NXP ZigBee 如何增加自定义 Cluster

(shaozhong.liang)

基本概念

在 ZigBee 规范中有 Profile, Cluster，Attribute，Endpoint 等概念。Profile 用来规范 ZigBee 相关产品需要要满足那些要求。ZigBee 联盟则已经规范了一些 Profile，如 Home Automation(0x0104)，Smart Energy(0x0109)，Building Automation(0x0105)等。其中 Home Automation Profile 就规定了智能家居都要做什么，HA Profile ID 是 0x0104。

在一个 Profile 的规范下，一个 Cluster 实际上是一组属性和命令的集合，是设备之间的一个通信接口。Cluster 并不是只存在单独某一个设备之上，而是联系两个设备的纽带。因此，对于一个设备，有“Input Cluster”和“Output Cluster”之分。也可以描述为一个 Cluster 包括 Server 端和 Client 端。一个 Cluster 被定义好之后，它包含的命令是固定的。例如用于“控制器”和“开关”之间的一个 Cluster，包含一条命令“Toggle”，含义是“改变开关状态”。则该 Cluster 对于控制器而言是 Output Cluster（发出“Toggle”命令），对于开关而言是 Input Cluster（接收“Toggle”命令）。智能家居 HA 下的一个调光器，操作这个调光器就需要一些命令，比如变亮，变暗，关灯，开灯这些，另外，这个调光器也会有一些 Attribute 属性，比如当前的亮度，由亮变暗的时长。HA Profile 已经规定了调光器应该有哪些 Cluster，如 Color Control Cluster，Ballast Configuration Cluster 等。每个 Cluster 被分配一个唯一 16 位 Cluster ID，从 0xFC00 开始是厂家自定义 Cluster ID。

一个 Cluster 包含多个 Attribute。Attribute 定义了设备的物理特性或当前状态。例如开关的状态或温度计值等皆可称为属性。每个属性具有唯一的 Attribute ID 值。例如 ZHA Profile 的 Basic Cluster(0x0000)有下面 Attribute 属性：

Basic – Cluster id 0x0000 Attribute

Id#	Name	Type	Range	Man/Opt
0x0	ZCLVersion	Uint8	Type range	M
0x4	ManufacturerName	String	0-32 byte	O
0x5	ModelIdentifier	String	0-32 byte	O
0x6	DateCode	String	0-32 byte	O
0x7	PowerSource	8 bit enum	Type range	M

总结说来，Profile 规范了应该包括哪些 Cluster；一个 Cluster 会有一个 Cluster ID；在一个 Cluster 下有多个 Command 和多个 Attribute；在一个 Cluster 下面 Command 和 Attribute 的 ID 要唯一。Command 和 Attribute 在不同的 Cluster 下可以重复，不同的 Profile 下 Cluster 也可以重复。

下面是 ZHA Profile 下安防 IAS 设备支持的 Cluster 列表：

Cluster	Attributes	Commands
0x0000 (Basic)	ZCL Version (0x0000)	(I) Reset To Factory Defaults (0x00)
	Application Version (0x0001)	
	Stack Version (0x0002)	
	Hardware Version (0x0003)	
	Manufacturer Name (0x0004)	
	Model Identifier (0x0005)	
	Date Code (0x0006)	
	Power Source (0x0007)	
Power Configuration (0x0001)	Battery Voltage (0x0020)	/
	Battery Percentage Remaining (0x0021)	
Identify (0x0003)	Identify Time (0x0000)	(0) Identify Query Response (0x00)
		(I) Identify (0x00)
		(I) Identify Query (0x01)
IAS Zone (0x0500)	Zone State (0x0000)	(O) Zone Status Change Notification (0x00)
	Zone Type (0x0001)	(O) Zone Enroll Request (0x01)
	Zone Status (0x0002)	(I) Zone Enroll Response (0x00)
	IAS CIE Address (0x0010)	
	Zone ID (0x0011)	

增加自定义 Manufacture Specific Cluster

NXP ZigBee 协议栈本身包含了大量符合 ZigBee 联盟制定的协议规范的 Cluster 定义，能满足大部分应用，这些 Cluster 可以与其它符合 ZigBee 标准协议的厂商直接对接使用。但是针对不同公司，需要有自己的特色即产品差异化，就需要针对自己产品增加独特的 Manufacture Specific Cluster。NXP ZigBee 协议栈提供很好的 Cluster 扩展性，可以满足客户的定制化要求。以下介绍如何增加自定义 Cluster 的步骤。

例如某一厂家需要开发一款通过 ZigBee 控制厨房的燃气炉。由于在 ZHA Profile 中没有定义燃气炉 Cooker 的 Cluster，客户开发人员可以需要根据 Cooker 的功能特点和需求自定义 Manufacture Specific Cluster，Cluster ID 使用 0xFC00。下面是 Cooker Cluster 需要定义的 Attribute 属性。

Attribute Name	Attribute Type	Mandatory
Current Temperature	int16	Y
Target Temperature	int16	Y
Maximum Temperature	int16	Y
Minimum Temperature	int16	Y
Cooker Mode	enum	Y
Timer Started	bool	N
Number of seconds passed	uint16	N
Expiry in seconds	uint16	N
Maximum Expiry in seconds	uint16	N
Minimum Expiry in seconds	uint16	N

Data Types

```
E_ZCL_NULL
E_ZCL_BOOL
E_ZCL_UINT8
E_ZCL_UINT16
E_ZCL_UINT32
E_ZCL_UINT64
E_ZCL_INT8
E_ZCL_INT16
E_ZCL_INT32
E_ZCL_INT64
E ZCL ENUM8
```

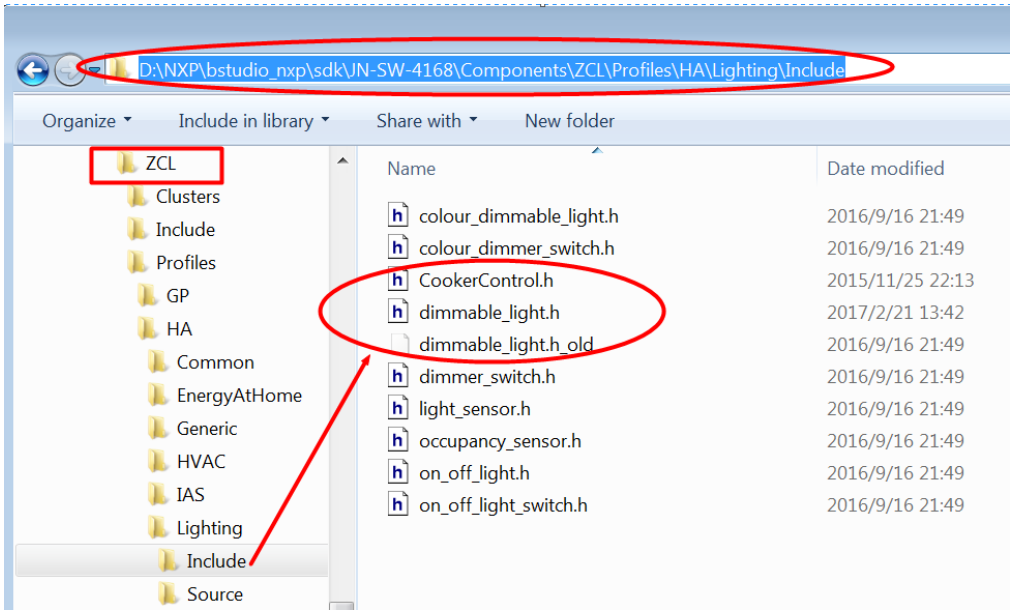
Attribute Flags

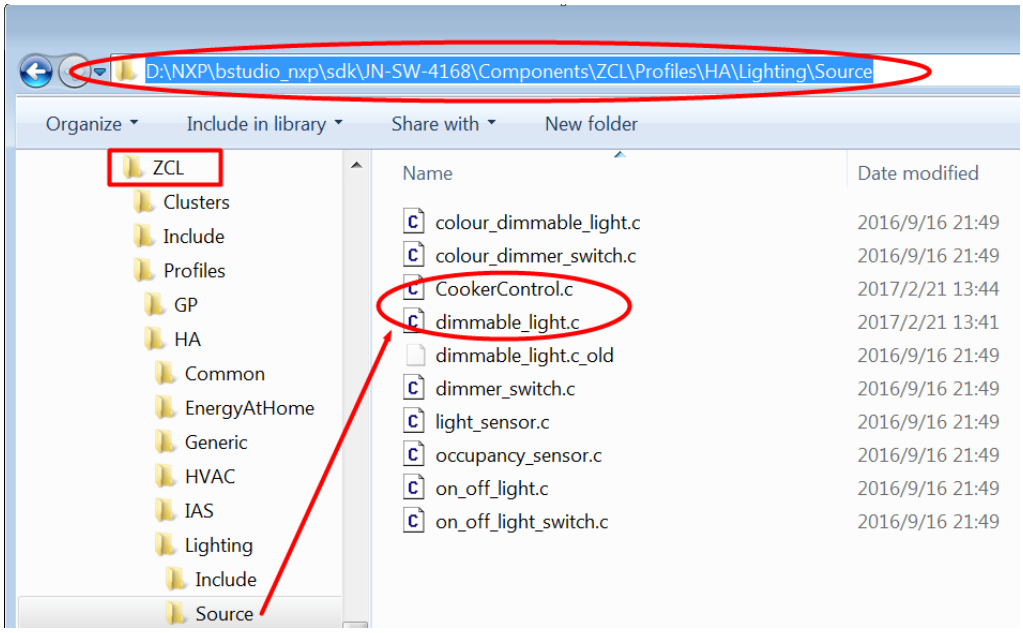
```
E_ZCL_AF_RD     // Readable
E_ZCL_AF_WR     // Writable
E_ZCL_AF_RP     // Report attribute
E_ZCL_AF_MS     // Manufacturer specific
E_ZCL_AF_CA     // Client attribute
E_ZCL_AF_SE     // Scenes extension
E_ZCL_AF_MR     // Mirrored attribute
/* Attribute Control Bits */
E_ZCL_ACF_RS    // Attribute Request Status */
E_ZCL_ACF_RP    // Is attribute reportable */
E_ZCL_ACF_MR    // Is attribute value is Mirrored attribute */
```

在 $sdk\JN-SW-4168\Components\ZCL\Profiles\HA\Lighting\Source\CookerControl.c 文件中定义 Cooker Cluster 的 Attribute 属性数组。

```
/* Cluster ID's */
#define COOKER_CONTROL_CLUSTER_ID_COOKER_CONTROL  0xFC00

/****************************************************************************/

/****************************************************************************/
/***        Local Variables                                               ***/
/****************************************************************************/
#ifdef COOKER_CONTROL_SERVER
const tsZCL_AttributeDefinition asCLD_CookerControlClusterAttributeDefinitions[] = {

        {E_CLD_COOKER_CLUSTER_ATTR_ID_CURRENT_TEMP,     (E_ZCL_AF_RD|E_ZCL_AF_RP|E_ZCL_AF_MS),       E_ZCL_INT16,    (uint32)(&((tsCLD_CookerControl*)(0))->i16CurrentTempValue),
        {E_CLD_COOKER_CLUSTER_ATTR_ID_TARGET_TEMP,      (E_ZCL_AF_RD|E_ZCL_AF_WR|E_ZCL_AF_MS),       E_ZCL_INT16,    (uint32)(&((tsCLD_CookerControl*)(0))->i16TargetTempValue),
        {E_CLD_COOKER_CLUSTER_ATTR_ID_MAX_TEMP,         (E_ZCL_AF_RD|E_ZCL_AF_MS),                   E_ZCL_INT16,    (uint32)(&((tsCLD_CookerControl*)(0))->i16MaxTempValue),
        {E_CLD_COOKER_CLUSTER_ATTR_ID_MIN_TEMP,         (E_ZCL_AF_RD|E_ZCL_AF_MS),                   E_ZCL_INT16,    (uint32)(&((tsCLD_CookerControl*)(0))->i16MinTempValue),
        {E_CLD_COOKER_CLUSTER_ATTR_ID_COOKER_MODE,      (E_ZCL_AF_RD|E_ZCL_AF_WR|E_ZCL_AF_MS),       E_ZCL_ENUM8,    (uint32)(&((tsCLD_CookerControl*)(0))->eCookerMode),

#ifdef CLD_CC_ATTR_TIMER_ENABLED
        {E_CLD_COOKER_CLUSTER_ATTR_ID_TIMER_STATUS,     (E_ZCL_AF_RD|E_ZCL_AF_RP|E_ZCL_AF_MS),       E_ZCL_BOOL,     (uint32)(&((tsCLD_CookerControl*)(0))->bTimerStarted),
        {E_CLD_COOKER_CLUSTER_ATTR_ID_SECONDS_PASSED,   (E_ZCL_AF_RD|E_ZCL_AF_MS),                   E_ZCL_UINT16,   (uint32)(&((tsCLD_CookerControl*)(0))->u16SecondsPassed),

#ifdef CLD_CC_ATTR_MAX_EXPIRY_TIMER
        {E_CLD_COOKER_CLUSTER_ATTR_ID_MAX_EXPIRY_IN_SECONDS,  (E_ZCL_AF_RD|E_ZCL_AF_WR|E_ZCL_AF_MS),  E_ZCL_UINT16,   (uint32)(&((tsCLD_CookerControl*)(0))->u16MaxExpiryInSecs),
#endif
#ifdef CLD_CC_ATTR_MIN_EXPIRY_TIMER
        {E_CLD_COOKER_CLUSTER_ATTR_ID_MIN_EXPIRY_IN_SECONDS,  (E_ZCL_AF_RD|E_ZCL_AF_WR|E_ZCL_AF_MS),  E_ZCL_UINT16,   (uint32)(&((tsCLD_CookerControl*)(0))->u16MinExpiryInSecs),
#endif
#endif
    };

    tsZCL_ClusterDefinition sCLD_CookerControl = {
            COOKER_CONTROL_CLUSTER_ID_COOKER_CONTROL,
            TRUE,
            E_ZCL_SECURITY_NETWORK,
            (sizeof(asCLD_CookerControlClusterAttributeDefinitions) / sizeof(tsZCL_AttributeDefinition)),
            (tsZCL_AttributeDefinition*)asCLD_CookerControlClusterAttributeDefinitions,
            NULL
    };
```

在$sdk\JN-SW-4168\Components\ZCL\Profiles\HA\Lighting\Include 和 Source 目录中分别修改 CookerControl.h、CookerControl.c、dimmable_light.h、dimmable_light.c 四个文件，加入 Cooker Cluster 的定义。





Add Cooker Cluster definition in the cluster header file:
$sdk\JN-SW-4168\Components\ZCL\Profiles\HA\Lighting\Include\dimmable_light.h

```
91 #ifdef CLD_COOKER_CONTROL
92 #include "CookerControl.h"
93 #endif
```

Add the cluster instance into the cluster instance structure inside device.

```
126 /* We are adding a cooker control cluster as an optional cluster */
127 #if (defined CLD_COOKER_CONTROL) && (defined COOKER_CONTROL_SERVER)
128     tsZCL_ClusterInstance sCookerControlServer;
129 #endif
```

Add the shared attribute structure into the main device structure

```
231     /* We are adding the shared attributes for the optional Cooker Control cluster */
232 #if (defined CLD_COOKER_CONTROL) && (defined COOKER_CONTROL_SERVER)
233     /* LevelControl Cluster - Server */
234     tsCLD_CookerControl sCookerControlServerCluster;
235 #endif
```

Add the CookerControlCreateCookerControl function into the DimmableLight register function:
$sdk\JN-SW-4168\Components\ZCL\Profiles\HA\Lighting\Source\dimmable_light.c

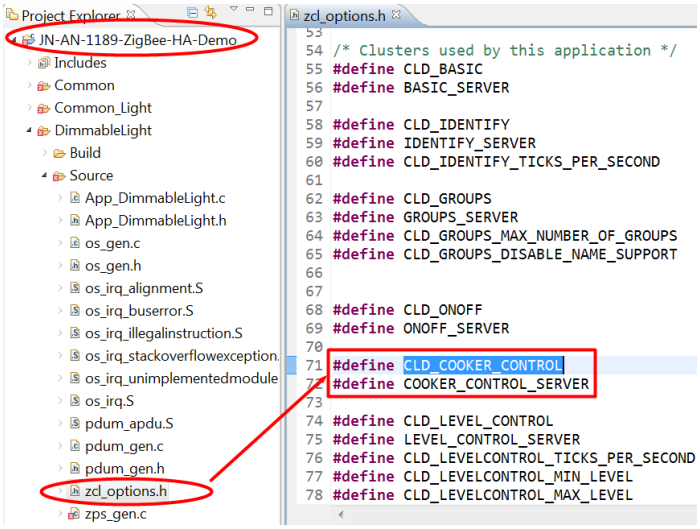
Include the cluster header file.

```
91 #ifdef CLD_COOKER_CONTROL
92 #include "CookerControl.h"
93 #endif
```

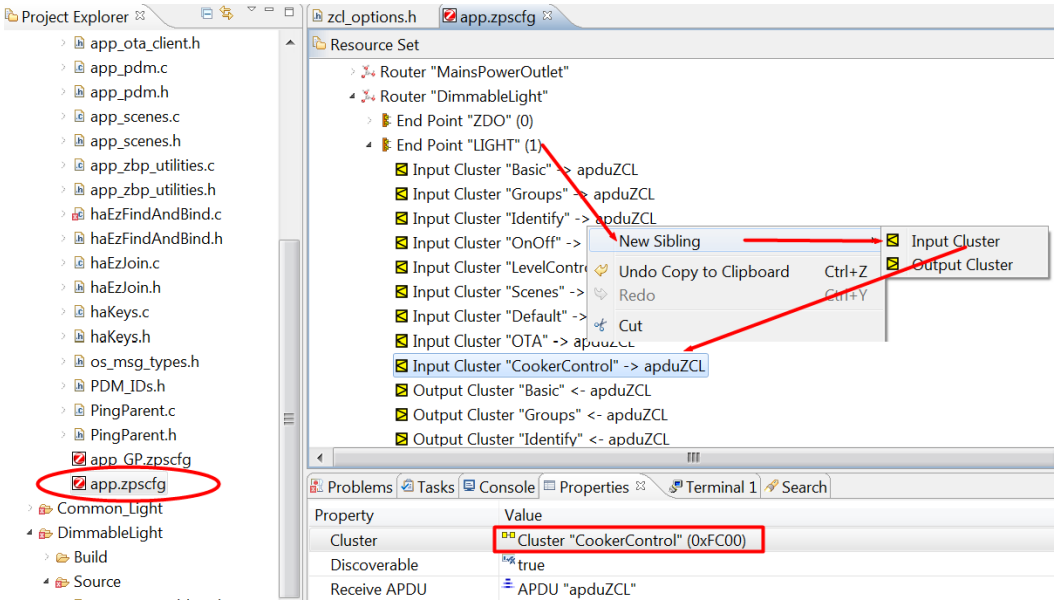
Call the ‘Create Cluster’ function which will initialize the cluster and any attributes to their default value

```
200     /* Initialising the optional Cooker Control Cluster */
201 #if (defined CLD_COOKER_CONTROL) && (defined COOKER_CONTROL_SERVER)
202     /* Create an instance of a Cooker Control cluster as a server */
203     eCLD_CookerControlCreateCookerControl(&psDeviceInfo->sClusterInstance.sCookerControlServer,
204                         TRUE,
205                         &sCLD_CookerControl,
206                         &psDeviceInfo->sCookerControlServerCluster,
207                         &au8CookerControlServerAttributeControlBits[0]);
208 #endif
```

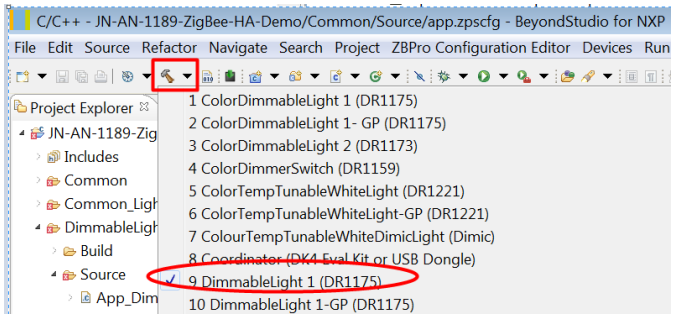
Add the cluster macros that are associated with the Cooker Control Cluster.



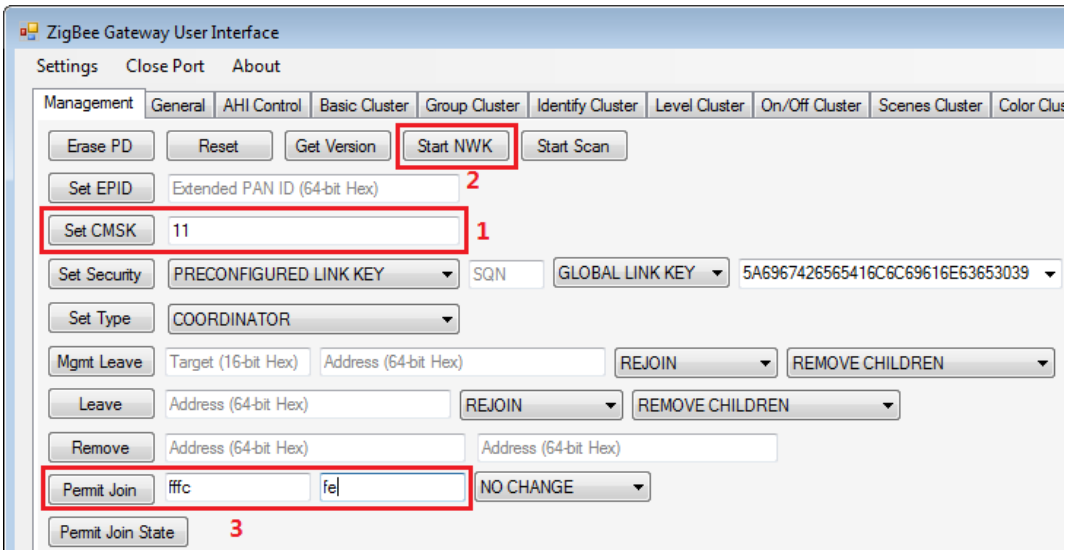
- Right click on “End Point ‘LIGHT’ (1)” in DimmableLight.
- Right click on the HOME_AUTOMATION profile.
- Select New Child -> Input Cluster.
- Select the undefined cluster.
- Select the Properties tab.
- In the Cluster tab, select the CookerControl cluster. Set Cluster ID as 0xFC00
- Leave Discoverable as “true”.
- Select the “apduZCL” for the receive APDU.



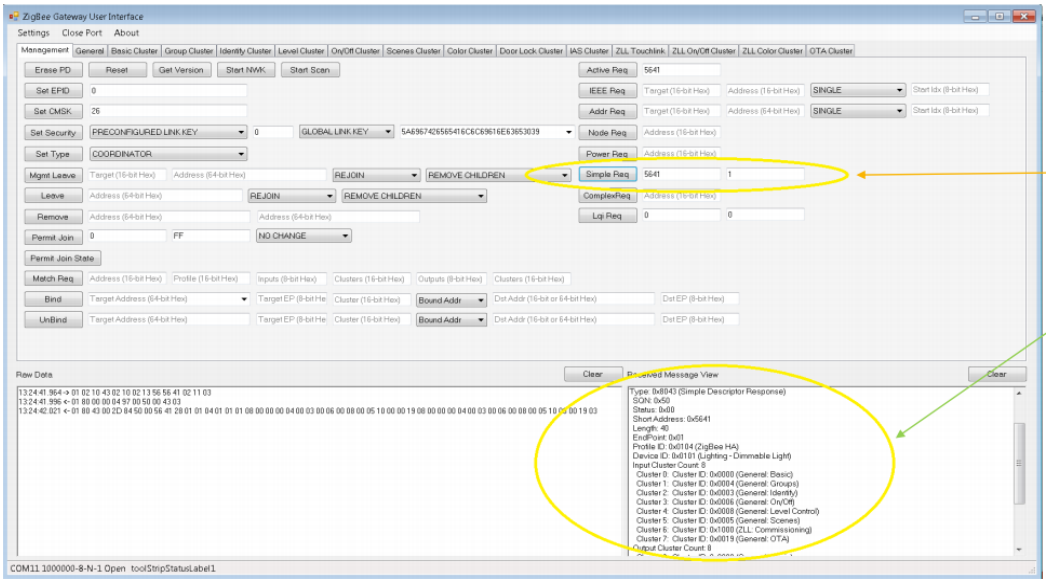
上述修改完毕后，重新编译整 DimmableLight 工程。生产的 bin 文件将会包含新增加的 Cooker Cluster 功能。



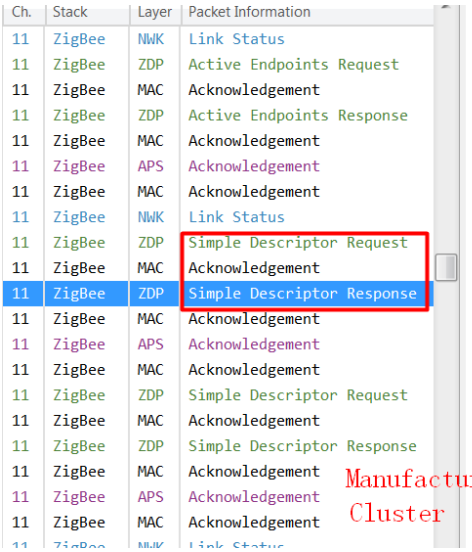
编译成功后将 DimmableLight_JN5169_DR1175_LED_EXP_MONO.bin 件烧入 JN-5169 芯片。然后启动 PC 端的 ZigBee Gateway 测试软件 ZGWUI.exe，设置"Set CMSK" 、"Start NWK" 和"Permit Join"功能后，建立 ZigBee 网络并允许新设备加入网络。



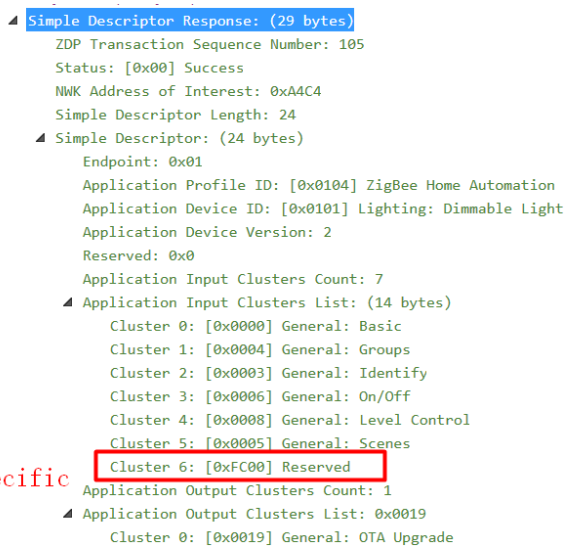
在 ZGWUI 通过发送 Simple Descriptor Discovery 请求查找某个 Endpoint 上全部的 Cluster 信息。



通过网络抓包可以看到 Simple Descriptor Response 包含了 Cooker Cluster(0xFC00)，至此增加自定义 Cluster 完成。用户可以通过 Read、Write 和 Commands 的方式操作这个 Cooker Cluster 的各项属性，完成对燃气炉的智能控制。



Ch.	Stack	Layer	Packet Information
11	ZigBee	NWK	Link Status
11	ZigBee	ZDP	Active Endpoints Request
11	ZigBee	MAC	Acknowledgement
11	ZigBee	ZDP	Active Endpoints Response
11	ZigBee	MAC	Acknowledgement
11	ZigBee	APS	Acknowledgement
11	ZigBee	MAC	Acknowledgement
11	ZigBee	NWK	Link Status
11	ZigBee	ZDP	Simple Descriptor Request
11	ZigBee	MAC	Acknowledgement
11	ZigBee	ZDP	Simple Descriptor Response
11	ZigBee	MAC	Acknowledgement
11	ZigBee	APS	Acknowledgement
11	ZigBee	MAC	Acknowledgement
11	ZigBee	ZDP	Simple Descriptor Request
11	ZigBee	MAC	Acknowledgement
11	ZigBee	ZDP	Simple Descriptor Response
11	ZigBee	MAC	Acknowledgement
11	ZigBee	APS	Acknowledgement
11	ZigBee	MAC	Acknowledgement



Manufacture Specific Cluster