ZigBee 3.0：在 KW41Z 上添加端点（Endpoint）

原文：https://community.nxp.com/docs/DOC-340508

本文介绍了如何为 AN12061-MKW41Z-AN-Zigbee-3-0-Base-Device 应用指南中的 Router 程序添加额外的端点（endpoint）。

Router 程序的主端点是一个作为服务器（Server）的被开/关群集（On/OFF cluster）控制的灯。接下来的步骤描述了如何添加两个新的作为客户端（Client）的带开/关群集的端点。

请注意，这些更改只能使新添加的端点被识别，并没有加入从新端点读取输入或发送指令的功能。

Router/app_zcl_cfg.h

第一步是将新端点（Switch1，Switch2）添加到 ZCL 配置文件中。

```
/* Endpoints */
#define ROUTER_ZDO_ENDPOINT         (0)
#define ROUTER_APPLICATION_ENDPOINT (1)
#define ROUTER_SWITCH1_ENDPOINT     (2)
#define ROUTER_SWITCH2_ENDPOINT     (3)
```

Router/app_zps_cfg.h

第二步是更新 ZigBee 配置文件，将 AF_SIMPLE_DESCRIPTOR_TABLE_SIZE 从 2 增加到 4。设置为 4，是因为这是应用程序端点的数量（在我们的例子中是 3）+ 1（ZDO 的端点的数量）。

```
/****************************************************************
*************/
/* ZPS AF Layer Configuration Parameters */
/****************************************************************
*************/

#define AF_SIMPLE_DESCRIPTOR_TABLE_SIZE 4
```

Router/app_zcl_globals.c

第三步是更新 ZigBee 的群集（cluster）配置文件，将新端点（Switch1，Switch2）和它们的群集加到 Router 程序上。为此，需要将已配置的端点从 1 更改为 3，并且还需要将端点 Map 列表更改为：

```
PUBLIC uint8 u8MaxZpsConfigEp = 3;
PUBLIC uint8 au8EpMapPresent[3] =
```

```
{ ROUTER_APPLICATION_ENDPOINT,ROUTER_SWITCH1_ENDPOINT,ROUTER_SWITCH2
_ENDPOINT };
```

Switch1 和 Switch2 包含了 Basic 群集（0x0000）服务器和客户端，Identify 群集（0x0003）服务器和客户端，OnOff 群集（0x0006）客户端，Group 群集（0x0004）客户端。

这些群集被添加到输入群集列表中（服务器端）和输出群集列表中（在客户端）。但是只有在列表中被启用过的群集，才能通过 DiscFlag 被识别。因此，假设您需要添加一个 OnOff 群集客户端，则需要把群集 ID（0x0006，对应 OnOff）添加到输入群集列表中（服务器端）和输出群集列表中（客户端）。并且，因为这是一个客户端群集，需要把它在输出群集列表中设置为可识别。

```
PRIVATE const uint16 s_au16Endpoint2InputClusterList[5] =
{ HA_BASIC_CLUSTER_ID, HA_GROUPS_CLUSTER_ID, HA_IDENTIFY_CLUSTER_ID,\
 HA_ONOFF_CLUSTER_ID, HA_DEFAULT_CLUSTER_ID, };
PRIVATE const PDUM_thAPdu s_ahEndpoint2InputClusterAPdus[5] =
{ apduZCL, apduZCL, apduZCL, apduZCL, apduZCL, };
PRIVATE uint8 s_au8Endpoint2InputClusterDiscFlags[1] = { 0x05 };

PRIVATE const uint16 s_au16Endpoint2OutputClusterList[4] =
{ HA_BASIC_CLUSTER_ID, HA_GROUPS_CLUSTER_ID, HA_IDENTIFY_CLUSTER_ID,\
 HA_ONOFF_CLUSTER_ID, };
PRIVATE uint8 s_au8Endpoint2OutputClusterDiscFlags[1] = { 0x0f };

PRIVATE const uint16 s_au16Endpoint3InputClusterList[5] =
{ HA_BASIC_CLUSTER_ID, HA_GROUPS_CLUSTER_ID, HA_IDENTIFY_CLUSTER_ID,\
 HA_ONOFF_CLUSTER_ID, HA_DEFAULT_CLUSTER_ID, };
PRIVATE const PDUM_thAPdu s_ahEndpoint3InputClusterAPdus[5] =
{ apduZCL, apduZCL, apduZCL, apduZCL, apduZCL, };
PRIVATE uint8 s_au8Endpoint3InputClusterDiscFlags[1] = { 0x05 };

PRIVATE const uint16 s_au16Endpoint3OutputClusterList[4] =
{ HA_BASIC_CLUSTER_ID, HA_GROUPS_CLUSTER_ID, HA_IDENTIFY_CLUSTER_ID,\
 HA_ONOFF_CLUSTER_ID, };
PRIVATE uint8 s_au8Endpoint3OutputClusterDiscFlags[1] = { 0x0f };
```

现在请将这些新端点加入到 Simple Descriptor 结构体，并初始化该结构体（参照 zps_tsAplAfSimpleDescCont 和 ZPS_tsAplAfSimpleDescriptor 结构体声明，以了解如何正确填充各种参数）如下所示：

```
PUBLIC zps_tsAplAfSimpleDescCont
s_asSimpleDescConts[AF_SIMPLE_DESCRIPTOR_TABLE_SIZE] = {
 {
    {
```

```
        0x0000,
        0,
        0,
        0,
        84,
        84,
        s_au16Endpoint0InputClusterList,
        s_au16Endpoint0OutputClusterList,
        s_au8Endpoint0InputClusterDiscFlags,
        s_au8Endpoint0OutputClusterDiscFlags,
    },
    s_ahEndpoint0InputClusterAPdus,
    1
},
{
    {
        0x0104,
        0,
        1,
        1,
        5,
        4,
        s_au16Endpoint1InputClusterList,
        s_au16Endpoint1OutputClusterList,
        s_au8Endpoint1InputClusterDiscFlags,
        s_au8Endpoint1OutputClusterDiscFlags,
    },
    s_ahEndpoint1InputClusterAPdus,
    1
},
{
    {
        0x0104,
        0,
        1,
        2,
        5,
        4,
        s_au16Endpoint2InputClusterList,
        s_au16Endpoint2OutputClusterList,
        s_au8Endpoint2InputClusterDiscFlags,
        s_au8Endpoint2OutputClusterDiscFlags,
     },
     s_ahEndpoint2InputClusterAPdus,
```

```
    1
 },
 {
    {
       0x0104,
       0,
       1,
       3,
       5,
       4,
       s_au16Endpoint3InputClusterList,
       s_au16Endpoint3OutputClusterList,
       s_au8Endpoint3InputClusterDiscFlags,
       s_au8Endpoint3OutputClusterDiscFlags,
    },
    s_ahEndpoint3InputClusterAPdus,
    1
 },
};
```

Router/zcl_options.h

该文件用来设置 ZCL 所使用的选项。

端点的数量

端点的数量由 1 增加到 3：

```
/* Number of endpoints supported by this device */
#define ZCL_NUMBER_OF_ENDPOINTS                 3
```

启用客户端群集（Client Clusters）

启用新端点的客户端群集的功能：

```
/****************************************************************************/
/*                     Enable Cluster                                       */
/*                                                                          */
/* Add the following #define's to your zcl_options.h file to enable         */
/* cluster and their client or server instances                             */
/****************************************************************************/
#define CLD_BASIC
#define BASIC_SERVER
#define BASIC_CLIENT
```

```
#define CLD_IDENTIFY
#define IDENTIFY_SERVER
#define IDENTIFY_CLIENT

#define CLD_GROUPS
#define GROUPS_SERVER
#define GROUPS_CLIENT

#define CLD_ONOFF
#define ONOFF_SERVER
#define ONOFF_CLIENT
```

Router/app_zcl_task.c

基本设备数据结构

创建与新端点相关联的基本设备（Basic Device）的结构体来储存数据：

```
/****************************************************************************/
/***        Exported Variables                                            ***/
/****************************************************************************/
tsZHA_BaseDevice sBaseDevice;
tsZHA_BaseDevice sBaseDeviceSwitch1;
tsZHA_BaseDevice sBaseDeviceSwitch2;
```

注册基本设备端点——APP_ZCL_vlnitialise()

两个新的基本设备及其端点都应该在协议栈中被注册，以使其可用。

```
if (eZCL_Status != E_ZCL_SUCCESS)
{
                DBG_vPrintf(TRACE_ZCL, "Error:
eZHA_RegisterBaseDeviceEndPoint(Light): %02x\r\n", eZCL_Status);
}
/* Register Switch1 EndPoint */
eZCL_Status =   eZHA_RegisterBaseDeviceEndPoint(ROUTER_SWITCH1_ENDPOINT,
                                                                &APP_ZC
L_cbEndpointCallback,
                                                                &sBaseD
eviceSwitch1);
if (eZCL_Status != E_ZCL_SUCCESS)
```

```
{
            DBG_vPrintf(TRACE_ZCL, "Error:
eZHA_RegisterBaseDeviceEndPoint(Switch1): %02x\r\n", eZCL_Status);
}
/* Register Switch2 EndPoint */
eZCL_Status =  eZHA_RegisterBaseDeviceEndPoint(ROUTER_SWITCH2_ENDPOINT,
                                                &APP_ZC
L_cbEndpointCallback,
                                                &sBaseD
eviceSwitch2);
if (eZCL_Status != E_ZCL_SUCCESS)
{
            DBG_vPrintf(TRACE_ZCL, "Error:
eZHA_RegisterBaseDeviceEndPoint(Switch2): %02x\r\n", eZCL_Status);
}
```

恢复出厂设置功能——vHandleClusterCustomCommands()

当基本群集服务器（Basic Cluster Server）收到“恢复出厂设置”命令时，两个新的基本设备将通过重新注册来恢复出厂设置：

```
case GENERAL_CLUSTER_ID_BASIC:
{
     tsCLD_BasicCallBackMessage *psCallBackMessage =
(tsCLD_BasicCallBackMessage*)psEvent->uMessage.sClusterCustomMessage
.pvCustomData;
     if (psCallBackMessage->u8CommandId ==
E_CLD_BASIC_CMD_RESET_TO_FACTORY_DEFAULTS )
     {
            DBG_vPrintf(TRACE_ZCL, "Basic Factory Reset Received\n");
            FLib_MemSet(&sBaseDevice,0,sizeof(tsZHA_BaseDevice));
            APP_vZCL_DeviceSpecific_Init();
            eZHA_RegisterBaseDeviceEndPoint(ROUTER_APPLICATION_ENDPOINT,
                                                &APP_ZCL_cbEndp
ointCallback,
                                                &sBaseDevice);
            eZHA_RegisterBaseDeviceEndPoint(ROUTER_SWITCH1_ENDPOINT,
                                                &APP_ZCL_cbEndp
ointCallback,
                                                &sBaseDeviceSwi
tch1);
            eZHA_RegisterBaseDeviceEndPoint(ROUTER_SWITCH2_ENDPOINT,
                                                &APP_ZCL_cbEndp
ointCallback,
```

```
                                                    &sBaseDeviceSwi
tch2);
        }
}
break;
```

基础服务器群集（Basic Server Cluster）的数据初始化——APP_vZCL_DeviceSpecific_Init（）

初始化基础集群的默认属性值：

```
sBaseDevice.sOnOffServerCluster.bOnOff = FALSE;
FLib_MemCpy(sBaseDevice.sBasicServerCluster.au8ManufacturerName,
"NXP", CLD_BAS_MANUF_NAME_SIZE);
FLib_MemCpy(sBaseDevice.sBasicServerCluster.au8ModelIdentifier, "BDB-
Router", CLD_BAS_MODEL_ID_SIZE);
FLib_MemCpy(sBaseDevice.sBasicServerCluster.au8DateCode, "20150212",
CLD_BAS_DATE_SIZE);
FLib_MemCpy(sBaseDevice.sBasicServerCluster.au8SWBuildID, "1000-0001",
CLD_BAS_SW_BUILD_SIZE);

sBaseDeviceSwitch1.sOnOffServerCluster.bOnOff = FALSE;
FLib_MemCpy(sBaseDeviceSwitch1.sBasicServerCluster.au8ManufacturerNa
me, "NXP", CLD_BAS_MANUF_NAME_SIZE);
FLib_MemCpy(sBaseDeviceSwitch1.sBasicServerCluster.au8ModelIdentifie
r, "BDB-Sw1", CLD_BAS_MODEL_ID_SIZE);
FLib_MemCpy(sBaseDeviceSwitch1.sBasicServerCluster.au8DateCode,
"20170310", CLD_BAS_DATE_SIZE);
FLib_MemCpy(sBaseDeviceSwitch1.sBasicServerCluster.au8SWBuildID,
"1000-0001", CLD_BAS_SW_BUILD_SIZE);

sBaseDeviceSwitch2.sOnOffServerCluster.bOnOff = FALSE;
FLib_MemCpy(sBaseDeviceSwitch2.sBasicServerCluster.au8ManufacturerNa
me, "NXP", CLD_BAS_MANUF_NAME_SIZE);
FLib_MemCpy(sBaseDeviceSwitch2.sBasicServerCluster.au8ModelIdentifie
r, "BDB-Sw2", CLD_BAS_MODEL_ID_SIZE);
FLib_MemCpy(sBaseDeviceSwitch2.sBasicServerCluster.au8DateCode,
"20170310", CLD_BAS_DATE_SIZE);
FLib_MemCpy(sBaseDeviceSwitch2.sBasicServerCluster.au8SWBuildID,
"1000-0001", CLD_BAS_SW_BUILD_SIZE);
```

Router/app_zcl_task.h

基本设备的数据结构设置为可供其他模块使用：

```
/****************************************************************************/
/***        Exported Variables                                            ***/
/****************************************************************************/
extern tsZHA_BaseDevice sBaseDevice;
extern tsZHA_BaseDevice sBaseDeviceSwitch1;
extern tsZHA_BaseDevice sBaseDeviceSwitch2;
```

Router/app_router_node.c

启用 ZCL 事件处理程序 - vAppHandleAfEvent（）

发往两个新端点的数据消息将传递给 ZCL 进行处理：

```
if (psZpsAfEvent->u8EndPoint == ROUTER_APPLICATION_ENDPOINT
||  psZpsAfEvent->u8EndPoint == ROUTER_SWITCH1_ENDPOINT
||  psZpsAfEvent->u8EndPoint == ROUTER_SWITCH2_ENDPOINT)
{
      DBG_vPrintf(TRACE_APP, "Pass to ZCL\n");
      if ((psZpsAfEvent->sStackEvent.eType ==
ZPS_EVENT_APS_DATA_INDICATION) ||
          (psZpsAfEvent->sStackEvent.eType ==
ZPS_EVENT_APS_INTERPAN_DATA_INDICATION))
     {
           APP_ZCL_vEventHandler( &psZpsAfEvent->sStackEvent);
      }
}
```

原文链接：https://community.nxp.com/docs/DOC-340508