易思 ZigBee 开发教程

IAR 开发环境和驱动软件的安装

Contents

ES Technology

2014 年 10 月 12 号

版本：V2.0

1. IAR 集成开发环境的使用

本章主要讲述IAR开发环境进行CC2530SoC的开发，如果用户已经熟悉IAR开发环境，完全可以跳过本章，直接进行后面章节的学习。

1.1 IAR 集成开发环境简介

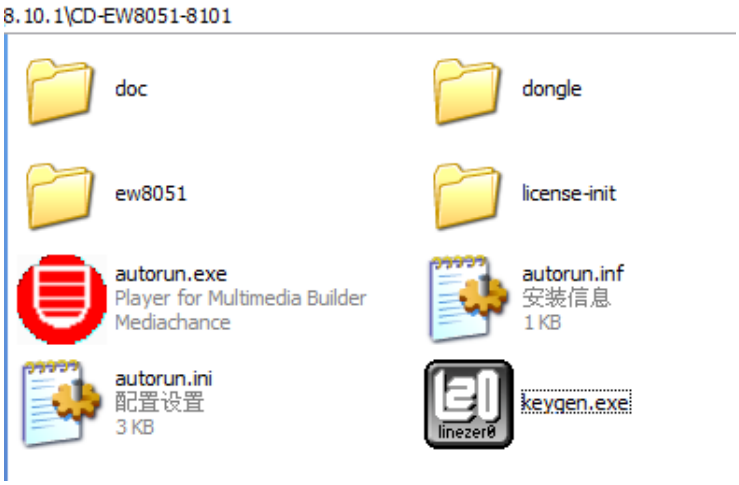
IAR Embedded Workbench（又称EM）的C交叉编译器是一款完整、稳定且容易使用的专业嵌入式应用开发工具，EW对不同的微处理器提供统一的用户界面，目前可以支持至少35种的8位、16位、32位的MCU.

1.2 系统要求和安装前注意

操作系统：Windows XP 32 & 64位，Windows Vista，Windows 7 32 & 64位，**注意使用win7系统的电脑必须是旗舰版或者企业版，并且以管理员用户安装和运行，这样才能正常工作。**Windows 7 home basic这些版本是不支持的。即使可以安装成功，在使用的时候也会有某些功能不支持。

1.3 安装 IAR8.10

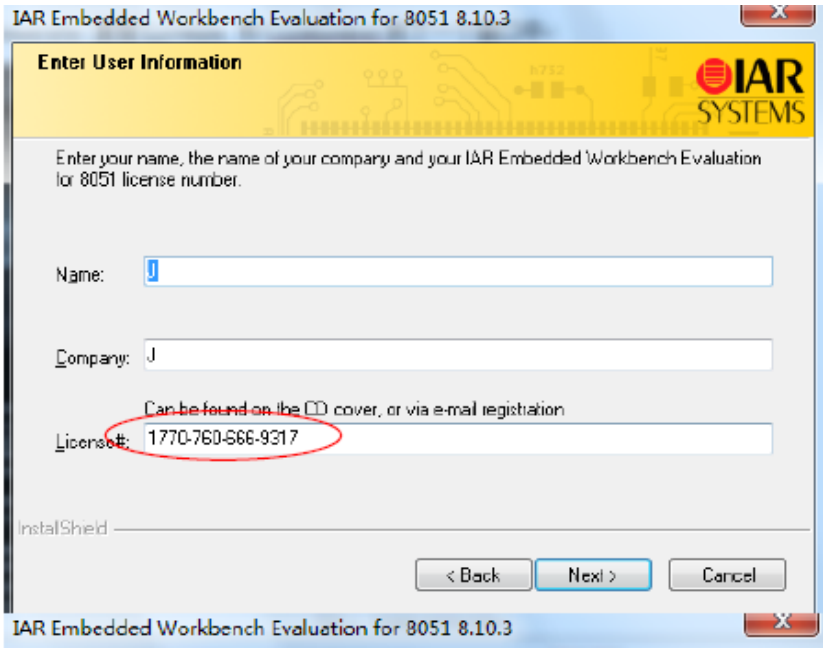
双击运行autorun.exe，然后再跳出的画面中选择第二项，Install IAR Embedded Workbench

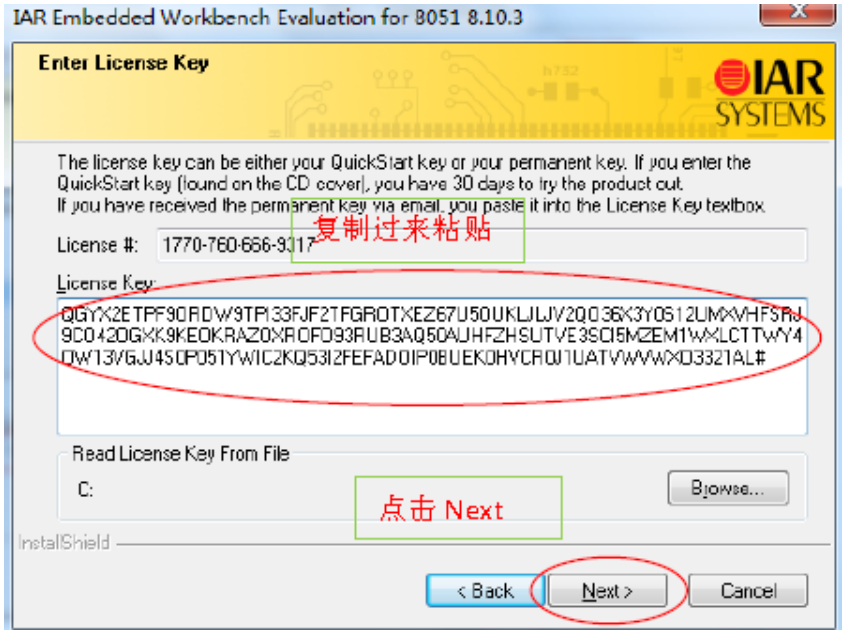


根据提示一路next，到Enter User Information这一项，提示输入license。

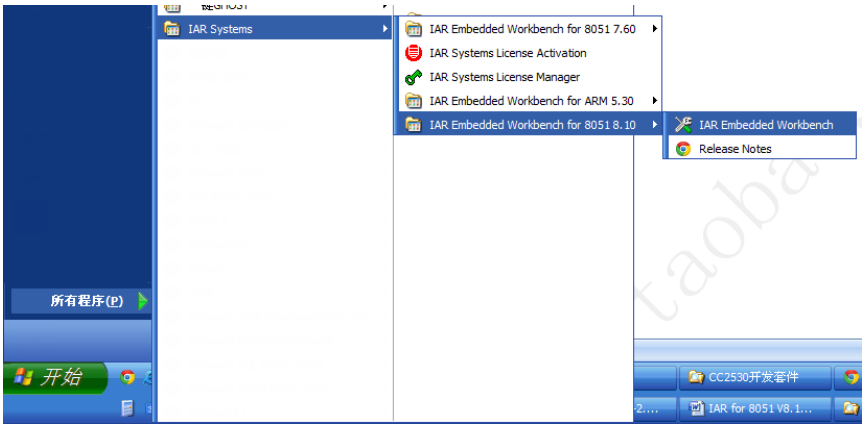
这里使用keygen生成iar的license，如果大家有能力使用正版，最好使用正版软件。运行资料里的keygen生成license。位于如下图位置。运行后选择第一项for MCS-51 v8.10。

复制license number到iar安装向导中，next后再复制license key

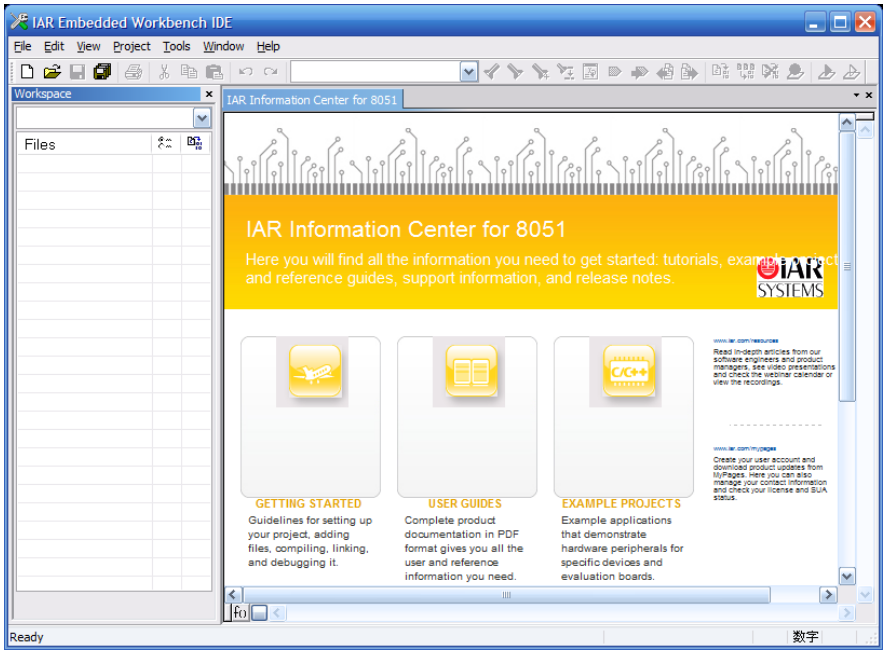




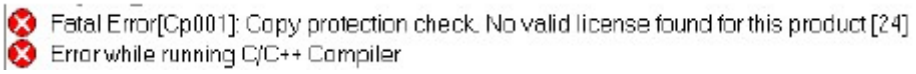
最后一路next，安装结束后在开始菜单中找到IAR软件，默认安装的位置如下图：



运行的IAR软件如下图：



如果再iar的使用过程中出现如下错误

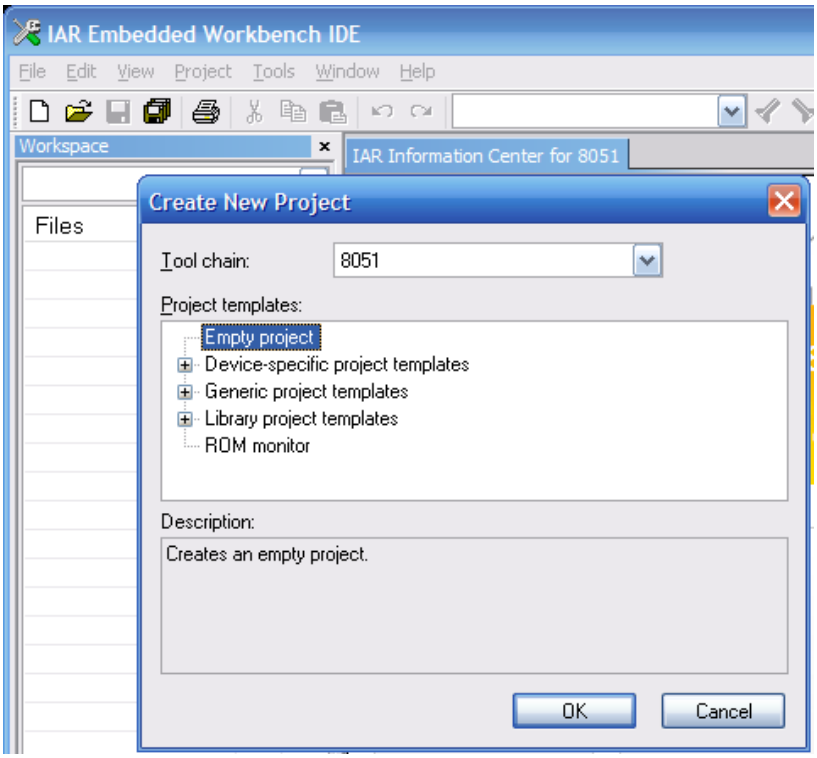


这是由于iar的license未能安装成功，多数有由于未能使用管理员运行注册机导致。

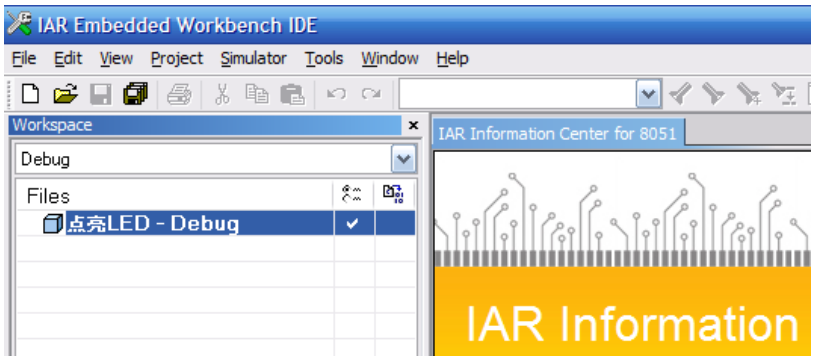
1.4 IAR 的使用

1.4.1 建立一个新工程

打开iar，点击菜单栏的Project，在弹出的下拉惨淡中选择Create NewProject，如图所示

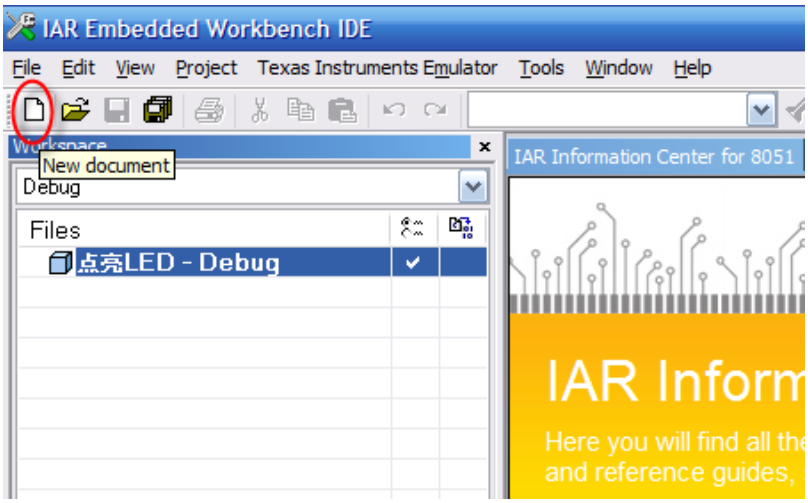


选择“Empty project”，单击OK，然后会询问保存project，选择一个合适的目录，我这里保存的目录在CC2530基础测试程序\1_点亮LED 目录下，然后填入合适的工程名，然后单击OK，如下图：

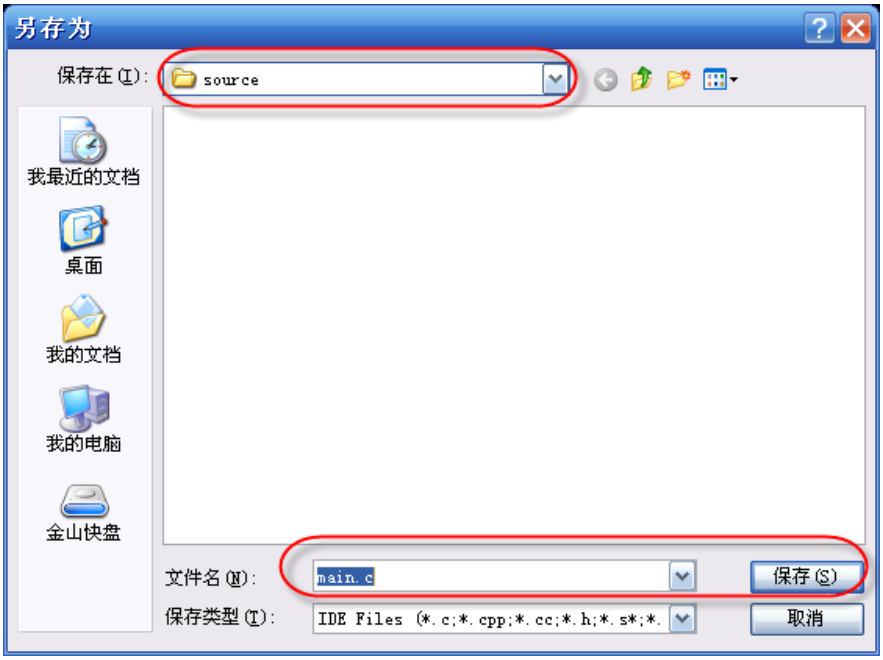


1.4.2 建立一个源文件

单击New document按钮，新建一个文本文件。

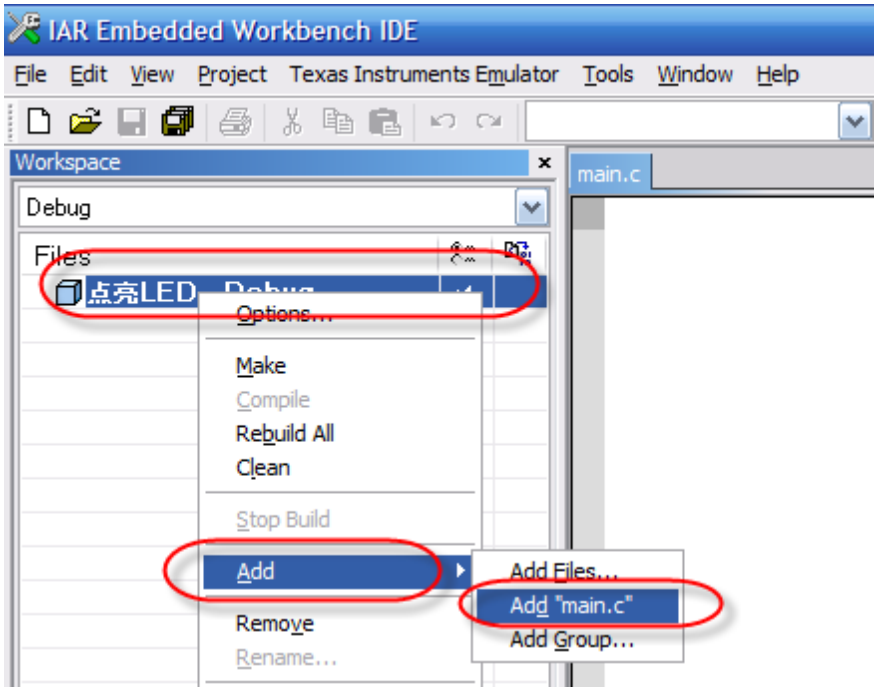


新建了文件之后单价保存按钮，保存为文件名为：main.c到source目录下（source是在IAR工程目录内新建的用来专门保存源码的目录）



1.4.3 添加源文件到工程

右击工程名，选择add->Add main.c，也可以使用add files，手动选择main.c

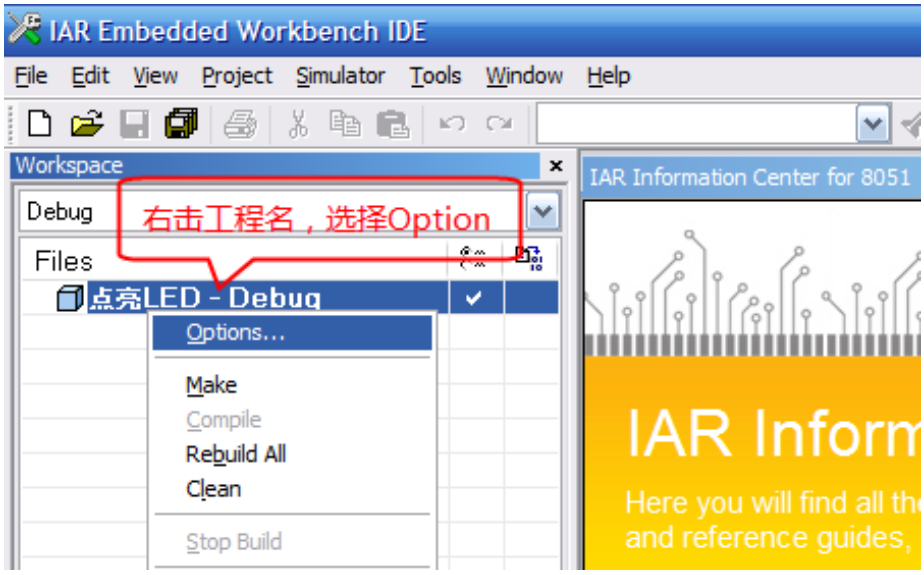


向main.c中输入一下代码，然后保存，代码如下：

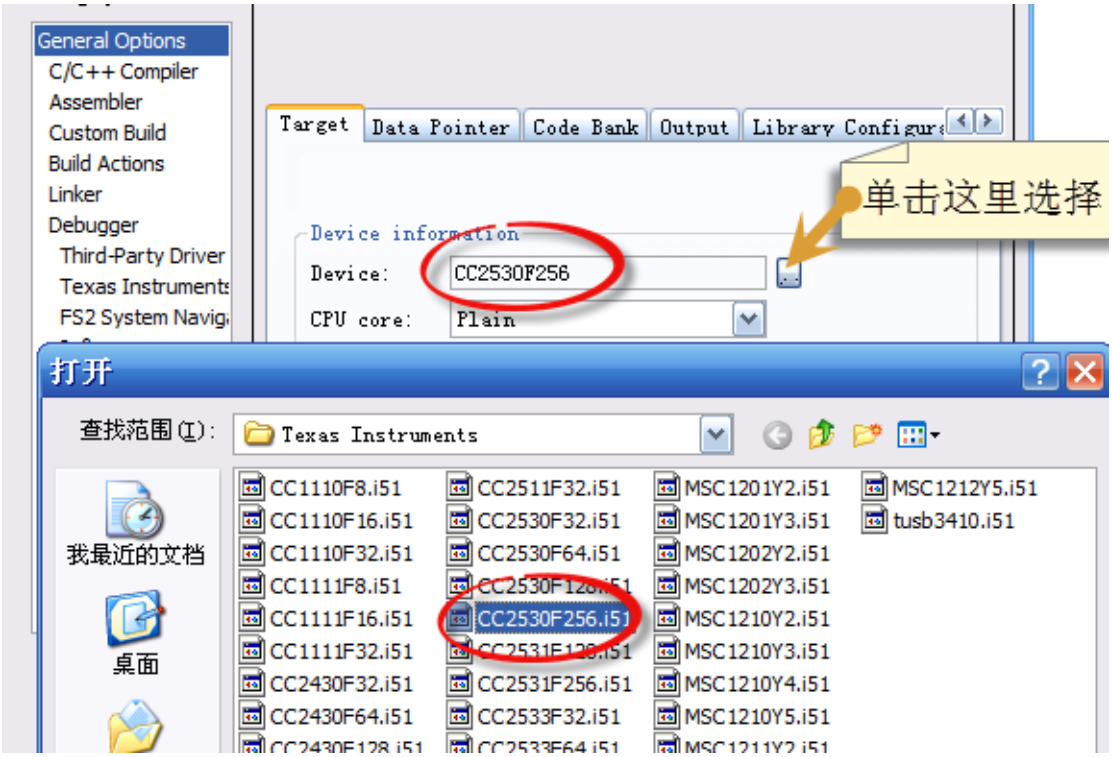
```
#include <ioCC2530.h>
#define BV(n) (1<<(n))

void delay(unsigned int time)
{
   int i,j;
   for(i=0;i<time;i++)
      for(j=0;j<1000;j++);
}
int main()
{
   //设置 P1.4 端口方向为输出
   P1DIR |= BV(4);

   //设置 P1.4 端口为 GPIO 功能
   P1SEL &= ~BV(4);

   while(1){
      P1_4=1;//点亮 led
      delay(1000);
      P1_4=0;//熄灭 led
      delay(1000);
   }
}
```

1.4.4 工程设置

在左边的Workspace中右击工程名，然后选择Option，进入Option工程配置对话框，注意，Option对话框我们将经常用到，先记住它是如何打开的，如下图：



配置目标芯片在出现的对话框中，第一件事情就是选择该project所使用的Device，左边选择General Option，然后在右边的一些列的选项卡中选择Target，如下图，设置Device，我们这里使用的芯片是CC2530F256，因此选择CC2530F256.i51，（该文件的完整的默认路径为：C:\Program Files\IAR Systems\Embedded Workbench 6.0\8051\config\devices\Texas Instruments）



- 设置Code和Memory Model

在code类型中有Near和Banked两项可选择

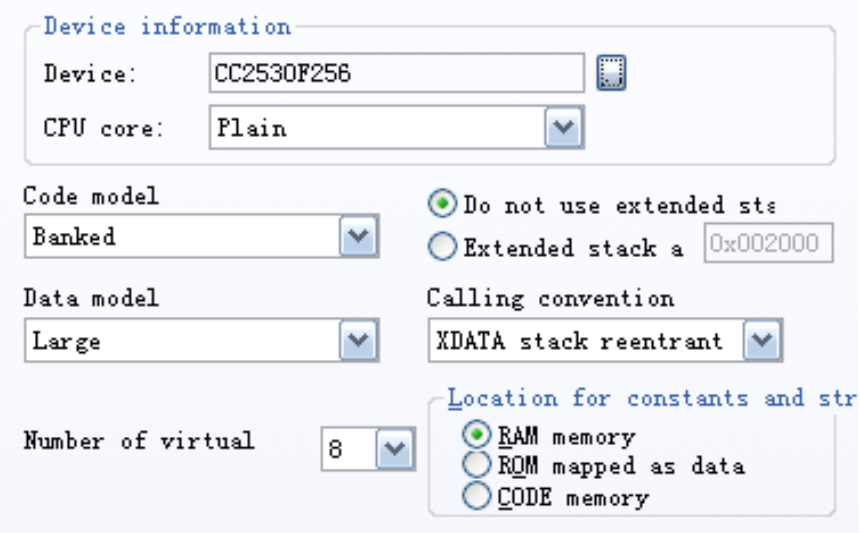
“Near”当不需要Bank支持可以选择Near，例如，你只需要访问64K flash空间的时候，不需要更多的flash空间，比如你使用的是CC2530F32或CC2530F64，或者使用的CC2530F256但并不需要那么大的flash空间时，可以选择Near。

“Banked”选择该项时标明你需要更多的空间能够仿真CC253xF128或者CC253xF256的整个Flash空间。

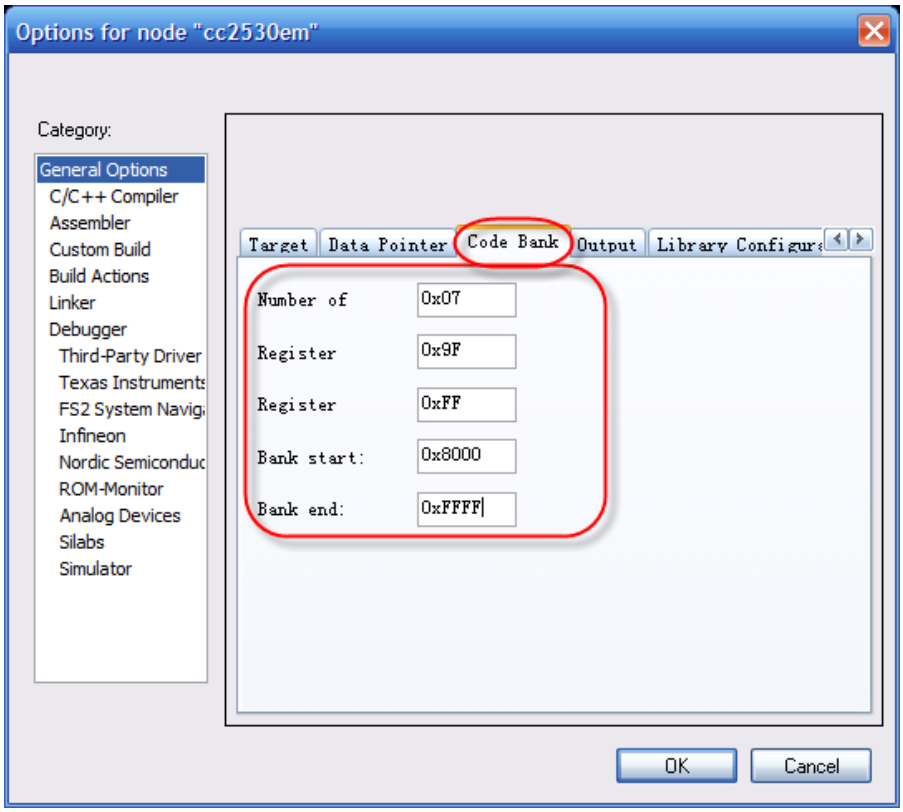
默认Near code model中的data model是Small，默认的Banked，data model为Large，data model决定编译器或者连接器如何使用8051的内存来存储变量，选择small data model，变量典型的存储在DATA内存空间，如果使用Large data model，变量存储在XDATA空间。在CC2530用户手册和IAR 8051编译器参考手册中会详细描述变量内存空间。

在这里，重要的事情是，8051使用不同的指令来访问various memory spaces访问IDATA，一般情况下，比仿真XDATA要快，但通常XDATA的空间会比IDATA大。

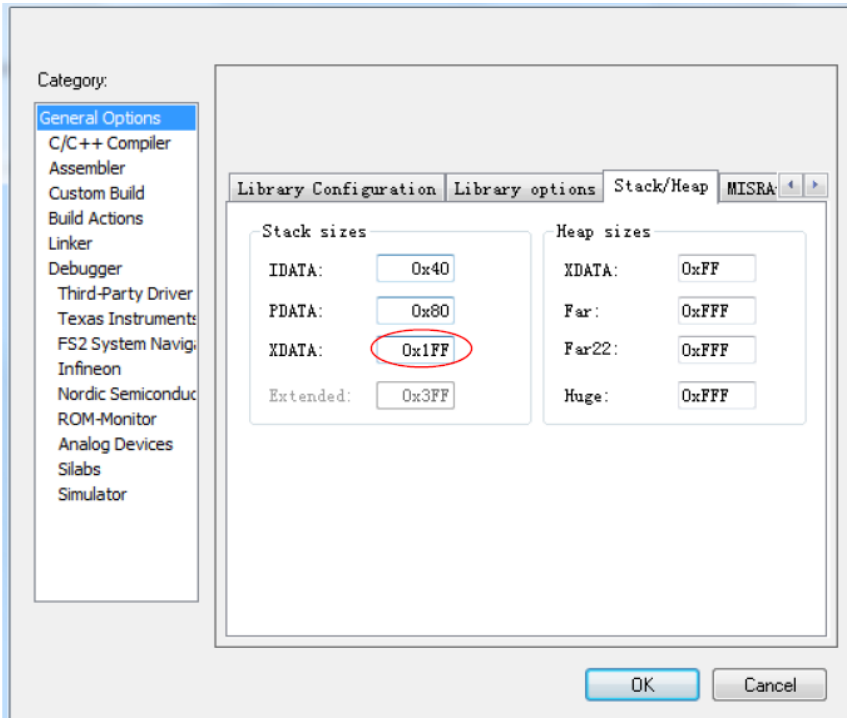
在Z-STACK协议栈中，使用large memory model来支持CC2530F256，这样协议栈可以存储在XDATA区域，以上设置结束后，如下图所示。



在Banked code model中，有一些额外的选项需要注意，选择Code Bank tab，如下图，CC2530使用7个code banks，为了访问整个256K的Flash空间，Number of必须设置为0x07，Register 0x9F是CC2530的FMAP寄存器，用来控制当前那个code bank映射到8051的地址空间，第三个Register未使用，最好设置0xFF。

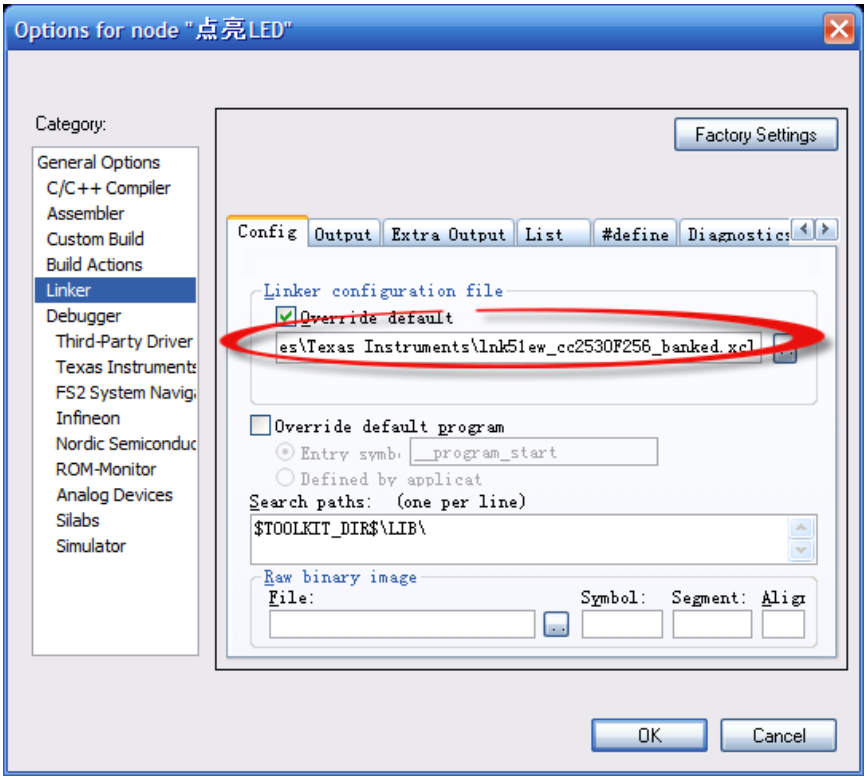


在Stack/heap标签，XDATD文本框内设置为0X1FF



- 设置链接器

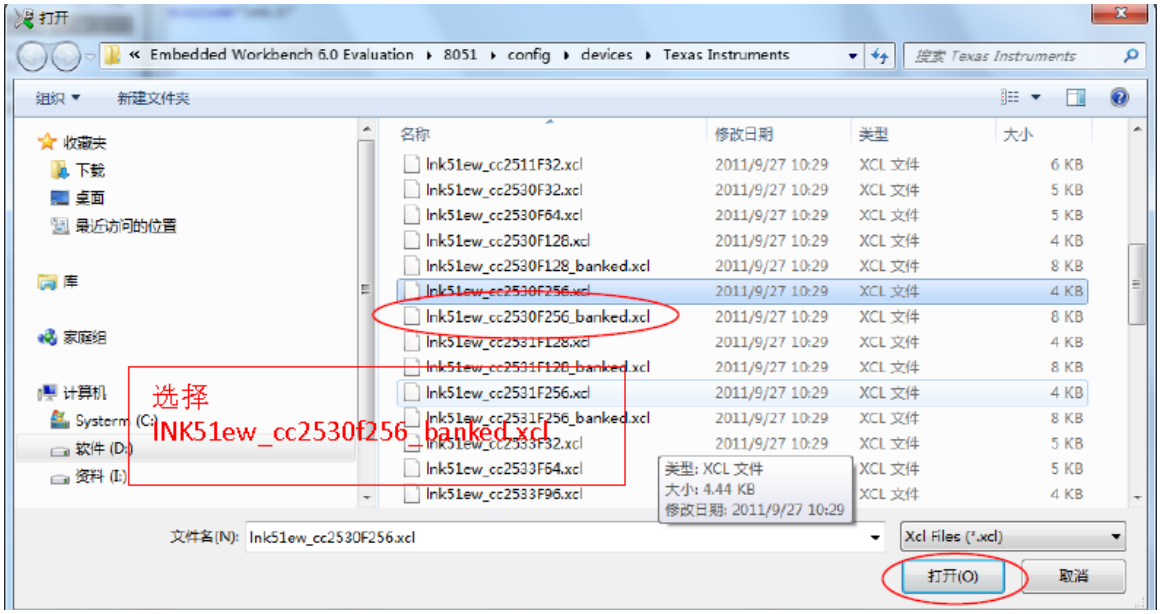
在左边的选项中选择Linker，并在右边的选项卡中选择Config一页，在Linker Command file 中复选Override default，例如，我们选择lnk51ew_CC2530F256_banked.xcl，banked表示使用banked code model。



默认路径为：

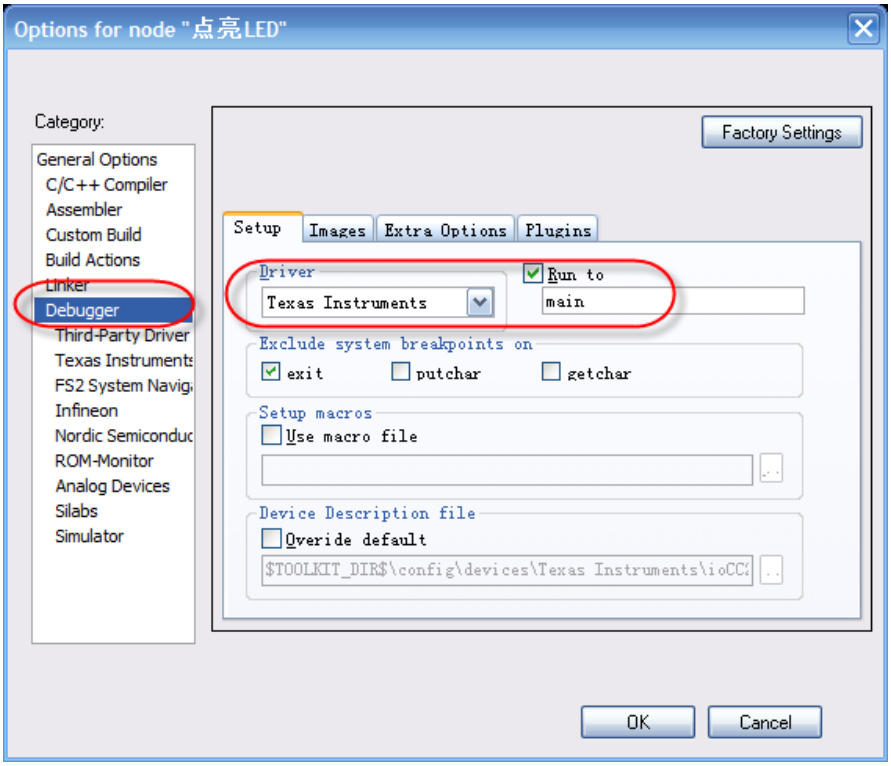
$TOOLKIT_DIR$\config\devices\Texas Instruments\lnk51ew_CC2530F256_banked.xcl

或者自己定位到安装目录下：



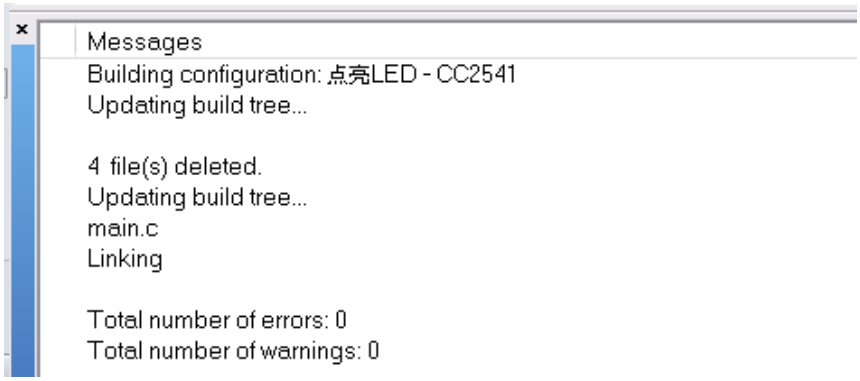
- 设置仿真器调试

最后，在Debugger选项中，选择Texas Instruments为Driver。



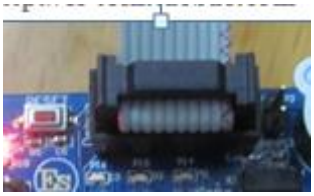
1.4.5 源文件的编译和下载

编译过程中如果出现错误，请根据错误提示修改不小心造成的语法问题。Project-Make 编译后显示0错误和0警告。注意有时候编译会有一些warning，这个可以暂时不管，直接下载程序运行调试即可。

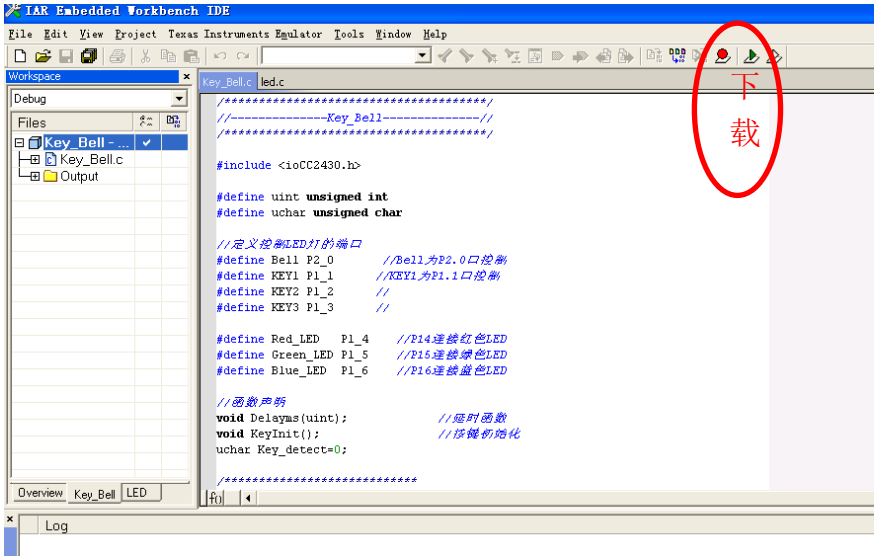


在运行代码之前，首先将仿真器和开发板连接好，仿真器一端使用USB线连接电脑，一端通过10芯排线连接到开发模块上。开发板只有一个接口可以和仿真

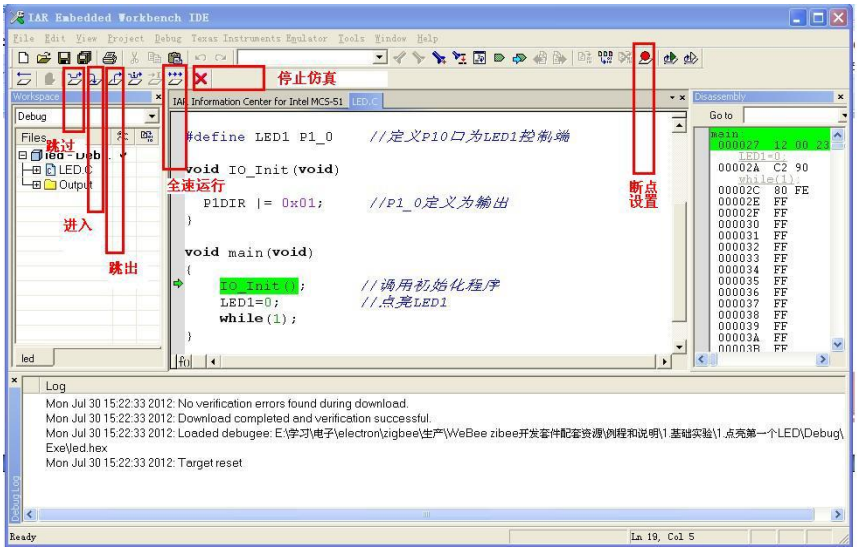
器的10芯排线连接，一看就知道的，如下图：



需要注意的是，不管使用CC-Debugger仿真器还是SmartRF04EB仿真器，在调试下载程序之间，**第一次下载程序时需要按仿真器的复位按钮**，再进行下载操作，否则会导致下载错误。



下载完成，进入仿真调试界面，常用按钮如下图所示。



点击GO(全速运行)，程序执行。使仿真器可以直接在IAR中下载程序并调试。结束后程序仍然保留在芯片flash内，相当于烧写工具。非常方便。

1.5 打开一个保存的 IAR 工程

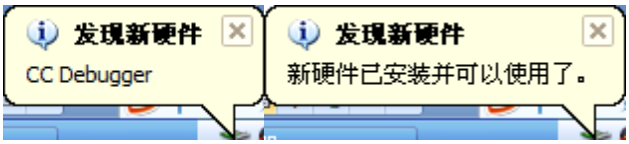
我们已经创建好了所有的基础测试程序，他们被创建在一个IAR工程内。我们建立的IAR工程所有工程参数已经设置好，不需要再设置，可以直接下载。但是因为可能电脑系统不一样或者安装目录不一样，会导致linker文件找不到或者device设置的改变，请仔细对比按照上述IAR的配置设置工程文件的参数。

打开的方法可以在IAR界面中点击 File->Open->Workspace打开.eww工程文件或者直接双击.eww文件进行打开。

2. 仿真器驱动和其他软件的安装

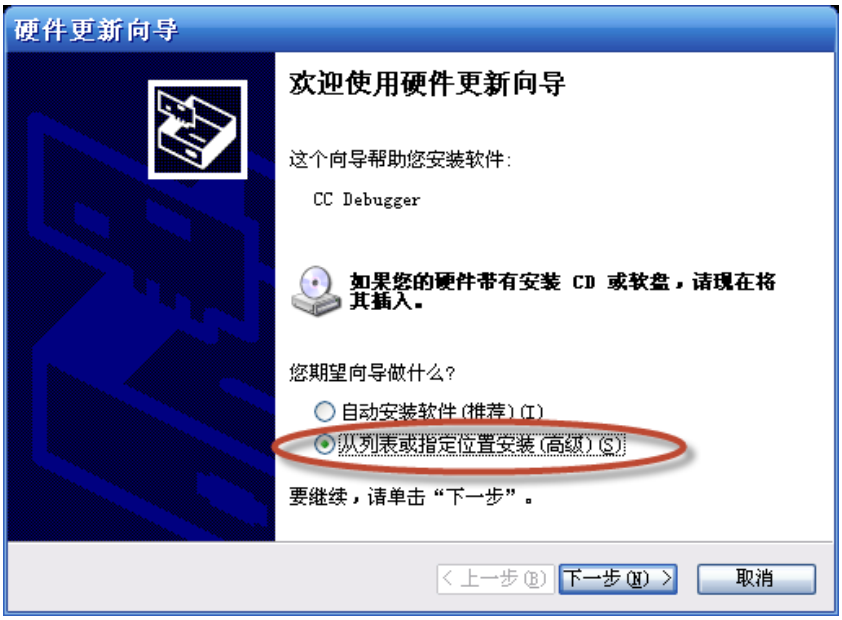
2.1 仿真器驱动安装

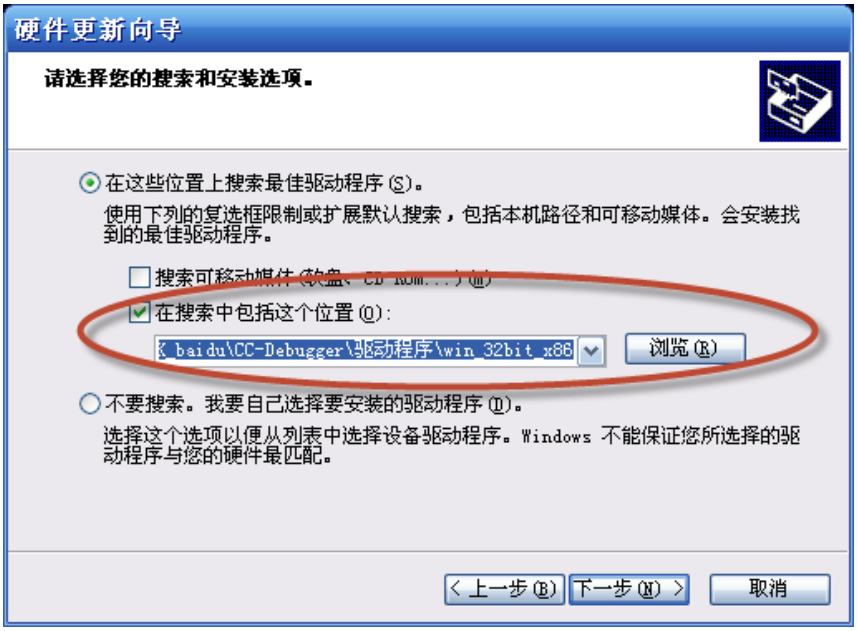
连接CC-Debugger仿真器或者SmartRF04eb仿真器，PC通知栏会告知发现新硬件，如果电脑中存在仿真器的驱动，则随机会跳出驱动安装成功的提示：



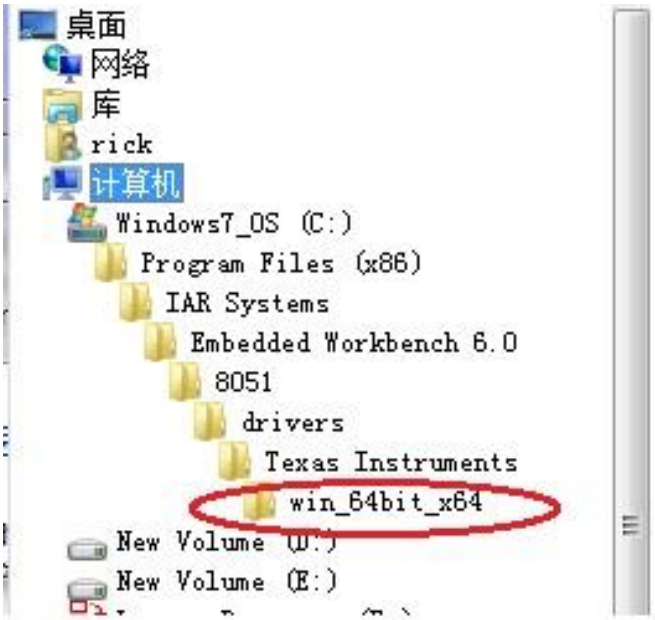
如果没有，出现安装成功。则先确定cc-debugger是否与pcusb连接OK，连接OK后，电脑上肯定会出现新新硬件的提醒。如果插上电脑没有任何反应，更换mini-usb线。如果有提示新硬件，但显示驱动未能安装成功，打开电脑的设备管理器。如下图，

如果框起来的有感叹号或者别的，需要更新一下仿真器的驱动。右击CC Debugger，或者SmartRF04EB，选择更新驱动程序。



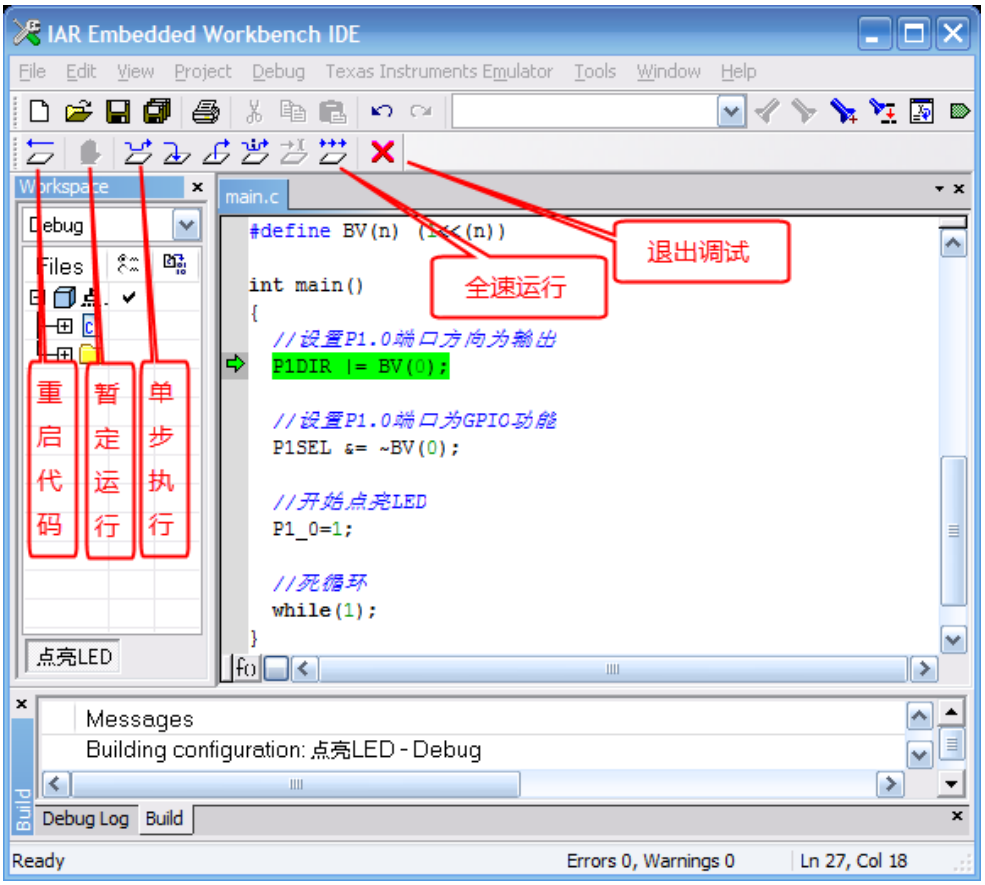


定位到仿真器驱动即可，驱动地址在IAR安装目录下：C:\Program Files\IAR Systems\Embedded Workbench 6.0\8051\drivers\Texas Instruments\srf04eb



- 程序仿真调试

程序仿真调试界面截图如下。

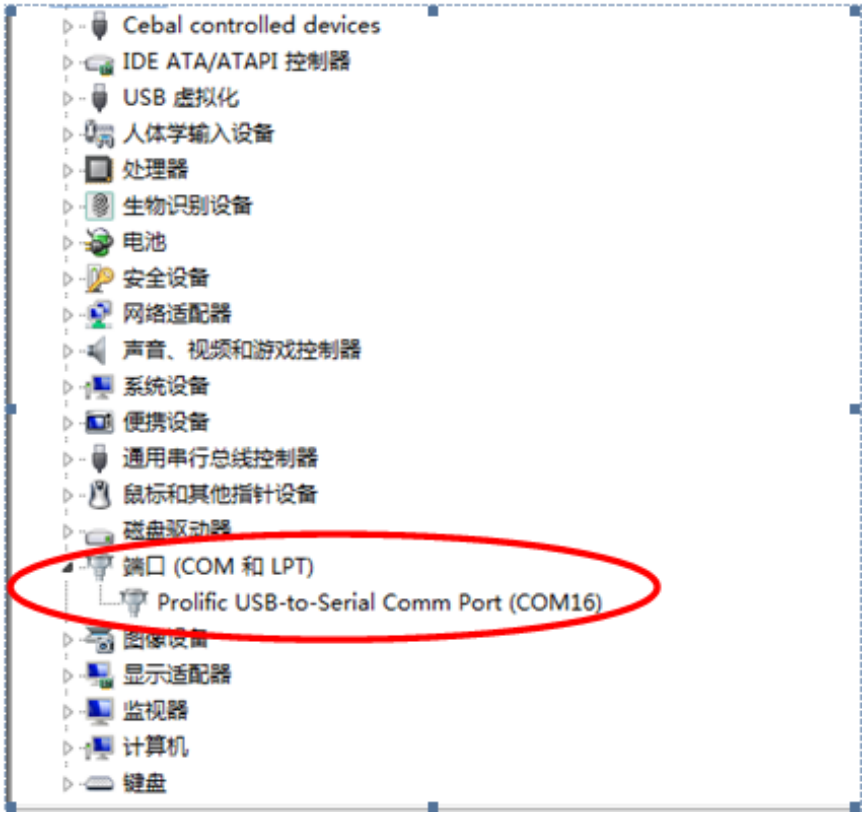


2.2 USB 转串口驱动（PL2303 驱动）

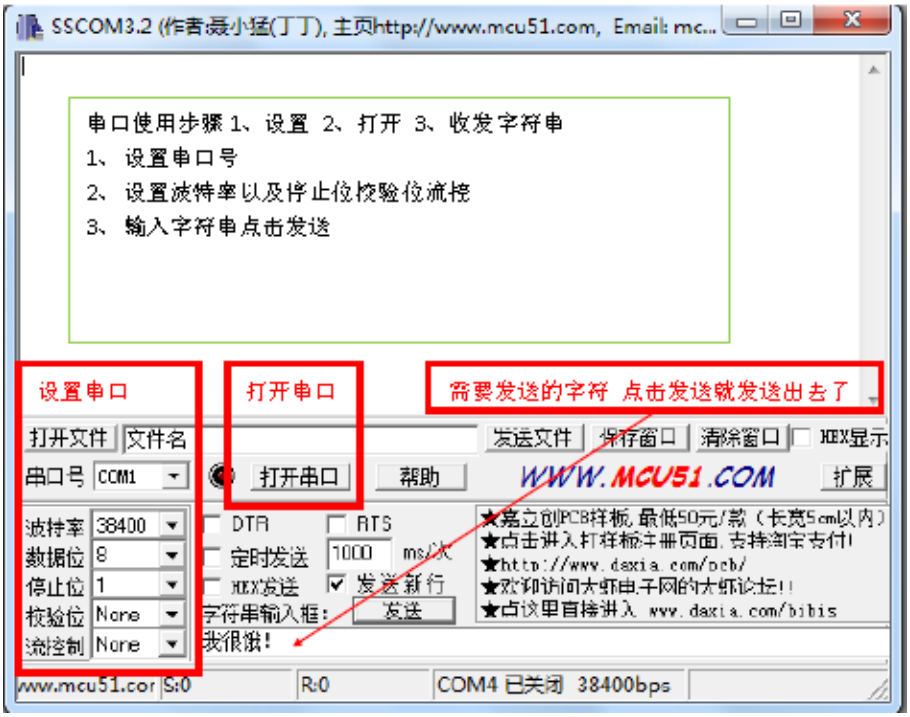
实现模块与计算机通信必须安装USB转串口驱动。模块板子已经集成USB转串口模块PL2303，只要在电脑端安装PL2303驱动，即可以实现板子与PC进行串口通信。安装的驱动非常简单，只要双击PL2303_driver.exe,然后点确定，安装到底就可以。驱动文件如下图所示。

本驱动适合Xp系统和win7系统，win8系统暂时不支持。

安装完驱动就可以在设备管理器，硬件里看到新增加的串口（Com port），如下图所示。



使用串口调试助手可以观察电脑的串口数据，串口调试助手的界面如下图所示。

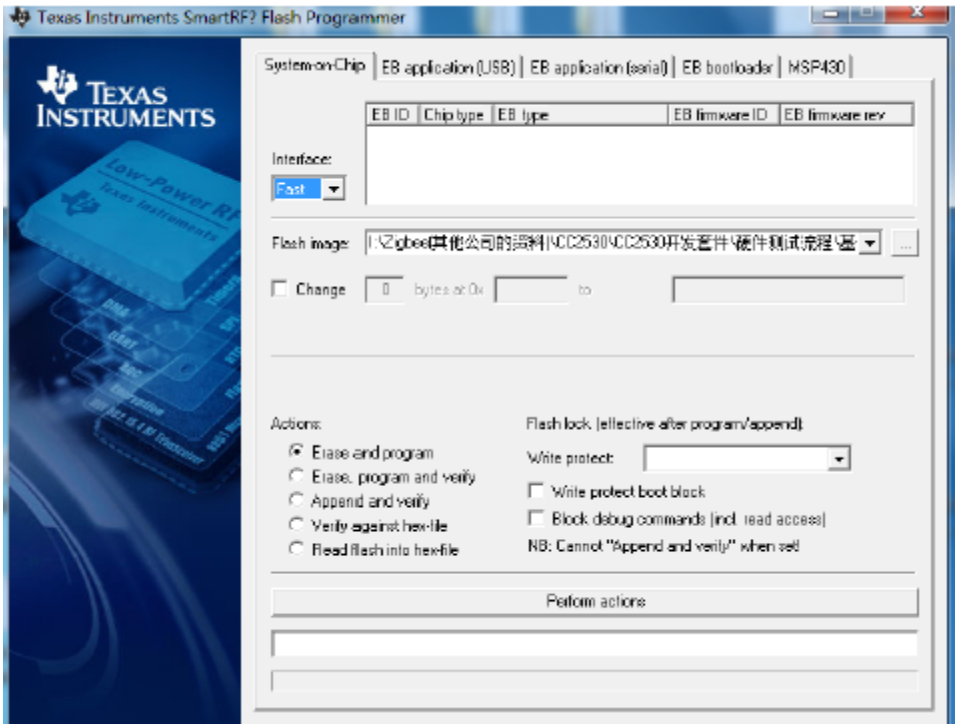


2.3 使用 Flash Programmer 直接烧写 hex 到芯片中。

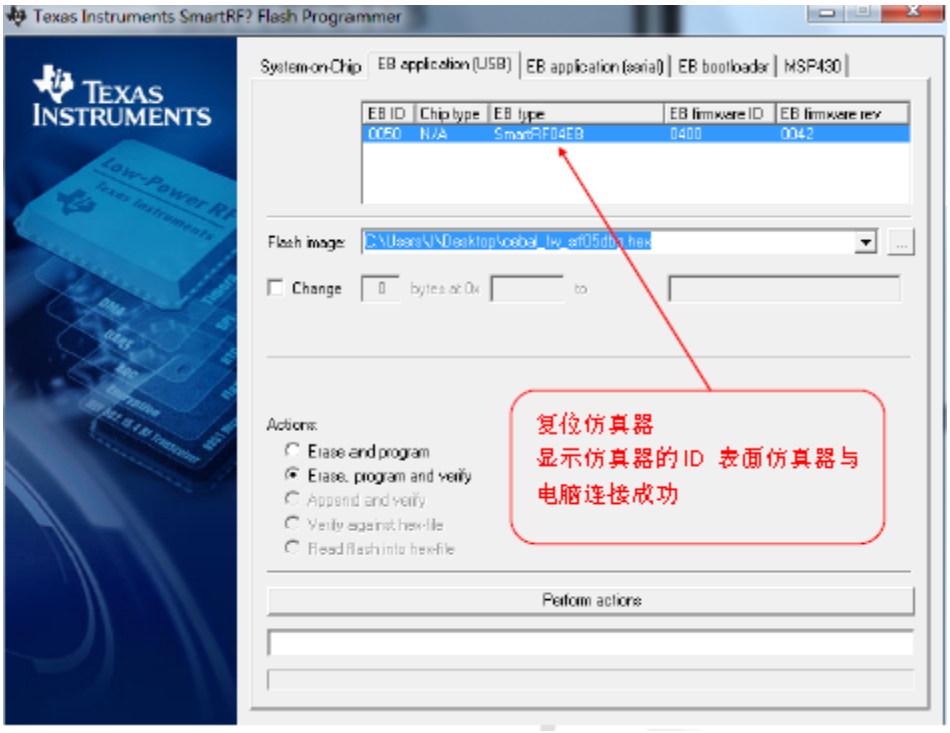
我们详细的介绍一下使用TI的flash programmer烧写工具烧写hex文件，flash programmer是ti开发的hex文件烧写工具，通过Flash Programmer不光可以给目标芯片烧写程序，而且还而已更新CC-Debugger仿真器的固件程序，功能非常强大。

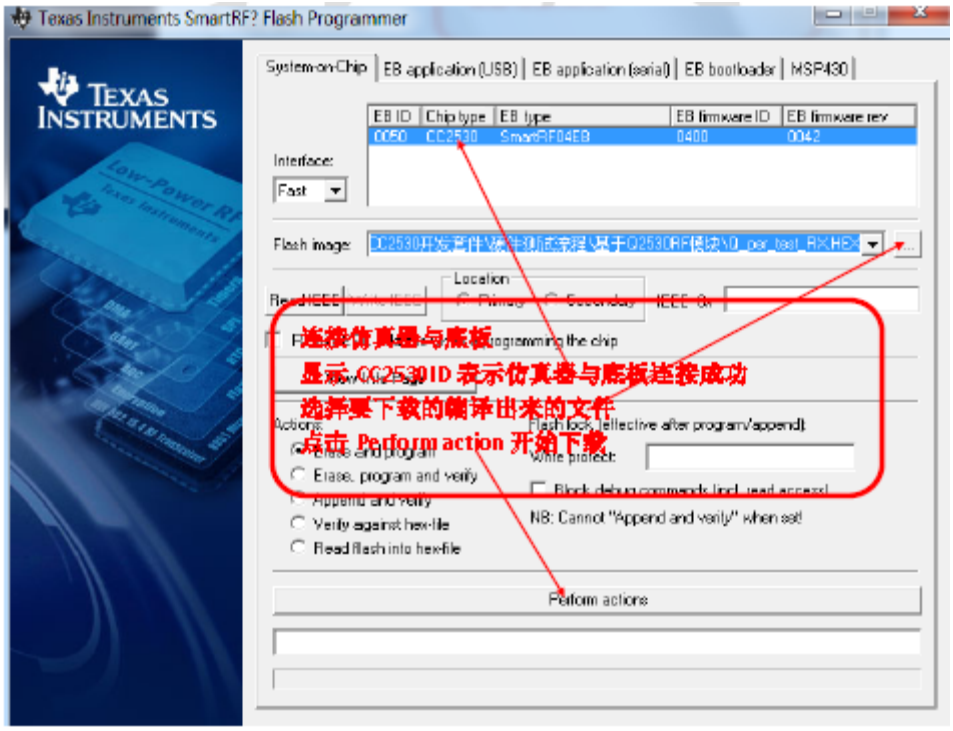
首先点击SmartRF Flash Programmer安装文件进行安装，按照默认直接安装即可，无需破解和特别设置。安装完成如下图所示：

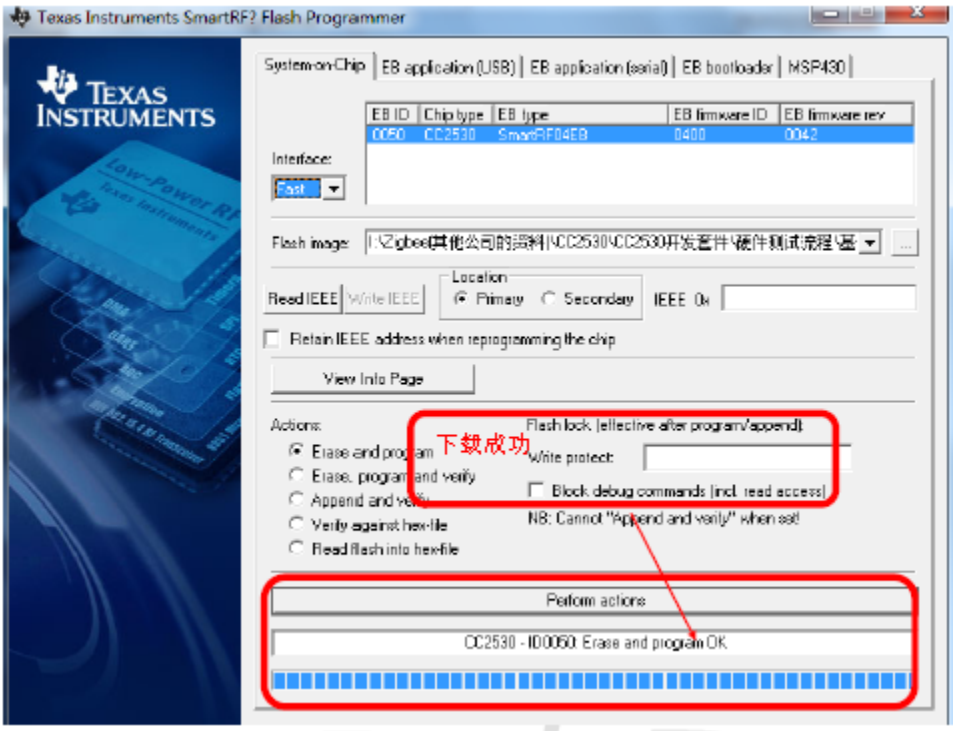
打开软件如下图所示：



接上仿真器和开发板，按下仿真器的复位键，软件即可识别到开发板，如下图所示：



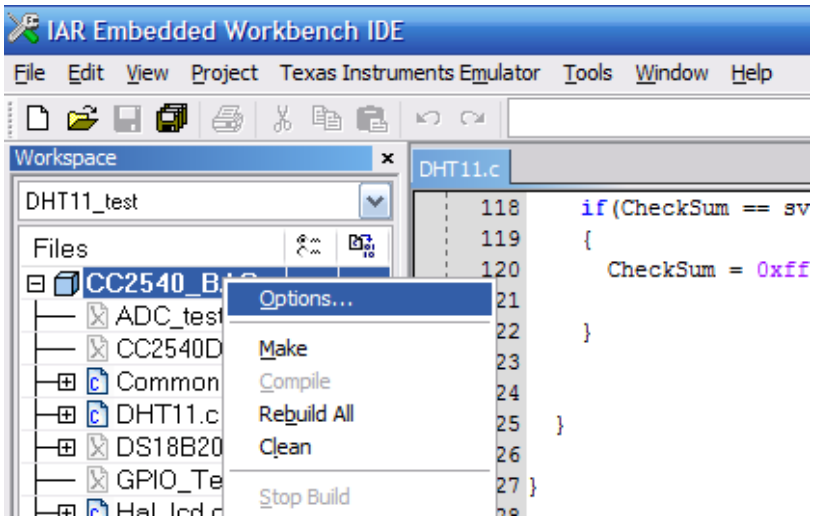




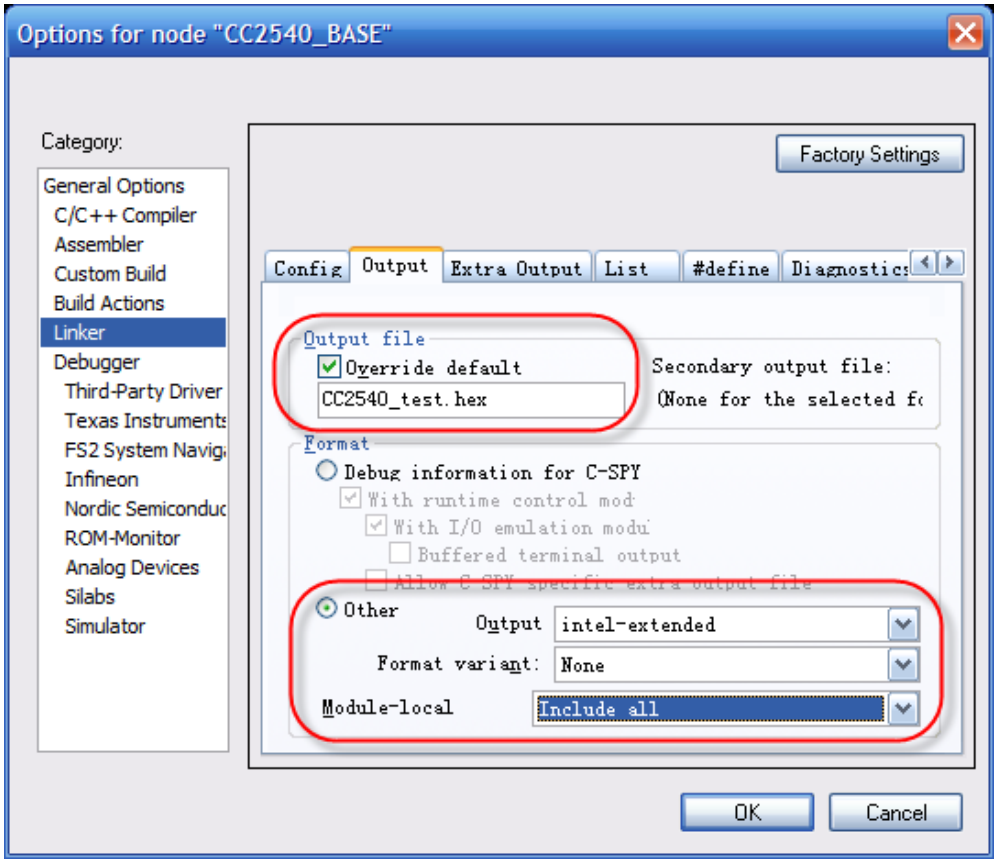
2.4 Hex 文件的生成和下载。

设置IAR产生Hex文件，以CC2540EM测试代码为例，如使用2530EM，相应处选择CC2530xxx即可。

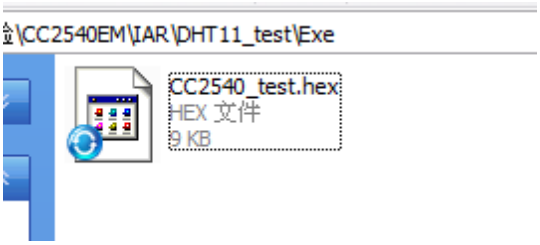
打开CC2540EM_BASE.eww，右击Workspace中的项目名称，然后打开Options对话框。如下图。



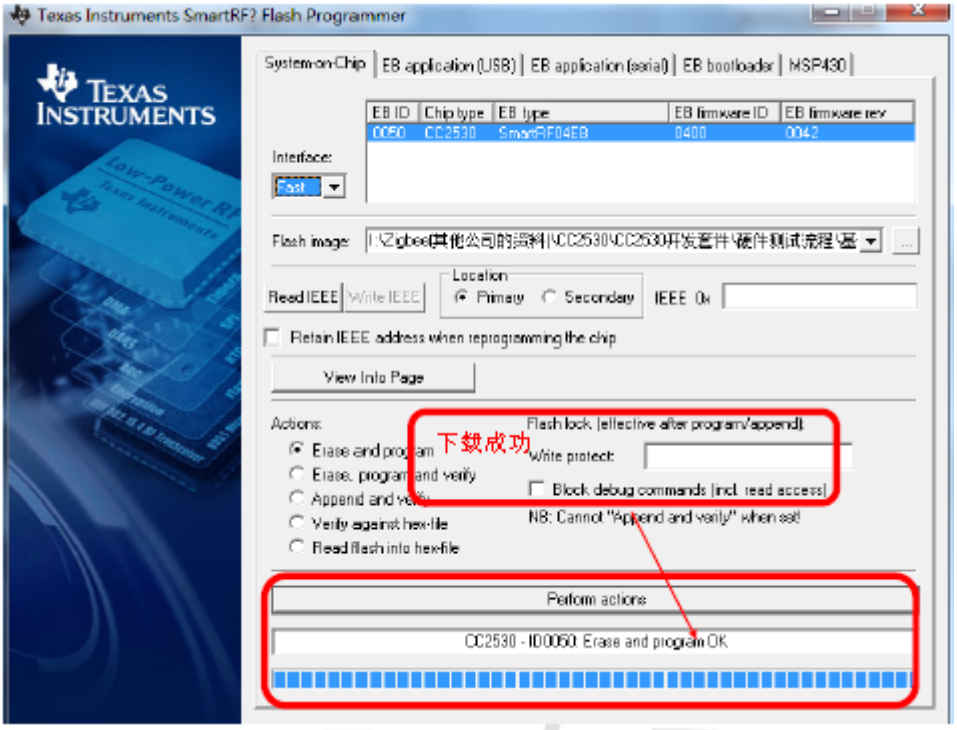
在打开的Options中选择Linkder中的Output选项卡，设置如下图：



然后重新编译工程，这时，会在工程的相应配置目录（对应Workspace中列表框中的名称）下的/Exe文件夹下产生可烧写的hex文件。如下图：



下载过程如下图所示：



Action中的常见操作：

Erase：擦除芯片，当遇到芯片被锁住时，可以点该选项擦除芯片。

Erase and program：擦除并且编程，快速对芯片编程

Erase program and verify：步骤2基础上添加hex验证操作，比较耗时。

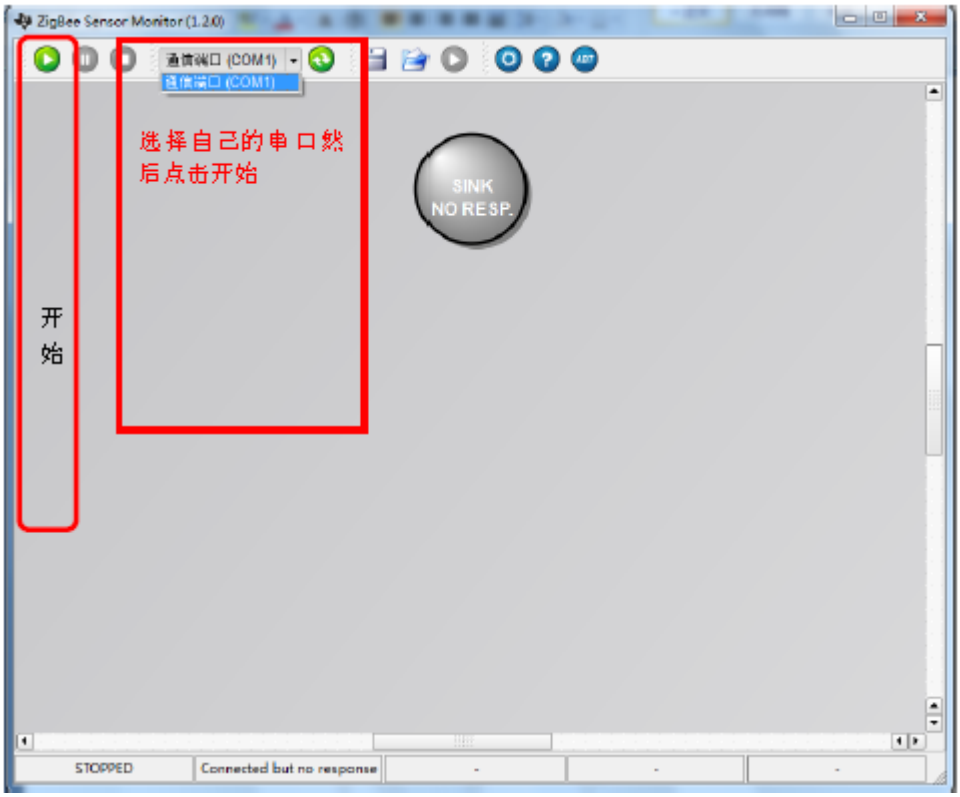
Read flash into hex-file：从芯片中读出hex，并且写到（覆盖）在Flash image中选择的hex文件。

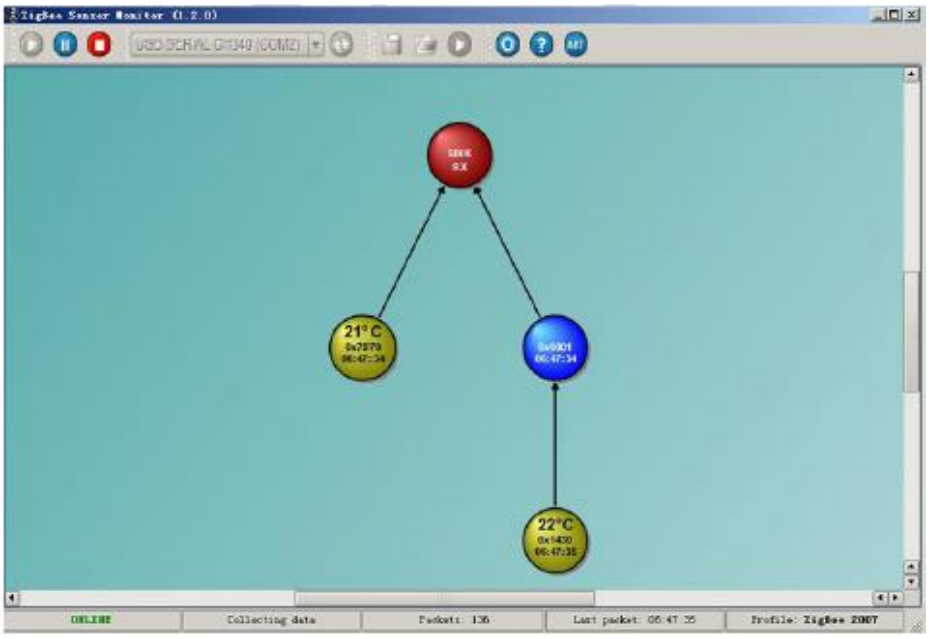
其他选项含义请查阅ti帮助手册。

2.5 ZigBee Sensor Monitor 的安装

双击ZigBee Sensor Monitor安装文件，直接进行安装，安装完成如下图所示：

安装完成后，打开软件，界面如下图所示：

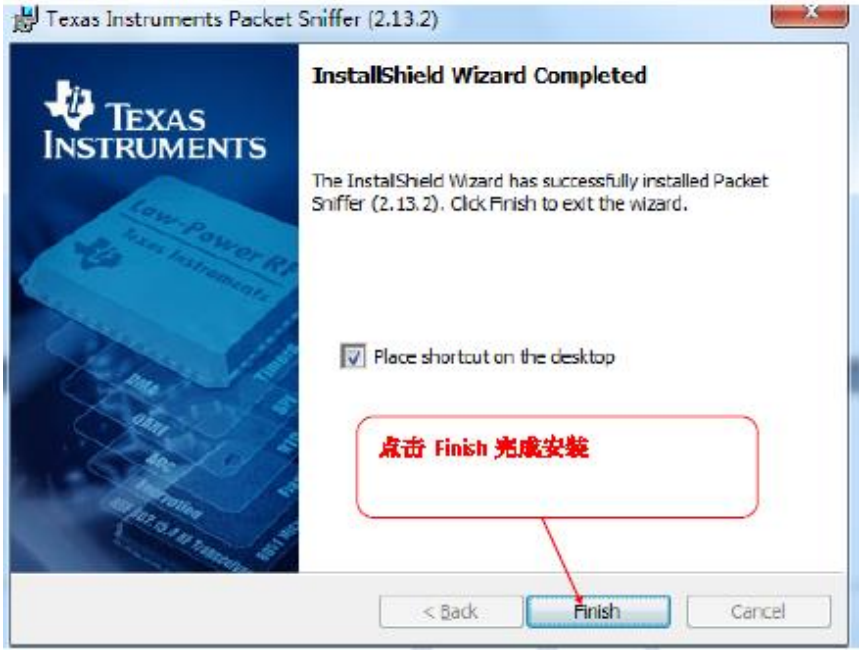




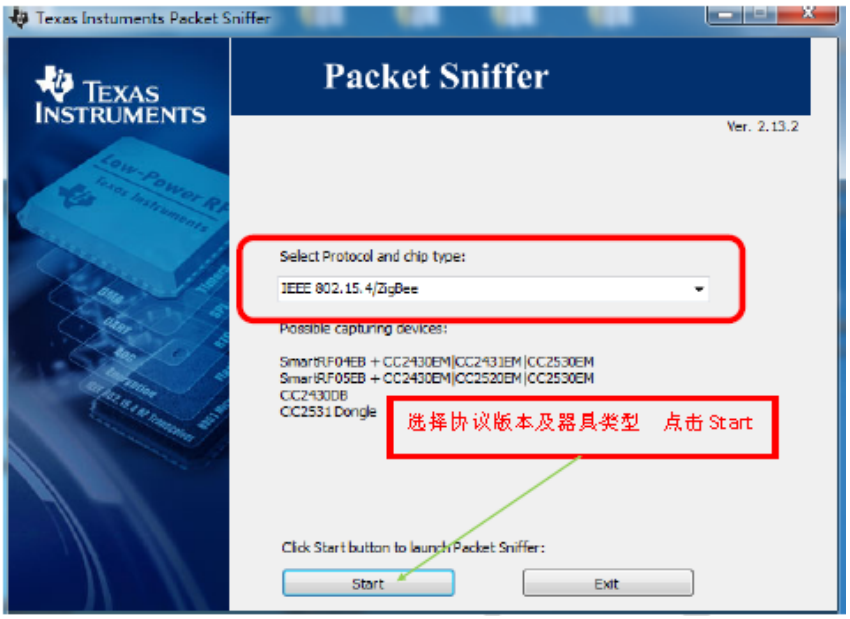
2.6 抓包软件 Packet Sniffer 的安装和使用

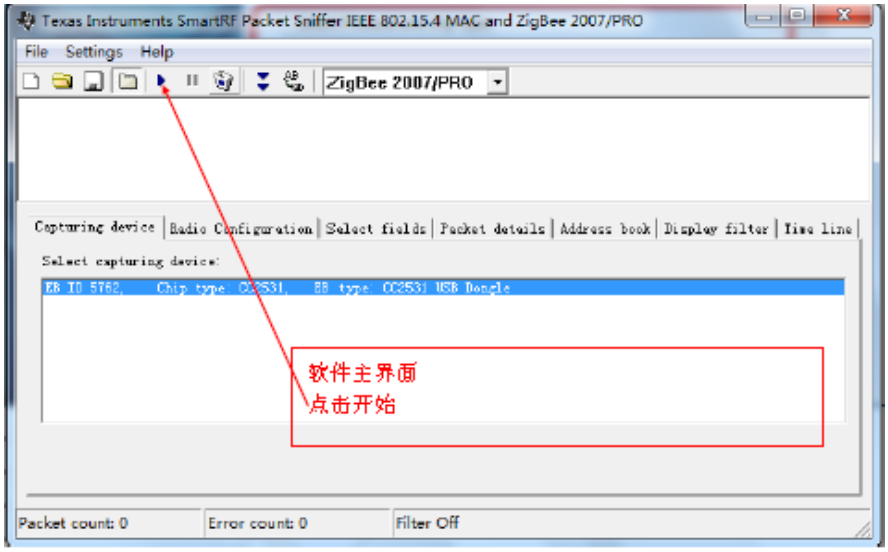
该软件配合一个仿真器和一个CC2530开发板即可以当协议栈分析仪，对通信数据进行监测。使用的时候只要仿真器一端连着CC2530开发板，另外一端通过USB线连接电脑即可，CC2530开发板无需下载代码。首先点击SmartRF™ Packet Sniffer安装文件进行安装。

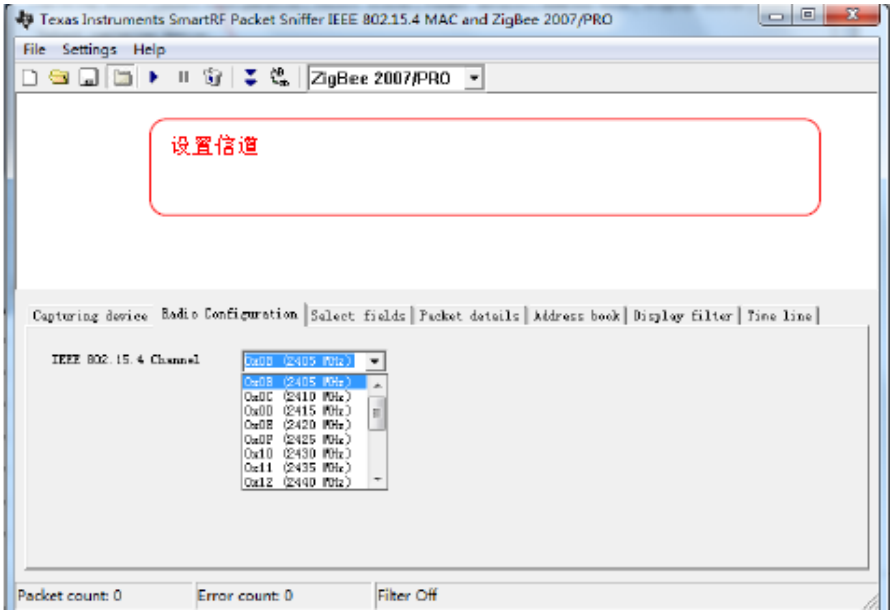
安装完成如下图所示。

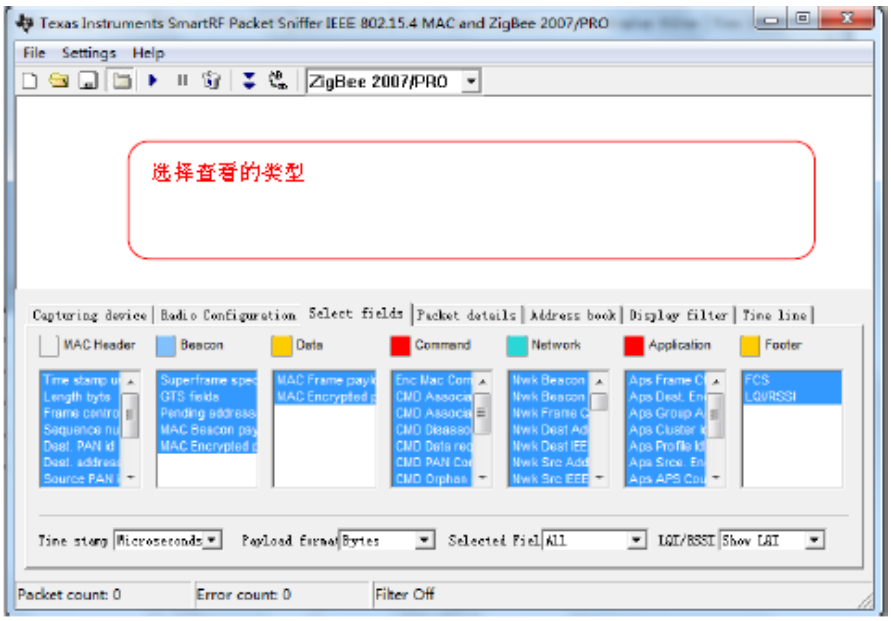

Texas Instruments Packet Sniffer (2.13.2)
InstallShield Wizard Completed
The InstallShield Wizard has successfully installed Packet Sniffer (2.13.2). Click Finish to exit the wizard.
Place shortcut on the desktop
点击 Finish 完成安装
< Back
Finish
Cancel

打开软件，界面如图所示


Texas Instruments Packet Sniffer
Packet Sniffer
Ver. 2.13.2
Select Protocol and chip type:
IEEE 802.15.4/ZigBee
Possible capturing devices:
SmartRF04EB + CC2430EM|CC2431EM|CC2530EM
SmartRF05EB + CC2430EM|CC2520EM|CC2530EM
CC2430DB
CC2531 Dongle
选择协议版本及器具类型 点击 Start
Click Start button to launch Packet Sniffer:
Start
Exit


Texas Instruments SmartRF Packet Sniffer IEEE 802.15.4 MAC and ZigBee 2007/PRO
File Settings Help
ZigBee 2007/PRO
Capturing device | Radio Configuration | Select fields | Packet details | Address book | Display filter | Time line
Select capturing device:
软件主界面
点击开始
Packet count: 0
Error count: 0
Filter Off


Texas Instruments SmartRF Packet Sniffer IEEE 802.15.4 MAC and ZigBee 2007/PRO
File Settings Help
ZigBee 2007/PRO
设置信道
Capturing device | Radio Configuration | Select fields | Packet details | Address book | Display filter | Time line
IEEE 802.15.4 Channel
Packet count: 0
Error count: 0
Filter Off


Texas Instruments SmartRF Packet Sniffer IEEE 802.15.4 MAC and ZigBee 2007/PRO
File Settings Help
ZigBee 2007/PRO
选择查看的类型
Capturing device | Radio Configuration | Select fields | Packet details | Address book | Display filter | Time line
MAC Header
Beacon
Data
Command
Network
Application
Footer
Packet count: 0
Error count: 0
Filter Off

ZigBee 数据包结构

从 PacketSniffer 软件抓到的数据包可以看到每个数据包有很多段组成，这与 Z-Stack 协议栈是对应的，由于 Z-Stack 协议栈是采用分层结构实现的，所以数据包显示时也是 不同的层使用不同的颜色，这样读者很容易查看到相应的数据。

刚接触数据包时，会有很多疑问，例如，数据包是怎么构成的，如何分析这些数据 包呢，请读者注意，这是 zigbee 网络内的数据包，因此，这些数据包肯定符合 zigbee 协议的，所以需要从 zigbee 协议各层的数据包构成入手，然后慢慢分析这些数据包的 构成。下面分析一个数据包

P.nbr.	Time (us)	Length	Frame control field					Sequence number	Dest. PAN	Dest. Address	Beacon request	LQI	FCS
			Type	Sec	Pnd	Ack.req	PAN_compr						
RX	+253187												
34	=124182952	10	CMD	0	0	0	0	0x75	0xFFFF	0xFFFF		155	OK

可以看到“Frame control field”、“Sequence number”，“Dest PAN”，“Dest Address” 等不同的是数据段。这需要从 zigbee 协议中介质访问控制层（MAC）数据包的构成讲起。 MAC 数据包结构

长度：	2	1	0/2	0/2/8	0/2	0/2/8
名称：	帧控制域	序列号	目的PAN	目的地址	源PAN	源地址

结合上述数据包的格式就很容易理解上述数据包各个段的含义。

下面结合抓到的数据包来分析网络的简历过程以及数据传输的过程，用户数据在数 据包的哪个位置。

P.nbr.	Time (us)	Length	Frame control field					Sequence number	Dest. PAN	Dest. Address	Source Address	Short addr Assoc. status	LQI	FCS
			Type	Sec	Pnd	Ack.req	PAN_compr							
RX	+656000													
38	=125463966	10	CMD	0	0	0	0	0x77	0xFFFF	0xFFFF	Beacon request		147	OK
RX	+2585													
39	=125466551	28	BCN	0	0	0	0	0xE7	0x9C72	0x0000	Superframe specification: BO 15 SO 15 F.CAP 15 BLE 0 Coord 1 Assoc 1	GTS fields: Len 0 Permit 0	Beacon payload: 00 22 84 A0 BC 59 02 00 4B 12 00 FF FF FF 00		
RX	+509981													
40	=125976532	21	CMD	0	0	1	0	0x78	0x9C72	0x0000	0xFFFF	0x00124B000259B7F2	Association request: Alt.coord 0 FFD 1 Power 1 Idle.RX 1 Sec 0 Alloc.addr 1	141	OK
RX	+1056													
41	=125977588	5	ACK	0	0	0	0	0x78					107	OK
RX	+493738													
42	=126471326	18	CMD	0	0	1	1	0x79	0x9C72	0x0000	0x00124B000259B7F2	Data request	141	OK
RX	+960													
43	=126472286	5	ACK	0	1	0	0	0x79					110	OK
RX	+1474													
44	=126473760	27	CMD	0	0	1	1	0xAE	0x9C72	0x00124B000259B7F2	0x00124B000259BCA0	Short_addr Assoc_status: 0x99A3 Successful	110	OK
RX	+1248													
45	=126475008	5	ACK	0	0	0	0	0xAE					141	OK
RX	+38113													
46	=126513121	39	DATA	0	0	0	1	0x7A	0x9C72	0xFFFF	0x99A3	MAC payload: 08 00 FD FF A3 99 1E 92 08 00 13 00 00 00 00 00 00 A3 99 F2 B7 59 02 00 4B 12 00 0E	NWK Frame control field: Type DATA Version 0x2 DR 0 MF 0 Sec 0 SR 0 DIEEE 0 SIEEE 0	NWK Dest Addre 0xFF
RX	+8264													
47	=126521385	39	DATA	0	0	0	1	0xAF	0x9C72	0xFFFF	0x0000	MAC payload: 08 00 FD FF A3 99 1D 92 08 00 13 00 00 00 00 00 00 A3 99 F2 B7 59 02 00 4B 12 00 0E	NWK Frame control field: Type DATA Version 0x2 DR 0 MF 0 Sec 0 SR 0 DIEEE 0 SIEEE 0	NWK Dest Addre 0xFF

第 1~7 行是协调器建立 zigbee 网路耦合终端节点加入该网路的过程

第 1 行，终端节点发送信标（beacon）请求。

第 2 行，协调器一斤固件里了 zigbee 无线网络，在 zigbee 无线网络中，

协调的网络 地址必定是 0x0000，第 2 行所示数据包中的“Source Address”就是协调器的网络地址。

第 3 行，终端节点发送加入网络请求（Association Request）。

第 4 行，协调器对终端节点加入网络请求作出应答，从哪里可以确定是对节点的加 入网络请求作出的应答呢，一个很明显的方法是观察序列号，第 3、4 行显示的数据包 中，“Sequence Number”是相同的，都是 0x78,。

第 5 行，终端节点收到协调器的应答后，发送数据请求（data request），请求协调

器 分 配 网 络地 址 ，从 该 数 据包 同 事可 以 得 到的 信 息是 ：终端 节 点 的 IEEE 地址是

0x00124B000259B7F2.

第 6 行，协调器对终端节点的数据请求作出应答（序列号也是相同的，0xAE）

第 7 行，协调器将分配的网络地址发送给终端节点，新分配的网络地址是 0x99A3. 从第 9 行开始，终端节点就是用自己的网路哦地址 0x99A3 与协调器通信了。 可能有的读者会有这样的疑问，节点加入网络后，分配到的网络地址，此时为什么

不适用节点的 IEEE 地址作为源地址进行通信呢？

这主要是由于 IEEE 地址是 64 位的，而节点的网络地址是 16 位，对于无线通信来说， 数据长度越长，功耗越大。

这是，按下 bb 板的 up 按键，向协调器发送闪烁 led 的命令，然后可以在抓到的数

据包中看到这样的数据段，在 APS payload 中的三个字节正是我们发送的 buffer。

S Counter	APS Payload	LQI	FCS
171	02 E8 03	52	OK

回顾一下发送的代码片段，SAMPLEAPP_FLASH_DURATION 宏定义的值为 1000ms， 16 进制为 0x03E8，代码中，将这两个字节的低 8 位赋值给了 buffer[1]，高 8 位赋值给 了 buffer[2]，然后一个全局的计数变量赋值给了 buffer[0],我是第二次 up 键后接的上 图,因此抓包得到的和我们分析的一致：0x02 0xE8 0x03

```
00341:   if ( keys & HAL_KEY_SW_1 )
00342:   {
00343:     /* This key sends the Flash Command is sent to Group 1.
00344:      * This device will not receive the Flash Command from this
00345:      * device (even if it belongs to group 1).
00346:      */
00347:     SampleApp_SendFlashMessage( SAMPLEAPP_FLASH_DURATION );
00348:   }
```

```
00438: void SampleApp_SendFlashMessage( uint16 flashTime )
00439: {
00440:   uint8 buffer[3];
00441:   buffer[0] = (uint8)(SampleAppFlashCounter++);
00442:   buffer[1] = LO_UINT16( flashTime );
00443:   buffer[2] = HI_UINT16( flashTime );
00444:
00445:   if ( AF_DataRequest( &SampleApp_Flash_DstAddr, &SampleApp_epDesc,
00446:                        SAMPLEAPP_FLASH_CLUSTERID,
00447:                        3,
00448:                        buffer,
00449:                        &SampleApp_TransID,
00450:                        AF_DISCV_ROUTE,
00451:                        AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
```