00	读/写	波特率尾数，与BAUD_E 决定波特率

U0BUF（串口0 收发缓冲器）

位号	位名	复位值	可操作性	功能描述
7:0	DATA[7:0]	0X00	读/写	UART0 收发寄存器

ADCCON1

位号	位名	复位值	可操作性	功能描述
7	EOC	0	读/写	ADC 结束标志位 0 ADC 进行中，1 ADC 转换结束

6	ST	0	读/写	手动启动AD 转换（读1 表示当前正在进行AD 转换） 0 没有转换，1 启动AD 转换 （STSEL=11）
5:4	STSEL[1:0]	11	读/写	AD 转换启动方式选择 00 外部触发 01 全速转换，不需要触发 10 T1 通道0 比较触发 11 手工触发
3:2	RCTRL[1:0]	00	读/写	16 位随机数发生器控制位（写01，10 会在执行后返回00） 00 普通模式（13x 打开） 01 开启LFSR 时钟一次 10 生成调节器种子 11 信用随机数发生器
1:0	-	11	读/写	保留，总是写设置为1

ADCCON3

位号	位名	复位值	可操作性	功能描述
7:6	SREF[1:0]	00	读/写	选择单次AD 转换参考电压 00 内部1.25V 电压 01 外部参考电压AIN7 输入 10 模拟电源电压 11 外部参考电压AIN6-AIN7 输入
5:4	SDIV[1:0]	01	读/写	选择单次A/D 转换分辨率 00 8 位 (64dec) 01 10 位 (128dec) 10 12 位 (256dec) 11 14 位 (512dec)
3:0	SCH[3:0]	00	读/写	单次A/D 转换选择，如果写入时ADC正在运行，则在完成序列A/D 转换后立刻开始，否则写入后立即开始A/D转换，转换完成后自动清0 0000 AIN0 0001 AIN1 0010 AIN2 0011 AIN3 0100 AIN4 0101 AIN5 0110 AIN6 0111 AIN7 1000 AIN0- AIN1 1001 AIN2- AIN3 1010 AIN4- AIN5

				1011 AIN6- AIN7 1100 GND 1101 正电源参考电压 1110 温度传感器 1111 1/3 模拟电压

3、实验相关函数

void Delay(uint n)；定性延时，参见实验1

void initUARTtest(void)；函数原型：

```
void initUARTtest(void)
{
    CLKCONCMD &= ~0x40; //晶振
    while(!(SLEEPSTA & 0x40)); //等待晶振稳定
    CLKCONCMD &= ~0x47; //TICHSPD128 分频，CLKSPD 不分频
    SLEEPCMD |= 0x04; //关闭不用的RC 振荡器
    PERCFG = 0x00; //位置1 P0 口
    P0SEL = 0x3c; //P0 用作串口
    U0CSR |= 0x80; //UART 方式
    U0GCR |= 10; //baud_e = 10;
    U0BAUD |= 216; //波特率设为57600
    UTX0IF = 1;
    U0CSR |= 0X40; //允许接收
    IEN0 |= 0x84; //开总中断，接收中断
}
```

函数功能： 将系统时钟设为高速晶振，将P0 口设置为串口0 功能引脚，串口0 使用UART模式，波特率设为57600，允许接收。在使用串口之前调用。

void UartTX_Send_String(char *Data,int len)函数原型：

```
void UartTX_Send_String(char *Data,int len)
{
  int j;
  for(j=0;j<len;j++)
  {
    U0DBUF = *Data++;
    while(UTX0IF == 0);
    UTX0IF = 0;
  }
}
```

函数功能：串口发送数据， *data 为发送缓冲的指针，len 为发送数据的长度，在初始化串口后才可以正常调用。

void initTempSensor(void)函数原型：

```
void initTempSensor(void){
  DISABLE_ALL_INTERRUPTS();
  SET_MAIN_CLOCK_SOURCE(0);
  *((BYTE __xdata*) 0xDF26) = 0x80;
```

```
}
```

函数功能：将系统时钟设为晶振，设AD 目标为片机温度传感器。

INT8 getTemperature(void)函数原型：

```
INT8 getTemperature(void){
  UINT8 i;
  UINT16 accValue;
  UINT16 value;
  accValue = 0;
  for( i = 0; i < 4; i++ )
  {
    ADC_SINGLE_CONVERSION(ADC_REF_1_25_V | ADC_14_BIT |
ADC_TEMP_SENS);
    ADC_SAMPLE_SINGLE();
    while(!ADC_SAMPLE_READY());
    value = ADCL >> 2;
    value |= (((UINT16)ADCH) << 6);
    accValue += value;
  }
  value = accValue >> 2; // devide by 4
  return ADC14_TO_CELSIUS(value);
}
```

函数功能：连续进行4次AD转换，将得到的结果求均值后将AD 结果转换为温度返回。

重要的宏定义

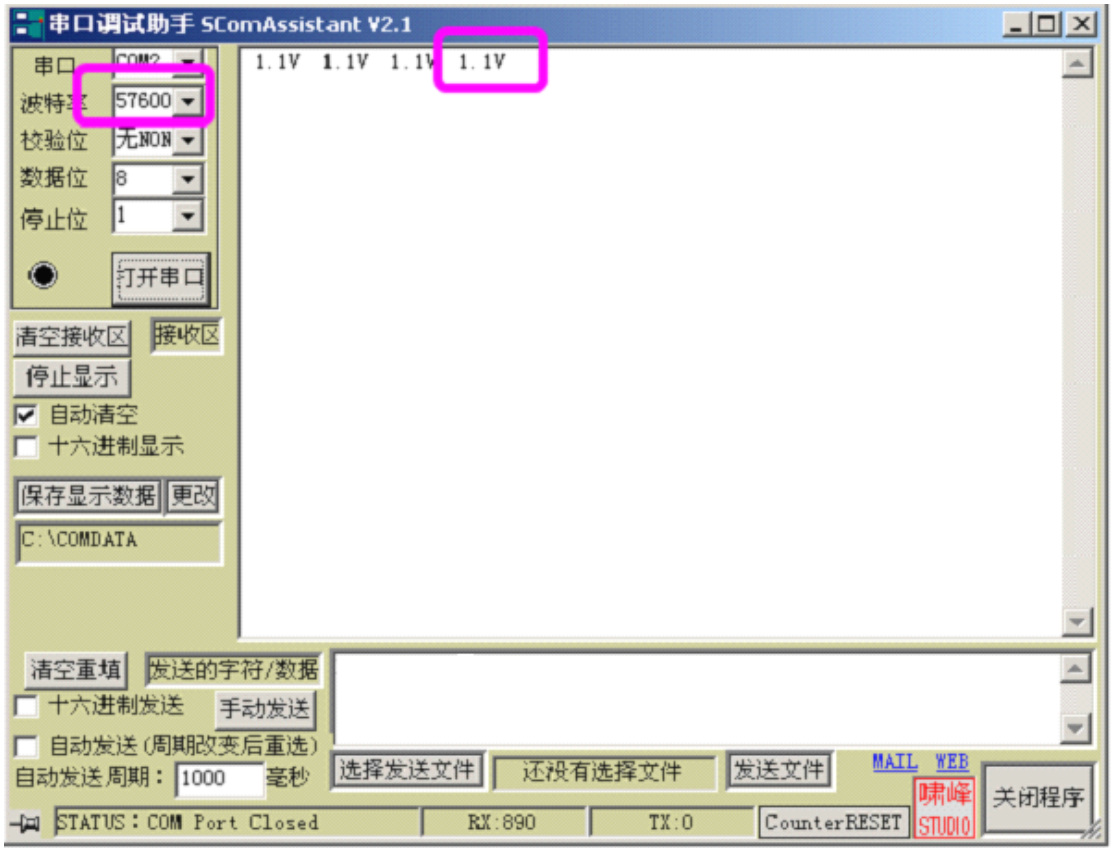
将片内温度传感器AD 转换的结果转换成温度。

```
#define ADC14_TO_CELSIUS(ADC_VALUE) ( ((ADC_VALUE) >> 4) - 315)
```

1.4.2 CC2530 基础实验 11：1/3AVDD

1、实验介绍

将AD 的源设为1/3 电源电压，并将转换得到温度通过串口送至电脑。



2、实验相关寄存器

实验中操作了的寄存器有CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，IEN0，U0DUB，ADCCON1，ADCCON3，ADCH，ADCL 等寄存器。

CLKCONCMD 参见实验10
SLEEPCMD 参见实验10
PERCFG 参见实验10
U0CSR 参见实验10
U0GCR 参见实验10
U0BAUD 参见实验10
U0BUF 参见实验10
ADCCON1 参见实验10
ADCCON3 参见实验10
IEN0 参见实验5

3、实验相关函数

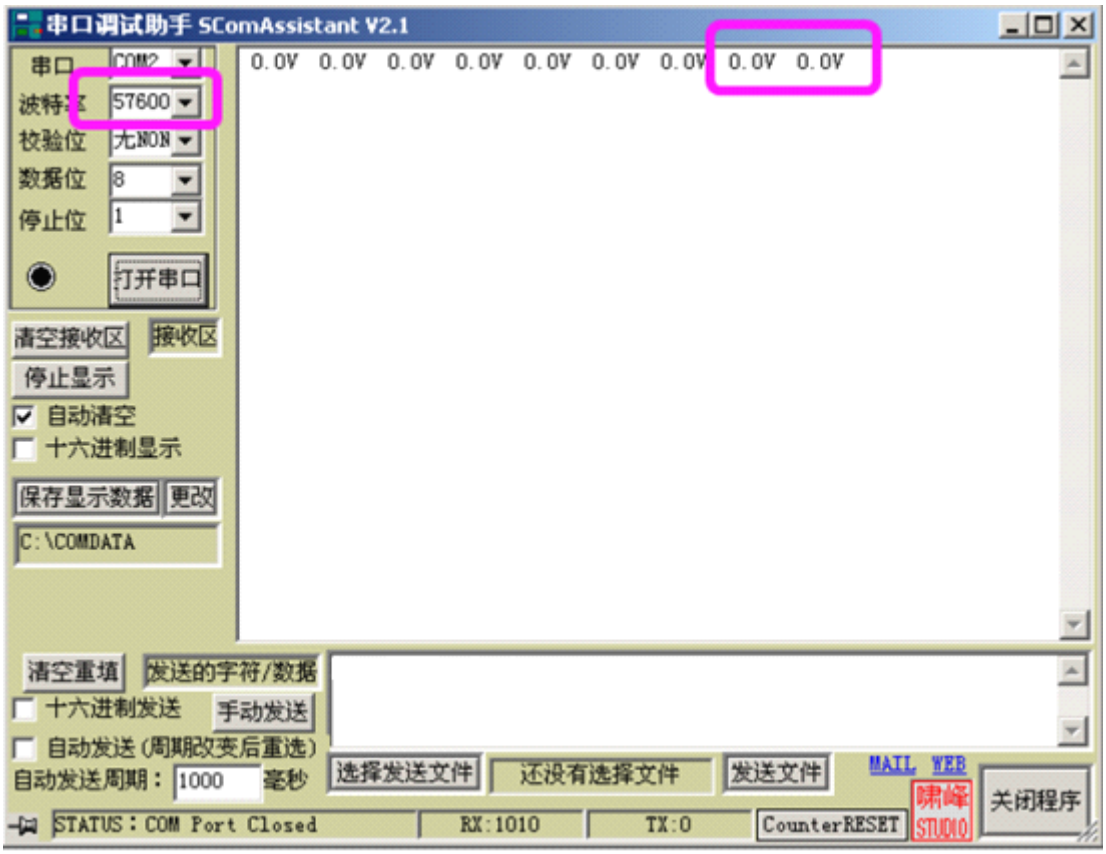
void Delay(uint n)；定性延时，参见实验1
void initUARTtest(void)；参见实验10
void UartTX_Send_String(char *Data,int len)；参见实验10
void InitialAD(void)；函数原型：

```
void InitialAD(void)
{
    //P1 out
    P1DIR = 0x03; //P1  控制LED
    led1 = 1;
```

```
    led2 = 1; //关LED
    ADCH &= 0X00; //清EOC 标志
    ADCCON3=0xbf; //单次转换,参考电压为电源电压，对1/3 AVDD 进行A/D 转换
    //14位分辨率
    ADCCON1 = 0X30; //停止A/D
    ADCCON1 |= 0X40; //启动A/D;
}
```

函数功能：将AD 转换源设为电源电压，ADC 结果分辨率设为14 位（最高精度），AD 模式为单次转换，启动ADC 转换。

1.4.3 CC2530 基础实验 12：AVDD



1、实验介绍

将 AD 的源设为电源电压，AD 参考电压为 AVDD，并将转换得到温度通过串口送至电脑。

2、实验相关寄存器

实验中操作了的寄存器有 CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，IEN0，U0DUB，ADCCON1，ADCCON3，ADCH，ADCL 等寄存器。

CLKCONCMD 参见实验 10

SLEEPCMD 参见实验 10

PERCFG 参见实验 10

U0CSR 参见实验 10

U0GCR 参见实验 10

U0BAUD 参见实验 10
U0BUF 参见实验 10
ADCCON1 参见实验 10
ADCCON3 参见实验 10
IEN0 参见实验 5

3、实验相关函数

void Delay(uint n)；定性延时，参见实验 1
void initUARTtest(void)；参见实验 10
void UartTX_Send_String(char *Data,int len)；参见实验 10
void InitialAD(void)；函数原型：

```
void InitialAD(void)
{
    //P1 out
    P1DIR = 0x03; //P1 控制 LED
    led1 = 1;
    led2 = 1; //关 LED
    ADCH &= 0X00; //清 EOC 标志
    ADCCON3=0xb7; //单次转换,参考电压为电源电压，对 AVDD 进行 A/D 转换
    //14 位分辨率
    ADCCON1 = 0X30; //停止 A/D
    ADCCON1 |= 0X40; //启动 A/D
}
```

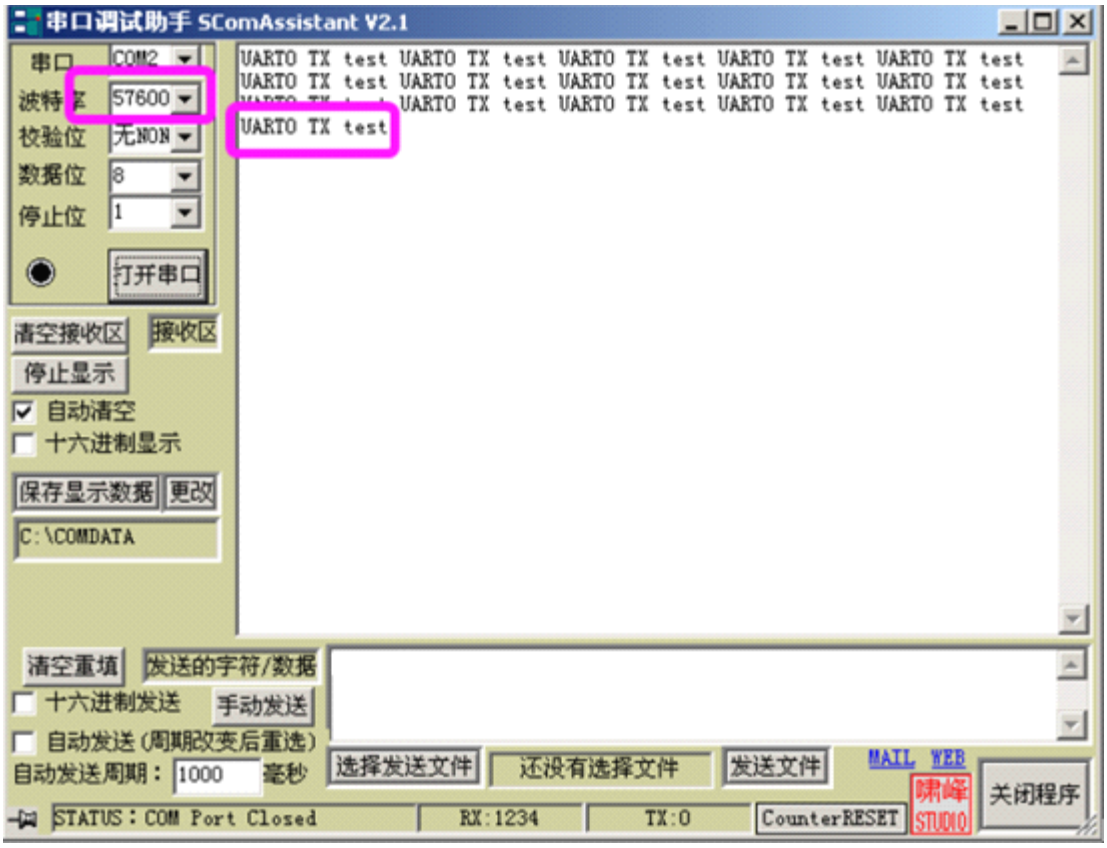
函数功能：设 P10,P11 设为输出控制 LED 灯，将 AD 转换源设为电源电压，参考电压为电源电压，ADC 结果分辨率设为 14 位（最高精度），AD 模式为单次转换，启动 ADC 转换。

1.5 UART 串口

1.5.1 CC2530 基础实验 13：单片机串口发数

1、实验介绍

从 CC2530 上通过串口不断地发送字串“UART0 TX Test”。实验使用 CC2530 的串口 1，波特率为 57600。



2、实验相关寄存器

实验中操作了的寄存器有：P1，P1DIR，CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，IEN0，U0DUB 等寄存器。

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

PERCFG 参见实验10

U0CSR 参见实验10

U0GCR 参见实验10

U0BAUD 参见实验10

U0BUF 参见实验 10

IEN0 参见实验 5

3、实验相关函数

void Delay(uint n)；定性延时，参见实验 1

void initUARTtest(void)；函数原型：

```
void initUARTtest(void)
{
    CLKCONCMD &= ~0x40; //晶振
    while(!(SLEEPSTA & 0x40)); //等待晶振稳定
    CLKCONCMD &= ~0x47; //TICHSPD128 分频，CLKSPD 不分频
    SLEEPCMD |= 0x04; //关闭不用的 RC 振荡器
    PERCFG = 0x00; //位置 1 P0 口
    P0SEL = 0x3c; //P0 用作串口
    P2DIR &= ~0XC0; //P0 优先作为串口 0
    U0CSR |= 0x80; //UART 方式
    U0GCR |= 10; //baud_e
    U0BAUD |= 216; //波特率设为 57600
    UTX0IF = 0;
}
```

函数功能：初始化串口 0，将 I/O 映射到 P0 口，P0 优先作为串口 0 使用，UART 工作方式，波特率为 57600。使用晶振作为系统时钟源。

void UartTX_Send_String(char *Data,int len)函数原型：

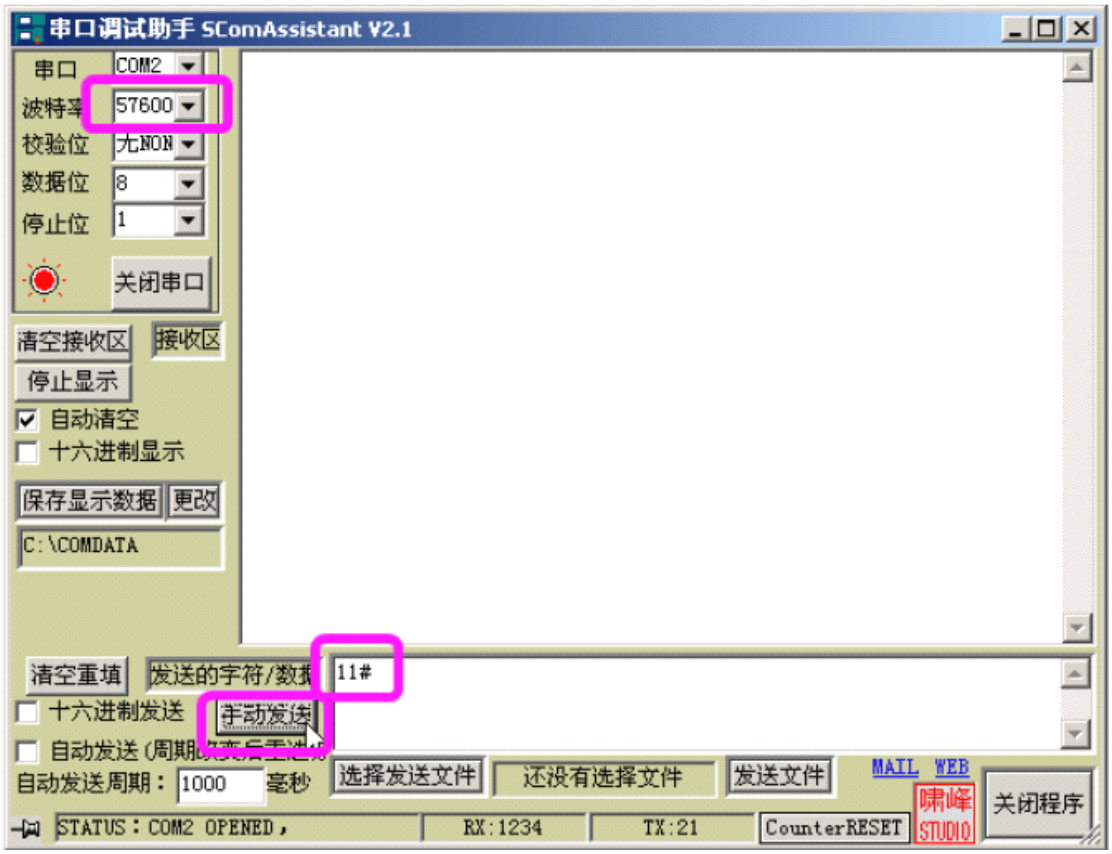
```
void UartTX_Send_String(char *Data,int len)
{
    int j;
    for(j=0;j<len;j++)
    {
        U0DBUF = *Data++;
        while(UTX0IF == 0);
        UTX0IF = 0;
    }
}
```

函数功能：串口发字串，*Data 为发送缓存指针，len 为发送字串的长度，只能是在初始化函数 void initUARTtest(void)之后调用才有效。发送完毕后返回，无返回值。

1.5.2 CC2530 基础实验 14：在 PC 用串口控制 LED

1、实验介绍

在 PC 上从串口向 CC2430 模块发送命令，即可控制 LED 灯的亮灭，控制数据的格式为“灯编号开丨关#”，LED1，LED 2，0 是关灯，1 是开灯，如打开 LED2 的命令是“21#”.



2、实验相关寄存器

实验中操作了的寄存器有：P1，P1DIR，P1SEL，CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，IEN0，U0DUB 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

PERCFG 参见实验10

U0CSR 参见实验10

U0GCR 参见实验10

U0BAUD 参见实验10

U0BUF 参见实验10

3、实验相关函数

void Delay(uint n)；定性延时，参见实验1

void initUARTtest(void)；函数原型：

```
void initUARTtest(void)
```

```
{
    CLKCONCMD &= ~0x40; //晶振
    while(!(SLEEPSTA & 0x40)); //等待晶振稳定
    CLKCONCMD &= ~0x47; //TICHSPD128 分频，CLKSPD 不分频
    SLEEPCMD |= 0x04; //关闭不用的 RC 振荡器
    PERCFG = 0x00; //位置 1 P0 口
    P0SEL = 0x3c; //P0 用作串口
    P2DIR &= ~0XC0; //P0 优先作为串口 0
    U0CSR |= 0x80; //UART 方式
    U0GCR |= 10; //baud_e
    U0BAUD |= 216; //波特率设为 57600
    UTX0IF = 0;
}
```

函数功能：初始化串口0，将I/O映射到P0口，P0优先作为串口0使用，UART工作方式，波特率为57600。使用晶振作为系统时钟源。

void UartTX_Send_String(char *Data,int len)； 函数原型：

```
void UartTX_Send_String(char *Data,int len)
{
    int j;
    for(j=0;j<len;j++)
    {
        U0DBUF = *Data++;
        while(UTX0IF == 0);
        UTX0IF = 0;
    }
}
```

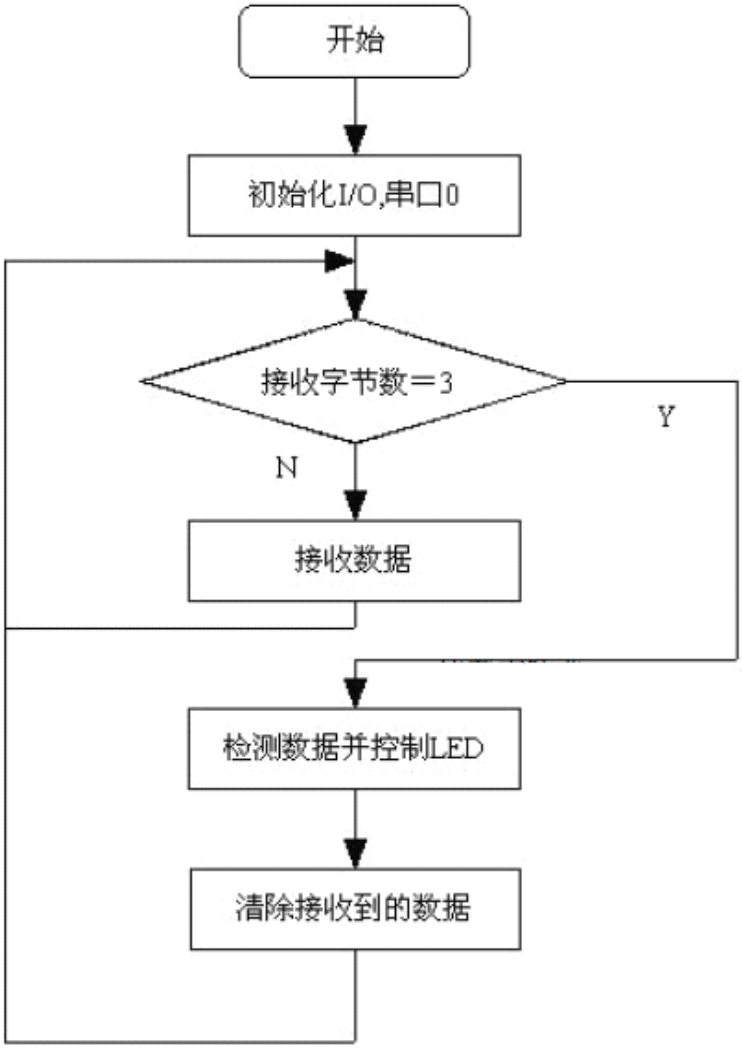
函数功能：串口发字串，*Data为发送缓存指针，len为发送字串的长度，只能是在初始化函数void initUARTtest(void)之后调用才有效。发送完毕后返回，无返回值。

void UART0_ISR(void)函数原型：

```
__interrupt void UART0_ISR(void)
{
    URX0IF = 0; //清中断标志
    temp = U0DBUF;
}
```

函数功能：一旦有数据从串口送到CC2530，则立即进入中断，进入中断后将接收的数据先存放到temp 变量，然后在主程序中去处理接收到的数据。

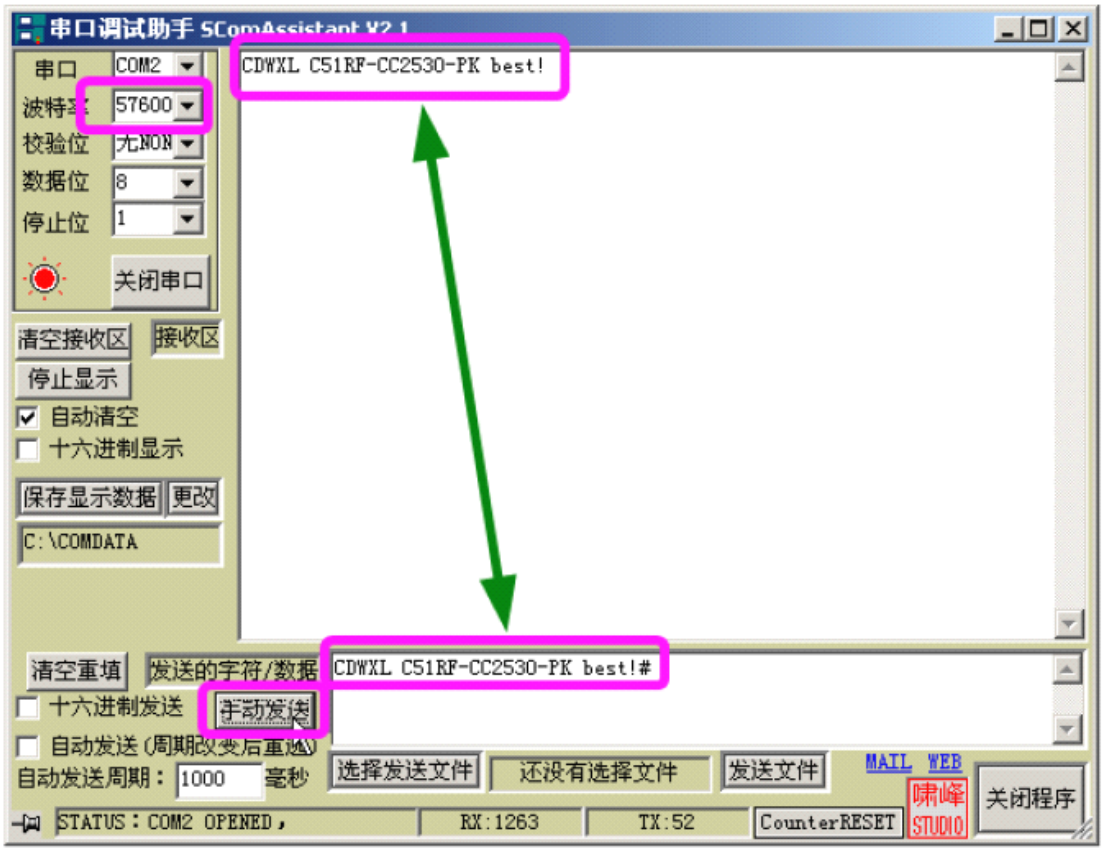
4、实验流程图



1.5.3 CC2530 基础实验 15：PC 串口收数并发数

1、实验介绍

在PC上从串口向CC2530发任意长度为30字节的字串，若长度不足30字节，则以“#“为字串末字节，CC2530在收到字节后会将这一字串从串口反向发向PC，用串口助手可以显示出来。



2、实验相关寄存器

实验中操作了的寄存器有：P1，P1DIR，P1SEL，CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，IEN0，U0DUB 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

PERCFG 参见实验10

U0CSR 参见实验10

U0GCR 参见实验10

U0BAUD 参见实验10

U0BUF 参见实验10

3、实验相关函数

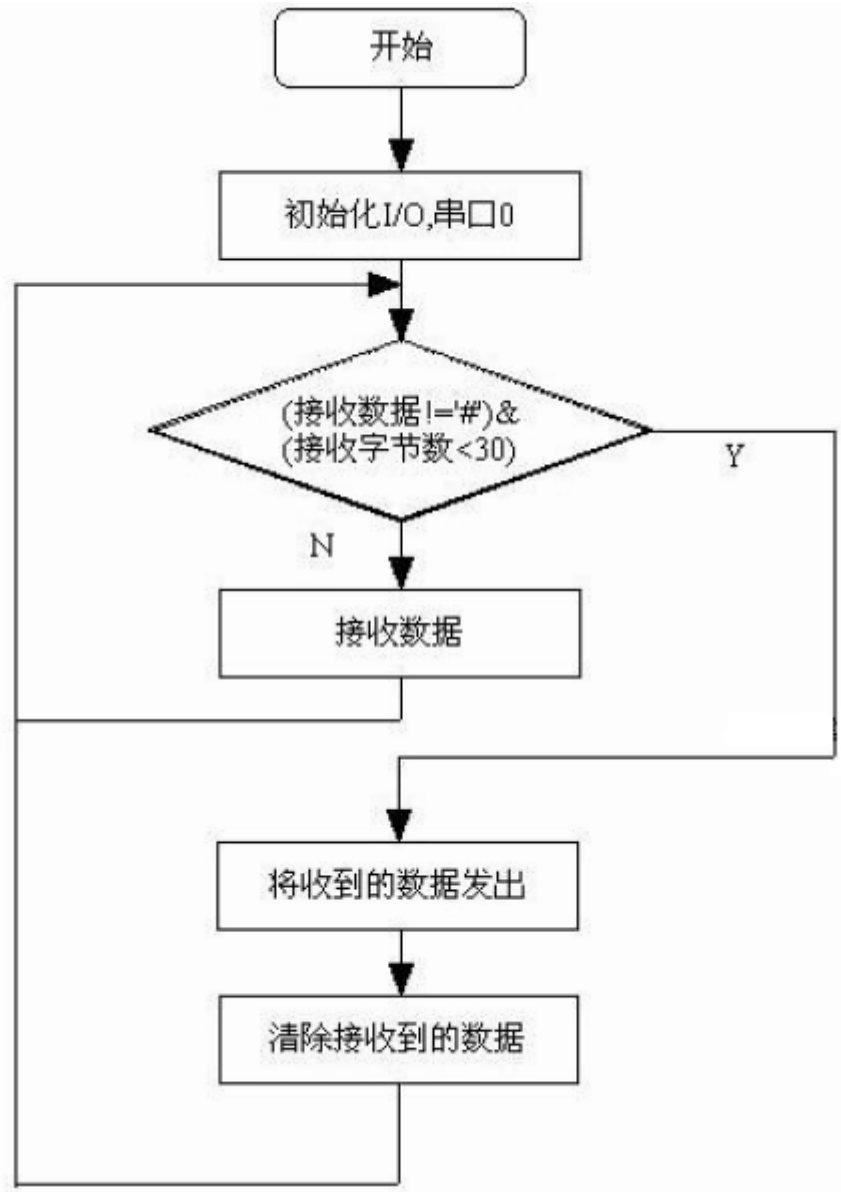
void Delay(uint n)；定性延时，参见实验1

void initUARTtest(void)；参见实验15

void UartTX_Send_String(char *Data，int len)；参见实验15

void UART0_ISR(void)；参见实验15

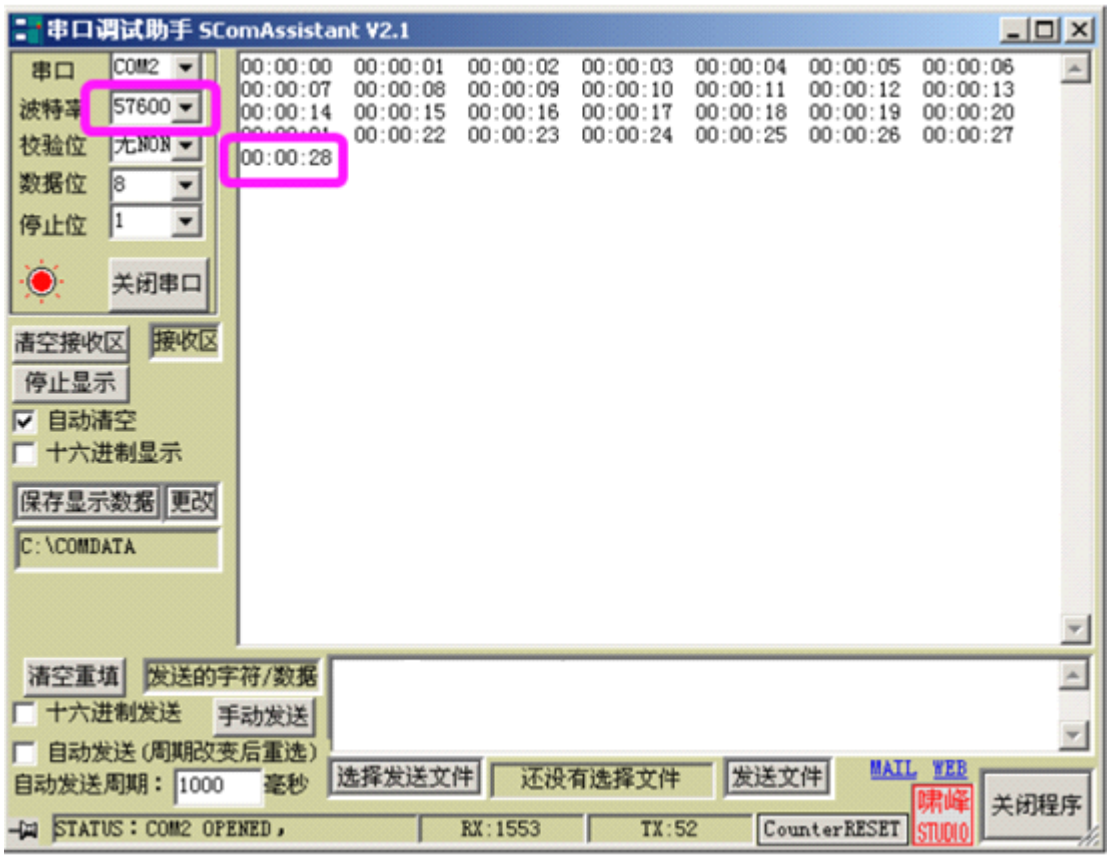
4、实验流程图



1.5.4 CC2530 基础实验 16：串口时钟 PC 显示

1、实验介绍

利用CC2530 定时器1 产生秒信号，通过串口显示时钟。



2、实验相关寄存器

实验中操作了的寄存器有：P1，P1DIR，P1SEL，T1CTL，T1CCTL0，T1CC0H，T1CC0L，IEN0，IEN1，CLKCONCMD，SLEEPCMD，PERCFG，U0CSR，U0GCR，U0BAUD，IEN0，U0DUB 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

T1CTL 参见实验4

IEN0 参见实验10

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

PERCFG 参见实验10

U0CSR 参见实验10

U0GCR 参见实验10

U0BAUD 参见实验10

U0BUF 参见实验10

IEN1（中断使能寄存器1）

位号	位名	复位值	操作性	功能描述
7:6	-	00	读	没有，读出为0
5	P0IE	0	读/写	P0口中断使能 0关中断，1开中断

4	T4IE	0	读/写	定时器 4 中断使能 0 关中断，1 开中断
3	T3IE	0	读/写	定时器 3 中断使能 0 关中断，1 开中断
2	T2IE	0	读/写	定时器 2 中断使能 0 关中断，1 开中断
1	T1IE	0	读/写	定时器 1 中断使能 0 关中断，1 开中断
0	DMAIE	0	读/写	DMA传输中断使能 0 关中断，1 开中断

T1CCTL0（T1 通道0 捕获/比较寄存器）

位号	位名	复位值	操作性	功能描述
7	CPSEL	0	读/写	T1通道0 捕捉设定 0 捕捉引脚输入，1 捕捉RF 中断
6	IM	1	读/写	T1通道0 中断掩码 0 关中断，1 开中断
5:3	CMP[2:0]	000	读/写	T1通道0 模式比较输出选择，指定计数 值过T3CC0 时的发生事件 000 输出置1（发生比较时） 001 输出清0（发生比较时） 010 输出翻转 011 输出置1（发生上比较时） 输出清0（计数值为0 或UP/DOWN 模式下发生下比较） 100 输出清0（发生上比较时） 输出置1（计数值为0 或UP/DOWN 模式下发生下比较） 101 预留，110 预留，111 预留
2	MODE	0	读/写	T1通道0 模式选择 0 捕获，1 比较
1:0	CPM[1:0]	0	读/写	T1通道0 捕获模式选择 00 没有捕获

				01 上升沿捕获 10 下降沿捕获 11 边沿捕获

T1CC0H（T1 通道0 捕获值/比较值高字节寄存器）

位号	位名	复位值	操作性	功能描述
7	T1CC0[15:8]	0X00	读/写	T1 通道 0 捕获值/比较值高字节

T1CC0L（T1 通道0 捕获值/比较值低字节寄存器）

位号	位名	复位值	操作性	功能描述
7	T1CC0[7:0]	0X00	读/写	T1 通道 0 捕获值/比较值低字节

3、实验相关函数

void Delay(uint n)；定性延时，参见实验1

void initUARTtest(void)；参见实验15

void UartTX_Send_String(char *Data，int len)；参见实验15

void UART0_ISR(void)；参见实验15

void InitT1(void)；函数原型：

```
void InitT1(void)
{
  T1CCTL0 = 0X44;
  //T1CCTL0 (0xE5)
  //T1 ch0 中断使能
  //比较模式
  T1CC0H = 0x03;
  T1CC0L = 0xe8;
  //0x0400 = 1000D)
  T1CTL |= 0X02;
  //start count
  //在这里没有分频。
  //使用比较模式 MODE = 10(B)
  IEN1 |= 0X02;
  IEN0 |= 0X80;
  //开T1 中断
}
```

函数功能：开T1中断，T1为比较计数模式。因T1计数时钟为0.25M（见void InitClock(void)说明），T1CC0 = 0X03E8 ＝ 1000，因此250 次中断溢出为1s。

void InitClock(void)；函数原型：

```
void InitClock(void)
{
  CLKCONCMD = 0X38;
  //TICKSPD = 111 定时器计数时钟源 0.25M
  while(!(SLEEPCMD&0X40));
```

```
  //等晶振稳定
}
```

函数功能：设置系统时钟为晶振，同时将计数器时钟设为0.25M。晶振振荡稳定后退出函数。

void InitUART0(void)；函数原型：

```
void InitUART0(void)
{
  PERCFG = 0x00; //位置1 P0 口
  P0SEL = 0x3c; //P0用作串口
  U0CSR |= 0x80; //UART 方式
  U0GCR |= 10; //baud_e
  U0BAUD |= 216; //波特率设为57600
  UTX0IF = 1;
  U0CSR |= 0X40; //允许接收
  IEN0 |= 0x84; //开总中断，接收中断
}
```

函数功能：设置系统时钟为晶振，同时将计数器时钟设为0.25M。晶振振荡稳定后退出函数。

void InitUART0(void)；函数原型：

```
void InitUART0(void)
{
  PERCFG = 0x00; //位置1 P0 口
  P0SEL = 0x3c; //P0用作串口
  U0CSR |= 0x80; //UART 方式
  U0GCR |= 10; //baud_e
  U0BAUD |= 216; //波特率设为57600
  UTX0IF = 1;
  U0CSR |= 0X40; //允许接收
  IEN0 |= 0x84; //开总中断，接收中断
}
```

函数功能：串口 0 映射位置1，UART 方式，波特率57600，允许接收，开接收中断。

void T1_ISR(void)；函数原型：

```
__interrupt void T1_ISR(void)
{
    IRCON &= ~0x02;//清中断标志
    counter++;
    if(counter == 250)
    {
        counter = 0;
        timetemp = 1; //一秒到
        led1 = ~led1; // 调试指示用
    }
}
```

函数功能：T1中断服务程序，每250次中断将timetemp置1，表示1秒时间到，同时改变

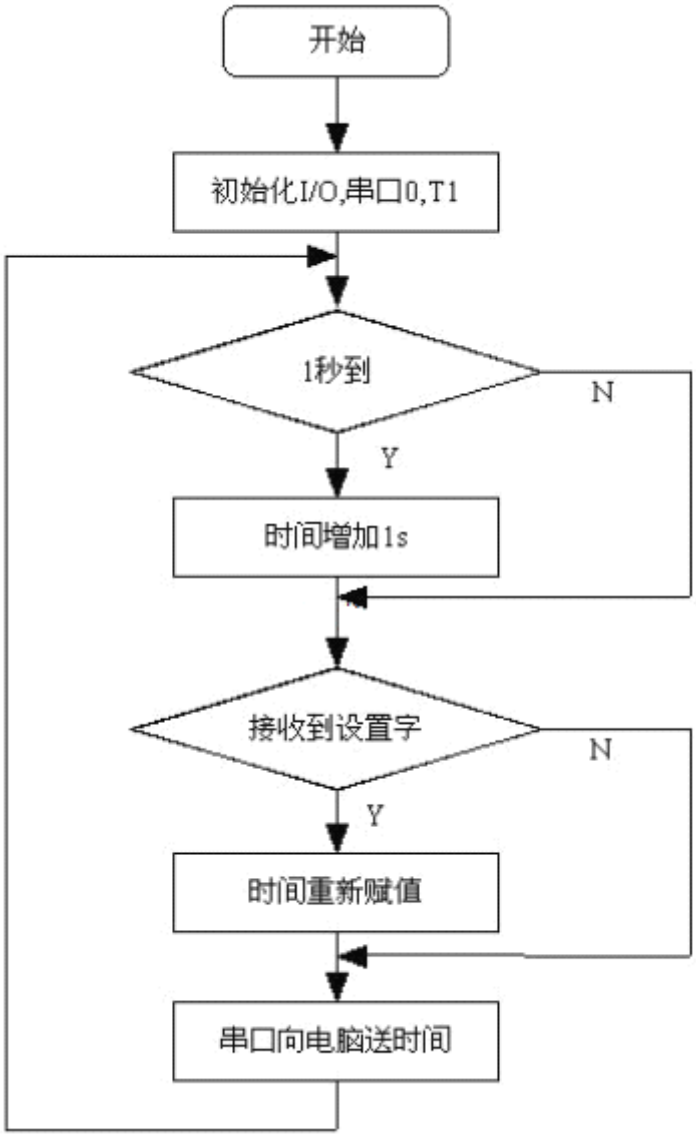
LED的状态。

void UART0_ISR(void)；函数原型：

```
__interrupt void UART0_ISR(void)
{
URX0IF = 0; //清中断标志
temp = U0DBUF;
}
```

函数功能：从串口 0 接收用于设置时间的字符。

4、实验流程图



1.6 睡眠定时器实验

1.6.1 CC2530 基础实验 17：系统睡眠工作状态

1、实验介绍

在小灯闪烁 10 次以后进入低功耗模式PM3。CC2530 一共有4 种功耗模式，分别是PM0，PM1，PM2，PM3，以PM3 功耗最低。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL，CLKCONCMD，SLEEPCMD，PCON 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

PCON（电源模式控制寄存器）

位号	位名	复位值	可操作性	功能描述
7:2	-	0X00	读/写	预留
1	-	0	读	预留，读出为 0
0	IDLE	0	读/写	电源模式控制，写 1 将进入由 SLEEPCMD.MODE 指定的电源模式，读出一定为 0。

3、实验相关函数

void Delay(void); 参见实验1

void Initial(void); 参见实验17

4、重要的宏定义

设置 CC2530 功耗模式，选定后立刻进入相应功耗模式。

```
#define SET_POWER_MODE(mode) \
do { \
    if(mode == 0) { SLEEPCMD &= ~0x03; } \
    else if (mode == 3) { SLEEPCMD |= 0x03; } \
    else { SLEEPCMD &= ~0x03; SLEEPCMD |= mode; } \
    PCON |= 0x01; \
    asm("NOP"); \
}while (0)
```

5、功耗测定方法

将本次实验的程序写入无线的CC2530模块，将测量电流表串接入CC2530模块的供电电路，待小灯同步闪烁后测电流，然后根据P=U*I即可得到功率。提示：可以在程序中将小灯关闭，进一步降低功耗。

1.6.2 CC2530 基础实验 18：系统唤醒

1、实验介绍

本次实验使能外部I/O中断(按下CC2000板的OK或CANCEL按键)唤醒CC2430，每次唤醒LED闪烁10次，然后进入低功耗模式，在进入PM3之前程序会将两个LED灯关闭。在应用中也可以不关闭以指示CC2530 处于低功耗模式，可以中断激活。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL，P1IEN，P1CTL，IEN2，IEN0，P1IFG，P1INP，P2INP， CLKCONCMD，SLEEPCMD 等寄存器。

P1 参见实验1
P1DIR 参见实验1
P1SEL 参见实验1
P1IEN 参见实验9
PICTL 参见实验9
IEN2 参见实验9
IEN0 参见实验5
P1IFG 参见实验9
P1INP 参见实验2
CLKCONCMD 参见实验10
SLEEPCMD 参见实验10

P2INP（P2 输入模式寄存器）

位	名称	复位	读/写	描述
7	PDUP2	0	R/W	端口2上拉/下拉选择。选择所有端口2引脚功能设置为上拉/下拉输入 0：上拉 1：下拉
6	PDUP1	0	R/W	端口1上拉/下拉选择。选择所有端口1引脚功能设置为上拉/下拉输入 0：上拉 1：下拉
5	PDUP0	0	R/W	端口0上拉/下拉选择。选择所有端口0引脚功能设置为上拉/下拉输入 0：上拉 1：下拉
4:0	MDP2_[4:0]	00000	R/W	P2.4到P2.0 I/O输入模式 0：上拉/下拉 1：三态

3、实验相关函数

void Delay(void); 参见实验1
void Init_IO_AND_LED(void);函数原型：

```
void Init_IO_AND_LED(void)
{
    P1DIR = 0X03;
```

```
    RLED = 1;
    YLED = 1;
    P0SEL &= ~0X30; //P04 P05作为普通IO
    P0SEL |= 0XC0;   //P06 ,07不作为普通IO,也不会产生中断信号
    P0DIR &= ~0X30; //输入
    P0INP &= ~0x30; //
    PICTL |= 0X11;    //下降沿
    EA = 1;
    P0IE = 1;
    P0IEN |= 0x30;
    P0IFG = 0;    //P04 p05中断标志清0
    P0IF = 0;
   P0IFG = 0x00;
   };
```

函数功能：置 P10，P11 为输出，打开P0 口的中断，P0 口下降沿触发中断。

void PowerMode(uchar sel);函数原型：

```
void PowerMode(uchar sel)
{
    uchar i，j;
    i = sel;
    if(sel<4)
    {
        SLEEPCMD &= 0xfc;
        SLEEPCMD |= i;
        for(j=0;j<4;j++);
        PCON = 0x01;
    }
    else
    {
        PCON = 0x00;
    }
}
```

函数功能：使系统进入 sel 指定的电源模式下，这里的sel 只能是0-3 之间的数，程序只能在CPU 全速运行时执行，也就是说函数中能使系统从全速运行进入PM0-PM3 而不可以从PM0-PM3进入全速运行。

1.6.3 CC2530 基础实验 19：睡眠定时器使用

1、实验介绍

在小灯快速闪烁5次后进入睡眠状态PM2，在PM2下睡眠定时器SLEEPCMD TIMER（ST）仍然可以正常工作，从0x000000到0xffffff反复计数，当ST计数超过写入ST[2－0]的0x000f00时，系统由中断唤醒，小灯闪烁5次后进入PM2，这样周而复始的唤醒工作然后睡眠。系统睡眠的时间为8 分32 秒，这已经是最长睡眠时间。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL，IEN0，ST2，ST1，ST0，CLKCONCMD，SLEEPCMD 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

IEN0 参见实验5

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

ST2（睡眠定时器2）

位号	位名	复位值	操作性	功能描述
7:0	ST2[7:0]	0X00	读/写	睡眠定时器计数/比较值[23－16]位。读出为 ST 计数值，写入为比较值。读寄存器应先读 ST0，写寄存器就后写 ST0。

ST1（睡眠定时器1）

位	名称	复位	读/写	描述
7：0	ST1[7：0]	0x00	R/W	睡眠定时器计数/比较值。读取时，寄存器返回睡眠定时器计数的中位[15：8]。写该寄存器时设置比较值的中位[15：8]。当读寄存器 ST0 时，读取值是锁定的。当写 ST0 时，写入值是锁定的。

ST0（睡眠定时器0）

位	名称	复位	读/写	描述
7：0	ST0[7：0]	0x00	R/W	睡眠定时器计数/比较值。读取时，寄存器返回睡眠定时器计数的低位[7：0]。写该寄存器时设置比较值的低位[7：0]。除非 STLOAD.LDRDY 为 1，否则忽略对该寄存器的写。

3、实验相关函数

void Delay(void);参见实验1

void Init_SLEEPCMD_TIMER(void);函数原型：

```
void Init_SLEEPCMD_TIMER(void)
{
    ST2 = 0X00;
```

```
    ST1 = 0X0f;
    ST0 = 0X00;
    EA = 1; //开中断
    STIE = 1;
    STIF = 0;
}
```

函数功能：打开睡眠定时器 SLEEPCMD TIMER(ST)中断，设置ST 的中断发生时间为计数值达到0x000f00 时。

void LedGlint(void);函数原型：

```
void LedGlint(void)
{
    uchar jj=10;
    while(jj--)
    {
        RLED = !RLED;
        Delay(10000);
    }
}
```

函数功能：让 LED 闪烁5 次，无返回值。

void ST_ISR(void)； 函数原型：

```
__interrupt void ST_ISR(void)
{
    STIF = 0;
    LEDBLINK = 1;
}
```

函数功能：睡眠定时器中断服务程序，清中断标志，无其他操作。

4、重要的宏定义

使模块上的可控制

```
#define LED_ENABLE(val)             \
do{                                 \
    If(val==1)                      \
    {                               \
        P1SEL &= ~0X03;             \
        P1DIR |= 0X03;              \
        RLED = 1;                   \
        GLED = 1;                   \
    }                               \
    else                            \
    {                               \
        P1DIR &= ~0X03;             \
    }                               \
}while(0)
#define RLED P1_0
#define GLED P1_1
```

选择系统工作时钟源并关闭不用的时钟源。

```
#define SET_MAIN_CLOCK_SOURCE(source)       \
do {                                                \
    if(source) {                                \
        CLKCONCMD |= 0x40; /*RC*/           \
        while(!(SLEEPCMD&0X20)); /*待稳*/    \
        SLEEPCMD |= 0x04; /*关掉不用的*/        \
    }                                           \
    else {                                      \
        SLEEPCMD &= ~0x04; /*全开*/             \
        while(!(SLEEPSTACMD&0X40));/*待稳*/ \
        asm("NOP");                             \
        CLKCONCMD &= ~0x47; /*晶振*/        \
        SLEEPCMD |= 0x04; /*关掉不用的*/     \
    }                                           \
}while (0)
#define CRYSTAL 0
#define RC 1
```

选择系统低速时钟源。

```
#define SET_LOW_CLOCK(source)                                   \
do{                                                             \
    (source==RC)?(CLKCONCMD |= 0X80):(CLKCONCMD &= ~0X80);   \
}while(0)
```

1.6.4 CC2530 基础实验 20：定时唤醒

1、实验介绍

这个实验利用睡眠定时器工作在多个电源模式下这一特性来实现定时唤醒，最长的唤醒时隔为8 分32 秒，而最短的时隔可达30 余微秒。实验中在设定好唤醒时间后让CC2530 进入PM2 模式，在达到指定时间后小灯闪烁，之后再次是设定唤醒时间，进入PM2，唤醒的循环。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL，ST2，ST1，ST0，CLKCONCMD，SLEEPCMD等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

ST2 参见实验19

ST1 参见实验19

ST0 参见实验19

3、实验相关函数

void Delay(void); 参见实验1

void LedGlint(void)； 参见实验 19

void Init_SLEEPCMD_TIMER(void)函数原型：

```
void Init_SLEEPCMD_TIMER(void)
{
    EA = 1; //开中断
    STIE = 1;
    STIF = 0;
}
```

函数功能：打开睡眠定时器（ST）的中断，并且将ST 的中断标志位清零。在使用ST 时必须于addToSLEEPCMDTimer()前调用本函数。

void addToSLEEPCMDTimer(UINT16 sec)函数原型：

```
void addToSLEEPCMDTimer(UINT16 sec)
{
    UINT32 SLEEPCMDTimer = 0;
    SLEEPCMDTimer |= ST0;
    SLEEPCMDTimer |= (UINT32)ST1 << 8;
    SLEEPCMDTimer |= (UINT32)ST2 << 16;
    SLEEPCMDTimer += ((UINT32)sec * (UINT32)32768);
    ST2 = (UINT8)(SLEEPCMDTimer >> 16);
    ST1 = (UINT8)(SLEEPCMDTimer >> 8);
    ST0 = (UINT8) SLEEPCMDTimer;
}
```

函数功能：设置睡眠时间，在 sec 秒以后由ST 唤醒CC2530，在调用这个函数之前必须先调用Init_SLEEPCMD_TIMER （ ）， 否则不能唤醒CC2530 。通常在这个函数以后会出现SET_POWER_MODE(2)语句。

4、重要的宏定义

改变系统的电源功耗模式

```
#define SET_POWER_MODE(mode)                                \
do {                                                         \
    if(mode == 0) { SLEEPCMD &= ~0x03; }                    \
    else if (mode == 3) { SLEEPCMD |= 0x03; }               \
    else { SLEEPCMD &= ~0x03; SLEEPCMD |= mode; }           \
    PCON |= 0x01;                                           \
    asm("NOP");                                             \
}while (0)
```

1.7 看门狗

看门狗的作用是在 CPU 在软件中跑飞的情况下的一种恢复方式，当软件在选择时间间隔内未清除看门狗，看门狗复位系统。看门狗可用在对电噪声、电源故障、静电放电等高可靠要求情况下。如果看门狗不需要应用，可配置成看门狗定时器用做间隔定时，在选定时间间隔产生中断。

看门狗定时器的特性如下：

- 4 个可选择时间间隔
- 看门狗模式
- 定时器模式
- 定时器模式下中断请求产生

看门狗定时器做为一般用途可作为看门狗定时器和定时器用，WDCTL 寄存器控制看门狗模块操作。

1.7.1 CC2530 基础实验 21：看门狗模式

1、实验介绍

程序在主程序中没有连续改变小灯的状态，而在开始运行时将其关闭，延时后点亮。实验现象是一只小灯不断闪烁，这是因为程序中启动了看门狗，看门狗时间长度为1 秒，如果1 秒内没有复位看门狗的话，系统将复位。系统复位后再次开启看门狗，1 秒后复位。

2、实验相关寄存器

实验中操作了的寄存器有 P1，P1DIR，P1SEL，WDTCL，CLKCONCMD 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

WDCTL（看门狗定时器控制寄存器）

位号	位名	复位值	操作性	功能描述
7:4	CLR[3:0]	0000	读/写	看门狗复位，先写 0xa 再写 0x5 复位看门狗，两次写入不超过 0.5 个看门狗周期，读出为 0000。
3	EN	0	读/写	看门狗定时器使能位，在定时器模式下写 0 停止计数，在看门狗模式下写 0 无效。0 停止计数，1 启动看门狗/开始计数。
2	MODE	0	读/写	看门狗定时器模式 0 看门狗模式 ，1 定时器模式。
1:0	INT[1:0]	00	读/写	看门狗时间间隔选择。 00 1 秒 01 0.25 秒 10 15.625 毫秒 11 1.9 毫秒（以 32.768K 时钟计算）

3、实验相关函数

void Delay(void)函数原型：

```
void Delay(void)
{
    uint n;
    for(n=50000;n>0;n--);
    for(n=50000;n>0;n--);
```

```
    for(n=50000;n>0;n--);
    for(n=50000;n>0;n--);
    for(n=50000;n>0;n--);
    for(n=50000;n>0;n--);
    for(n=50000;n>0;n--);
}
```

函数功能：软件延时 10.94ms。

void Init_IO(void);函数原型：

```
void Init_IO(void)
{
    P1DIR = 0x03;
    led1 = 1;
    led2 = 1;
}
```

函数功能：将 P10，P11 设置为输出控制LED。

void Init_Watchdog(void);函数原型：

```
void Init_Watchdog(void)
{
    WDCTL = 0x00;
    //时间间隔一秒，看门狗模式
    WDCTL |= 0x08;
    //启动看门狗
}
```

函数功能：以看门狗模式启动看门狗定时器，看门狗复位时隔 1 秒。

void Init_Clock(void);函数原型：

```
void Init_Clock(void)
{
    CLKCONCMD = 0X00;
}
```

函数功能：将系统时钟设为晶振，低速时钟设为晶振，程序对时钟要求不高，不用等待晶振稳定。

1.7.2 CC2530 基础实验 22：喂狗

1、实验介绍

本实验与实验 20 一都以看门狗为学习目标，在实验21 学会初始化看门狗，同时也知道了看门狗的作用，本实验着重学习复位看门狗。复位看门狗后小灯不会闪烁。

2、实验相关寄存器

实验中操作了的寄存器有P1，P1DIR，P1SEL， WDTCL，CLKCONCMD 等寄存器。

P1 参见实验1

P1DIR 参见实验1

P1SEL 参见实验1

CLKCONCMD 参见实验10

SLEEPCMD 参见实验10

WDCTL 参见实验20

3、实验相关函数

void Delay(void); 参见实验20

void Init_IO(void); 参见实验20

void Init_Watchdog(void); 参见实验20

void Init_Clock(void); 参见实验20

void FeetDog(void)函数原型：

```
void FeetDog(void)
{
    WDCTL = 0xa0;
    WDCTL = 0x50;
}
```

函数功能：复位看门狗，必须在看门狗时间间隔内调用本函数复位看门狗，系统会被强制复位，此时调用本函数已无意义。