# 第一部分 物联网简介

根据物联网中信息的生成、传输、处理和应用的过程，可以把物联网系统从结构上分为3层：感知层、传输层（网络层）、应用层。
网关节点实现与互联网等外部网络的链接，将收到的数据转发到外部网络上。
物理传感器：是检测物理量的传感器，它是利用某些物理效应，将被测的物理量转化为便于处理的能量信号的装置。
化学传感器：必须具有对被测化学物质的形状或分子结构进行俘获的功能，并且还能将所获得的化学量有效地转换为电信号。
RFID是一种无线射频识别技术。
物联网的特点：
一是互联网特征；二是识别与通信特征；智能化特征。
物联网大规模地应用面临着的挑战，至少包括三个方面：
一是成本的挑战；二是安全的挑战；三是侵犯隐私的威胁。
无线通信技术包括：

- 移动通信网络
- 宽带无线接入
- 射频与微波通信（短距离无线通信）

短距离无线通信三个重要特征：低成本、低功耗、对等通信
Zigbee的优势：Zigbee相比于现有的wifi、蓝牙等无线技术更加安全、可靠，同时由于其组网能力强、具备网络自愈能力并且功耗更低

# 第二部分 Zigbee

## 第一章 Zigbee概述

**什么是短距离无线通信?**
在一般意义上，只要通信收发方通过无线电波传输信息，并且传输距离限制在较短的范围内，通常是几十米以内，就可以称为短距离无线通信。
**什么是Zigbee?**
Zigbee是IEEE802.15.4协议的代名词。根据这个协议规定的技术是一种短距离、低功耗的无线通信技术。
其特点是近距离、低复杂度、自组织、低功耗、低数据速率、低成本，主要适合用于自动控制和远程控制领域，可以嵌入各种设备。
Zigbee可工作在2.4GHz频段（全球流行）、868MHz（欧洲）915MHz（美国流行）3个频段上。

**什么是cc2530?**

CC2530是一款高性能和低功耗的8051微控制器核。集成符合Zigbee标准的2.4GHz的RF无线电收发机。

**Zigbee设备类型**

1. Zigbee协调器（ZC Coordinator）：上电启动和配置网络，一旦完成后相当于路由器功能。每个Zigbee网络必须有一个。
2. Zigbee路由器（ZR Router）：允许其他设备接入、协助子节点通信、座机座位终端节点应用
3. Zigbee终端设备（ZED End-device）：向路由节点传递数据、没有路由功能、低功耗（Zigbee的低功耗主要体现在这里）、可选择休眠与唤醒（终端节点电池供电）

## （一）Zigbee协调器（Coordinator）

它包含所有的网络信息，是3种设备中最复杂的，存储容量大、计算能力强。主要用于：发送网络信标、建立一个网络、管理网络节点、存储网络节点信息、寻找一对一节点间的路由信息并且不断的接收信息。一旦网络建立完成，这个协调器的作用就像路由器节点。

## （二）Zigbee路由器（Router）

允许其他设备加入这个网络，跳跃路由，辅助子树下电池供电终端的通信。通常，路由器全时间处在活动状态，因此为电源供电。但是在树状拓扑中，允许路由器操作周期运行，因此这个情况下允许路由器电池供电。

## （三）Zigbee终端设备（End-device）

不参与网络维护，只负责感知和发送信息，可以休眠和唤醒，采用电池供电。

## （四）Zigbee网络

一个Zigbee网络由一个协调器节点、多个路由器和多个终端设备节点组成。

**拓扑结构：**Zigbee支持三种自组织无线网络类型：星型结构、网状结构和簇状结构。

Zigbee采用自组织网的通信方式。

动态路由结合网状拓扑结构可以确保Zigbee网络在工业多变环境下的稳定性。

## （五）Zigbee协议

Zigbee协议由应用层、网络层、数据链路层（MAC）、物理层组成。网络层以上协议由Zigbee联盟制定，IEEE802.15.4负责物理层和链路层标准。

物理层功能：由射频收发器以及底层的控制模块构成。

数据链路层功能：为高层访问物理信道提供点到点通信的服务接口。

网络层功功能：提供一些必要的函数，确保Zigbee的MAC层正常工作，并且为应用层提供

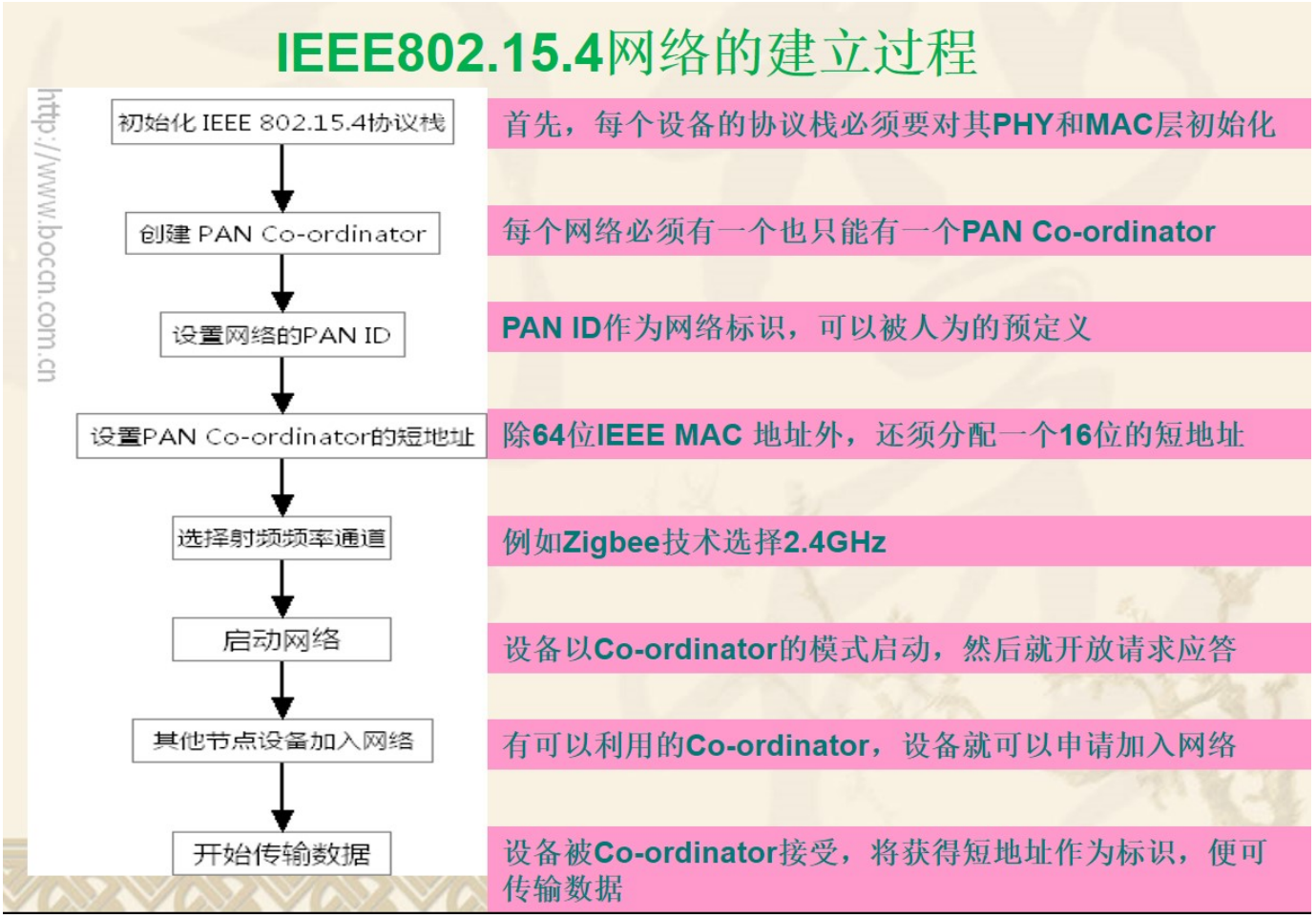
合适的服务接口
应用层功能：主要负责把不同的应用映射到Zigbee网络上，具体包括安全与鉴权、设备发现、服务发现等。
Zigbee应用领域：智能家居、工业领域、智能交通等。

# 第二章 Zigbee-cc2530-IAR

## （一）Zigbee芯片

全球有多家Zigbee芯片，具有代表性的是TI公司的CC2530，存储容量最大支持256K，通信距离可以达到400米。
主要特点：高性能和低功耗的8051微控制器核、集成符合IEEE802.15.4标准的2.4GHz的RF无线电收发机.....

## （二）IEEE802.15.4网络的建立过程




Zigbee网络地址分为两种：1、全球唯一的64位的MAC IEEE地址 2、Zigbee网络内唯一的16位的短地址，用于在Zigbee网络中辨识设备。
一个节点是一个设备，有一个射频端，一个64为IEEE地址，一个16位网络地址。

## （三）Z_stack协议栈相关概念

Z-stack是TI公司开发的一款ZigBee协议栈，经过了ZigBee联盟的认可，为全球总多开发商所广泛采用。

TI公司的Z-Stack协议栈装载在一个基于IAR开发环境的工程里。

Z-Stack采用事件轮询机制，当各层初始化之后，系统进入低功耗模式，当事件发生时，唤醒系统，开始进入中断处理事件，结束后继续进入低功耗模式。如果同时有几个事件发生，判断优先级，逐次处理。这种软件架构可以极大滴降低系统功耗。

Z-Stack采用分层的软件结构：**硬件抽象层（HAL）**提供各种硬件模块的驱动。**操作系统抽象层OSAL**实现了一个易用的操作系统平台。

整个Z-Stack的主要工作流程，大致分为系统启动，驱动初始化，OSAL初始化和启动，进入任务轮询几个阶段。

**协议栈主函数**

协议栈已经将主函数放在了库文件中，程序先是从main函数开始运行，main函数实现的功能是，初始化硬件、初始化网络（加入/创建网络）、初始化任务列表、进入任务处理循环。


**OSAL操作系统函数**

顺利完成上述初始化后，开中断执行osal_start_system( )函数开始运行OSAL系统

该任务调度函数按照优先级检测各个任务是否就绪，如果存在就绪的任务则调用tasksArr[ ]中相对应的任务处理函数去处理该事件，直到执行完所有就绪的任务

如果任务列表中没有就绪的任务，则可以使处理器进入睡眠状态实现低功耗

osal_start_system( )一旦执行，则不再返Main( )函数

**OSAL是协议栈的核心**

Z-Stack的任何一个子系统都作为OSAL的一个任务，因此在开发应用层的时候，必须通过创建OSAL任务来运行程序。

通过osalInitTasks()函数创建OSAL任务，其中TaskID为每个任务的唯一标识号
任何OSAL任务必须分为两步：一是进行任务初始化、二是处理任务事件。

### （四）IAR

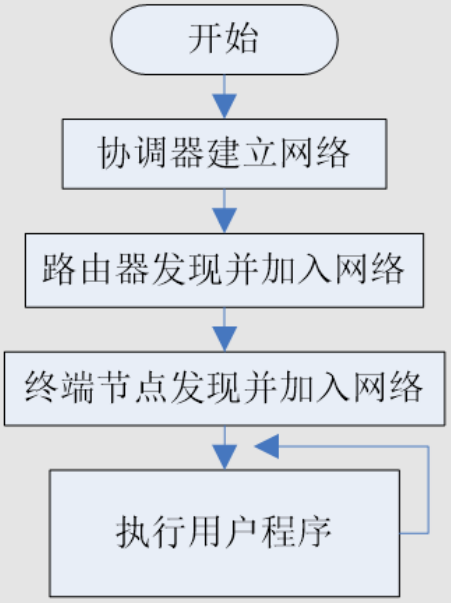
创建新项目的后缀名为.ewp。

# 第三章 Zigbee系统应用开发流程分析

### （一）Zigbee特性

ZigBee采用数据帧的概念，每个帧包含了众多信息，如时间、地址、命令、同步等，真正的数据只占很少部分，这正是ZigBee高可靠传输的关键。
ZigBee网络中传输的三类数据：周期性数据（如水电气数据）、间断性数据（如电灯、家用电器控制）、反复性的低反应时间的数据（如鼠标、操作杆传输的数据）
1ZigBee的频带和数据传输率：频带：2.4GHz；适用范围：全球；数据传输率：250kbps；信道数：16.
ZigBee网络层服务分为两种：1、网络层数据服务：产生网络协议数据单元、选择通信路由；2、网络层管理服务：配置新设备、开始新网络、加入和离开网络等等
Z-Stack协议栈的运行流程：

## 2. Z-STACK 协议栈的运行流程





# 第四章 Zigbee点播、广播以及组播通信

Zigbee网络数据发送，不同的组网模式涉及到不同的网络发送设置。在Zigbee网络中进行数据通信主要有三种类型：广播（Broadcast）、单播（Unicast）和组播（Multicast）。
那么Zigbee是如何实现上述通信方式的呢?
Zigbee协议栈将数据通信息过程高度抽象，使用一个函数完成数据的发送，以不同的参数来选择数据发送方式。Zigbee协议栈中数据发送函数原型如下：
**afStatus_t AF_DataRequest(afAddrType_t *dstAddr , .....,uint8 radius)**
在上面的函数中，第一个参数是一个指向afAddrType_t类型的结构体的指针，该结构体拥有一个枚举类型参数afAddrMode_t addrMode，该枚举类型结构如下：

```
typedef enum
{
    afAddrNotPresent = AddrNotPresent,    //按照绑定方式（实现将两个设备进行绑定，不需要设置地址）进行点播
```

```
    afAddr16Bit = Addr16Bit,          //指定目标地址进行点播
    afAddrGroup = AddrGroup,         //组播
    afAddrBroadcast = AddrBroadcast //广播
}afAddrMode_t;
```

他们分别对应着如注释所示的发送方式。上述的AddrNotPresent、AddrGroup等是一个常数，在Zigbee协议栈中的定义如下：

```
enum
{
    AddrNotPresent = 0,
    AddrGroup = 1,
    Addr16Bit = 2,
    Addr64Bit = 3,
    AddrBroadcast = 15
};
```

具体发送步骤如下（以单播为例）：

- 首先，定义一个afAddrType_t类型的变量：**afAddrType_t SendDataAddr;**
- 然后，将其addrMode参数设置为Addr16Bit（发送方式）：**SendDataAddr.addrMode=(afAddrMode_t) Addr16Bit; SendDataAddr.addr.shortAddr=XXXX;** 其中XXXX代表目标节点的短地址，如协调器的短地址为0x0000.
- 最后，调用AF_DataRequest函数发送数据即可：**AF_DataRequest(&SendDataAddr,.....)**

# 一、任务、事件以及串口通信

## （一）事件的设置

**事件驱动，任务轮询：给任务添加事件的函数 osal_set_event(taskId, SYS_EVENT_MSG);**
参数说明：
taskID：任务的ID
SYS_EVENT_MSG：事件

## （二）事件与消息

SYS_EVENT_MSG是系统事件，也是协议栈已经定义好的系统事件。一个任务最多可以有16个事件（因为事件号是一个16bit的常量，一个位表示一个事件，在ZcomDef.h中定

义），SYS_EVENT_MSG已经占用了0x8000，故自定义的事件只能有15个。
事件的提取和清除可以用简单的位操作指令实现：
事件的提取：events & SYS_EVENT_MSG 与
事件的清除：events ^ SYS_EVENT_MSG 异或
系统事件包括了各种系统消息（message），一个事件可以包括255个消息（系统时间的消息号是一个8bit常量，定义在ZcomDef.h中）。
**消息与事件的联系**
OSAL在后台维护了一个消息队列，每一个消息都会被放到这个消息队列中去，当任务接收到事件以后（调用osal_set_event方法），从消息队列中获取属于自己的信息，然后进行处理。

### （三）定时器的调用

协议栈允许任务调用定时器，定时器能用1ms的增量进行设置。
启用函数 **osal_start_timerEx()**
函数原型 **uin8 osal_start_timerEx(uint8 taskID, uint16 event_id, uint16 timeout_value);**
参数说明：
taskID：任务
event_id：事件
timeout_value：定时毫秒数

### （四）串口操作

串口发送接收数据的基本步骤：

- 初始化串口（设置波特率、中断等）：uint8 HalUARTOpen(uint8 port, halUARTCfg_t * config);
- 向缓冲区发送数据：uint16 HalUARTWrite(uint8 port, uint8* buf, uint16 len);
- 从接收缓冲区读取数据：uint16 HalUARTRead(uint8 port, uint8* buf, uint16 len);

## 二、点播、广播以及组播

### （一）点播

1、绑定方式的点播

```
GenericApp_DstAddr.addrMode = (afAddrMode_t) AddrNotPresent;
GenericApp_DstAddr.endPoint = 0;
GenericApp_DstAddr.shortAddr = 0;
```

2、指定地址的点播方式

```
GenericApp_DstAddr.addrMode = (afAddrMode_t) Addr16Bit;
GenericApp_DstAddr.shortAddr = XXXX; //XXXX表示要送到的地址
```

## （二）广播

```
GenericApp_DstAddr.addrMode = (afAddrMode_t) AddrBoradcast;
GenericApp_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
GenericApp_DstAddr.shortAddr = 0xFFFF; //0xFFFF表示网络上的所有设备，包括睡眠中的设备；0xFFFD除睡眠中的设备，网络中所有空闲且打开接收的设备；0xFFFC所有路由器，包括协调器。
```

## （三）组播

组播的实现需要以下步骤：

1、声明一个组对象 aps_Group_t  SampleApp_Group;

2、对aps_Group_t结构体赋值，示例如下：

```
SampleApp_Group.ID = 0x0003;//组ID
osal_memcpy(SampleApp_Group.name,"Group 3",7)
```

3、设定通信的目标地址

```
SampleApp_Flash_DstAddr.addrMode = (afAddrMode_t)afAddrGroup;
SampleApp_Flash_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
SampleApp_Flash_DstAddr.addr.shortAddr = SAMPLEAPP_FLASH_GROUP;
```

4、注册端点描述符

```
// Fill out the endpoint description.
SampleApp_epDesc.endPoint = SAMPLEAPP_ENDPOINT;
SampleApp_epDesc.task_id = &SampleApp_TaskID;
SampleApp_epDesc.simpleDesc = (SimpleDescriptionFormat_t
*)&SampleApp_SimpleDesc;
SampleApp_epDesc.latencyReq = noLatencyReqs;
// Register the endpoint description with the AF
afRegister( &SampleApp_epDesc );
```

5、在本任务里将端点加入到组中

```
aps_AddGroup(SAMPLEAPP_ENDPOINT, &SampleApp_Group);
```

6、按照组播地址向对方发送数据

```
if ( AF_DataRequest( &SampleApp_Periodic_DstAddr, &SampleApp_epDesc,
                     SAMPLEAPP_PERIODIC_CLUSTERID,
                     1,
                     (uint8*)&SampleAppPeriodicCounter,
                     &SampleApp_TransID,
                     AF_DISCV_ROUTE,
                     AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
{
}
else
{
    // Error occurred in request to send.
}
```

7、若要把一个谁被加入到组中的端点从组中移除，调用aps_RemoveGroup即可

```
aps_RemoveGroup(SAMPLEAPP_ENDPOINT, SAMPLEAPP_FLASH_GROUP);
```

# 三、通信信道以及PANID的设置

## （一）通信信道

Zigbee在3个频段定义了27个物理信道，其中在2.4GHz频段上定义了16个250kb/s信道，信道间隔为5MHz。
可以在f8wConfig.cfg里设置信道

```
/* Default channel isChannel 11 - 0x0B */
// Channels are definedin the following:
//        0     : 868 MHz     0x00000001
//        1 - 10 : 915 MHz     0x000007FE
//       11 - 26 : 2.4 GHz     0x07FFF800
//-DMAX_CHANNELS_868MHZ     0x00000001
//-DMAX_CHANNELS_915MHZ     0x000007FE
//-DMAX_CHANNELS_24GHZ      0x07FFF800
//-DDEFAULT_CHANLIST=0x04000000// 26 - 0x1A
//-DDEFAULT_CHANLIST=0x02000000// 25 - 0x19
//-DDEFAULT_CHANLIST=0x01000000// 24 - 0x18
//-DDEFAULT_CHANLIST=0x00800000// 23 - 0x17
//-DDEFAULT_CHANLIST=0x00400000// 22 - 0x16
//-DDEFAULT_CHANLIST=0x00200000// 21 - 0x15
//-DDEFAULT_CHANLIST=0x00100000// 20 - 0x14
//-DDEFAULT_CHANLIST=0x00080000// 19 - 0x13
//-DDEFAULT_CHANLIST=0x00040000// 18 - 0x12
//-DDEFAULT_CHANLIST=0x00020000// 17 - 0x11
//-DDEFAULT_CHANLIST=0x00010000// 16 - 0x10
//-DDEFAULT_CHANLIST=0x00008000// 15 - 0x0F
//-DDEFAULT_CHANLIST=0x00004000// 14 - 0x0E
//-DDEFAULT_CHANLIST=0x00002000// 13 - 0x0D
//-DDEFAULT_CHANLIST=0x00001000// 12 - 0x0C
```

-DDEFAULT_CHANLIST=0x00000800 // 11 - 0x0B　这里默认使用的是编号为11的信道

当建网过程开始后，网络层将请求MAC层对规定的信道或由物理层默认的有效信道进行能量检测扫描，以检测可能的干扰。网络层管理实体对能量扫描的结果以递增的方式排序，丢弃那些能量值超出可允许能量水平的信道，然后再由网络层管理实体执行一次主动扫描，结合检查PAN描述符，对剩下的信道选择一个合适的建立网络。

## （二）PANID

当一个环境中存在多个Zigbee网络时，16个信道可能就不够用了，如果两个网络设置在同一个默认信道，就有可能A的终端加到B的网络中去。解决这个问题的方法是：使用PANID给网络编号。

PANID范围是0X0001——0XFFFF；

可以在f6wConfig.cfg文件中配置PANID

```
/* Define the default PAN ID.
 *
 * Setting this to a value other than 0xFFFF causes
 * ZDO_COORD to use this value as its PAN ID and
 * Routers and end devices to join PAN with this ID
 */
-DZDAPP_CONFIG_PAN_ID=0xFFFF
```

设置这个值是一个非0XFFFF的值
协调器会使用这个值作为他的PANID
路由器和终端会加入到这个PANID中

如果这里设置为0XFFFF，那么协调器则随机产生一个值作为自己的PANID；路由器和终端设备则会在自己的默认信道上随机选择一个网路加入，加入之后协调器的PANID即为自己的PANID。

当一个协调器的PANID设置好以后，比如1234，周边的路由器、终端节点加入了网络1234，一切都正常运行。但是当协调器断电重启就不能回到原来的网络中去了。因为协调器重启时发现有一个1234网络在运行，所以，他的PANID自动加1了，成为了另一个网络，只有将所有路由器和终端节点都重启才能组成一个网络。