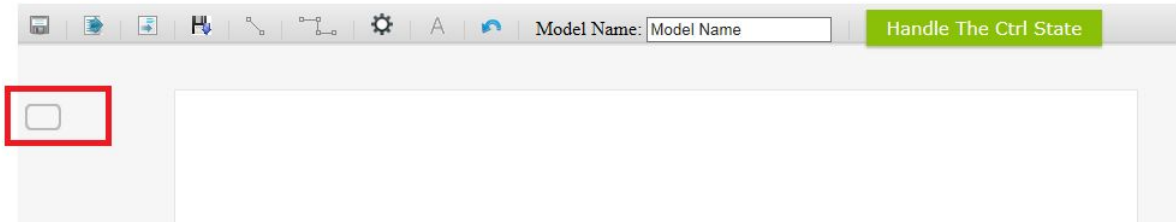
**User Manual for HA_Designer and JsonToIEC Compiler**

## 1. HA_Designer (source code is on: https://github.com/cjjjcj4/HA_Designer.git )

HA_Designer is a javascript Tool for designing Hybrid Systems. It's based on HTML5 Canvas Element and JavaScript. It supports designing of single HA module and multiple models combined HA system.

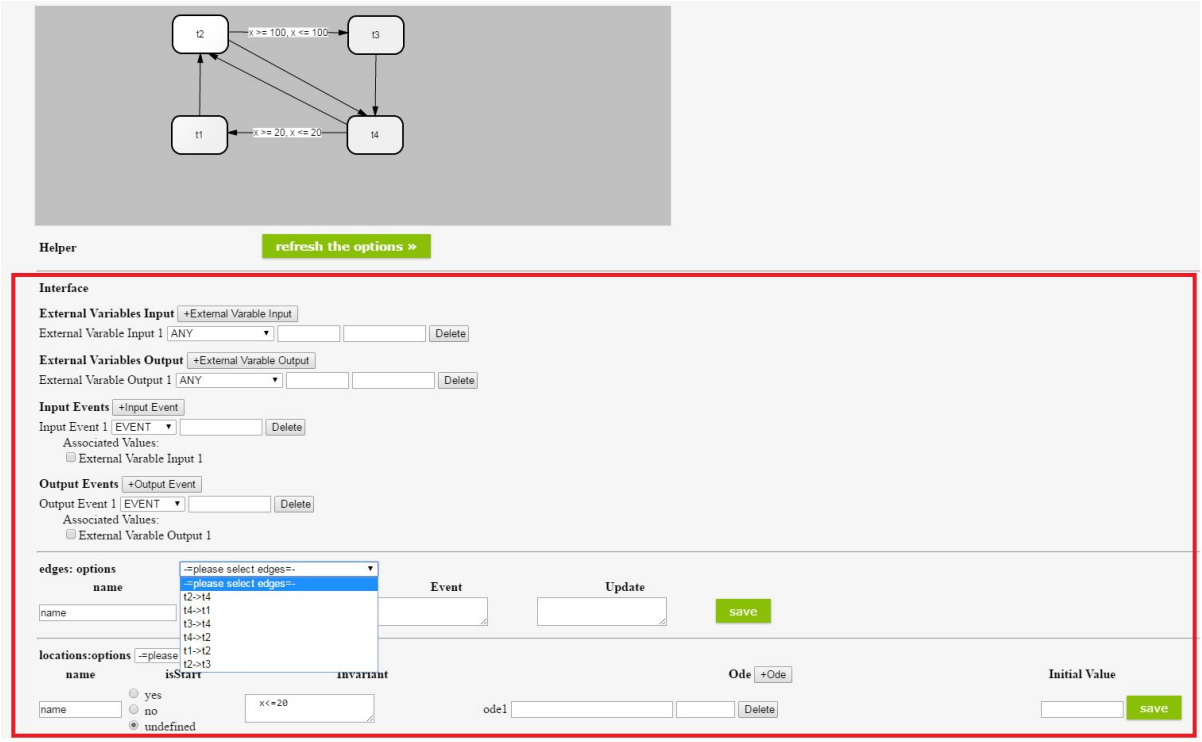
### 1.1 Design a single HA Model

Step 1: Double click /HA_Designer/editor/editor.html to open the tool.
Step 2: Draw locations by dragging the rectangle on the left.




Step 3: Draw edges using the straight connector on the top toolbar.


Step 4: Once you finish drawing, edit hidden data in the editing area.

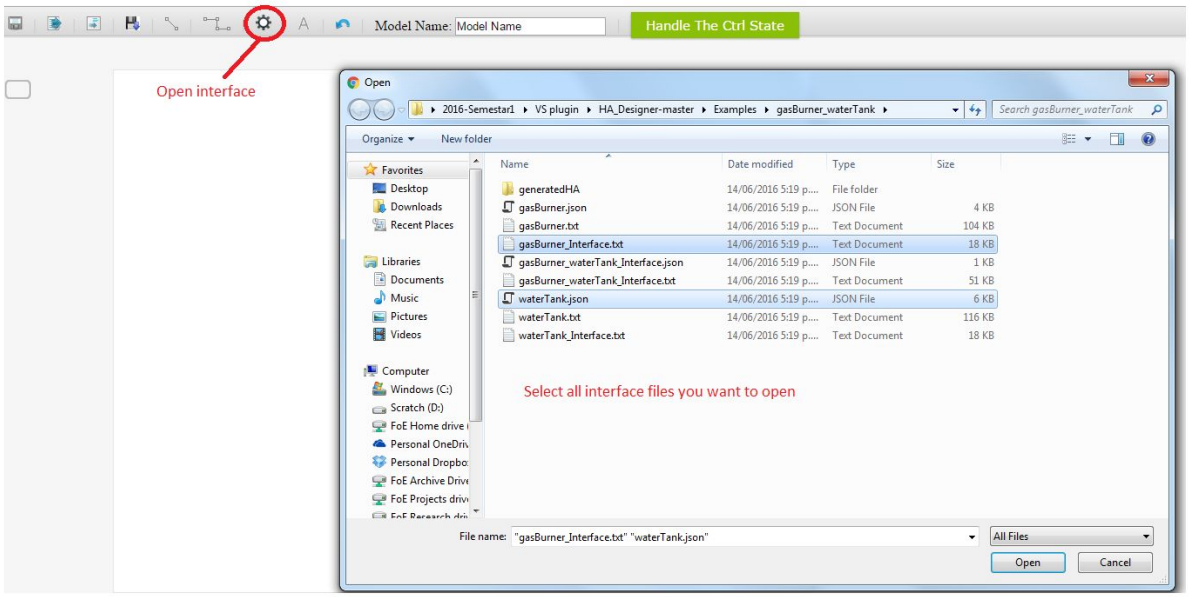



Step 5: When the designing is done, you can 1) save the diagram, so that you can reuse it by loading it to the tool next time. It will generate a file named YourModelName.txt. 2) export

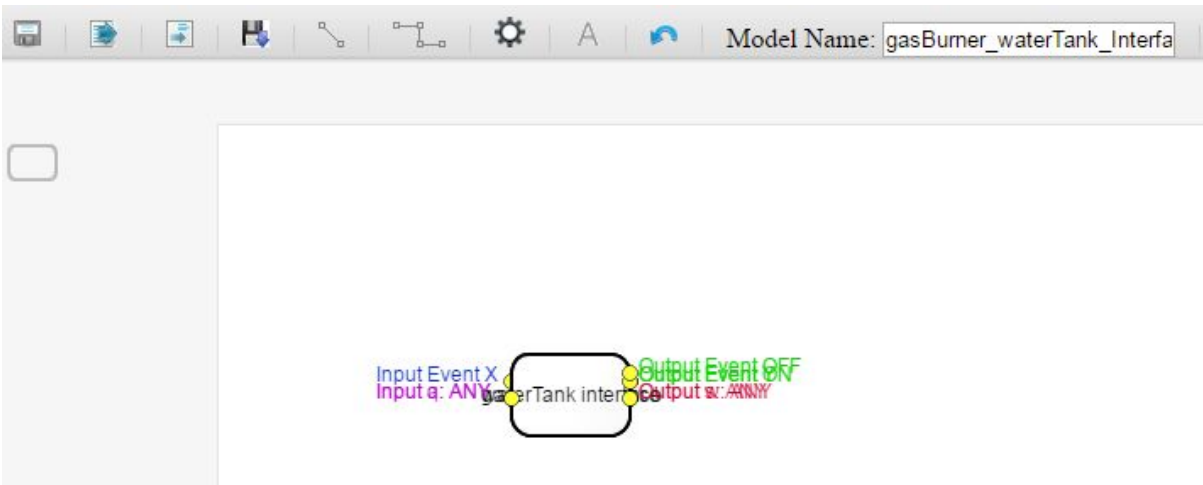
the diagram to JSON for being compiled to IEC-61499 function block. It will generate YourModelName.json file. 3) save interface for connecting with other models. This generates YourModelName_Interface.txt

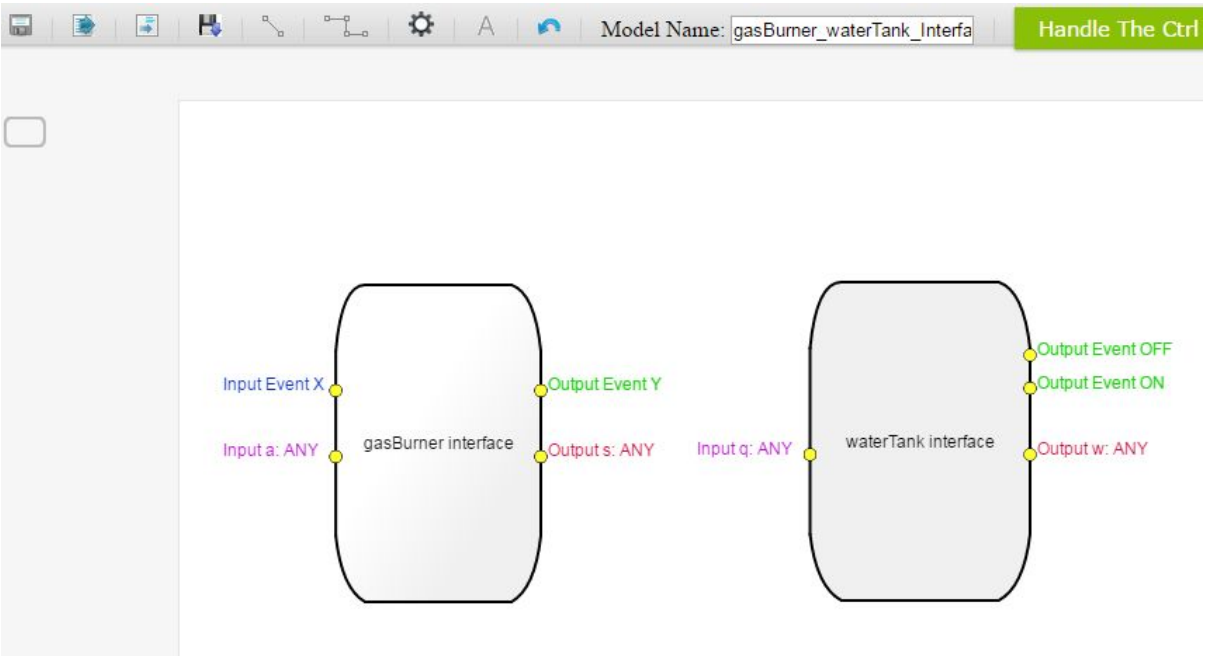

1.2 Combine multiple HA Models

If the system is composed by multiple modules, you can design each module first by following steps in 1.1. You also need to export each model to JSON, and save interface of it. Once you've done this, open all interface files which you want to connect together by clicking "Open interface" icon on the top toolbar.

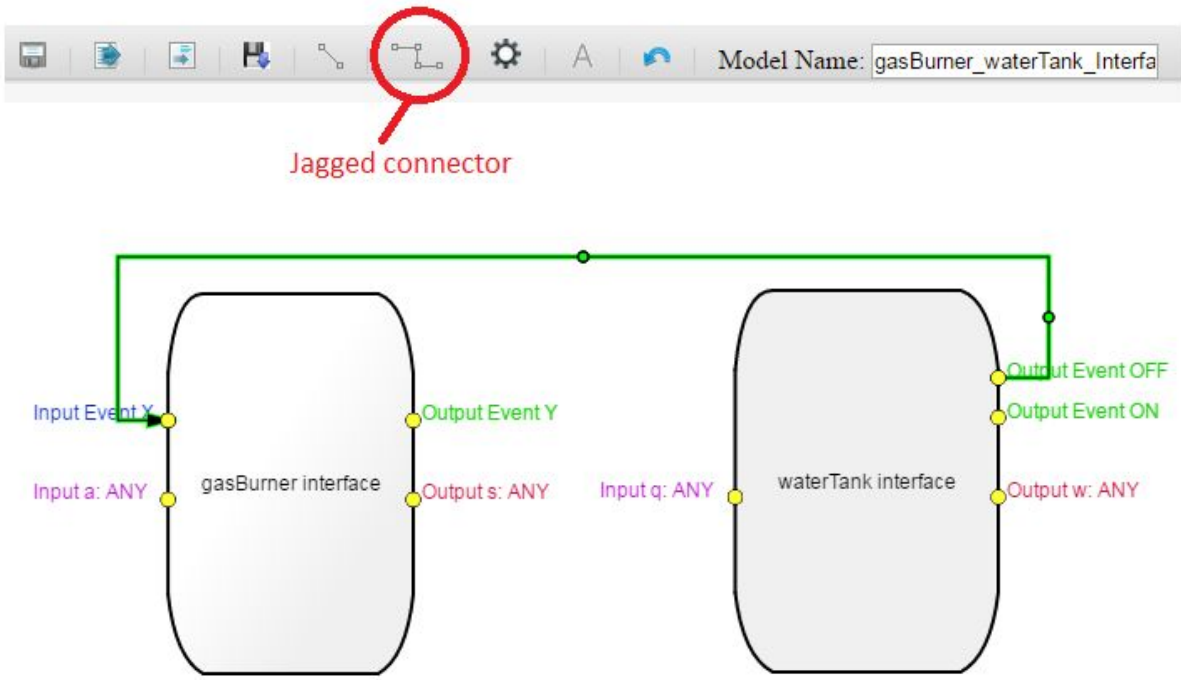



The files may overlap once they are loaded, you can adjust them by dragging and resizing.



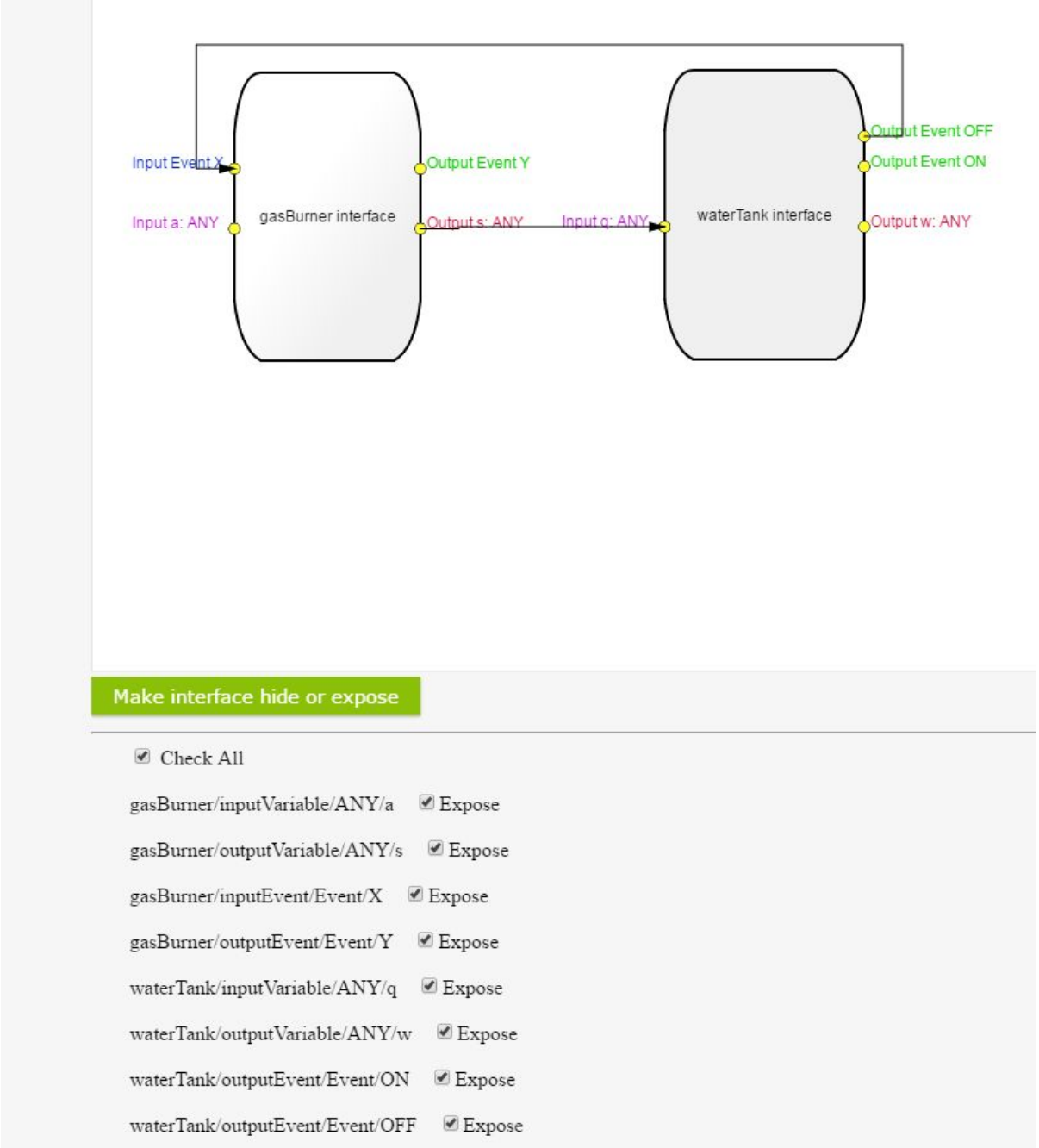



Then you can connect them by using jagged connector.




Note: you can only connect output event to input event, and output variable to input variable. Moreover, only variables with the same types can be connected, otherwise there will be a popup alert.

Once you've done the connecting, you can choose interfaces to expose in the editing area below. All interfaces are default hidden, which means they can be used only within the system. You can expose them to out side of the system by checking corresponding checkboxes.




The last step is to export the system to JSON by clicking the same button as export single module. By doing this, a file named ModelName1_ModelName2.json will be generated.

**2. JsonToIEC Compiler (source code is on:**
**https://github.com/UoA-ECE-RP/sha/tree/JsonToIEC )**
JsonToIEC compile HA_Designer generating JSON files to IEC-61499 function blocks.
2.1 Compile single module
To compile a single HA model, you can use bash file compileJson.sh. It is needed to be modified before executing by following the instruction.
The example file is as below:
*#!/bin/bash*

```
cd ./language
export PYTHONPATH=`pwd`

#on windows
#the first path is where the json file, and the second path is where you want to save the
#converted .fbt file
#python shac.py ../examples/hajson/waterTank.json fbt C:\\vsplugin\\sha\\generatedHA\\

#on linux or mac
#the first path is where the json file, and the second path is where you want to save the
#converted .fbt file
python shac.py ../examples/hajson/waterTank.json fbt
/Users/jinchen/Desktop/2016_Semester1/VS\ plugin/JsonToIEC/generatedHA/
python shac.py ../examples/hajson/gasBurner.json fbt
/Users/jinchen/Desktop/2016_Semester1/VS\ plugin/JsonToIEC/generatedHA/
```

You can run it by executing below command under JsonToIEC folder:
jinchen$ sudo ./compileJson.sh

It will generate a YourModelName.fbt file under the assigned folder.

2.1 Compile multiple modules combined system
To compile a multiple modules combined system (I.e. the JSON file you generated by connecting multiple interfaces), you can execute compileJsonToCFB bash file. It is needed to be modified before executing by following the instruction.

```
#!/bin/bash

cd ./language
export PYTHONPATH=`pwd`

#the json file should be put in the language folder for now, and the last parameter is the path
#you want to save the generated .cfb file
python shac.py gasBurner_waterTank_Interface.json cfb
/Users/jinchen/Desktop/2016_Semester1/VS\ plugin/JsonToIEC/generatedHA/
```

Note: you need to put all JSON files of the single models composing this composite system under /examples/hajson/ folder before you execute the bash, otherwise the compiler will can't find the files.

You can run the bash file  by executing below command under JsonToIEC folder:
jinchen$ sudo ./compileJson.sh

It will generate a cjtest.cfb file under the assigned folder.

**3. Import .fbt and .cfb files into BlokIDE**

3.1 Import .fbt file into BlokIDE

The .fbt file can be imported directly to BlokIDE by using “Import IEC61499 files” function. Note: you can only select folder rather than files to import, so you need to put all .fbt files into a folder, then choose that folder when importing.

3.2 Import .cfb file into BlokIDE

Before you import a .cfb file, you need to import all single models composing this composite system into BlokIDE first.

For now, BlokIDE doesn’t support .cfb file importing. Fortunately, we have a work around to do this. Firstly, open an existing BlokIDE project. Then add a new Composite function block by right click the name of the project in visual studio. After that, you should have a new file named CompositeFunctionBlock1.cfb under your project folder. So you can copy the json converted .cfb file to the same folder, and replace the existing CompositeFunctionBlock1.cfb by it. At last, you can open the CompositeFunctionBlock1.cfb in BlokIDE now.