



# 基于开发板的开发指南

StartKit\_NB\_V1.0

中国电信集团公司

2019 年 09 月

---

## 文档信息

文档名称	基于开发板的开发指南_StartKit_NB_V1.0
文件编号	V1.0 （内部版本号 1.0.0）
编制人	张海名、王艺等
保密级别	供使用 NB 模组接入平台的开发者使用

分发范围：终端开发者

## 目录

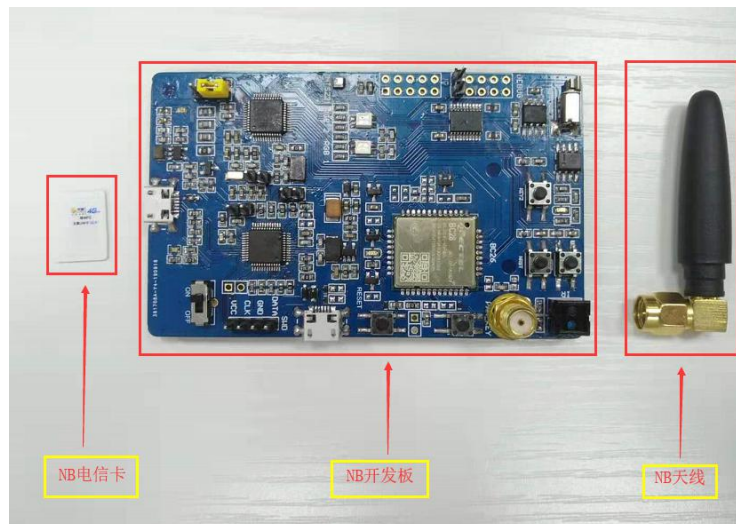
一、 清单列表.....	2
二、 硬件连接.....	3
三、 软件安装.....	4
四、 软件开发.....	7
1. 代码结构介绍.....	7
2. 开发开发板程序.....	10
3. 编译开发板程序.....	10
4. 下载开发板程序.....	10
五、 软件测试.....	11
六、 样例说明.....	12
开发板程序开发步骤.....	12
附 1 中国电信物联网开放平台操作步骤.....	17
1、 用户登录平台.....	17
2、 在平台创建产品和设备.....	17
3、 在平台创建属性和服务.....	19
4、 从平台获取终端接入地址、接入端口、服务 ID.....	25
附 2 深入开发参考.....	25
附 3 数据编解码规则.....	29

## 一、清单列表

硬件：

电信开发套件：NB 电信卡、NB 开发板、NB 天线；

自备硬件工具：USB 数据线（安卓数据线）、杜邦线、usb 转 ttl 模块、USB Hub（可选）；



电信开发套件：



自备硬件工具：

软件：

电信软件套件：开发板工程代码包 (skit\_ctnb\_st.zip)；

自备软件工具：ST-Link 驱动、串口工具 StartKit 或其它串口工具(配合 usb 转 ttl 模块使用)、Keil5 软件、Keil.STM32F1xx\_DFP.2.2.0 库包；

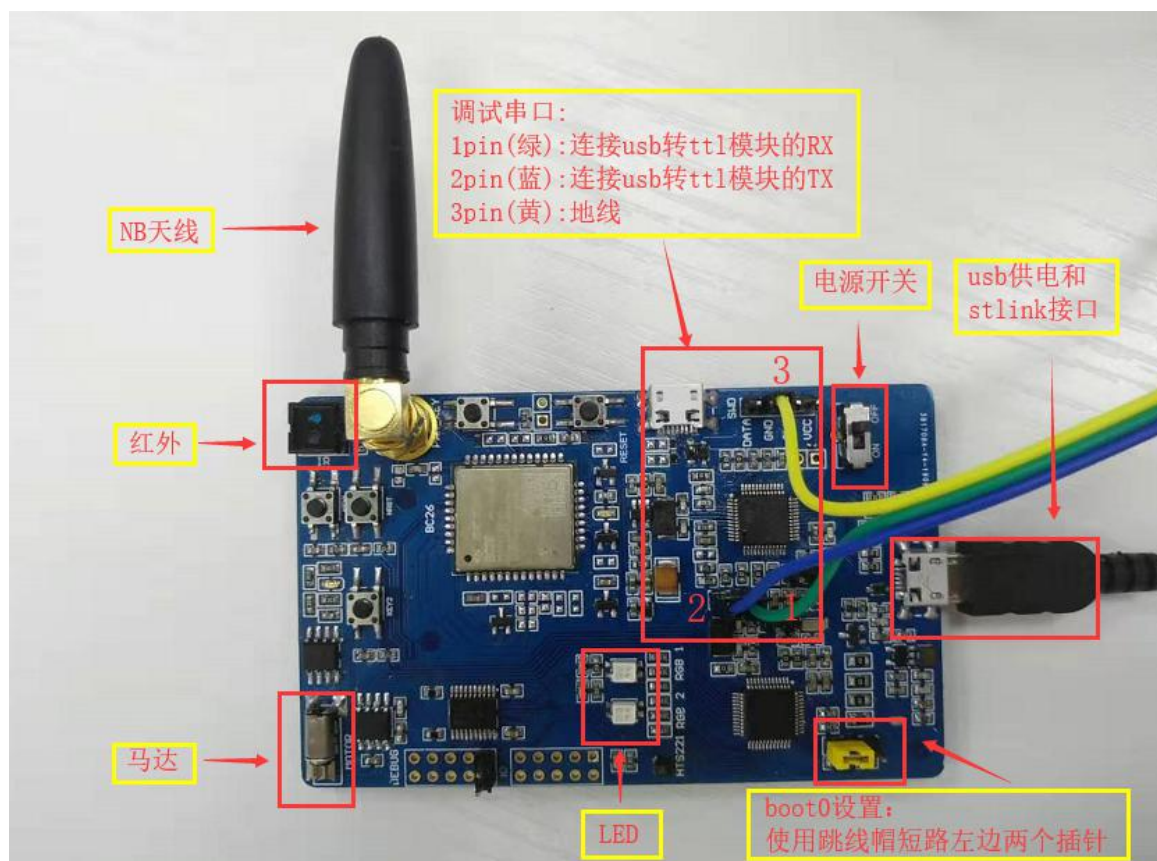
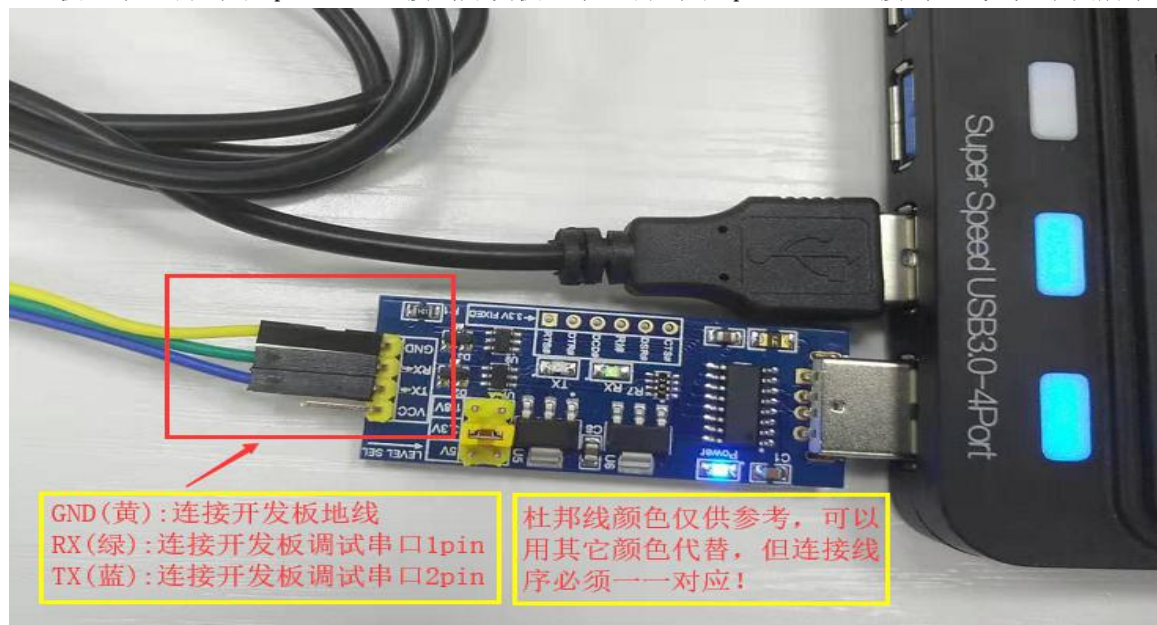
文档：

《Quectel\_BC35-G&BC28&BC95 R2.0\_AT\_Commands\_Manual\_V1.5》

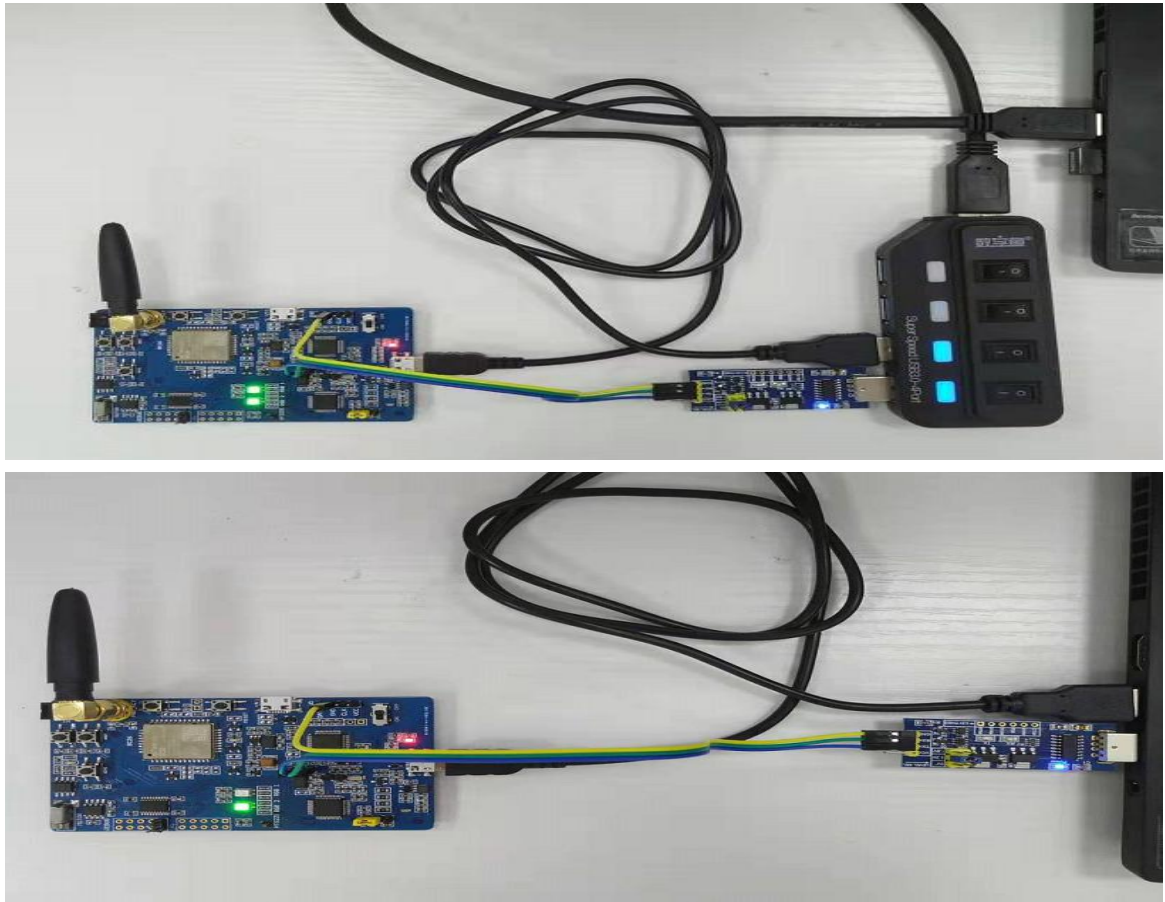
## 二、硬件连接

NB 开发板套件使用 NB 开发板，通过板载 ST-Link 工具下载 mcu 程序，使用 usb 转 ttl 模块进行串口调试。

1. 在断电条件下插入电信 NB-IOT 通信卡(必须事先实名认证)，将 usb 数据线一端插入 hub 接口，另一端插入开发板上的 microusb 接口，用杜邦线将 usb 转 ttl 模块上的 RX 连接到开发板上串口调试的 1pin、TX 连接到开发板上串口调试的 2pin、GND 连接到地线，如下图所示：



2. 开发板供电方式主要有三种，第一通过 hub 供电，第二通过电脑 usb 供电，第三通过 5v 的 usb 接口专用充电器供电（常用安卓手机充电器也可满足需求），前两者如下图所示：




### 三、软件安装

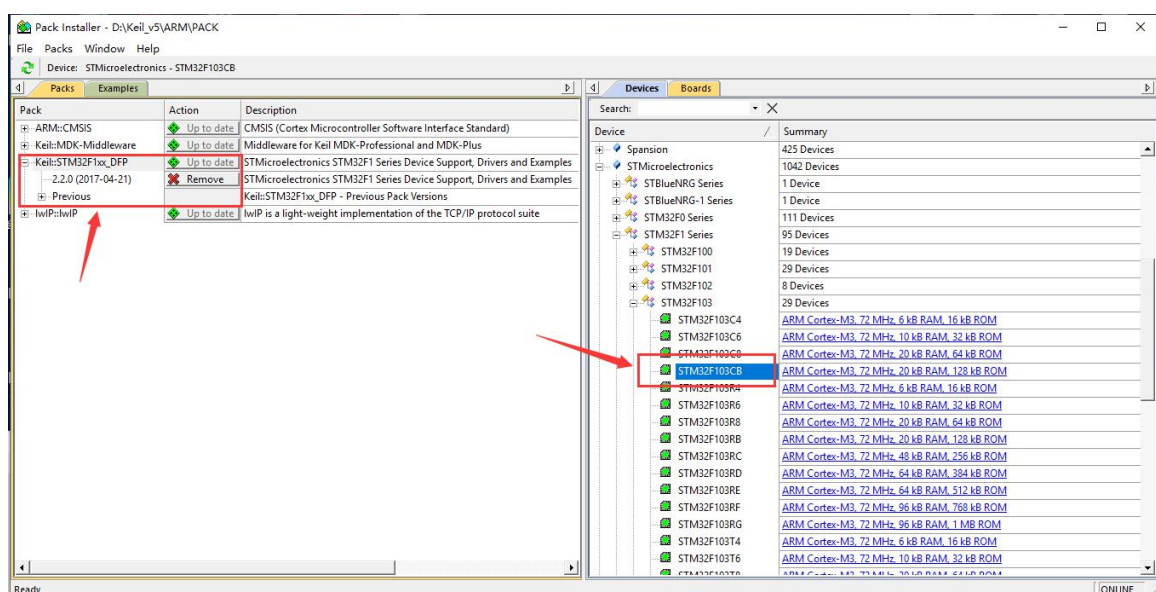
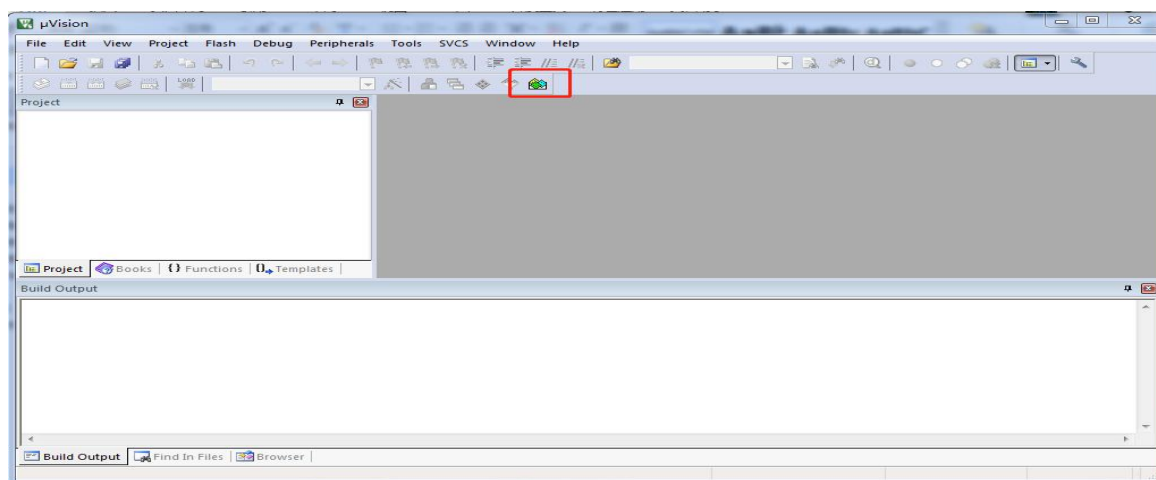
1. 安装 STLink 驱动(自备)；
2. 如果需要 Debug 工具，请安装 StartKit.exe 专用工具或其它串口工具 (自备)；
3. 安装 Keil5 软件(自备)；
4. Keil5 导入库包 Keil.STM32F1xx\_DFP.2.2.0 或最新库包(自备)；

导入库包 Keil.STM32F1xx\_DFP.2.2.0 可选择以下两种方法中的一种：


方法一：

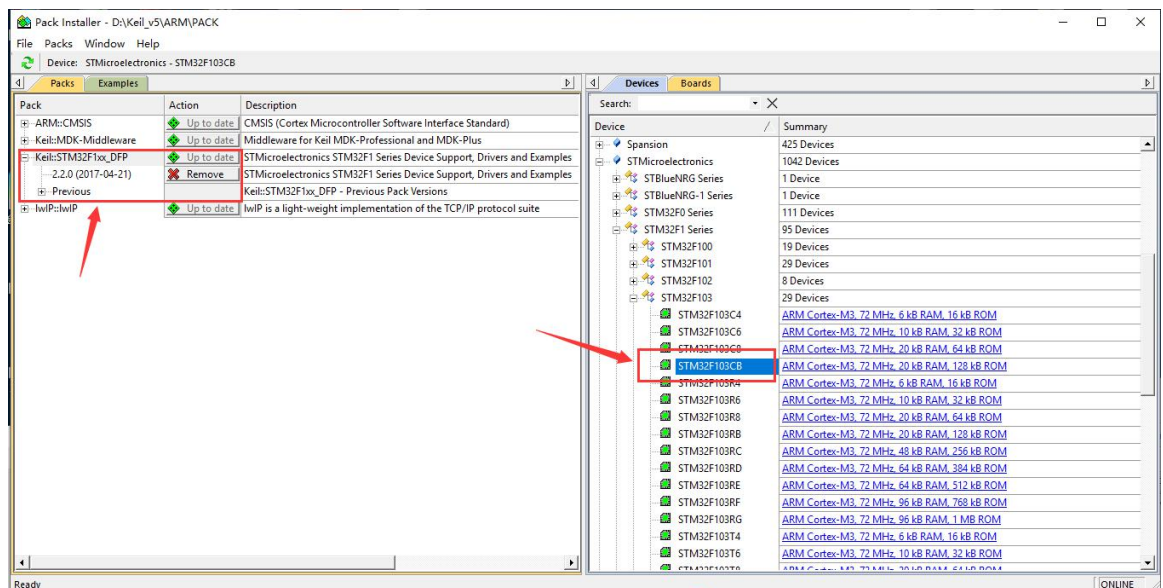
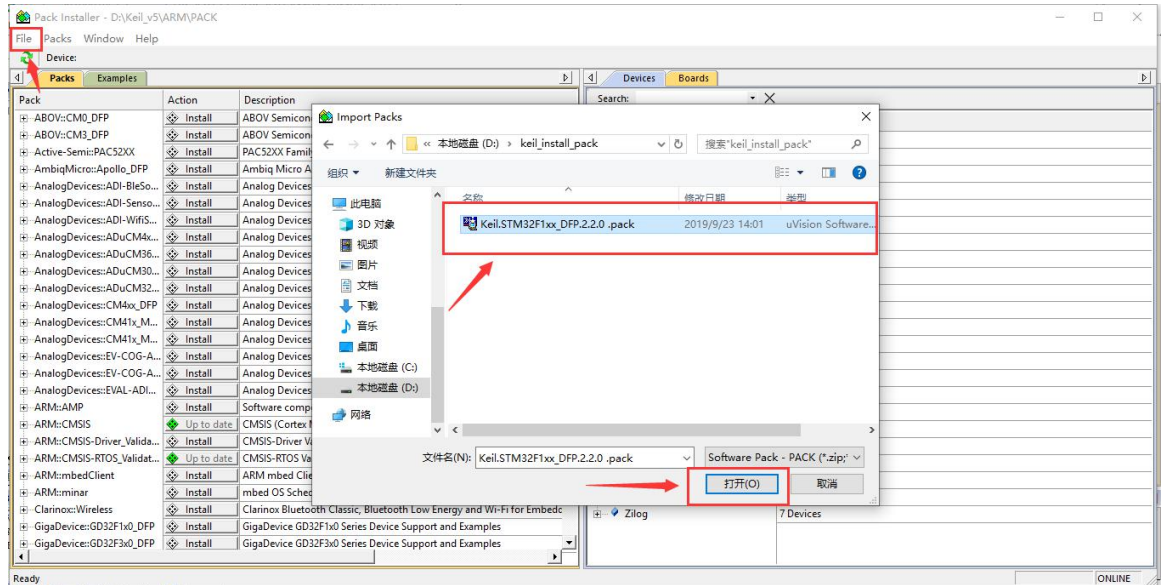
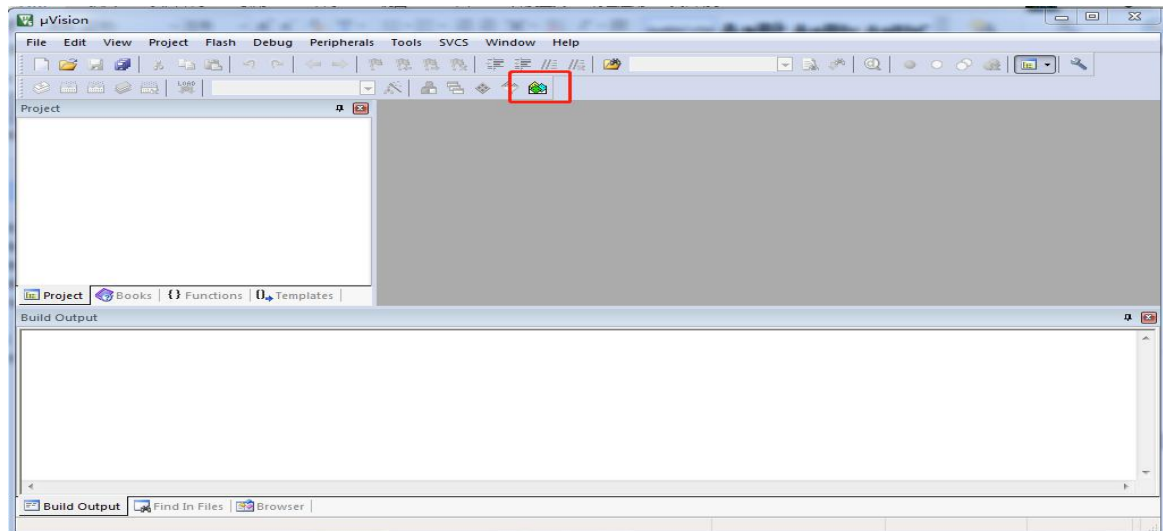
打开工程，点击‘Pack Installer’按钮，然后在 pack installer 界面从 Packs 树形图中选择 STM32F1xx\_DFP，点击‘Install’按钮，安装 Keil.STM32F1xx\_DFP 库包，安装完成后应该从右边 device 树状列表中就能查到设备型号“STM32F103CB”。





方法二：

下载 Keil.STM32F1xx\_DFP.2.2.0 包，下载地址为：<http://www.keil.com/pack>，打开工程，点击 ‘Pack Installer’ 按钮 ，然后在 pack installer 界面点击菜单 file，选择 import 菜单，选择下载的安装包进行导入操作，安装完成后应该从右边 device 树状列表中就能查到设备型号 “STM32F103CB”。





## 四、软件开发

开始进入软件开发前，请先熟悉 NB 开发板相关 AT 指令文档，从中国电信物联网开放平台（<https://www.ctwing.cn>）下载代码源文件。

### 1. 代码结构介绍

1.1 下载开发板程序文件 skit\_ctnb\_st.zip，解压文件到自己工作目录中（建议直接解压在根目录下，如：D:\skit\_ctnb\_st）；

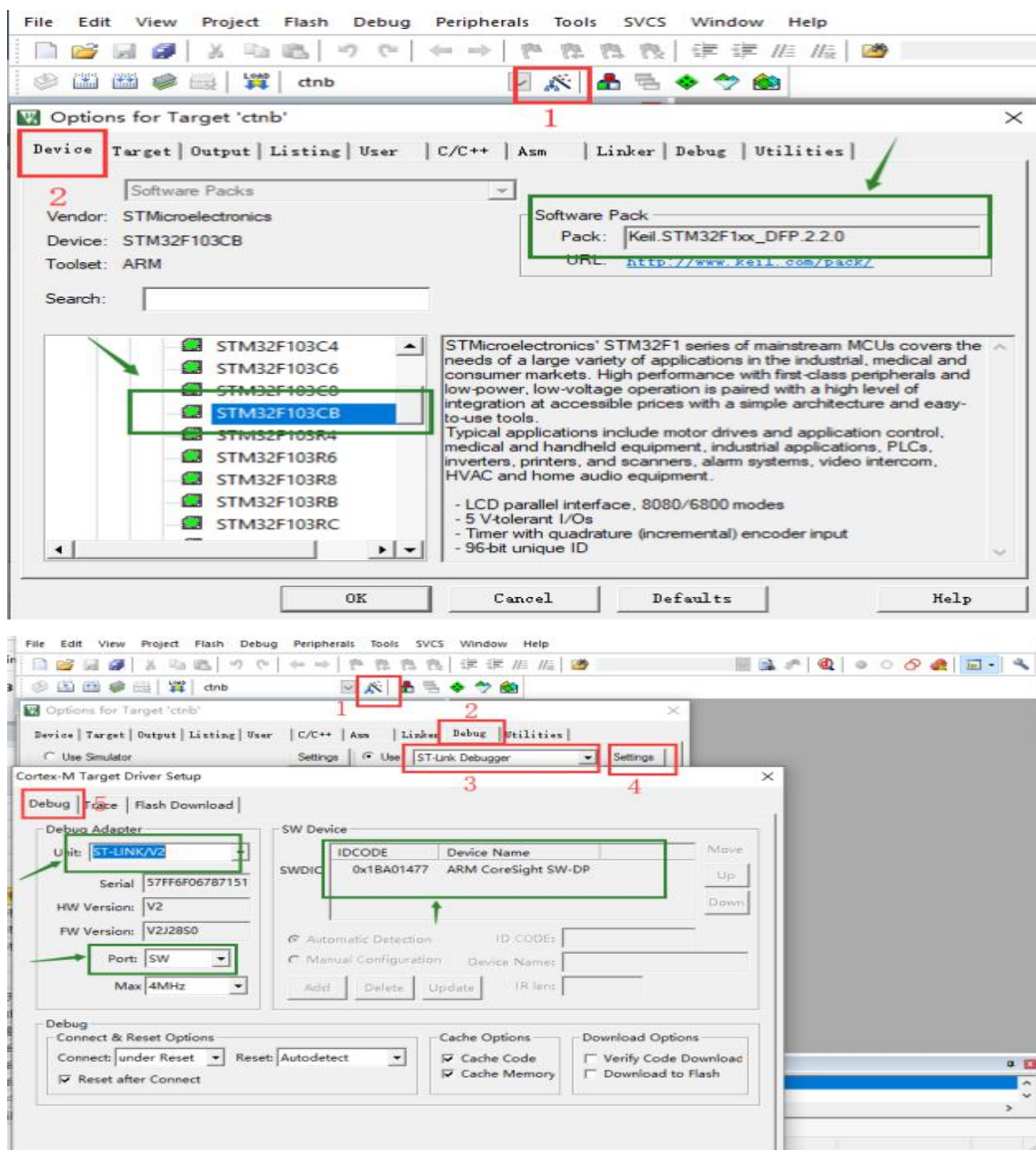
文件目录结构如下：

名称	修改日期	类型	大小
bsp	2019/9/23 16:41	文件夹	
Drivers	2019/9/23 16:41	文件夹	
MDK-ARM	2019/9/23 16:41	文件夹	
RTOS	2019/9/23 16:41	文件夹	
system	2019/9/23 16:41	文件夹	
user	2019/9/23 16:41	文件夹	
.mxproject	2019/9/23 16:37	MXPROJECT 文件	5 KB
ctnb.ioc	2019/9/23 16:37	STM32CubeMX	6 KB
mx.scratch	2019/9/23 16:37	SCRATCH 文件	5 KB

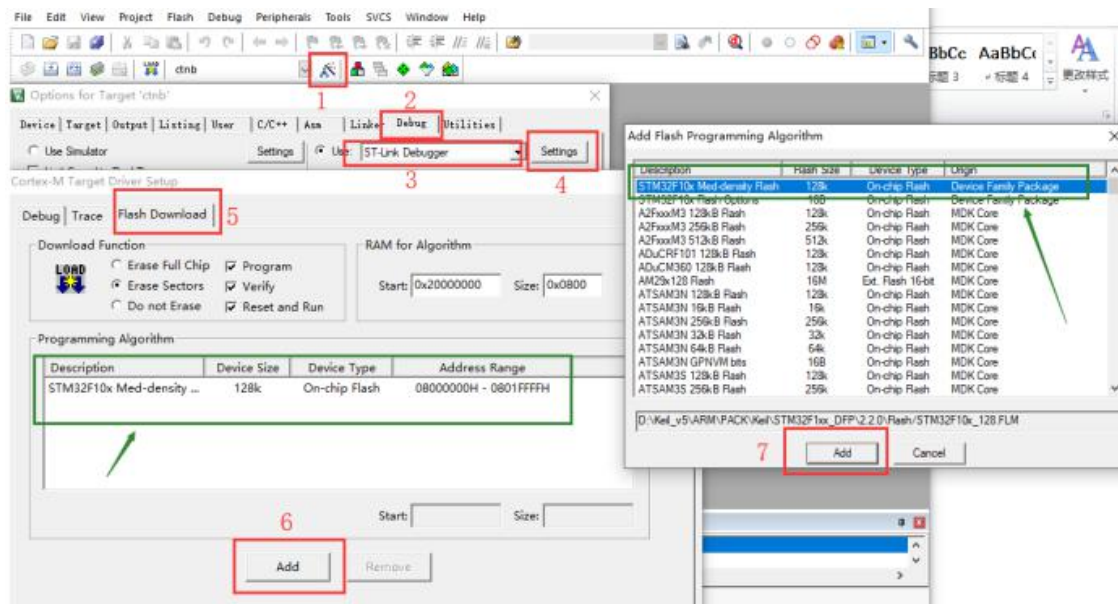
- bsp:** 存放 BSP 驱动文件（如 LED，电机，温湿度传感器等），共用工具类文件（如 MD5，字符串处理函数等）；
- Drivers:** 存放 STM32 平台的 HAL 库文件和 CMSIS 接口文件；
- MDK-ARM:** 存放 Keil 工程启动文件；
- RTOS:** 存放操作系统 RTOS 相关文件（如 OS 线程，日志等）；
- system:** 存放 main 函数入口文件和是 STM32 系统接口文件（如中断，系统时钟初始化、HAL 初始化等）；
- user:** 存放用户线程文件（如 与通信模组的 AT 操作和业务实现函数等）；

1.2 开发板程序是基于 MDK IDE 编写，使用 Keil5 打开工程：进入 MDK-ARM 目录，双击文件 ctnb.uvprojx，或者打开 Keil 软件后选择 project 菜单中的 open project，即可打开代码工程。

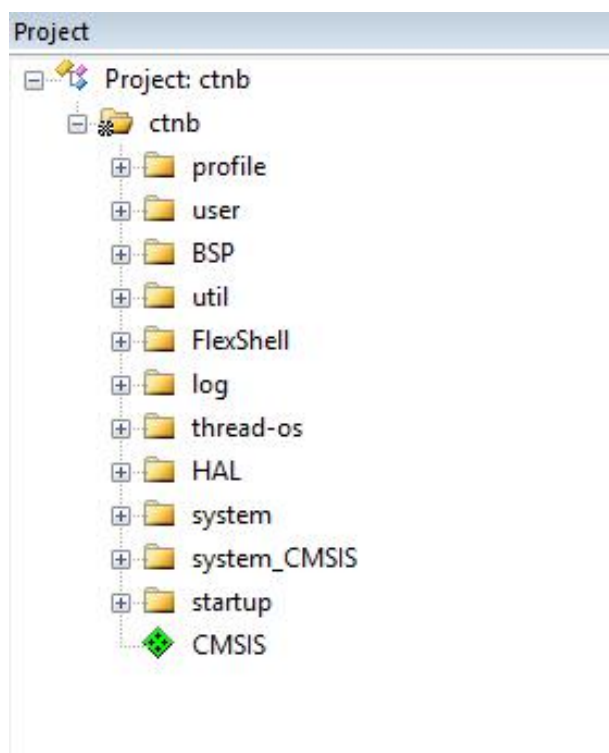
确认配置信息（按照红色步骤点击菜单，确认绿色内容）：  
ST-Link 如下图：



如果没有下载配置内容，可以自行点击‘Add’添加下载配置，如下图：



工程架构介绍:



**profile:** 存放编解码文件;

**user:** 业务实现的具体存放区, 目前里面已经实现了 nb\_thread, bsp\_thread, ctnb\_thread 等, 如果用户想创建自己的业务逻辑可以参考这里的 thread。

**BSP:** 存放 BSP 驱动文件 (如 LED, 电机, 温湿度传感器、GPIO、SPI、IIC, 红外传感器、BSP 通用接口等);

**util:** 共用工具类文件 (如 MD5, 字符串处理函数等);

**FlexShell:** 命令行实现的具体文件, 用户想自己创建测试命令, 可以在 FlesShellUser.c

文件里追加。

**log:** 日志打印文件，用户可以选择 UART 或者 RTT 进行日志数据的打印。

**thread-os:** 一个基于线程的 RTOS。用户不需要关心该文件。

**HAL:** st 公司提供的库文件。

**system:** 存放 main 函数入口文件和是 STM32 系统接口文件（如中断，系统时钟初始化、HAL 初始化等）；

**system\_CMSIS:** 系统时钟配置文件。

**startup:** stm32 启动文件。如果用户想修改栈大小，可以修改该文件

**备注:**如需使用日志、定时器、线程、shell 命令详见附件

## 2. 开发开发板程序

请参考 NB 相关 AT 指令文档开发自己的业务程序，详见样例说明中的[开发步骤](#)。


## 3. 编译开发板程序

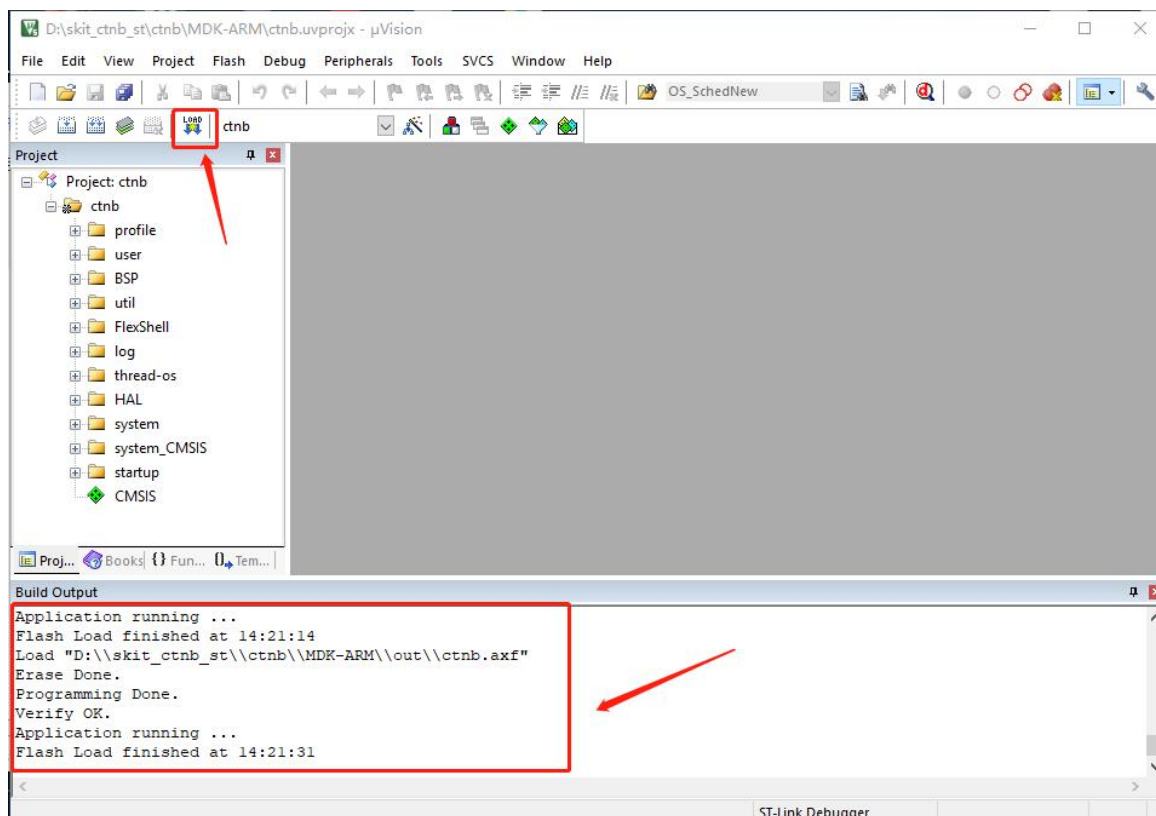
连接好开发板，点击左上角编译图标进行代码编译，确认编译结果没有 error 出现，编译成功后输出如下：

```
Build Output
..\system\Src\main.c(236): warning: #177-D: function "MX_USART2_UART_Init" was declared but never referenced
static void MX_USART2_UART_Init(void)
..\system\Src\main.c(201): warning: #177-D: function "MX_RTC_Init" was declared but never referenced
static void MX_RTC_Init(void)
..\system\Src\main.c: 3 warnings, 0 errors
compiling stm32flxx_hal_msp.c...
compiling stm32flxx_it.C...
compiling system_stm32flxx.c...
assembling startup_stm32f103xb.s...
linking...
Program Size: Code=38854 RO-data=2546 RW-data=3292 ZI-data=5644
".\out\ctnb.axf" - 0 Error(s), 7 Warning(s).
Build Time Elapsed: 00:00:55
```

ST-Link Debugger

## 4. 下载开发板程序

编译成功后点击 load 图标进行下载调试，下载成功后输出如下图：



## 五、软件测试

5.1 通过 Keil 下载程序到开发板后，开发板会自动运行用户代码，如果设备登录，可以在中国电信物联网开放平台查看终端是否在线状态；



5.2 平台查看上报的数据；



5.3 平台执行指令下发，查看指令下发日志，查指令执行状态；





5.4 开发板可以根据用户自己编写的程序逻辑验证数据上报或者指令下发是否测试通过。

## 六、样例说明

基于 NB 开发板的样例程序通过串口使用 AT 指令与 NB 模组通信，所有 AT 指令集参数说明详见《Quectel\_BC35-G&BC28&BC95 R2.0\_AT\_Commands\_Manual\_V1.5》

## 开发板程序开发步骤

1、连接好开发板，搭建好开发环境；

2、在中国电信物联网开放平台创建产品，添加设备(设备 IMEI 信息在模组上面)，新增属性/服务，详细步骤见“[中国电信物联网开放平台操作步骤](#)”，创建完成后，用户获取并保存如下所需信息：

- a. 接入 IP 地址 (hostIP)，接口端口(hostPort)，
- b. 服务 ID (用于设置数据上报或下行的 DatasetID)；

备注：

属性和服务可以使用导入物模型（从其它产品导出的物模型）方式快速生成，创建成功后支持导出物模型（供其它产品使用）和生成设备侧编解码文件功能；

3、获取代码包，使用 Keil 软件打开代码工程；

- a. 主要关注 user 目录中的如下代码文件：

```
main.c           //主程序入口
userconfig.h     //相关配置信息
nb_thread.h      //业务接口头文件
nb_thread.c      //业务接口源代码文件
```

如果需要使用自定义物模型编解码代码，请使用前面生成的设备侧编解码文件替换 profile 目录中的两个文件：

```
app_aep_profile.c //编解码源代码文件
app_aep_profile.h //编解码头文件
```

- b. AT 指令代码编写实例（拼接 AT 串+发送 AT 指令）：

```
//拼接并发送 AT 指令
printf("AT+CTM2MREG=%d\r\n", CTIOT_REG_LIFETIME);
```

#### 4、修改接入配置信息 (userconfig.h)

```

1  #ifndef USERCONFIG_H
2  #define USERCONFIG_H
3
4
5  /**
6   * server params config
7   *
8   */
9
10
11  #define CTIOT_INIT_IP "221.229.214.202"
12  #define CTIOT_INIT_PORT 5683
13  #define CTIOT_REG_LIFETIME 3600
14
15
16
17 #endif
18
19

```

#### 5、业务场景接口开发

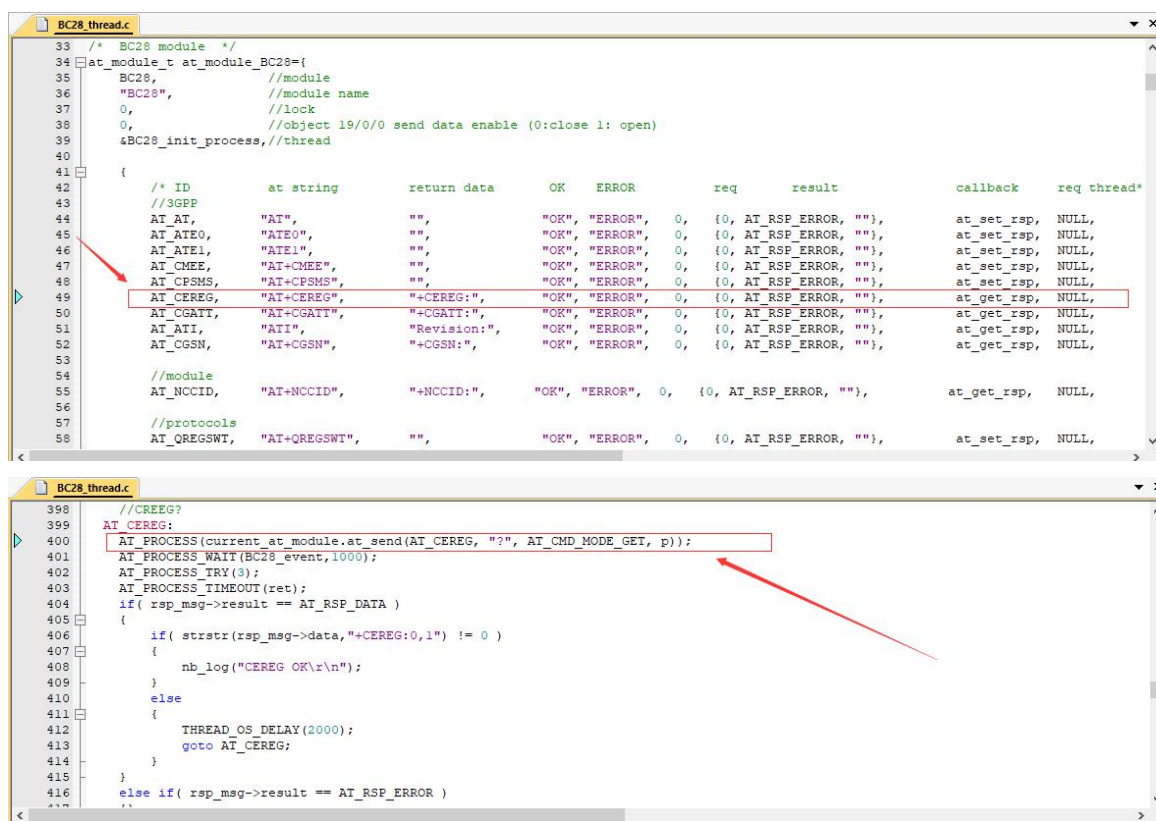
##### a) 查询 NB 网络状态

AT 指令: AT+CEREG?r\n

详见 BC28\_thread.c 中的

AT\_PROCESS(current\_at\_module.at\_send(AT\_CEREG, "?", AT\_CMD\_MODE\_GET, p)),

功能代码在 BC28\_thread.c 中已实现, 无需修改;



```

33 /* BC28 module */
34 at_module_t at_module_BC28={
35     BC28, //module
36     "BC28", //module name
37     0, //lock
38     0, //object 19/0/0 send data enable (0:close 1: open)
39     &BC28_init_process, //thread
40 }
41
42 {
43     /* ID      at string      return data      OK      ERROR      req      result      callback      req thread */
44     //3GPP
45     AT_AT, "AT", "", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
46     AT_ATE0, "ATE0", "", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
47     AT_ATE1, "ATE1", "", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
48     AT_CMEE, "AT+CMEE", "", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
49     AT_CPSMS, "AT+CPSMS", "", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
50     AT_CEREG, "AT+CEREG", "+CEREG:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
51     AT_CGATT, "AT+CGATT", "+CGATT:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
52     AT_ATI, "ATI", "Revision:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
53     AT_CGSN, "AT+CGSN", "+CGSN:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
54
55     //module
56     AT_NCCID, "AT+NCCID", "+NCCID:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
57
58     //protocols
59     AT_QREGSWT, "AT+QREGSWT", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
60 }
61
62 //CEREG?
63 AT_CEREG:
64 AT_PROCESS(current_at_module.at_send(AT_CEREG, "?", AT_CMD_MODE_GET, p));
65 AT_PROCESS_WAIT(BC28_event, 1000);
66 AT_PROCESS_TRY(3);
67 AT_PROCESS_TIMEOUT(ret);
68 if( rsp_msg->result == AT_RSP_DATA )
69 {
70     if( strstr(rsp_msg->data, "+CEREG:0,1") != 0 )
71     {
72         nb_log("CEREG OK\r\n");
73     }
74     else
75     {
76         THREAD_OS_DELAY(2000);
77         goto AT_CEREG;
78     }
79 }
80 else if( rsp_msg->result == AT_RSP_ERROR )
81 {
82 }
83 }

```

##### b) 设备初始化

AT 指令: AT+NCDP=<ip\_addr>[,<port>]<CR>

修改 userconfig.h 中的宏定义 CTIOT\_INIT\_IP、CTIOT\_INIT\_PORT、CTIOT\_REG\_LIFETIME (IP 地址、端口号和 lifetime);

```

BC28_thread.c
54 //module
55 AT_NCCID, "AT+NCCID", "+NCCID:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
56
57 //protocols
58 AT_QREGSWI, "AT+QREGSWI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
59 AT_NCDP, "AT+NCDP", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
60 AT_QCFG, "AT+QCFG", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
61 AT_REG, "AT+QLWSREGIND", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
62 AT_SENDDATA, "AT+QLWULDATAEX", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
63 AT_STATUS, "AT+NMSTATUS", "+NMSTATUS:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
64 AT_NNMI, "AT+NNMI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
65 AT_NSMI, "AT+NSMI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
66 AT_QLWULDATASTATUS, "AT+QLWULDATASTATUS", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
67
158 }
159 }
160
161 //*****lwm2m api*****
162 static int8_t at_config_IP_port(char *serverIP, uint16_t port)
163 {
164   int8_t ret;
165   char buff[64]={0};
166
167   sprintf(buff, "%s,%d\r\n",serverIP, port);
168   ret = current_at_module.at_send(AT_NCDP, buff, AT_CMD_MODE_SET, current_at_module.module->thread);
169
170   return ret;
171 }
172
456 if( rsp_msg->result == AT_RSP_ERROR )
457 {
458 }
459
460 //configure server and port
461 AT_PROCESS(at_config_IP_port(CTIOT_INIT_IP, CTIOT_INIT_PORT)); //set
462 AT_PROCESS_WAIT(BC28_event,1000);
463 AT_PROCESS_TRY(3);
464 AT_PROCESS_TIMEOUT(ret);
465 if( rsp_msg->result == AT_RSP_ERROR )
466 {
467 }
  
```

### c) 设备登录

AT指令: **AT+QLWSREGIND=0\r\n**

```

BC28_thread.c
59 AT_NCDP, "AT+NCDP", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
60 AT_QCFG, "AT+QCFG", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
61 AT_REG, "AT+QLWSREGIND", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
62 AT_SENDDATA, "AT+QLWULDATAEX", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
63 AT_STATUS, "AT+NMSTATUS", "+NMSTATUS:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
64
267 static int8_t at_lwm2m_register(void)
268 {
269   int8_t ret;
270
271   ret = current_at_module.at_send(AT_REG, "0", AT_CMD_MODE_SET, current_at_module.module->thread);
272
273   return ret;
274 }
275
484 //login
485 AT_PROCESS(at_lwm2m_register()); //set
486 AT_PROCESS_WAIT(BC28_event,1000);
487 AT_PROCESS_TRY(3);
488 AT_PROCESS_TIMEOUT(ret);
489 if( rsp_msg->result == AT_RSP_ERROR )
490 {
491 }
492
  
```

### d) 查询订阅是否完成

AT指令: **AT+NMSTATUS?\r\n**

```

BC28_thread.c
62 AT_SENDDATA, "AT+QLWULDATAEX", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
63 AT_STATUS, "AT+NMSTATUS", "+NMSTATUS:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
64 AT_NNMI, "AT+NNMI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
65 AT_NSMI, "AT+NSMI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
66 AT_QLWULDATASTATUS, "AT+QLWULDATASTATUS", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
67
  
```

```

BC28_thread.c
285
286 static int8_t at_lwm2m_get_status(void)
287 {
288     int8_t ret;
289
290     ret = current_at_module.at_send(AT_STATUS, "?", AT_CMD_MODE_GET, current_at_module.module->thread);
291
292     return ret;
293 }
294

BC28_thread.c
493 //get login status
494 login_status:
495     AT_PROCESS(at_lwm2m_get_status()); //get
496     AT_PROCESS_WAIT(BC28_event,1000);
497     AT_PROCESS_TRY(3);
498     AT_PROCESS_TIMEOUT(ret);
499     if( rsp_msg->result == AT_RSP_DATA )
500     {
501         if( strcmp(rsp_msg->data, "+NMSTATUS:REGISTERED") == 0 || strcmp(rsp_msg->data, "+NMSTATUS:REGISTERING") == 0 )
502         {
503             THREAD_OS_DELAY(2000);
504             goto login_status;
505         }
506         else if( strcmp(rsp_msg->data, "+NMSTATUS:MO_DATA_ENABLED") == 0 )
507         {
508             current_at_module.module->send_data_enable = 1;
509         }
510     }
511     else if( rsp_msg->result == AT_RSP_ERROR )
512     {
513     }
514     nb_log("login success\r\n");
515     login_flag = 1;

```

### e) 发送数据

AT指令: AT+QLWULDATAEX=<length>, <data> , <mode>[, <seq\_num>]<CR>

开发者发送的数据编码格式请参考“[附 3 数据编解码规则](#)”中的编解码规则详细描述，根据业务逻辑，修改 nb\_thread.c 中的拼接 AT 指令代码（本样例中主要是检测红外传感器状态控制 LED，马达，红外传感器的状态数据上报），Data 字段中，SEND\_DATA（02）为数据上报的固定字段，占一个字节，dataSetID 为平台配置的服务 ID，占两个字节，data\_len 为后面的 payload 长度，占两个字节，实际 payload 为三个传感器的状态数据。

```

app_aep_profile.c
135 /******
136
137 static void send_sensor_data(void)
138 {
139     int8_t ret;
140     sensor_report reportFlag;
141     AepString reportstr;
142
143     getStatus(BSP_ALL, &BSP_status);
144
145     //fill sensor data
146     reportFlag.sensor_data[0] = BSP_status.IR_status;
147     reportFlag.sensor_data[1] = BSP_status.led2_status<<4 | BSP_status.led1_status;
148     reportFlag.sensor_data[2] = BSP_status.motor_status;
149
150     reportstr = codeDataReportByIdentifierToStr("sensor_report", &reportFlag);
151     app_log(reportstr.str);
152
153     ret = send_msg(reportstr.str, reportstr.len, SEND_MODE_CON);
154     if( ret < 0 )
155     {
156         app_log("send sensor data err\r\n");
157     }
158
159     free(reportstr.str);
160 }

BC28_thread.c
61 AT REG, "AT+QLWSREGIND", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
62 AT SENDDATA, "AT+QLWULDATAEX", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
63 AT STATUS, "AT+NMSTATUS", "+NMSTATUS:", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_get_rsp, NULL,
64 AT NNMI, "AT+NNMI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,
65 AT NSMI, "AT+NSMI", "", "OK", "ERROR", 0, {0, AT_RSP_ERROR, ""}, at_set_rsp, NULL,

```



```

BC28_thread.c
295 int8_t at_BC28_send(char *data, uint32_t mode, uint8_t messageID, void *thread)
296 {
297     int8_t ret;
298     char buff[512]={0,};
299
300     sprintf(buff, "%d,%s,0x%04x,%d", strlen(data)/2,data, mode,messageID);
301     ret = current_at_module.at_send(AT_SENDDATA, buff, AT_CMD_MODE_SET, thread);
302
303     return ret;
304 }

```

#### f) 指令接收与应答

下行报文主要通过 app\_aep\_profile.c 中函数 decode\_aep\_data(char \*data)进行处理。

开发者接收的数据编码格式请参考“[附 3 数据编解码规则](#)”中的编解码规则详细描述，本样例中的指令下发业务通过 BC28\_thread.c 中的 BC28\_rcv\_process 接收进程处理。首先解析数据，然后将处理结果拼接 AT 指令返回给平台。不同的服务通过服务 ID 来区分，需要与平台上配置的服务 ID 一一对应。指令处理完需要调用 AT+QLWULDATAEX 来发送指令的结果响应，Data 字段中，SEND\_RESPONSE(86)为指令响应的固定字段。

```

BC28_thread.c
521 PROCESS_THREAD(BC28_rcv_process, ev, pdata)
522 {
523     uint8_t ret;
524     rsp_msg_t *rsp_msg;
525     uint32_t os_event;
526
527     THREAD_BEGIN()
528
529     while(1)
530     {
531         THREAD_WAIT_EVENT(&ret,0);
532         if( os_event_pend_event(&BC28_rcv_event, &os_event, NULL) == OS_EVENT_OK )
533         {
534             if( (os_event & BC28_EVENT_RECVDATA) == BC28_EVENT_RECVDATA )
535             {
536                 rsp_msg = current_at_module.at_get_URC_result(AT_RECV);
537                 if( rsp_msg != NULL && rsp_msg->result == AT_RSP_OK )
538                 {
539                     char *p = strstr(rsp_msg->data, "+NNMI:");
540                     if( p != NULL )
541                     {
542                         p += strlen("+NNMI:");
543                         p = strstr(p, ",");
544                         decode_aep_data(p+1);
545                     }
546
547                     //clear data
548                     ret = current_at_module.at_get_cmd(AT_RECV);
549                     if( ret > 0 )
550                     {
551                         current_at_module.module->cmd_table[ret].rsp.data = "";
552                     }
553                 }
554             }
555         }
556     }
557 }

```

```

app_aep_profile.c
204
205 void decode_aep_data(char *data)
206 {
207     int8_t ret;
208     AepCmdData result;
209
210     result = decodeCmdDownFromStr(data);
211     if( result.code == AEP_CMD_SUCCESS )
212     {
213         if( strcmp(result.serviceIdentifier, "motor_control") == 0 )
214         {
215             motor_control_ack ack;
216             AepString rsp;
217             uint8_t value;
218
219             value = *((uint8_t *)result.data);
220             app_log("get value:%d\r\n",value);
221             if( value <= 1 )
222             {
223                 setStatus(MOTOR1, value);
224             }
225
226             ack.control_int = *((int *)result.data);
227             ack.taskId = result.taskId;
228             rsp = codeDataReportByIdentifierToStr("motor_control_ack", &ack);
229             ret = send_msg(rsp.str, rsp.len, SEND_MODE_CON);
230             if( ret < 0 )
231             {
232                 app_log("send motor_control_ack err\r\n");
233             }
234             free(rsp.str);
235         }
236         else if( strcmp(result.serviceIdentifier, "report_period_set") == 0 )
237         {
238             report_period_set_ack ack;

```

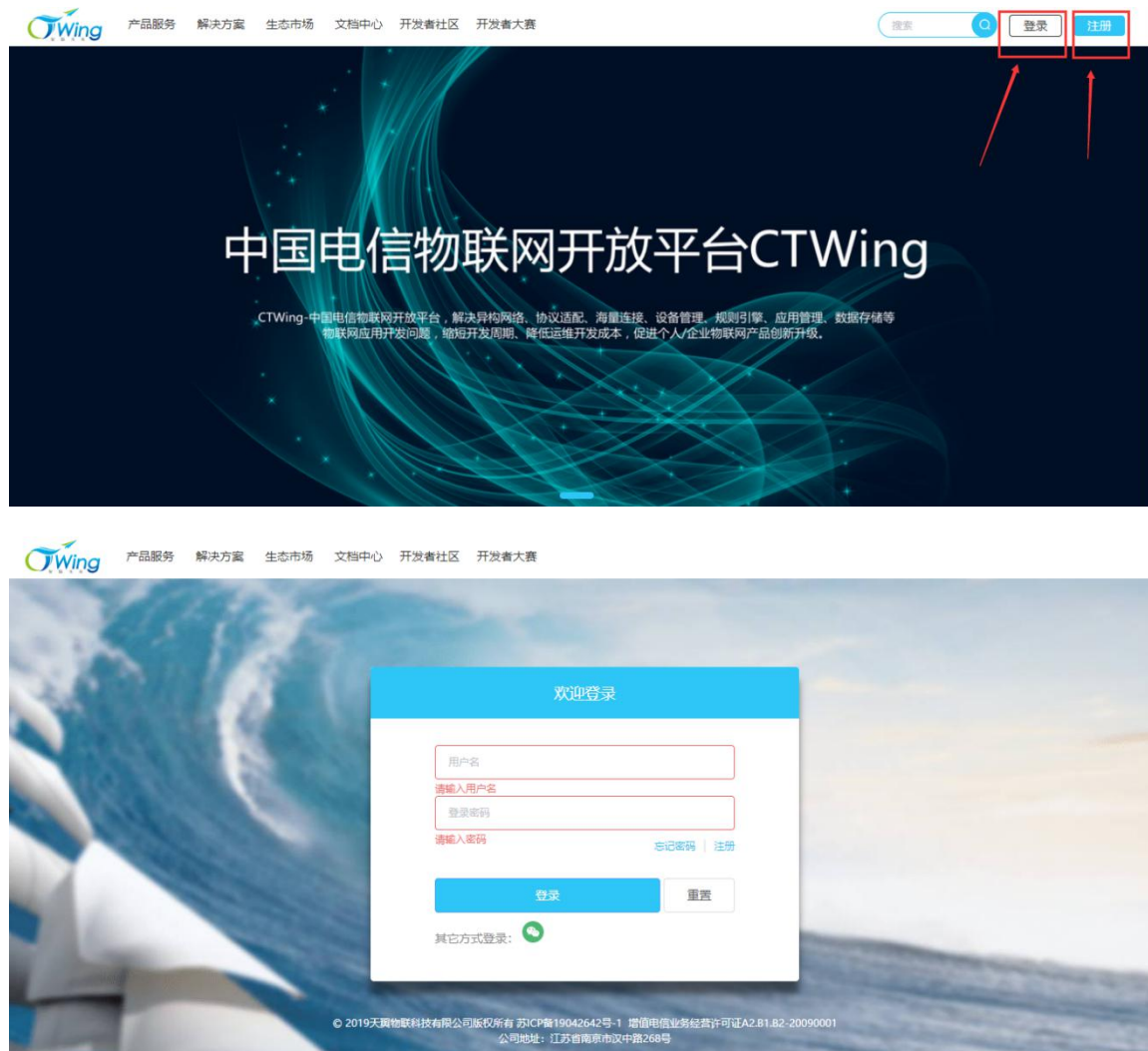
#### g) 其他业务操作请参考 AT 指令集说明文档



## 附 1 中国电信物联网开放平台操作步骤

### 1、用户登录平台

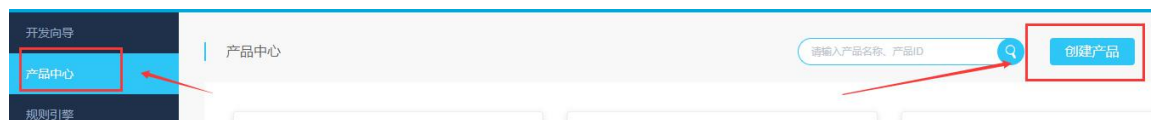
网站：<https://www.ctwing.cn/>



如果有账号，直接输入用户名和密码登录，否则先注册账号后再重新登录。

### 2、在平台创建产品和设备

选择产品中心，点击创建产品，弹出创建产品信息框；



创建产品

\* 产品名称

请输入产品名称

\* 产品分类

请选择

请选择

请选择

\* 节点类型

设备

网关

\* 接入方式

选择接入方式

产品描述

输入产品描述

确定

取消

产品信息如下：

创建产品

\* 产品名称

NB终端业务场景测试

\* 产品分类

智慧水利

水利设备

自动化CTU

\* 节点类型

设备

网关

\* 接入方式

设备直连

\* 网络类型

NB-IoT

\* 通信协议

LWM2M

\* 通信协议

LWM2M

\* 认证方式

IMEI认证

\* Endpoint格式

imei

\* 是否是DTLS方式

是

否

\* 是否透传

是

否

\* 消息格式

紧凑型二进制

\* 省电模式

DRX

产品描述

输入产品描述

添加设备如下：

< NB终端业务场景测试

产品概况

服务定义

设备管理

设备信息 / 群组管理

请输入设备名称、设备ID、IMEI

设备名称

设备ID

添加设备

\* 设备名称

NB终端A01

\* IMEI号

867726038384824

是否开启自动订阅

是

否

确定

取消

添加设备

批量删除

导入

导出

时间

最后离线时间

状态

操作

### 3、在平台创建属性和服务

属性定义：

产品概况 **服务定义** 设备管理 事件上报 数据查看 指令下发日志 订阅管理 远程升级管理

**属性列表** 服务列表 导入模板下载 查看物模型 导入物模型 导出 生成设备侧编解码

请输入属性名称、属性标识 新增属性

属性名称	属性标识	属性ID	数据类型	数据定义	操作
配置整形	set_int	1	整型	取值范围：0-100	 
控制整形	control_int	2	整型	取值范围：0-1	 
温度数据	temperature_data	3	float浮点型	取值范围：1-100	 
湿度数据	humidity_data	4	float浮点型	取值范围：1-100	 
传感器数据	sensor_data	5	定长binary	长度：3字节	 

属性名称：配置整形

\* 属性名称  \* 属性标识

\* 数据类型 ☒ 整型 ☐ 定长字符串 ☐ 定长binary ☐ 无符号整型 ☐ 变长字符串 ☐ 变长binary

☐ 时间戳 ☐ float浮点型 ☐ double浮点型 ☐ 枚举型

\* 长度(字节)  \* 单位

\* 最小值  \* 最大值

属性名称：控制整形

\* 属性名称  \* 属性标识

\* 数据类型 ☒ 整型 ☐ 定长字符串 ☐ 定长binary ☐ 无符号整型 ☐ 变长字符串 ☐ 变长binary

☐ 时间戳 ☐ float浮点型 ☐ double浮点型 ☐ 枚举型

\* 长度(字节)  \* 单位

\* 最小值  \* 最大值

属性名称：温度数据

* 属性名称	* 属性标识
<input type="text" value="温度数据"/>	<input type="text" value="temperature_data"/>
* 数据类型	
<input type="radio"/> 整型 <input type="radio"/> 定长字符串 <input type="radio"/> 定长binary <input type="radio"/> 无符号整型 <input type="radio"/> 变长字符串 <input type="radio"/> 变长binary <input type="radio"/> 时间戳 <input checked="" type="radio"/> float浮点型 <input type="radio"/> double浮点型 <input type="radio"/> 枚举型	
* 长度(字节)	* 单位
<input type="text" value="4字节"/>	<input type="text" value="无"/>
* 最小值	* 最大值
<input type="text" value="1"/>	<input type="text" value="100"/>

属性名称：湿度数据

* 属性名称	* 属性标识
<input type="text" value="湿度数据"/>	<input type="text" value="humidity_data"/>
* 数据类型	
<input type="radio"/> 整型 <input type="radio"/> 定长字符串 <input type="radio"/> 定长binary <input type="radio"/> 无符号整型 <input type="radio"/> 变长字符串 <input type="radio"/> 变长binary <input type="radio"/> 时间戳 <input checked="" type="radio"/> float浮点型 <input type="radio"/> double浮点型 <input type="radio"/> 枚举型	
* 长度(字节)	* 单位
<input type="text" value="4字节"/>	<input type="text" value="无"/>
* 最小值	* 最大值
<input type="text" value="1"/>	<input type="text" value="100"/>

属性名称：传感器数据

* 属性名称	* 属性标识
<input type="text" value="传感器数据"/>	<input type="text" value="sensor_data"/>
* 数据类型	
<input type="radio"/> 整型 <input type="radio"/> 定长字符串 <input checked="" type="radio"/> 定长binary <input type="radio"/> 无符号整型 <input type="radio"/> 变长字符串 <input type="radio"/> 变长binary <input type="radio"/> 时间戳 <input type="radio"/> float浮点型 <input type="radio"/> double浮点型 <input type="radio"/> 枚举型	
* 长度(字节)	* 单位
<input type="text" value="3"/>	<input type="text" value="无"/>

服务定义：

产品概况 **服务定义** 设备管理 事件上报 数据查看 指令下发日志 订阅管理 远程升级管理

属性列表 **服务列表** 导入模板下载 查看物模型 导入物模型 导出 生成设备侧编解码

请输入服务名称、服务ID、服务标识 请选择服务类型 **新增服务**

服务ID	服务标识	服务名称	服务类型	参数列表	操作
2	temperature_humidity...	温湿度上报	数据上报		
1	sensor_report	传感器上报	数据上报		
8001	motor_control	电机控制	指令下发		
9001	motor_control_ack	电机控制响应	指令下发响应		
9002	report_control_ack	上报控制响应	指令下发响应		
9003	report_period_set_ack	参数配置响应	指令下发响应		
8002	report_control	上报控制	指令下发		
8003	report_period_set	参数配置	指令下发		

新增服务操作：

**新增参数**

\* 属性名称: 传感器数据

属性标识: sensor\_data

数据类型: 定长binary

单位: 无

长度(字节): 3

确定 取消

服务名称：温湿度上报

\* 服务类型: 数据上报

\* 服务名称: 温湿度上报

\* 服务ID: 2

\* 服务标识: temperature\_humidity\_report

\* 参数列表:

- 参数名称: 温度数据
- 参数名称: 湿度数据

描述: 请输入描述

服务名称：传感器上报



<b>* 服务类型</b> 数据上报	<b>服务ID</b> 1
<b>* 服务名称</b> 传感器上报	<b>* 服务标识</b> sensor_report
<b>* 参数列表</b> 1. 参数名称: 传感器数据	
<b>描述</b> 请输入描述	

服务名称: 电机控制

<b>* 服务类型</b> 指令下发	<b>服务ID</b> 8001
<b>* 服务名称</b> 电机控制	<b>* 服务标识</b> motor_control
<b>* 参数列表</b> 1. 参数名称: 控制整形	

服务名称: 电机控制响应

<b>* 服务类型</b> 指令下发响应	<b>服务ID</b> 9001
<b>* 服务名称</b> 电机控制响应	<b>* 服务标识</b> motor_control_ack
<b>* 参数列表</b> 1. 参数名称: 控制整形	

服务名称: 上报控制响应

<b>* 服务类型</b> 指令下发响应	<b>服务ID</b> 9002
<b>* 服务名称</b> 上报控制响应	<b>* 服务标识</b> report_control_ack
<b>* 参数列表</b> 1. 参数名称: 控制整形	

服务名称: 参数配置响应

\* 服务类型  
指令下发响应

服务ID  
9003

\* 服务名称  
参数配置响应

\* 服务标识  
report\_period\_set\_ack

\* 参数列表  
1. 参数名称: 配置整形

服务名称: 上报控制

\* 服务类型  
指令下发

服务ID  
8002

\* 服务名称  
上报控制

\* 服务标识  
report\_control

\* 参数列表  
1. 参数名称: 控制整形

服务名称: 参数配置

\* 服务类型  
指令下发

服务ID  
8003

\* 服务名称  
参数配置

\* 服务标识  
report\_period\_set

\* 参数列表  
1. 参数名称: 配置整形

使用物模型导入导出功能快速创建属性和服务

导入物模型:

先把获取到其它产品的物模型保存到电脑上, 然后按照如下所示操作步骤导入物模型文件。

产品概况
服务定义
设备管理
事件上报
数据查看
指令下发日志
订阅管理
远程升级管理

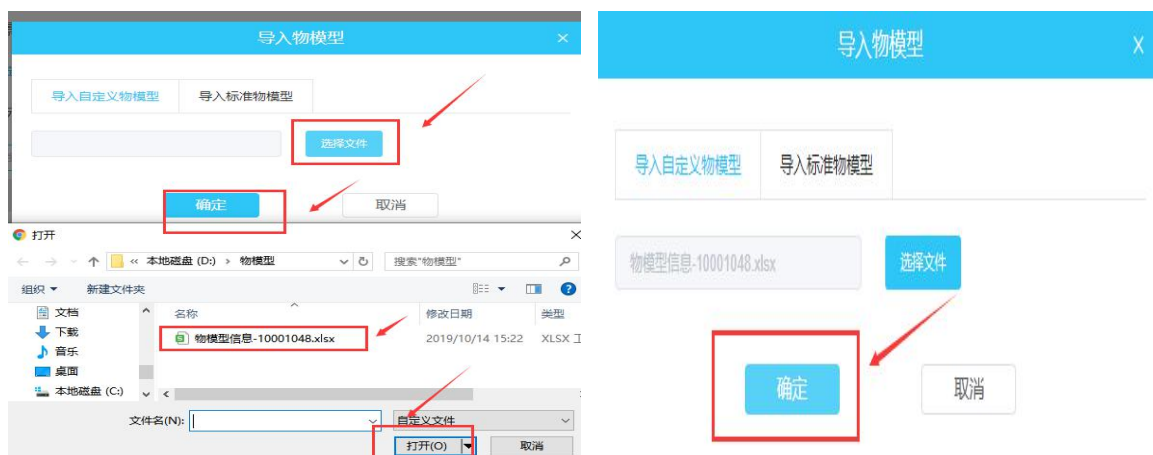
属性列表 / 服务列表

导入模板下载
查看物模型
导入物模型
导出
生成设备侧编解码

请输入属性名称、属性标识

新增属性

属性名称	属性标识	属性ID	数据类型	数据定义	操作
暂无数据					



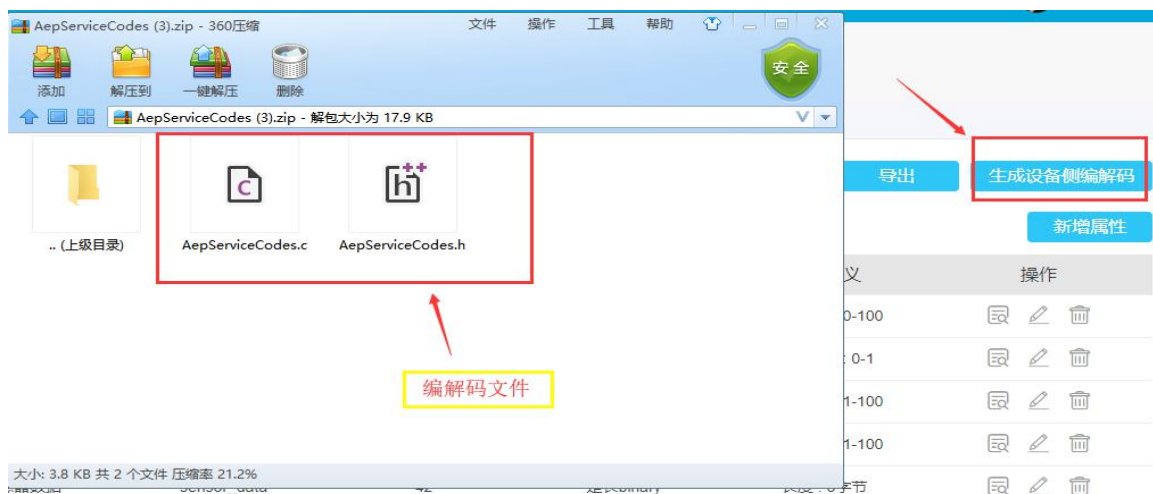
### 导出物模型：

导出后，物模型文件自动保存到浏览器的默认下载路径的目录下。



### 生成设备侧编解码：

生成编解码文件后，共有 2 个文件会保存到浏览器的默认下载路径的目录下。



## 4、从平台获取终端接入地址、接入端口、服务 ID

< NB终端业务场景测试

产品概况 服务定义 设备管理 事件上报 数据查看 指令下发日志 订阅管理 远程升级管理

产品ID	10018519	产品名称	NB终端业务场景测试
产品分类	智慧城市 -> 能源管理 -> 水表	认证方式	IMEI认证
省电模式	DRX	节点类型	设备
接入方式	设备直连	网络类型	NB-IoT
通信协议	LWM2M	Endpoint格式	imei
设备总数	1个	在线设备	0个
激活设备	0个	消息格式	紧凑型二进制
是否是DTLS方式	否	MasterKey	577e652a839444c787edd68f9a336cdb
设备接入IP端口	221.229.214.202:5683	创建时间	2019-08-27 16:55:07

产品概况 服务定义 设备管理 事件上报 数据查看 指令下发日志 订阅管理 远程升级管理

属性列表 / 服务列表

导入模板下载 查看物模型 导入物模型 导出 生成设备侧编解码

请输入服务名称、服务ID、服务标识 请选择服务类型 新增服务

服务ID	服务标识	服务名称	服务类型	参数列表	操作
2	temperature_humidity...	温湿度上报	数据上报		
1	sensor_report	传感器上报	数据上报		
8001	motor_control	电机控制	指令下发		
9001	motor_control_ack	电机控制响应	指令下发响应		
9002	report_control_ack	上报控制响应	指令下发响应		
9003	report_period_set_ack	参数配置响应	指令下发响应		
8002	report_control	上报控制	指令下发		
8003	report_period_set	参数配置	指令下发		

## 附 2 深入开发参考

### 1. log 配置

该系统提供日志打印接口，默认打开 UART 打印。打开 os\_config.h，修改宏 THREAD\_OS\_LOG，0 为禁用，2 为 UART 打开。如果不需要打印，只需修改为 0。

```

33
34
35 #define THREAD_OS_LOG 2u /* ----- LOG MANAGEMENT ----- */
36 /* Disable (0) ; UART (2) */

```

### 2. thread-os 介绍

thread-os 是基于 poll 线程模型设计的一块轻量级线程操作系统。对于该系统的使用可以参考用户线程（BSP\_thread 等）。

用户创建自己的线程需要以下步骤：

### 2.1 声明线程

```
THREAD_INIT(xxx_process, "xxx thread");
```

### 2.2 线程实体

```
PROCESS_THREAD(xxx_process, ev, pdata)
{
    THREAD_BEGAIN()
    While(1)
    {
        //user code
        THREAD_OS_DELAY(xxxx);
    }
    THREAD_END()
}
```

### 2.3 启动线程

```
thread_os_register(&xxx_process, NULL);
```

## 3. 软定时器的使用

该 RTOS 实现的软定时器驱动源来自与 systick 中断，用户可以自行根据具体的项目配置。该定时器需要根据自身系统使用的定时器总数进行配置。默认最大可分配个数为 20 个。定时器一共有两种工作模式，分别是连续模式和一次模式。在单独使用软定时器时，用户需要注册超时回调函数。如果配合线程使用，则不需要注册回调函数。以下是使用步骤：

### 3.1 声明定时器句柄

```
static soft_timer_t *nb_timer;
```

### 3.2 注册定时器句柄，选择工作模式，以及是否注册回调函数

OS\_ONE\_SHOT\_MODE 为单次模式

OS\_PERIODIC\_MODE 为多次模式

```
soft_timer_register(&nb_timer, OS_PERIODIC_MODE, CALLBACK, NULL)
```

### 3.3 启动定时器

```
soft_timer_start(nb_timer, delaysms)
```

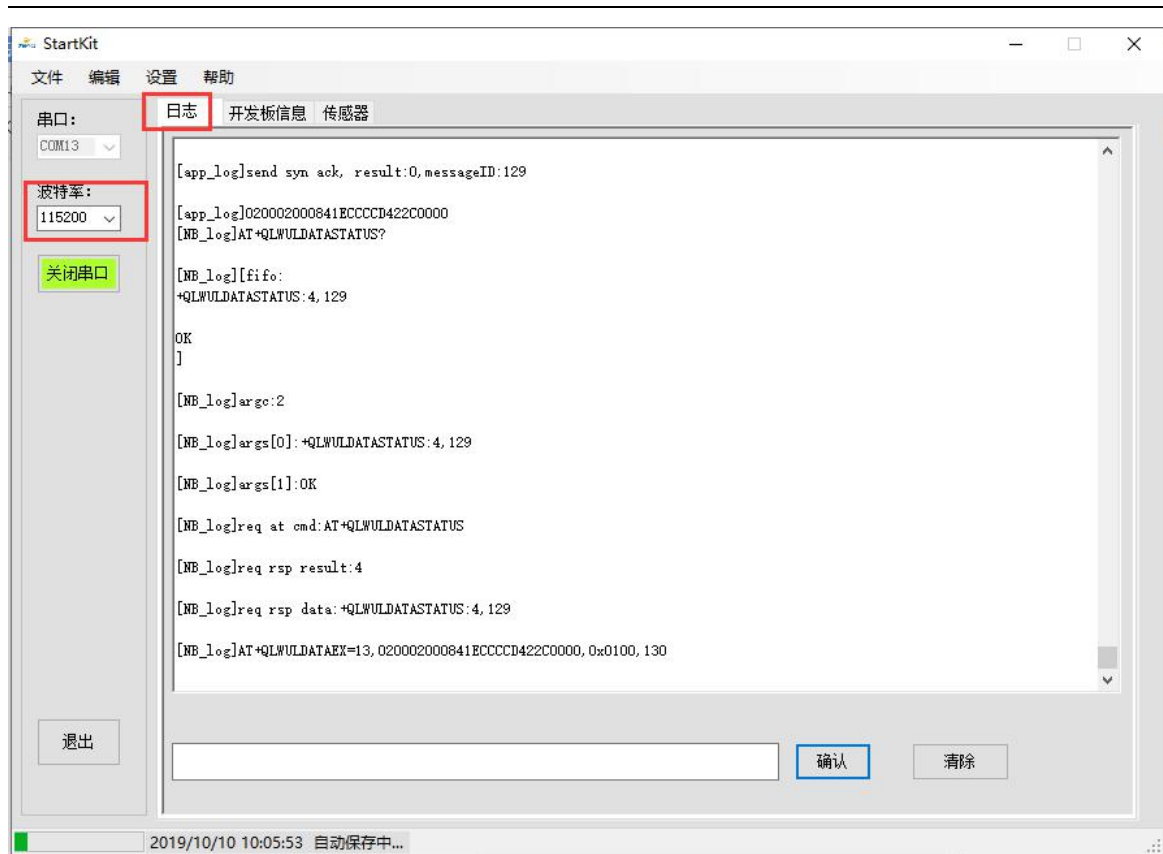
### 3.4 关闭定时器

```
soft_timer_stop(nb_timer)
```

## 4. 终端串口调试工具 Shell 使用（以 UART port 讲解）

Shell Port 默认使用串口。连接好串口之后，打开串口调试工具，配置串口波特率为 115200，其它功能使用按照下图所示操作即可。







重启开发板，串口工具会打印出 log 信息。在输入框中输入？敲回车键，会打印当前系统支持哪些命令。如下：

```
? : dispaly current cmd
help: dispaly current cmd
reset: restart system
ps: display current user thread
suspend:<x> x:suspend pid
resume:<x> x:resume pid
restart:<x> x:restart pid
clear: Clear screen, reposition cursor to top left
settime:<x> x:(decimal numbers)unix(posix)timestamp
gettime: display unix(posix)timestamp and UTC time
led:<x>,<y> x:device id ; y:mode
motor:<x>,<y> x:device id ; y:mode
infrared:<x>,<y> x:device id ; y:mode
humitemp:<x>,<y> x:device id ; y:mode
fifo:<x> x=1 fifo in,x=2 fifo out
AT:<x> x= Manufacturer of module 'at string'
board: nb_starter_kit
module: NB module information
```

部门命令解释如下：

reset	系统复位
ps	显示当前注册的线程状态
suspend	用于挂起已经启动的线程，其中 PID 为线程名字
resume	用于恢复被挂起的线程
restart	用于重启线程
led	控制 led 灯打开和关闭
motor	控制电机向前，向后，停止
infrared	获取红外状态
humitemp	获取温湿度数据

### 附 3 数据编解码规则

业务数据上报的 Data 字段编码格式：

CMDType	DatasetID	Payload
---------	-----------	---------

参数名称	参数类型	类型说明	参数说明
CMDType	Unsigned Integer ( 1 )	必选定长	报文类型，固定为0x02
DatasetID	Unsigned Integer ( 2 )	必选定长	服务ID
Payload	Binary	必选变长	上报数据（二进制格式）

- CMDType：必选字段，固定1Byte，值为0x02
- DatasetID：必选字段，固定2Byte，填写平台分配的服务ID
- Payload：必选变长字段，类型为Binary，按照服务定义的格式编写

业务指令下发、指令下发响应的 Data 字段编码格式：

CMDType	DatasetID	TaskID	Payload
---------	-----------	--------	---------

参数名称	参数类型	类型说明	参数说明
CMDType	Unsigned Integer ( 1 )	必选定长	报文类型，下发指令固定为0x06，指令响应固定为0x86
DatasetID	Unsigned Integer ( 2 )	必选定长	服务ID
Task ID	Unsigned Integer ( 2 )	必选定长	任务ID
Payload	Binary	必选变长	指令或指令响应数据（二进制格式）

- CMDType：必选字段，固定1Byte，下行指令该值为0x06，指令响应该值为0x86
- DatasetID：必选字段，固定2Byte，填写平台分配的服务ID
- TaskID：必选字段，固定2Byte，由平台分配，用于关联下发指令和回复响应，指令和响应中的TaskID必须一致
- Payload：必选变长字段，类型为Binary，按照服务定义的格式编写

注：变长参数包含2Byte的长度字段。

**Payload 的编码格式：**

Payload Length	Parameter 1	Parameter 2	.....
----------------	-------------	-------------	-------

- Payload Length: 固定2Byte，表示Parameters的总长度；
- Parameter1、Parameter2等需要按照对应服务的参数列表的顺序填写，Parameter的类型要与服务中定义的参数类型一致。