JN-5169 ZigBee 源代码代码阅读分析注释

下面代码以 NXP JN5169 ZigBee 参考设计 JN-AN-1189-ZigBee-HA-Demo 中的 OccupancySensor 应用代码为例，阅读、分析了 ZigBee 代码结构。为了便于理解代码的主要脉络，相关函数只保留关键的代码片段。请参照完整的源代码了解更多细节。

整个代码的入口函数

```
PUBLIC void vAppMain(void)
{
    ………
    /* start the RTOS */
    OS_vStart(vInitialiseApp, vUnclaimedInterrupt, vOSError);
    DBG_vPrintf(TRACE_START, "OS started\n");

    /* idle task commences here */
    while (TRUE)
    {
        vAHI_WatchdogRestart();

        PWRM_vManagePower();
    }
}
```

系统初始化函数调用顺序

```
vAppMain
      --> vInitialiseApp
         --> APP_vInitialiseNode
```

```
PUBLIC void APP_vInitialiseNode(void)
{
    ………
    PDM_eReadDataFromRecord(PDM_ID_APP_REMOTE_CONTROL,
                            &sDeviceDesc,
                            sizeof(tsDeviceDesc),
                             &u16ByteRead);

    ………
    /* Initialise ZBPro stack */
    ZPS_vAplSecSetInitialSecurityState(ZPS_ZDO_PRECONFIGURED_LINK_KEY,
          (uint8 *)&s_au8LnkKeyArray, 0x00, ZPS_APS_GLOBAL_LINK_KEY);

    vEZ_RestoreDefaultAIBChMask();
    /* Initialize ZBPro stack */
    ZPS_eAplAfInit();

    /*Set Save default channel mask as it is going to be manipulated */
    vEZ_SetDefaultAIBChMask();

    APP_ZCL_vInitialise();


    /* If the device state has been restored from flash, re-start the stack
     * and set the application running again.
     */

    if (sDeviceDesc.eNodeState == E_RUNNING)
    {
       app_vRestartNode();
    }
    else
    {
        app_vStartNodeFactoryNew();
    }
    /* Register callback that will handle ZDP (mgmt) leave requests */
    ZPS_vAplZdoRegisterZdoLeaveActionCallback(vHandleZdoLeaveRequest);
    ………
}
```

当恢复工厂设置后，需要重新入网

```
PRIVATE void app_vStartNodeFactoryNew(void)
{
    eEZ_UpdateEZState(E_EZ_START);

    /* Stay awake for joining */
    DBG_vPrintf(TRACE_SENSOR_NODE, "\nAPP Sensor Node: Factory New Start");
}
```

启动 3s 定时器，扫描信道
```
PUBLIC ZPS_teStatus eEZ_UpdateEZState(teEZ_State eEZState)
{

    sEZModeData.u8EZSetUpState = eEZState;

    if (sEZModeData.u8EZSetUpState == E_EZ_START )
    {
       /*If the State is start then start the Timers*/
       OS_eStartSWTimer(APP_JoinTimer,APP_TIME_MS(3000),NULL);
       OS_eStartSWTimer(APP_BackOffTimer,APP_TIME_MS(60000),NULL);

       DBG_vPrintf(TRUE, "\nDBG sEZModeData.u8EZSetUpState == E_EZ_START\n");
    }
    else if (sEZModeData.u8EZSetUpState == E_EZ_DEVICE_IN_NETWORK )
    {
       /*Restart The stack*/
       eStatus= ZPS_eAplZdoStartStack();

       DBG_vPrintf(TRUE, "\nDBG ZPS_eAplZdoStartStack\n");
    }
    return eStatus;
}
```

这个是 Sensor 的主处理任务，处理各种 Timer 和 ZigBee 协议栈传递给应用的消息
```
OS_TASK(APP_ZHA_Sensor_Task)
{

    if (OS_eCollectMessage(APP_msgEvents, &sAppEvent) == OS_E_OK)
    {
       vAppHandleAppEvent(sAppEvent);
    }

    else if(OS_eCollectMessage(APP_msgZpsEvents, &sAppStackEvent) == OS_E_OK)
    {

       if (TRUE == bCheckIsZCLEvent(sAppStackEvent))
       {
              vAPP_ZCL_SendStackEventToZCL(sAppStackEvent);
       }
       else
       {
              vAppHandleStackEvent(sAppStackEvent);
       }
    }
    else
    {
       DBG_vPrintf(TRACE_SENSOR_NODE, "\nAPP ZHA Sensor Task: Timer/Manual OS Event");
       vAppHandleStackEvent(sAppStackEvent);
    }
}
```

任务处理函数，根据当前状态，进入不同的处理函数
```
PRIVATE void vAppHandleStackEvent(ZPS_tsAfEvent sStackEvent)
{
    /* Handle events depending on node state */
    switch (sDeviceDesc.eNodeState)
    {
       case E_STARTUP:
              vAppHandleStartup(sStackEvent, E_EZ_JOIN);
              break;
       case E_REJOINING:
              vAppHandleStartup(sStackEvent, E_EZ_REJOIN);
              break;
       case E_RUNNING:
              vAppHandleRunning(sStackEvent);
              break;
       default:
              break;
    }
}
```

事件处理函数调用顺序
```
vAppHandleStackEvent
      --> vAppHandleStartup
```

```
         --> vHandleNetworkJoinAndRejoin
            --> vEZ_EZModeNWKJoinHandler
```

状态机处理函数，根据节点当前的状态调用相应的处理函数

```
void vEZ_EZModeNWKJoinHandler(ZPS_tsAfEvent *pZPSevent,teEZ_JoinAction eJoinAction)
{
    bool_t bBackOffTimerExpirredFlag;
    bBackOffTimerExpirredFlag = bBackOffTimerExpirred();

    if( bBackOffTimerExpirredFlag && (sEZModeData.u8EZSetUpState != E_EZ_BACKOFF) )
    {
        /*Just Back Off with a timer for back off timing don't do anything else*/
        sEZModeData.u8EZSetUpState = E_EZ_BACKOFF;
    }
    else
    {
        /*Run The state machine */
        switch (sEZModeData.u8EZSetUpState)
        {
            case E_EZ_START:
            {
                vSetUpStart();
                break;
            }
            case E_EZ_WAIT_DISCOVERY_TIMEOUT:
            {
                vWaitDiscovery(pZPSevent);
                break;
            }
            case E_EZ_JOINING_NETWORK:
            {
                vJoiningNetwork(pZPSevent);
                break;
            }
            case E_EZ_DEVICE_IN_NETWORK:
            {
                vDeviceInTheNetwork(pZPSevent,eJoinAction);
                break;
            }
            case E_EZ_BACKOFF:
            {
                vBackOff(bBackOffTimerExpirredFlag);
                break;
            }
            default :
                break;
        }
    }
}
```

发送 Beacon Request，寻找能够批准自己加入网络的父节点

```
PRIVATE void vSetUpStart(void)
{

    if(OS_eGetSWTimerStatus(APP_JoinTimer) == OS_E_SWTIMER_EXPIRED)
    {
        vEZ_ClearStateVariables(); /*Clear All the variables related to set up*/
        eEZ_SetChannel();          /*Set Appropriate Channel in APS channel Mask*/

        vAttemptDiscovery();
    }
}
```

网络发现过程，可以扫描指定某个信道，也可以扫描多个信道，尝试 3 次

```
PRIVATE void vAttemptDiscovery(void)
{
    uint8 eStatus;
    teEZ_State EZ_State;
    uint16 u16TimeOut;

    sEZModeData.u8ScanAttempts++;
    if(bRejoin)
    {
        vEZ_ReJoin();
        EZ_State = E_EZ_JOINING_NETWORK;
        u16TimeOut = JOINING_TIMEOUT_IN_MS;
    }
    else
    {
        ZPS_vNwkNibClearDiscoveryNT(ZPS_pvAplZdoGetNwkHandle());
        /*Apply Association Filter*/
        ZPS_bAppAddBeaconFilter( &sBeaconFilter);
        eStatus = ZPS_eAplZdoDiscoverNetworks( ZPS_psAplAibGetAib()->u32ApsChannelMask );
        EZ_State = E_EZ_WAIT_DISCOVERY_TIMEOUT;
        u16TimeOut = DISCOVERY_TIMEOUT_IN_MS;
    }
    vStartStopTimer(APP_JoinTimer,APP_TIME_MS(u16TimeOut),
                    &(sEZModeData.u8EZSetUpState),EZ_State);
}
```

处理 ZigBee 协议栈 NwK 上报的事件消息，发现网络，或者没有合适网络。同一信道可以有多个网络

```
PRIVATE void vWaitDiscovery(ZPS_tsAfEvent *pZPSevent)
{

    if(OS_eGetSWTimerStatus(APP_JoinTimer) == OS_E_SWTIMER_EXPIRED)
    {
        DBG_vPrintf(TRUE, "\nDBG vWaitDiscovery APP_JoinTimer\n");
        vReDiscover();
    }
    else
    {
        if (pZPSevent->eType != ZPS_EVENT_NONE)
        {
            switch(pZPSevent->eType)
            {
                case(ZPS_EVENT_NWK_DISCOVERY_COMPLETE):
                    vHandleDiscovery(pZPSevent);
                    break;

                case(ZPS_EVENT_NWK_JOINED_AS_ROUTER):     /* fall through*/
                case(ZPS_EVENT_NWK_JOINED_AS_ENDDEVICE):
                    vHandleJoinedNwk();
                    break;

                case(ZPS_EVENT_NWK_FAILED_TO_JOIN):
                    if (ZPS_psAplAibGetAib()->u64ApsUseExtendedPanid != 0)
                    {
                        ZPS_vNwkNibSetExtPanId(ZPS_pvAplZdoGetNwkHandle(),
                              ZPS_psAplAibGetAib()->u64ApsUseExtendedPanid);
                    }
                    vReDiscover();
                    break;

                default:
                    break;
            }
        }
    }
}
```

尝试 3 次失败后，返回 E_EZ_START 状态，重新开始扫描下一个信道

```
PRIVATE void vReDiscover(void)
{

    if (sEZModeData.u8ScanAttempts == MAX_DISCOVERY_ATTEMPT_PER_CHANNEL)
    {
       vStartStopTimer(APP_JoinTimer,APP_TIME_MS(DISCOVERY_TIMEOUT_IN_MS),
            &(sEZModeData.u8EZSetUpState),E_EZ_START);
    }
    else
    {
        vAttemptDiscovery();
    }
}
```

收到协调器的 Beacon 帧，发现合适网络，保存网络信息，并根据信道质量(LQI)排序，找到最合适的网络

```
PRIVATE void vHandleDiscovery(ZPS_tsAfEvent *pZPSevent)
{
    if((pZPSevent->uEvent.sNwkDiscoveryEvent.u8NetworkCount == 0) )
    {
       vStartStopTimer(APP_JoinTimer,APP_TIME_MS(DISCOVERY_TIMEOUT_IN_MS),
                  &(sEZModeData.u8EZSetUpState),E_EZ_WAIT_DISCOVERY_TIMEOUT);
    }
    else
    {
        vEZ_SortAndSaveNetworks(&(pZPSevent->uEvent.sNwkDiscoveryEvent));
        vEZ_JoinSavedNetwork();
    }
}
```

尝试加入某个网络，发送关联请求(Associate Request)

```
PRIVATE void vEZ_JoinSavedNetwork(void)
{
    uint8 u8Status;

    if( sEZModeData.u8DiscoveredNwkCount > 0)
    {
        while( sEZModeData.u8JoinIndex < sEZModeData.u8DiscoveredNwkCount  )
        {
            u8Status = ZPS_eAplZdoJoinNetwork(
               &(sEZModeData.asSavedNetworks[sEZModeData.u8JoinIndex]));

            sEZModeData.u8JoinIndex++;
            if(ZPS_E_SUCCESS != u8Status )
            {
                /* This should not happen */
            }
            else
            {
                vStartStopTimer( APP_JoinTimer, APP_TIME_MS(JOINING_TIMEOUT_IN_MS),
                     &(sEZModeData.u8EZSetUpState),E_EZ_JOINING_NETWORK );
                return ;
            }
        }
        if (sEZModeData.u8JoinIndex >= sEZModeData.u8DiscoveredNwkCount)
        {
            vStartStopTimer( APP_JoinTimer, APP_TIME_MS(RESTART_TIME_IN_MS),
                     &(sEZModeData.u8EZSetUpState),E_EZ_START );
        }
    }
    else
    {
        DBG_vPrintf(TRACE_EZMODE, "NO Open NWK\n\n\n");
        vStartStopTimer( APP_JoinTimer, APP_TIME_MS(RESTART_TIME_IN_MS),
                &(sEZModeData.u8EZSetUpState),E_EZ_START );
    }
}
```

收到 ZigBee 协议栈的消息，处理是否加入网络成功消息。

```
PRIVATE void vJoiningNetwork(ZPS_tsAfEvent *pZPSevent)
{

  if(OS_eGetSWTimerStatus(APP_JoinTimer) == OS_E_SWTIMER_EXPIRED)
  {
    DBG_vPrintf(TRUE, "\nDBG Going vHandleJoinFailed\n");
    vHandleJoinFailed();
  }
  else
  {
    if (pZPSevent->eType != ZPS_EVENT_NONE)
    {
        switch(pZPSevent->eType)
      {
        case(ZPS_EVENT_NWK_JOINED_AS_ROUTER):     /* fall through*/
        case(ZPS_EVENT_NWK_JOINED_AS_ENDDEVICE):
            vHandleJoinedNwk();
          break;

        case(ZPS_EVENT_NWK_FAILED_TO_JOIN):
            vHandleJoinFailed();
          break;

        default:
          break;
      }
    }
  }
}
```

节点入网成功，收到 Coordinator 的 Transport Key 消息，停止定时器，设置节点为入网状态

```
PRIVATE void vHandleJoinedNwk(void)
{
  DBG_vPrintf(TRUE, "\nDBG vHandleJoinedNwk...\n");

  OS_eStopSWTimer(APP_JoinTimer);
  OS_eStopSWTimer(APP_BackOffTimer);

  /*Rejoin is cleared now */
  bRejoin=FALSE;
  /* Now device part of network. Update state variable in persistent storage */
  sEZModeData.u8EZSetUpState = E_EZ_DEVICE_IN_NETWORK;

  /* Store EPID for future rejoin */
  ZPS_eAplAibSetApsUseExtendedPanId(ZPS_u64NwkNibGetEpid( ZPS_pvAplZdoGetNwkHandle()));

  ZPS_u16AplZdoLookupAddr(ZPS_psAplAibGetAib()->u64ApsTrustCenterAddress);
}
```

如果加入网络失败，则继续尝试下一个网络

```
PRIVATE void vHandleJoinFailed(void)
{
  DBG_vPrintf(TRUE, "\nvHandleJoinFailed  bRejoin =%d\n", bRejoin);

  if(bRejoin)
  {
    vStartStopTimer( APP_JoinTimer, APP_TIME_MS(1000),
         &(sEZModeData.u8EZSetUpState),E_EZ_START );
  }else
  {
    vEZ_JoinSavedNetwork();
  }
}
```

加入网络成功，设置节点为 E_RUNNING 状态，保存网络信息(PANID,Network_Key,etc)等信息到 EEPROM

```
PUBLIC void vHandleNetworkJoinAndRejoin( ZPS_tsAfEvent *pZPSevent, teEZ_JoinAction
                                        eJoinAction  )
{
    teEZ_State ezState;
    /*Call The EZ mode Handler passing the events*/
    vEZ_EZModeNWKJoinHandler(pZPSevent,eJoinAction);
    ezState = eEZ_GetJoinState();
    if(ezState == E_EZ_DEVICE_IN_NETWORK)
    {
      int i;
      void* pvNwk;
      ZPS_tsNwkNib *psNwkNib;

      pvNwk = ZPS_pvAplZdoGetNwkHandle();
      ZPS_psNwkNibGetHandle( pvNwk)->sPersist.u8CapabilityInformation &= 0x7f;

      OS_eStopSWTimer(APP_JoinTimer);
      vStopPollTimerTask();
      vStopBlinkTimer();
      sDeviceDesc.eNodeState = E_RUNNING;
      vHandleNewJoinEvent();

      PDM_eSaveRecordData(PDM_ID_APP_REMOTE_CONTROL,
                          &sDeviceDesc,
                          sizeof(tsDeviceDesc));
      ZPS_vSaveAllZpsRecords();

#if PRINT_NETWORK_KEY
    /* Get the nib handle so we can access the nib members */
    psNwkNib = ZPS_psNwkNibGetHandle(pvNwk);

    DBG_vPrintf(TRUE, "\nAPP NWK : \nPan ID = %x\nFunction Pan = %x\nExtended Pan = %d\nOutput Frame
Counter = %d\nChannel = %d",
                  psNwkNib->sPersist.u16VsPanId,
                  ZPS_u16NwkNibGetMacPanId(pvNwk),
                  (uint32)psNwkNib->sPersist.u64ExtPanId,
                  psNwkNib->sTbl.u32OutFC,
                  psNwkNib->sPersist.u8VsChannel);


    DBG_vPrintf(TRUE,"\nTransport Network Key : ");
    for(i = 0;i<16;i++)
    {
      DBG_vPrintf(TRUE, "%x:", psNwkNib->sTbl.psSecMatSet[0].au8Key[i]);
    }
#endif

    vAttemptToSleep();
 }
}
```