Foreword

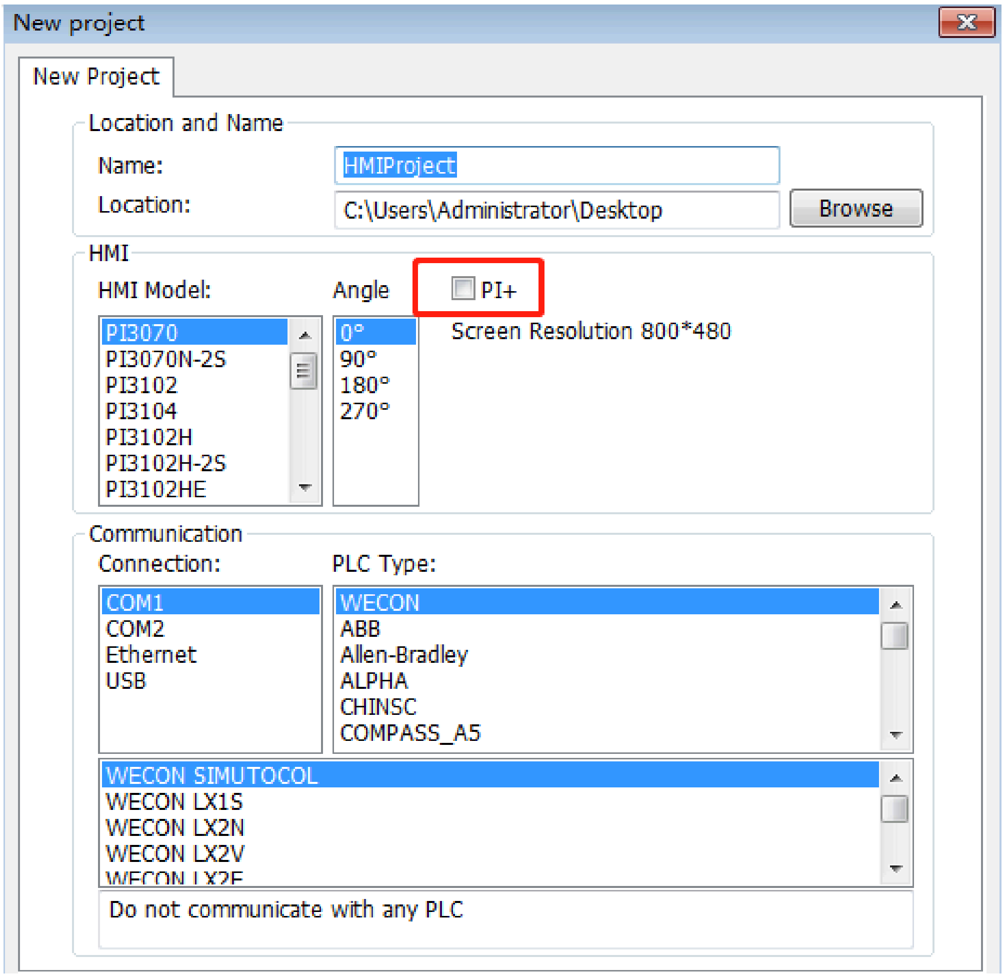
last modified by Joey

on 2022/06/16 15:07

Table of Contents

Lua script is currently only available in PI+ or models ending with i(like PI3070i). Only these two types of HMI can use the Lua editor and object to develop lua scripts. The following operations are based on PI3070i project.

This article only introduces the use of lua object in PIStudio.


New project

New Project

Location and Name

Name: HMIProject

Location: C:\Users\Administrator\Desktop

Browse

HMI

HMI Model: Angle ☐ PI+

Screen Resolution 800*480

PI3070
PI3070N-2S
PI3102
PI3104
PI3102H
PI3102H-2S
PI3102HE

0°
90°
180°
270°

Communication

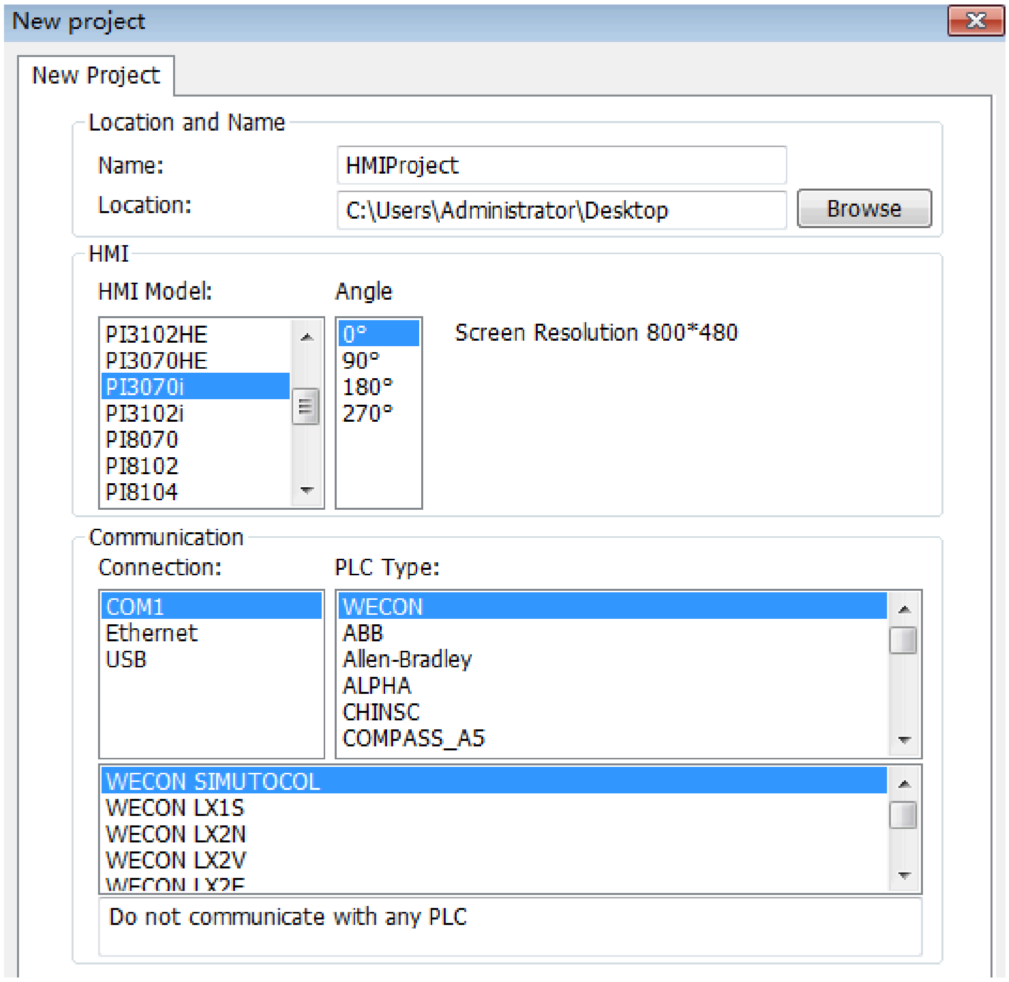
Connection: PLC Type:

COM1
COM2
Ethernet
USB

WECON
ABB
Allen-Bradley
ALPHA
CHINSC
COMPASS_A5

WECON SIMUTOCOL
WECON LX1S
WECON LX2N
WECON LX2V
WECON LX2E

Do not communicate with any PLC



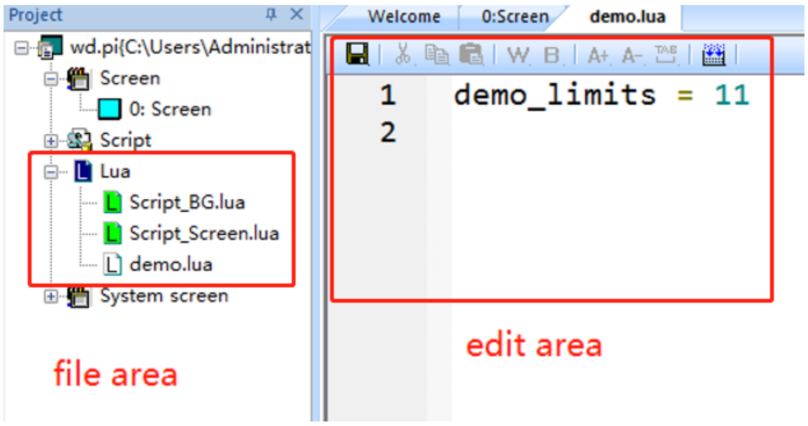
Name: HMIProject
Location: C:\Users\Administrator\Desktop

HMI Model: PI3102HE, PI3070HE, PI3070i, PI3102i, PI8070, PI8102, PI8104
Angle: 0°, 90°, 180°, 270°
Screen Resolution 800*480

Connection: COM1, Ethernet, USB
PLC Type: WECON, ABB, Allen-Bradley, ALPHA, CHINSC, COMPASS_A5
WECON SIMUTOCOL, WECON LX1S, WECON LX2N, WECON LX2V, WECON LX2E
Do not communicate with any PLC

1 Lua editor

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



```
demo_limits = 11
```

1.1 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.

1.1.1 Lua node

1. 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);

2.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.

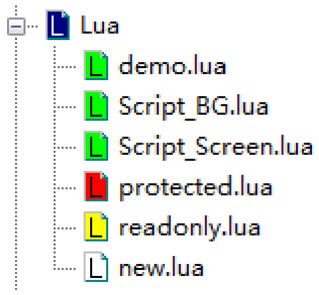
Script_BG.lua: is the global background script file, which includes initialization and timing functions

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.

1.1.2 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

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

(2) Rename: right-click child node to rename.

(3) 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

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

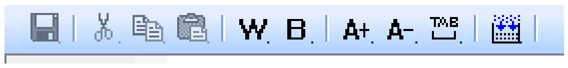
#Note: If user forgets the password, the script file can never be recovered, the software will not storage any passwords, developers need to keep the password by themselves carefully.

1.2 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.

1.2.1 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.

1.2.2 Editor area

(1) 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

(2) 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.

(3) Automatic indentation: The tab key can add 4 spaces

(4) Calling prompt: only available for Lua interface functions

(5) 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

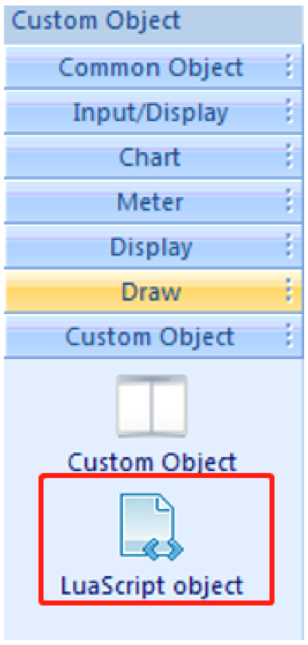
(7) Interface document: The scripts listed in the interface document will be distinguished by the highlighted color in the editing area. You can check whether the interface is called incorrectly by judging the color change. Detailed description: Lua interface manual.

1.2.3 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.

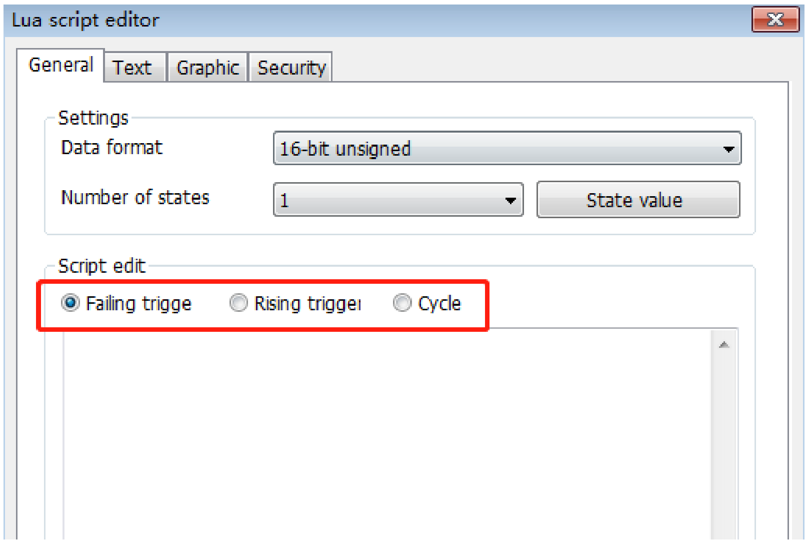
2 Lua object

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.



2.1 How to use LuaScrip object

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;

(1) Falling script: after the button is pressed, the code chunk of the area is executed;

(2) Rising script: after the button is raised, the code chunk in the area is executed;

(3) Timing script: periodically execute the code.

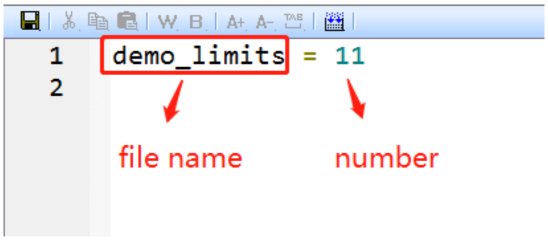
#Note: The timing script is not commonly used because it is similar to the screen background script function. It is not recommended to write and call function here. It should be written in Script_Screen.lua, which is convenient for editing, searching, and replacing operations.

Warning: It is not allowed to write endless loops or call functions that will loop enlessly 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.

3 Platform

3.1 File header identification

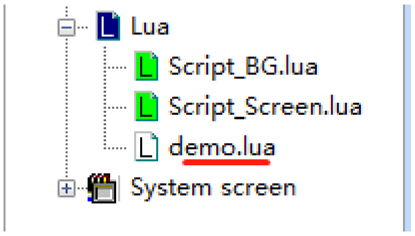
Any new file will automatically generate a line of code, as shown below:



```
demo_limits = 11
```

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.

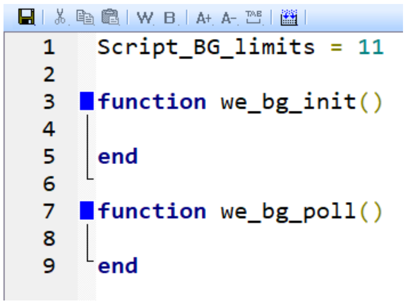
Ones digit=1: indicates that the functions or global variables defined in this file can be used in the screen code file (Script_Screen.lua) or LuaScript components. If it is 0, it is not available.

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

3.2 Special 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.

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

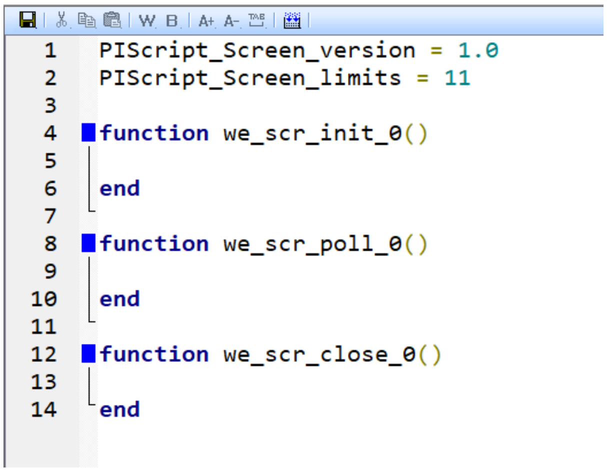


```lua
Script_BG_limits = 11

function we_bg_init()

end

function we_bg_poll()

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.

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



```lua
PIScript_Screen_version = 1.0
PIScript_Screen_limits = 11

function we_scr_init_0()

end

function we_scr_poll_0()

end

function we_scr_close_0()

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_0(), 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.

3.2 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.