舞动的梦想

SOCZigBee

无线盛世《快速进入ZB世界》

Ver:1

进入Zigbee世界的准备工作

- 首先，我们需具备一些硬件设备及平台。以下我就罗列一下Zigbee开发基本工具：
- 计算机：不管是设计电路还是编程开发都是离不开它的。
- Zigbee开发板：对于初学者来说，Zigbee开发板无疑是最佳选择。有了开发板，你可以在我们成熟设计的基础上学习或者做自己的设计。
- Zigbee模块：集MCU，RF，天线设计于一体的Zigbee模块。使用它，我们可省去设计天线及IC周边电路设计的复杂工作。

进入Zigbee世界的准备工作

- Zigbee仿真器：是集烧写程序、在线编程和在线仿真功能于一身的开发过程工作中必不可少的开发工具。编程器既能对CC243x芯片（其实包括TI产品中的CC系列的大部分芯片）进行烧写程序（hex标准文件程序），也能对CC243x芯片进行在线编程和仿真，让我们能方便地在线调试开发，从而大大地提高了开发效率。
- Zigbee协议分析仪：ZigBee的设计开发者必不可少的工具！ZigBee协议分析仪具有广泛的功能，包括：分析以及解码在PHY、MAC、NETWORK/SECURITY、APPLICATION FRAMEWORK、和APPLICATION PROFICES等各层协议上的信息包；显示出错的包以及接入错误；指示触发包；在接收和登记过程中可连续显示包。

进入Zigbee世界的准备工作

- 再次,我们需要在将用于开发Zigbee的计算机平台上安装这些软件：
- Zigbee协议分析软件（sniffer）
- 程序烧写软件（Flash Programmer）
- IAR公司的EW8051 version 7.20I/W32
  。

- Zigbee协议分析软件安装
  详见《Zigbee协议分析仪使用手册》文档的安装Zigbee协议分析软件章节。
- 程序烧写软件（Flash Programmer）安装详见《Flash Programmer的安装.doc》文档。
- EW8051 version 7.20I/W32的安装详见《EW8051 version 7.20I_W32的安装.doc》文档。

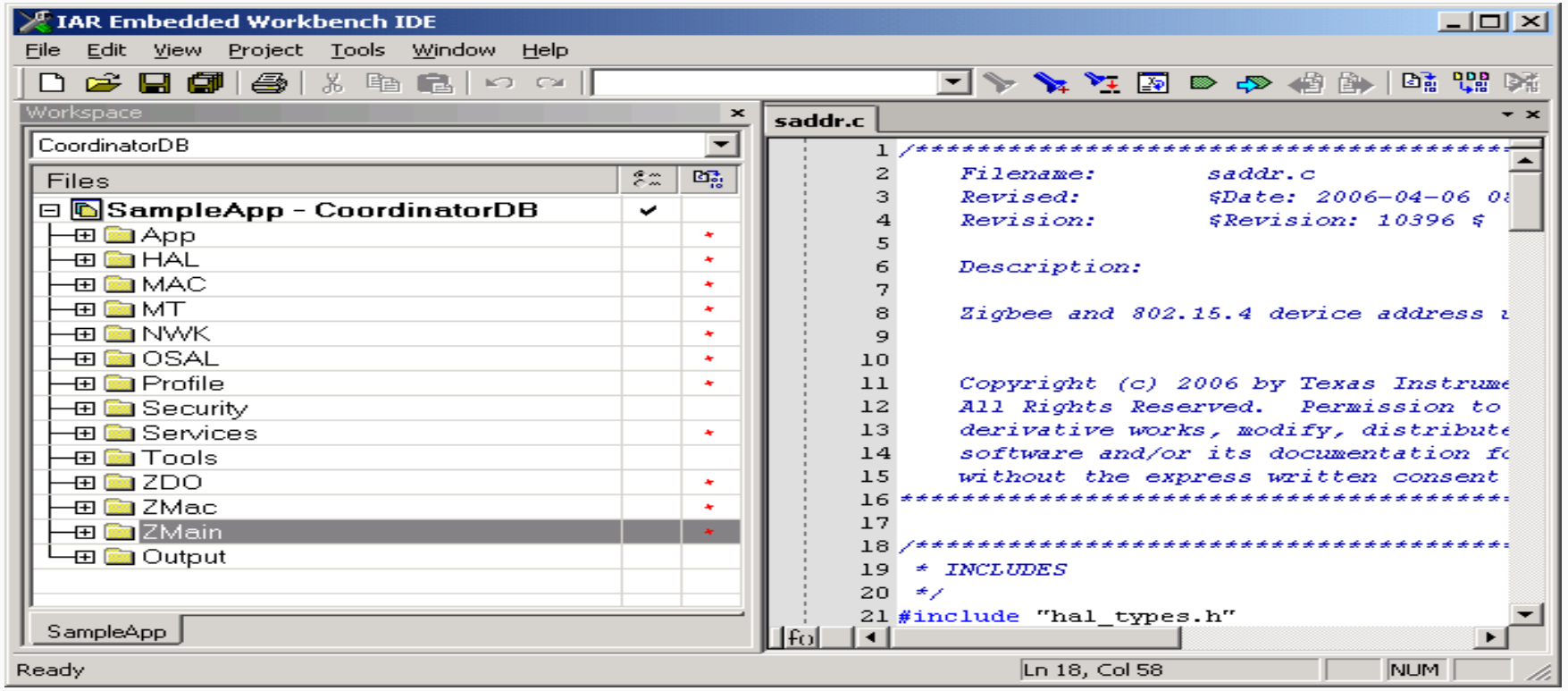
进入Zigbee世界的准备工作

- 接下来,我们就来了解一下TI的ZStack吧.
- 首先,我们可以在TI网站(http://www.ti.com)下载协议栈.
- 接着,我们就可以在PC上安装好TI的ZStack,详细安装过程可以参考详见无线盛世《TI ZStack安装.doc》文档。

TI ZStack框架介绍

SOCZigBee

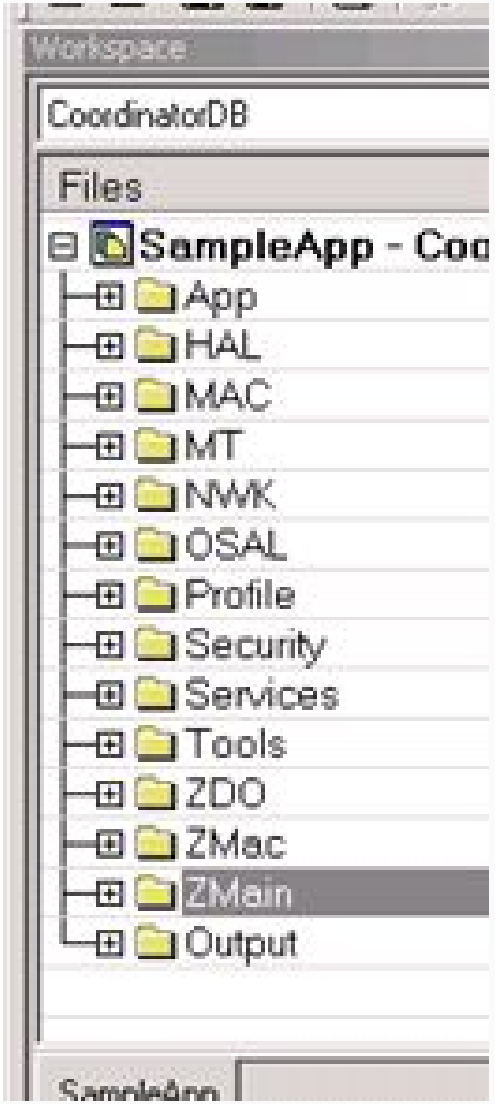
打开TI ZStack附带的简单实例SampleApp.
可以看到TI ZStack的大体框架,如下图所示:



TI ZStack框架介绍

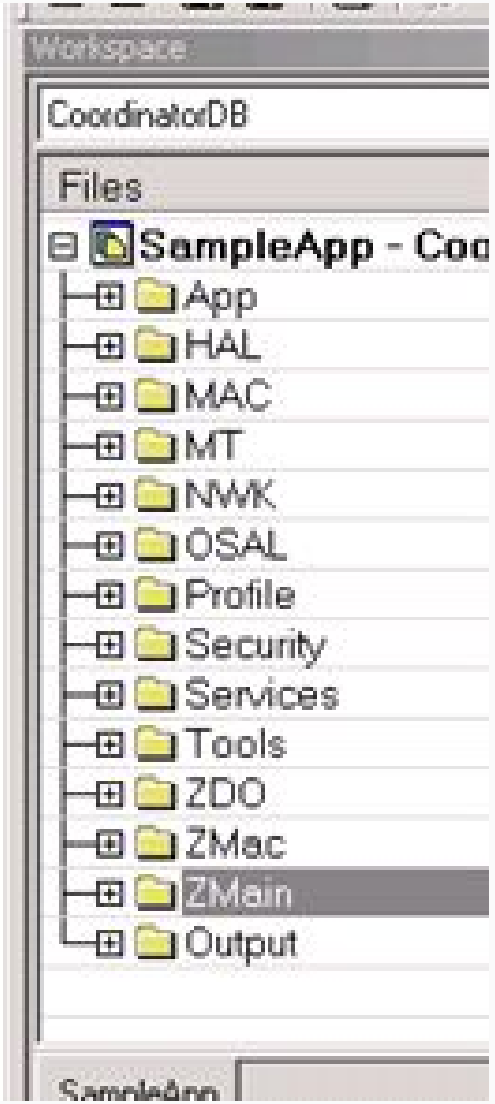
SOCZigBee

- App：应用层目录，这也是用户创建各种不同工程的区域；
- HAL：硬件层目录，包括着与硬件相关的配置及操作函数；
- MAC：MAC层目录，包括着MAC层配置参数文件及MAC LIB库的函数接口文件；
- MT：包括基于AF层的调试函数文件，主要包括串口等通信函数；
- NWK：网络层目录，包括着网络层配置参数文件及MAC LIB库的函数接口文件；
- OSAL：系统目录，包括协议栈系统文档；



TI ZStack框架介绍

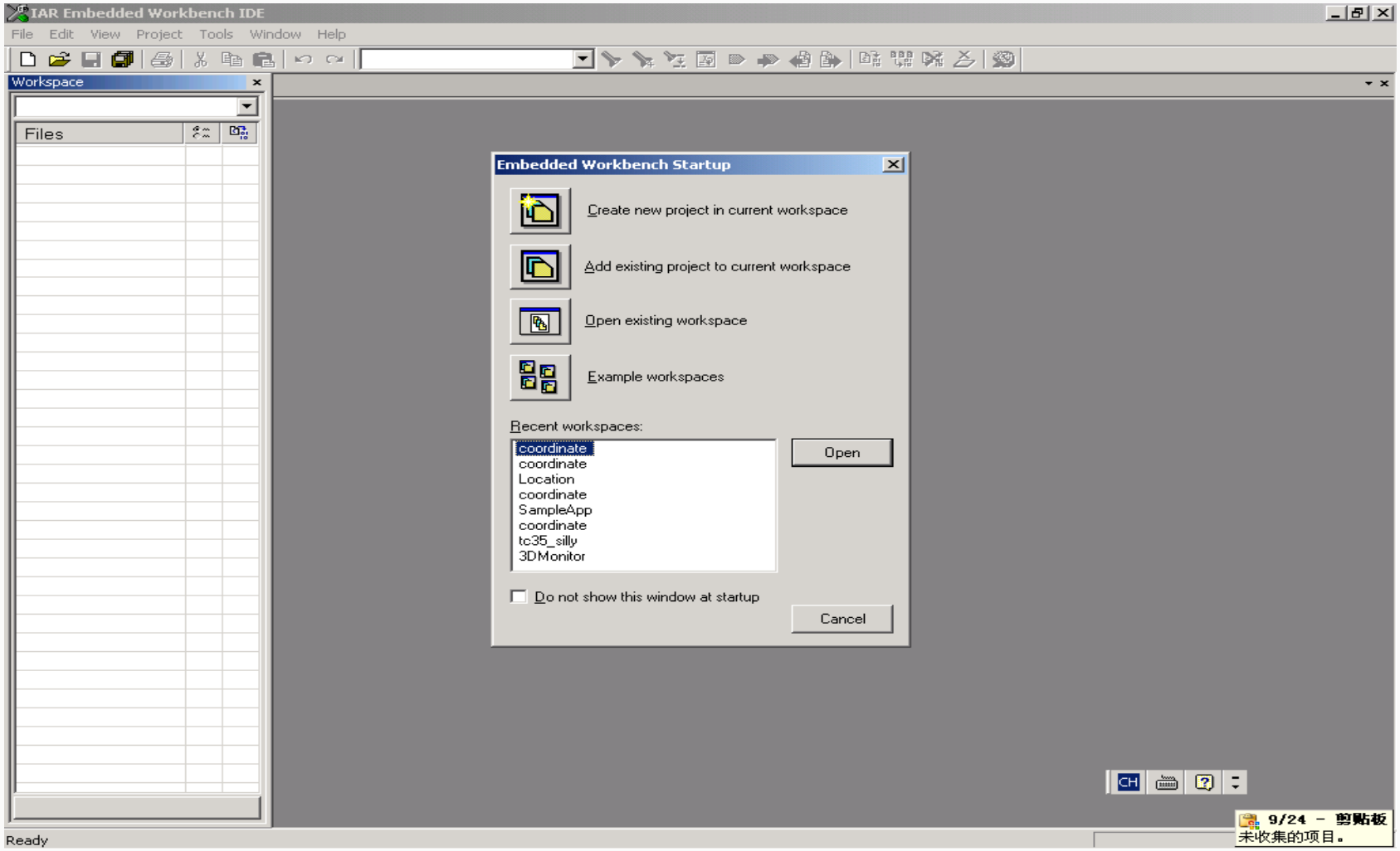
- Profile：AF层目录，包括AF层处理函数文件；
- Security：安全层目录，安全层处理函数，比如加密函数等;
- Services：地址处理函数目录，包括着地址模式的定义及地址处理函数文档；Tools：工程配置目录，包括协议栈等配置文档；
- ZDO：ZDO层目录，包括ZDO层处理函数文档；
- ZMac：MAC层目录，包括MAC层参数配置及MAC层LIB库函数回调处理函数
- ZMain：主函数目录，包括入口函数及硬件配置文件；
- Output：输出文件目录，这是EW8051 IDE自动生成的；



TI ZStack的编译与烧写

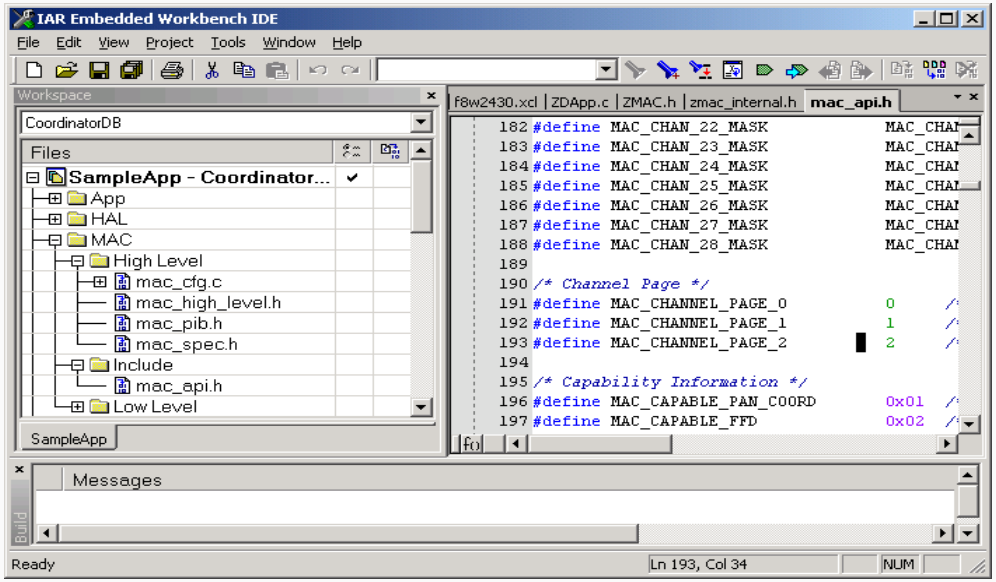
SOCZigBee

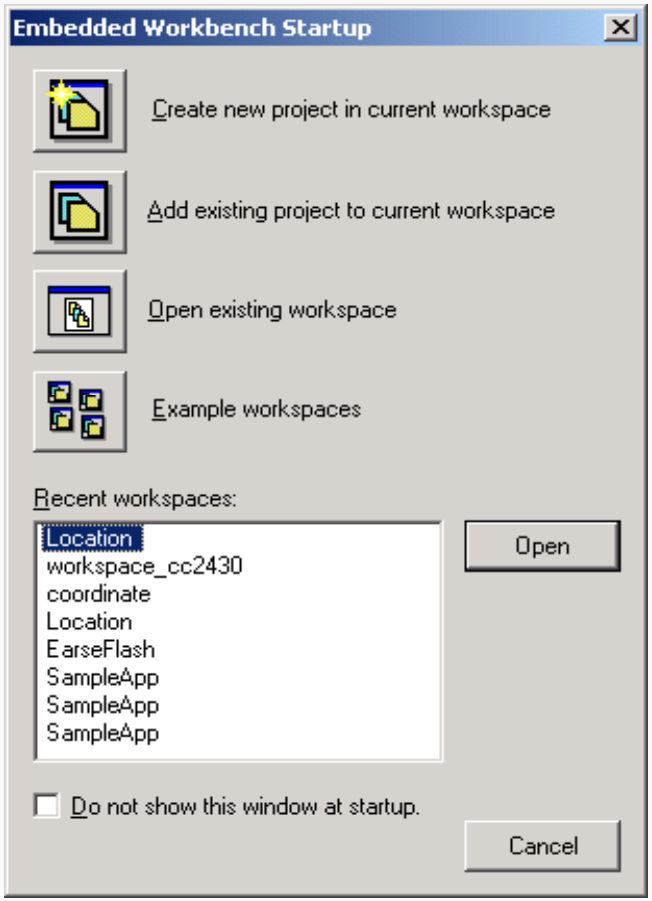
- 打开EW8051 IDE:



TI ZStack的编译与烧写

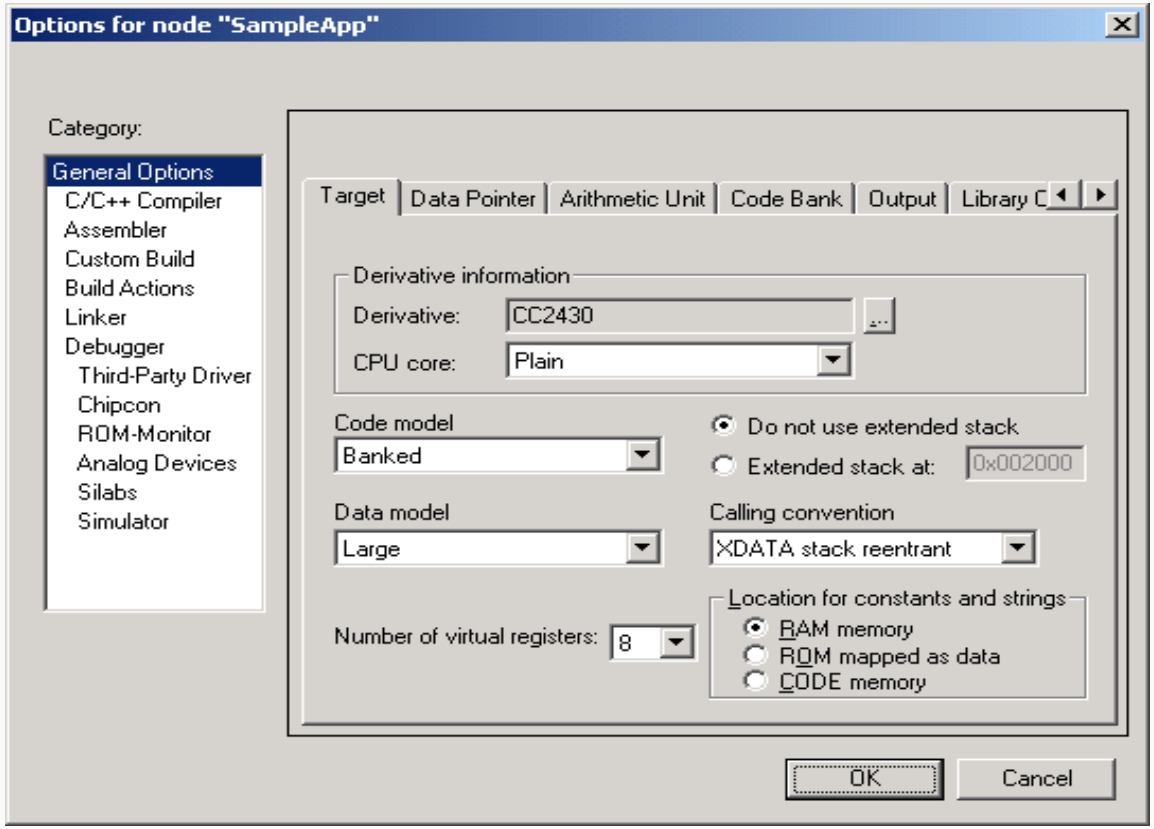
- 点击跳出对话框(如右图)中的“**Open existing workspace**”按钮.打开TI ZStack附带的实例SampleApp

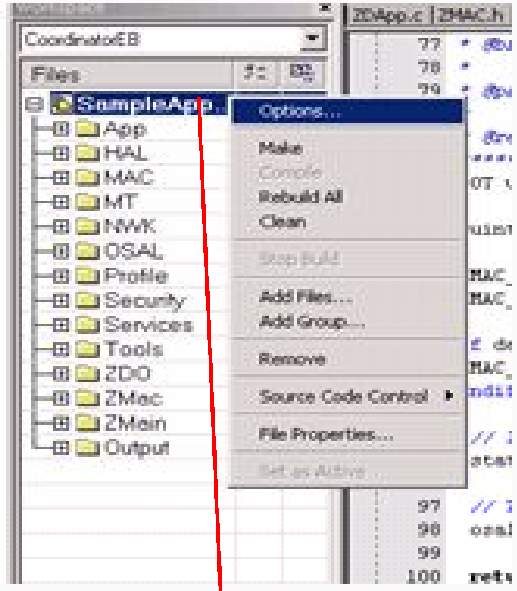




TI ZStack的编译与烧写

- 我们可以选择菜单Project->Options、右击菜单options或者通过热键（ALT+F7）打开工程属性设置。





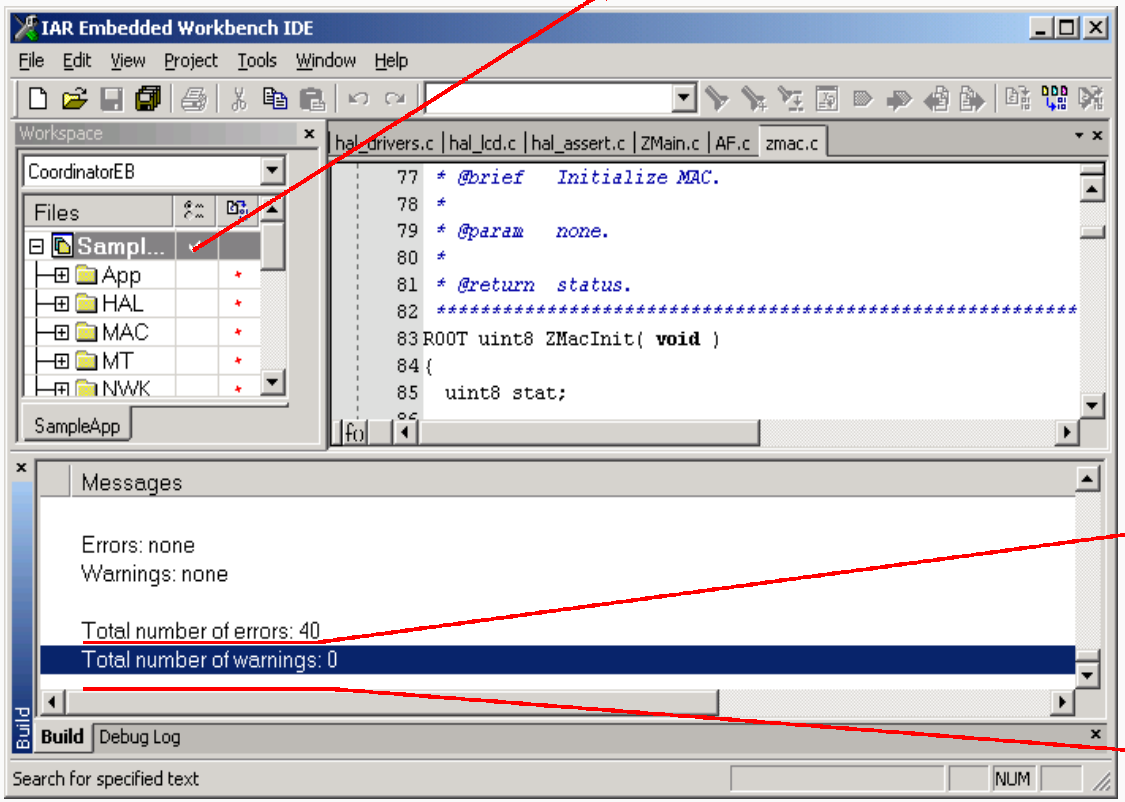
右击

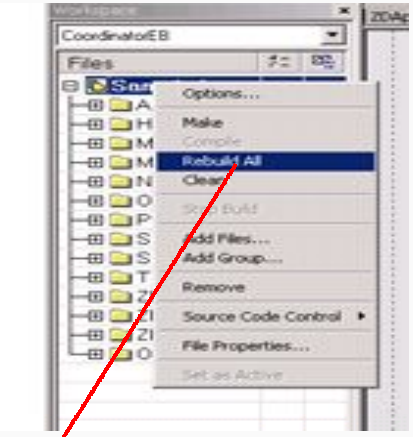
TI ZStack的编译与烧写

SOCZigBee

- 选择Project->Rebuild all或者工程右击菜单中的Rebuild all重新编译所有文件。

右击此处





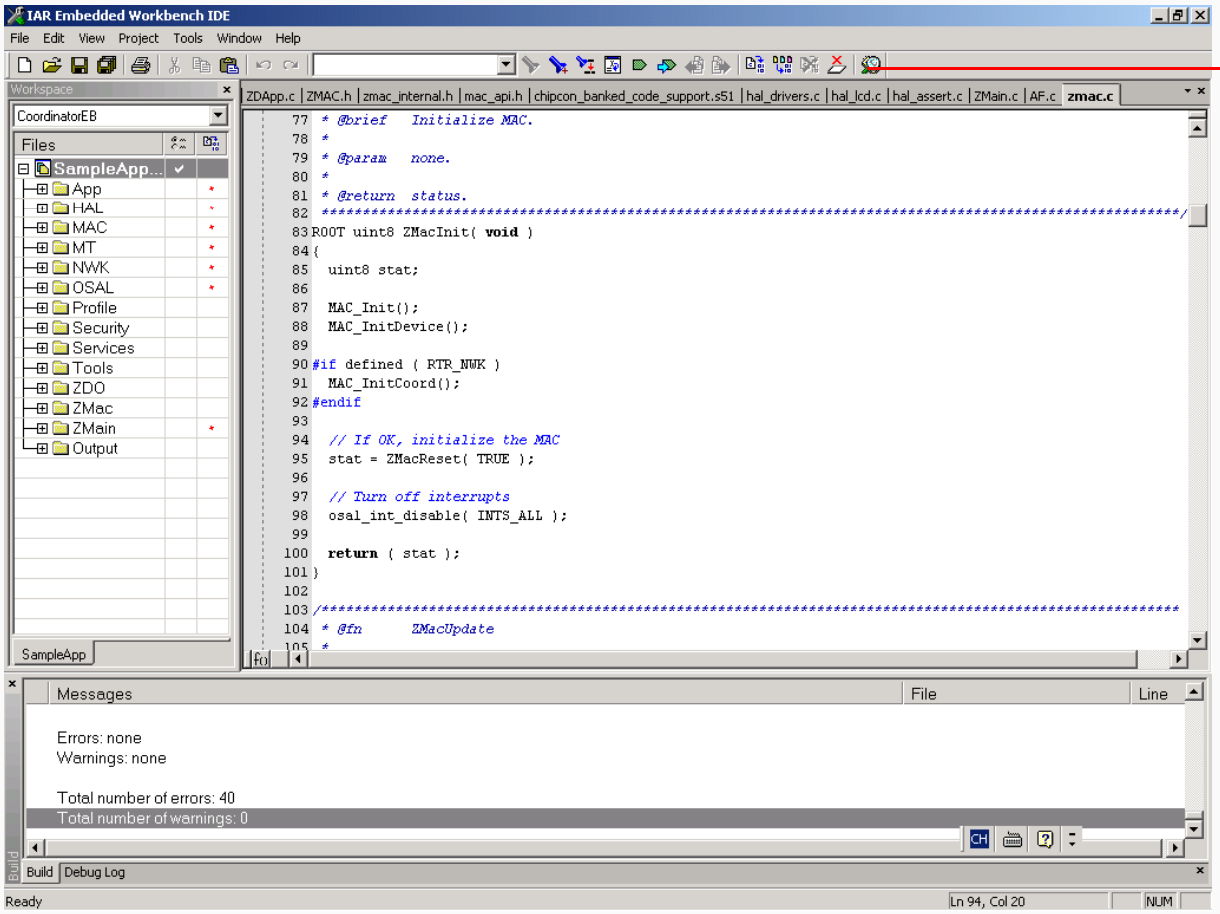
单击此处进行编译

编译错误总数，数量为0时才能完成编译操作。

编译警告总数，警告一般是指可能存在错误或者使用非常规语句。就算有警告也能通过编译

TI ZStack的编译与烧写

- 选择Project->Debug或者热键 （Ctrl+D ）给开发板上的Zigbee模块下载程序

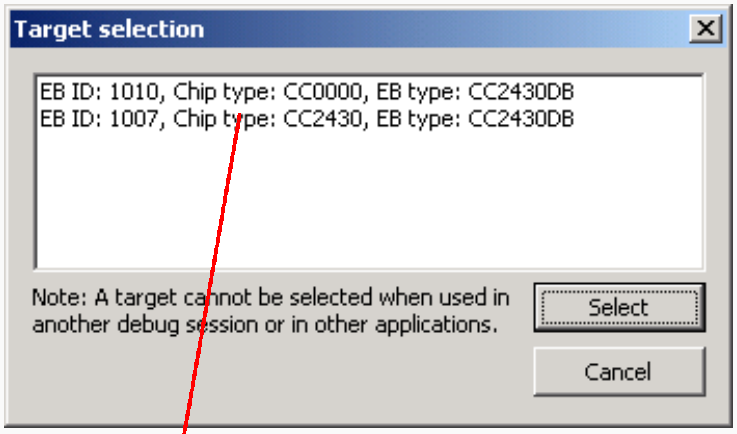


也可以点击此按钮进行烧写

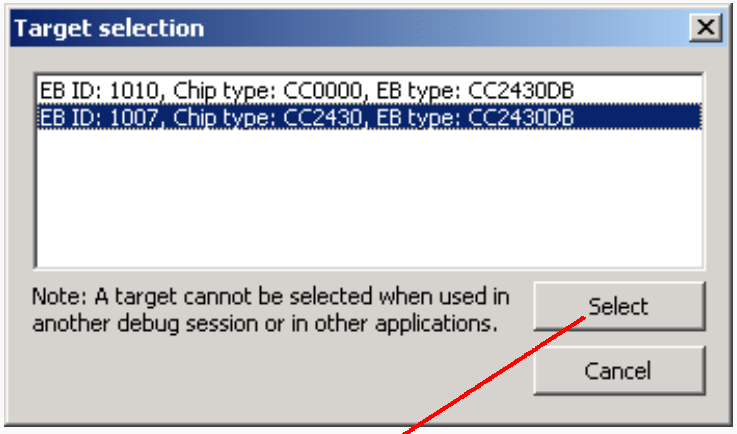
TI ZStack的编译与烧写

SOCZigBee

- 执行Zigbee模块烧写程序时，如果计算机连接了多个Zigbee USB工具则会出现如下对话框，让我们选择目标板Zigbee编程器编号。我们要选择与目标板（装有Zigbee模块的开发板或者我们生产的产品）相连接的Zigbee编程器。



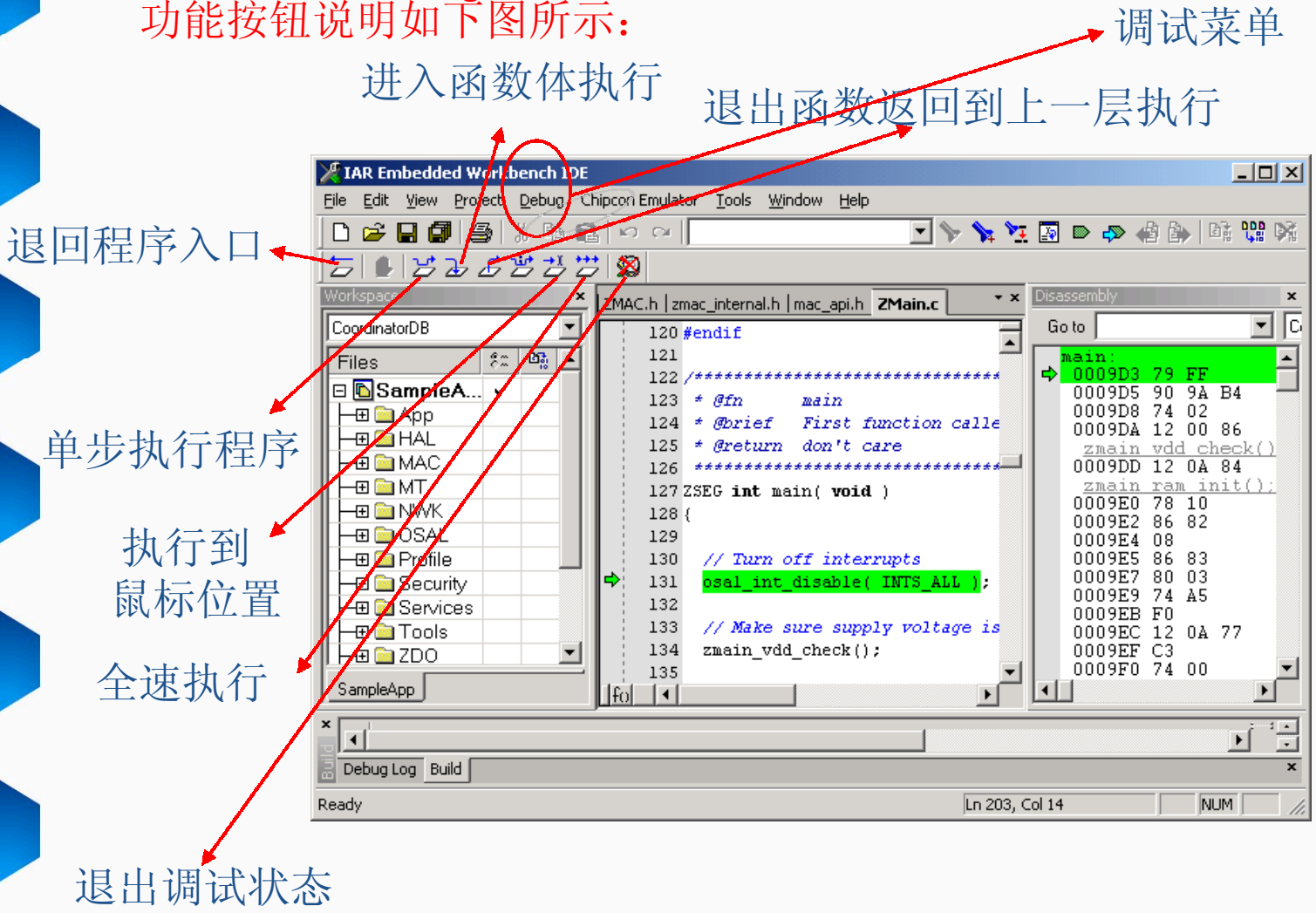
点击目标编程器或双击选中



选择按钮

TI ZStack的编译与烧写

- 如果选择了Debug模式，我们要实现在线调试程序功能，编译时EW8051功能按钮说明如下图所示：



工程选项设置注意事项

SOCZigBee

- Debug模式和Release模式选择。

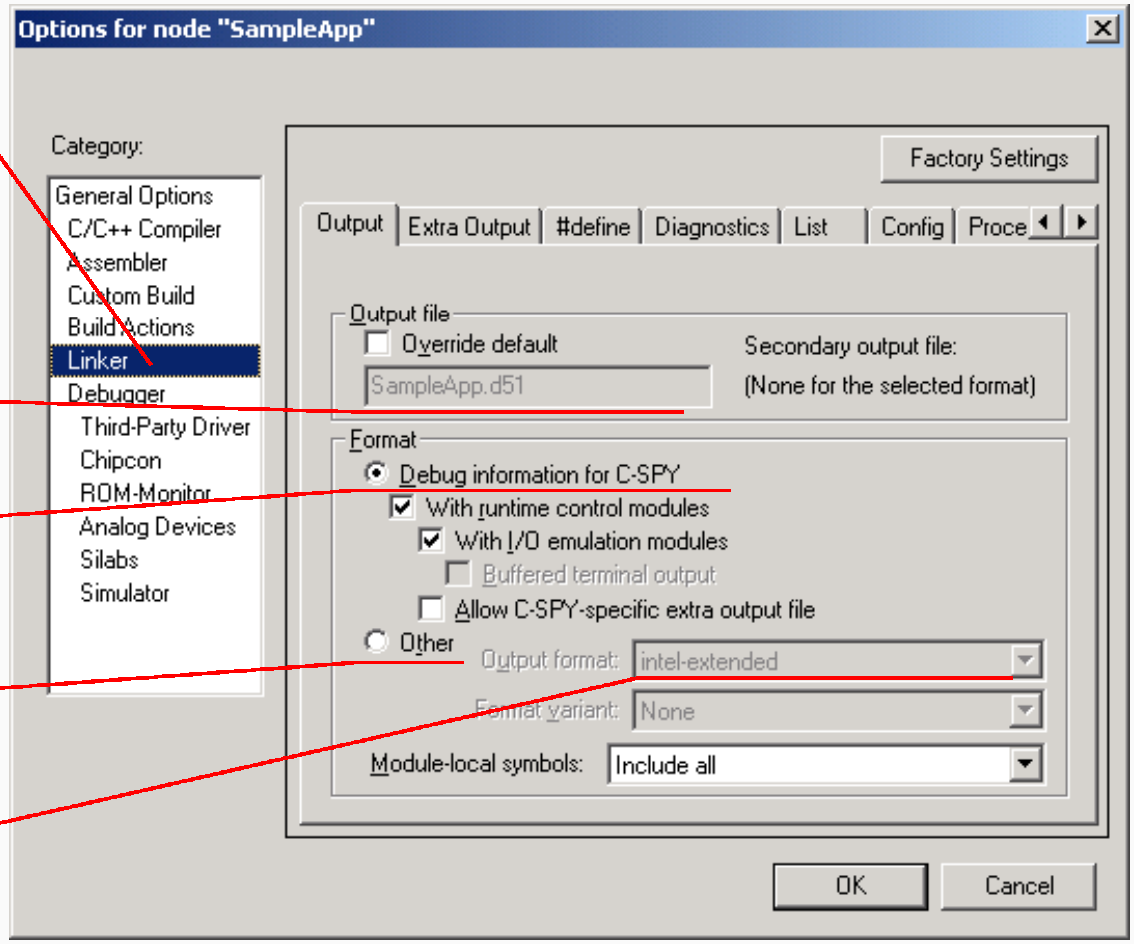
点击此选项

更改默认输出文件名
建议扩展名改为hex

选择Debug模式

选择Release模式

选择输出文件格式
一般选择intel ext模式



工程选项设置注意事项

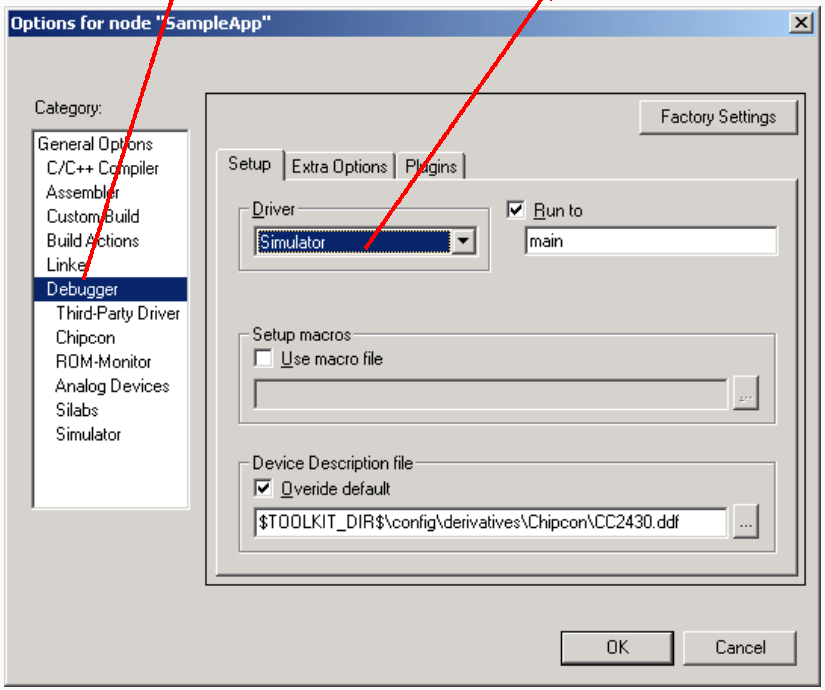
SOCZigBee

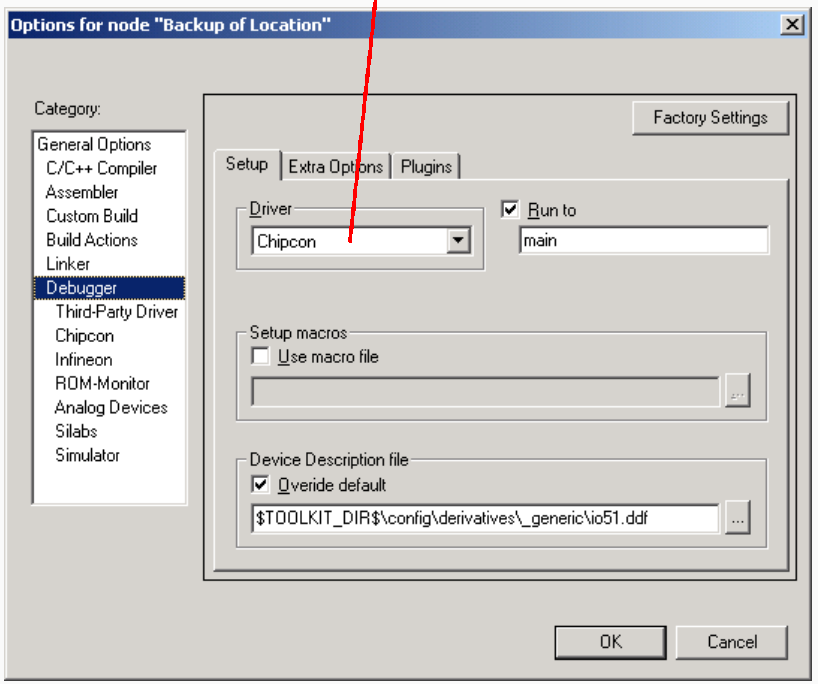
- 如果选择了Debug模式，我们还要记得选择硬件仿真，当然我们也可以使用软件仿真。

选中此选项

选择软件仿真

选择硬件仿真，需连上硬件

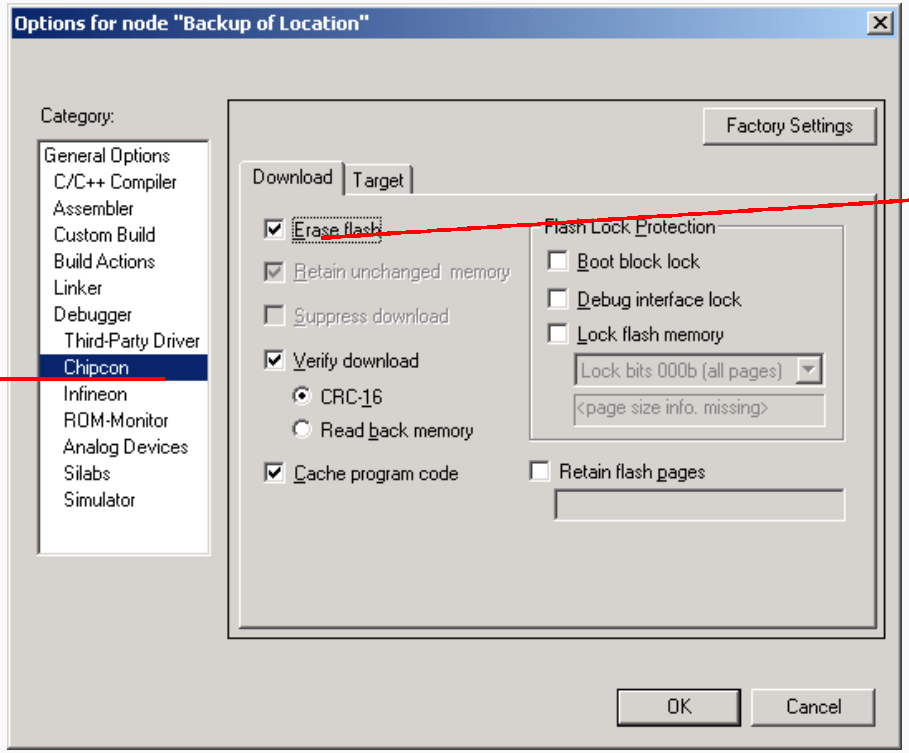




工程选项设置注意事项

SOCZigBee

- Debug模式下，如果选择了硬件在线仿真的方式调试软件的话，第一次下载软件时我们还要记得擦除Flash。但烧写程序的时候就会把Flash中的所有内容都擦除掉，我们在擦除Flash后就一定要记得重新给Zigbee模块写入合法IEEE地址（FFFFFFFF FFFFFFFF不合法）。



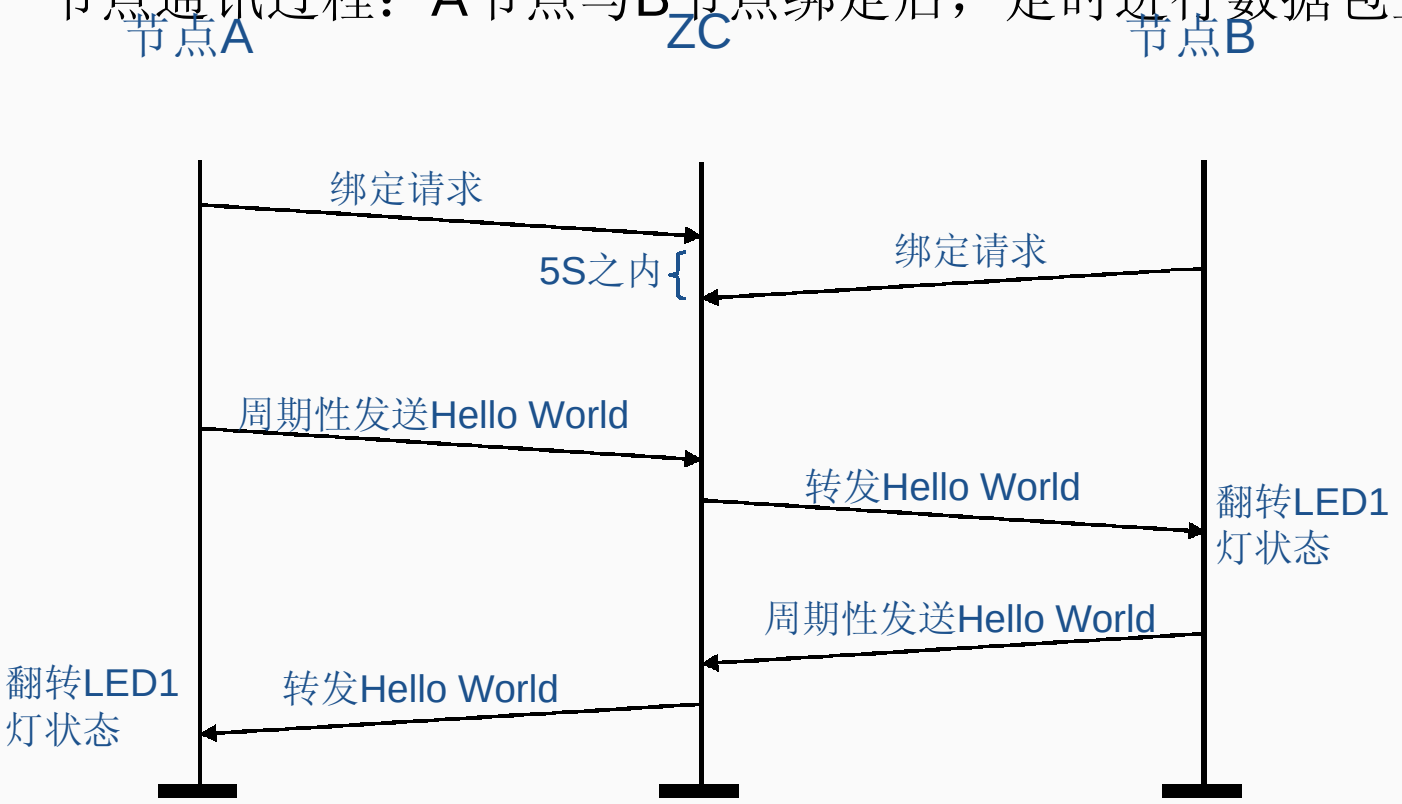
选中擦除Flash

点击选中此选项

应用实例-GenericApp

功能:实现了以5秒为周期与绑定节点互发“Hello World”数据包。

节点通讯过程：A节点与B节点绑定后，定时进行数据包互发。



开发板上资源使用

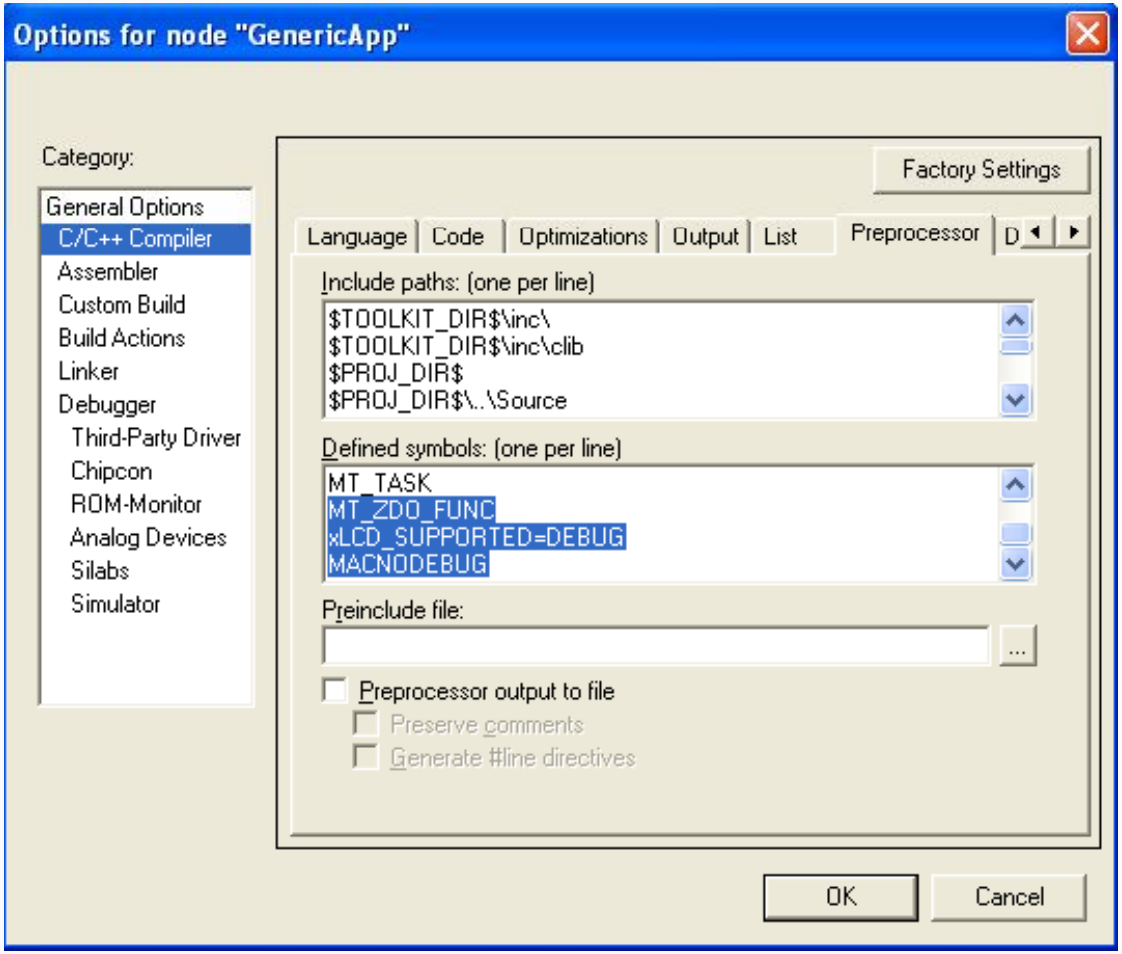
板上资源	ZStack中对应名称	功能描述
S1	Shift	按键：发送绑定请求。
LED1	HAL_LED_1	灯：入网灯亮，绑定解绑定灯状态先暗后亮，收到数据包灯状态翻转。
J3	SERIAL_PORT1	串口：PC机串口连接ZC，得出网络拓扑图。

程序改动-配置相关宏

文件目录：ZStack-1.4.2-1.1.0\Projects\zstack\Samples\GenericApp\CC2430DB\GenericApp.eww

配置相关宏：

Option->C/C++ Compiler->Preprocessor->Defined symbols中添加：



程序改动-按键处理函数修改

SOCZigBee

按键处理函数修改:

```
void GenericApp_HandleKeys( byte shift, byte keys )
{
  // Shift is used to make each button/switch dual purpose.
  if ( shift )
  {
    /*if ( keys & HAL_KEY_SW_1 )
    {
    }
    if ( keys & HAL_KEY_SW_2 )
    {
    }
    if ( keys & HAL_KEY_SW_3 )
    {
    }
    if ( keys & HAL_KEY_SW_4 )
    {
    }*/
    ZDApp_SendEndDeviceBindReq( GenericApp_epDesc.endPoint );
  }
  else
  {
    if ( keys & HAL_KEY_SW_1 )
```

程序改动-事件处理函数修改

SOCZigBee

应用事件处理函数GenericApp_ProcessEvent()
针对ZDO_STATE_CHANGE消息的修改:

ZMain | GenericApp | GenericApp * | hal_led

```
case ZDO_STATE_CHANGE:
  GenericApp_NwkState = (devStates_t)(MSGpkt->hdr.status);
  if ( (GenericApp_NwkState == DEV_ZB_COORD)
      || (GenericApp_NwkState == DEV_ROUTER)
      || (GenericApp_NwkState == DEV_END_DEVICE) )
  {
    HalLedSet(HAL_LED_4,HAL_LED_MODE_ON);//点亮LED1,以告知用户其正确入网
    // Start sending "the" message in a regular interval.

    osal_start_timer( GENERICAPP_SEND_MSG_EVT,
                      GENERICAPP_SEND_MSG_TIMEOUT );
  }
  break;
```

程序改动-MSG数据包处理函数修改

MSG信息包处理函数GenericApp_MessageMSGCB()修改,实现接收到“Hello world”数据包时灯状态翻转的功能。

```
void GenericApp_MessageMSGCB( afIncomingMSGPacket_t *pkt )
{
  switch ( pkt->clusterId )
  {
    case GENERICAPP_CLUSTERID:
      // "the" message
#if defined( LCD_SUPPORTED )
      HalLcdWriteScreen( (char*)pkt->cmd.Data, "rcvd" );
#elif defined( WIN32 )
      WPRINTSTR( pkt->cmd.Data );
#endif
      HalLedSet(HAL_LED_4,HAL_LED_MODE_TOGGLE);//使LED1(D2,D4)翻转
      break;
  }
}
```

组网-两节点通讯1

1、功能：

ZC建网，一节点ZR或ZE入网，两节点绑定互发“Hello World”。

2、现象：

节点成为网络中成员，LED1亮

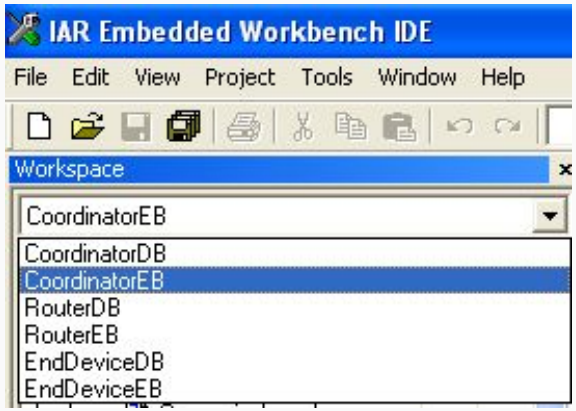
功能绑定：LED1先灭后亮

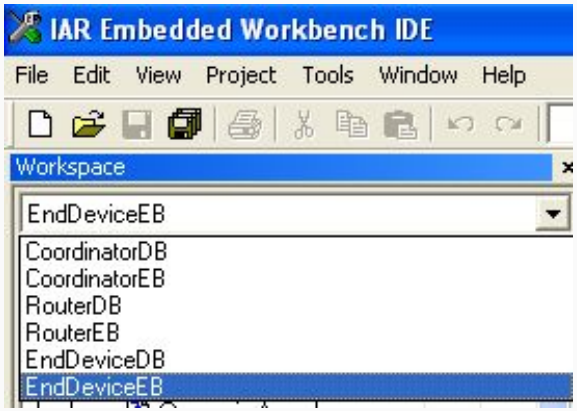
节点收到“Hello World”数据包，LED1状态翻转

3、操作例子：

选中CoordinatorEB编译并下载一ZC节点

选中EndDeviceEB编译并下载一ZE节点



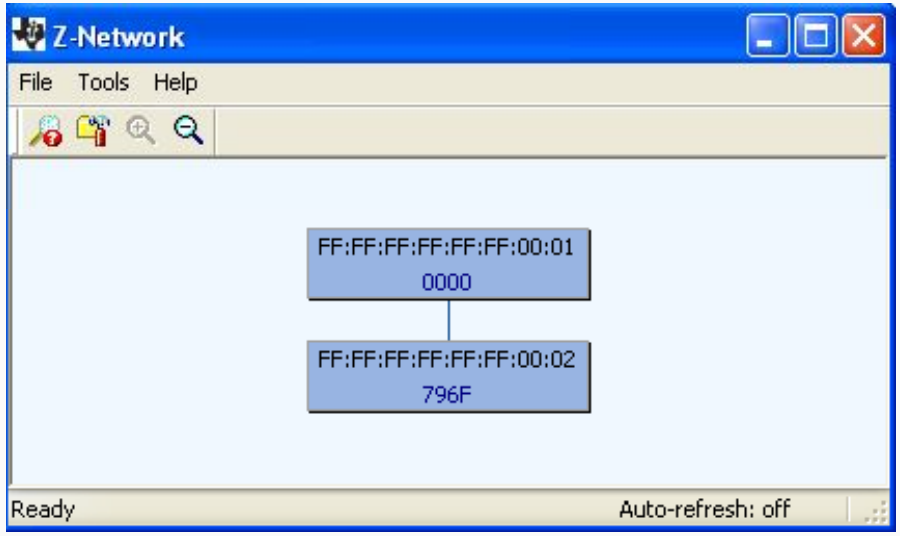


组网-两节点通讯2

将ZC节点的串口与PC串口相联，并打开ZNetwork软件

ZC上电，ZE上电

使用ZNetwork软件即可得到相应拓扑图



通过两节点绑定按钮S1实现事务绑定

绑定后两节点灯状态周期性翻转

组网-星状网络通讯1

1、功能：

ZC建网，m个ZR及n个ZE入网，网络结构为星状，ZC处于星状中心节点。节点绑定互发“Hello World”。

2、现象：

节点成为网络中成员，LED1亮

功能绑定：LED1先灭后亮

节点收到“Hello World”数据包，LED1状态翻转

3、操作例子：

设置相关网络配置参数：

网络拓扑三参数：网络深度、父节点最大子节点数及最大路由子节点数。

#define MAX_NODE_DEPTH 1 //网络深度为1

byte CskipRtrs[MAX_NODE_DEPTH+1]={3,0};允许ZC最多含3个ZR子节点

Byte CskipChldrn[MAX_NODE_DEPTH+1]={100,0};//允许ZC最多含100个子节点

路由方式： #define NWK_MODE NWK_MODE_STAR //星状路由

GenericApp.map | nwk_globals * | **nwk_globals *** | ZGlobals | ZComDef | f8w2

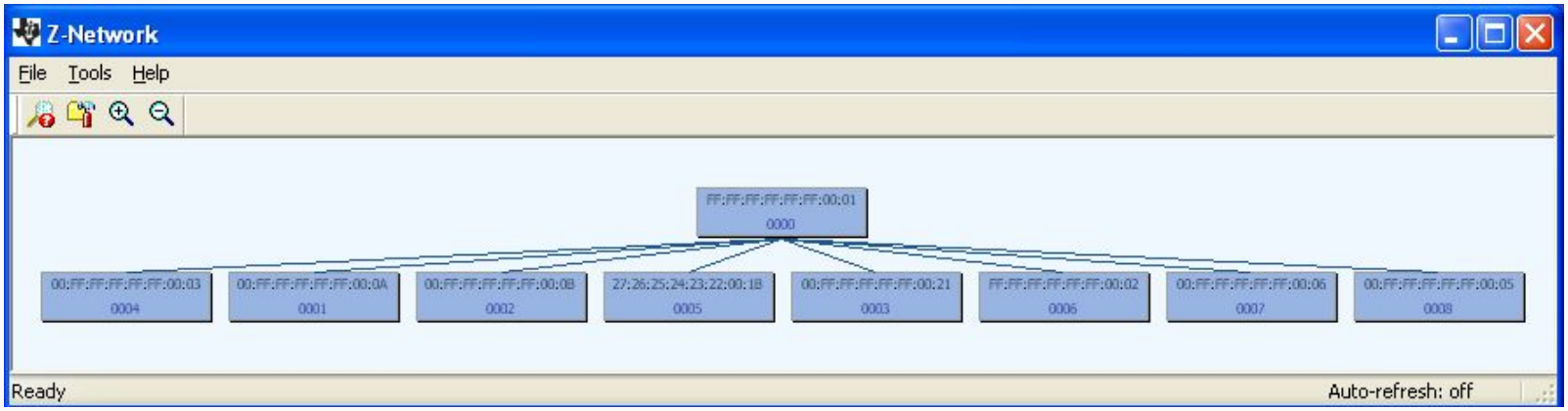
```
#define HOME_CONTROLS           1
#define BUILDING_AUTOMATION     2
#define GENERIC_STAR            3
#define GENERIC_TREE            4

#define STACK_PROFILE_ID        GENERIC_STAR

#if ( STACK_PROFILE_ID == HOME_CONTROLS )
    #define MAX_NODE_DEPTH      5
    #define NWK_MODE            NWK_MODE_MESH
    #define SECURITY_MODE       SECURITY_RESIDENTIAL
#if   ( SECURE != 0  )
    #define USE_NWK_SECURITY    1   // true or false
    #define SECURITY_LEVEL      5
#else
    #define USE_NWK_SECURITY    0   // true or false
    #define SECURITY_LEVEL      0
#endif

#elif ( STACK_PROFILE_ID == GENERIC_STAR )
    //#define MAX_NODE_DEPTH      5
    #define MAX_NODE_DEPTH      1
    #define NWK_MODE            NWK_MODE_STAR
    #define SECURITY_MODE       SECURITY_RESIDENTIAL
```

GenericApp.map | **nwk_globals** | nwk_globals | ZGlobals | ZComDef | f8w2430.xcl | f8wCc

```
  // change this if using a different stack profile...
  // Cskip array
  uint16 *Cskip;

  #if ( STACK_PROFILE_ID == HOME_CONTROLS )
    byte CskipRtrs[MAX_NODE_DEPTH+1] = {6,6,6,6,6,0};
    byte CskipChldrn[MAX_NODE_DEPTH+1] = {20,20,20,20,20,0};
  #elif ( STACK_PROFILE_ID == GENERIC_STAR )
    //byte CskipRtrs[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
    //byte CskipChldrn[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
    byte CskipRtrs[MAX_NODE_DEPTH+1] = {3,0};
    byte CskipChldrn[MAX_NODE_DEPTH+1] = {100,0};
  #elif ( STACK_PROFILE_ID == NETWORK_SPECIFIC )
    byte CskipRtrs[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
    byte CskipChldrn[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
  #endif // STACK_PROFILE_ID
#endif  // RTR_NWK
```

组网-星状网络通讯2

将ZC节点的串口与PC串口相联，并打开ZNetwork软件
ZC上电，其他ZR，ZE节点上电
使用ZNetwork软件即可得到相应拓扑图



使用节点按钮S1进行两点绑定或多点绑定，多点绑定即对同一节点多次与其他不同节点进行绑定。绑定后的节点灯状态周期性翻转

组网-网状网络通讯1

1、功能：

ZC建网，m个ZR及n个ZE入网，网络结构为树状。节点绑定互发“Hello World”。树状网络和网状网络区别在于其网络模式分别是树状路由和网状路由。现在网状网络通讯为例。

2、现象：

节点成为网络中成员，LED1亮

功能绑定：LED1先灭后亮

节点收到“Hello World”数据包，LED1状态翻转

3、操作例子：

设置相关网络配置参数：

网络拓扑三参数：网络深度、父节点最大子节点数及最大路由子节点数。

```
#define MAX_NODE_DEPTH 5 //网络深度为
byte CskipRtrs[MAX_NODE_DEPTH+1]={3,3,3,3,3,0};//最大路由子节点数为2
Byte CskipChldrn[MAX_NODE_DEPTH+1]={20,20,20,20,20,0};//最大子节点数为20
```

路由方式： #define NWK_MODE NWK_MODE_MESH //网状路由

GenericApp.map | nwk_globals | **nwk_globals *** | ZGlobals | ZComDef | f8w2430.

```
#define GENERIC_TREE          4

#define STACK_PROFILE_ID      HOME_CONTROLS

#if ( STACK_PROFILE_ID == HOME_CONTROLS )
    #define MAX_NODE_DEPTH      5
    #define NWK_MODE            NWK_MODE_MESH
    #define SECURITY_MODE       SECURITY_RESIDENTIAL
#if   ( SECURE != 0  )
    #define USE_NWK_SECURITY    1   // true or false
    #define SECURITY_LEVEL      5
#else
    #define USE_NWK_SECURITY    0   // true or false
    #define SECURITY_LEVEL      0
#endif
```

GenericApp.map | **nwk_globals *** | nwk_globals | ZGlobals | ZComDef | f8w2430.xcl | f8wC

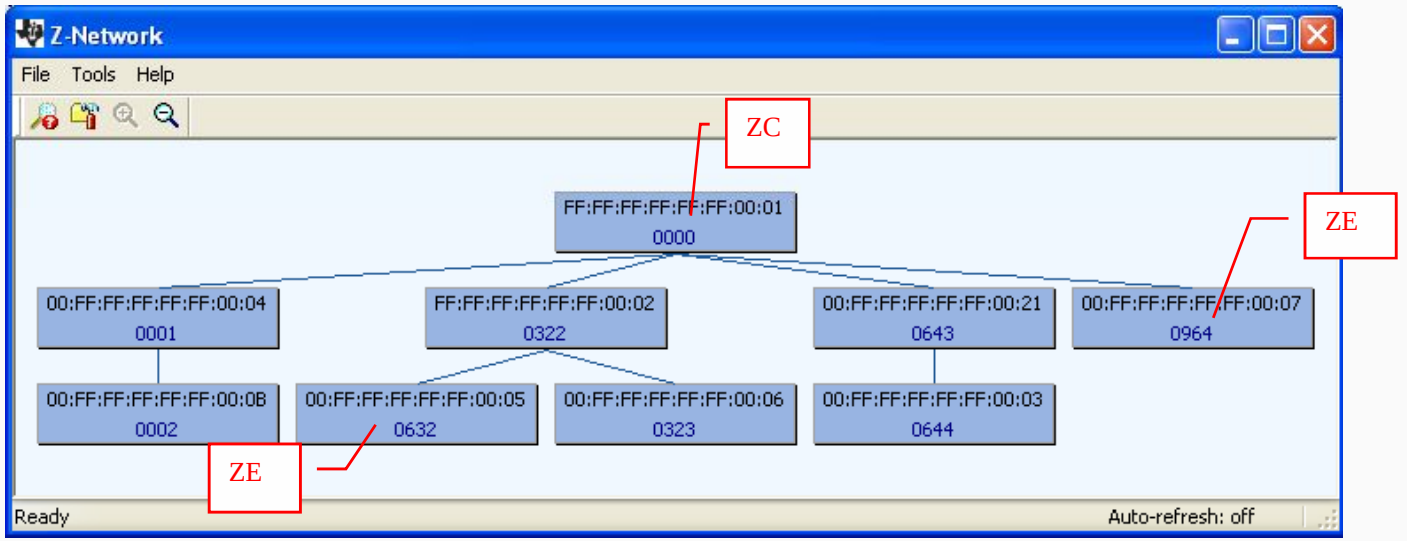
```
  uint16 *Cskip;

  #if ( STACK_PROFILE_ID == HOME_CONTROLS )
    //byte CskipRtrs[MAX_NODE_DEPTH+1] = {6,6,6,6,6,0};
    byte CskipRtrs[MAX_NODE_DEPTH+1] = {3,3,3,3,3,0};
    byte CskipChldrn[MAX_NODE_DEPTH+1] = {20,20,20,20,20,0};
  #elif ( STACK_PROFILE_ID == GENERIC_STAR )
    byte CskipRtrs[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
    byte CskipChldrn[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
  #elif ( STACK_PROFILE_ID == NETWORK_SPECIFIC )
    byte CskipRtrs[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
    byte CskipChldrn[MAX_NODE_DEPTH+1] = {5,5,5,5,5,0};
  #endif // STACK_PROFILE_ID
#endif  // RTR_NWK
```

组网-网状网络通讯2

将ZC节点的串口与PC串口相联，并打开ZNetwork软件

ZC上电，其他ZR、ZE节点上电

使用ZNetwork软件即可得到相应拓扑图



通过两节点绑定按钮S1实现事务绑定

绑定后两节点灯状态周期性翻转