

1.	简介.....	1
2.	关于本手册.....	1
3.	定义.....	1
4.	预备知识.....	3
4.1	ZigBee 网络.....	3
4.2	设备类型.....	3
4.2.1	协调器.....	3
4.2.2	路由器.....	4
4.2.3	终端设备.....	4
4.3	栈规范.....	4
4.4	编址.....	4
4.4.1	网络地址分配.....	5
4.4.2	单播、多播和广播.....	5
4.5	绑定.....	7
4.5.1	建立一个绑定表.....	7
4.6	路由.....	8
4.6.1	路由协议.....	8
4.6.2	表存储.....	10
5.	实验前的准备工作.....	11
5.1	相关软件的安装.....	11
5.2	相关开发工具的 PC 机端驱动程序的安装.....	11
5.3	供电.....	11
5.4	IEEE 地址.....	12
6.	<b>ZigBee-2006 网络网状实时定位实验</b> .....	13
13.1	实验内容.....	13
13.2	实验设备.....	13
13.3	实验步骤.....	13
13.3.1	打开 Location 工程.....	13
13.3.2	选择相应的工程配置.....	13
13.3.3	指定设备为网关节点、参考节点或盲节点.....	14
13.3.4	编译并下载应用工程.....	14
13.4	实验操作.....	15
7.	<b>创建用户 Z-Stack 应用工程</b> .....	18
14.1	复制和重命名文件/文件夹.....	18
14.2	编辑工程文件.....	22
14.3	编辑源文件.....	24
14.4	测试被修改的工程和源文件.....	25

## 1. 简介

创维特公司 ZigBee 无线定位系统是目前市场上在最短时间内开发全面 ZigBee 无线定位系统应用的最强大的工具之一,采用德州仪器公司(TI)的 CC2430 和 CC2431 无线 ZigBee 片上系统,结合符合 ZigBee-2006 协议规范的 Z-Stack 协议栈,用户可用来进行 ZigBee-2006 网状网络实时定位系统的开发。

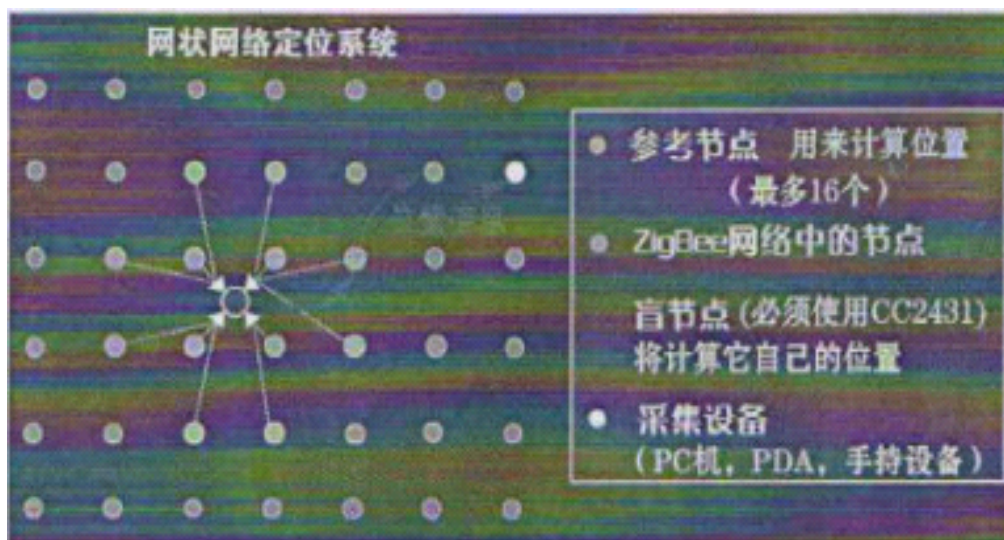
## 2. 关于本手册

本手册是创维特公司 ZigBee 无线定位系统的用户使用手册,适用于创维特公司系列无线 ZigBee 定位开发系统。本手册旨在帮助用户了解定位系统硬件的使用方法、注意事项,相关软件的安装和使用方法以及如何快速完成定位系统定位演示实验。

## 3. 定义

CVT-WSNEMK-V2.1	无线 ZigBee 定位系统节点板。 节点类型: 可作为 ZigBee 定位网关节点、参考节点或盲节点。 供电方式: USB 供电、外置电源供电或电池供电三种供电方式 串行接口: DB9 串行接口或 USB 串行接口两种串行接口方式 指示灯: 连接指示灯 D2,数据收发指示灯 D3 可用跳线设置为抓包分析器 Packet Sniffer
CVT-ZNJD	无线 ZigBee 定位系统工业节点。 节点类型: 可作为 ZigBee 定位考节点、盲节点 供电方式: 外置电源供电或充电电池供电两种供电方式 串行接口: RS232 串行接口或 RS485 串行接口两种串行接口方式 指示灯: 分别有连接指示灯,数据收发指示灯
CC2000Debugger/ Packet Sniffer	专为支持 TI CC2430/CC2431 芯片而推出的仿真器/调试器,可与 IAR EW8051 集成开发环境无缝连接,具有代码高速下载、在线调试、断点、单步、变量观察、寄存器观察等功能,实或对 CC2430/CC2431 片上系统的实时在线仿真、调试。也可作为 IEEE 802.15.4/ZigBee 协议分析仪,可以全面解码复杂的 ZigBee 协议帧。

本手册中的描述假设用户使用 Windows XP 操作系统。



### ZigBee-2006 网状网络实时定位系统

- **参考节点 (设备 ID: 0x0010)** 参考节点的位置固定，一般应干线供电，在空闲时接收器始终打开以便可以始终响应盲节点的位置请求报文。参考节点的坐标位置应由用户指定，它自己根本不会计算自己的坐标，因此参考节点可以由 CC2430 或 CC2431 构成，为降低成本，建议用户使用 CC2430。
- **盲节点 (设备 ID: 0x0011)** 盲节点是可移动节点，可由电池供电并可进入睡眠模式。它将查询射频范围内（即 1 跳邻居）节点的位置并从每一个响应节点的响应报文获取接收信号强度。盲节点必须由 CC2431 构成，以便使用 CC2431 内部的硬件定位引擎来计算盲节点的位置。
- **网关节点(设备 ID: 0x0012)** 该设备是用户 PC 机与参考节点和盲节点之间的桥梁，可作为参考节点和盲节点的无线配置和调试工具，有时我们也称其为无线网关。

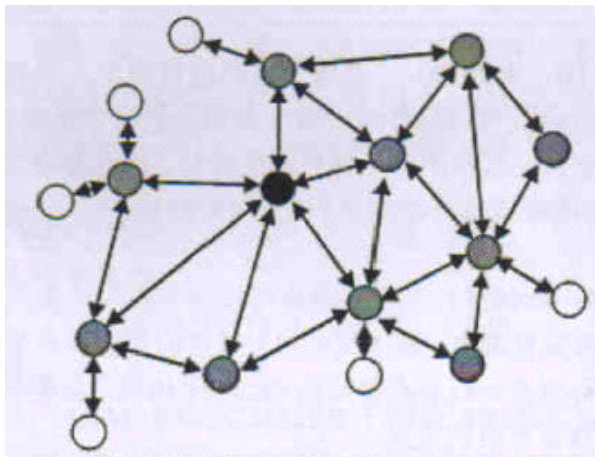
## 4. 预备知识

### 4.1 ZigBee 网络

一个 ZigBee 网络是由干线供电或电池供电设备构成的多跳网络。这意味着在一个 ZigBee 网络中希望交换数据的两个设备可能需要依靠其他中间设备才能成功完成数据交换。因为网络的这个协作性质，每一个设备需要实现恰当的功能：(1)执行具体的网络功能 (2)配置某些参数为特定值。一个设备所执行的网络功能决定了该设备在网络中的角色，这被称为一个设备类型。需要被配置为特定值的这些参数的集合与这些特定值被称为一个栈规范。

### 4.2 设备类型

在一个 ZigBee 网络中有 3 种逻辑设备类型：(1)协调器 (2)路由器 (3)终端设备。一个 ZigBee 网络由一个协调器节点和多个路由器和终端节点组成。注意：设备类型不以任何方式限制运行在一个特殊设备上的应用类型。



上图显示了一个 ZigBee 网络的例子，它由一个 ZigBee 协调器（黑色），多个路由器（红色）和终端设备（白色）构成。

#### 4.2.1 协调器

协调器是“启动”一个 ZigBee 网络的设备。它是网络中的第一个设备。协调器节点选择一个信道和一个网络标示符(也称为个域网络标识符 PANID)，然后启动该网络。

协调器设备也可被随意地用来协助在网络中建立安全和应用级绑定。

注意：协调器的角色主要是和启动和配置网络有关。一旦这个工作完成，协调器的行为就像一个路由器节点（它甚至可以离开网络）。网络的后续操作不依赖协调器的存在，这是因为 ZigBee 网络的分布式性质。

### 4.2.2 路由器

一个路由器执行如下功能 (1) 允许其他设备加入网络 (2) 多跳路由 (3) 协助它的子终端设备通信。

一般而言，路由器被期望一直处于激活状态，因此它必须由干线电源供电。被称为“簇树(Cluster Tree)”的一种特殊的网络运行模式，允许路由器周期性循环操作，因此使得它们可被电池供电。

### 4.2.3 终端设备

一个终端设备没有维护网络基础结构的职责，它可以选择睡眠和唤醒。因此，它可以作为一个电池供电节点。

一般来说，一个终端设备的存储器需求（特别是 RAM 需求）是比较少的。

## 4.3 栈规范

需要被配置为特殊值的栈参数的集合以及这些值被称为栈规范。栈规范包含的这些参数由 ZigBee 联盟定义。

网络中的所有设备必须符合相同的栈规范（即所有设备的栈规范参数配置为相同值）。

为了提高互用性，ZigBee 联盟已经为 ZigBee-2006 规范定义了一个栈规范。所有符合这个栈规范的设备可与网络中来自其它厂商也符合该栈规范的设备互操作。

如果应用开发者选择更改这些参数中任意一个的设置，这些设备将不再能实现与来自其它厂商符合 ZigBee 栈规范的设备的互操作。因此，“封闭网络”的开发者可以选择更改栈规范变量的设置。这些栈规范被称为“专用网络”的栈规范。

一个设备所遵循的栈规范的标识符在该设备的信标传送中被提供。这使得一个设备在加入一个网络之前确定该网络的栈规范。“专用网络”的栈规范的 ID 为 0 而 ZigBee-2006 栈规范的 ID 为 1。

## 4.4 编址

ZigBee 设备有两种类型的地址。一种是 64 位 IEEE 地址（也称为 MAC 地址或扩展地址）。另一种是 16 位网络地址（也称为逻辑地址或短地址）。

64 位 IEEE 地址是全球唯一的地址，它被分配给设备终生使用。它通常在制造或安装时被

设置。这些地址由 IEEE 提供并分配。有关如何获得这些地址的一个块的更多信息可访问 <http://standards.ieee.org/regauth/oui/index.shtml>。

当一个设备加入网络后，它被分配一个 16 位地址，这个地址供该设备在网络中使用。该地址仅在该网络中唯一。它被用来在网络中识别设备和发送信息。

#### 4.4.1 网络地址分配

ZigBee 使用一个分布式的编址方案来分配网络地址。该方案确保了所有被分配的网络地址在整个网络中是唯一的。这是必要的，所以在一个特定数据包应当被路由到哪个设备上这个问题上不存在歧义。同样，编址算法的分布式特性确保了一个设备仅和它的父设备通信以接收一个全网络范围内唯一的地址。地址分配无需全网络范围的通信这有助于网络的可测量性。

编址方案需要一些参数被提前知道并被配置在每一个加入网络的路由器。它们是 MAX\_DEPTH, MAX\_ROUTERS 和 MAX\_CHILDREN 参数。它们是栈规范的一部分并且 ZigBee-2006 栈规范已经为这些参数定义了值 (MAX\_DEPTH = 5, MAX\_CHILDREN=20, MAX\_ROUTERS=6)。

MAX\_DEPTH 参数决定网络的最大深度。协调器在深度 0，它的子节点在深度 1，这些子节点的子节点在深度 2，依此类推。因此，MAX\_DEPTH 参数限制了网络的物理“长度”。

MAX\_CHILDREN 参数决定一路由器（或协调器）可拥有的子节点的最大数量。

MAX\_ROUTERS 参数决定一路由器（或协调器）可拥有的具有路由能力的子节点的最大数量。该参数是 MAX\_CHILDREN 参数的一个子集，(MAX\_CHILDREN-MAX\_ROUTERS)是终端设备的数量。

如果开发者希望改变这些值，他们需要遵守以下步骤：

首先，必须确保这些参数的新值是合法的。因为总的地址空间被限制为  $2^{16}$ ，因此这些参数可被设置为多大是有限制的。

选择合法值后，开发者需要确保不使用标准栈规范而是设置为“专用网络”栈规范。

#### 4.4.2 单播、多播和广播

在 ZigBee 网络中，数据包可被单播，多播或广播。一个单播数据包被发送给一个单一设备，一个多播数据包被发送给一组设备，一个广播数据包一般被发送给网络中的所有设备。下面是更详细的解释。



#### 4.4.2.1 单播

这是普通地址模式，它被用来发送一个数据包给一个单一设备，该设备的地址是已知的。

#### 4.4.2.2 间接寻址

应用并不知道数据包的最终目的地。目的地址从驻留在发送设备协议栈中的“绑定表”中被查询。这个特性称为源绑定。

当数据包被向下发送给协议栈时，目的地址从绑定表中被查询和使用。然后，数据包将被认为是一个正常的单播数据包。如果多个目的设备被发现，数据包的一个拷贝将被发送给它们中的每一个。

在 ZigBee 的以前版本(ZigBee-2004). 有一个选项，可以在协调器上存储绑定表。在那种情况下，发送设备应该发送数据包给协调器，然后该数据包被转发到在协调器的绑定表中被找到的最终目的地。这个可选的特性被称为协调器绑定。

#### 4.4.2.3 广播

当应用想发送一个报文给网络中的所有设备时，使用广播。有以下几种广播地址：

- 0xFFFF 报文将被发送给网络中的所有设备（包括睡眠设备）。对于睡眠设备，报文被保存在它的父设备直到睡眠设备查询它或报文被超时。
- 0xFFFD 报文将被发送给在空闲时接收器处于打开状态(RXONWHENINLE)的全部设备，即除了睡眠设备外的所有设备。
- 0xFFFC 报文被发送给所有的路由器(包括协调器)。

#### 4.4.2.4 组寻址

当应用想发送一个报文给一组设备时，使用组寻址。

注意：组也可以被用来间接寻址。在绑定表中找到的目的地址既可以是一个单播地址也可以是一个组地址。还要注意，在组被建立以前，广播地址是组地址的一个简单特例。

## 4.5 绑定

绑定是控制信息流从一个应用到另一个应用(或多个应用)的一种机制。在 ZigBee-2006 发行版中, 绑定机制可在所有设备上被实施并被称为源绑定。

绑定允许一个应用在不知道目的地址的情况下发送一个报文, 应用支持子层 APS 从它的绑定表中决定目的地址, 然后发送报文到目的应用(或多个应用)或组。

**注意:** 与 **ZigBee 1.0** 发行版在协调器中存储所有的绑定条目不同, 现在所有的绑定条目被存储在发送数据的设备。

### 4.5.1 建立一个绑定表

有 3 种方法来建立一个绑定表:

- **ZigBee 设备对象绑定请求** 一个调试工具可告诉设备做一个绑定记录。
- **ZigBee 设备对象终端设备绑定请求** 两个设备可告诉协调器它们想建立一个绑定表记录。协调器将做出配合并在两个设备中创建绑定表条目。
- **设备应用** 设备上的一个应用可建立或管理一个绑定表。

#### 4.5.1.1 ZigBee 设备对象绑定请求

任何设备或应用可发送一个 ZDO 报文给另一个设备(通过空间)来为网络中发送 ZDO 报文的那个设备建立一个绑定记录。这被称为辅助绑定并且这将为发送设备创建一个绑定条目。

#### 4.5.1.2 ZigBee 设备对象终端设备绑定请求

这种机制在被选定的设备上利用一个按钮按下或其他类似的行动来在一个超时周期内进行绑定。终端设备绑定请求报文在超时周期内将在协调器被收集并且基于已协商一致的规范 ID (profile ID)和簇 ID (cluster ID), 一个绑定表条目被创建。默认的终端设备绑定超时为 16 秒。

#### 4.5.1.3 设备应用绑定管理器

另一种在设备上登记绑定条目的方法是应用为它自己管理绑定表。这意味着应用将通过调用绑定表管理函数登记和撤走本地的绑定表条目。



## 4.6 路由

一个网状网络被描述成为这样一个网络：在该网络中，路由信息被分散提供，包括许多对等设备为彼此利益而协作处理路由。

路由对应用层而言是完全透明的。应用仅简单地将发送给任何设备的数据向下传递给协议栈，然后协议栈负责寻找一条路由。这样一来，应用并不知道它实际上正在一个多跳网络中运行。

路由也使用了 ZigBee 网络的“自我愈合”特性。如果某个无线连接停止了，路由功能将自动查找一个新的路由来避免某些无效连接。这大大提高了无线网络的可靠性，它是 ZigBee 的重要特性之一。

### 4.6.1 路由协议

ZigBee 使用了一种基于 AODV（“按需距离矢量”）的路由协议。它简化了在传感器网络中的使用，适用于支持移动节点，连接故障和数据包丢失的环境。

当一个路由器从它的应用或另一个设备收到一个单播数据包时，网络层按照下述过程进行转发。如果目的地是一个相邻路由器（包括它的子设备），数据包将被直接传送给目的设备。再则，路由器将检查其路由表中与数据包路由目的地相对应的一个条目，如果有一个包含目的地址的路由表条目存在，数据包将被转发给存储在该路由表条目中的下一跳地址。如果没有一个包含目的地址的路由表条目存在，一个路由发现将被启动并且数据包将被缓存直到该过程完成。

ZigBee 终端设备不执行任何路由功能。一个终端设备要发送一个数据包给任何设备，仅简单地将数据包发送给它的父设备，它的父设备将代表它执行路由。同样地，当任何设备要发送一个数据包给一个终端设备并且启动路由发现，该终端设备的父设备将代表它做出响应。

注意：ZigBee 地址分配方案使得基于它的地址来获得任何目的的一条路由成为可能。在 Z-Stack 中，该机制在一旦正常的路由过程不能被启动（一般情况下，由于缺乏路由表空间）时作为一种自动退却。

此外，在 Z-Stack 中，路由实施已经优化了路由表的存储。一般而言，每一个目的设备需要一个路由表条目。但通过合并某一父设备的所有终端设备的所有条目为该父设备的条目，这样存储被优化了而不丧失任何功能。

ZigBee 路由器，包括协调器，执行以下路由功能 (1) 路由发现和选择 (2) 路由维护 (3) 路由过期。

#### 4.6.1.1 路由发现和选择

路由发现是一个过程，网络设备由此协作去寻找和建立通过网络的路由。对于某个目的的设备，一个路由发现可被任何路由设备启动并总是被执行。路由发现机制在源和目的设备之间搜索所有可能的路径并试图选择最好的可能路由。

通过选择最小成本路由来执行路由选择。每一个节点不断地保持与它的所有邻居的“链接费用”的追踪。链接费用通常是接收信号强度的一个函数。通过为一条路由上的所有链接加入链接费用，该路由的“路由费用”被得出。路由算法尝试选择具有最少“路由费用”的路由。

通过使用请求/响应报文来发现路由。一个源设备为一个目的地址请求一条路由，通过广播一个路由请求(RREQ)报文给它的邻居。当一个节点收到一个 RREQ 报文后，它依次重新广播该 RREQ 报文。但在这样做之前，它更新该报文中的费用字段，为这个最新链接加入链接费用。这样一来，该 RREQ 报文就携带了沿着它所经过的所有链接的链接费用总和。该过程被重复，直到该 RREQ 报文到达目的设备。该 RREQ 报文的多个副本将通过不同的可能路由到达目的设备。这些 RREQ 报文中的每一个将包含它所通过路由的总路由费用。目的设备选择包含最佳路由的 RREQ 报文并发回一个路由响应(RREP)报文给源设备。

RREP 报文被沿着这些中间节点反向单播直到它到达最初的请求节点。因为 RREP 报文被沿着这些中间节点反向传回源设备，这些中间节点升级它们的路由表来指示到达目的地的路由。

一旦一条路由被创建，就可按此路由传送数据包。当一个节点失去了与它下一跳节点的连接（当发送数据包时，它接收不到 MAC 确认），该节点通过发送一个 RERR 报文给它的所有潜在接收该 RERR 报文的节点来作废它的路由。当收到 RREQ，RREP 或 RERR 报文后，节点升级它们的路由表。

#### 4.6.1.2 路由维护

网状网络提供了路由维护和自我愈合。中间节点保持对沿着一条链接传输失败的追踪。如果一个链接被确定为是损坏的，上游节点将为使用该链接的所有路由启动路由维修。该项工作在下次一个数据包要使用该路由时通过启动路由重发现来完成。如果路由重发现不能被启动或由于某些原因失败，一个 RERR 报文被发送回源设备，然后它将负责启动一个新的路由发现。无论哪种方式路由将得到自动重建。

#### 4.6.1.3 路由过期

路由表保持着已建立路由的条目。在一段时间内，如果没有数据包被沿着一条路由发送，那么该路由将被标记为过期。过期的路由不被删除直到需要空间。因此在不是绝对必要的情况下，路由不被删除。

## 4.6.2 表存储

路由功能需要路由器维持一些表。

### 4.6.2.1 路由表

每一个 ZigBee 路由器（包括 ZigBee 协调器），包含一个路由表，设备在路由表中存储参与数据包路由所需要的信息。每一个路由表条目包含了目的地址，下一跳节点和链接状态。发送到目的地址的所有数据包通过下一跳节点被路由。此外，为了回收那些已不再被使用的条目所占的空间，路由表中的条目可被标记为过期。

路由表容量指示一个设备路由表有一个空闲的路由表条目或它已经有一个对应目的地址的路由表条目。

### 4.6.2.2 路由发现表

路由器设备参与路由发现，维持一个路由发现表，在一个路由发现进行的同时该表被用来存储临时信息。这些条目只维持路由发现操作这么长时间。一旦一个条目过期，它可被另一个路由发现操作使用。因此，这些条目的数量决定了网络中可被同时执行的路由发现的最大数量。

## 5. 实验前的准备工作

### 5.1 相关软件的安装

创维特公司 ZigBee 定位开发系统的相关软件有：

- Z-Stack v1.4.2 协议栈
- SmartRF Flash Programmer 工具软件
- General Packet Sniffer 工具软件
- IAR Embedded WorkBench(EW8051)集成开发环境
- 上位机定位引擎监控(TI CC2430/CC2431)软件
- 上位机定位管理软件“定位系统实验平台”

以上相关软件的安装与使用请参看我们配套提供的“ZigBee 无线定位开发系统用户使用手册”的第 6 节，此处不再赘述。

另外，由于某些实验需要使用串口通信，建议用户在 PC 上安装一个串口调试软件以方便实验现象的观察。建议使用“串口调试助手 v2.2”，该软件可在网上免费下载使用。

### 5.2 相关开发工具的 PC 机端驱动程序的安装

创维特公司无线 ZigBee 定位开发系统的相关开发工具有：

- CC2000 Debugger 仿真器/调试器软件

以上相关开发工具的 PC 机端驱动程序的安装请参看我们配套提供的“ZigBee 无线定位开发系统用户使用手册”的 5.3 节和 5.4 节，此处不再赘述。

另外，考虑到用户可能使用笔记本电脑进行开发，而目前大多数主流笔记本电脑不提供串口，为了方便用户在没有串口的笔记本电脑上和我们的目标板进行串口通信，我们在 CVT-WSNEMK-V2.1 上使用了 CP2102 芯片进行 USB 到串口的转换，CP2102 驱动程序的安装请参看我们配套提供的“ZigBee 无线定位开发系统用户使用手册”的 5.3 节。

### 5.3 供电

给目标板供电是用户在进行实验前需要注意的问题，请参看我们配套提供的“ZigBee 无线定位开发系统用户使用手册”的 5.1.1 节。

## 5.4 IEEE 地址

创维特公司 ZigBee 无线定位开发系统中的每一个 Zigbee 节点都已经被预编程了一个唯一的 64 位 IEEE 地址，以 Little-Endian 格式存储，位于 CC2430/CC2431 的 Flash 存储器的最高 8 个字节中。在每个 Zigbee 节点显眼位置都贴有相应的 IEEE 地址。

Z-Stack 将 Flash 存储器中保存 IEEE 地址的区域看作“仅写一次”存储器。当试图在该区域写一个 IEEE 地址时，仅在该区域的内容为空 (0xFFFFFFFFFFFFFFFF) 时成功。换句话说，若该区域的值是不为 0xFFFFFFFFFFFFFFFF 的其他任何 8 字节值，被认为是一个有效的 IEEE 地址，将不被修改。

如果用户选择在下载实监控程序到 CC2430/CC2431 的 Flash 存储器前首先进行全片擦除操作，CC2430/CC2431 的 IEEE 地址将被清空(0xFFFFFFFFFFFFFFFF)，那么实验程序在运行时会使 CVT-WSNEMK-V2.1 上的 D2（绿色）和 D3（黄色）指示灯同时快速闪烁以指示用户该设备的 IEEE 地址无效，然后该设备将产生一个随机的 IEEE 地址来使用，如果未包含 ZTOOL\_Px(x 取 1 或 2)定义，则该 IEEE 地址将被写入 CC2430/CC2431 的 Flash 中。如果用户想恢复 CVT-WSNEMK-V2.1 出厂时分配的 IEEE 地址，可以使用 SmartRF Flash Programmer 工具软件将我们出厂时分配的 IEEE 地址写入到 CVT-WSNEMK-V2.1。SmartRF Flash Programmer 工具软件的使用方法请参看我们配套提供的“ZigBee 无线定位开发系统用户手册”的 6.2 小节。

## 6. ZigBee-2006 网络网状实时定位实验

本实验是 ZigBee-2006 网状网络实时定位系统实验，可作为用户二次开发的参考。

### 13.1 实验内容

采用 8 个参考节点作为固定节点，2 个盲节点作为可移动的节点。在用户 PC 机上运行创维特公司 Z-Location 上位机定位软件，通过网关节点对参考节点的位置(X 和 Y)进行无线配置，对盲节点参数进行无线配置。然后用户 PC 机通过网关节点来捕获 2 个盲节点的坐标数据，并进行实时位置显示。

### 13.2 实验设备

实验设备	数量	备 注
CVT-WSNEMK-V2. (带 CC2430 模块)	1 个	作为网关节点 (协调器设备)。
CVT-WSNEMK-V2. (带 CC2430 模块)	8 个	作为参考节点(路由器设备)。
CVT-WSNEMK-V2. (带 CC2431 模块)	2 个	作为盲节点(路由器设备)。
CC2000 Zigbee 仿真器与包分析器	1 个	下载和调试程序。

### 13.3 实验步骤

#### 13.3.1 打开 Location 工程

Location 工程位于“...\Projecs\zstack\Samples\Location\CC2430DB”文件夹中，用鼠标左键双击“SampleApp.eww”文件，Location 工程在 IAR Embedded WorkBench 集成开发环境中被打开。

#### 13.3.2 选择相应的工程配置

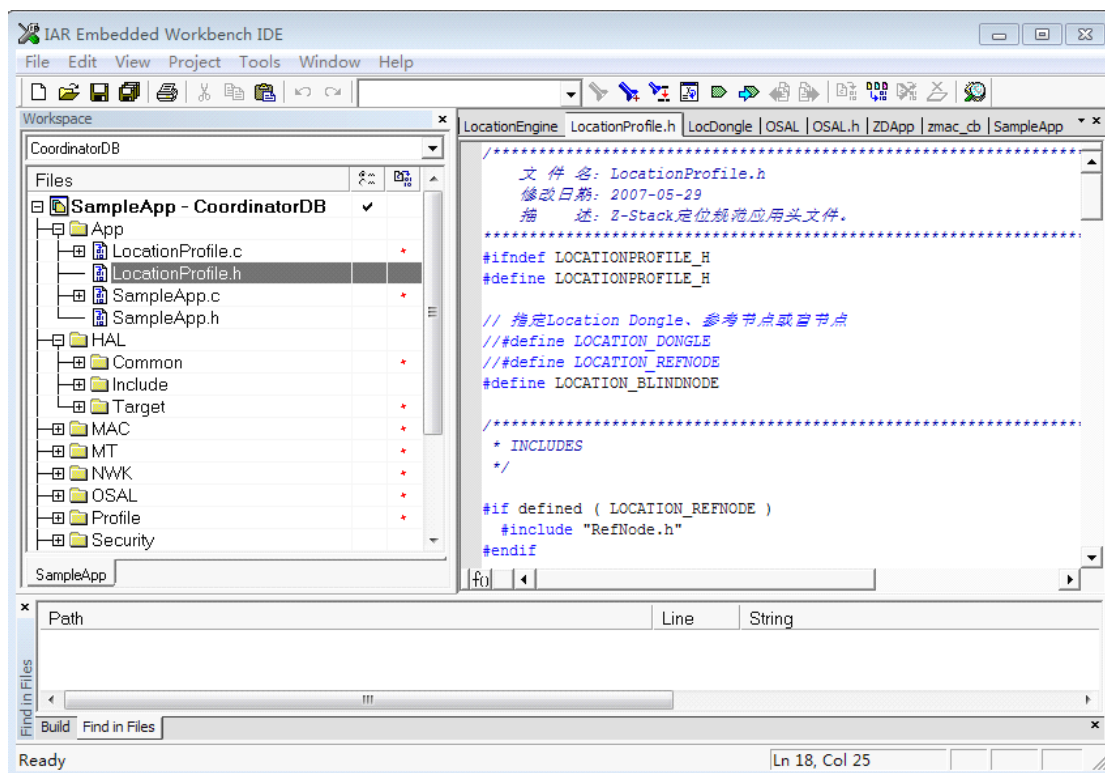
**Workspace** 下的下拉列表中的选项是工程配置项目，本实验我们使用其中的这两个工程配置：

- CoordinatorEB 将 CVT-WSNEMK-V2.1 配置为协调器设备
- RouterEB 将 CVT-WSNEMK-V2.1 配置为路由设备



### 13.3.3 指定设备为网关节点、参考节点或盲节点

如下图所示：



在 LocationProfile.h 文件内容的开始部分，用户可以通过包含不同的预定义来决定设备是作为网关节点（Location Dongle）、参考节点还是盲节点。

- 包含#define LOCATION\_DONOLE 设备将作为网关节点（Location Dongle）
- 包含#define LOCATION\_REFODE 设备将作为参考节点
- 包含#define LOCATION\_BLINDNODE 设备将作为盲节点

### 13.3.4 编译并下载应用工程

在选择了相应的工程配置选项并包含了相应的设备类型（Location Dongle、参考节点或盲节点）预定义后，就可以编译并下载应用工程到 CVT-WSNEMK-V2.1 无线 Zigbee 定位系统节点板中。

在本实验中，我们使用 1 个 CVT-WSNEMK-V2.1（带 CC2430 模块）作为网关节点（Location Dongle），8 个 CVT-WSNEMK-V2.1(带 CC2430 模块)作为参考节点（路由器），2 个 CVT-WSNEMK-V2.1 (带 CC2431 模块)作为盲节点（路由器）。

下面介绍编译和下载过程。

选择 CoordinatorEB 工程配置, 在 LocationProfile.h 文件内容的开始部分只包含#define LOCATION\_DONGLE 预定义, 通过点击 **Project** 下拉菜单中的 **Rebuild All** 项来编译应用工程, 编译过程大约需要 30~40 秒。编译完成后, 用 CC2000 调试仿真器的 10PIN 扁平电缆调试接头插入 CVT-WSNEMK-V2.1 上的 JTAG 调试接口 JP1 上, 然后将 CC2000 调试仿真器用 USB 电缆与用户 PC 机的 USB 端口连接, 用户 PC 机的 USB 端口通过 CC2000 给 CVT-WSNEMK-V2.1 供电(用跳线帽将 J5 的 BAT 和 VCC3.3 连接, 将 J3 的 EXT5 和 VCC5 连接), 此时 CVT-WSNEMK-V2.1 上的电源指示灯 D4 (红色)将被点亮。

通过点击 **Project** 下拉菜单中的 **Debug** 项 (或按 Ctrl+D 快捷键) 来下载应用工程, 下载过程大约需要 20~30 秒。在下载过程结束后, IAR Embedded WorkBench 集成开发环境进入到调试界面, 通过点击 **Debug** 下拉菜单中的 **Stop Debugging** 项 (或按 Ctrl+Shift+D 快捷键) 来退出调试界面。

断开 CVT-WSNEMK-V2.1 与 CC2000 调试仿真器的连接。

选择 RouterEB 工程配置, 在 LocationProfile.h 文件内容的开始部分只包含#define LOCATION\_REFNODE 预定义, 用 CC2000 调试仿真器连接 CVT-WSNEMK-V2.1 (带 CC2430 模块) 与用户 PC 机, 然后编译并下载应用工程, 再对另外 7 个 CVT-WSNEMK-V2.1(带 CC2430 模块)进行应用工程的下载。

断开 CVT-WSNEMK-V2.1 与 CC2000 调试仿真器的连接。

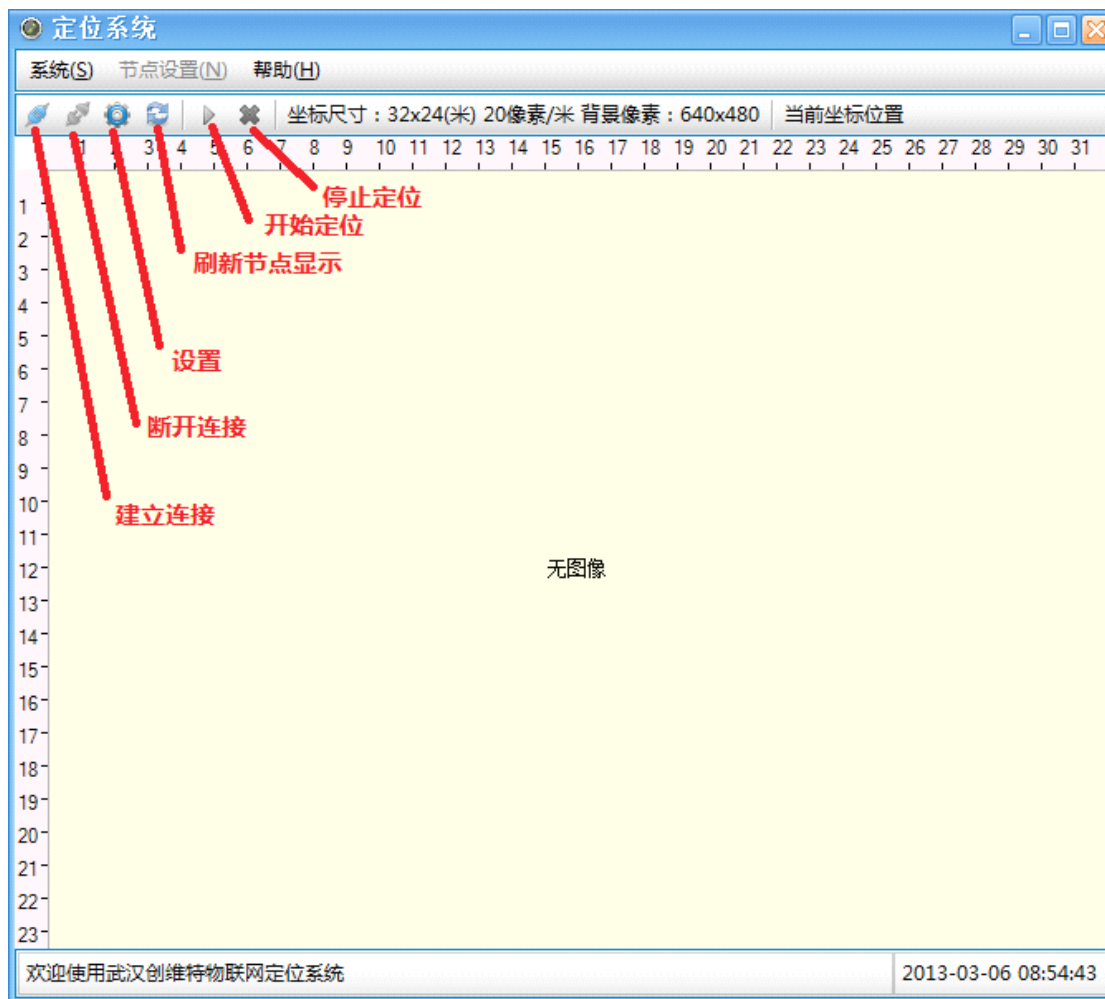
选择 RouterEB 工程配置, 在 LocationProfile.h 文件内容的开始部分只包含#define LOCATION\_BLINDNODE 预定义, 用 CC2000 调试仿真器连接 CVT-WSNEMK-V2.1 (带 CC2431 模块)与用户 PC 机, 然后编译并下载应用工程, 再对另外 1 个 CVT-WSNEMK-V2.1 (带 CC2431 模块)进行应用工程的下载。


## 13.4 实验操作

将网关节点 (Location Dongle) 用 USB 电缆直接连接到用户 PC 的 USB 端口, 将网关节点上的 J5 用跳线帽将 EXT3.3 和 VCC3.3 连接, J3 用跳线帽将 VBUS 和 VCC5 连接, 确保用户 PC 机的 USB 端口给网关节点 (Location Dongle) 正常供电。在用户 PC 机上运行创维特公司 A-Location 上位机定位软件。给所有的参考节点上电, 一般采用外置电源供电方式, 需用跳线帽将 J5 的 EXT3.3 和 VCC3.3 连接, 将 J3 用跳线帽将 EXT5 和 VCC5 连接。给所有盲节点上电, 一般采用电池供电, 以方便移动, 需用跳线帽将 J5 的 BAT 和 VCC3.3 连接, J3 用跳线帽将 EXT5 和 VCC5 连接。然后用户 PC 机通过网关节点 (Location Dongle) 对参考节点的坐标以及盲节点的参数进行配置。然后将参考节点布置在用户指定的实际物理位置上。现在就可以通过网关节点 (Location Dongle) 捕获盲节点的坐标并在用户 PC 上进行实时显示了。


详细操作步骤如下: (实验前请先确定 PC 机上已经安装好 PC 端定位管理软件, 安装请参看我们配套提供的 “ZigBee 无线定位开发系统用户使用手册” 的第 6 节)

1) 打开 PC 端定位管理软件, 如下图:



- 2) 点击设置图标, 进行 PC 串口选择, 波特率不需要修改。(默认选择的是串口 1, 若使用的是串口 1 则此步骤可以跳过)

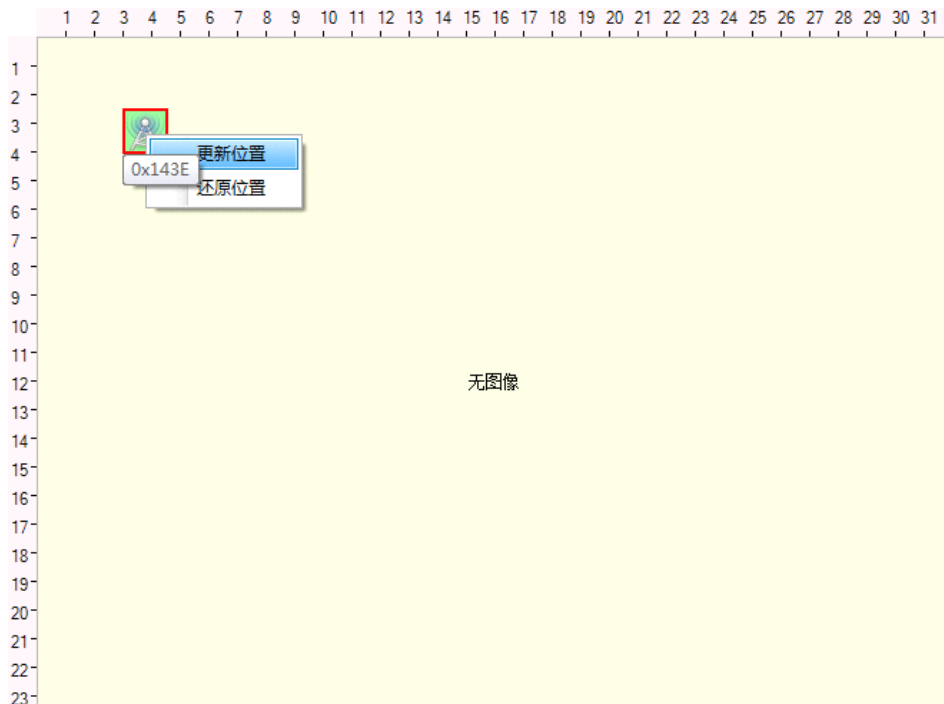



- 3) 连接好网关节点的串口和电源, 点击连接图标, 连接正常软件下方会出现如下提示。

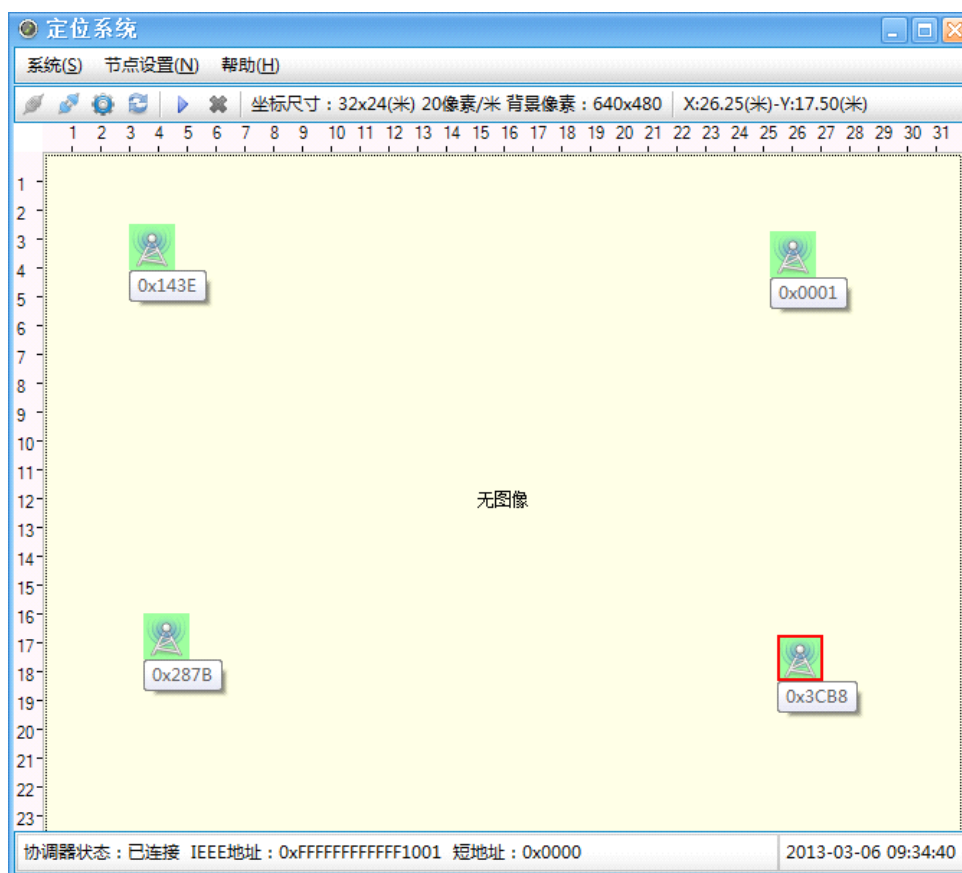
协调器状态: 已连接 IEEE地址: 0xFFFFFFFFFFFF1001 短地址: 0x0000

- 4) 将参考节点上电, 上电后软件上会出现一个参考节点图标, 使用鼠标按住节点拖动到任意参考位置上, (注意参考位置的设定一定要根据实际摆放情况而定, 参考节点的位置会直接影响到盲节点的定位准确度) 然后点击鼠标右键, 选择更新位置, 此时当前

的参考节点的参考位置就被网关记录下来了，下次再上电的时候，该参考节点会自动更新到设置的参考位置上。依次设置其他所有参考节点（一个一个的上电设置，否则将无法区分显示的参考节点和实际节点的对应关系）。

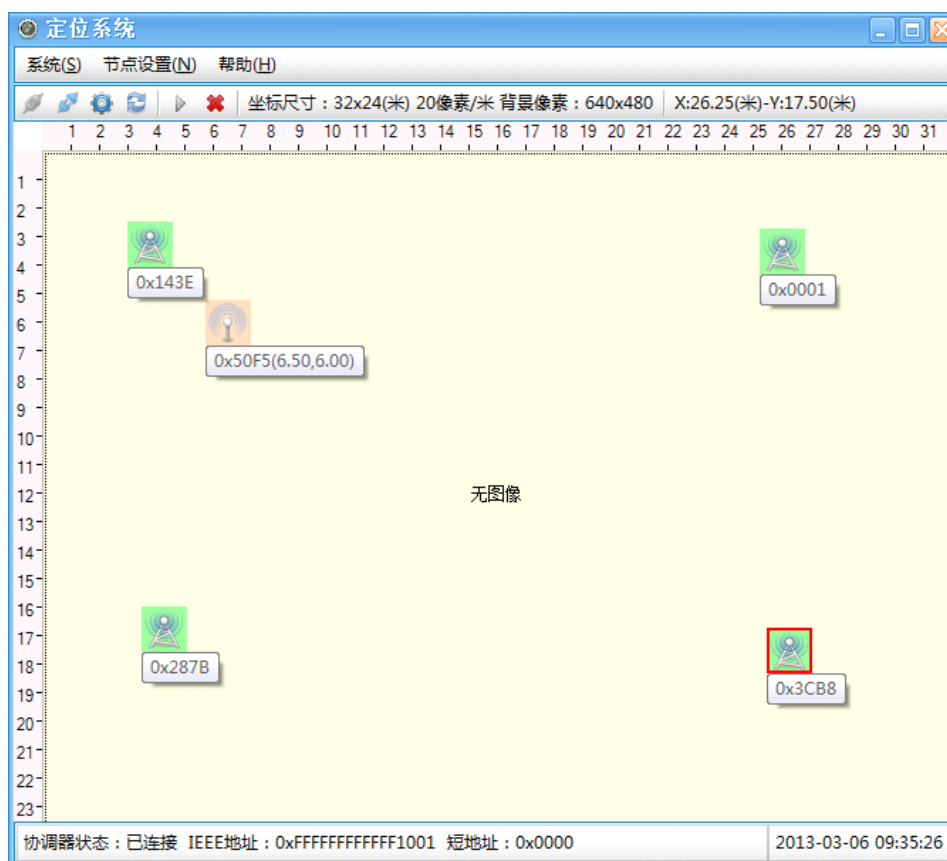


5) 确认所有的参考节点都设定完成后，点击 ，开始定位。



6) 将盲节点上电，此时可以看到软件上盲节点图标会定位到所有参考节点的中间，并根

据移动情况变化位置。



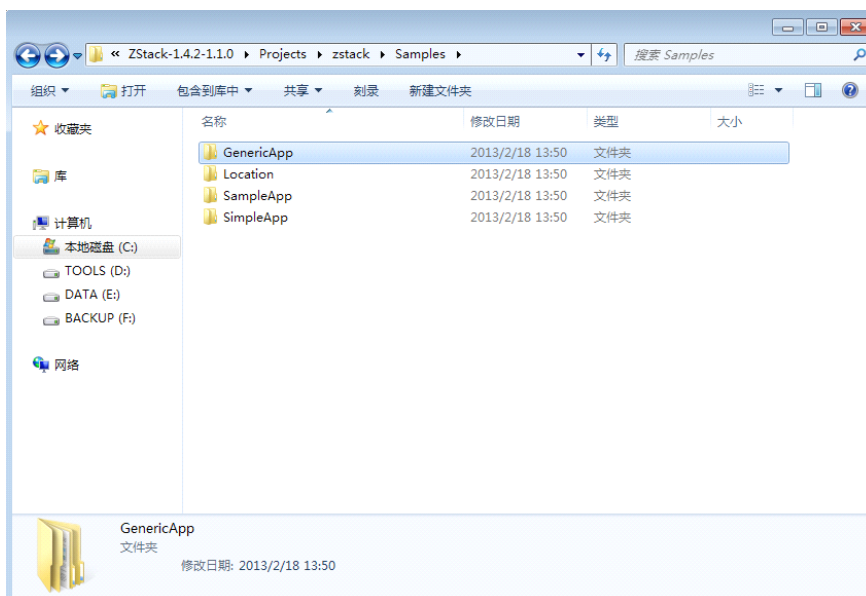
## 7. 创建用户 Z-Stack 应用工程

本节描述了如何一步一步“克隆”一个 Z-Stack 样板工程来作为开发一个新应用的模板。

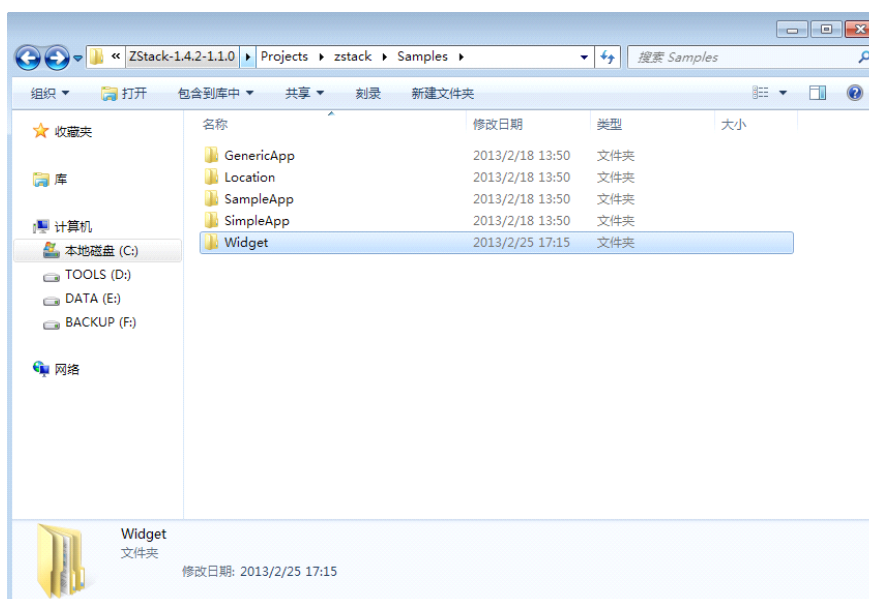
### 14.1 复制和重命名文件/文件夹

确定你想要模仿的应用类型。如果你的应用使用串口来发送和接收串行数据(不使用 Z-Tool), 你应该从 SerialApp 工程开始, 该工程位于... \Projects\zstack\Utilities。如果你的应用不使用串口 (除了 Z-Tool 外), 你应该从 GenericApp 工程开始, 该工程位于... \Projects\zstack\Samples。

- 请看下面的这个例子, 我们将复制和修改... \Projects\zstack\Samples\GenericApp 工程:

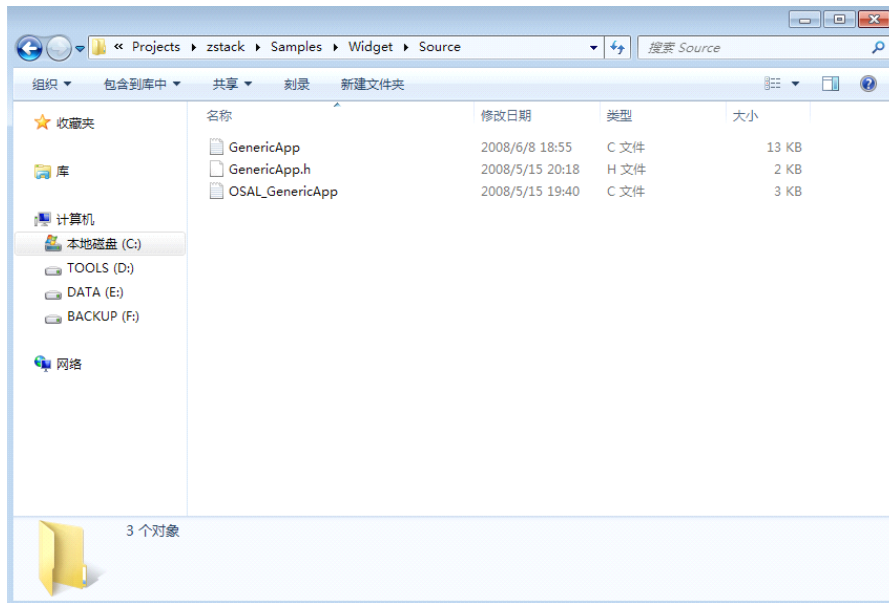


- 复制 GencricApp 文件夹并重命名为 Widget:

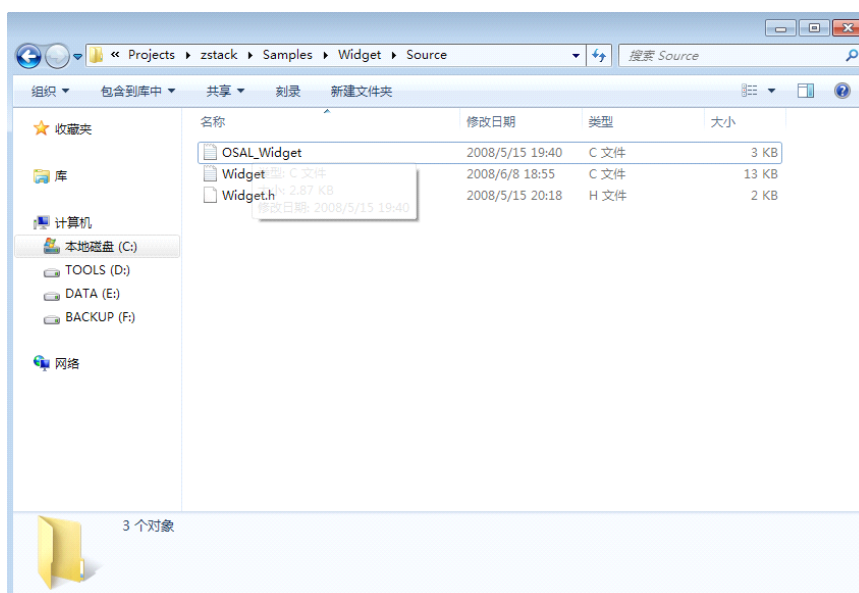




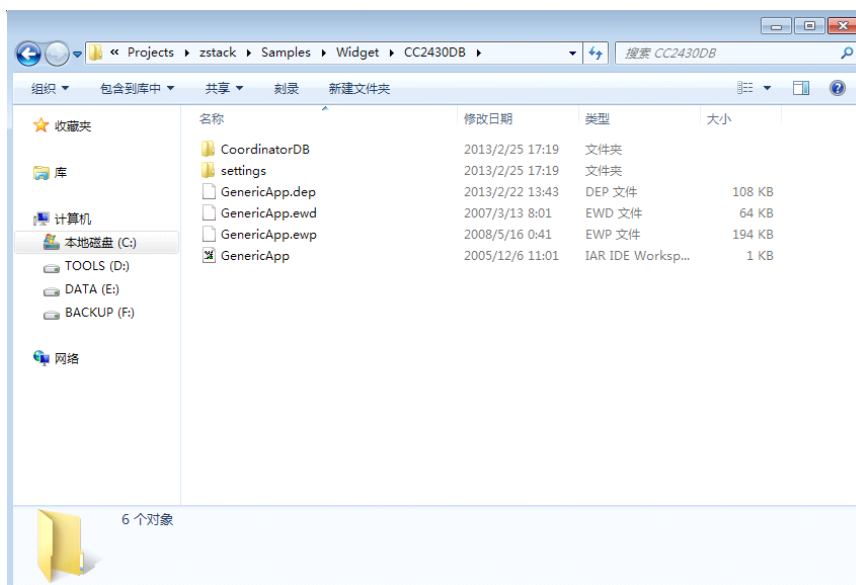
- 打开 Widget 文件夹中的 Source 文件夹：



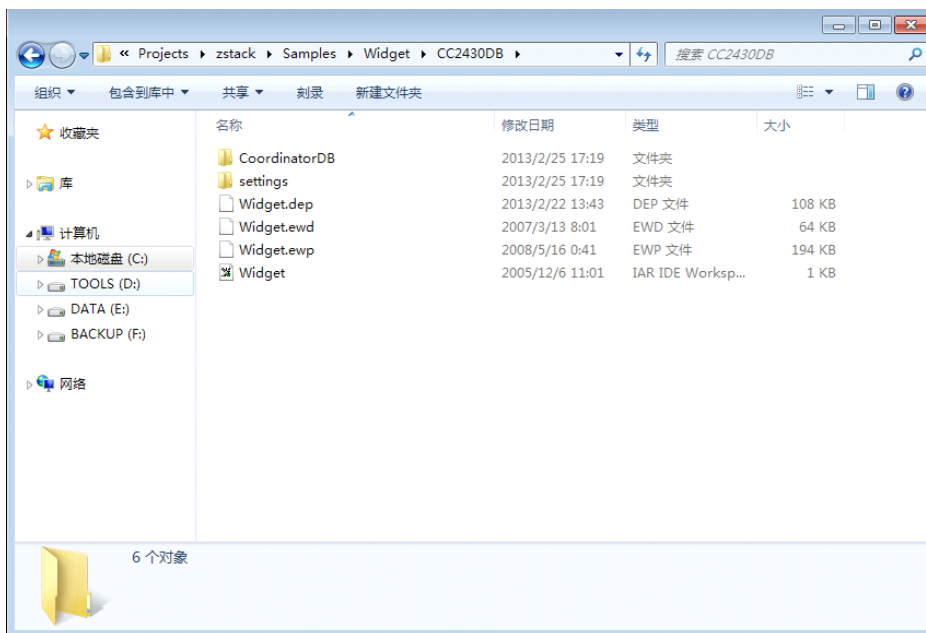
- 给每一个文件改名，用 Widget 替换 GencricApp:



- 打开 Widget 文件夹中的 CC2430DB 文件夹:



- 给每一个文件改名，用 Widget 替换 GencricApp:



## 14.2 编辑工程文件

用记事本打开...\Widget\CC2430DB 文件夹中的 Widget.eww 文件 (该文件是 IAR Embedded WorkBench 的工作空间文件), 准备对它进行编辑:

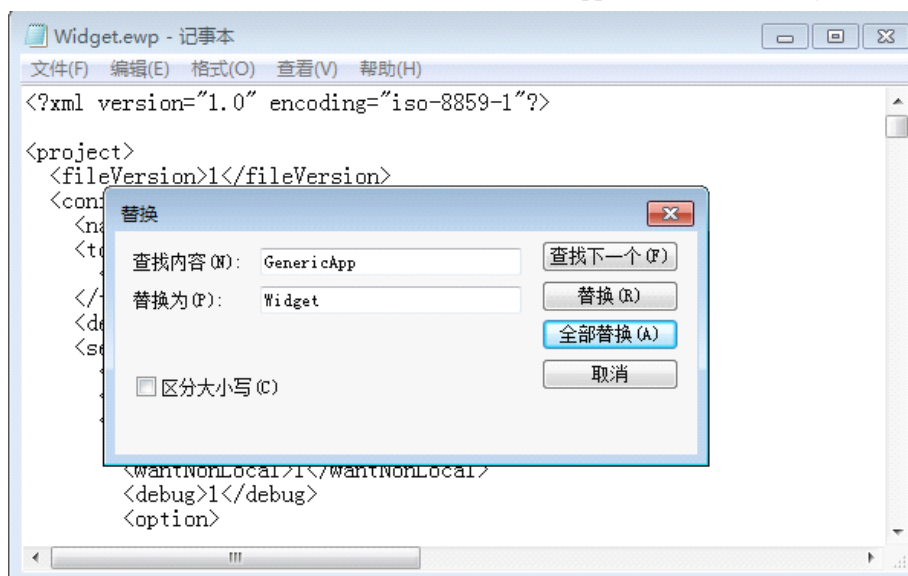
```
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?xml version="1.0" encoding="iso-8859-1"?>

<workspace>
  <project>
    <path>$WS_DIR$\GenericApp.ewp</path>
  </project>
  <batchBuild>
    <batchDefinition>
      <name>Build All</name>
      <member>
        <project>GenericApp</project>
        <configuration>CoordinatorDB</configuration>
      </member>
      <member>
        <project>GenericApp</project>
        <configuration>CoordinatorEB</configuration>
      </member>
      <member>
        <project>GenericApp</project>
        <configuration>RouterDB</configuration>
      </member>
      <member>
        <project>GenericApp</project>
        <configuration>RouterEB</configuration>
      </member>
      <member>
        <project>GenericApp</project>
        <configuration>EndDeviceDB</configuration>
      </member>
      <member>
        <project>GenericApp</project>
        <configuration>EndDeviceEB</configuration>
      </member>
    </batchDefinition>
  </batchBuild>
</workspace>
```

将该文件中的 GencricApp 全部替换为 Widget:

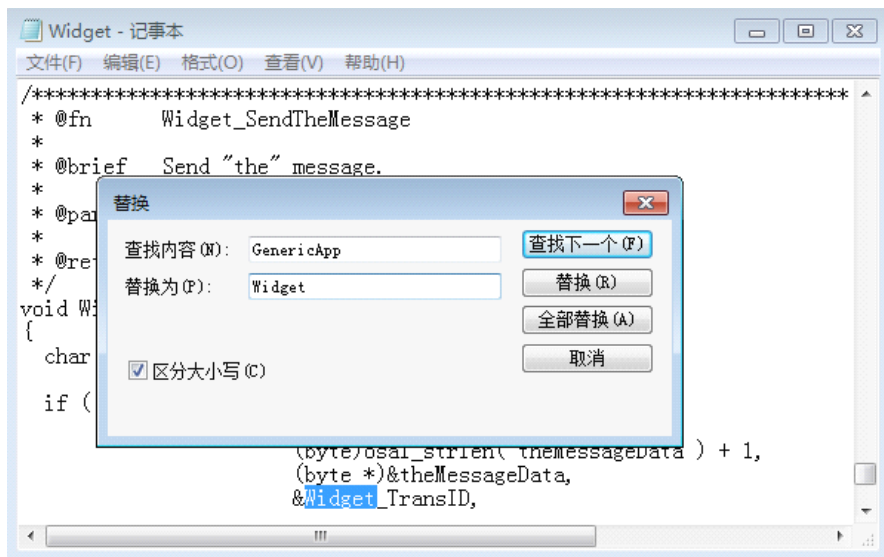
```
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?xml version="1.0" encoding="iso-8859-1"?>
<workspace>
  <project>
    <path>$WS_DIR$\Widget.ewp</path>
  </project>
  <batchBuild>
    <batchDefinition>
      <name>Build All</name>
      <member>
        <project>Widget</project>
        <configuration>CoordinatorDB</configuration>
      </member>
      <member>
        <project>Widget</project>
        <configuration>CoordinatorEB</configuration>
      </member>
      <member>
        <project>Widget</project>
        <configuration>RouterDB</configuration>
      </member>
      <member>
        <project>Widget</project>
        <configuration>RouterEB</configuration>
      </member>
      <member>
        <project>Widget</project>
        <configuration>EndDeviceDB</configuration>
      </member>
      <member>
        <project>Widget</project>
        <configuration>EndDeviceEB</configuration>
      </member>
    </batchDefinition>
  </batchBuild>
</workspace>
```

用记事本打开...\Widget\CC2430DB 文件夹中的 Widget.ewp 文件（该文件是 IAR Embedded WorkBench 的工程文件），将该文件中的 GenericApp 全部替换为 Widget：

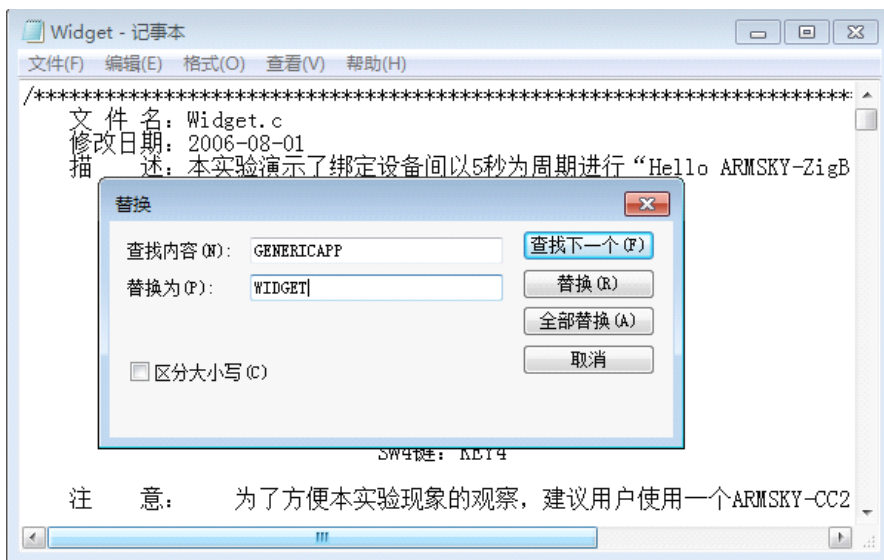


## 14.3 编辑源文件

用记事本打开...\Widget\Source 文件夹中的 Widget.c 文件，选中区分大小写，将该文件中的 GencricApp 全部替换为 Widget:



然后再把 GENERICAPP 全部替换为 WIDGET:

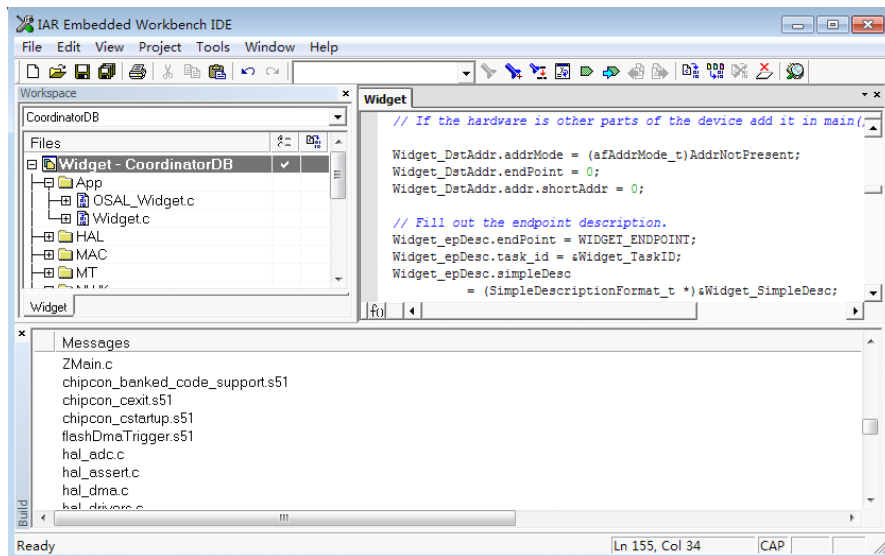


用记事本打开...\Widget\Source 文件夹中的 Widget.h 文件，按上述步骤进行同样的替换操作。

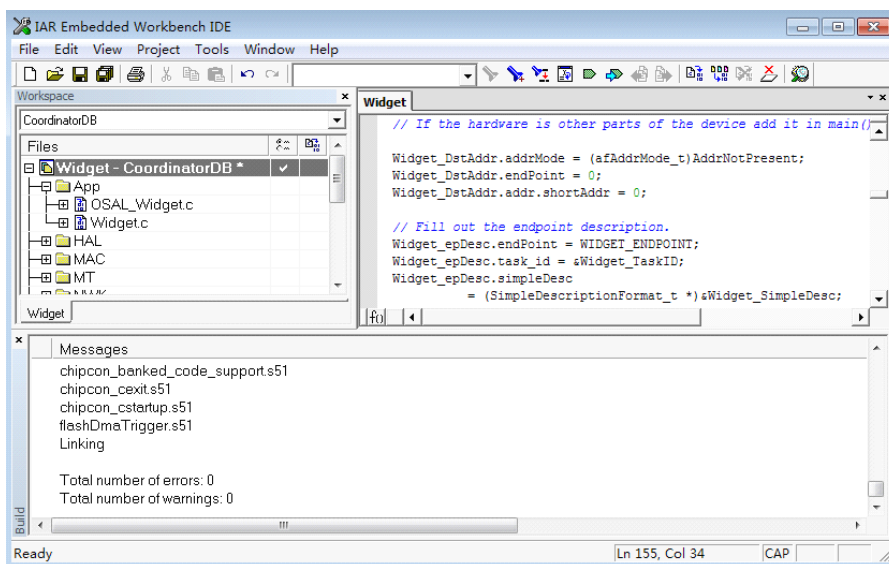
用记事本打开...\Widget\Source 文件夹中的 OSAL\_Widget.c 文件，将该文件中的 GencricApp 全部替换为 Widget:

## 14.4 测试被修改的工程和源文件

在完成工程文件和源文件的修改后，可以通过编译工程来测试更改。双击...\\Widget\\CC2430DB 文件夹中的 Widget.eww 文件来启动 IAR Embedded Workbench，鼠标右键点击文件列表中的 Widget-CoordinatorDB 项，在弹出菜单上选择 **Rebuild All**：



编译器和链接器的状态显示在 **Messages** 框内，**Messages** 框一般在集成开发环境的底部：



现在，Widget 工程已经被当作创建一个实际工程的模板。一般情况下，Widget.c 和 Widget.h 文件可能需要被修改来建立用户个性化的应用。