Zigbee 技术规范与协议栈分析*

李战明，刘 宝，骆东松

（兰州理工大学 电气工程与信息工程学院，甘肃 兰州，730050）

摘 要：以 Zigbee 2006 协议栈为对象，研究其技术规范，使用串口调试助手及数据分析仪解读其协议栈程序，详细解释 OSAL(操作系统)的工作流程及整个协议栈的实现过程，以达到提高 Zigbee 源代码的可读性和可操作性的目的。

关键词：技术规范；Zigbee 协议栈；串口调试助手；数据分析议；OSAL

中图分类号：TN915 **文献标识码**：B

Analysis of Zigbee technology specification and protocol stack

LI Zhan Ming, LIU Bao, LUO Dong Song

(College of Electrical and Information Engineering, Lanzhou Univ. of Tech., Lanzhou 730050, China)

Abstract: to improve the readability and operability of Zigbee source code, this paper studied the technology specification of Zigbee-2006 protocol stack, made use of serial debugging assistant and packet sniffer to unscramble the procedure, explained in detail the entire work flow of OSAL and the implementation process of the protocol stack.

Key words: technology specification; Zigbee protocol stack; serial debugging assistant; packet sniffer; OSAL

伴随无线传感器网络的迅猛发展，Zigbee 技术作为最近发展起来的一种短距离无线通信技术，以其低功耗、自组织、安全可靠、支持大量节点等优势，被业界认为是最有可能应用在工控场合的无线方式。到目前为止，节点已经应用于工业监控、智能家庭、安全医疗等多个领域，具有很大的发展空间。

Zigbee 协议栈(Z-STACK)作为 Zigbee 技术的核心，是开发 802.15.4/Zigbee 必须掌握的关键技术。协议栈发展至今已有四种版本(见表 1)，尽管实现功能越发完善，但是并未移植标准的操作系统统一任务调度，嵌套相当复杂，而显得源代码的可读性和可操作性较差，开发者在理解和实现协议的过程中仍会遇到很多困难。本文通过对最典型的、起到承上启下作用的 Zigbee-2006 协议栈的解读，对协议程序的运行过程提供一种准确的解释分析，降低开发者的阅读难度和工作量，为协议栈自身的发展、Zigbee 设备的开发及应用的推广提供强有力的技术支持。

* 基金项目：甘肃省科技攻关项目(项目编号：0804GKCA0461)

1 协议栈体系结构及规范

Zigbee 协议栈体系结构如图 1 所示[1]，协议栈的层与层之间通过服务接入点(SAP)进行通信。SAP 是某一特定层提供的服务与上层之间的接口。大多数层有两个接口：数据实体接口和管理实体接口。数据实体接口的目标是向上层提供所需的常规数据服务；管理实体接口的目标是向上层提供访问内部层参数、配置和管理数据的服务[2]。

1.1 物理层服务规范

物理层通过射频固件和硬件提供 MAC 层与物理无线信道之间的接口。从概念上说，物理层还应包括物理层管理实体(PLME)，以提供调用物理层管理功能的管理服务接口；同时 PLME 还负责维护物理层 PAN 信息库(PHY PIB)。物理层通过物理层数据服务接入点(PD-SAP)提供物理层数据服务；通过物理层管理实体服务接入点(PLME-SAP)提供物理层管理服务。

表1 Zigbee协议栈各版本间的区别

		2004	2006	2007	PRO
冲突避免	支持运行中冲突检测并采用新 RF 信道或 PAN ID			支持	支持
自动分配地址管理	设备地址按照一个分等级的、分布式计划自动分配	支持	支持	支持	
	设备地址随机自动分配				支持
组寻址	设备可以按组分配，并可被唯一寻址		支持	支持	支持
集中数据采集	多对一路由允许整个网络发现数据集合体				支持
	源路由允许数据集合体以经济的方式回应所有发送者				支持
安全机制	任何设备可以作为信任中心				支持
	在特定的信任中心决策下允许高层安全机制，并且需要应用层连接钥匙、各实体鉴定等等				支持
网络可测量性	网络范围受寻址法则的限制，支持几百个设备	支持	支持	支持	
	降低寻址法则对网络范围的限制，支持几千个设备				支持
信息包	小于 100 字节	支持	支持		
	大信息包，支持分裂和重组			支持	支持
标准调试	标准的启动程序和属性支持调试功能		支持	支持	支持
Mesh 网络	每一个设备都支持它的邻居的路径跟踪				支持
串库	支持 Zigbee 串库，作为一种协议栈工具，为开发者提供无价的资源		支持	支持	

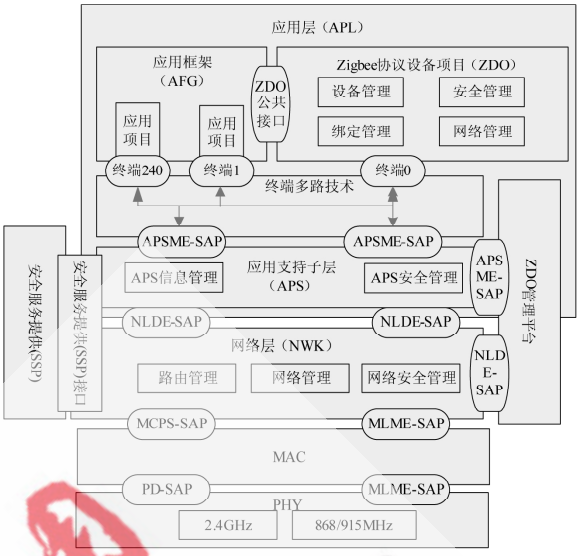

图1 Zigbee协议栈体系结构

1.2 MAC层服务规范

MAC层提供特定服务会聚子层(SSCS)和物理层之间的接口。从概念上说，MAC层还应包括MAC层管理实体(MLME)，以提供调用MAC层管理功能的管理服务接口；同时MLME还负责维护MAC PAN信息库(MAC PIB)。MAC层通过MAC公共部分子层(MCPS)的数据SAP(MCPS-SAP)提供MAC数据服务；通过MLME-SAP提供MAC管理服务。这两种服务通过物理层PD-SAP和PLME-SAP提供了SSCS和PHY之间的接口。除了这些外部接口外，MCPS和MLME之间还隐含了一个内部接口，用于MLME调用MAC数据服务。

1.3 应用层规范

Zigbee应用层包括APS子层、ZDO(包含ZDO管理平台)和厂商定义的应用对象。应用支持子层(APS)提供了网络层(NWK)和应用层(APL)之间的接口，功能是通过ZDO和厂商定义的应用对象都可以使用的一组服务来实现。数据和管理实体分别由APSDE-SAP和APSME-SAP提供。APSDE提供的数据传输服务在同一网络的两个或多个设备之间传输应用层PDU；APSME提供设备发现和绑定服务，并维护管理对象数据库——APS信息库(AIB)。

1.4 网络层规范

网络层应提供保证IEEE 802.15.4 MAC层正确工作的能力并为应用层提供合适的服务接口。数据和管理实体分别由NLDE-SAP和NLME-SAP提供。具体来说，NLDE提供的服务：一是在应用支持子层PDU基础上添加适当的协议头产生网络协议数据单元(NPDU)；二是根据路由拓扑，把NPDU发送到通信链路的目的地址设备或通信链路的下一跳。而NLME提供的服务包括配置新设备、创建新网络、设备请求加入/离开网络和Zigbee协调器或路由器请求设备离开网络、寻址、近邻发现、路由发现、接收控制等。网络层的数据和管理服务由MCPS-SAP和MLME-SAP提供了应用层和MAC子层之间的接口。除了这些外部接口，在NWK内部NLME和NLDE之间还有一个同隐含接口，允许NLME使用NWK数据服务。

2 协议栈程序分析

2.1 运行环境

软件环境：IAR 7.20、串口调试工具、数据分析仪以及各硬件驱动软件等。

硬件环境：PC(.NET 1.1架构，Windows 98以上，1个串口，1个USB接口)、CC2430 ZigBee开发板(至少包括一个网络协调器和一个终端设备，验证阶段可省略路由设备)等。

2.2 程序流程

将各开发板与PC正确连接，运行各软件，当程序烧至开发板后，启动即调用主函数ZSEG int main(void)。主函数的主要工作流程如图2所示。需要注意的是，Zigbee协议栈的精华在于操作系统OSAL的任务调度，因此，在进入主循环处理函数之前的准备工作中，操作系统的初始化尤为重要。osalTaskInit()、osalAddTasks()、osalInitTasks()三个函数的调用构成了协议栈的七大任务列表，其具体实现结果如图3。

进入主循环处理函数以后，始终周期扫描此任务列表，这7个任务由taskID和taskPriority来决定扫描和处理顺序。在循环扫描的过程中，数据的传输使用直接存取(DMA)控制器进行操作，可以减轻8051CPU核传送数据时的负担，实现CC2430在高效利用电源条件下得

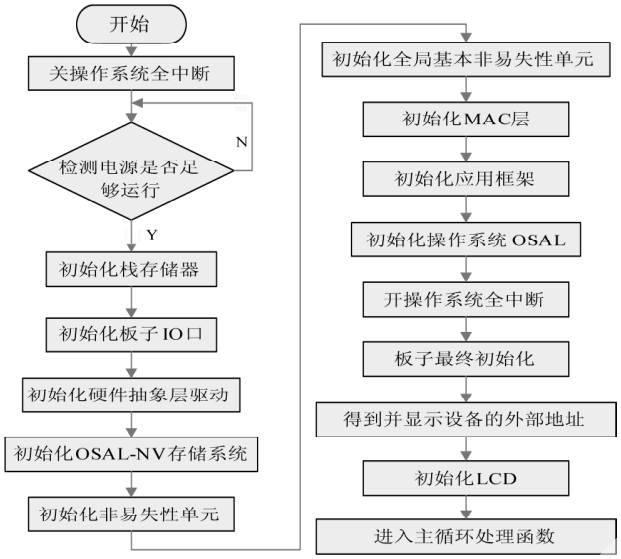

图 2 协议栈工作流程

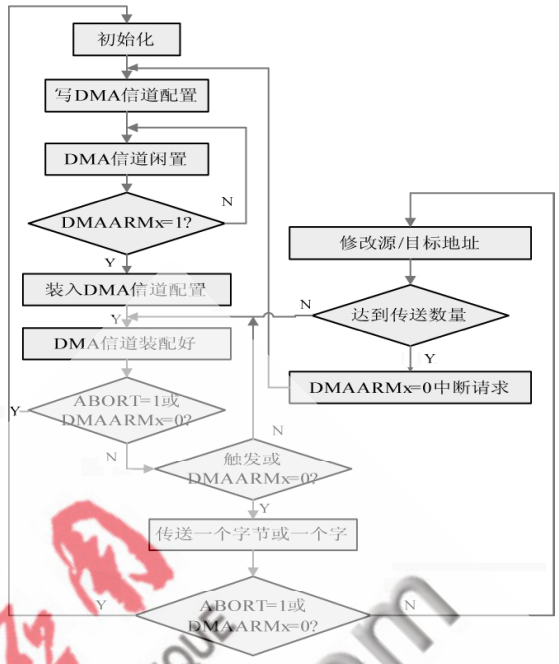

图 4 DMA 操作流程

Expression	Value	Location	Type
srchTask	0xE496	R3:R2	struct osalTaskRec*
next	0xE4B2	XData:0xE496	struct osalTaskRec*
next	0xE488	XData:0xE4B2	struct osalTaskRec*
next	0xE4A4	XData:0xE488	struct osalTaskRec*
next	0xE4C0	XData:0xE4A4	struct osalTaskRec*
next	0xE4CE	XData:0xE4C0	struct osalTaskRec*
next	0xE4DC	XData:0xE4CE	struct osalTaskRec*
next	0x0000	XData:0xE4DC	struct osalTaskRec*
next	<unavailable>	XData:0x0	struct osalTaskRec*
pfnInit	0xFFFFAD	XData:0x2	pTaskInitFn
pfnEventPr...	0xFFFFFF	XData:0x5	pTaskEventHandlerFn
taskID	' ' (0xFF)	XData:0x8	byte
taskPriority	' ' (0xFF)	XData:0x9	byte
events	65535	XData:0xA	uint16
pfnInit	SampleApp_Init (0x2D4C5)	XData:0xE4DE	pTaskInitFn
pfnEventProc...	SampleApp_ProcessEvent (...	XData:0xE4E1	pTaskEventHandlerFn
taskID	' ' (0x06)	XData:0xE4E4	byte
taskPriority	'2' (0x32)	XData:0xE4E5	byte
events	32768	XData:0xE4E6	uint16
pfnInit	ZDApp_Init (0x1EBAC)	XData:0xE4D0	pTaskInitFn
pfnEventProces...	ZDApp_event_loop (0x1EC41)	XData:0xE4D3	pTaskEventHandlerFn
taskID	' ' (0x05)	XData:0xE4D6	byte
taskPriority	'2' (0x32)	XData:0xE4D7	byte
events	0	XData:0xE4D8	uint16
pfnInit	APS_Init (0x2F38A)	XData:0xE4C2	pTaskInitFn
pfnEventProcessor	APS_event_loop (0x2F3AB)	XData:0xE4C5	pTaskEventHandlerFn
taskID	' ' (0x04)	XData:0xE4C8	byte
taskPriority	'2' (0x32)	XData:0xE4C9	byte
events	0	XData:0xE4CA	uint16
pfnInit	MT_TaskInit (0x19CF1)	XData:0xE4A6	pTaskInitFn
pfnEventProcessor	MT_ProcessEvent (0x19D26)	XData:0xE4A9	pTaskEventHandlerFn
taskID	' ' (0x02)	XData:0xE4AC	byte
taskPriority	'2' (0x32)	XData:0xE4AD	byte
events	0	XData:0xE4AE	uint16
pfnInit	Hal_Init (0x39590)	XData:0xE48A	pTaskInitFn
pfnEventProcessor	Hal_ProcessEvent (0x395E6)	XData:0xE48D	pTaskEventHandlerFn
taskID	' ' (0x00)	XData:0xE490	byte
taskPriority	'2' (0x32)	XData:0xE491	byte
events	0	XData:0xE492	uint16
pfnInit	nwk_init (0x13D66)	XData:0xE4B4	pTaskInitFn
pfnEventProcessor	nwk_event_loop (0x13DCD)	XData:0xE4B7	pTaskEventHandlerFn
taskID	' ' (0x03)	XData:0xE4BA	byte
taskPriority	'? (0x02)	XData:0xE4BB	byte
events	0	XData:0xE4BC	uint16
pfnInit	macTaskInit (0x2D351)	XData:0xE498	pTaskInitFn
pfnEventProcessor	macEventLoop (0x2D35D)	XData:0xE49B	pTaskEventHandlerFn
taskID	' ' (0x01)	XData:0xE49E	byte
taskPriority	'? (0xE6)	XData:0xE49F	byte
events	0	XData:0xE4A0	uint16

图 3 Zigbee 协议栈 OSAL 任务列表

高性能，其操作流程如图 4 所示。

作为协调器，如果程序使用了串口调试助手，则 DMA 将上位机的数据按照一个字节波特率加一个字节数据的形式组装到 cfg->rxBuf 中供其他函数调用，并且通过回调函数 SPIMgr_ProcessZToolData (uint8 port, uint8 event)将任务的 ID 和强制事件发送到任务列表中，供主循环处理函数扫描；作为终端节点和路由设备，无法使用串口调试助手，则通过回调函数 SPIMgr_ProcessZAppData (uint8 port, uint8 event) 将任务的 ID 和强制事件发送到任务列表中。当扫描至参数 events=1，则进入相应层的处理程序进行任务 ID 和 events 的约定比对，完成相应的功能，具体流程如图 5 所示。

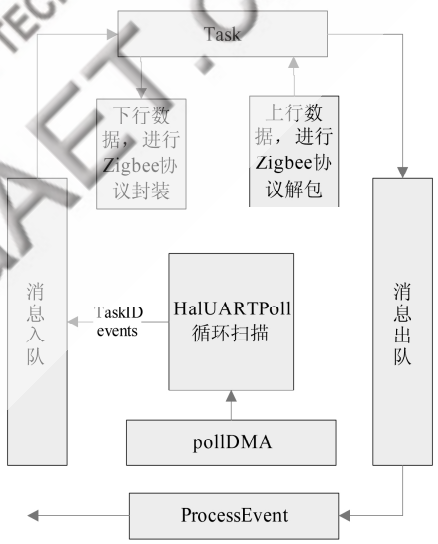

图 5 操作系统工作流程

2.3 组网

利用数据分析仪[3]记录监控协调器与终端设备的组网过程如图 6 所示。

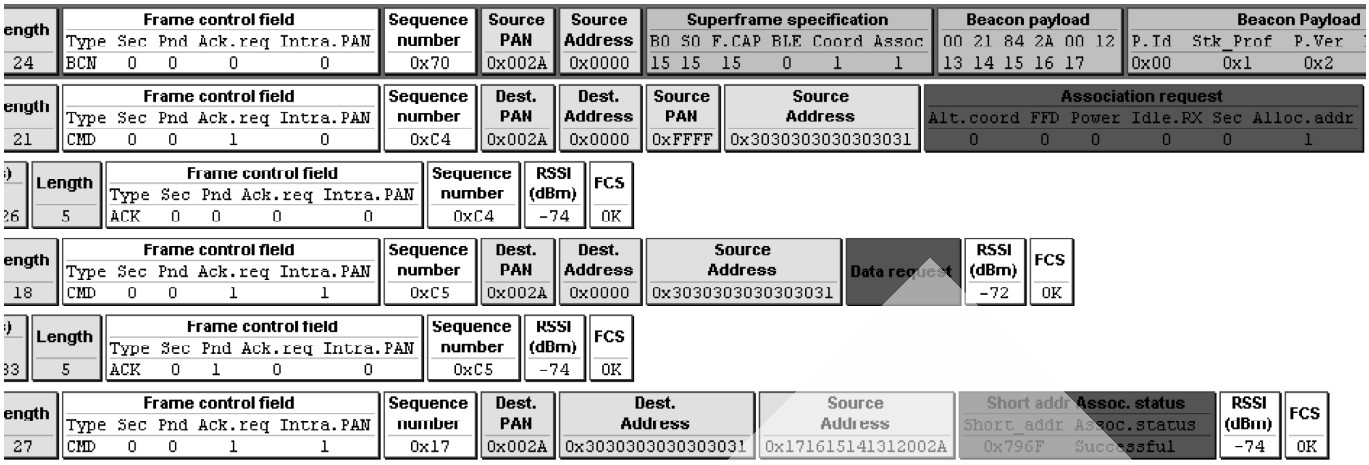

图 6 协调器与终端设备关联过程

首先，Zigbee 协调器上电以后，不断周期发送空的数据包，在允许的通道内搜索其他的 Zigbee 协调器，并基于每个允许通道中所检测到的通道能量及网络号，选择惟一的 16 位 PAN ID，建立自己的网络[4]。一旦一个新网络被建立，Zigbee 路由器与终端设备就可以加入到网络中了。而终端设备上电以后，重复发送信标请求，要求加入到最近的网络中。当协调器发现终端设备发出的信标请求，则响应一个超帧结构，用于设备间的同步，一旦同步成功，则实现图 5 中的关联过程，由终端设备向协调器发送关联请求，协调器同意则回应终端设备并自动分配 16 位的短地址，至此，两者组网成功。

网络形成后，可能会出现网络重叠及 PAN ID 冲突的现象。协调器可以初始化 PAN ID 冲突解决程序，改变一个协调器的 PAN ID 与信道，同时相应修改其所有的子设备。通常，Zigbee 设备会将网络中其他节点信息存储在一个非易失性的存储空间——邻居表中。加电后，若子设备曾加入过网络，则该设备会执行孤儿通知程序来锁定先前加入的网络。接收到孤儿通知的设备检查它的邻居表，并确定设备是否是它的子，若是，设备会通知子设备它在网络中的位置，否则子设备将作为一个新设备来加入网络。而后，该子设备将产生一个潜在双亲表，并尽量以合适的深度加入到现存的网络中[5]。

通常，设备检测通道能量所花费的时间与每个通道可利用的网络可通过 ScanDuration 扫描持续参数来确定，一般设备要花费 1 min 的时间来执行一个扫描请求，对于 Zigbee 路由器与终端设备来说，只需要执行一次扫描即可确定加入的网络。而协调器则需要扫描两次，一次采样通道能量，另一次则用于确定存在的网络。

限于篇幅，本文没有列出各种帧结构以及消息的处理过程，对于 Zigbee 协议的具体应用即是对数据包的封装与分解，这些分析、学习对于编写上位机软件，开发通过 Zigbee 协议与上位机进行交互的轻量级的现场设备，都是有意义的。已经据此以 C++ 开发出用户自定义的上位机程序，实现对传感器数据的监视和现场采集、数据库入库、趋势图等功能。

参考文献

[1] ZigBee Alliance. ZigBee Document 053474r13 [S]. December 1, 2006

[2] ZHENG JianLiang, Lee Myung. A Comprehensive Performance Study of IEEE 802.15.4[M]. IEEE Press Book,2004.

[3] Chipcon, Packet Sniffer for IEEE802. 15. 4 and Zigbee [S]. User Manual. Oslo, Norway, Oct. 2004.

[4] KINNEY P. Zigbee Technology ： Wireless Control that Simply Works [S] . Zigbee Alliance ， Oct . 2004.

[5] Zigbee Alliance. Network Specification (Draft Version 1.0) [S] . 2004.

（收稿日期：2008-11-20）

(上接第 44 页)

应用研究，2006，23(5):1-4.

[11]官荷卿，张文博，魏俊，等. 一种应用敏感的 Web 服务请求调度策略[J]，计算机学报，2006，29(7)：1189-1198

[12]单志广，林闯，肖人毅，等.Web QoS控制研究综述[J].计算机学报，2003, 27(2): 145-156.

[13]ZENG LangZhao, BENATALLAH B, DUMAS M. Quality Driven Web Services Composition [J].IEEE Transaction on Software Engineering, 2004, 30(5):311-327.

(收稿日期：2008-12-03)