# Swaiot物端对接工具使用指南

### 简介：

swaiot iot-tools是一个用于快速对接创维物联网平台的物端SDK生成工具。通过可视化的配置：平台、属性、功能等参数，自动生成对接SDK的接口与回调函数。在尽可能减少代码出错问题的同事，实现快速对接“创维IOT”平台的能力。您可以按照如下地址得到安装包文件：

工具及说明下载地址：

https://github.com/swaiot/swaiot\_tools

*本说明版权属于 深圳创维-RGB电子有限公司，由SWAIOT实验室维护。如有更改，恕不另行通知。20206.16*

### 使用说明：

下面您可以根据如下说明，生成设备端对接SDK用于快速接入创维物联网系统。

###### 工具简介

首先请根据平台下载iot-tools的安装包，并安装应用。在安装完成后会直接打开工具，同时在桌面生成应用图标。工具基本界面如下图：



工具首页中简单描述了工具的使用方式。您可以在左侧功能栏中切换配置页，并最终导出自动配置好的SDK对接代码。

同时工具中还提供了一个微型的“指令调试助手”，用于协助您在没有物联网模块的情况下进行MCU调代码调试。

###### 平台设置

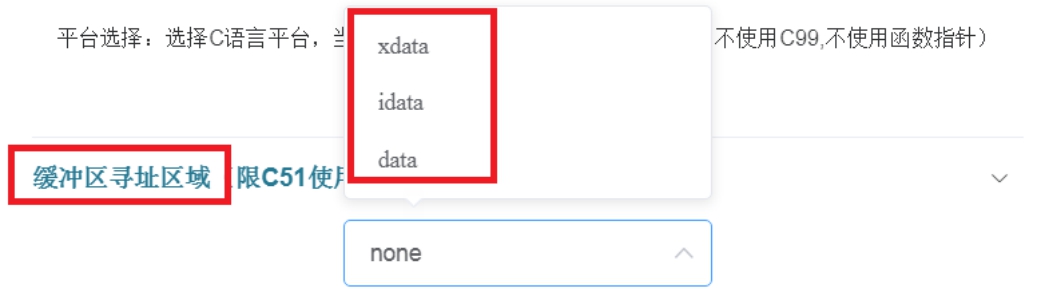
平台设置的主要功能是用于区分不同MCU带来的：大小端问题、整型（int）的位数问题、以及C51平台的内存寻址区域问题等。您可以根据您所使用的MCU平台来配置相关设置。默认 Arm Cortex-M系类使用小端+int32，C51及其兼容芯片使用大端+int16的形式。下图为平台设置界面。您可以点击选项卡进行详细配置。（默认支持Cortex-M内核MCU）



**“平台选择”**：点击平台设置选项卡后界面：



**“缓存区寻址区域”**：C51需要注意缓存区的寻址区域的选择，每次操作后，软件会记住上一次的设置，如需更改请注意。



###### 功能设置

功能设置的主要目的是优化SDK所需要的功能。比如说通讯协议中只使用了int8（Byte）变量，那么所有为int16、int32、float等服务的代码便都是冗余的。“功能设置”可以可视化的帮您进行代码裁切。



**“支持智慧屏功能”**：将对SDK添加心跳自动回复功能，以及sn码设置相关回调函数。



**“支持属性类型”**：将改变默认支持的属性类型，与属性设置联动。（如果如果关闭了部分类型，即使属性设置中纯在遗留的属性项，最终页不会导出与此类型相关的接口）



**“可选扩展功能”**：（WIFI物联模块的专属功能，智慧屏不支持），请根据需求选择添加功能



###### 属性设置

属性设置的主要功能是设置产品“三元组”（产品类型、产品型号、厂家代号）、设置产品通讯属性。具体属性与三元组可通过siaiot开放平台获取，或从对接工程师处索取。*（需要注意的是添加通讯属性中的数据类型是通过“功能设置而联动的”，如果在已经添加了部分属性后修改了“支持属性”，那么最终代码中将不会体现这些已取消类型的属性。纵然这些属性已经被写进属性列表，但代码中不会体现。）*



**“设置三元组”**：设置产品类型id、产品型号（字符串）、厂家ID。类型编号指的是产品所属的类型，比如洗衣机，台灯等等，这个可以产品类型表,品牌编号指的是品牌代码，比如创维:1，产品型号指的是产品自己的型号，比如净水器：T5A，产品型号名字符长度不能超过8个字符，不能有分号。



**“添加通讯属性”**：属性指的是需要与APP端进行交互的信息，属性名称需要选择选择数据类型。属性的个数最多有31个。属性名称最多8个字符，为了更方便的与合作方沟通，建议填写属性备注，有利于双方工作更好的展开。

可以重新编辑和删除属性。



###### 优化设置

默认无需修改优化设置。在MCU的RAM或ROM不足时，可以通过修改或者关闭循环缓冲区，用于减少系统消耗。



*（注意，应答设置本身不会明显减少系统消耗，因此建议保持周期答复模式）*

**“缓冲设置”**：打开循环缓冲区后，缓冲区的大小可以进行设置，byte旁边的数字表示最大支持4位数。



**“答复设置”**：有两种答复，直接答复和延迟答复。直接答复一调用发送函数就会被触发需要手动设置200ms后的答复，延迟答复是已经设置好200m后在发送（建议使用周期答复）



###### 导出设置

选择好输出路径之后，点击导出配置代码



**“生成的文件”**：生成的文件夹名就是swaiot,里面包含代码文件和属性表，请将swaiot文件添加到产品工程中，编译需要将swaiot和swaiot/lib添加到include路径中。

其中

product\_config.xlsx文件：设置信息一览表，您可以将表格中的预留信息填写清楚，用于内部对接以及资料归档。

iot*receive*handler.c文件：自动生成的回调函数，用于处理模块或者智慧屏下发的指令。

iot*user*config.h文件：自动生成的函数函数接口，内部包含了所有可用的函数接口。

lib文件夹：物联网协议依赖文件夹。无需阅读。



###### 调试工具

为了应对部分开发过程中缺少开发模块或app调试界面，又或者无法快速解析模块传输数据的问题，我们提供了一个简单的调试工具用模拟模块功能以及解析发送数据。



用电脑通过串口与MCU相连，使用调试工具模拟wifi模块或者智慧屏幕的相关指令是否正确。

点击“发送通讯属性”后，可以看到一份和“属性设置”中属性相同的属性表，用于模拟云端与物端的数据通讯、解码、格式是否正常。



# Swaiot物端SDK使用指南

### 简介：

使用“物端对接工具（iot-tools）”可以导出一包代码物端对接代码。您可以将代码移植到您所设置的平台，实现快速物联对接。

*本说明版权属于 深圳创维-RGB电子有限公司，由SWAIOT实验室维护。如有更改，恕不另行通知。20206.16*

### 使用说明：

下面您可以根据如下说明，接入创维SWAIOT物联平台。

###### 文件结构说明

使用“物端对接工具（iot-tools）”导出配置代码后，会在导出文件夹中生成一个名为“swaiot”的文件夹。“swaiot”文件夹的目录结构如下：

**swaiot**  
 **├── iot*receive*handler.c**  
 **├── iot*user*config.h**  
 **├── lib**  
 │ ├── iot*base.h*  
***│*** *├── iot*config.h  
 **│** ├── iot*core.c*  
***│*** *├── iot*core.h  
 **│** ├── iot*interface.h*  
***│*** *├── iot*loop.c  
 │ └── iot*reply.c*  
 *\*\*└── product*config.xlsx\*\*

备注：

*product\_config.xlsx文件：设置信息一览表，用于内部对接以及资料归档。*

*iotreceivehandler.c文件：自动生成的回调函数，用于处理模块或者智慧屏下发的指令。*

*iotuserconfig.h文件：自动生成的函数函数接口，内部包含了所有可用的函数接口。*

*lib文件夹：物联网协议依赖文件夹。需要添加到include路径中，但无需阅读。*

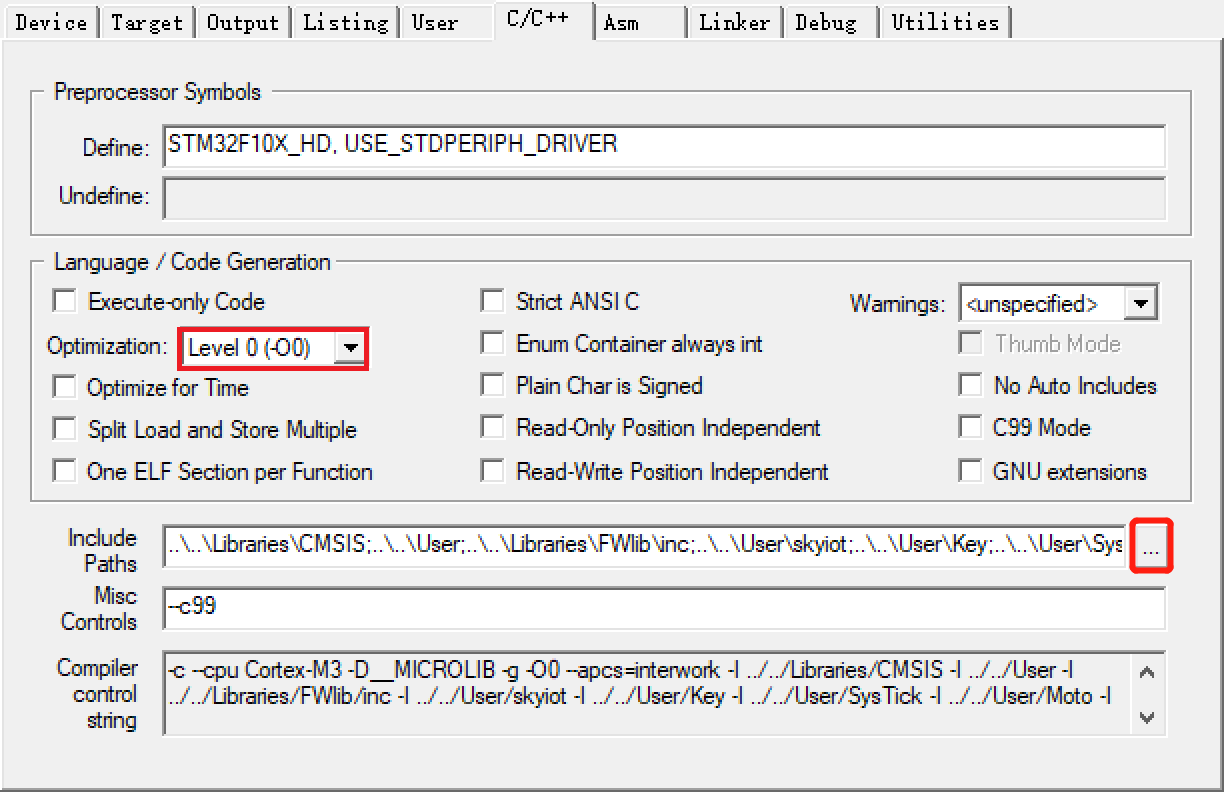
**请将swaiot文件添加到产品工程中，编译需要将swaiot和swaiot/lib添加到include路径中。**

###### Keil以及MDK移植参考及接口说明

1、将swaiot文件夹添加至工产品原始工程程中。  
![添加工程](data:text/html; charset=utf-8;base64,)

2、将swaiot和swaiot/lib两个文件夹添加至Include Pahts中。根据代码大小设置优化等级

由于SDK中函数使用量较多，因此建议C51代码优化开到最大，其他平台根据Flash和ram的剩余量进适当优化。



添加路径

（C51平台）

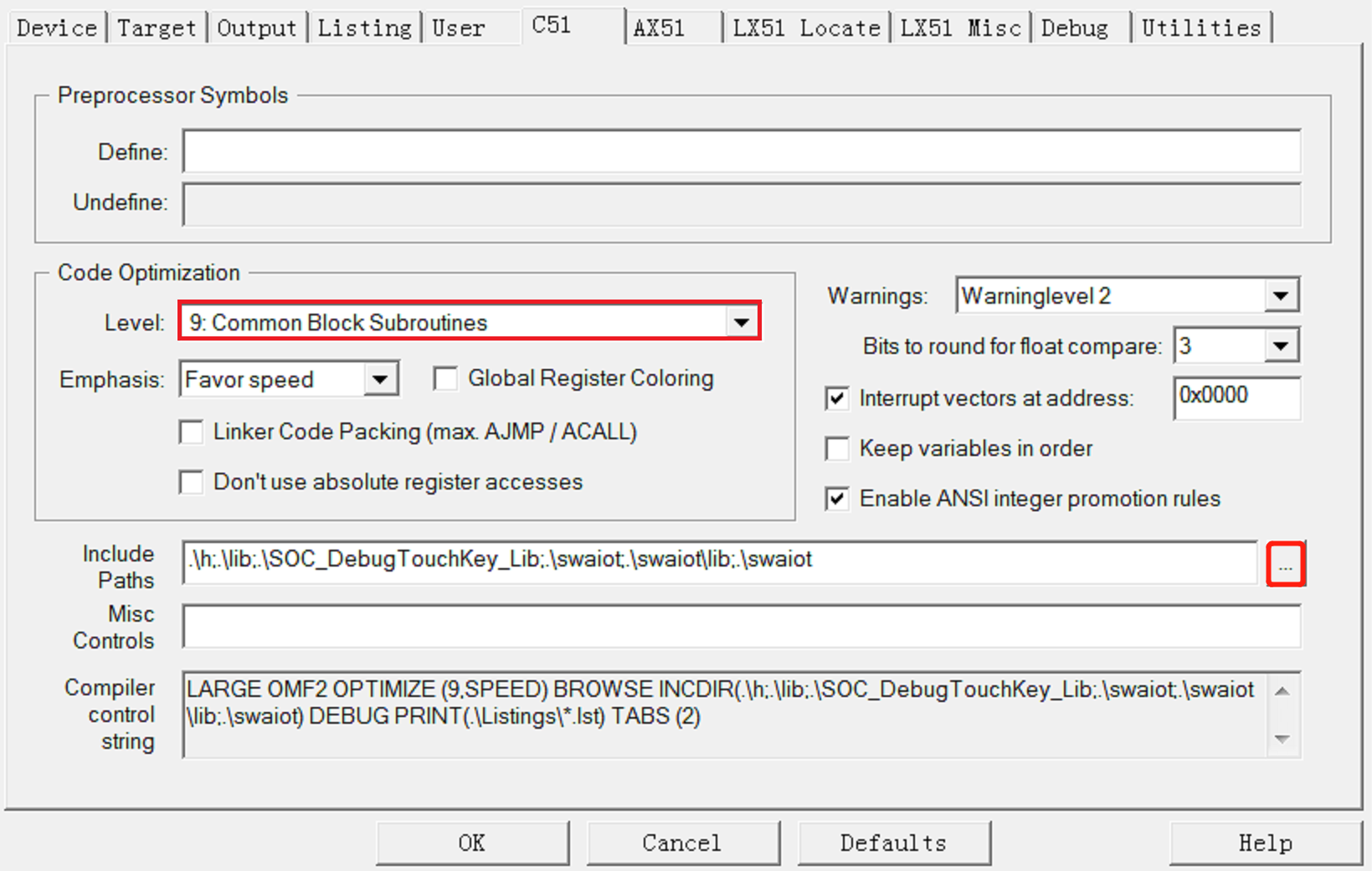
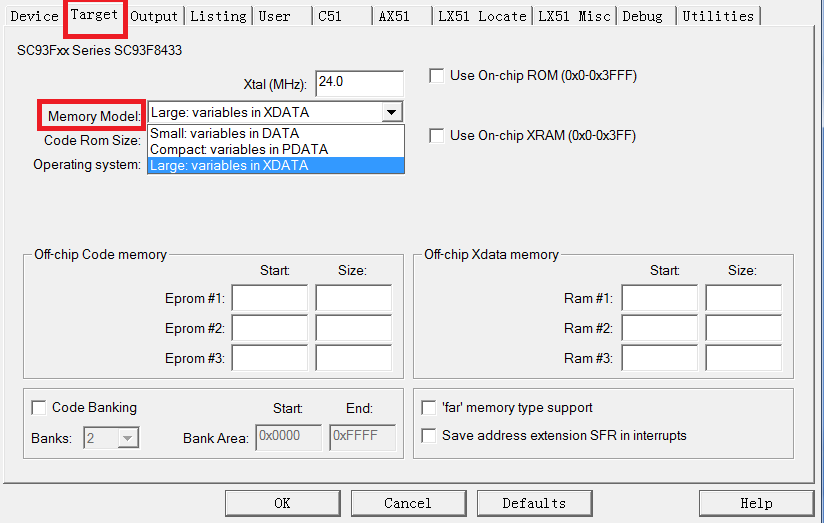


image-20200617163209824

另外对于C51平台存在多种寻址模式可以选择，编译时如果出现DATA溢出，请将内存模式设置为xdata



寻址

3、对iot*user*config.h文件的修改及调用，适配串行口发送宏定义。

①、修改如下代码中的宏定义“SEND*TO*NET\_CMD”，将宏定义所替换的字符串替换为当前的串口发送函数。

如#define SEND*TO*NET*CMD(dataP,dataSize) uart*write\_bytes(dataP, dataSize)

**注：函数原型为 void uart*write*bytes(char \*dataP, int dataSize)**

// Add header file of serial port sending function or declaration of serial port sending function   
  
  
// Replace "// uart\_write\_bytes(dataP, dataSize)" with serial port sending function of your platform   
// "Datap" is a char \* array, and "datasize" is the array length   
/\*\* Define of serial port sending function. "Datap" is a char \* array, and "datasize" is the array length \*\*/  
#define SEND\_TO\_NET\_CMD(dataP, dataSize) // uart\_write\_bytes(dataP, dataSize)

②、修改MCU接收中断（也可以是任意处理接收数据的代码节点，不局限于中断），添加接收函数。

在串口数据接收的C文件中引用头文件 iot*user*config.h，并将接收到的数据逐个注如下函数。

注：函数只接收单个byte。如果存在多个，请按顺序逐次调用并传参。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Function....: swaiotPushUartData  
Description.: Receive and analyze uart data  
Parameters..: byte, enter a byte from uart register  
Return......: NONE  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern void swaiotPushUartData(unsigned char byte);

③、在主函数中初始化SDK，并定时调用处理函数。

**函数initIotCore**： 请在主函数中初始化调用initIotCore函数，函数参数：

version： 字符型指针，指向一个长度为4的char形数组。[1.0.0.0]代表版本：1.0.0.0

（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

platformName：字符型指针，指向一个字符串（以\0结尾），代表平台名称。

（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

serialNumber： 字符型指针，指向一个长度为32的char形数组。默认全为0。

（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

**函数swaiotEventHandler\_10HZ**：请在任意位置以100MS的间隔调用

（100ms只是建议值，实际调用中不要求严格的保持100ms）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Function....: initIotCore  
Description.: initial firmware version(pointer of 4Bytes buffer) ; MCU's Type description (string); IOT's serial number(pointer of 32Bytes buffer)  
Parameters..: version, firmware version(pointer of 4Bytes buffer)  
Parameters..: platformName, String ending with 0   
Parameters..: serialNumber, pointer of 32Bytes buffer  
Return......: NONE  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern void initIotCore(char \*version, char \*platformName, unsigned char \*serialNumber);  
/\*\*  
Timing function of automatic parsing  
\*\*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Function....: swaiotEventHandler\_10HZ  
Description.: Timing function of automatic parsing, send command to module  
Parameters..: NONE  
Return......: NONE  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
extern void swaiotEventHandler\_10HZ(void);

④、执行模块功能：（如果使用wifi模块，下述三个接口功能必须实现）

iot*user*config.h中的所有函数都具备了相应的功能，比如调用sendCmdSetModuleConnect()设备将进入配网状态、调用sendCmdSetModuleUnbind()系统进入强制解绑状态、调用sendCmdSetModuleFacTest()模块进入工厂自检模式。（对接智慧屏时，功能还输仅支持获取时间，与属性操作函数，心跳、串码回复灯功能SDK会自动完成）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdSetModuleConnect   
Description.: set module to connect mode   
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdSetModuleConnect(void);   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdSetModuleFacTest   
Description.: set module to factory test mode   
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdSetModuleFacTest(void);   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdSetModuleUnbind   
Description.: set module to forced unbind mode   
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdSetModuleUnbind(void);

除此之外，根据您使用iot*tools工具中不同的配置，会生成不同的功能函数，每个函数都有相关的备注提示您其功能。您可以根据个人需求逐个添加调用。*  
*<img src="https://docman.skyworthiot.com/uploads/prod-access-doc/images/m*216b5571bf439bdbd1d5d54112efa124\_r.png" alt="可选扩展功能" style="zoom:50%;" />

对应：

#ifdef LISTENER\_MODULE\_STATUS   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdGetModuleState   
Description.: get module state  
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdGetModuleState(void);   
#endif   
  
#ifdef SUPPORT\_GET\_UTC   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdGetUtc   
Description.: get the current UTC time in seconds   
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdGetUtc(void);   
#endif   
#ifdef SUPPORT\_GET\_IP   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdGetIp   
Description.: get module IP address (Exclusive to WiFi module)  
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdGetIp(void);   
#endif   
#ifdef SUPPORT\_GET\_SSID   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdGetIp   
Description.: get module SSID (Exclusive to WiFi module)  
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdGetSsid(void);   
#endif   
#ifdef SUPPORT\_GET\_MAC   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdGetMac   
Description.: get module MAC address (Exclusive to WiFi module)  
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdGetMac(void);   
#endif   
#ifdef SUPPORT\_GET\_RSSI   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendCmdGetRssi   
Description.: get module RSSI (need special modules)  
Parameters..: NONE   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendCmdGetRssi(void);  
#endif  
……………………

⑤、属性发送函数（**由于为了更简单的兼容51系芯片，SDK中不使用函数指针进行动态注入**）

根据您使用iot\_tools工具中配置的不同的属性，系统会自动生成属性更新函数。函数命名规则为：

extern void sendcmdProp**XxxXxx**(type value)

其中XxxXxx代表属性名被转义成驼峰写法的字符串名称,例如：



添加通讯属性

对应：

#ifdef CMD\_MASK\_PROD\_POW\_S   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: sendcmdPropPowS   
Description.: update property POW\_S  
Parameters..: value, property of type char   
Return......: NONE   
Explain.....: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
extern void sendcmdPropPowS(char value);  
#endif

您可以直接调用并更新属性（传输是否成功，您可以使用iot\_tools中的调试助手进行本地验证）

4、对iot*receive*handler.c文件的接口说明

在iot*receive*handler.c文件中描述了所有接收事件的回调函数，如更新一个属性、模块状态发生变化、更新当前时间等。具体函数说明均可参考源码中所附带的备注。下面对自动生成函数的结构予以说明。

①、模块状态监听函数（智慧屏对接无效）

如果您在iot\_tools工具配置了监听模组状态，生成文件中会包含此函数，您可以根据枚举内容来确定模块具体状态。（具体状态有那些，可以参考物联网通信协议指令说明）

#ifdef LISTENER\_MODULE\_STATUS   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: moduleStateChangeHandler   
Description.: Callback function for updating module state   
Parameters..: ENUM\_MODULE\_STATE\_SKY state : it is an enumeration of states. Please refer to the iot\_base.h   
Return......: NONE   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
 void moduleStateChangeHandler(ENUM\_MODULE\_STATE\_SKY state){   
 // update state   
 }   
#endif

您可以通过此函数的状态反馈来确定WIFI灯或其他状态提示方式的状态。以WIFI模块为例（*智慧屏不支持此功能，不会触发。*），收到：

（WIFI提示灯状态）

STA*STATE*CONFIGING：WIFI提示灯慢速闪烁（5分钟）

STA*STATE*CONNECTED\_SERVER：WIFI提示灯常亮

STA*STATE*CONNECTED或STA*STATE*DISCONNECT\_SERVER：WIFI提示灯熄灭

（出场检测提示状态）

STA*STATE*IN*FACTORY*MODE：进入工厂自检，需要提示

STA*STATE*FACTORY\_OVER：工厂自检成功提示

STA*STATE*FACTORY\_FAIL：工厂自检失败提示

void moduleStateChangeHandler(ENUM\_MODULE\_STATE\_SKY state){   
 // update state   
 if(state == STA\_STATE\_NONE){  
 // 无效状态，不做任何处理  
 }else if(state == STA\_STATE\_CONFIGING){  
 // 配网或联网中，需要提示wifi状态（WIFI指示灯闪烁5分钟）  
 }else if(state == STA\_STATE\_CONNECTED){  
 // 联网成功，可忽略或提示wifi状态（WIFI指示灯快速闪烁）  
 }else if(state == STA\_STATE\_DISCONNECT\_SERVER){  
 // 连接断开，需要提示关闭WIFI提示灯  
 }else if(state == STA\_STATE\_CONNECTED\_SERVER){  
 // 服务器连接成功，需要提示wifi状态（WIFI指示灯常量）  
 }else if(state == STA\_STATE\_IN\_FACTORY\_MODE){  
 // 进入模块工厂自检状态，需要提示，以便于工厂厂测人员处理（如全屏闪烁）  
 }else if(state == STA\_STATE\_FACTORY\_OVER){  
 // 工厂自检成功，需要提示，以便于工厂厂测人员处理（如全屏熄灭，恢复关机状态）  
 }else if(state == STA\_STATE\_FACTORY\_FAIL){  
 // 工厂自检失败，需要提示，以便于工厂厂测人员处理（如全常亮，或报错。）  
 }else if(state == STA\_STATE\_BACK\_MODE){  
 // 忽略  
 }else if(state == STA\_STATE\_ABANDON){  
 // 忽略  
 }else if(state == STA\_STATE\_FORCE\_UNBIND\_OVER){  
 // 强制解绑成功，可忽略（之后会收到STA\_STATE\_CONFIGING）  
 }else if(state == STA\_STATE\_FORCE\_UNBIND\_FAIL){  
 // 强制解绑失败，可忽略  
 }else if(state == STA\_STATE\_PRODUCT\_UPGRADE\_MODE){  
 // 进入升级状态，如使用升级模式，需要提示  
 }else if(state == STA\_STATE\_PRODUCT\_UPGRADE\_OVER){  
 // 升级成功，如使用升级模式，需要提示  
 }else if(state == STA\_STATE\_PRODUCT\_UPGRADE\_FAIL){  
 // 升级失败，如使用升级模式，需要提示  
 }   
 }

枚举ENUM*MODULE*STATE*SKY的原型存放在“swaiot/lib/iot*base.h”中。

②、属性更新函数：

根据您在iot\_tools中配置的不同属性类型，生成文件中会包含不同的属性过呢高兴函数。函数在接收到数据时被调用，通过传参name来判断属性名称，并通过value来传递属性值。

代码中会自动生成通过strcmp生成判断结构，你可以在if语句中添加自己需要的操作。

例如设置了一个名为POW\_S的8位属性，代码中会自动添加

if(0 == strcmp(name, PROD*POW*S\_NAME))

相关状态判断

#ifdef SUPPORT\_DATA\_TYPE\_INT8   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: receiveAPropU8   
Description.: Callback function for updating 8-bit property   
Parameters..: char \*name: property name(string)   
Parameters..: iot\_u8\_t value: command value(uint8\_t).   
Return......: NONE   
Explain.....: Distinguishing property names by using strcmp(s1, s2)   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
 void receiveAPropU8(char \*name, iot\_u8\_t value){   
 // has a uint8\_t data   
 if(0 == strcmp(name, PROD\_POW\_S\_NAME)){   
 // has POW\_S   
 }   
………………  
 }   
#endif   
………………  
 void receiveAPropU16(char \*name, iot\_u16\_t value){  
………………  
 }   
………………  
 void receiveAPropS32(char \*name , iot\_s32\_t value){   
………………  
 }   
………………  
 void receiveAPropF32(char \*name , float value){   
………………  
 }   
………………  
 void receiveAPropStr(char \*name , char \*value\_p){   
………………  
 }

其中 :

void receiveAPropU8(char \*name, iot*u8*t value) 代表八位无符号属性的接收回调函数

void receiveAPropU16(char \*name, iot*u16*t value) 代表十六位无符号属性的接收回调函数

void receiveAPropS32(char \*name , iot*s32*t value) 代表三十二位有符号属性的接收回调函数

void receiveAPropF32(char \*name , float value)代表三十二位单浮点属性的接收回调函数

void receiveAPropStr(char \*name , char \*value\_p) 代表字符串属性的接收回调函数

③、其他功能回调函数

根据您使用iot\_tools工具中配置的不同的功能，系统会自动生成不同的回调函数。函数具体功能请参照代码中的代码说明。例如下面便是更新时间的回调函数，使用者可以在函数中添加更新时间相关处理代码。

#ifdef SUPPORT\_GET\_UTC   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Function....: moduleTimeHandler   
Description.: Callback function for updating time   
Parameters..: uint32\_t utc: UTC time in seconds units, which is based on the time 1970/1/1 8:0:0   
Return......: NONE   
Explain.....: 2018.01.01 0:0:0(UTC/GMT 0) is 80 09 49 5A 5A is high, 80 is low   
............: If you use Beijing time (GMT + 8), you need to subtract 8\*3600 from the original data   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   
 void moduleTimeHandler(iot\_s32\_t utc){   
 // exp : 2018.01.01 0:0:0(UTC/GMT 0) is 80 09 49 5A   
 // If you use Beijing time (GMT + 8), you need to subtract 8\*3600 from the original data   
 }   
#endif   
………………