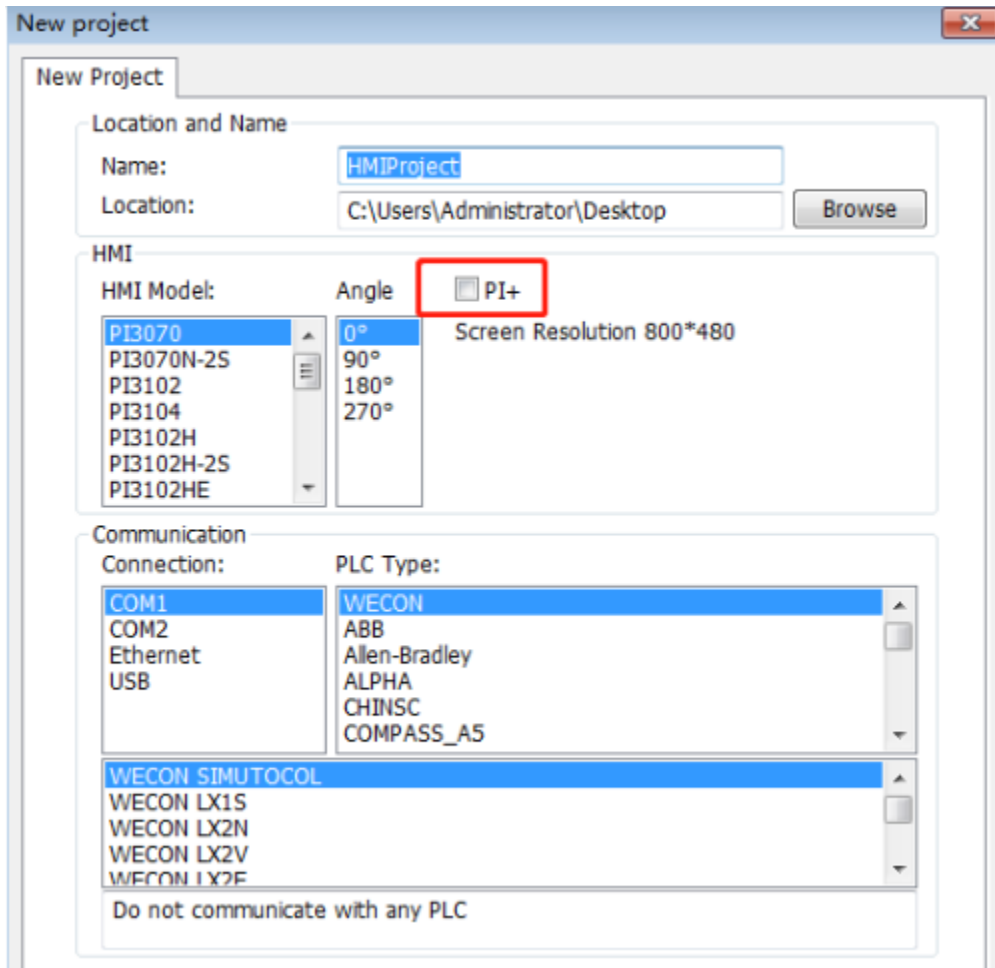
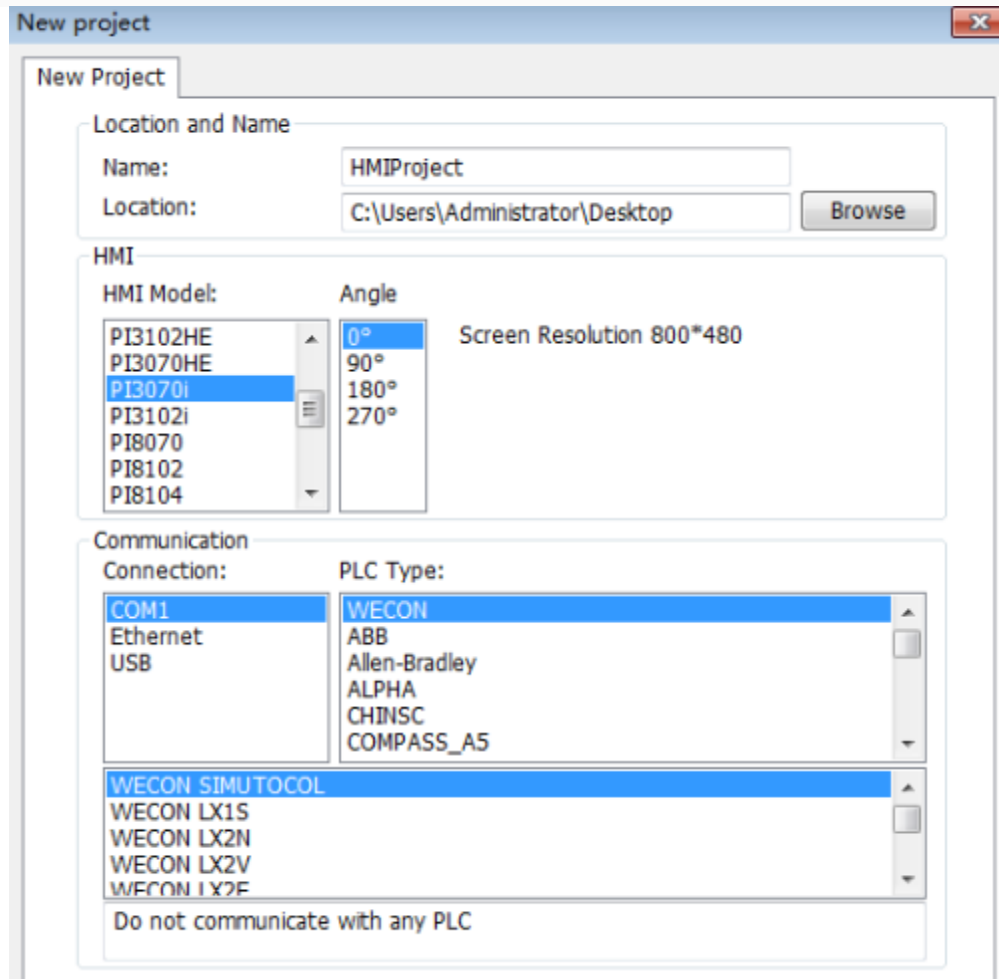


09 Lua Editor

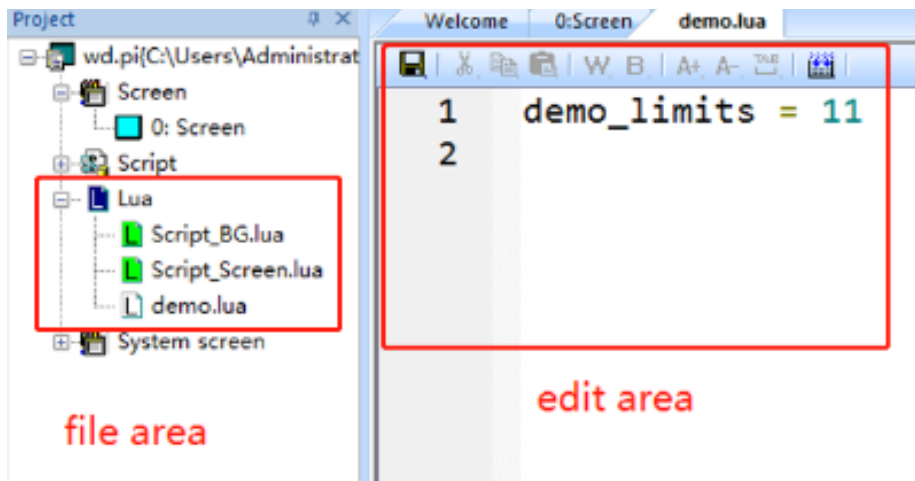
Lua script is only supported on HMI+, i series, ie series, ig series or models whose upgrade OS version to HMI V2.0 and above;





Lua Editor

The editor is divided into two areas, one is the script file area, and the other is the script editing area.



Script file area

The script file area is located on the left side, named after the Lua, It will be referred to as the Lua node hereinafter.

Lua Node

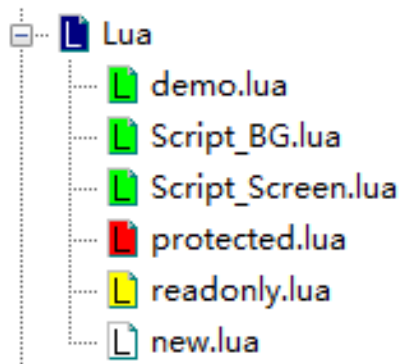
- Create new file: right-click the Lua node to select new file, and enter a name in pop-up window, (the name starts with an underscore or letter);
- Refresh list: the file edited (new/deleted/renamed) by the user externally, the configuration is not refreshed. In this case this function needs to be used to synchronize changes;

Note:

- There is no script file for the newly created project. Following two new files will automatically generate.
- Script_BG.lua: is the global background script file, which includes initialization and timing functions
- Script_Screen.lua: is a screen script file, and each screen can have initialization, timing, and close functions
- At any time, as long as there is a new operation, the software will judge whether the above two files exist, and which one is missing will be automatically filled.

Lua Child Node

After creating a new file, expand the Lua node to display several script files. The following is referred to as child nodes.



- Child node color description:
- Red: encrypted file
- Yellow: read-only file
- Green: existed file or refresh the list to turn white to green
- White: The newly created file, and the project will turn green after refreshing, protecting or reopening the project

Delete: right-click the child node to delete. The deleted node will be placed in the recycle bin.

Rename: right-click child node to rename.

Encryption: right-click the child node, select encryption. Each file can be set with an independent password(length 1-6, only uppercase/lowercase letters and numbers are accepted).

- Encryption mode: When this mode is turned on, you need to verify the password to view and edit.
- Read-only mode: This mode is opened without verifying the password, if you want to edit, you must first release the protection

Unprotect: Right-click on the child node, select Unprotect, enter the password to release.

⚠ **Note:** If user forgets the password, the script file can never be recovered, the software will not storage any passwords, developers need to keep the password by themselves carefully.

Script editor

The script editor is divided into three blocks: the upper part is the toolbar, the middle part is the editing area, and the lower part is the output area.

Tool Bar

Open any script file, the toolbar is as follows:



Save, cut, copy, paste, word address, bit address, zoom in, zoom out, display spaces and tabs, grammar check.

Find and replace, undo and redo and more functions see shortcut keys manual.

Editor Area

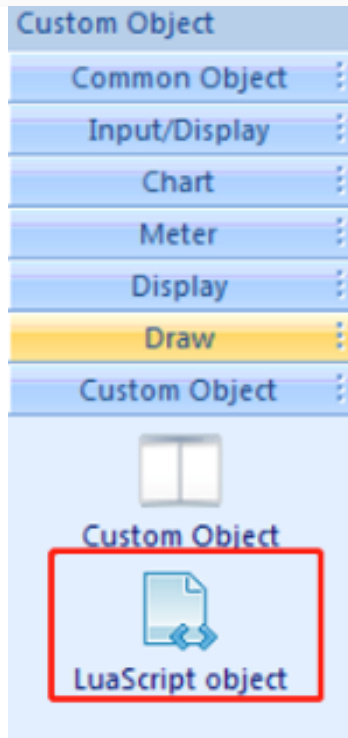
- **Highlight color:** Lua's built-in keywords has a different color with the Lua interface functions. The functions written by users will not be highlighted
- **Auto-complete:** Lua function auto-complete prompts, Lua chunk auto-complete (such as if xx then end), also supports the completion of all the words in this script file. Use the Enter key to select candidates words, use the Tab key to turn off automatic completion.
- **Automatic indentation:** The tab key can add 4 spaces
- **Calling prompt:** only available for Lua interface functions
- **Hovering hint:** for example the mouse stays near the o letter position of demo() , it will display the prototype of the function
- **(6) Shortcut keys:** In order to facilitate users, various shortcut keys have been added. Detailed description: Configure Lua shortcut keys.doc
- **Interface document:** The scripts listed in the interface document will be distinguished by the highlighted color in the editing area. You can check whether the interface is called incorrectly by judging the color change. Detailed description: Lua interface manual.

Output Area

Click the syntax button in the toolbar to perform a grammar check. If the script is written incorrectly, it will be displayed in the window. Double-click the prompt to locate the "near" line of the error.

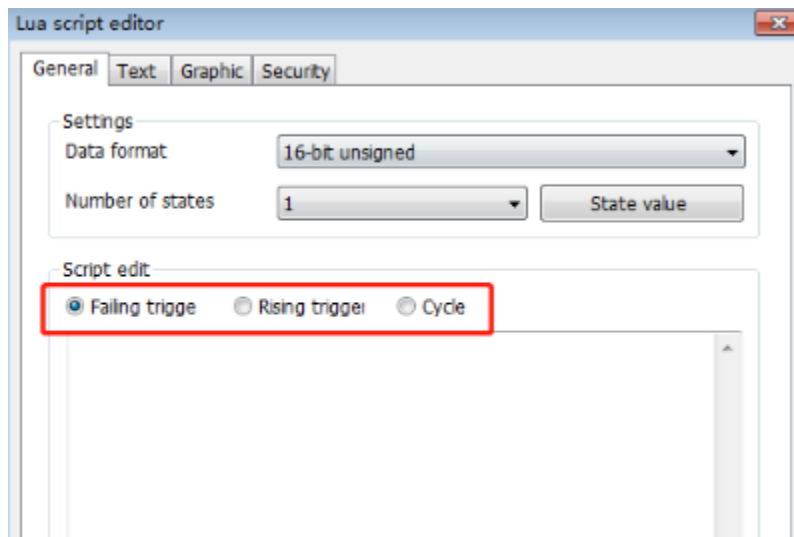
Lua Object

Lua objects are mainly used to call functions and complete user interface interactions. Normally, functions should not be written in objects. Next, we will explain how to use the objects.



How to use Lua button

Find the custom part->LuaScript on the right side, drag and draw to generate a part. Double-click the part to pop up a dialog box.



Click Lua object will trigger two actions, that is, rising and falling scripts;

- **Falling script:** after the button is pressed, the code chunk of the area is executed;
- **Rising script:** after the button is raised, the code chunk in the area is executed;
- **Timing script:** periodically execute the code.

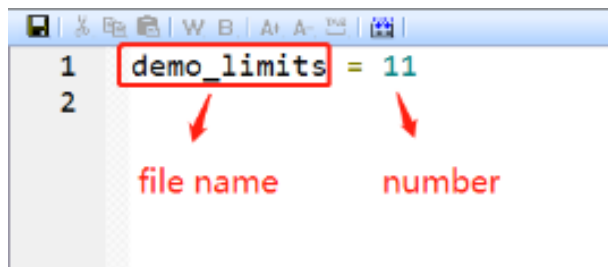
Note:

- The timing script is not commonly used because it is similar to the screen background script function. It is not recommended to write and call function here. It should be written in Script_Screen.lua, which is convenient for editing, searching, and replacing operations.
- It is not allowed to write endless loops or call functions that will loop endlessly in the widget. Once used, the entire screen will be stuck, and any buttons will no longer be used. We have to restart the screen to restore.

Platform

File header identification

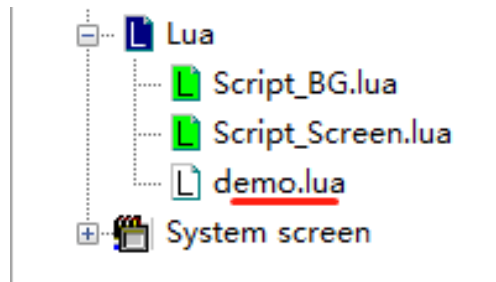
Any new file will automatically generate a line of code, as shown below:



```
1 demo_limits = 11
2
```

The format of this variable: file name_limits = number

The file name comes from the newly created file name (without suffix):



- **_limits:** Fixed format, cannot be modified
- **Description :**
- **Tens digit=1:** indicates that the function or global variable defined in this file can be used in the background code file (Script_BG.lua). If it is 0, it is not available.
- **Ones digit=1:** indicates that the functions or global variables defined in this file can be used in the screen code file (Script_Screen.lua) or LuaScript components. If it is 0, it is not available.

Note: If you rename the file manually, you must modify the limits in the file accordingly, otherwise the script file will not take effect.

Built-In script file

As mentioned above, the first new build will automatically generate two script files. These two files cannot be deleted and renamed, which will be explained in detail here.

Script_BG.lua is a global background script file, which includes initialization and polling functions

```
1  Script_BG_limits = 11
2
3  function we_bg_init()
4
5  end
6
7  function we_bg_poll()
8
9  end
```

- we_bg_init(): global initialization, that is, execute once when power on.
- we_bg_poll(): Global polling, that is, the cycle is repeated after power on.

Script_Screen.lua is a screen script file, and each screen has an initialization, polling, and closing functions

```
1  PIScript_Screen_version = 1.0
2  PIScript_Screen_limits = 11
3
4  function we_scr_init_0()
5
6  end
7
8  function we_scr_poll_0()
9
10 end
11
12 function we_scr_close_0()
13
14 end
```

- we_scr_init_0(): execute this function when screen 0 is initialized
- we_scr_poll_0(): when in picture 0, the function is executed repeatedly
- we_scr_close_0(): execute this function when screen 0 is closed

The above is the initialization function of screen 0. For screen 100, you can replace 0 with 100. You can add your own function: function we_scr_init_100() end

Note: The above function has a fixed name. If it is modified, it will not cause crash, but the function can be found.

Example: After the screen 0 initialization function is changed to we_scr_init_100(), the script will not be executed when screen 0 is initialized, and then no script will be executed, and nothing will be done because the "we_scr_init_0" function cannot be found.

Script type (operating mechanism)

There are two types of scripts:

- Active: script automatically executed by HMI

- Passive: Script executed only after user trigger

The active types includes:

- Global `_bg_` related functions in `Script_BG.lua`
- Screen `_scr_` related functions in `Script_Screen.lua`

Polling script in Lua object

The passive type scripts include: The rising and falling script in Lua object.