CC2530 微控制器应用开发速查宝典（网络版）

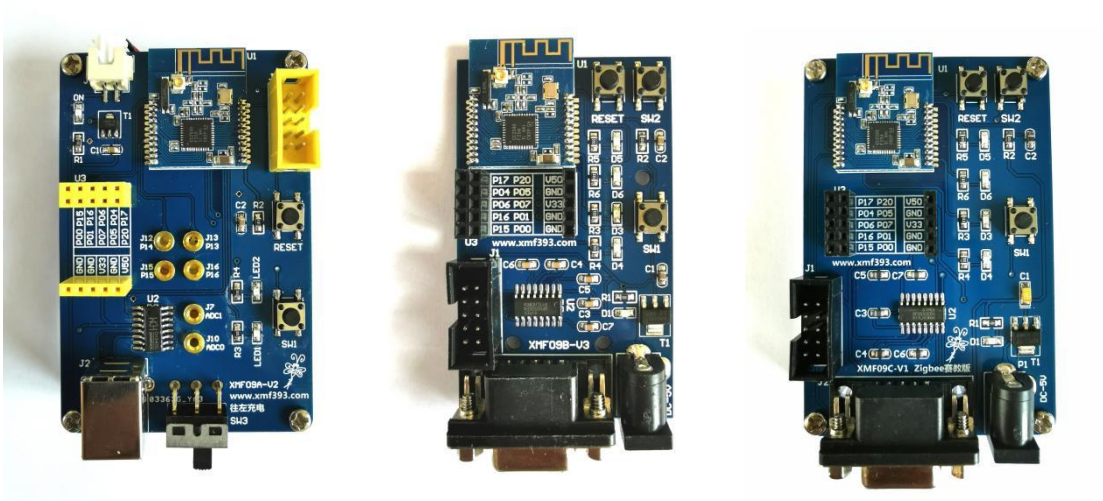
小蜜蜂老师 欧浩源

佛山市图志科技有限公司 广东职业技术学院

2021 年 08 月 12 日

【CC2530 开发套件 XMF09A/XMF09B/XMF09C】简介

该开发套件以 CC2530 无线模块为核心，兼容全国职业院校职业技能大赛“物联网技术应用”赛项的国赛设备，对接 1+X 证书“传感网应用开发”的考试要求，面向《CC2530 微控制器应用开发》、《BasicRF 与 Zigbee 无线传感网应用开发》、《物联网硬件技术基础》等课程的课堂教学与实训拓展，针对不同的应用需求，设计有三个型号：



【XMF09A】：兼容国赛蓝色 Zigbee 节点盒，外观稍有不同，功能略加改善，直接替换。
【XMF09B】：兼容国赛黑色 Zigbee 小模块，外观尺寸一样，功能完全兼容，直接替换。
【XMF09C】：功能和 XMF09B 完全一样，外观和结构更合理，适合课堂教学与实训开展。

XMF09A		XMF09B		XMF09C	
LED1	P1_0	D3	P1_0	D3	P1_0
LED2	P1_1	D4	P1_1	D4	P1_1
SW1	P1_2	D5	P1_3	D5	P1_3
锂电池	1000mAh/3.7V	D6	P1_4	D6	P1_4
香蕉插座	P1_3、P1_4、P1_5	SW1	P1_2	SW1	P1_2
	P1_6、P0_0、P0_1	SW2	P0_1	SW2	P0_1
通信接口	CH340 USB 方头接口	通信接口	RS232 DB9 母头	通信接口	RS232 DB9 母头
供电方式	仿真器供电 锂电池供电 USB 口供电	供电方式	仿真器供电 5V 直流电源	供电方式	仿真器供电 5V 直流电源
15 脚传感器模块扩展口		15 脚传感器模块扩展口		15 脚传感器模块扩展口	
亚克力底板 + 软磁条				亚克力底板	
兼容国赛蓝色 Zigbee 节点盒		兼容国赛黑色 Zigbee 小模块		面向课堂教学与实训拓展	

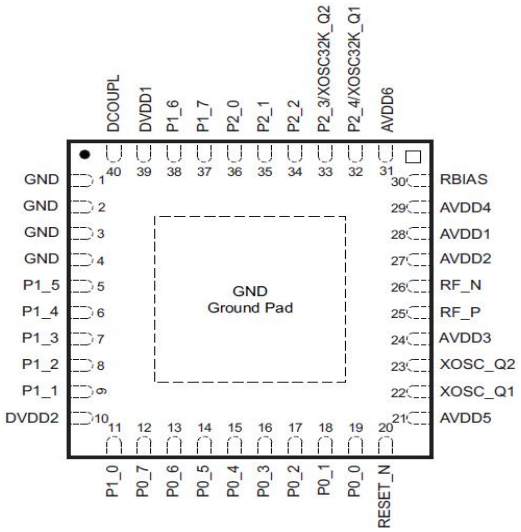
经过多年教学积累与开发经验，形成了以 CC2530 开发套件为核心，配套**教学视频、速查宝典、应用笔记、选择填空题库、程序设计题库、项目案例分析、技能大赛题解、师生交流社群**等全方位教学资源的**岗课赛证创**生态体系。资源目录汇总，详见以下链接：

【小蜜蜂笔记网】CC2530 专题栏目：https://www.xmf393.com/2019/10/20/xmf09b/
【XMF09A/XMF09B/XMF09C 开发套件】购买淘宝店铺：xmfkj.taobao.com
【小蜜蜂老师】欧浩源，欢迎交流：ohy3686@qq.com

目　录

【1】CC2530 微控制器概述

- 内核：增强型 8051
- 封装：QFN40
- Flash：256KB(CC2530F256)
- SRAM：8KB
- 无线外设：2.4GHz，IEEE802.15.4
- 21 个可编程数字 I/O 引脚
- 18 个中断源
- 4 个通用定时器
- 1 个睡眠定时器
- 1 个看门狗定时器
- 2 路串行通信接口
- 8 路 12 位 A/D 输入通道
- 5 通道 DMA 控制器
- 5 种运行模式，3 种低功耗运行模式



【2】CC2530 复位概述

- **CC2530 有 5 个复位源，以下事件将产生复位：**

  1-强制 RESET_N 输入引脚为低电平，即手动复位。

  2-上电复位条件。

  3-布朗输出复位条件。

  4-看门狗定时器复位条件。

  5-时钟丢失复位条件。

- **复位之后，初始条件如下：**

  1-I/O 引脚配置为**带上拉输入模式**（P1_0 和 P1_1 为没有上拉/下拉的输入模式）。

  2-CPU 程序计数器装载 **0x0000**，程序执行从这个地址开始。

  3-所有外设寄存器初始化为各自的**复位值**。

  4-看门狗定时器**禁用**。

  5-时钟丢失探测器**禁用**。

【3】IAR 环境下 CC2530 的开发流程

1-在 IAR 中新建 Workspace 工作区。

2-在工作区中新建一个空的工程。

3-为工程配置 2 个参数：指定芯片型号和指定仿真器类型。

4-新建 C 语言代码文件，并保存（如果已有 C 代码文件，该步骤略过）

5-将代码文件添加到工程中来。

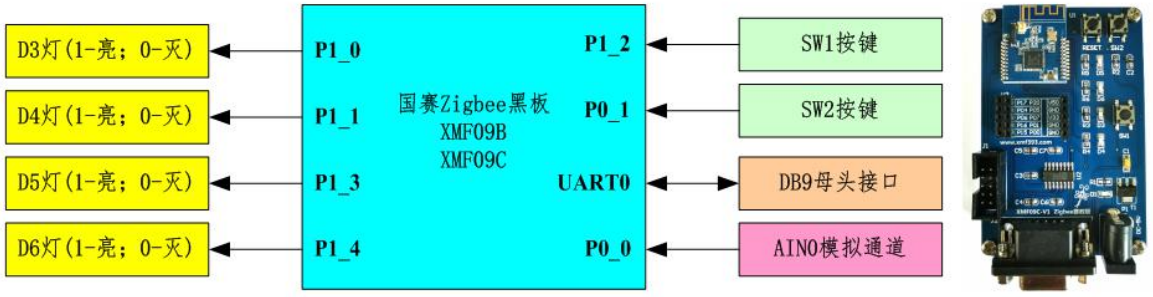
6-根据应用需求编写代码。

7-编译代码。

8-连接仿真器和目标板。

9-下载代码，运行调试。

【4】CC2530 开发板 XMF09C 功能框图



- 4 个 LED 灯，2 个按键，1 路串口，1 个 15 针传感器扩展接口，可用仿真器供电。
- 兼容物联网技能大赛国赛 Zigbee 小模块，对接传感网应用开发 1+X 证书考试。
- 视频教程、教案课件、配套教材、项目案例、习题实训、应用笔记，全方位支持。
- 更多详细资料详见【小蜜蜂笔记网】：www.xmf393.com

【5】CC2530 的 I/O 引脚概述

CC2530 采用 QFN40 封装，共有 40 个引脚，其中 21 个为可编程数字 I/O 引脚，分为 3 个端口组。P0 端口 8 个引脚，P1 端口 8 个引脚，P2 端口 5 个引脚。其中，P1_0 和 P1_1 引脚没有上拉/下拉能力，但具有 20mA 的高驱动输出能力，其余的 I/O 引脚具有 4mA 的输出驱动能力。在 P2 端口的 5 个引脚中，其中 2 个用于仿真器接口，2 个引脚用于外部 32KHz 晶振，在应用中，我们能够使用的数字 I/O 引脚实际只有 17 个。

【6】PxSEL 端口功能寄存器

位	位名称	复位值	操作	描述
7：0	SELPx[7:0]	0x00	R/W	设置 Px_7 到 Px_0 端口的功能。 0：对应端口被设置为通用 I/O 功能。 1：对应端口被设置为外设功能。
设计参考	P1SEL &= ～0x13; P0SEL |= 0x45;	//将 P1_4、P1_1 和 P1_0 设置成通用 I/O 功能， 0001 0011。 //将 P0_6、P0_2 和 P0_0 设置成外设功能， 0100 0101。		

【7】PxDIR 端口方向寄存器

位	位名称	复位值	操作	描述
7：0	DIRPx[7:0]	0x00	R/W	设置 Px_7 到 Px_0 端口的传输方向。 0：输入。 1：输出。
设计参考	P0DIR &= ～0x14; P1DIR |= 0x26;	//将 P0_4 和 P0_2 设置输入方向， 0001 0100。 //将 P1_5、P1_2 和 P1_1 设置输出方向， 0010 0110。		

【8】P0INP 端口输入配置寄存器

位	位名称	复位值	操作	描述
7：0	MDP0[7:0]	0x00	R/W	设置 P0_7 到 P0_0 端口的输入模式。 0：上拉/下拉（需要结合 P2INP 联合配置）。 1：三态。

【9】P1INP 端口输入配置寄存器

位	位名称	复位值	操作	描述
7：2	MDP1[7:2]	0x00	R/W	设置 P1_7 到 P1_2 端口的输入模式。 0：上拉/下拉（需要结合 P2INP 联合配置）。 1：三态。
1：0	----	00	R0	不使用。

【10】P2INP 端口输入配置寄存器

位	位名称	复位值	操作	描述
7	PDUP2	0	R/W	为 P2 端口所有引脚选择上拉或下拉。 0：上拉。　　1：下拉。
6	PDUP1	0	R/W	为 P1 端口所有引脚选择上拉或下拉。 0：上拉。　　1：下拉。
5	PDUP0	0	R/W	为 P0 端口所有引脚选择上拉或下拉。 0：上拉。　　1：下拉。
4：0	MDP2_[4:0]	0x00	R/W	设置 P2_4 到 P2_0 端口的输入模式。 0：上拉/下拉。　　1：三态。
设计参考	将 P0_1 和 P0_3 端口设置成上拉模式 P0INP &= ～0x0A;　　//先将 P0_3 和 P0_1 端口设置成上拉/下拉。 P2INP &= ～0x20;　　//再将 P0 端口的所有引脚设置上拉模式。			

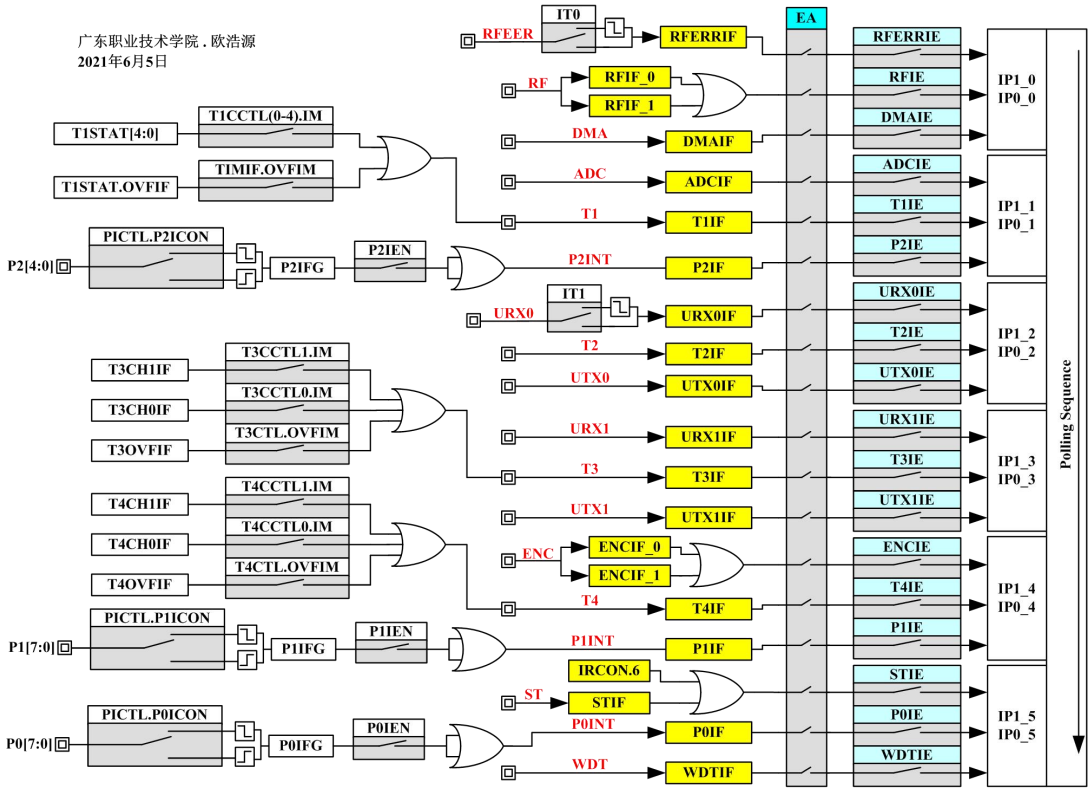
【11】CC2530 的 18 个中断源

中断号	中断名称	中断描述	中断向量	中断屏蔽位	中断标志位
0	RFERR	RF 发送完成或接收 FIFO 溢出	03H	IEN0.RFERRIE	TCON.RFERRIF [1]
1	ADC	ADC 转换结束	0BH	IEN0.ADCIE	TCON.ADCIF [1]
2	URX0	USART0 接收完成	13H	IEN0.URX0IE	TCON.URX0IF [1]
3	URX1	USART1 接收完成	1BH	IEN0.URX1IE	TCON.URX1IF [1]
4	ENC	AES 加密/解密完成	23H	IEN0.ENCIE	SOCON.ENCIF
5	ST	睡眠计数器比较	2BH	IEN0.STIE	IRCON.STIF
6	P2INT	I/O 端口 2 外部中断	33H	IEN2.P2IE	IRCON2.P2IF [2]
7	UTX0	USART0 发送完成	3BH	IEN2.UTX0IE	IRCON2.UTX0IF
8	DMA	DMA 传送完成	43H	IEN1.DMAIE	IRCON.DMAIF
9	T1	定时器 1 捕获/比较/溢出	4BH	IEN1.T1IE	IRCON.T1IF [1] [2]
10	T2	定时器 2	53H	IEN1.T2IE	IRCON.T2IF [1] [2]
11	T3	定时器 3 捕获/比较/溢出	5BH	IEN1.T3IE	IRCON.T3IF [1] [2]
12	T4	定时器 4 捕获/比较/溢出	63H	IEN1.T4IE	IRCON.T4IF [1] [2]
13	P0INT	I/O 端口 0 外部中断	6BH	IEN1.P0IE	IRCON.P0IF [2]
14	UTX1	USART1 发送完成	73H	IEN2.UTX1IE	IRCON2.UTX1IF
15	P1INT	I/O 端口 1 外部中断	7BH	IEN2.P1IE	IRCON2.P1IF [2]
16	RF	RF 通用中断	83H	IEN2.RFIE	S1CON.RFIF [2]
17	WDT	看门狗定时溢出	8BH	IEN2.WDTIE	IRCON2.WDTIF

[1] 当调用中断服务函数时，硬件清除标志位。

[2] 存在另外的 IRQ 掩码和 IRQ 标志位。

【12】CC2530 常用中断源的原理结构（完整版详见 CC2530 数据手册）



【13】中断服务函数编写规则

```
#pragma vector = <中断向量>
_ _interrupt void <函数名称> (void)
{
    /*...在这里编写中断处理函数的具体程序...*/
}
```

【14】IEN0 中断使能寄存器 0（可位寻址）

位	位名称	复位值	操作	描述
7	EA	0	R0	中断系统使能控制位，即:总中断。 0：禁止所有中断。　　1：允许所有中断。
6	----	00	R0	未使用，读为 0。
5	STIE	0	R/W	睡眠定时器中断使能。 0：中断禁止。　　1：中断使能。
4	ENCIE	0	R/W	AES 加密/解密中断使能。 0：中断禁止。　　1：中断使能。
3	URX1IE	0	R/W	USART1 接收中断使能。 0：中断禁止。　　1：中断使能。
2	URX0IE	0	R/W	USART0 接收中断使能。 0：中断禁止。　　1：中断使能。
1	ADCIE	0	R/W	ADC 中断使能。 0：中断禁止。　　1：中断使能。
0	RFERRIE	0	R/W	RF 发送/接收 FIFO 中断使能。 0：中断禁止。　　1：中断使能。

【15】IEN1 中断使能寄存器 1（可位寻址）

位	位名称	复位值	操作	描述
7:6	----	00	RO	不使用，读为 0。
5	P0IE	0	R/W	端口 0 中断使能。 0：中断禁止。　　1：中断使能。
4	T4IE	0	R/W	定时器 4 中断使能。 0：中断禁止。　　1：中断使能。
3	T3IE	0	R/W	定时器 3 中断使能。 0：中断禁止。　　1：中断使能。
2	T2IE	0	R/W	定时器 2 中断使能。 0：中断禁止。　　1：中断使能。
1	T1IE	0	R/W	定时器 1 中断使能。 0：中断禁止。　　1：中断使能。
0	DMAIE	0	R/W	DMA 中断使能。 0：中断禁止。　　1：中断使能。

【16】IEN2 中断使能寄存器 2（在 ioCC2530.h 头文件中没做位定义，故不能位寻址）

位	位名称	复位值	操作	描述
7:6	----	00	RO	不使用，读为 0。
5	WDTIE	0	R/W	看门狗定时器中断使能。 0：中断禁止。　　1：中断使能。
4	P1IE	0	R/W	端口 1 中断使能。 0：中断禁止。　　1：中断使能。
3	UTX1IE	0	R/W	USART1 发送中断使能。 0：中断禁止。　　1：中断使能。
2	UTX0IE	0	R/W	USART0 发送中断使能。 0：中断禁止。　　1：中断使能。
1	P2IE	0	R/W	端口 2 中断使能。 0：中断禁止。　　1：中断使能。
0	RFIE	0	R/W	RF 一般中断使能。 0：中断禁止。　　1：中断使能。

设计参考	在引用“ioCC2530.h”头文件进行程序设计时，对 IEN2 需要进行字节操作。 IEN2 |= 0x20;　　//看门狗定时器中断使能 IEN2 |= 0x10;　　//P1 端口外部中断使能 IEN2 |= 0x04;　　//串口 0 发送中断使能 IEN2 |= 0x02;　　//P0 端口外部中断使能

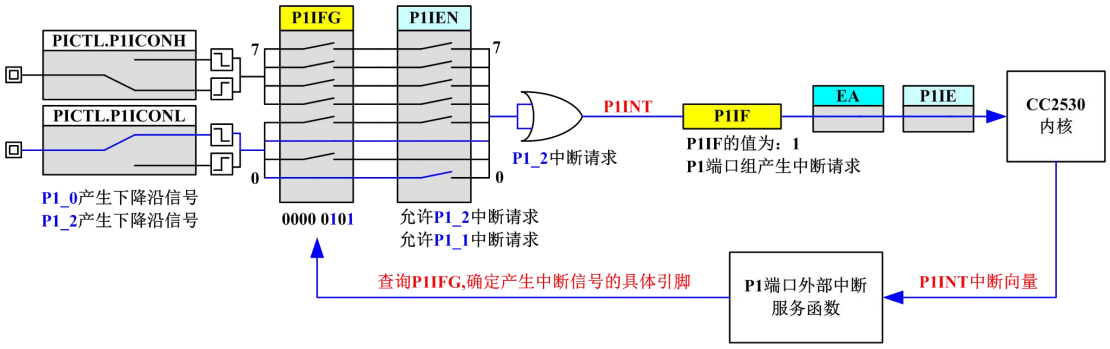
【17】外部中断的工作原理

CC2530 的 P0、P1 和 P2 端口中，每一个引脚在设置为输入后，都可以用于产生外部中断。产生外部中断的信号有上升沿和下降沿 2 种，通过 PICTL 端口输入信号控制寄存器设置。三个端口组作为三个独立的中断源，有三个中断使能控制位：IEN1.P0IE、IEN2.P1IE 和 IEN2.P2IE。每个端口组又有三个端口中断屏蔽寄存器 P0IEN、P1IEN 和 P2IEN，用于使能**端口组**中具体产生外部中断信号的**引脚**。在使能**端口外部中断**和**总中断**后，还需要通过 PxIEN 寄存器来设置当前**端口组**中，那些**引脚**具有外部中断功能。

只要端口的引脚产生了设定的中断触发信号，PxIFG 寄存器中对应的位便会置 1。在使

能了 PxIEN 寄存器和 EA 总中断后，只要该端口组中有一个或多个引脚产生了中断触发信号，对应的 PxIF 标志位置 1，内核响应便会响应该端口的外部中断请求，但内核并不知道是端口组中的哪一个引脚产生的请求。也就是说，不管是 P1_0 产生的中断请求，还是 P1_2 产生的中断请求，在内核看来都是 P1 端口产生的中断请求。因此，在进行中断服务函数设计时，还需要对 PxIFG 进行判断，才能确定是哪一个具体引脚产生的外部中断请求。

可以结合 P1 端口的外部中断结构，理解外部中断的工作原理。



【18】P0IEN P0 端口中断使能寄存器

位	位名称	复位值	操作	描述
7:0	P0_[7:0]IEN	0x00	R/W	端口 P0_7 到 P0_0 中断使能。 0：中断禁止。　　1：中断使能。

【19】P1IEN P1 端口中断使能寄存器

位	位名称	复位值	操作	描述
7:0	P1_[7:0]IEN	0x00	R/W	端口 P1_7 到 P1_0 中断使能。 0：中断禁止。　　1：中断使能。

【20】P2IEN P2 端口中断使能寄存器

位	位名称	复位值	操作	描述
7:6	----	0x00	RO	不使用，读为 0。
5	DPIEN	0x00	R/W	USB D+中断使能。
4:0	P2_[4:0]IEN	0x00	R/W	端口 P2_4 到 P2_0 中断使能。 0：中断禁止。　　1：中断使能。

【21】PICTL 端口输入信号控制寄存器

位	位名称	复位值	操作	描述
7	PADSC	0	R/W	控制 I/O 引脚输出模式下的驱动能力。
6:4	----	000	RO	未使用。
3	P2ICON	0	R/W	端口 P2_4 到 P2_0 中断触发方式选择。 0：上升沿触发。　　1：下降沿触发。
2	P1ICONH	0	R/W	端口 P1_7 到 P1_4 中断触发方式选择。 0：上升沿触发。　　1：下降沿触发。
1	P1ICONL	0	R/W	端口 P1_3 到 P1_0 中断触发方式选择。 0：上升沿触发。　　1：下降沿触发。
0	P0ICON	0	R/W	端口 P0_7 到 P0_0 中断触发方式选择。 0：上升沿触发。　　1：下降沿触发。

【22】P0IFG P0 端口中断状态标志寄存器

位	位名称	复位值	操作	描述
7：0	P0IF_[7:0]	0x00	R/W0	端口 P0_7 到 P0_0 的中断状态标志。 当输入端口有未响应的中断请求时，相应标志位硬件自动置 1，需要通过软件人工清 0。 【注】：该标志必须在清除端口中断标志 P0IF 之前清除。 0：无中断请求。　　1：有中断请求。

【23】P1IFG P1 端口中断状态标志寄存器

位	位名称	复位值	操作	描述
7：0	P1IF_[7:0]	0x00	R/W0	端口 P1_7 到 P1_0 的中断状态标志。 当输入端口有未响应的中断请求时，相应标志位硬件自动置 1，需要通过软件人工清 0。 【注】：该标志必须在清除端口中断标志 P1IF 之前清除。 0：无中断请求。　　1：有中断请求。

【24】P2IFG P2 端口中断状态标志寄存器

位	位名称	复位值	操作	描述
7:6	----	00	R0	未使用，读为 0。
5	DPIF	0	R/W0	USB D+中断标志位。
4：0	P2IF_[4:0]	0x00	R/W0	端口 P2_4 到 P2_0 的中断状态标志。 当输入端口有未响应的中断请求时，相应标志位硬件自动置 1，需要通过软件人工清 0。 【注】：该标志必须在清除端口中断标志 P2IF 之前清除。 0：无中断请求。　　1：有中断请求。

【25】CC2530 的定时器资源概述

CC2530 有 4 个通用定时器 T1、T2、T3 和 T4，1 个睡眠定时器和 1 个看门狗定时器。

定时器 1：16 位定时器，支持间隔定时、输入捕获、输出比较、PWM 输出、触发 DMA，是 CC2530 中功能最全的一个定时器，**在应用中应优先选用**。该定时器具有 5 个独立的捕获/比较通道，每个通道使用 1 个 I/O 引脚，有 3 种工作模式：自由运行模式、模模式和正计数/倒计数模式。

定时器 2：16 位定时器，主要用于 CSMA-CA 算法提供定时，用户一般不使用该定时器。

定时器 3 和**定时器 4**：8 位定时器，支持间隔定时、输入捕获、输出比较，DMA 触发功能，具有 2 个独立的捕获/比较通道，每个通道使用 1 个 I/O 引脚，有 4 种工作模式：自由运行模式、倒计数模式、模模式、正计数/倒计数模式。

睡眠定时器：24 位正计数定时器，运行在 32KHz 的时钟频率，主要用于设置系统进入和退出低功耗休眠模式之间的周期。

看门狗定时器：15 位计数器，运行在 32KHz 的时钟频率，可以设置为**看门狗模式**或**定时器模式**。当它作为一个通用的定时器使用时，只有四个可选的定时间隔：1s、0.25s、15.625ms 和 1.9ms。

【26】定时器 1 工作原理概述

CC2530 的定时器 1 是一个独立的 16 位计数器，在每个活动时钟边沿**递增**或**递减**。当计

数达到最大计数值时溢出，产生一个中断请求。通过设置 T1CTL 控制寄存器**启动**或**停止**计数器，当一个不是 00 的值写入到 T1CTL.MODE 时，计数器开始运行；如果 00 写入到该位，计数器则停止在它现在的置上。

可以通过两个 8 位的寄存器**读取 16 位计数器的值**：T1CNTH 和 T1CNTL。当读取 T1CNTL 时，计数器的高位字节就会缓冲到 T1CNTH，因此，必须先读取 T1CNTL，然后再读取 T1CNTH。对 T1CNTL 寄存器的所有写入访问将会**复位 16 位计数器**。

【27】定时器最大计数值计算公式

$$最大计数值=\frac{定时时长}{定时器计数周期}=\frac{0.1S}{\frac{1}{16M}\times 128}=12500=0x30D4$$

$$最大计数值=\frac{定时时长}{定时器计数周期}=\frac{0.1S}{\frac{1}{32M}\times 128}=25000=0x61A8$$

【28】T1CCxH 定时器 1 通道 x 最大计数值高 8 位寄存器

位	位名称	复位值	操作	描述
7:0	T1CCx[15:8]	0x00	R/W	定时器 1 通道 0 到通道 4 捕获/比较值的高 8 位字节。

【29】T1CCxL 定时器 1 通道 x 最大计数值低 8 位寄存器

位	位名称	复位值	操作	描述
7:0	T1CCx[7:0]	0x00	R/W	定时器 1 通道 0 到通道 4 捕获/比较值的低 8 位字节。
设计参考	使用 16MHz 系统时钟的 128 分频作为定时器 1 的计数信号，定时 0.1 秒的最大计数值。 T1CC0L = 0xD4;　　//先写 T1CC0 寄存器的低 8 位 T1CC0H = 0x30;　　//后写 T1CC0 寄存器的高 8 位 在程序设计的时候，要注意：先写低 8 位寄存器，再写高 8 位寄存器。			

【30】T1CNTH 定时器 1 计数器高 8 位寄存器

位	位名称	复位值	操作	描述
7:0	T1CCx[15:8]	0x00	R/W	在读取 T1CNTL 时，计数器缓存的高 8 位字节。

【31】T1CNTL 定时器 1 计数器低 8 位寄存器

位	位名称	复位值	操作	描述
7:0	T1CCx[7:0]	0x00	R/W	计数器低 8 位字节。往该寄存器中写入任何值，该计数器都会被清除为 0x0000，初始化所有通道的输出引脚。

【32】T1CTL 定时器 1 控制寄存器

位	位名称	复位值	操作	描述
7:4	----	0000	R0	未使用，读为 0。
3:2	DIV[1:0]	00	R/W	定时器 1 时钟分频设置。 00：1 分频。　　01：8 分频。 10：32 分频。　　11：128 分频。
1:0	MODE[1:0]	00	R/W	定时器 1 工作模式。 00：暂停运行。　　01：自由运行模式。 10：模模式。　　11：正计数/倒计数模式。

设计参考	选择系统时钟的 128 分频作为定时器的时钟源，工作模式为模模式。 T1CTL = 0x0e;　　// 推荐对整个字节一次性赋值，0 0 0 0 1 1 1 0 注意：一旦设置了定时器 1 的工作模式，该定时器就立刻开始定时计数工作了。

【33】T1CCTL0 定时器 1 通道 0 捕获/比较控制寄存器

位	位名称	复位值	操作	描述
7	RFIRQ	0	R/W	当设置时，使用 RF 中断捕获，而不是常规捕获输入。
6	IM	1	R/W	**通道 0 中断屏蔽。** 0：禁止通道 0 中断。　　1：使能通道 0 中断。
5:3	CMP[2:0]	000	R/W	**通道 0 比较模式选择。** 当定时器的值等于 T1CC0 中的比较值，选择操作输出。 000：在比较设置输出。 001：在比较清除输出。 010：在比较切换输出。 011：在向上比较设置输出，在 0 清除。 100：在向上比较清除输出，在 0 设置。 101：TI 通道 0 没有使用。 110：TI 通道 0 没有使用。 111：初始化输出引脚，CMP[2:0]不变。
2	MODE	0	R/W	**定时器 1 通道 0 的模式选择。** 0：捕获模式。　　1：比较模式。
1:0	CAP[1:0]	00	R/W	**通道 0 捕获模式选择。** 00：未捕获。　　01：上升沿捕获。 10：下降沿捕获。　　11：所有沿捕获。
设计参考	将定时器 1 通道 0 的模式选择为比较模式 T1CCTL0 |= 0x04;　　//模模式定时，需开启通道 0 的比较模式，否则无法进入中断			

【34】T1STAT 定时器 1 状态寄存器

位	位名称	复位值	操作	描述
7:6	----	00	RO	未使用，读为 0。
5	OVFIF	0	R/W0	定时器 1 计数器溢出中断标志。 当计数器在自由运行模式或模模式下，达到最终计数值时设置，写 1 没有影响。
4	CH4IF	0	R/W0	定时器 1 通道 4 的中断标志。当通道 4 中断条件发生时设置，写 1 没有影响。
3	CH3IF	0	R/W0	定时器 1 通道 3 的中断标志。当通道 3 中断条件发生时设置，写 1 没有影响。
2	CH2IF	0	R/W0	定时器 1 通道 2 的中断标志。当通道 2 中断条件发生时设置，写 1 没有影响。
1	CH1IF	0	R/W0	定时器 1 通道 1 的中断标志。当通道 1 中断条件发生时设置，写 1 没有影响。
0	CH0IF	0	R/W0	定时器 1 通道 0 的中断标志。当通道 0 中断条件发生时设置，写 1 没有影响。

【35】定时器输入比较模式

当一个通道配置为输入捕获通道，那么和该通道相关的 I/O 引脚必须被配置为输入。

在启动定时器之后，输入引脚的一个上升沿、下降沿或任何边沿都将触发一个捕获，即把 16 位计数器内容捕获到相关的捕获寄存器中。因此，定时器可以捕获一个外部事件发生的时间。16 位捕获寄存器的内容可以从寄存器 T1CCxH:T1CCxL 中读出。

当捕获发生时，会硬件设置 IRCON.T1IF 标志和该通道的中断标志 T1STAT.CHxIF。如果初始化时分别设置了相应的中断屏蔽位 T1CCTLx.IM 和 IEN1.T1EN，将产生一个中断请求。

【36】定时器输出比较模式

在输出比较模式中，与该通道相关的 I/O 引脚设置为输出。

在启动定时器之后，将计数器和通道比较寄存器 T1CCxH:T1CCxL 的内容进行比较。如果比较寄存器等于计数器的内容，输出引脚则根据比较输出模式 T1CCTLx.CMP 中的设置进行设置、清除或切换。

当比较发生时，会硬件设置 IRCON.T1IF 标志和该通道的中断标志 T1STAT.CHxIF。如果才初始化时分别设置了相应的中断屏蔽位 T1CCTLx.IM 和 IEN1.T1EN，将产生一个中断请求。

【37】定时器 1 五个独立通道的 I/O 引脚映射

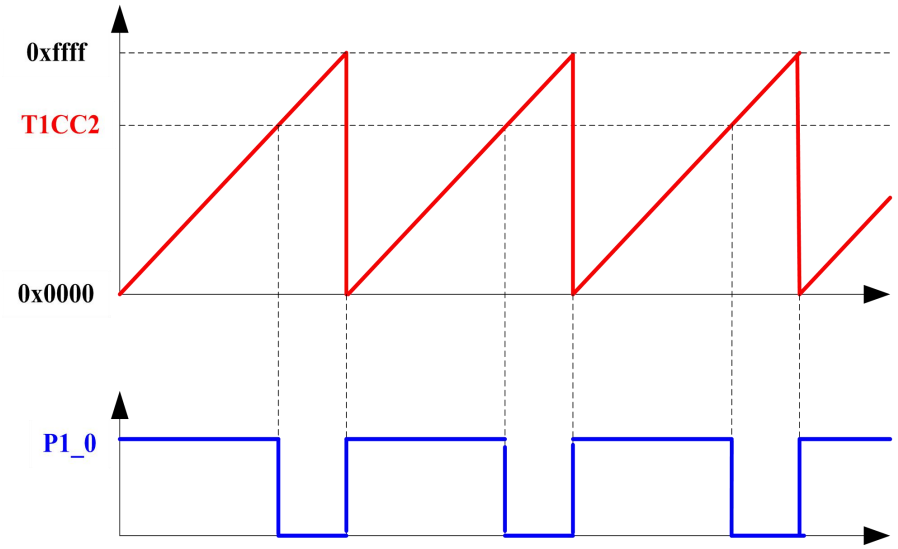
外设功能	P0 端口								P1 端口							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
备用位置 1		4	3	2	1	0										
备用位置 2	3	4												0	1	2
说　明	0：通道 0 捕获/比较引脚。　1：通道 1 捕获/比较引脚。　2：通道 2 捕获/比较引脚。 3：通道 3 捕获/比较引脚。　4：通道 4 捕获/比较引脚。															

【38】PERCFG 外设控制寄存器（结合定时器外设 I/O 引脚映射）

位	位名称	复位值	操作	描述
7	----	0	R/W	没有使用。
6	T1CFG	0	R/W	定时器 1 的 I/O 位置。 0：备用位置 1。　1：备用位置 2。
5	T3CFG	0	R/W	定时器 3 的 I/O 位置。 0：备用位置 1。　1：备用位置 2。
4	T4CFG	0	R/W	定时器 4 的 I/O 位置。 0：备用位置 1。　1：备用位置 2。
3:2	----	00	R0	没有使用。
1	U1CFG	0	R/W	USART1 的 I/O 位置。 0：备用位置 1。　1：备用位置 2。
0	U0CFG	0	R/W	USART0 的 I/O 位置。 0：备用位置 1。　1：备用位置 2。

设计参考	初始化定时器 1 通道 1 的外设映射为备用位置 2。 PERCFG |= 0x40;　//先将定时器 1 的外设映射选择位置 2，即通道 1 为 P1_1。 P1SEL |= 0x02;　//再将 P1_1 的端口功能选择为外设功能。

【39】硬件 PWM 脉冲调制信号输出原理



在上图中，P1_0 为**定时器 1 的通道 2**，在**自由运行模式**下，PWM 脉宽信号的**周期**不变，**占空比**由 T1CC2 决定。T1CCTL2.CMP 的值设置为 100，其比较模式为：**在向上比较清除输出，在 0 设置。**也就是，当定时器 1 的计数器从 0x0000 开始计数，当计数值达到 T1CC2 时，产生比较，P1_0 引脚的输出信号变为**低电平**；计数器继续到 0xFFFF 时恢复 0x0000，此时 P1_0 引脚的输出信号变为**高电平**，开始下一个 PWM 周期。

【设计参考】初始化定时器 1 通道产生 PWM 信号

```
例：采用 16MHz 系统时钟的 1 分频作为定时器 1 的计数信号，将其通道 2 的 I/O 映射到备用位置 2，即 P1_0
引脚，设计初始化函数，在 P1_0 引脚输出占空比为 20%的 PWM 信号。
unsigned int pwm_duty = 13107;          //定义 PWM 信号的占空比为 20%
void Init_PWM()
{
  PERCFG |= 0x40;                       //定时器 1 的外设引脚选择备用位置 2
  P1SEL |= 0x01;                        //定时器 1 通道 2，P1_0 设置为外设功能
  T1CCTL2 = 0x64;                       //设置定时器 1 通道 2 的比较模式
  T1CC2L = pwm_duty % 256;              //设置通道 2 的比较值，即占空比参数
  T1CC2H = pwm_duty / 256;
  T1CTL = 0x01;                         //不分频，自由运行模式，启动定时器
}
```

【40】定时器 3 和定时器 4 概述

CC2530 的**定时器 3** 和**定时器 4** 是两个 **8 位计数器**，在每个时钟边沿递增或递减。每个定时器有两个独立的比较通道，每个通道使用一个 I/O 引脚。这两个定时器有四种工作模式，分别是：**自由运行模式**，**倒计数模式**，**模模式**和**正计数/倒计数模式**。

在**倒计数模式**中，定时器启动后，计数器载入 TxCC0 的内容，然后计数器**倒计数**，直到其值为 0x00。当达到 0x00 时，设置 TIMIF.TxOVFIF 标志位，如果设置了相应的中断屏蔽位 TxCTL.OVFIM，就会产生一个中断请求。需要注意的是，倒计数模式中，定时器只运行一次，一般用于需要事件超时间隔的应用。

【41】TxCNT 定时器 3/定时器 4 计数器（“x”为 3 或 4）

位	位名称	复位值	操作	描述
7:0	CNT[7:0]	0x00	R	定时器 8 位计数器的当计数值。

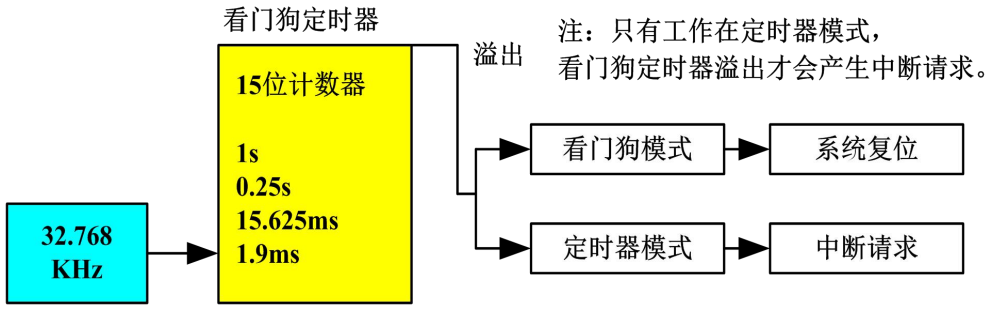
【42】TxCC0 定时器 3/定时器 4 通道 0 的捕获/比较值（“x”为 3 或 4）

位	位名称	复位值	操作	描述
7:0	VAL[7:0]	0x00	R	定时器通道 0 的捕获/比较值。

【43】TxCTL 定时器 3/定时器 4 控制寄存器（“x”为 3 或 4）

位	位名称	复位值	操作	描述
7:5	DIV[2:0]	000	R/W	**分频系数。** **000**：1 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	R/W	**启动定时器。** **0**：暂停定时器。 **1**：启动定时器。
3	OVFIM	1	R/W0	**溢出中断屏蔽。** **0**：中断禁止。 **1**：中断使能。
2	CLR	0	R0/W1	**清除计数器。** 写 1 到 CLR 复位计数器到 0x00，并初始化相关通道所有的输出引脚。读总是为 0。
1:0	MODE[1:0]	00	R/W	**定时器的工作模式。** **00**：自由运行，从 0x00 到 0xFF 反复计数。 **01**：倒计数，从 TxCC0 到 0x00 计数。 **10**：模，从 0x00 到 TxCC0 反复计数。 **11**：正计数/倒计数，从 0x00 到 TxCC0，再到 0x00 反复计数。
设计参考	选择系统时钟的 128 分频作为定时器的时钟源，工作模式为模模式，启动定时器 3。 T3CTL = 0xFE; // 对整个字节一次性赋值，1111 1110			

【44】看门狗定时器的原理结构



【45】WDCTL 看门狗控制寄存器

看门狗定时器 WDT 可以配置为一个**看门狗定时器**或一个**通用的定时器**。WDT 频率由 32kHz **时钟源**规定，包括一个 15 **位计数器**，该计数器的内容用户不能获得，它的的运行由 WDCTL 寄存器控制。

位	位名称	复位值	操作	描述
7:4	CLR[3:0]	0000	R/W	**清除定时器。** **先写 0xA 接着写 0x5** 到这些位，定时器被清除（即加载 0）. 当看门狗定时器是 IDLE 时，写这些位没有影响。当运行在定时器模式，定时器可以通过写 1 到 CLR[0]（不管其他 3 位）被清除为 0x0000（但不是停止）。
3:2	MODE[1:0]	00	R/W	**模式选择。** **00：** IDLE　　**01：** IDLE（未使用，等于 00 设置） **10：** 看门狗模式　　**11：** 定时器模式
1:0	INT[1:0]	00	R/W	**定时器间隔选择。**注意间隔只能在 WDT 处于 IDLE 时改变，这样间隔必须在定时器启动的同时设置。 **00：** 1s　　**01：** 0.25s **10：** 15.625ms　　**11：** 1.9ms
设计参考	**将看门狗定时器设为看门狗模式，喂狗时间为 0.25 秒。** WDCTL = 0x09;　　// 0000 10 01 **将看门狗定时器设为定时器模式，定时间隔为 1 秒。** WDCTL = 0x0C;　　// 0000 11 00			

【46】CLKCONCMD 时钟控制命令寄存器

位	位名称	复位值	操作	描述
7	OSC32K	1	R/W	**32KHz 时钟振荡器选择。** 设置该位置能发起一个时钟源改变，在 CLKCONSTA.OSC32K 位上，反映当前的设置。 若要改变该位，必须选择 16MHz 的内部 RC 振荡器为系统时钟。 **0：32KHz XOSC　　1：32KHz RCOSC**
6	OSC	1	R/W	**系统时钟源选择。** 设置该位置能发起一个时钟源改变，在 CLKCONSTA.OSC 位上，反映当前的设置。 **0：32MHz XOSC　　1：16MHz RCOSC**
5:3	TICKSPD[2:0]	001	R/W	**定时器标记输出设置。** 不能高于通过 OSC 位设置的系统时钟设置。 **000：32MHz　　001：16MHz** **010：8MHz　　011：4MHz** **100：2MHz　　101：1MHz** **110：500KHz　　111：250KHz**
2：0	CLKSPD[2:0]	001	R/W	**时钟速度，表示当前系统时钟频率。** 不能高于通过 OSC 位设置的系统时钟设置。 **000：32MHz　　001：16MHz** **010：8MHz　　011：4MHz** **100：2MHz　　101：1MHz** **110：500KHz　　111：250KHz** 【注意】：CLKCONCMD.CLKSPD 可以设置为任意值，但是结果受 CLKCONCMD.OSC 的限制。即如果将 CLKCONCMD.OSC 设置为 1 且 CLKCONCMD.CLKSPD 设置为 000，则 CLKCONCMD.CLKSPD 读出 001 且实际 CLKSPD 是 16MHz。

【47】CLKCONSTA：时钟控制状态寄存器

位	位名称	复位值	操作	描述
7	OSC32K	1	R	当前选择的 32KHz 时钟源。 0：32KHz XOSC　　1：32KHz RCOSC
6	OSC	1	R	当前选择的系统时钟源。 0：32MHz XOSC　　1：16MHz RCOSC
5:3	TICKSPD[2:0]	001	R	当前设置的定时器标记输出。 000：32MHz　　001：16MHz 010：8MHz　　011：4MHz 100：2MHz　　101：1MHz 110：500KHz　　111：250KHz
2：0	CLKSPD[2:0]	001	R	当前时钟速度。 000：32MHz　　001：16MHz 010：8MHz　　011：4MHz 100：2MHz　　101：1MHz 110：500KHz　　111：250KHz
设计参考	将 CC2530 的系统时钟源从 16MHz 的内部 RC 振荡器切换成 32MHz 的外部晶振。 CLKCONCMD &= ～0x40;　　//OSC 位清 0，选择系统时钟源为 32MHz 晶振 while(CLKCONSTA & 0x40);　　//等待外部晶振稳定 CLKCONCMD &= ～0x07;　　//设置当前系统时钟的速度为 32MHz			

【48】串行通信接口概述

CC2530 有 2 路功能完成相同的串行通信接口 USART0 和 USATT1，它们能够分别运行于**异步 UART 模式**或者**同步 SPI 模式**。在 UART 模式中，有 2 个独立的中断向量：**发送中断**和**接收完成中断**。当数据缓冲寄存器就绪，准备接受新的发送数据时，就产生一个中断请求。该中断在传送开始后立刻发生，也就是，**当字节正在发送时，新的字节能够装入数据缓冲器。**

【49】PERCFG 外设控制寄存器（结合串口外设 I/O 引脚映射）

位	位名称	复位值	操作	描述
7	----	0	R/W	没有使用。
6	T1CFG	0	R/W	定时器 1 的 I/O 位置。 0：备用位置 1。　　1：备用位置 2。
5	T3CFG	0	R/W	定时器 3 的 I/O 位置。 0：备用位置 1。　　1：备用位置 2。
4	T4CFG	0	R/W	定时器 4 的 I/O 位置。 0：备用位置 1。　　1：备用位置 2。
3:2	----	00	R0	没有使用。
1	U1CFG	0	R/W	USART1 的 I/O 位置。 0：备用位置 1。　　1：备用位置 2。
0	U0CFG	0	R/W	USART0 的 I/O 位置。 0：备用位置 1。　　1：备用位置 2。
设计参考	初始化定时器 1 通道 0 的外设映射为备用位置 1。 PERCFG &= ～0x01;　　//串口 0 的引脚映射到位置 1，即 P0_2 和 P0_3 P0SEL |= 0x0C;　　//将 P0_2 和 P0_3 端口设置成外设功能，即 0000 1100。			

【50】USART 串口引脚映射关系

【USART0 外设 I/O 引脚映射】

UART0	P0 端口								P1 端口							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
备用位置 1					TX	RX										
备用位置 2											TX	RX				

【USART1 外设 I/O 引脚映射】

UART1	P0 端口								P1 端口							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
备用位置 1			RX	TX												
备用位置 2									RX	TX						

【51】串口波特率的设置

【波特率的计算公式】

$$波特率=\frac{(256+BAUD\_M)\times 2^{BAUD\_E}}{2^{28}}\times F$$

（F 为系统时钟频率：16MHz 或 32MHz）

波特率(bit/s)	32MHz 的系统时钟		16MHz 的系统时钟	
	UxBAUD.BAUD_M	UxGCR.BAUD_E	UxBAUD.BAUD_M	UxGCR.BAUD_E
4800	59	7	59	8
9600	59	8	59	9
19200	59	9	59	10
57600	216	10	216	11
115200	216	11	216	12

设计参考

在 16MHz 系统时钟下，将串口 0 的波特率设置为 9600。

```
U0BAUD = 59;          //16MHz 系统时钟 9600 波特率的 UxBAUD.BAUD_M
U0GCR = 9;            //16MHz 系统时钟 9600 波特率的 UxGCR.BAUD_E
```

【52】UxBAUD USARTx 波特率控制寄存器

位	位名称	复位值	操作	描述
7:0	BAUD_M[7:0]	0x00	R/W	波特率小数部分的值。 BAUD_E 和 BAUD_M 决定了 UART 波特率和 SPI 的主 SCK 时钟。 UART 波特率的设置参数，参照本文【XX】串口波特率的设置。

【53】UxDBUF USARTx 接收/发送数据缓存寄存器

位	位名称	复位值	操作	描述
7:0	DATA[7:0]	0x00	R/W	UART 接收和发送数据。 写入该寄存器的时候，数据被写入到内部数据传送寄存器。 读取该寄存器的时候，数据来自内部的读取数据寄存器。

【54】UxGCR USARTx 通用控制寄存器

位	位名称	复位值	操作	描述
7	CPOL	0	RO/W1	SPI 的时钟极性。 0：负时钟极性。　　1：正时钟极性。
6	CPHA	0	R/W	SPI 时钟相位。 0：当 SCK 从 0 到 1 时，数据输出到 MOSI；当 SCK 从 1 到 0 时，MISO 数据输入。 1：当 SCK 从 1 到 0 时，数据输出到 MOSI；当 SCK 从 0 到 1 时，MISO 数据输入。
5	ORDER	0	R/W	传送位顺序。 0：LSB 先传送。　　1：MSB 先传送。
4:0	BAUD_E[4:0]	0	R/W	波特率指数值。 BAUD_E 和 BAUD_M 决定了 UART 波特率和 SPI 的主 SCK 时钟。

【55】UxCSR USARTx 控制和状态寄存器

位	位名称	复位值	操作	描述
7	MODE	0	R/W	USART 模式选择。 0：SPI 模式。　　1：UART 模式。
6	REN	0	R/W	UART 接收器使能。 0：禁用接收器。　　1：使能接收器。 注意：在 UART 完全配置之前不使能接收。
5	SLAVE	0	R/W	SPI 主或从模式选择。 0：SPI 主模式。　　1：SPI 从模式。
4	FE	0	R/WO	UART 数据帧错误状态。 0：无数据帧错误。 1：字节收到不正确的停止位。
3	ERR	0	R/WO	UART 奇偶错误状态。 0：无奇偶错误检测。 1：字节收到奇偶错误。
2	RX_BYTE	0	R/WO	接收字节状态。UART 模式和 SPI 模式从模式。 当读 U0DBUF 寄存器，该位自动清除，通过写 0 有效丢弃 U0DBUF 中的数据。 0：没有接收到字节。 1：准备好接收字节。
1	TX_BYTE	0	R/WO	发送字节状态。UART 模式和 SPI 模式从模式。 0：字节没有被传送。 1：写到数据缓存寄存器的最后字节被传送。
0	U0CFG	0	R/W	USART 传送/接收主动状态。在 SPI 从模式下，该位等于从模式选择。 0：USART 空闲。 1：在传送或者接收模式 USART 忙碌。

【56】UxUCR USARTx UART 控制寄存器

位	位名称	复位值	操作	描述
7	FLUSH	0	R0/W1	**清除单元。** 当设置 1 时，该事件将会立即停止当前操作并且返回单元的空闲状态。
6	FLOW	0	R/W	**UART 硬件流使能。**用 RTS 和 CTS 进行引脚流控制。 **0**：禁止流控制。　　**1**：使能流控制。
5	D9	0	R/W	**UART 奇偶校验位。** 当使能奇偶校验，写入 D9 的值决定发送的第 9 位的值；若收到的第 9 位不匹配收到字节的奇偶校验，接收时报告 ERR。 **0**：奇校验。　　**1**：偶校验。
4	BIT9	0	R/W	**UART 9 位数据使能。** **0**：8 位传送。　　**1**：9 位传送。
3	PARITY	0	R/W	**UART 奇偶校验使能。** **0**：禁用奇偶校验。　　**1**：使能奇偶校验。
2	SPB	0	R/W	**UART 停止位的位数。** **0**：1 位停止位。　　**1**：2 位停止位。
1	STOP	1	R/W	**UART 停止位的电平。** 该电平必须不同于开始位的电平。 **0**：停止位低电平。　　**1**：停止位高电平。
0	START	0	R/W	**UART 开始位的电平。** 闲置线的极性采用选择的起始位电平的相反电平。 **0**：起始位低电平。　　**1**：起始位高电平。

【57】ADC 概述

CC2530 的 ADC 支持多达 14 位的模拟数字转换，具有多达 12 位的 ENOB（有效数字位），有序列转换和单次转换 2 种方式。它有 16 个输入通道，包括 8 路外部模拟输入通道 AIN0～AIN7、4 路外部差分输入、1 路内部温度传感器电压、1 路三分之一 AVDD5 电压、正参考电压和 GND 地。它有一个参考电压发生器，可提供 4 路参考电压，分别是：AVDD 引脚、AIN7 通道、AIN6 和 AIN7 的差分电压以及内部的 1.25V 电压。

【58】APCFG 模拟 I/O 配置寄存器

位	位名称	复位值	操作	描述
7：0	APCFG[7:0]	0x00	R/W	**模拟外设 I/O 配置。** 用于选择 P0_7～P0_0 作为模拟 I/O 端口。 8 路模拟量输入来自 P0 端口组的 8 个 I/O 引脚，不必通过编程将这些引脚变为模拟输入，当相应的模拟输入引脚在 APCFG 寄存器中被禁用时，该通道将被跳过。 **0**：模拟 I/O 禁用 **1**：模拟 I/O 使用

【59】ADCH ADC 数据低位寄存器

位	位名称	复位值	操作	描述
7：0	ADC[13:6]	000000	R	ADC 转换结果的高位部分。

【60】ADCL ADC 数据低位寄存器

位	位名称	复位值	操作	描述
7：2	ADC[5:0]	000000	R	ADC 转换结果的低位部分。
1：0	----	00	R0	没有使用，读出来一直都是 0。

【61】ADCCON1 ADC 控制寄存器 1

位	位名称	复位值	操作	描述
7	EOC	0	R/H0	**转换结束。**当 ADCH 被读取的时候被清除。如果已读取前一数据之前，完成一个新的转换，EOC 位仍然为高。 0：转换没有完成。　　1：转换完成。
6	ST	0	R1/W	**开始转换。**读为 1，直到转换完成。 0：没有转换正在进行。 1：如果 ADCCON1.STSEL=11，并且没有序列正在运行，就启动一个序列转换。
5：4	STSEL[1:0]	11	R/W1	**启动选择。**选择该事件，将启动一个新的转换序列。 00：P2.0 引脚的外部触发。 01：全速，不等待触发器。 10：定时器 1 通道 0 比较事件。 11：ADCCON1.ST = 1。
3:2	RCTRL[1:0]	00	R/W	**控制 16 位随机数发生器。**当写 01 时，操作完成时设置将自动返回到 00。 00：正常运行。 01：LFSR 的时钟一次。 10：保留。 11：停止，关闭随机数发生器。
1:0	----	11	R0	没有使用，一直设为 11。

【62】ADCCON2 ADC 控制寄存器 2

位	位名称	复位值	操作	描述
7：6	SREF[1:0]	00	R/W	**选择参考电压，用于序列转换。** 00：内部参考电压。 01：AIN7 引脚上的外部参考电压。 10：AVDD5 引脚。 11：AIN6-AIN7 差分输入外部参考电压。
5：4	SDIV[1:0]	00	R/W	为包含在转换序列内的通道设置**抽取率**。抽取率也决定完成转换需要的时间和分辨率。 00：64 抽取率（7 位 ENOB）。 01：128 抽取率（9 位 ENOB）。 10：256 抽取率（10 位 ENOB）。 11：512 抽取率（12 位 ENOB）。
3：0	SCH[3:0]	0000	R/W	**序列通道选择。** 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：正电压参考。
				1110：温度传感器。	1111：AVDD/3。

【63】ADCCON3 ADC 控制寄存器 3

位	位名称	复位值	操作	描述
7：6	EREF[1:0]	00	R/W	**选择用于单通道转换的参考电压。** **00**：内部参考电压。 **01**：AIN7 引脚上的外部参考电压。 **10**：AVDD5 引脚。 **11**：AIN6-AIN7 差分输入外部参考电压。
5：4	EDIV[1:0]	00	R/W	**设置用于额外转换的抽取率。**抽取率也决定完成转换需要的时间和分辨率。 **00**：64 抽取率（7 位 ENOB）。 **01**：128 抽取率（9 位 ENOB）。 **10**：256 抽取率（10 位 ENOB）。 **11**：512 抽取率（12 位 ENOB）。
3：0	ECH[3:0]	0000	R/W	**单个通道选择。** 选择写 ADCCON3 触发单个转换所在的通道号码。当单个转换完成，该位自动清除。 **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**：正电压参考。 **1110**：温度传感器。 **1111**：AVDD/3。
设计参考	启动一次 A/D 转换，参考电压选择 AVDD5 引脚，256 抽取率，AIN3。 ADCCON3 = (0x80 | 0x20 | 0x03);　　//1010 0011 或者：ADCCON3 = 0xA3;			

【64】ST0 睡眠定时器 0

位	位名称	复位值	操作	描述
7：0	ST0[7:0]	0x00	R/W	**睡眠定时器技术/比较值。** 当读取的时候，该寄存器返回睡眠定时**计数的低位[7:0]**。当写该寄存器时，设置睡眠定时器**比较值的低位[7:0]**。 当 STLOAD.LDRDY 为 1 时，写该寄存器才有效，否则，忽略。

【65】ST1 睡眠定时器 1

位	位名称	复位值	操作	描述
7：0	ST0[7:0]	0x00	R/W	**睡眠定时器技术/比较值。**

				当读取的时候，该寄存器返回睡眠定时**计数的中间位[15:8]**。 当写该寄存器时，设置睡眠定时器**比较值的中间位[15:8]**。 在读取 ST0 寄存器时，该值是锁定的；在写 ST0 寄存器的时候，该值也是锁定的。

【66】ST2 睡眠定时器 2

位	位名称	复位值	操作	描述
7：0	ST0[7:0]	0x00	R/W	**睡眠定时器技术/比较值。** 当读取的时候，该寄存器返回睡眠定时**计数的高位[23:16]**。 当写该寄存器时，设置睡眠定时器**比较值的高位[23:16]**。 在读取 ST0 寄存器时，该值是锁定的；在写 ST0 寄存器的时候，该值也是锁定的。

【67】STLOAD 睡眠定时器加载状态寄存器

位	位名称	复位值	操作	描述
7：1	----	0000000	R0	保留。
0	LDRDY	1	R	**加载准备好。** 当睡眠定时器加载 24 位比较值时，该位是 0。 当睡眠定时器准备好开始加载一个新的比较值时，该位是 1。

【68】睡眠定时器的基本使用

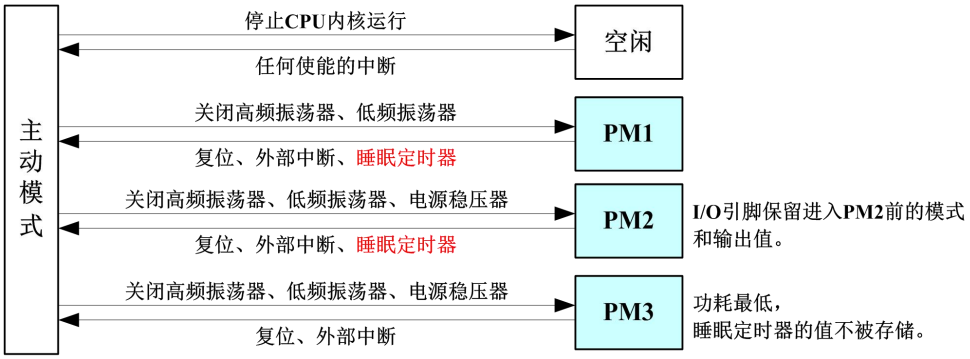
睡眠定时器是一个 24 位定时器，运行在一个 32KHz 的时钟频率（RCOSC 或 XOSC）上。该定时器在复位之后立即启动，如果没有中断就继续运行。**当睡眠定时器的计数值等于 24 位比较器的值，就会发生一次定时器比较，中断标志位 STIF 被设置**。如果中断允许，则进入中断服务函数。

定时器当前计数值可以从睡眠定时器寄存器 ST2:ST1:ST0 中读取，**ST0 寄存器必须在 ST1 和 ST2 之前读取**，以捕获一个正确的睡眠定时器计数值。当 STLOAD.LDRAY 的值为 1 时，写入 ST0 寄存器发起加载新的比较值，即写入 ST2:ST1:ST0 寄存器的最新的值。

在比较值加载期间，STLOAD.LDRDY 的值为 0，此时软件不能开始一个新的加载，直到 STLOAD.LDRDY 的值恢复为 1，才可以向 ST2:ST1:ST0 写入新的比较值，**先写 ST2 和 ST1，最后写 ST1 寄存器**。

除了 PM3 之外，在所有供电模式中，睡眠定时器都处于运行状态。在 PM1 和 PM2 下睡眠定时器比较事件用于唤醒设备，返回主动模式。

【69】五种低功耗运行模式



【70】PCON 供电模式控制寄存器

位	位名称	复位值	操作	描述
7：1	------	0000000	R/W	未使用，总是写作 0000000。
0	IDLE	0	RO/W	**供电模式控制。** 写 1 到该位强制设备进入 SLEEPCMD.MODE 设置的供电模式（注意：MODE=0x00 且 IDLE=1 将停止 CPU 内核活动）。 这个位读出来一直是 0。当活动时，所有的使能中断将清除这个位，设备将重新进入主动模式。

【71】SLEEPCMD 睡眠模式控制寄存器

位	位名称	复位值	操作	描述
7	OSC32K_CALDIS	0	R/W	**禁用 32KHz RC 振荡器校准。** **0：**使能 32KHz RC 振荡器校准。 **1：**禁止 32KHz RC 振荡器校准。 这个设置可以在任何时间写入，但是在芯片运行在 16MHz 高频 RC 振荡器之前不起作用。
6：3	----	0000	RO	保留。
2	----	1	R/W	保留，总是写作 1。
1：0	MODE[1:0]	00	R/W	**供电模式设置。** **00：**主动/空闲模式　　**01：**PM1 **10：**PM2　　**11：**PM3
设计参考	将系统切换到 PM2 运行模式 SLEEPCMD = (SLEEPCMD & 0xfc) | 0x02 ；　//运行模式设置为 PM2 PCON |= 0x01 ；　//强制设备进入 PM2 运行模式			

【72】SLEEPSTA 睡眠模式控制状态寄存器

位	位名称	复位值	操作	描述
7	OSC32K_CALDIS	0	R	**禁用 32KHz RC 振荡器校准。** 该位显示禁用 32KHz RC 校准的当前状态。在芯片运行在 32KHz RC 振荡器前，该位设置的值不等于 SLEEPCMD.OSC32K_CALDIS。
6：5	----	00	R	保留。
4：3	RST[1:0]	XX	R	**状态位，标识上一次复位的原因。** 如果有多个复位，该寄存器位只标记最新的事件。 **00：**上电复位和掉电探测　　**01：**外部复位 **10：**看门狗定时器复位　　**11：**时钟丢失复位
2：1	----	00	R	保留。
0	CLK32K	0	R	**32KHz 时钟信号（与系统时钟同步）**

注：设备的供电模式通过 SLEEPCMD.MODE 位和 PCON.IDLE 位来选择。设置 PCON 寄存器的 IDLE 位，进入 SLEEPCMD 寄存器 MODE 位所选的模式。

当设备从 PM1、PM2 和 PM3 中出来，它在 16MHz 开始，如果进入供电模式（设置 PCON.IDLE）且 CLKCONCMD.OSC=0 时，自动变为 32MHz；如果如果进入供电模式（设置 PCON.IDLE）且 CLKCONCMD.OSC=1 时，它继续运行在 16MHz。

【73】BasicRF 点对点无线开发要点概述

BasicRF 包括了 IEEE802.15.4 标准数据包的发送和接收，采用了与 IEEE802.15.4 MAC 兼容的数据包结构和 ACK 结构。在使用上，有如下的功能限制：

1-不具备“多跳”、“设备扫描”功能。

2-不提供多种网络设备，如协调器、路由器等。所有的节点为同一等级，只能实现点对点的数据传输。

3-传输时会等待信道空闲，但不按照 IEEE802.15.4 CSMA-CA 的要求进行两次 CCA 检测。

4-不支持重传。

在进行基于 BasicRF 软件包的点对点无线应用开发时，我们需要关注的有四个部分：**无线参数配置**、**无线模块初始化**、**无线数据发送**和**无线数据接收**。该软件包的其他部分，主要是管理 CC2530 硬件的内容。在实际开发中，你可以使用 BasicRF 软件包提供的 API 函数来控制硬件，也可以自己重写，就跟进行 CC2530 纯单片机应用开发一样。

【74】BasicRF 无线参数配置及初始化

BasicRF 软件包中需要配置 4 个无线参数：**本机地址**、**目标地址**、**网络 ID** 和**通信信道**。其中本机地址、网络 ID 和通信信道在初始化的时候写入到无线模块中，目标地址则是在数据发送和数据接收时使用。**值得注意的是：在进行点对点通信的各个节点中，网络 ID 和通信信道是一致的。**

在实际的应用开发中，不管是物联网技能大赛还是 1+X 证书考试，我们都不需要过多关注无线初始化底层代码的实现，只要把 4 个参数正确填好即可，如下图：

```
/*****点对点通讯地址设置******/
#define RF_CHANNEL                xx          //通信信道  11~26
#define PAN_ID                    0xxxxx      //网络ID
#define MY_ADDR                   0xxxxx      //本机地址
#define SEND_ADDR                 0xxxxx      //目标地址
/************************************************/
static basicRfCfg_t basicRfConfig;    //无线参数结构变量
// 无线RF初始化
void ConfigRf_Init(void)
{
    basicRfConfig.panId       =   PAN_ID;
    basicRfConfig.channel     =   RF_CHANNEL;
    basicRfConfig.myAddr      =   MY_ADDR;
    basicRfConfig.ackRequest  =   TRUE;
    while(basicRfInit(&basicRfConfig) == FAILED);
    basicRfReceiveOn();
}
```

【75】BasicRF 无线数据发送函数

函数原型	uint8 basicRfSendPacket(uint16 destAddr, uint8 *pPayload, uint8 length)
函数功能	将待发送数据缓冲区中指定长度的数据发送给目标地址的节点。
返回值	SUCCESS：数据发送成功。 FAILED：数据发送失败。
参数说明	destAddr：发送的目标地址。 pPayload：待发送数据的缓冲区指针。 length：待发送数据的长度。
所在文件	basicrf 文件夹 --> basic_rf.c
设计参考	uint8 send_buf[16]; uint8 status; status = basicRfSendPacket(SEND_ADDR, send_buf,16);

【76】BasicRF 判断是否已收到无线数据

函数原型	uint8 basicRfPacketIsReady(void)
函数功能	查询无线模块是否成功接收到数据。
返回值	返回 TRUE：表示已成功接收到无线数据，有数据可以读取。
参数说明	无。
所在文件	basicrf 文件夹 --> basic_rf.c
设计参考	该函数用于判断无线模块是否收到无线数据。当接收到无线数据后，相关的接收标志位置位，需要调用 basicRfReceive（）函数来读取已接收的数据，相关的接收标志位才会清除。

【77】BasicRF 无线数据读取函数

函数原型	uint8 basicRfReceive(uint8 *pRxData, uint16 len, int16 *pRssi)
函数功能	在已经接收到的无线数据中，读取指定长度的数据到数据接收缓冲区中。
返回值	实际读取到的数据长度。
参数说明	pRxData：存放读取数据的缓冲区指针。 len：需要读取数据的长度。 pRssi：保持上一次数据接收包信息变量位置，一般填 NULL。
所在文件	basicrf 文件夹 --> basic_rf.c

设计参考	先判断是否接收到无线数据，再调用该函数去读取。
	uint8 recv_buf[16];
	uint16 len;
	if(basicRfPacketIsReady() == TRUE)
	{
	len = basicRfReceive(recv_buf, 16,NULL);
	}

【78】结束语

本人自 2011 年承担 CC2530 微控制器应用开发相关课程以来，不知不觉已十年，中间踩过不少坑，绕过不少弯。因此，结合本人的教学积累及开发经验，围绕 CC2530 重要知识点和常用寄存器进行提炼整理，并附加了个人理解及应用范例，从而形成了本 CC2530 微控制器应用开发速查手册。

学无止境，由于水平局限，速查手册难免存在错漏，恳请各位勘误、斧正。