

SECS/GEM Communication

IAS CHAT

Revision : V1.0.0

2020.11.27

Revision History.

[illegible]

目次

1.	概述	3
2.	软件运行环境	3
3.	安装与启动	3
4.	功能	4
4.1.	工程的创建	4
4.2.	设备通信文件管理（创建、打开）	5
4.3.	设备通信流程定义	5
4.4.	设备通信消息定义（含数据结构）	6
4.5.	设备通信事件定义（自动化系统用）	8
4.6.	设备通信调试	8
4.7.	设备异常通信	11
4.8.	设备通信日志	12

1. 概述

本软件只适用于 SECSII 标准的自动化设备，提供工程式程序代码解决方案。主要功能:工程定义、设备通信文件的定义、设备通信流程定义、设备通信消息定义（包含数据结构）、设备通信事件定义、设备调试（地址、端口、日志）

2. 软件运行环境

该软件基于 JAVA SWING 开发，硬件的要求不是很高，只要能跑 JAVA 程序即可。

软件方面：

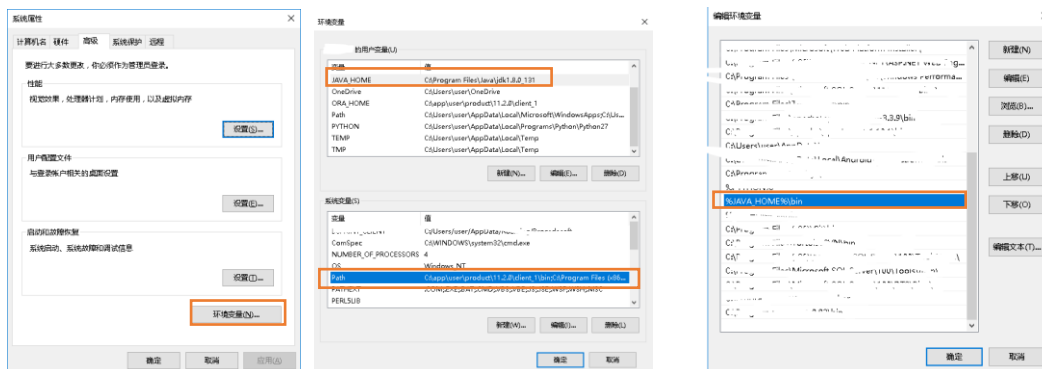
1. OS 可使用 WINDOWS、LINUX 等相关 OS 平台。
2. JAVA 版本 > 1.8

3. 安装与启动

安装 JAVA 环境，请使用 1.8 或以上版本，(已配置 JAVA 环境的可跳过该安装步骤，但需确定版本是否 >=1.8)
下载地址：<https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>

配置环境变量(Windows 版为例)

安装 JAVA 成功后，设置 JAVA_HOME 并将其增加到环境 PATH 中。



配置环境变量结束，打开命令提示符，测试下配置是否成功。

运行命令：`java -version`

出现以下结果则 OK，否则只能查找原因。

```
命令提示符
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation. 保留所有权利。

C:\Users\user>java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
```

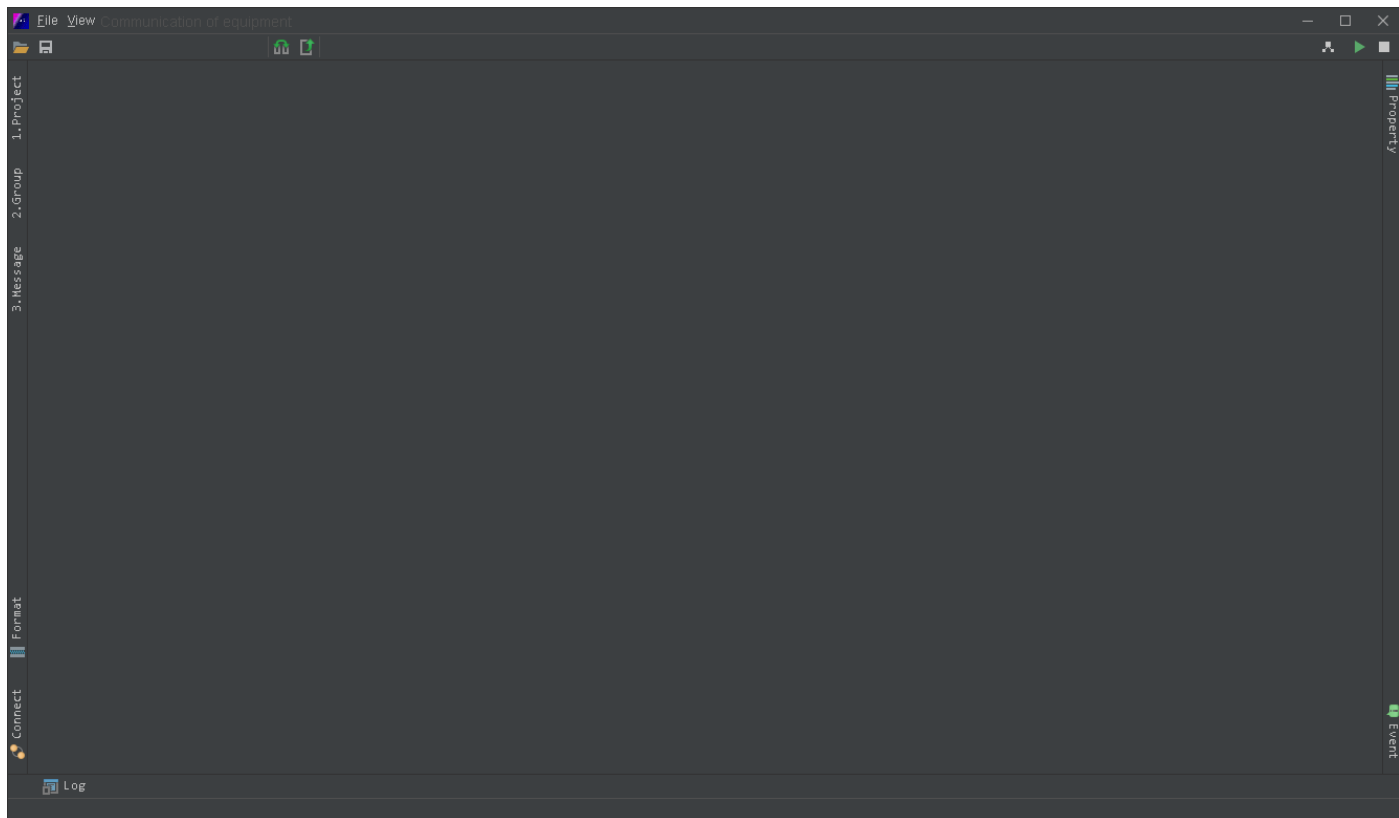
启动程序

使用脚本启动：

- Linux : `exec java -noverify -jar ias.application.chat-1.0.0.jar -Djava.io.tmpdir=/tmp/ems/application.chat`
- Windows : `java -noverify -jar -Dfile.encoding=UTF-8 ias.application.chat-1.0.0.jar`

初次启动的结果：

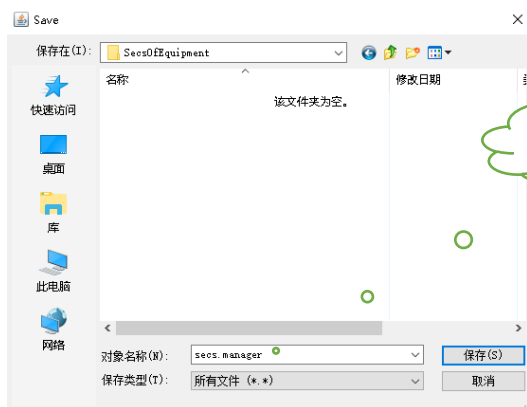
硬件配置低的情况 Splash 界面会稍微长些，请耐心等待。



4. 功能

4.1. 工程的创建

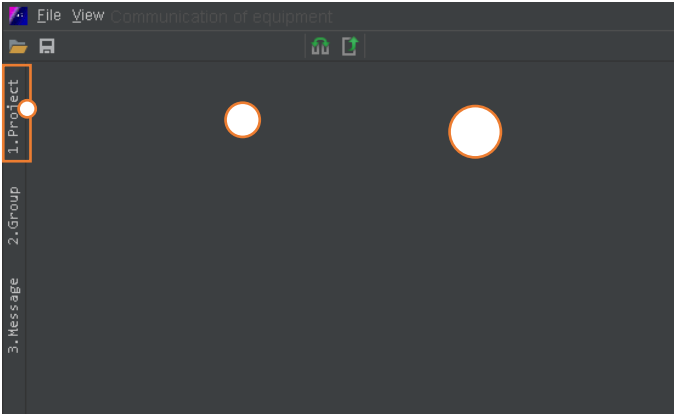
软件打开后，点击保存按钮，选择工程文件保存的路径，并命名为 **XXX 工程文件**（如:secs.manager），系统自动在该路径下生成 **secs.manager.ias** 文件以及相关配置文件，请勿删除下图的文件及文件夹。



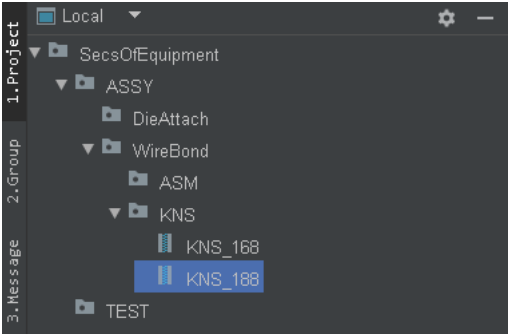
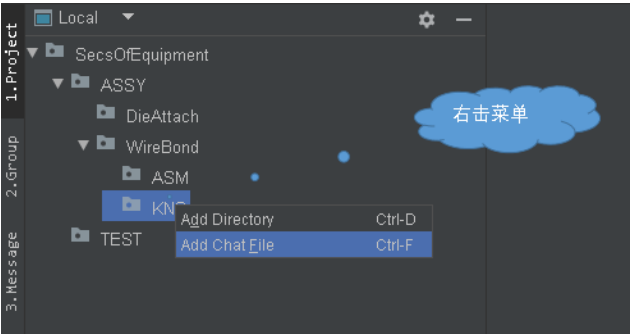
组织		新建	打开	选择
WORK > JAVA > SecsOfEquipment				
名称	修改日期	类型	大小	
ias	2020/11/26 10:41	文件夹		
secs.manager.ias	2020/11/26 10:41	IAS 文件	1 KB	

4.2. 设备通信文件管理（创建、打开）

打开 Project 窗口



对于每个机型可以根据厂商或者产线进行划分管理，比如以封装工厂为例，设备在产线可以划分组装和测试两大部门的设备，那么可以创建 **ASSY** 和 **TEST** 文件夹。对于设备来说，比如装片机 (**DieAttach**)、键合机 (**WireBond**)，那么在 **ASSY** 文件夹下建立 **DieAttach**、**WireBond**。键合机设备的厂商常用的有 **KNS**、**ASM**，在 **WireBond** 下建立 **KNS**、**ASM** 两个厂商文件夹来区分键合设备通信文件。**KNS** 键合机的版本比较多，最好每个版本建立一个通信文件（不排除版本不同，通信功能一致的设备，可以多个版本使用一个通信文件）。比如 **KNS168** 和 **KNS188** 两个版本，分别建立两个通信文件。



4.3. 设备通信流程定义

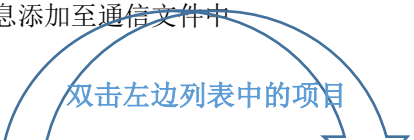
设备通信文件已经建立，为了让设备同软件进行交互式会话，那么需要在设备通信文件中建立交互式通信流程，通信流程可以分组划分，以便两者之间的通信不会扰乱，跳来跳去。通信流程分组初始模板的定义存放软件根目录下：**application-group.yml** 文件中。可以修改文件内容，重启即可升级分组模板。



分组信息如图：

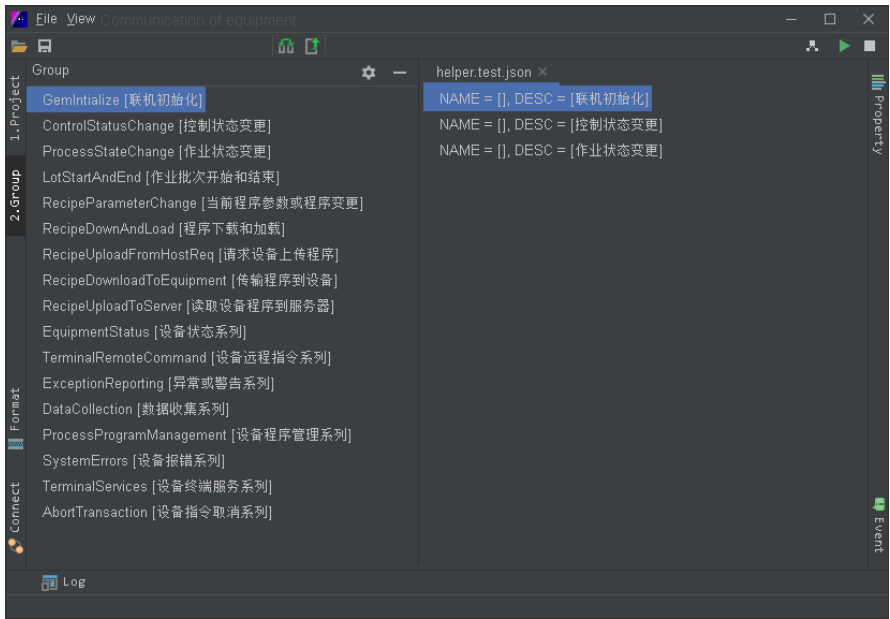
添加通信流程分组步骤：

1. 打开设备通信文件
2. 打开分组窗口，并双击所需要的通信流程分组
3. 分组信息添加至通信文件中



NAME	FILE LAST MOD	EXTENSION	SIZE
application-group.yml	2020/11/25 16:45	YML 文件	2 KB
application-message.yml	2020/11/25 16:45	YML 文件	38 KB
IAS.Application.Chat-1.0.0.jar	2020/11/25 16:47	JAR 文件	24,466 KB

如：添加初始化流程到通信文件中，以测试文件 `helper.test` 通信文件为例



4.4. 设备通信消息定义（含数据结构）

消息即为设备与软件之间的通信数据包，数据包本是通过二进制的方式进行传输的，但基于界面的方式来定义数据包和修改数据包，软件中使用了消息树，消息树的根为消息的定义，树叶为消息的内容。

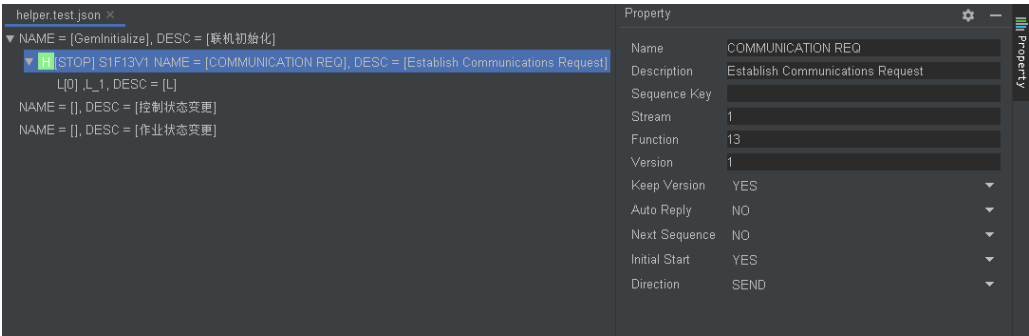
打开 **Message** 窗口，可以看到各种消息的定义。设备通信是基于流程的，所以我们需要把消息定义添加到流程中。比如设备同软件进行交互时，首先要进行握手通信。即 **HOST** 发送 **S1F13** 到设备，设备回复 **S1F14** 给 **HOST**。相反，部分设备也会发送 **S1F13** 给 **HOST**，此时 **HOST** 也必须回复 **S1F14** 给设备，否则设备会认为通信异常，断开连接重连。

S1F13R	H,E	L:2 <MODLN> <SOFTREV>	"Establish Communications Request"
S1F14	E	L:2 <COMMACK> {L:2 <MODLN> <SOFTREV> }	"Establish Communications Request Acknowledge"
		<COMMACK> := {B 0}	

对于这一过程，软件中消息的定义如下：

- 1. 打开 **Message** 窗口
- 2. 将 **S1F13** 消息拖拽至设备通信文件的初始化流程【**GemInitialize**】中
- 3. 将 **S1F13** 按照 **HOST->EQ** 的 **SECSII** 标准进行数据结构填充

HOST 发送 **S1F13** 的内容可为空列表，如图



KeepVersion -> **YES**, 发送消息后只接受 **S1F14V1** 的消息，否则接受任何 **S1F14** 的消息。

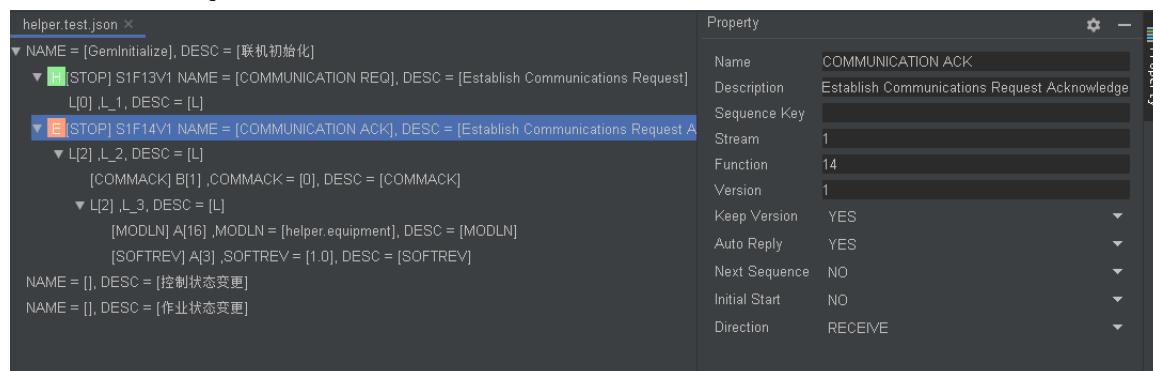
AutoReply -> **NO**, 一般为 **NO**。回复消息使用 **YES**, 比如 **S1F14**, **S5F2**, **S6F12**...

需要**注意**的是，如果设备发过来的主消息比如 **S1F13**，如果需要 **HOST** 自动回复，那么需要将该主消息的标记位置为 **YES**，否则 **HOST** 不会自动回复 **S1F14**，回复权交由 **EMS** 自动化系统处理。

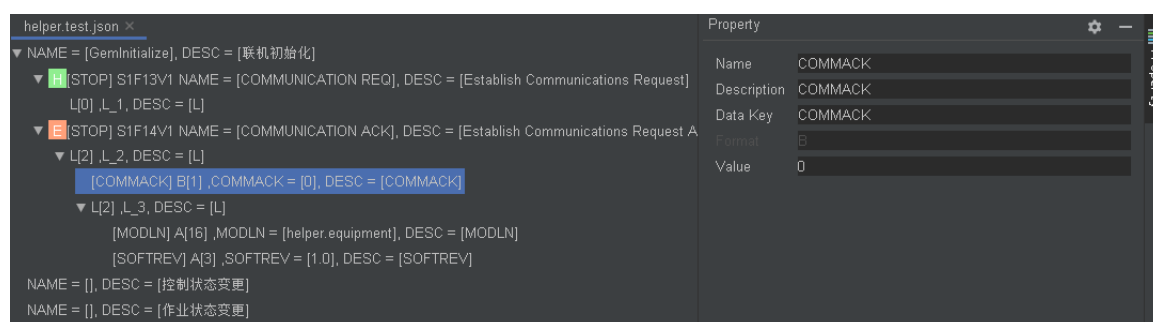
NextSequence -> **YES**, 需配合自动化程序，当前触发成功后，自动执行下一条消息。

InitialStart -> YES, 连接成功后, 自动执行的消息。这里用于 S1F13, 意味着 HOST 与设备连接成功后直接发送 S1F13, 与设备进行握手通信, 重要! 否则无法握手, 通信将会中断。

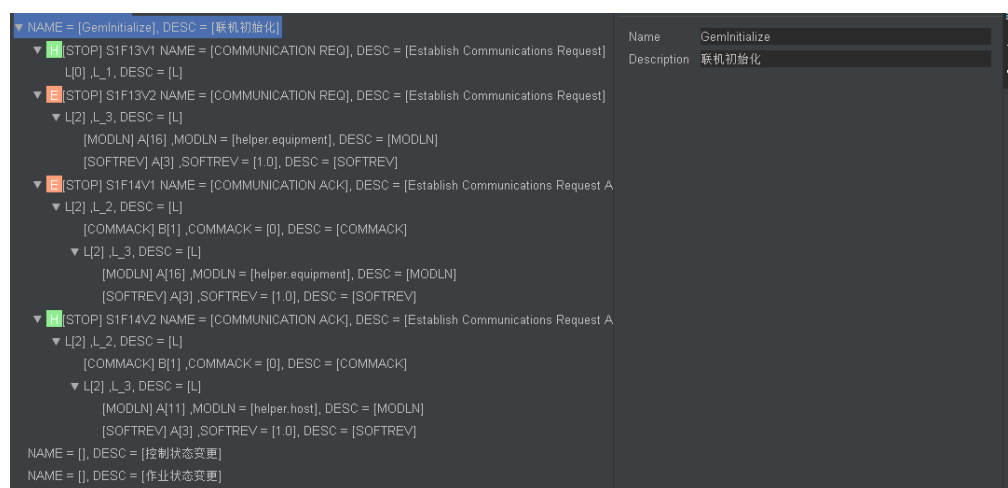
- 将 S1F14 消息拖拽至设备通信文件的初始化流程【GemInitialize】中
- 将 S1F14 按照 EQ->HOST 的 SECSII 标准进行数据结构填充



其中 COMMACK 的值很重要, 如果是 0, 则意味着与设备握手成功, 否则握手通信将被取消, 连接重启。做模拟通信设备时, 请将 COMMACK 的值设为 0, 后面模拟通信将用得到。



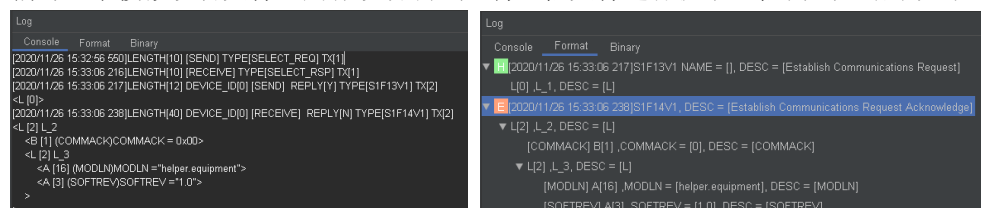
- 通信是双向的, 完整的握手通信结构如下。



helper.test.json

- 通信测试

编写一个模拟设备文件, 用作设备调试, 将通信文件进行握手通信测试, 结果如下



具体如何通信, 参考后面的通信调试说明。

4.5. 设备通信事件定义（自动化系统用）

设备在通信过程中会不断将自己的状态数据（通信、作业、程序等信号）发送给 HOST，HOST 收到这些数据后会对其进行解析、判断、控制等操作。收到状态数据后做什么样的处理我们可以通过触发事件来处理。

比如握手通信成功，意味着当前设备上线，我们会将上线信息记录至 EMS 服务系统。

事件定义如下：

Name	Description	Type	ResultTrigger	ListenerName	ExceptionHan...
GemStart	Commucation...	2	Y	GEM_START	1

EMS 端 HOST 通信代码：

```
this.addSequenceTrigger(1, 14, 1, item -> {
    if (item.getMessageStructureValue("COMMACK").getValue().equals("0")) {
        return "Y";
    } else {
        return "N";
    }
});
```

当 COMMACK 值为 0 时，触发结果为 Y，则执行 GEM_START 事件。HOST 执行 doGemStart 接口实现体将设备信息传送至 EMS 服务器。

```
@Override
public boolean doGemStart(SecsSequenceMessage secsSequenceMessage) throws Exception {
    return super.doGemStart(secsSequenceMessage);
}
```

4.6. 设备通信调试

设备通信流程文件编写结束，如何验证通信文件的有效性，需要做到如下两个步骤：

什么情况下需要设备通信流程文件进行调试：

1. 设备商的 SECS 功能是否健全，是否满足自动化开发的要求，那么就需要对设备进行一些指令调试。调试的结果直接反映在日志中，如果出现问题需要把日志信息 EMAIL 给设备商，让设备商解决该指令问题。调试结束后，需填写验机报告，并且日志文件存档。
2. 自动化开发时，需要调试自动化程序是否 OK，那么设备端的数据必不可少。但设备大部分时间是用来生产的，不可能一直空着给自动化程序员当作调试的设备。那么可以根据之前的验机调试结果，建立模拟设备通信文件（比如设备通信文件：NS-001，模拟设备通信文件为 NS-001-EQ），与实际设备通信进行通信模拟，在没有真机调试的情况下，模拟设备发送数据信号给 HOST，来方便的开发 EMS 自动化程序。

HOST 如何发送数据包给设备，设备如何发送数据包给 HOST，需要满足以下几个要素：

1. EQ 的 IP 地址和端口，并且 TCP 通信方式是 PASSIVE（一般都是 PASSIVE,无需关心对方地址来源，除非一些安全设备需要指定对方 IP 地址）
2. HOST 设定通信方式为 ACTIVE，并且指定对方设备的地址和端口。
3. HOST 端填写正确的设备代号，以便收集该设备的日志，区别于其他设备的日志。
4. T3-T8 的设定遵循 SECSII 协议的设定原则，并且 HOST 设定的范围与实际硬件性能（网络，设备反应速度）息息相关。一般设定（T3:45s, T5:10s, T6:5s, T7:10s, T8:5s），个别情况比如网络不稳定或者设备通信反应过慢，请把 T6 时间调高些，比如 10s，否则规定时间内收不到控制信号数据会自动断网重新握手连接。另外 LINK-TEST 时间根据自己的情况而定，一般设为 30s，意味着每 30s 发送一次心跳包，如果收不到回复，HOST 会自动断线重新握手通信。

本文中不使用真机设备进行调试说明（防止真机设备安全泄密），故建模拟设备，有条件的可以根据本文进行模拟调试，后面会提供比较全面的指令集模板，只要对一些 SVID、CEID 的值进行一些修改就可以拿去使用（每次从 0 搭建通信文件真的很累，而且容易出现遗漏，初次了解系统的最好从 0 开始，方便熟悉系统软件）。

调试步骤如下：

1. 打开 Connect 窗口，对 HOST 文件和模拟设备文件进行连接设置（双击通信文件，确保 HOST 和模拟器的文件都处于打开状态，选中其中的一个，进行连接设置编辑）。具体设置如下：

设备号：H-001

模拟设备号：



2. 点击窗口右上角的绿色运行按钮  或者 **【Shift+F10】**，启动 HOST 通信和模拟 EQ 通信（最好先启动 EQ 通信，因为 EQ 处于通信状态时，HOST 直接连接上，否则 HOST 发现连不上会连接等待或者休眠等待），启动成功连接时，工具栏按钮如图：。绿色三角变成灰色，将无法使用，剩下的红色方框和传输图标按钮可以使用。红色方框按下则关闭通信或者 **【Shift+F2】**。传输图标按钮按下或者 **【Ctrl+T】** 则发送选中的消息数据（必须选中消息的头部）。

启动后的状态界面如下：

4.7. 设备异常通信

设备通信过程不是所有的通信指令都能按照自己要求的格式反应到设备通信文件中，比如中设备发送的指令未在 HOST 端通信文件中定义或者与 HOST 端的通信文件定义的结构有区别，那么通信过程将会出现异常的指令回复。

例如 1：设备发送 S6F11，HOST 通信文件未定义 S6F11，结果如下：

ConsoleFormatBinary

[2020/11/27 13:51:52 135]LENGTH[14] DEVICE_ID[0] [RECEIVE] REPLY[N] TYPE[S6F11V0] TX[38]
<L [1]
<L [0]>
>
[2020/11/27 13:51:52 137]LENGTH[10] DEVICE_ID[0] [SEND] REPLY[N] TYPE[S6F0V0] TX[38]

ConsoleFormatBinary

[2020/11/27 13:51:52 135]S6F11V0, DESC = [Unsupported msg]
L[1],
L[0],
[2020/11/27 13:51:52 137]S6F0V0, DESC = [ABORT]

如果在 HOST 通信文件中定义 S6F11V1 的接收消息，那么结果就不一样了。HOST 端收到 S6F11 后，如果有回复消息定义，那么将自动回复 S6F12。定义格式如下：

helper.test.eq.json × helper.test.json ×

▼ NAME = [GemInitialize], DESC = [联机初始化]
▶ [H][STOP] S1F13V1 NAME = [COMMUNICATION REQ], DESC = [Establish Communica
▶ [E][STOP] S1F13V2 NAME = [COMMUNICATION REQ], DESC = [Establish Communica
▶ [E][STOP] S1F14V1 NAME = [COMMUNICATION ACK], DESC = [Establish Communica
▶ [H][STOP] S1F14V2 NAME = [COMMUNICATION ACK], DESC = [Establish Communica
▶ NAME = [], DESC = [控制状态变更]
▼ NAME = [], DESC = [作业状态变更]
▶ [E][STOP] S6F11V1 NAME = [Seq_2], DESC = [Event Report Send]
 ▼ L[1] ,L_4, DESC = [L]
 L[0] ,L_5, DESC = [L]
 ▶ [H][STOP] S6F12V99 NAME = [Seq_3], DESC = [Event Report Acknowledge]
 B[0] ,ACKC6 = [], DESC = [Data Collection Ack]

Property

NameSeq_2
DescriptionEvent Report Send
Sequence Key
Stream6
Function11
Version1
Keep VersionNO
Auto ReplyYES
Next SequenceNO
Initial StartNO
DirectionRECEIVE

运行结果：

```
[2020/11/27 14:11:54 013]LENGTH[14] DEVICE_ID[0] [RECEIVE] REPLY[Y] TYPE[S6F11V1] TX[1]  
<L [1] L_4  
<L [0]>  
>  
[2020/11/27 14:11:54 016]LENGTH[12] DEVICE_ID[0] [SEND] REPLY[N] TYPE[S6F12V99] TX[1]  
<B [0]>
```



测试用通信文件： helper.test.json

4.8. 设备通信日志

日志存放于设备软件的根目录下的 **logs** 文件夹。日志文件分两种：

1. 软件日志
2. 设备通信日志

2020-11-27
H-001
H-EQ
ias.application

日期文件夹同 **ias.application** 为软件日志，其他为各设备的通信日志。

设备通信日志内容样本如下：

```
2020-11-27 12:48:00.986 [BINARY] [Thread-16] INFO com.ias.core.log.ThreadLogWriter$1 - 00 00 00 0A FF FF 00 00 00 01 00 00 00 01
2020-11-27 12:48:00.987 [FORMAT] [Thread-17] INFO com.ias.core.log.ThreadLogWriter$1 - LENGTH[10] [SEND] TYPE[SELECT_REQ] TX[1]
2020-11-27 12:48:01.082 [LOG] [Thread-8] DEBUG com.ias.core.secs.SecsConnector - Message sender's thread id is 50 for transaction:1
2020-11-27 12:48:01.082 [LOG] [Thread-8] DEBUG com.ias.core.secs.SecsConnector - Message sender executed over for transaction:1
2020-11-27 12:48:01.083 [LOG] [Thread-8] DEBUG com.ias.core.secs.SecsConnector - Transaction 1 started for mq SELECT_REQ {00 00 00 0A FF FF 00 00 00 01 00 00 00 01}
2020-11-27 12:48:01.083 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Handle message before communication state.
2020-11-27 12:48:01.084 [BINARY] [Thread-33] INFO com.ias.core.log.ThreadLogWriter$1 - 00 00 00 0A FF FF 00 00 00 02 00 00 00 01
2020-11-27 12:48:01.085 [FORMAT] [Thread-34] INFO com.ias.core.log.ThreadLogWriter$1 - LENGTH[10] [RECEIVE] TYPE[SELECT_RSP] TX[1]
2020-11-27 12:48:01.086 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Received mq: SELECT_RSP {00 00 00 0A FF FF 00 00 00 02 00 00 00 01}
2020-11-27 12:48:01.087 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Received SELECT_RSP mq with SelectStatus: Communication Established
2020-11-27 12:48:01.087 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - BEFORE CONNECTION STATE: NOT_SELECTED
2020-11-27 12:48:01.087 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Connection State set to SELECTED
2020-11-27 12:48:01.087 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - BEFORE CONTROL STATE: EQUIPMENT_OFFLINE
2020-11-27 12:48:01.087 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Control State set to ATTEMPT_ONLINE
2020-11-27 12:48:01.088 [LOG] [Thread-8] DEBUG com.ias.core.secs.SecsConnector - Send message for transaction:0
2020-11-27 12:48:01.094 [BINARY] [Thread-42] INFO com.ias.core.log.ThreadLogWriter$1 - 00 00 00 0C 00 00 81 0D 00 00 00 00 02 01 00
2020-11-27 12:48:01.104 [FORMAT] [Thread-43] INFO com.ias.core.log.ThreadLogWriter$1 - LENGTH[12] DEVICE_ID[0] [SEND] REPLY[Y] TYPE[S1F13V1] TX[2]
<L [0]>
2020-11-27 12:48:01.187 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Handle message before communication state.
2020-11-27 12:48:01.198 [BINARY] [Thread-62] INFO com.ias.core.log.ThreadLogWriter$1 - 00 00 00 28 00 00 01 0E 00 00 00 00 00 02 01 02 21 01 00 01 02 41 10 68 65 6C
2020-11-27 12:48:01.201 [FORMAT] [Thread-63] INFO com.ias.core.log.ThreadLogWriter$1 - LENGTH[40] DEVICE_ID[0] [RECEIVE] REPLY[N] TYPE[S1F14V1] TX[2]
<L [2] L_2
<B [1] (COMMACK)COMMACK = 0x00>
<L [2] L_3
<A [16] (MODLN)MODLN = 'helper.equipment'>
<A [3] (SOFTREV)SOFTREV = '1.0'>
>
2020-11-27 12:48:01.202 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Received mq: S1F14 - {00 00 00 28 00 00 01 0E 00 00 00 00 00 02 01 02 21 01 00 01 02
2020-11-27 12:48:01.207 [LOG] [Thread-8] INFO com.ias.core.secs.message.a.c - Received S1F14 with COMMACK: Accepted
2020-11-27 12:48:01.211 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - BEFORE COMMUNICATION STATE: NOT_COMMUNICATING
2020-11-27 12:48:01.211 [LOG] [Thread-8] INFO com.ias.core.secs.SecsConnector - Communication State set to COMMUNICATING
2020-11-27 12:48:01.212 [LOG] [Thread-8] DEBUG c.i.a.chat.ui.main.navigate.b - Communication state changed. Bar button will be refreshed H-001-COMMUNICATING
```