

使用说明及测试指导

HIS 脚本

Rev1.0

1. 脚本介绍.....	4
2. 语法介绍.....	5
2.1. 数据类型.....	5
2.2. 运算符.....	5
2.3. 数据运算.....	5
2.4. IF 语句.....	6
2.5. FOR 语句.....	6
2.6. WHILE 语句.....	7
2.7. 保留字.....	7
2.8. 外部接口.....	7
2.9. 访问工控设备部分.....	7
2.10. 从工控设备输入部分.....	8
2.11. 函数.....	9
2.12. 变量作用域.....	9
2.13. FLASH 参数说明.....	10
3. HisUI 说明.....	11
4. HFPackTools 说明.....	12
5. 脚本升级.....	13
6. 脚本控制 E10 产品 GPIO.....	15
6.1. 功能需求.....	15
6.2. 功能实现.....	15
6.3. HisUI 工具测试.....	16

6.4.	脚本转换及设备加载脚本.....	17
6.5.	E10 产品测试.....	17
附录 A.	心跳包功能案例.....	18
A.1.	功能需求.....	18
A.2.	功能实现.....	18
A.3.	脚本转换及设备加载脚本.....	19
A.4.	产品测试.....	19
附录 B.	注册包功能案例.....	20
B.1.	功能需求.....	20
B.2.	功能实现.....	20
附录 C.	FLASH 参数使用案例.....	22
C.1.	功能需求.....	22
C.2.	功能实现.....	22
C.3.	产品测试.....	23
C.4.	其他说明.....	24

版本历史：

2017-06-19 初稿

2017-08-24 增加了 flash 参数说明及使用案例

1. 脚本介绍

汉枫 I.O.T 脚本 (HF IOT Script , 缩写 HIS)。注意版本必须为 1.09i 及以上。用于汉枫 IOT 工控设备的数据转换, 把脚本下载到工控设备后, 可自动完成下列功能:

- 自动定时发数据给 UART 或 Socket, 作为 Modbus 主站功能。
- 从 UART 或 Socket 收到数据后, 按脚本转换后转发

本文档所用的工具和样例脚本可从如下地址下载获取:

<http://pan.baidu.com/s/1qY0QPcS>

或者:

<https://www.dropbox.com/s/phnpvbgve43n86y/HIS%E8%84%9A%E6%9C%AC%E5%B7%A5%E5%85%B7.rar?dl=0>

目前支持的产品列表如下, 下文以 Eport-E10 或者 HF2211 等产品为例介绍。

超级网口类:

产品	Eport-E10	Eport-E30	Eport Pro-EP10	Eport Pro-EP20
				

以太网串口服务器类:

产品	HF5111A	HF5111B	HF5142
			

Wi-Fi 串口服务器类(W10 和 W20 分别是 HF2211 和 HF2221 内用的模块):

产品	Wport-W10	Wport-W20	HF2211	HF2221
				

2. 语法介绍

2.1. 数据类型

HIS 脚本支持三种数据类型：整数、浮点数、字符串。

变量使用时，自动定义和赋值。如：

```
a=10
```

定义一个整数 a 并赋值为 10

```
f=1.04
```

定义一个浮点数 f 并赋值为 1.04

```
s=" 1234" 或 s=[0x31,0x32,0x33,0x34]
```

定义一个字符串 s 并赋值

注：HIS 语法是大小写敏感的。

2.2. 运算符

HIS 支持运算符及运算符的优先级如下：

优先级从高到低	运算符
0	(,), [,], ..
1	*, /
2	+, -
3	>>, <<, &,
4	<, <=, >, >=, ==, !=
5	&&,
6	=

注：脚本不支持 ++, --, +=, -= 等操作

2.3. 数据运算

整数和浮点数支持 “+,-,*,/” 运算，整数支持位操作，如：<<,>>,&,|。

比较运算时，源操作数和目的操作数的数据类型必须同类（整数和浮点数视为同类，字符串只能和字符串进行比较）。

赋值操作时，同时会涉及到数据类型的转换。如：

a 为整数，f 为浮点数，s 为字符串

a=f 把 f 赋值给 a，同时，a 的类型自动转为浮点数（如果: f=a，不涉及类型转换）

a=s a 会转为字符串，并赋值为 s 的内容

s=f s 转为浮点数，并赋值

字符串的加法运算：

如：s1=" 123" , s2=" 456"

s3=s1+s2 结果 s3=" 123456"

字符串比较运算：字符串只支持"=="和"!="运算

字符串函数，列举如下：

如：s1=" 12345678" , a1=0x010203

a=s1.length() 结果 a 为整数 8

b=s1.charAt(2) 结果 b 为整数 2，参数为第几个字符

c=s1.substring(1,5) 结果 c 为字符串," 2345"

d=s1.replaceAt(1,' 5') 结果 d 为字符串" 15345678"

注：substring()函数，前一个参数为开始位置，后一个为结束位置，位置从零开始计数。字符串中包括开始位置的数据，但不包括结束位置的数据。

d=a1.toString(0,3) 结果 d 为字符串：[0x03,0x02,0x01,0x00]

e=a1.toString(2,0) 结果 e 为字符串：[0x01,0x02,0x03]

f=a1.prtString() 结果 f 为字符串：" 66051" ，(0x010203 等于十进制的

66051)

g=s1.stringHex() 结果 g 为字符串：[0x12,0x34,0x56,0x78]

h=s2.hexString() 结果 h 为字符串：" 12345678"

i=s1.indexOf("234") 结果 i 为整数：1

注：toString ()函数的源操作数为整数，2 个参数分别为整数中 4 个字节的位置（所以参数的取值为>=0, <=3）。例子 1，d 为从 0 字节到 3 字节，例子 2，e 为从 2 字节到 0 字节

2.4. IF 语句

例如：

IF (a>1)

.....

ELSE

.....

END

可以没有 ELSE，如：

IF (a>1)

.....

END

2.5. FOR 语句

例如：

FOR i,0,5,1 // i 为变量，后面 3 个参数依次是：初始值，结束值和步进值

k=i

END

注：初始值，结束值和步进值都必须为整数，当变量等于结束值时结束循环，步进值可以是正数和负数

2.6. WHILE 语句

例如：

```
i=0
WHILE (i<10)
    i=i+1          不支持++,--,+=,-=等操作
END
```

2.7. 保留字

HIS 的保留字都为大写的字符，列举如下：

```
"TIMER","END","MAC","SOCKBEAT","SYSTIME","RECV","CONN","DISCONN","SOCK","UART","SEND","WAIT","IF","ELSE","FOR","WHILE","INT","FLOAT","INPUT","FUNCTION","RETURN","TRUE","FALSE","STRING"
```

2.8. 外部接口

外部接口为脚本与工控设备交互的接口，分为 2 部分，分别为从访问工控设备部分和从工控设备输入部分。

2.9. 访问工控设备部分

保留字 MAC，为读取工控设备的 MAC 地址，如：

工控设备的 MAC 地址为“ ACCF23001234”

```
mac= MAC          定义变量 mac 并赋值为“ ACCF23001234”
```

SYSTIME 为读取系统时间的秒数，如：

```
t= SYSTIME
```

SOCKBEAT 为读取某个 Socket 的心跳字符串，如：

```
bt= SOCKBEAT(netp)  读取名为 netp 的 Socket 的心跳字符串
```

SEND 向工控设备某接口发送数据，如：

```
SEND(UART,uart0," 1234" )
```

```
SEND(SOCK,netp," 3456" )
```

注：第一个参数为工控设备接口类型，第二个为工控设备接口名称，第三个为数据内容。

读写设备 GPIO，如：

```
P10= GETHW(GPIO, 10)    // 读取 GPIO Pin10 的值
```

```
SETHW(GPIO, OUT, 10, 1) // 设置 GPIO Pin10 为输出高
```

```
SETHW(GPIO, IN, 10)     // 设置 GPIO Pin10 为输入
```

Close Socket，如：

```
DISCONN(SOCK, netp)     // netp 断开连接
```

保留字 WANIP，为读取工控设备的 WAN 口 IP 地址，如：

工控设备的 WAN 口 IP 地址为“ 192.168.11.100”

ip= WANIP 定义变量 ip 并赋值为“ 192.168.11.100”

2.10. 从工控设备输入部分

从工控设备输入部分接口以程序块表示。列举如下：

➤ 定时接口

```
TIMER HeartBeat 1000
    SEND(UART,uart0," 1234" )
END
```

定时接口名称为“HeartBeat”，定时 1000ms，END 为程序块结束标记。

➤ 接收接口

```
RECV SOCK netp
    len=INPUT.length()
    IF (len>10)
        tmp=INPUT.subString(3,10)
    ELSE
        tmp=" ERR"
    END
    SEND(UART,uart0,tmp)
    RETURN(FALSE)
END
```

注：Socket netp 接收数据是自动调用，INPUT 为输入的参数，RETURN(FALSE)说明本函数结束时不需要按原来的 ROUT 定义转发（通常本函数已调用的 SEND 函数发送）。RETURN(TRUE)时，需要按 ROUT 定义转发，转发内容为 OUTPUT 的字符串。

➤ RECV 接口为串口接收时：

```
RECV UART uart0
.....
END
```

➤ Socket 连接接口：

```
CONN SOCK netp
.....
END
```

当 Socket netp 建立连接时，自动调用

➤ Socket 断开连接接口：

```
DISCONN SOCK netp
.....
END
```

当 Socket netp 断开连接时，自动调用

2.11. 函数

例如：

```
FUNCTION func1 (arg1, arg2)
    tmp1= arg1
    tmp2= arg2
    RETURN(tmp1)
END
```

函数调用：

```
TIMER xxx 1000
.....
tmp= func1( "1234" , "abcd" )
.....
END
```

2.12. 变量作用域

所有变量分为全局变量和局部变量，全局变量为程序块外定义的变量，全局变量在工控设备运行期间一直有效，可以被其他程序块访问，也可以被工控设备外部读取，如网页等。

局部变量以程序的外部接口运行为单位，一个外部接口运行期间，产生的中间变量都为局部变量，同名的变量视为同一个变量。如：

```
a=""
b=" 1234"
TIMER HB 1000
j=10
ret= func(a);           #这个访问不会出错，TIMER 内的 j 可以被 func 访问
END
```

```
CONN SOCK netp
ret=func(b)             #这个访问会出错，因为 func 内的 j 没有定义
END
```

```
FUNCTION func (arg)
    b=arg
    c=a
    IF (j==2)
        b=" abcd"
    END
END
```

上面程序中，a,b 为全局变量，arg 为局部变量，j 为局部变量，函数 func 被 TIMER 调用，此时 func 的局部变量与 TIMER 一致，所以可以调用的 j（不推荐用这种写法，因为其他的外部接口访问 func 时可能出错）。一个外部接口程序块运行结束时，会释放全部局部变所有上面程序中，两个 ret 变量为不同的变量。

2.13. Flash 参数说明

如果同一个脚本，但是针对不同的模块有部分参数需要不同，这种情况下可以使用 Flash 参数。即对于这些参数脚本在初始化时会写入 Flash 中，通过 IOTService 工具可以对这些参数进行修改，修改成功后，脚本会优先使用这些修改后的值。从而实现不同模块使用同一个脚本文件，但这些参数具有不同的初始值。

Flash 参数在脚本中定义为常量，不允许修改，即脚本中对这些参数的赋值操作时，会报错。

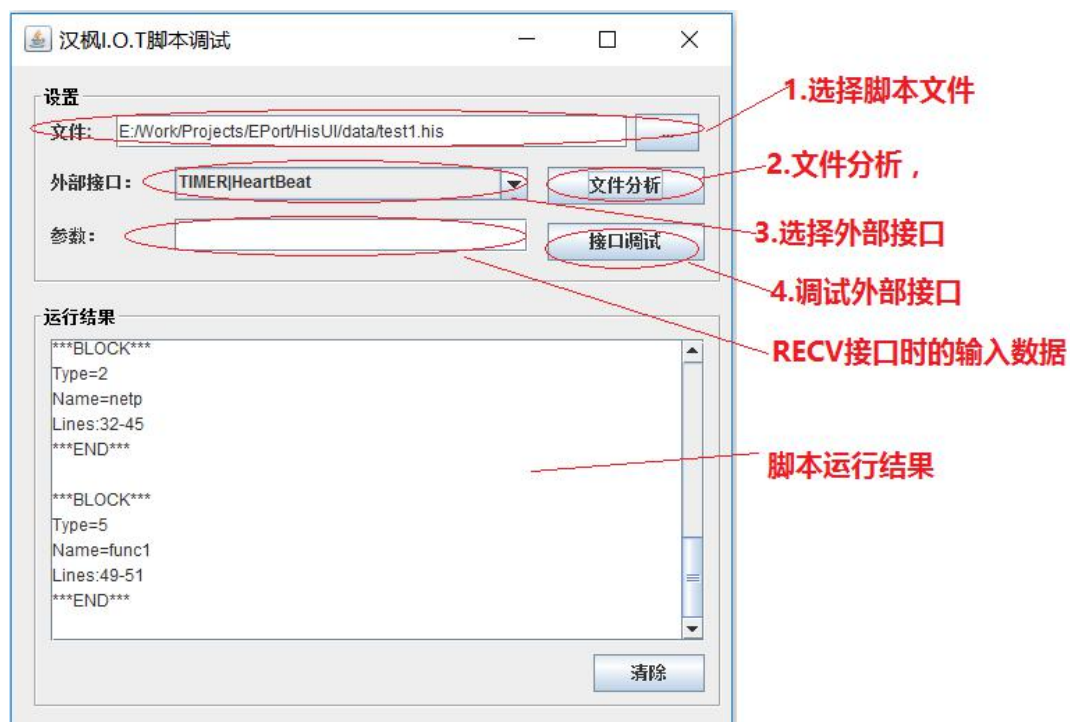
Flash 参数的定义：

```
FLASH(NUM)          iFls=10
FLASH(STRSTR) ssFls= "123456"           // 在工具中显示为 String
FLASH(STRHEX) shFls=[0x01,0x02,0x03]    // 在工具中显示为 16 进制
FLASH(BOOL)         bFls=1              // 取值为 0 或 1
```

另外，还有一个变量，FLASHMAGIC，这个变量告诉 Flash 参数什么时候需要修改。如果脚本的 FLASHMAGIC 和 Flash 中保存的 FLASHMAGIC 相同 则参数取值保持为 Flash 中；否则，脚本的 Flash 参数需要覆盖 Flash 中以保存的值。

3. HISUI 说明

HisUI为 HIS 脚本的可视化调试工具,可用于方便的调试脚本的功能,运行“HisUI.vbs”后加载文件进行测试。



注：本工具及后文介绍工具运行需要 JRE 运行环境，运行如下的指令确认 JRE 环境。

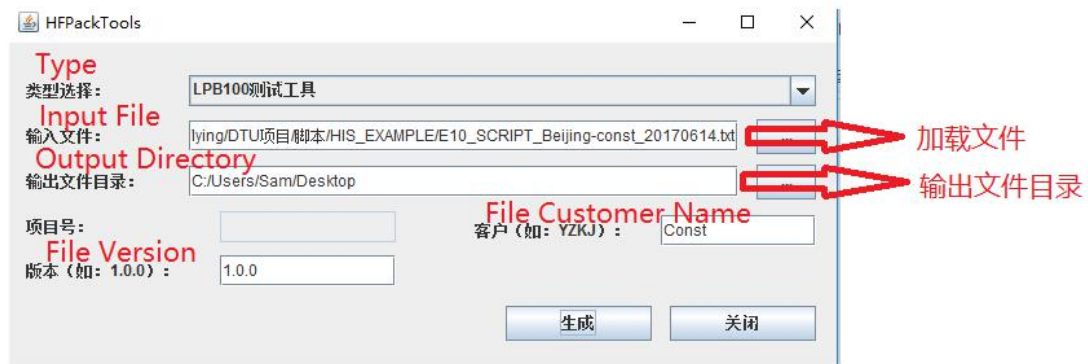
```
C:\Users\Sam>java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b15)
Java HotSpot(TM) Client VM (build 25.91-b15, mixed mode, sharing)
```

如未安装环境，可从如下链接中安装 JRE 环境。

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

4. HFPACKTOOLS 说明

HFPackTools 是脚本转换工具，如果脚本经 HisUI 工具已经测试通过，则实际把脚本载入到设备之前需要先做转换，转换的目的在于增加校验信息，设备检查脚本完整性，只有通过校验的脚本才可下载成功。运行“ HFPackTools.vbs” 打开工具，经过转换后，生成名为“ Eport_Const_1.0.0_20170618.his ”的文件，用于设备中加载脚本。



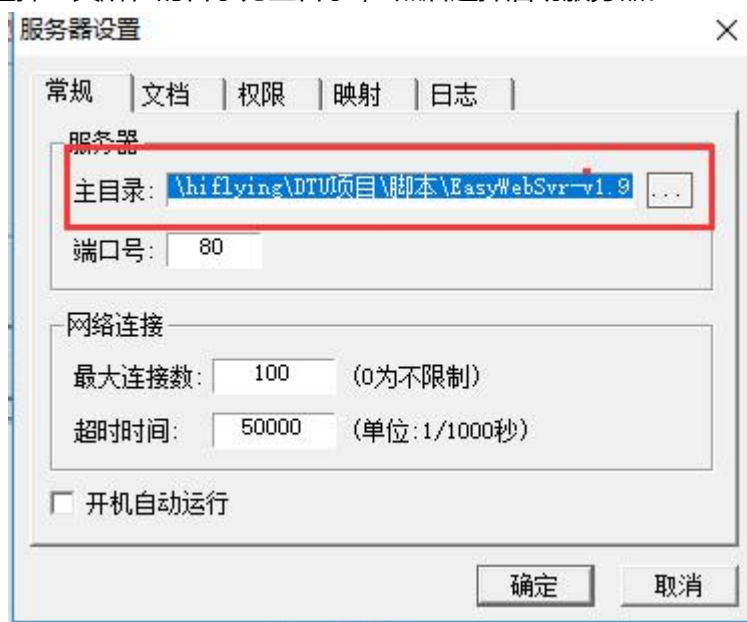
5. 脚本升级

注：固件版本号在 1.10h 版本以上的可以使用 IOTService 工具进行升级，实现方法参考附录 D。

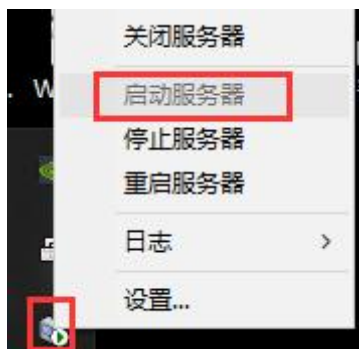
Step1：打开本地服务器软件“EasyWebSvr.exe”，右键点击设置选项。



Step2：选择工具所在的目录为主目录，然后选择启动服务器。



Step3：选择启动服务器。



Step4：将生成的固件放在该目录下。



Step5 : 运行 SecureCRT 串口软件 (其他串口软件也可) , 进入 CLI 命令模式 , 可看到 SYS 目录下有 Script 命令。

```

EPORT/SYS>
Version          Auth          Network      Telnet        web
NTP              MAC           JCMD         NAT           Ping
ProductID       CustomerID   UserID       CfgProtect   FactoryCfg
Script
EPORT/SYS>

```

Script 命令支持的指令如下 :

脚本查询 : Script

脚本删除 : Script Del

脚本下载 : Script http://192.168.199.100/Eport_Const_1.0.0_20170618.his

“http://192.168.199.100/Eport_Const_1.0.0_20170618.his” 为脚本下载路径(以上图为例)

Step6 : 运行脚本下载指令 , 下载成功后提示 Download Script OK! , 升级成功后请重启 (**脚本的功能需要重启后生效**) 。

```

EPORT/SYS>Script http://192.168.199.100/Eport_Const_1.0.0_20170618.his
Download Script OK!
EPORT/SYS>

```

注意 : 可查询确认当前脚本内容。

```

EPORT/SYS>Script
Script file:
SETHW(GPIO, OUT, 2, 1)

low1=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x30,0x0D,0x0A]
low2=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x30,0x00]
high1=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x31,0x0D,0x0A]
high2=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x31,0x00]

RECV SOCK netp

#设置GPIO1电平为高
IF (INPUT==high1)
    SETHW(GPIO, OUT, 2, 1)
    RETURN(FALSE)
END

IF (INPUT==high2)
    SETHW(GPIO, OUT, 2, 1)
    RETURN(FALSE)
END

#设置GPIO1电平为低
IF (INPUT==low1)
    SETHW(GPIO, OUT, 2, 0)
    RETURN(FALSE)
END

IF (INPUT==low2)
    SETHW(GPIO, OUT, 2, 0)
    RETURN(FALSE)
END

OUTPUT=INPUT
RETURN(TRUE)

END

```

6. 脚本控制 E10 产品 GPIO

6.1. 功能需求

netp Socket 通道接收到如下特定网络数据时，控制 GPIO2 输出高低电平，其他数据做默认透传功能。

接收到 “255:W:OSLED:1\0” 或者 “255:W:OSLED:1\r\n ” 数据时，GPIO2 输出高电平，对应如下十六进制数据：

```
32 35 35 3A 57 3A 4F 53 4C 45 44 3A 31 00
```

```
32 35 35 3A 57 3A 4F 53 4C 45 44 3A 31 0D 0A
```

接收到 “255:W:OSLED:0\0” 或者 “255:W:OSLED:0\r\n ” 数据时，GPIO2 输出低电平，对应如下十六进制数据：

```
32 35 35 3A 57 3A 4F 53 4C 45 44 3A 30 00
```

```
32 35 35 3A 57 3A 4F 53 4C 45 44 3A 30 0D 0A
```

6.2. 功能实现

测试脚本在 HIS_EXAMPLE 目录下 “E10_SCRIPT_Beijing-const_20170614.txt” 文件。脚本内容如下

```
SETHW(GPIO, OUT, 2, 1) //初始化 GPIO2 输出高电平
```

```
low1=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x30,0x0D,0x0A]
```

```
low2=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x30,0x00]
```

```
high1=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x31,0x0D,0x0A]
```

```
high2=[0x32,0x35,0x35,0x3A,0x57,0x3A,0x4F,0x53,0x4C,0x45,0x44,0x3A,0x31,0x00]
```

```
RECV SOCK netp //接收到 netp socket 的数据
```

```
#设置 GPIO1 电平为高
```

```
IF (INPUT==high1) //比较接收到的数据和 high1
```

```
SETHW(GPIO, OUT, 2, 1)
```

```
RETURN(FALSE)
```

```
END
```

```
IF (INPUT==high2)
```

```
SETHW(GPIO, OUT, 2, 1)
```

```
RETURN(FALSE)
```

```
END
```

```
#设置 GPIO1 电平为低
```

```
IF (INPUT==low1)
```

```
SETHW(GPIO, OUT, 2, 0)
```

```
RETURN(FALSE)
```

```
END
```

```

IF (INPUT==low2)
    SETHW(GPIO, OUT, 2, 0)
    RETURN(FALSE)
END

```

```

OUTPUT=INPUT
RETURN(TRUE)

```

END

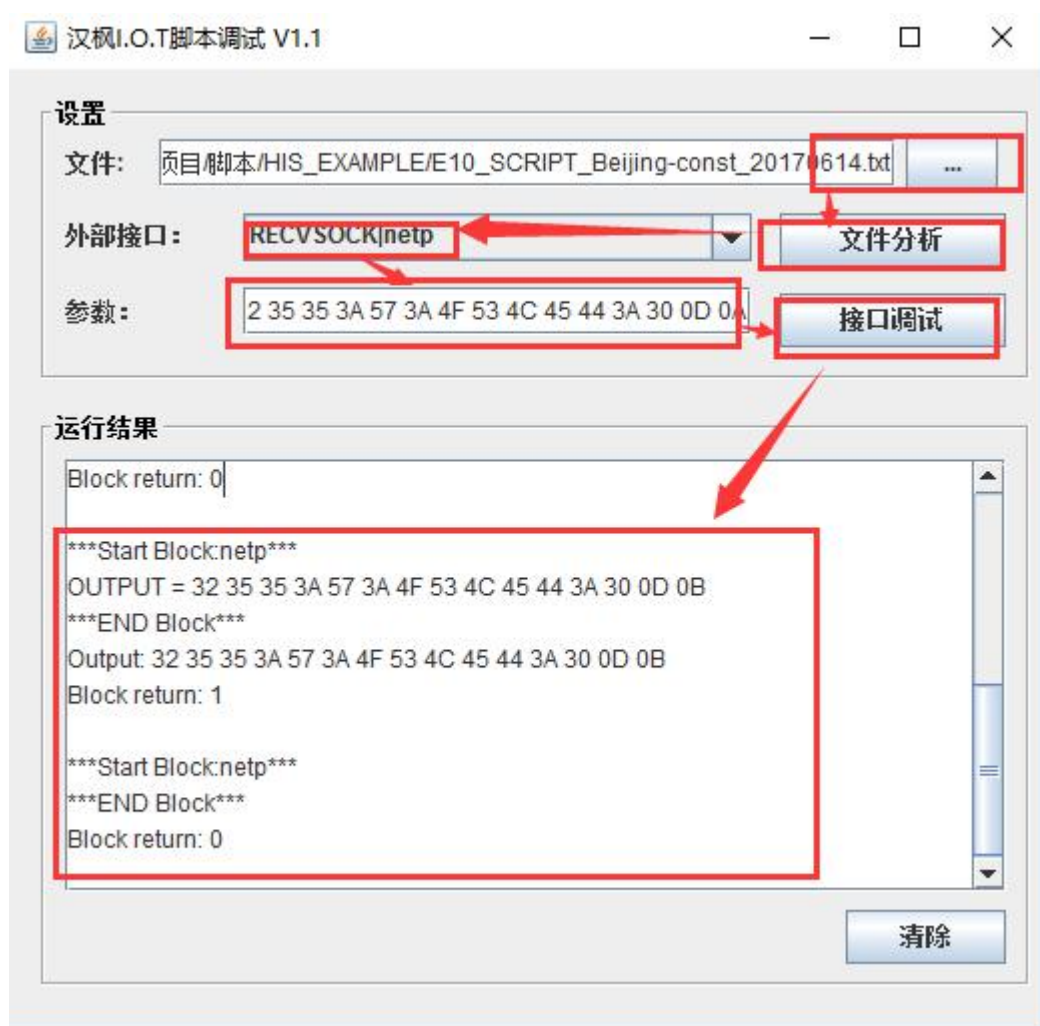
6.3. HisUI 工具测试

Step1 : 打开 HisUI 工具，并加载脚本文件

Step2 : 点击【文件分析】，选择脚本测试的接口。

Step3 : 输入测试的参数，并点击【接口调试】进行测试，参数内容如果以空格隔开，则认为是 16 进制格式（如下图），否则以 ASCII 码格式识别数据内容。

Step4 : 根据输入的不同参数，测试功能是否实现，如下样例中，Block 如果 return 1，则数据正常透传下发，如果 return 0 则数据不下发。

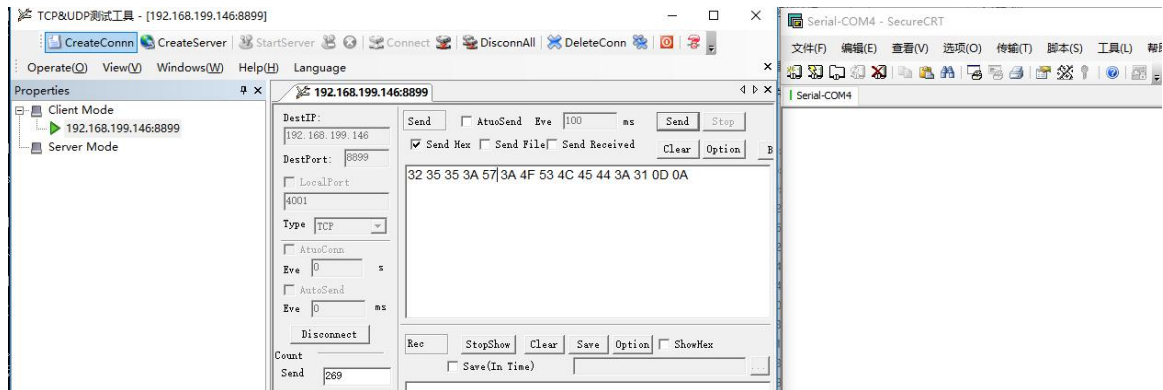


6.4. 脚本转换及设备加载脚本

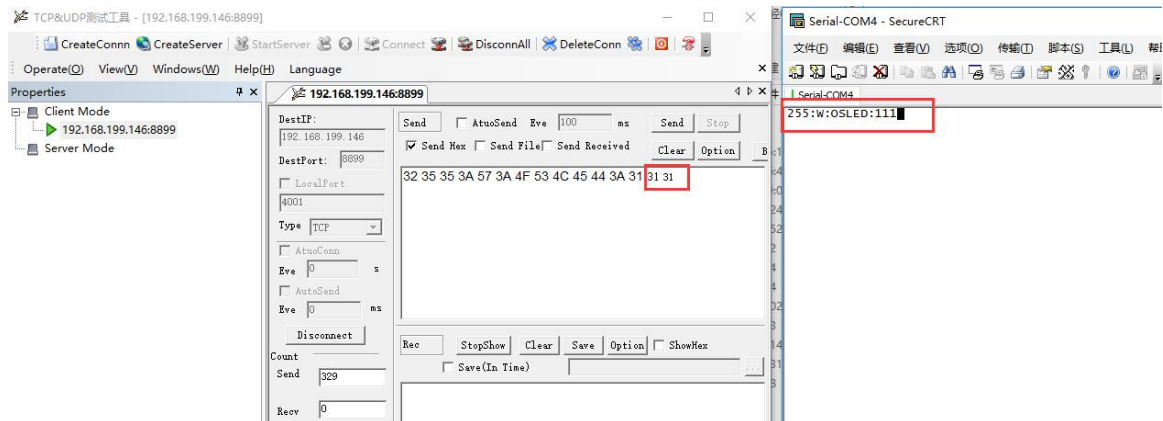
详见前述第 4、5 章节。

6.5. E10 产品测试

与 E10 产品 netpsocket 建立 TCP 连接, 发送测试数据, 如下测试数据可直接让 GPIO2 引脚输出高电平, 并且数据并不从串口输出。



稍稍改动数据内容, 可看到从串口输出对应数据。



附录 A. 心跳包功能案例

A.1. 功能需求

netp Socket 通道有网络连接时向串口发送“ netp connected ”，网络连接断开时向串口发送“ netp disconnected ”。

netp Socket 通道链接正常时，定时 30 秒发送心跳数据“heartbeat data”。

A.2. 功能实现

测试脚本在 HIS_EXAMPLE 目录下

“E10_E30_HF5111B_SCRIPT_example_heartbeat.txt” 文件。脚本内容如下

```
# “netp” 通道连接标志,0-未连接,1-连接成功
connFlag = 0
# “netp” 通道接收到数据的时间
recvTime = 0

# “netp” 通道心跳数据
space = [0x20]
beatData = "heartbeat" + space + "data"

connStr = "netp connected\r\n"

# “netp” 通道连接成功
CONN SOCK netp
connFlag=1
recvTime=SYSTIME
SEND(UART, uart0, connStr)
END

# “socka” 通道断开连接
DISCONN SOCK netp
connFlag=0
SEND(UART, uart0, "netp disconnected\r\n")
END

RECV SOCK netp
recvTime=SYSTIME
OUTPUT=INPUT
RETURN(TRUE)
END

TIMER HeartBeat 1000
t=SYSTIME
t=t-recvTime
```

```

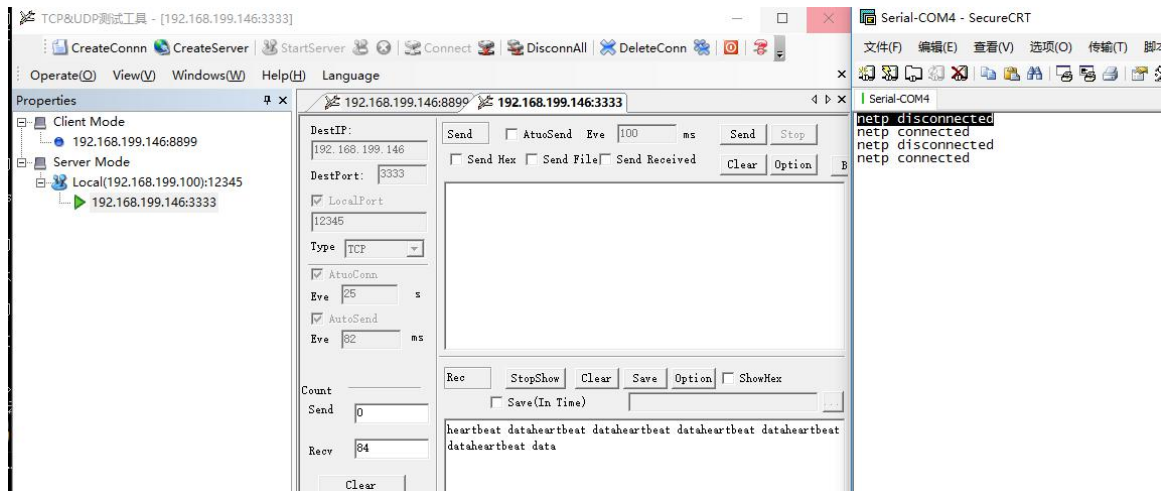
IF (connFlag == 1)
    IF (t > 30000)
        SEND(SOCK, netp, beatData)
        recvTime=SYSTIME
    END
END
END
    
```

A.3. 脚本转换及设备加载脚本

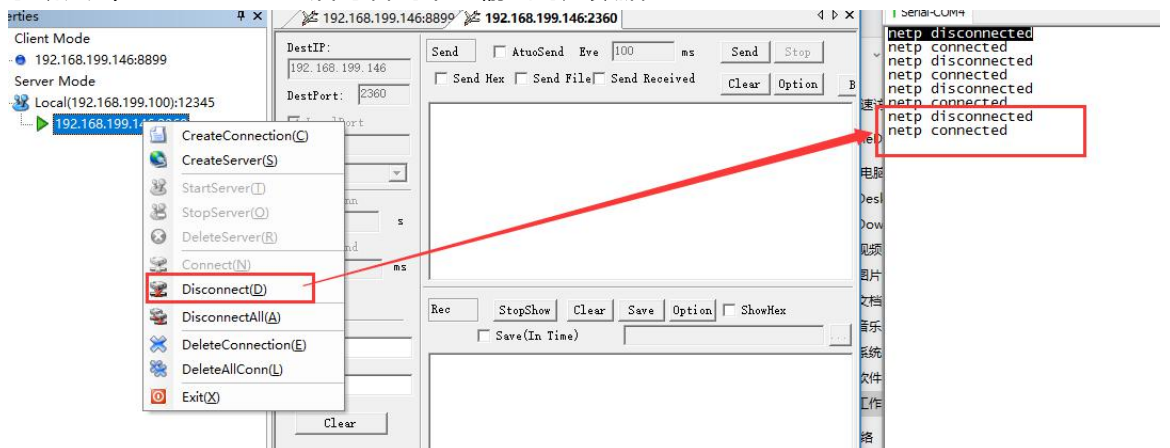
“E10_E30_HF5111B_SCRIPT_example_heartbeat.his” 是已经转换好的脚本，升级到产品中后并重启。

A.4. 产品测试

实际测试的效果



手动点击 Disconnect 之后可看到串口输出对应数据。



附录 B. 注册包功能案例

B.1. 功能需求

每次 netp 通道连接建立或者发送数据的时候，在数据头部增加额外的 id 信息或者 MAC 地址信息以供服务器区分设备。

B.2. 功能实现

测试脚本在 HIS_EXAMPLE 目录下

“E10_E30_HF5111B_SCRIPT_example_login.txt” 文件。脚本内容如下，默认脚本并没有启用注册包功能，可修改脚本 REGEN 和 REGTCP 入口参数，完成此案例的实际测试。

```
#---配置参数-----
# 注册包类型, 0-不开启注册包, 1-id 模式, 2-MAC 模式
REGEN = 0
# 注册包方式, 1-first 连接时发送, 2-every 每次发送数据时
REGTCP = 1
# 注册包 ID, 0-65535, 十六进制数组形式
REGID = [0x04, 0x57, 0xFB, 0xA8]
#-----

# 获取设备的 MAC 地址
mac = ""
macStr = MAC
mac = macStr.stringHex()

# “netp” 通道连接成功回调
CONN SOCK netp
IF (REGTCP == 1)
    IF (REGEN == 1)
        SEND(SOCK, netp, REGID)
    END
    IF (REGEN == 2)
        SEND(SOCK, netp, mac)
    END
END

# 串口接收到数据回调
RECV UART uart0
flag = 0
IF (REGTCP == 2)
    IF (REGEN == 1)
```

```
        OUTPUT = REGID + INPUT
        flag = 1
    END

    IF (REGEN == 2)
        OUTPUT = mac + INPUT
        flag = 1
    END
END

IF (flag == 0)
    OUTPUT = INPUT
END
RETURN(TRUE)
END
```

B.3. 其他说明

脚本的功能初始化参数可通过网页的形式进行修改,这样就可以方便的在网页上进行修改启用或者关闭对应功能,网页部分的修改案例近期发布再更新此功能。

附录 C. FLASH 参数使用案例

C.1. 功能需求

将心跳数据变量 beatData 定义成 flash 类型参数，通过 IOTService 工具（[关于 IOTService 工具使用参考 IOTServices 使用说明手册](#)）可以对这个参数进行修改，修改成功后，脚本会优先使用这些修改后的值。从而实现不同模块使用同一个脚本文件，但这些参数具有不同的初始值。netp Socket 通道链接正常时，定时 500 毫秒发送心跳数据。

C.2. 功能实现

测试脚本是将“E10_E30_HF5111B_SCRIPT_example_heartbeat.txt 文件里面的 beatData 变量定义为 FLASH(STRSTR) 类型变量及增加 FLASHMAGIC 变量定义(FLASHMAGIC 说明见 2.13 Flash 参数说明)。脚本内容如下，

```
#心跳包脚本示例
# "netp" 通道连接标志,0-未连接,1-连接成功
connFlag = 0
# "netp" 通道接收到数据的时间
recvTime = 0

# "netp" 通道心跳数据
FLASHMAGIC = 200
FLASH(STRSTR) beatData = "BEAT_DATA"
# "netp" 通道连接成功
CONN SOCK netp
connFlag=1
recvTime=SYSTIME
END

# "socka" 通道断开连接
DISCONN SOCK netp
connFlag=0
END

RECV SOCK netp
recvTime=SYSTIME
OUTPUT=INPUT
RETURN(TRUE)
END

TIMER HeartBeat 1000
t=SYSTIME
t=t-recvTime
IF (connFlag == 1)
    IF (t > 500)
```

SEND(SOCK, netp, beatData)

recvTime=SYSTIME

END

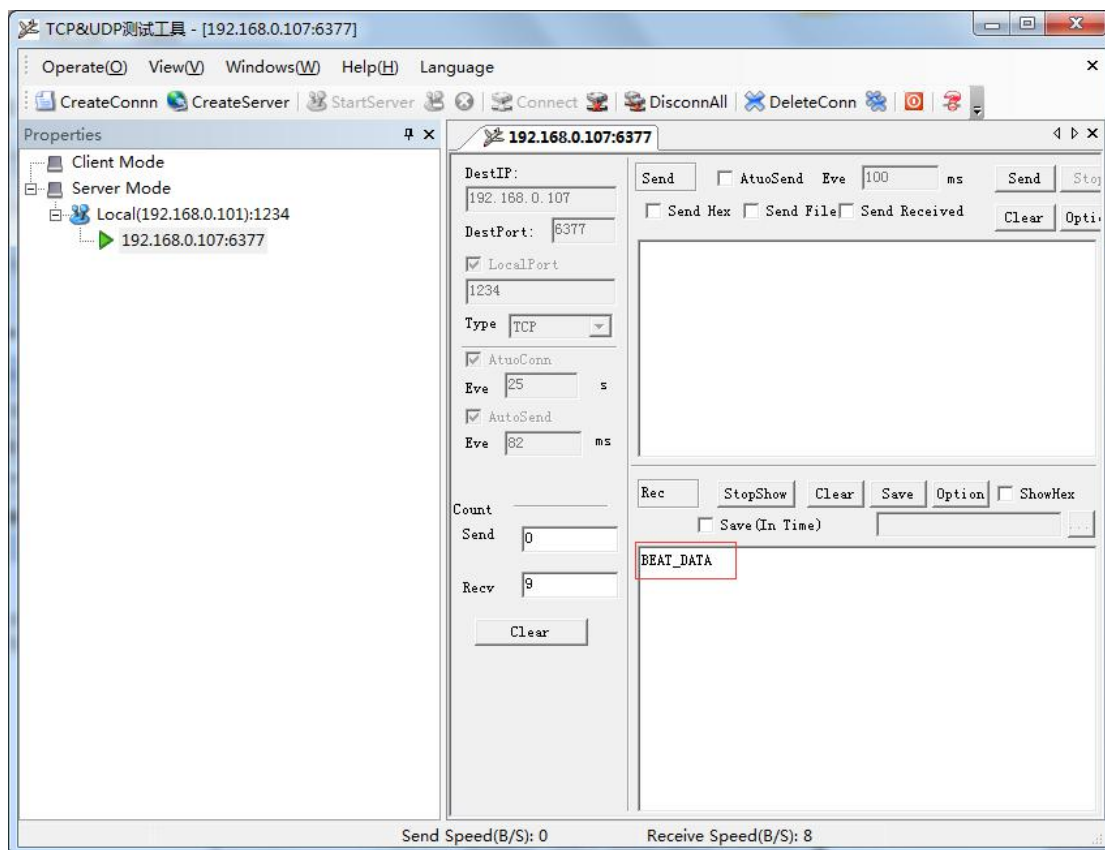
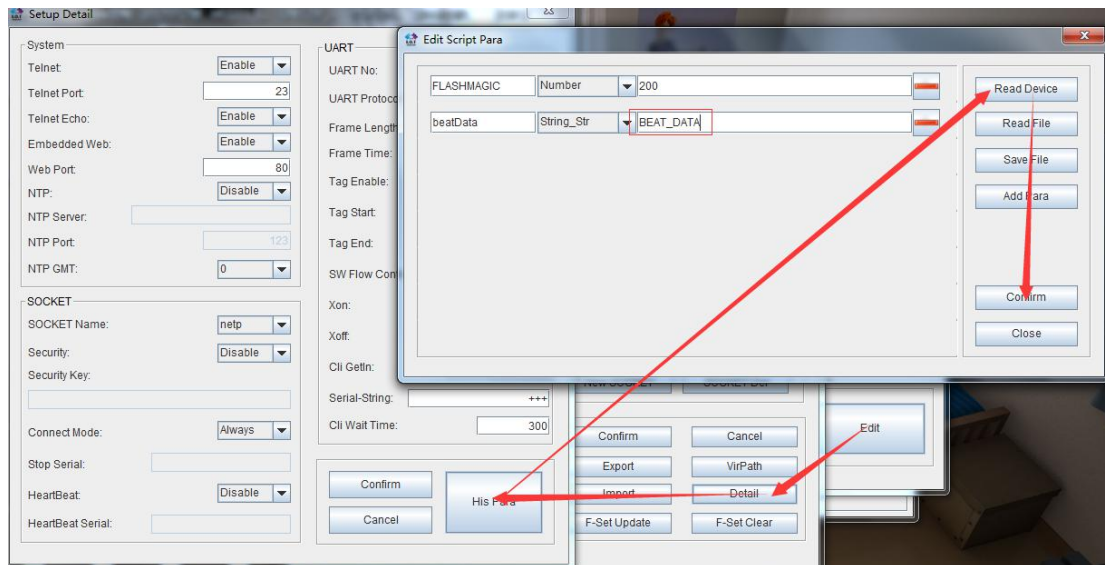
END

END

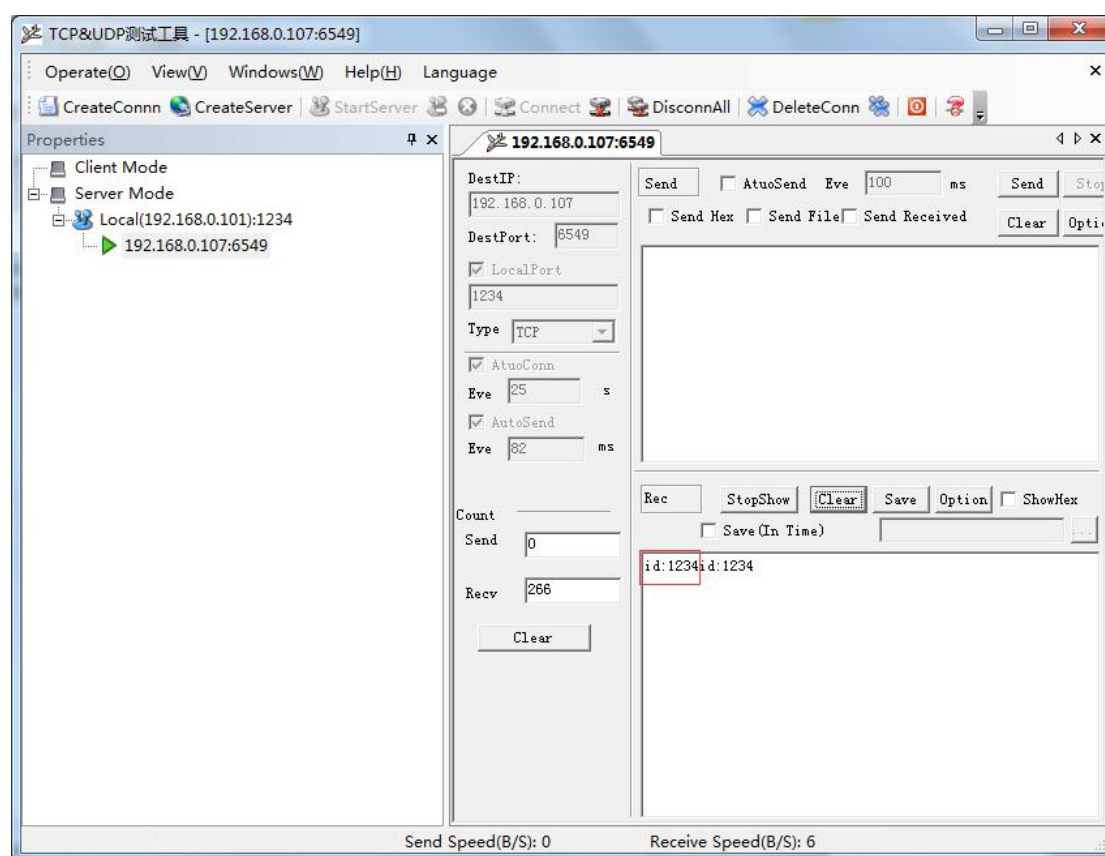
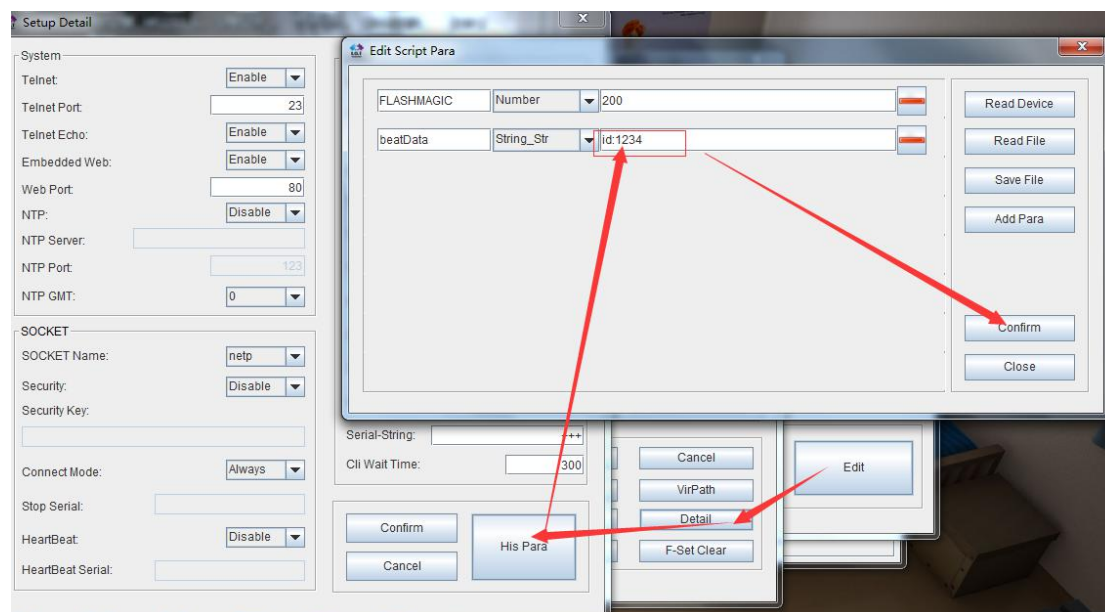
C.3. 产品测试

实际测试:

1.通过 IOTService 工具点击“读取设备参数”，获取到 FLASH 参数,参数为默认值时测试效果如图所示:



2.通过 IOTService 工具把心跳数据的参数的值修改为 id:1234,点击确认, 点击模块重启之后测试效果如下:



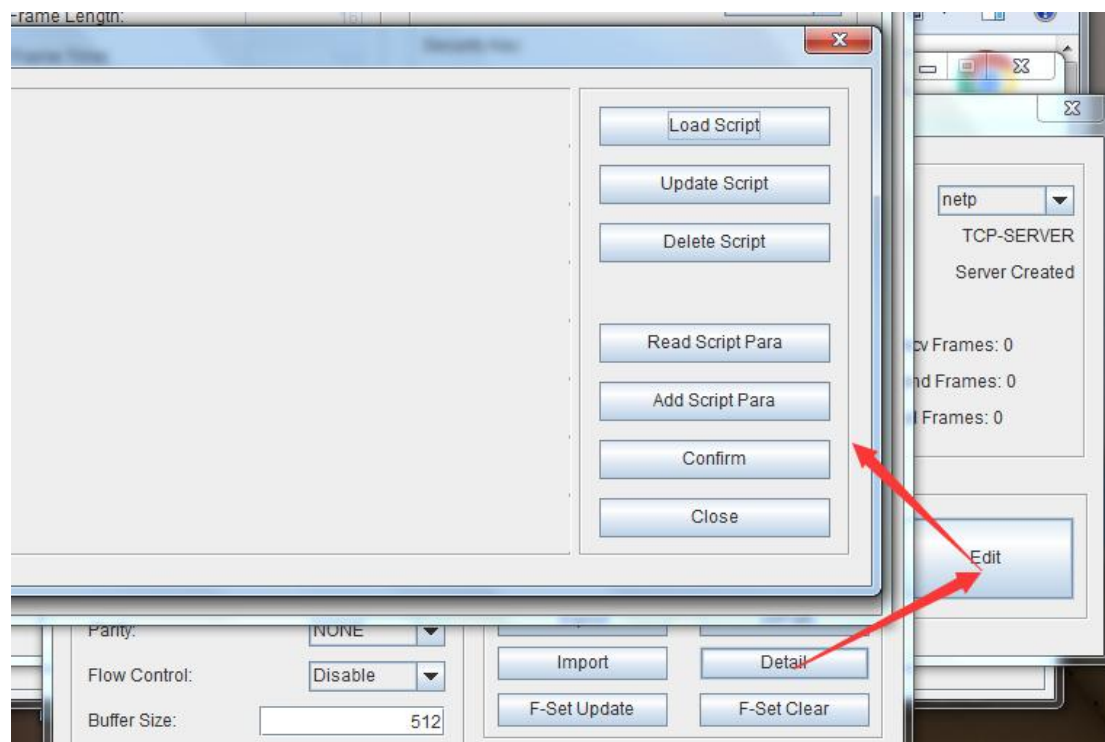
C.4. 其他说明

测试说明脚本里面可以将需要修改的参数定义为 flash 参数, 通过 IOTService 工具来修改这些参数的值来实现同一个脚本参数的初始化值不同, 从而达到我们可以任意修改参数值的目的。

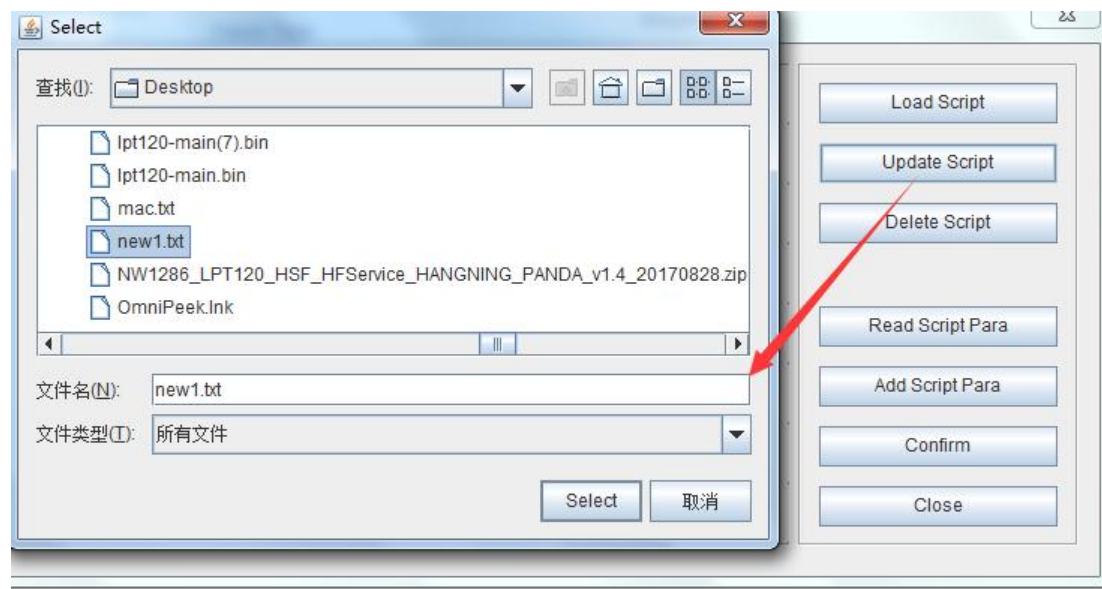
附录 D. IOTSERVICE 工具升级脚本方法

模块固件版本号在 1.10h 以上的版本都是可以通过 IOTService 导入脚本文件（注意：脚本文件格式为 txt）。

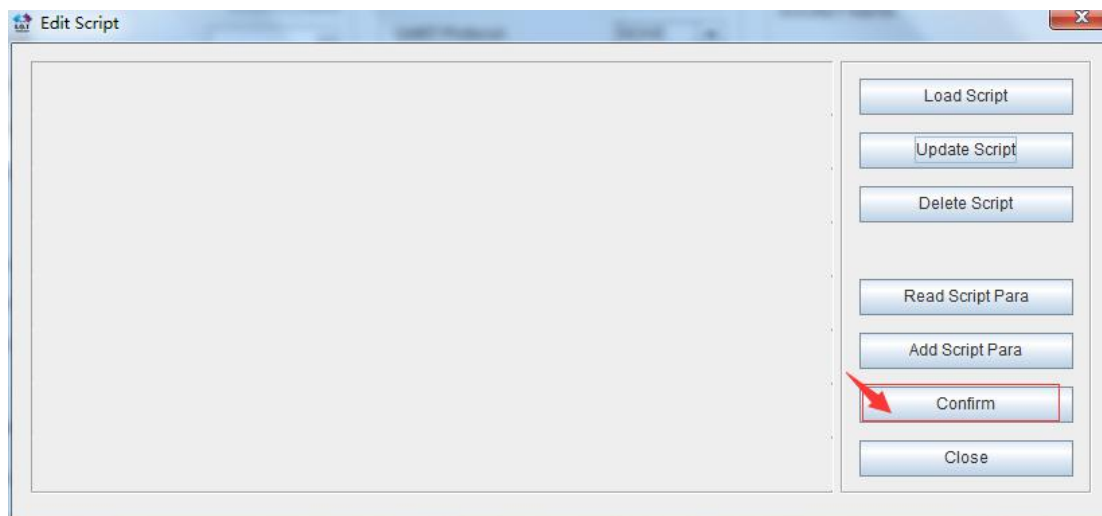
1、进入模块编辑界面，点击“高级设置”>“编辑脚本”。



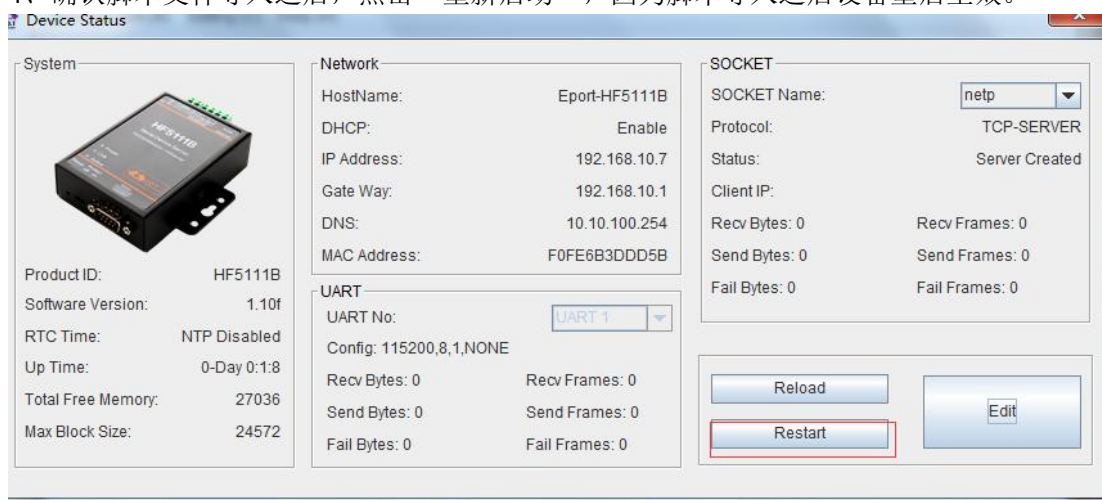
2、点击“更新脚本”进入选中导入的脚本文件 new1.txt。



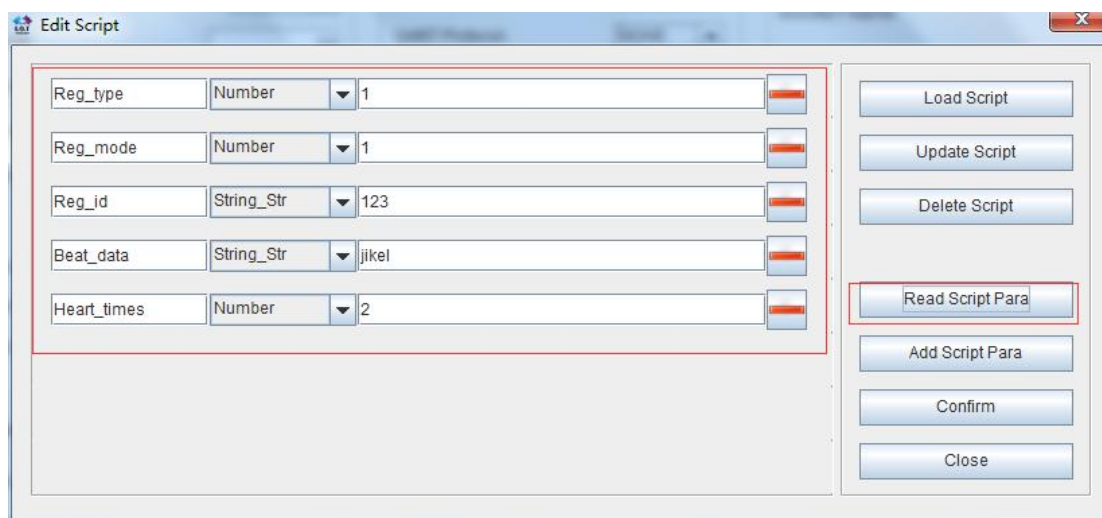
3、导入之后点击“确认”。



4、确认脚本文件导入之后，点击“重新启动”，因为脚本导入之后设备重启生效。



5、然后通过读取脚本参数来读取脚本里面的 FLASH 变量，然后可以通过工具直接对变量的值进行修改。



6、如果模块不需要脚本了，可以通过“删除脚本”来将脚本删除掉。

