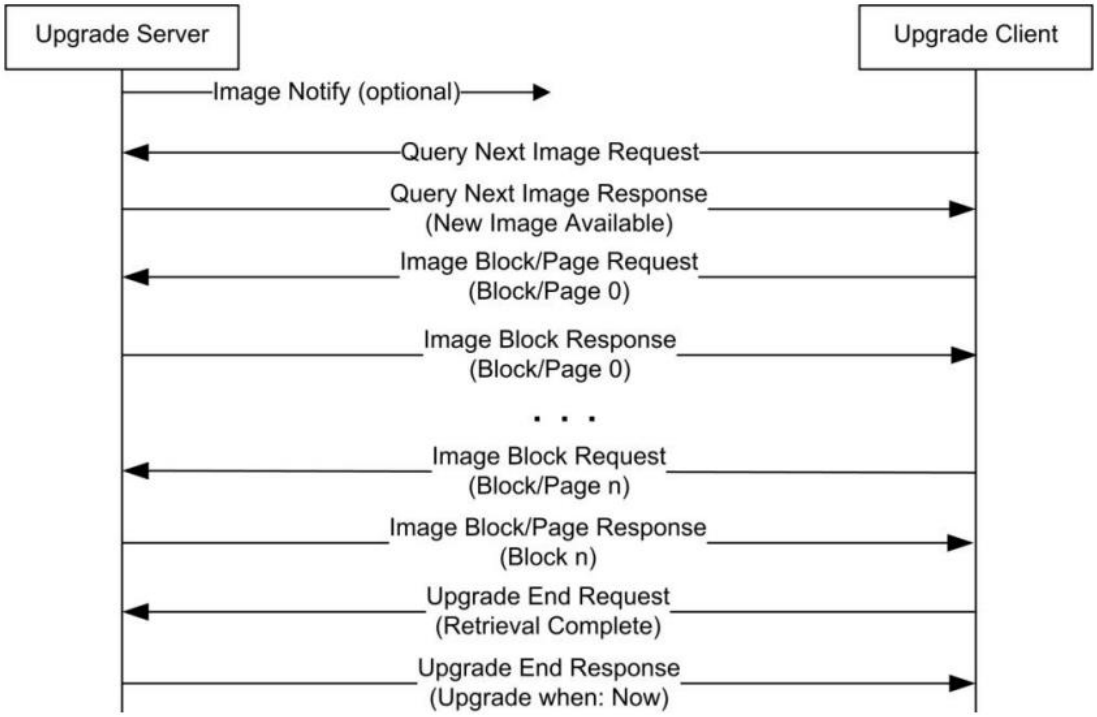
JN-5169 空间升级(OTA)详解

(shaozhong.liang@nxp.com)

1. ZigBee OTA 基本原理

OTA(Over The Air)通过无线方式进行设备固件升级。ZigBee 联盟在原有协议的框架上，提出了一种 OTA Upgrade(Cluster ID 0x0019)规范，其作为一个系统可选的功能模块。OTA 系统的结构示意图和服务器与客户端之间的数据交互过程如下。

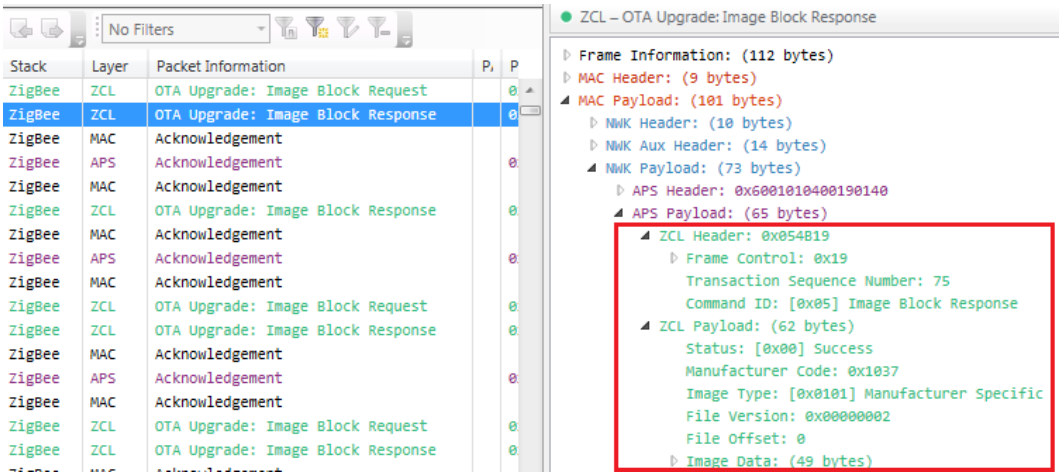


OTA 服务器与 OTA 客户端之间的数据交互过程如上图所示。首先 OTA 服务器通过单播或者广播方式向 OTA 客户端（节点）发送镜像公告(Image Notify)，指示新镜像已经准备好。收到镜像提示信息后，节点就向 OTA 服务器发送查询下一个镜像请求（Query Next Image Request），此请求信息包含了当前运行固件的版本信息。收到该请求后，OTA 服务器作出响应(Query Next Image Response)， 包括制造商 ID(Manufacturer Code)、镜像类型(Image Type)、新版本号(File Version)和文件大小(Image Size)。随后，OTA 客户端与 OTA 服务器通过二次握手机制，发送镜像块请求（Image Block Request）包含了文件版本号、镜像块偏移量和每次传送最大镜像块大小(OTA_MAX_MTU)，默认为 0x30，即为 48 字节。OTA 服务器发送的 Image Block Response，载荷记录格式与前者类似，并在最大镜像块大小字节后面附上 48 字节镜像块信息，从而完成一个镜像块传输周期。当 OTA 客户端收到镜像块数据后，把块数据写到第二存储区（客户端当前运行的镜像保存在第一存储区）。完成下载后，节点将对下载后的镜像进行 CRC 校验。最后，当节点需要更新时，把新镜像从第二存储区复制到第一存储区，新固件开始运行，从而完成了整个升级过程。

2. 镜像块请求

根据 ZigBee OTA 的规范，OTA 客户端向 OTA 服务器请求镜像的方式有两种，分别是镜像块请求与镜像页请求。前者是基于二次握手机制，镜像块请求包含了已下载镜像的偏移

量(File offset)与每次传输的镜像块大小等信息。当 OTA 服务器接收到该请求后，根据请求节点的短地址，镜像偏移量和镜像块大小等信息，通过串口向 OTA 应用控制台索取特定的镜像块。随后，OTA 服务器回复的镜像块响应包含了该镜像块数据。当节点收到镜像块数据后，把块数据写到第二存储区，随即更新下载镜像的偏移量，并准备下一轮的请求。



3. JN516x 如何切换运行 New Image

OTA 客户端将 New Image 逐一下载到 JN-516x 备份 Flash 区域。当全部的 New Image 下载完毕后，将对 New Image 进行必要的校验(CRC check, BIR header check)。如果 New Image 是完整合法的 Firmware，则切换到 New Image。切换的关键步骤是向当前 Flash 区域写 16 字节 0x00，擦除当前运行 Image 的 BIR 头信息，然后软复位重启。JN516x 复位重启后首先运行内嵌的 BootLoader 程序，由 BootLoader 检查内部 Flash，逐一搜索每个 Sector(32K/扇区)的开始 16 字节，如果是合法的 BIR 头信息，进行 Physical to Logical Flash Remapping，并运行 New Image，从而完成 New Image 切换过程。

JN516x/7x Flash Remapping

The JN516x and JN517x contain the ability to remap sectors of the internal Flash memory so that they appear at different locations within the memory map. The boot loader uses this functionality to ensure that, regardless of where an image is physically located, it appears to start at the beginning of the address space allocated to the internal Flash memory.

The Flash remapping uses two registers. These registers store the logical remapping of the physical sectors and, by default, are:

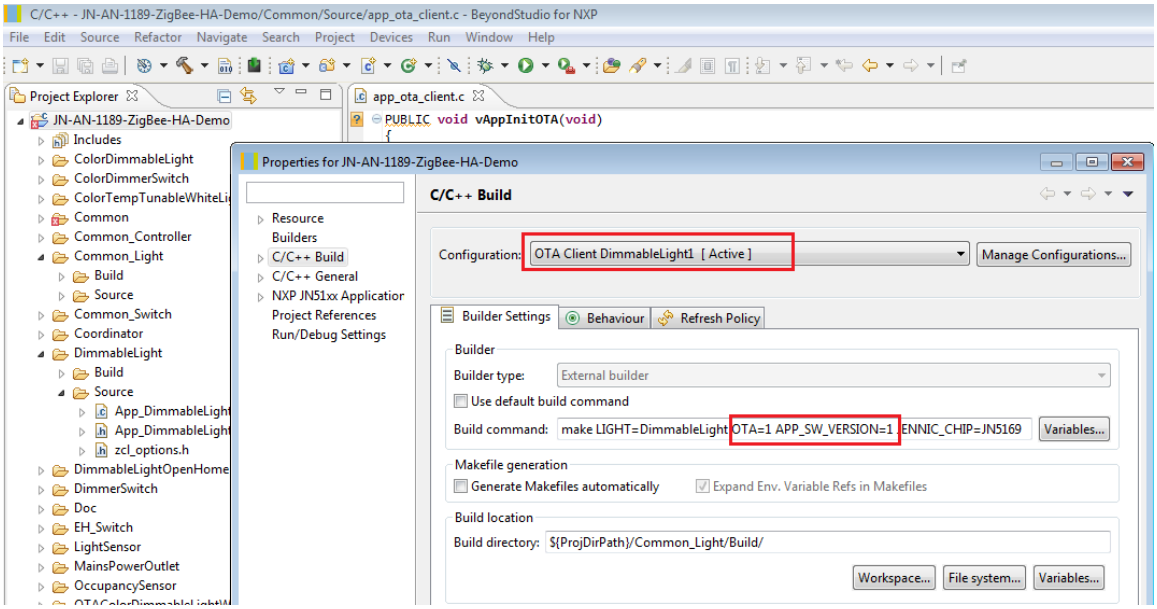
Register Name	Default Value	Comment
REG_SYS_FLASH_REMAP	0x76543210	Map physical sectors 0-7 to logical sectors 0-7
REG_SYS_FLASH_REMAP2	0xFEDCBA98	Map physical sectors 8-15 to logical sectors 8-15

Because of this Flash remapping, there is no need to modify the address to which the application is linked, since it will always start from 0x80000, and so the default linker file settings in the SDK remain valid. For example, in the case of JN5169, from the running application's viewpoint, it is always located in sectors 0 to 7 and the spare area is always sectors 8 to 15 (assuming the application requires 8 sectors of storage, i.e. it is between 224 and 256 Kbytes in size).

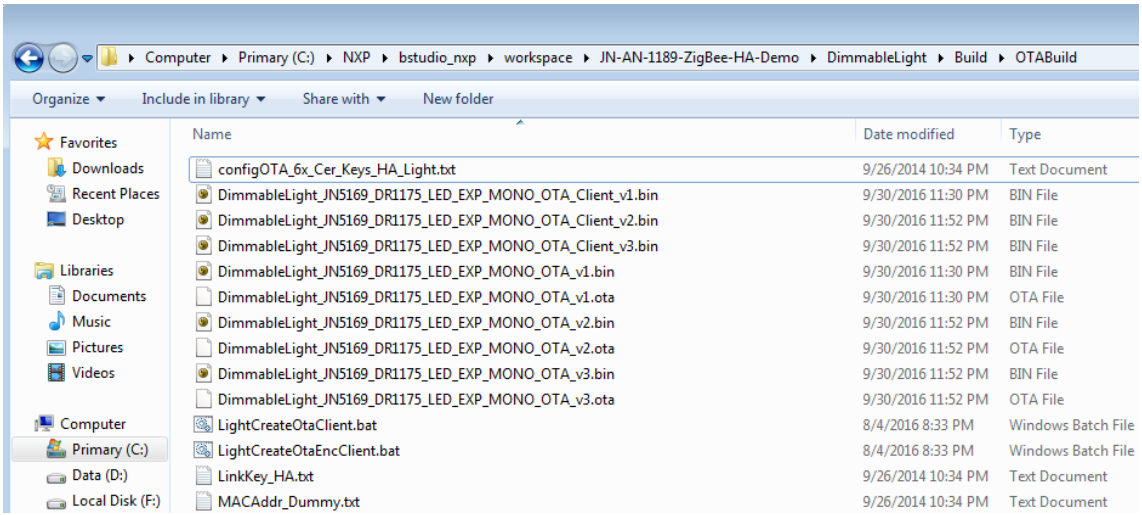
```
PUBLIC  teZCL_Status eOTA_ClientSwitchToNewImage(uint8 u8SourceEndPointId)
{
        ..........................

        if(!psOTA_Common->sOTACallBackMessage.sPersistedData.bIsNullImage)
        {
            vOtaFlashLockRead(psEndPointDefinition, psOTA_Common,u32Offset,OTA_FLS_MAGIC_NUMBER_LENGTH, au8MagicNumbers);

            if(bOtaIsImageValid(au8MagicNumbers))
            {
                /* persisted data changed send event to the  application to save it*/
                eOtaSetEventTypeAndGiveCallBack(psOTA_Common, E_CLD_OTA_INTERNAL_COMMAND_SAVE_CONTEXT,psEndPointDefiniti

                eOtaSetEventTypeAndGiveCallBack(psOTA_Common, E_CLD_OTA_INTERNAL_COMMAND_RESET_TO_UPGRADE,psEndPointDefi

                /* Invalidate current image and validate upgrade image */
#if JENNIC_CHIP_FAMILY == JN514x
                vOTA_SetImageValidityFlag(psOTA_Common->sOTACallBackMessage.u8CurrentActiveImageLocation,psOTA_Common,FAL
#endif

#if JENNIC_CHIP_FAMILY == JN516x
                /* Invalidate Internal Flash Header */
                bAHI_FlashInit(E_FL_CHIP_INTERNAL, NULL); /* Pass Internal Flash */
                /* Erase Internal Flash Header */
                DBG_vPrintf(TRACE_VERIF, "DELETE HEADER\n");
                bAHI_FullFlashProgram(0x00,16,au8Data);
#endif
                vOtaSwitchLoads();
```

4. 在 JN-5169 实现 ZigBee OTA 升级详细过程

NXP JN-5168/JN-5169 提供 512K Flash、32K SRAM，已经实现 OTA 空中升级功能。我们使用 JN-AN-1189 中的 DimmableLight 作为测试例子。选择”OTA Client DimmableLight”编译选项，其中编译选项 OTA=1 表示使能 OTA 功能， APP_SW_VERSION=1 表示生成第二版本升级.ota 文件。如果 APP_SW_VERSION=2 将会生成第三版本升级.ota 文件。

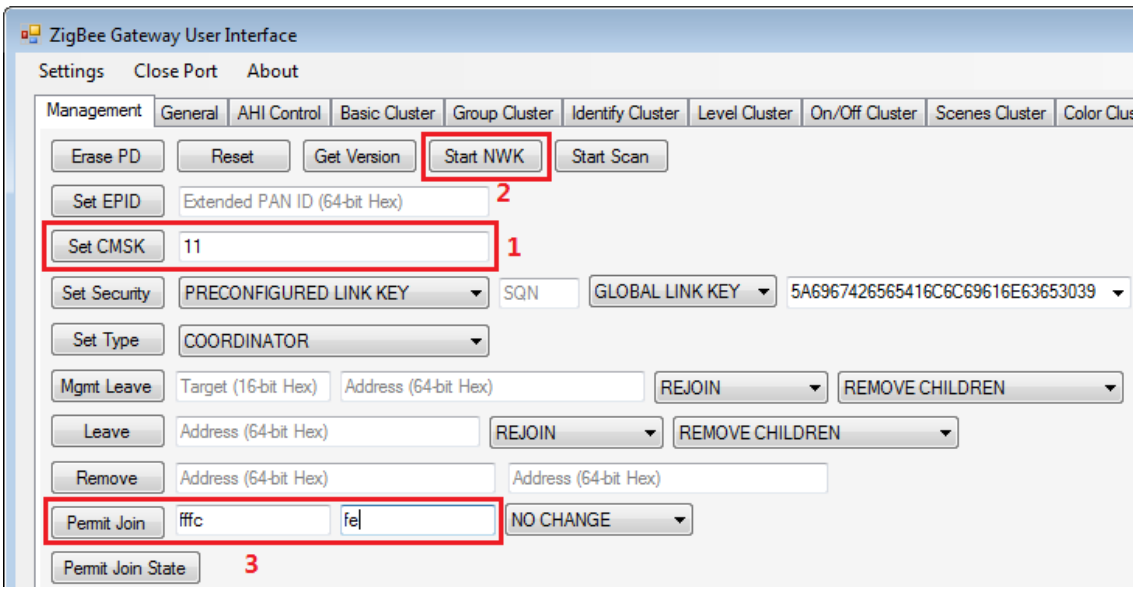


编译成功后，在” JN-AN-1189-ZigBee-HA-Demo\DimmableLight\Build\OTABuild”目录生成各个版本的升级文件。参考下面目录的各个文件。

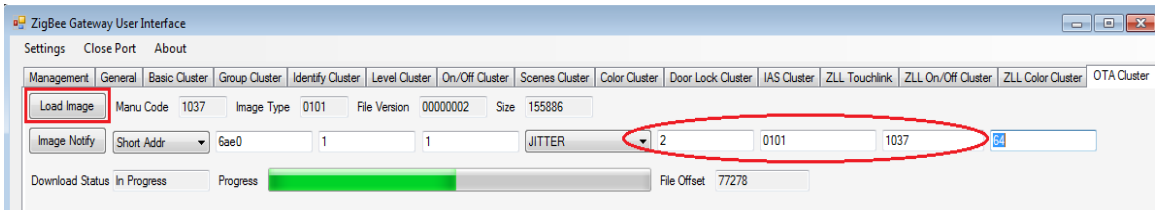


第一次将 DimmableLight_JN5169_DR1175_LED_EXP_MONO_OTA_Client_v1.bin 固件烧入 JN-5169 芯片。固件运行后将定时发出文件版本号为 1 的” Query Next Image Request”请求。

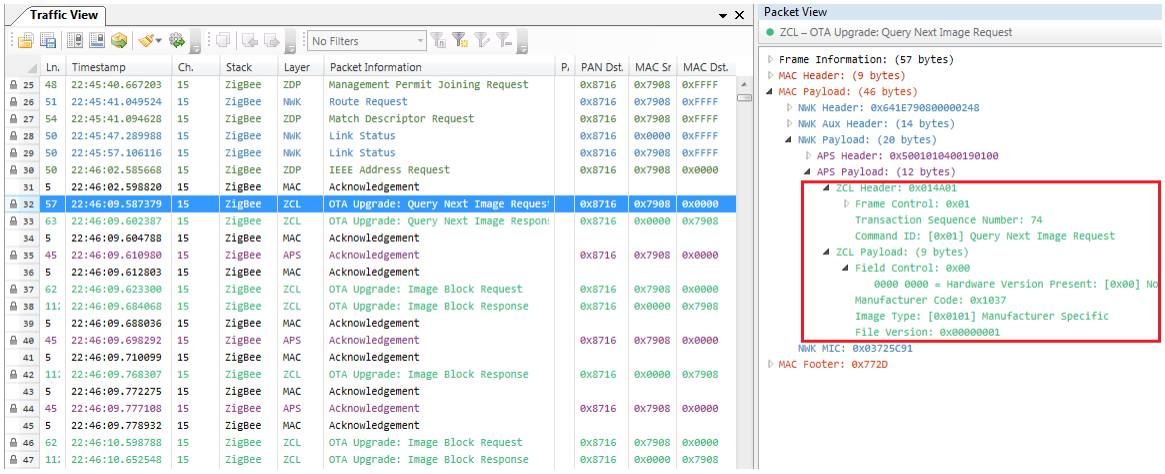
启动 PC 端的 ZigBee Gateway 测试软件 ZGWUI.exe，设置"Set CMSK" 、"Start NWK" 和 "Permit Join"功能后，建立 ZigBee 网络并允许新设备加入网络。



当升级设备入网后，在“OTA Cluster”中，点击”Load Image”按键，选择 DimmableLight_JN5168_DR1175_LED_EXP_MONO_OTA_v2.ota 文件。将”Manu Code”、”FileVersion”等内容作为“Image Notify“的参数。



设备将自动发送“Match Descriptor Request”匹配描述符查找 OTA 服务器。当找到符合条件的 OTA 服务器后，进行 ZigBee OTA 空中升级过程。



5. ZigBee OTA 注意事项

OTA 升级中每次传送最大镜像块大小(OTA_MAX_MTU)，主要受限于 IEEE802.15.4 MAC 包的总长度(127 字节), 除去各层(MAC/NWK/APS/ZCL)添加的头部信息外，基本上数据的长度不能超过 80 字节，建议使用默认 48 字节。

OTA 升级时根据 OTA 镜像头中的"Image Type"和“File Version”来确定该镜像是针对哪一种设备。例如上图中 0x0101 就是 DimmableLight 的"Image Type"。理论上只要升级镜像的"Image Type"不同，多个设备可以同时进行 OTA 空中升级。ZigBee 网关 Host 端实现 OTA 传输事务维护，当收到某个 OTA 镜像文件数据的请求时，Host 端回应相应 OTA 镜像的数据。我们的演示工具 ZGWUI.exe 在使用时只能支持用一个 OTA 镜像文件。

对于电池供电的 End-Device 进行 OTA 升级时考虑到省电，不能一直处于运行状态。用户可以在休眠唤醒后的入口 PWRM_CALLBACK(Wakeup)函数末尾在每次唤醒时调用 vRunAppOTAStateMachine 函数，请求 OTA 数据。另外休眠设备做 OTA 意义真的不大(除非客户有硬性要求)，一般的建议是对主电源供电的路由型设备保留 OTA 升级功能。这是因为一方面 End-Device 功能单一，功能开发定型后基本不会有什么修改；另一方面，进行 OTA 时间的过长(可能长达 10 天，因为只在唤醒后传输一次包)，频繁传输数据将会影响电池寿命。