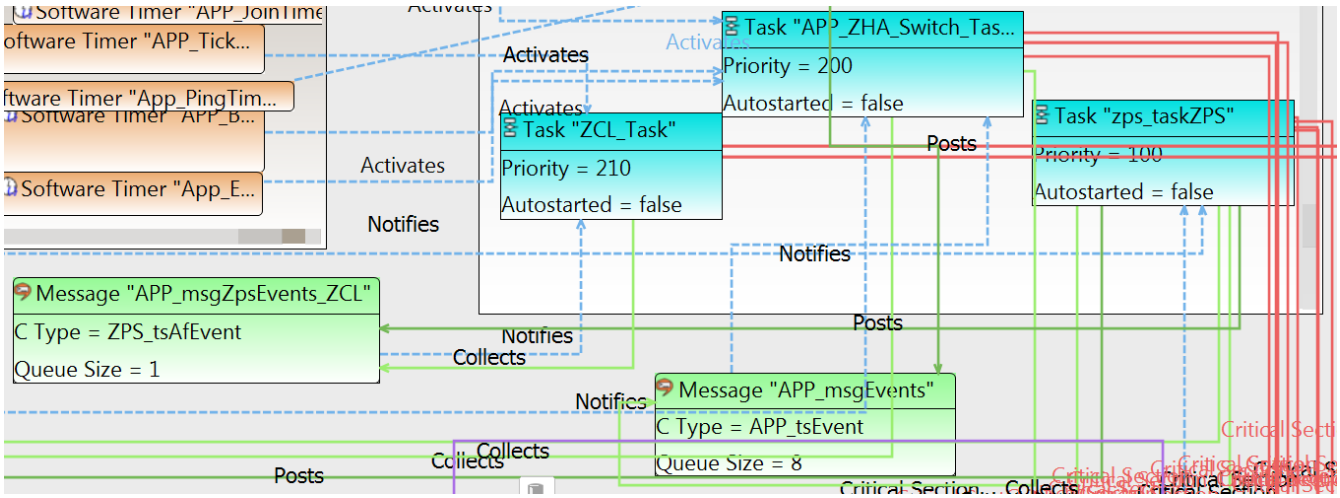
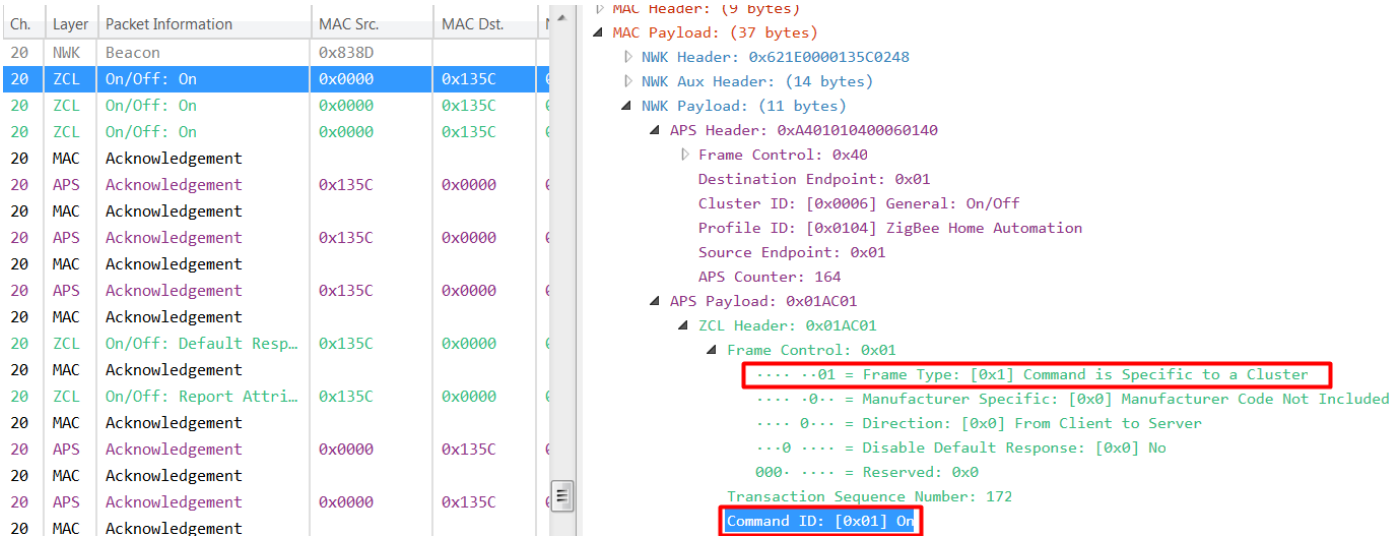
# NXP ZigBee 协议栈消息处理流程分析

(Shaozhong.Liang)

从 JenOS Configuration 图形可以看到，802.15.4 MAC 消息由任务 zps_taskZPS 处理完毕后，相关消息发送到消息队列 APP_msgZpsEvents_ZCL。任务 ZCL_Task 将会处理 ZigBee Cluster Library 的消息。大部分常见的 ZigBee 消息(如 OnOff, Read Attributes, Attribute Commands)将在这个任务进行处理。




我们以 JN-AN-1189-ZigBee-HA-Demo\DimmableLight 为例，分析收到 On/Off Cluster(0x0006)命令时的函数调用流程。




任务 ZCL_Task 主要处理 ZigBee Cluster Library 相关的消息。

```
OS_TASK(ZCL_Task)
{
    ZPS_tsAfEvent sStackEvent;
    tsZCL_CallBackEvent sCallBackEvent;
    sCallBackEvent.pZPSevent = &sStackEvent;

    /* If there is a stack event to process, pass it on to ZCL */
    sStackEvent.eType = ZPS_EVENT_NONE;
    if ( OS_eCollectMessage(APP_msgZpsEvents_ZCL, &sStackEvent) == OS_E_OK)
    {
        DBG_vPrintf(TRACE_ZCL, "\nZCL_Task event:%d",sStackEvent.eType);
```

```
            sCallBackEvent.eEventType = E_ZCL_CBET_ZIGBEE_EVENT;
            vZCL_EventHandler(&sCallBackEvent);
        }
}
vZCL_EventHandler →
   →vZCL_ZigbeeEventHandler
       →vZCL_HandleDataIndication
```

深入分析 vZCL_HandleDataIndication 函数的处理过程。函数 eZCL_SearchForClusterEntry 通过 u16ClusterId 找到对应的 Cluster Instance 实例，并调用 Cluster 注册的回调函数。这个回调函数在 vZCL_InitializeClusterInstance 初始化时作为参数保存在 Cluster Instance 的数据结构内存中。

```
PRIVATE void vZCL_HandleDataIndication(ZPS_tsAfEvent *pZPSevent)
{
    ……..
    // check the command is suitable for the endpoint - cluster, manufac Id, direction
    eCallbackReturn = eZCL_SearchForClusterEntry(pZPSevent->uEvent.sApsDataIndEvent.u8DstEndpoint,
                                                 pZPSevent->uEvent.sApsDataIndEvent.u16ClusterId,
                                                 !sZCL_HeaderParams.bDirection,
                                                 &psClusterInstance);
    ……
    switch (sZCL_HeaderParams.eFrameType)
    {
    case eFRAME_TYPE_COMMAND_ACTS_ACCROSS_ENTIRE_PROFILE:
        ……
        break;
    case eFRAME_TYPE_COMMAND_IS_SPECIFIC_TO_A_CLUSTER:
        {
            // check user custom callback
            if (psClusterInstance == NULL || psClusterInstance->pCustomcallCallBackFunction == NULL)
            {
                eZCL_SendDefaultResponse(pZPSevent, E_ZCL_CMDS_UNSUP_CLUSTER_COMMAND);
            }
            else
            {
                /* Input parameter checks & only interested in data indication events from here down */
                if((pZPSevent==NULL) || (psZCL_EndPointDefinition==NULL) || \
                    (pZPSevent->eType != ZPS_EVENT_APS_DATA_INDICATION))
                {
                    eCallbackReturn = E_ZCL_FAIL;
                }
                else
                {
                    eCallbackReturn = psClusterInstance->pCustomcallCallBackFunction(pZPSevent,
                                                                                     psZCL_EndPointDefinition,
                                                                                     psClusterInstance);
                }
        ……
}
```

在程序启动后调用代码入口 vAppMain 函数，初始化 Cluster Instance 实例，并注册回调函数的过程如下：

APP_vInitialiseNode

→eApp_HA_RegisterEndpoint

→eHA_RegisterDimmableLightEndPoint

→eCLD_OnOffCreateOnOff

```
PUBLIC teZCL_Status eHA_RegisterDimmableLightEndPoint(uint8 u8EndPointIdentifier,
                                                      tfpZCL_ZCLCallBackFunction cbCallBack,
                                                      tsHA_DimmableLightDevice *psDeviceInfo)
{
……..
    /* Mandatory server clusters */
#if (defined CLD_BASIC) && (defined BASIC_SERVER)
    /* Create an instance of a Basic cluster as a server */
    eCLD_BasicCreateBasic(&psDeviceInfo->sClusterInstance.sBasicServer,
                          TRUE,
                          &sCLD_Basic,
                          &psDeviceInfo->sBasicServerCluster,
                          &au8BasicClusterAttributeControlBits[0]);
#endif
……..
#if (defined CLD_ONOFF) && (defined ONOFF_SERVER)
    /* Create an instance of a On/Off cluster as a server */
    eCLD_OnOffCreateOnOff(&psDeviceInfo->sClusterInstance.sOnOffServer,
                          TRUE,
                          &sCLD_OnOff,
                          &psDeviceInfo->sOnOffServerCluster,
                          &au8OnOffServerAttributeControlBits[0],
                          &psDeviceInfo->sOnOffServerCustomDataStructure);
#endif
……..
}

PUBLIC   teZCL_Status eCLD_OnOffCreateOnOff(
                    tsZCL_ClusterInstance               *psClusterInstance,
                    bool_t                               bIsServer,
                    tsZCL_ClusterDefinition             *psClusterDefinition,
                    void                                *pvEndPointSharedStructPtr,
                    uint8                               *pu8AttributeControlBits,
                    tsCLD_OnOffCustomDataStructure      *psCustomDataStructure)
{
    ……
    // cluster data
    vZCL_InitializeClusterInstance(
            psClusterInstance,
            bIsServer,
            psClusterDefinition,
            pvEndPointSharedStructPtr,
            pu8AttributeControlBits,
```

```
            NULL,
            eCLD_OnOffCommandHandler);
    ……
}
```

在函数 eCLD_OnOffCommandHandler 的处理过程中，根据收到的 On,Off, Toggle 命令，分别调用对应的处理函数。

eCLD_OnOffCommandHandler
→eCLD_OnOffHandleOnCommand
→eCLD_OnOffHandleOffCommand
→eCLD_OnOffHandleToggleCommand

```
PUBLIC    teZCL_Status eCLD_OnOffCommandHandler(
                        ZPS_tsAfEvent               *pZPSevent,
                        tsZCL_EndPointDefinition    *psEndPointDefinition,
                        tsZCL_ClusterInstance       *psClusterInstance)
{
    ……..
    // SERVER
    switch(sZCL_HeaderParams.u8CommandIdentifier)
    {
    case(E_CLD_ONOFF_CMD_ON):
        eCLD_OnOffHandleOnCommand(pZPSevent,psEndPointDefinition,psClusterInstance,
                                  sZCL_HeaderParams.u8CommandIdentifier);
        break;

    case(E_CLD_ONOFF_CMD_OFF):
        eCLD_OnOffHandleOffCommand(pZPSevent,psEndPointDefinition,psClusterInstance,
                                  sZCL_HeaderParams.u8CommandIdentifier);
        break;

    case(E_CLD_ONOFF_CMD_TOGGLE):
        eCLD_OnOffHandleToggleCommand(pZPSevent,psEndPointDefinition,psClusterInstance,
                                  sZCL_HeaderParams.u8CommandIdentifier);
        break;
    ……..
    }
    ……..
    /* Generate a custom command event */
    ……..
    sOnOffCustomCallBackEvent.eEventType = E_ZCL_CBET_CLUSTER_CUSTOM;
    ……..
    // call callback
    psEndPointDefinition->pCallBackFunctions(&sOnOffCustomCallBackEvent);

    /* Generate a cluster update event */
    sOnOffCustomCallBackEvent.eEventType = E_ZCL_CBET_CLUSTER_UPDATE;
    psEndPointDefinition->pCallBackFunctions(&sOnOffCustomCallBackEvent);
    ……..
}
```

当我们考察 eCLD_OnOffHandleOnCommand 函数的具体处理过程时会发现在这个函数中，数据结构 psSharedStruct->bOnOff = 0x01 被修改，从而在逻辑上实现 On 的动作。物理状态改变则在后面的代码中。

```
PRIVATE    teZCL_Status eCLD_OnOffHandleOnCommand(
                            ZPS_tsAfEvent                *pZPSevent,
                            tsZCL_EndPointDefinition     *psEndPointDefinition,
                            tsZCL_ClusterInstance        *psClusterInstance,
                            uint8                         u8CommandIdentifier)
{
……
#if (defined CLD_LEVEL_CONTROL) && (defined LEVEL_CONTROL_SERVER)
    if(eCLD_LevelControlClusterIsPresent(psEndPointDefinition->u8EndPointNumber) == E_ZCL_SUCCESS)
    {
        /* If not already on, set it on */
        if((bool_t)psSharedStruct->bOnOff != TRUE)
        {
            eCLD_LevelControlSetOnOffState(psEndPointDefinition->u8EndPointNumber,
                                           TRUE,
                                           CLD_ONOFF_OFF_WITH_EFFECT_NONE);
        }
    }
    else
    {
        psSharedStruct->bOnOff = 0x01;
    }
#else
    psSharedStruct->bOnOff = 0x01;
#endif

    return eStatus;
}
```

函数 eCLD_OnOffHandleOnCommand 修改了 psSharedStruct->bOnOff 数据结构的状态值，改变了灯的逻辑状态。而真正改变 Light 灯的物理状态则在 Endpoint 的注册回调 APP_ZCL_cbEndpointCallback 函数。当改变数据结构状态值后，将会调用 Endpoint 的注册回调函数，事件类型分别是 E_ZCL_CBET_CLUSTER_CUSTOM 和 E_ZCL_CBET_CLUSTER_UPDATE。在这二个事件的处理过程中，将会调用灯的外设驱动程序 vWhiteLightSetLevels 函数，改变灯的物理状态。

```
PRIVATE void APP_ZCL_cbEndpointCallback(tsZCL_CallBackEvent *psEvent)
{
    ……
    switch (psEvent->eEventType)
    {
    ……
    case E_ZCL_CBET_CLUSTER_CUSTOM:
        switch(psEvent->uMessage.sClusterCustomMessage.u16ClusterId)
        {
            ……
            case GENERAL_CLUSTER_ID_ONOFF:
```

```
            {
        ……
                #elif (defined CLD_LEVEL_CONTROL) && !(defined ColorTempTunableWhiteLight)
                    /* level Control with on off */
                    vWhiteLightSetLevels(sLight.sOnOffServerCluster.bOnOff,
                                                        sLight.sLevelControlServerCluster.u8CurrentLevel);
                #else
                        /* must be on off with out level */
                #endif
            }
            break;
        ……
    case E_ZCL_CBET_CLUSTER_UPDATE:
        ……
        /* both level and on off present */
         vWhiteLightSetLevels(sLight.sOnOffServerCluster.bOnOff, sLight.sLevelControlServerCluster.u8CurrentLevel);
        ……
      break;

    default:
            DBG_vPrintf(TRACE_ZCL, "\nEP EVT: Invalid evt type 0x%x", (uint8)psEvent->eEventType);
            break;
    }
    ……
}
```

从 JN-AN-1189-ZigBee-HA-Demo\DimmableLight 处理流程我们可以分析得知，NXP ZigBee 协议栈已经实现了绝大部分处理代码。用户只需要在端点的注册回调函数 APP_ZCL_cbEndpointCallback 中修改并实现业务功能即可。整个 ZigBee 协议栈涉及的代码比较多，但真正需要用户修改、实现的用户代码其实并不多。往往只需要增加几百行代码即可完成一款新产品开发工作。