



Lua script to send SMS

Website: <http://www.we-con.com.cn/en>



Overview

This document describes the Wecon PI3070ig HMI to customize communication with the ATC SMS module via Lua script to send SMS,and introduces it in 3 parts:

- Protocol configuration
- Project introduction
- Script function
- Cable Wiring

I Protocol configuration

1.1 PIStudio

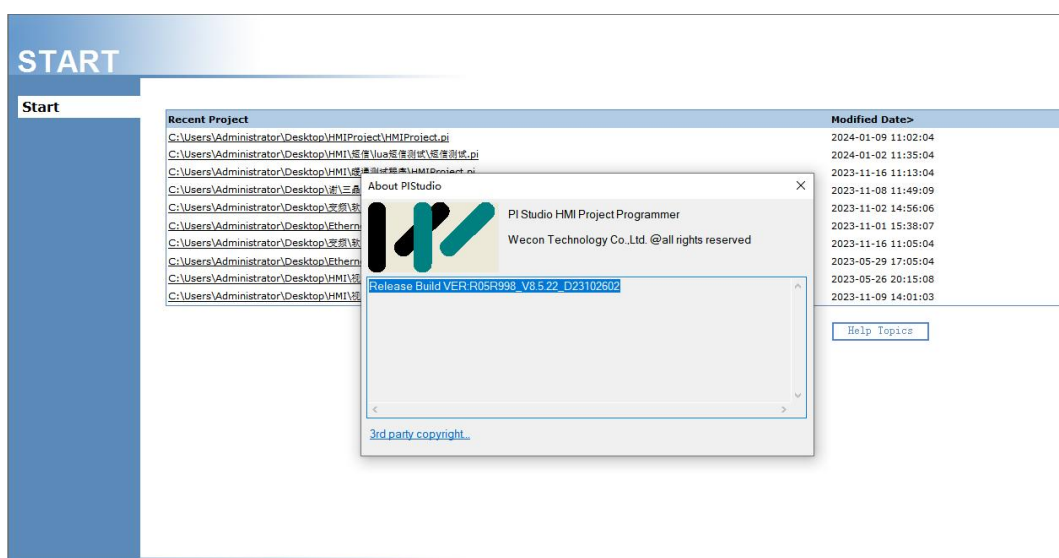


Figure 1-1

1.2 Configuring Protocol Parameters

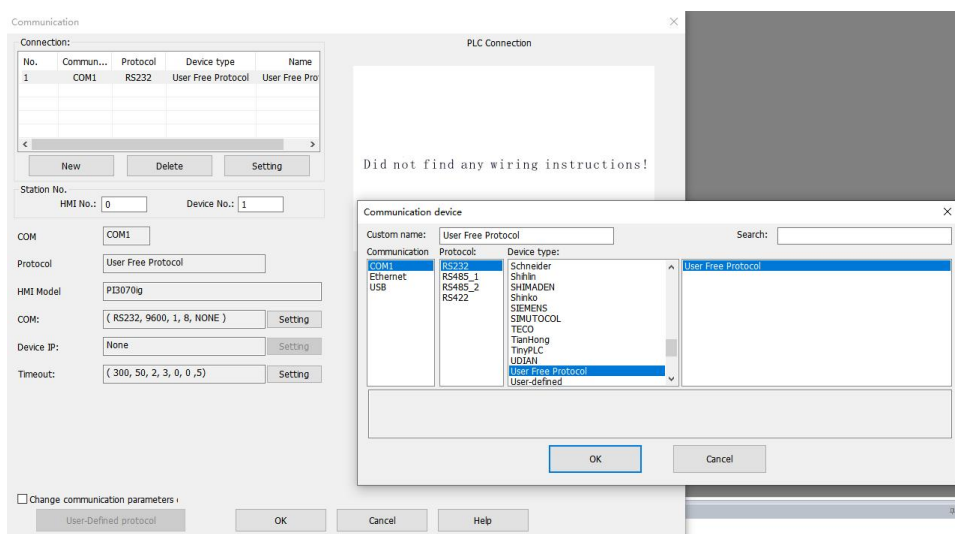


Figure 1-2

1.3 ATC SMS module

Connection to HMI using 232



Figure 1-3

II Project introduction

2.1 Project introduction

- ① Input the cell phone number to send SMS in number, don't need to include the country code.
- ② Input the content of the sent SMS in Content. There are five groups of phone numbers and text messages that can be added to this project.
- ③ "Debug Window": After entering the cell phone number and SMS content, click the debug button to bring up the HMI information display box.
- ④ "OFF": Click the OFF button to start the Lua script and the HMI starts sending SMS.

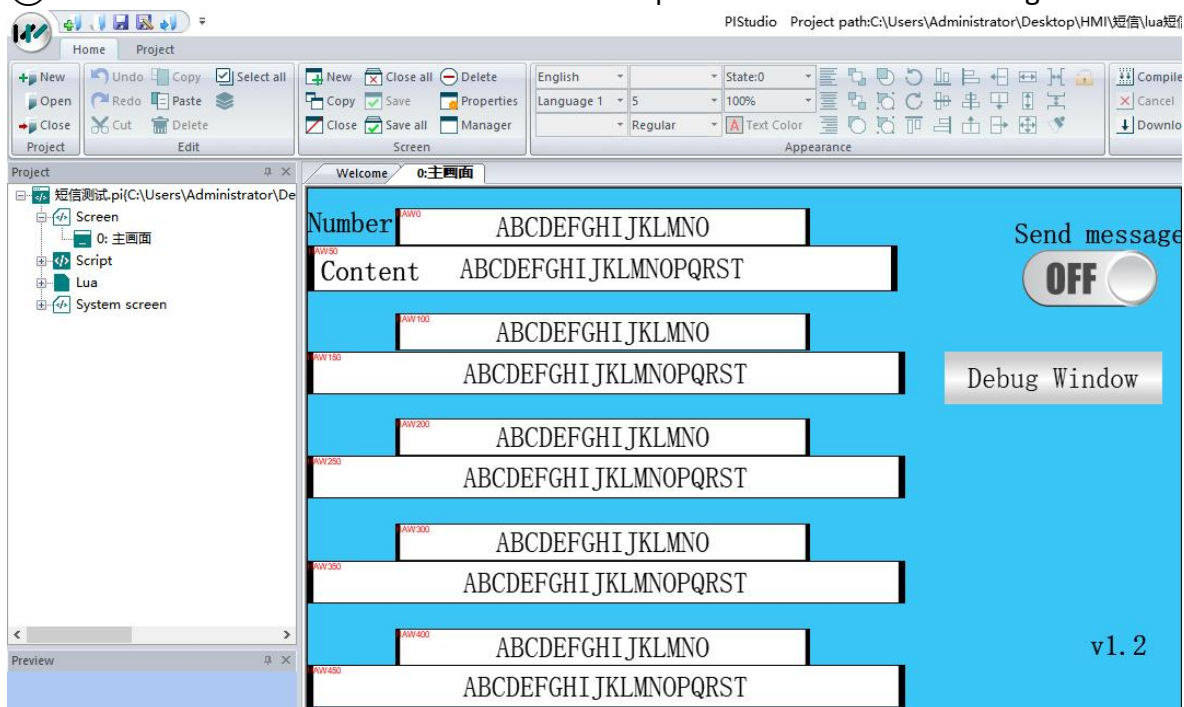


Figure 2-1

⑤ After programming,[Home] → [Compile],you can download the project into HMI when compilation complete

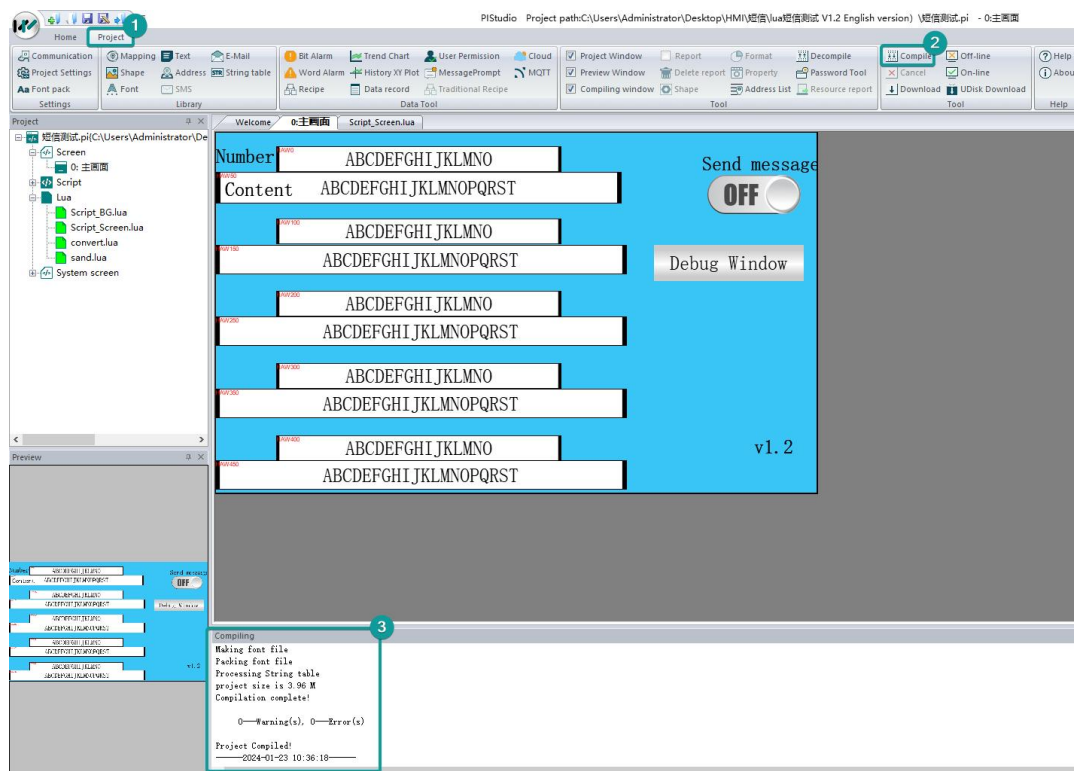


Figure 2-2

2.2 Downloading project

[Home]→[Download]→[PC Port]→[PC to HMI]→[OK].

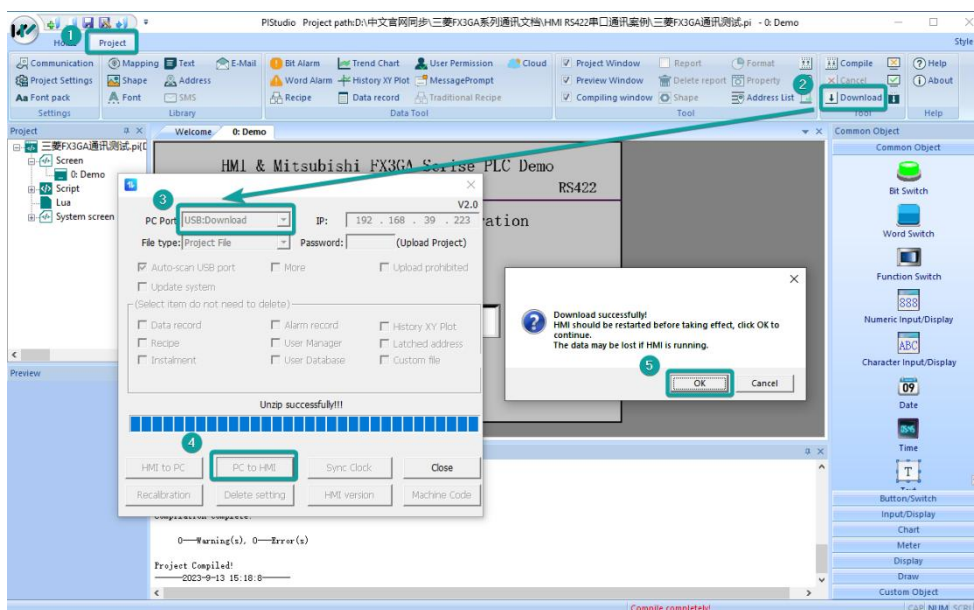


Figure 2-3

III Script function

3.1 Subroutines

```
convert_limits = 11
--*****Standard function*****
--[[
Used for packing an array into a string by bytes
-Put the table as the parameter tab
-The return parameter will be string
]]
function we_arrayToChar(tab)
    if type(tab) ~= "table" then
        return nil
    end
    local charString=""
    for i=1,#tab do
        charString=charString..string.pack("<B", tab[i])
    end
    return charString
end

--[[
Unpack a string into an array by bytes
-String as para
Will return an array
]]
function we_charToArray(str)
    if type(str) ~= "string" then
        return nil
    end
    return {string.byte(str, 1,#str)}
end
```

3.2 Main

```
sand_limits = 11

--V1.2 version English

--AT+CMGF=1           Message text Model
--AT+CMGS="+86XXXXXXX" set the phone number
```

```

local commNo=1                                -- Custom
local AT = {0x41, 0x54, 0x0D, 0x0A}          --0X0D, 0X0A / means line break
local END = {0x0D, 0x1A}                     --END symbol: 0X0D 0X1A
-- please set the END when your message Content finish.
local Text_Model = {0x41, 0x54, 0x2B, 0x43, 0x4D, 0x47, 0x46, 0x3D, 0x31, 0x0D, 0x0A}
--text Model
local socket = require("socket")              --delay funciton

function sendCompleteAT(tab)--Sending instructions
    usrprotocol.io_flush(commNo) --Clean cache
    local timeout = 1000
    local length = 4
    local char = we_arrayToChar(tab)
    local re0,rel = usrprotocol.io_write_read(commNo,char,length ,timeout)
    print("char",char)

    if re0~=nil then
        dxtab = we_charToArray(rel)
        for i=1,#dxtab do
            local _ = we_bas_setword(we_bas_newnoaddr("@W_HDW0",i-1),dxtab[i])
        end
    end
end

end

function send_AT() --Send AT
    sendCompleteAT(AT)
end

function send_Text_Mode() --Set Text mode
    sendCompleteAT(Text_Model)
end

function send_Number_Message()
    for i = 1, 5 do

        local phone = we_bas_getstring(we_bas_newnoaddr("@W_HAW0", (i-1)*100),11)
--Number
        local message = we_bas_getstring(we_bas_newnoaddr("@W_HAW50", (i-1)*100),11)
--message content,length can be modify

        -- 0x22= "      0x2B, 0x38, 0x36= +86
        local array_phone={0x41, 0x54, 0x2B, 0x43, 0x4D, 0x47, 0x53, 0x3D, 0x22, 0x2B,
0x38, 0x36} --AT+CMGS="+86
    end
end

```

```
        if phone ~= nil and phone ~= "" then
            for j = 1, 11 do
                --The number is inserted into the array and transformed into
the pattern I want
                array_phone[j+12]=string.byte(phone, j)
                --print(string.byte(phone, j))
            end
            array_phone[24] = 0x22 -- set quotation mark
            array_phone[25] = 0x0D
            array_phone[26] = 0x0A
        end
        sendCompleteAT(array_phone)
        socket.sleep(5)

        if message ~= nil and message ~= "" then
            local array_message = {}
            for v = 1, #message do
                array_message[v] = string.byte(message, v)
                --print(string.byte(message, v))
            end
            sendCompleteAT(array_message)
            sendCompleteAT(END)
            socket.sleep(5)
        end
    end
end

function luaButtonTriggerSendATComm()
    send_AT()
    socket.sleep(1)

    send_Text_Mode()
    socket.sleep(1)


    send_Number_Message()

End
```

IV Cable Wiring

4.1 Cable Wiring

COM1PIN Definition



PIN	Definition	PIN	Definition
1	RS422 TX+/RS485 A1+	2	RS232 RXD
3	RS232 TXD	4	RS485 B2-
5	GND	6	RS422 TX-/RS485 B1-
7	RS485 A2+	8	RS422 RX-
9	RS422 RX+		

Figure 4-1

Chapter 1 Cambria, 二号, 3 倍行距, 左对齐加粗, 第一个词首字母大写即可

1.1 Cambria, 小二, 2 倍行距, 左对齐加粗, 第一个词首字母大写

1.1.1 Cambria, 小三, 2 倍行距, 左对齐加粗, 第一个词首字母大写

正文: Calibri, 小四, 1 倍行距, 齐头式不缩进

1.1.1.1 Cambria, 小三, 2 倍行距, 左对齐, 加粗【第一个词首字母大写即可】

正文: Calibri, 小四, 1 倍行距, 齐头式不缩进

1.1.1.2 Cambria, 小四, 1.5 倍行距, 左对齐, 加粗【第一个词首字母大写即可】

正文: Calibri, 小四, 1 倍行距, 齐头式不缩进

(1) 一级序号标题: calibri, 小四, 1.5 倍行距, 齐头式不缩进

1) 二级序号标题: calibri, 小四, 1.5 倍行距, 文本前缩进 0.5 个字符, 悬挂缩进 0.5cm

正文: Calibri, 小四, 1 倍行距, 齐头式不缩进

2) 正文二级标题: calibri, 小四, 1.5 倍行距, 文本前缩进 1 个字符, 悬挂缩进 0.5cm

① 三级序号标题, calibri, 小四, 1.5 倍行距, 首行缩进 1 个字符

正文: Calibri, 小四, 1 倍行距, 齐头式不缩进

② 正文三级标题, calibri, 小四, 1.5 倍行距, 首行缩进 1 个字符

③ 正文三级标题, calibri, 小四, 1.5 倍行距, 首行缩进 1 个字符

(2) 正文一级标题: calibri, 小四, 1.5 倍行距, 齐头式不缩进

(3) 正文一级标题: calibri, 小四, 1.5 倍行距, 齐头式不缩进

📌 **Note:** calibri, 小四, 1 倍行距。如果只有一段话, 直接跟在后面。如果有好几点, 直接另起一行, 如下:

📌 **Note:** calibri, 小四, 1.5 倍行距。

📌 **Note:**

📌 空一格, calibri, 小四, 1 倍行距。

📌 空一格, calibri, 小四, 1 倍行距。

📌 空一格, calibri, 小四, 1 倍行距。

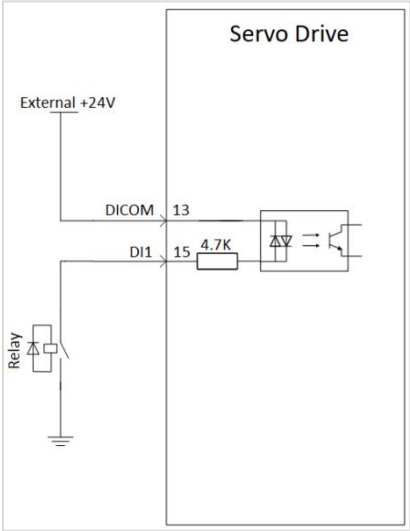


Figure 1-1 空一格 calibri, 小四, 段前段后 3 磅, 1 倍行距

PIN	Function	calibri, 小四, 1 倍行距, 白色, 加粗
1	RX-	calibri, 小四, 1 倍行距, 内容多的某些列, 左对齐, 其他居中
2	RX+	
3	TX-	

Table 1- 1 空一格, calibri, 小四, 段前段后 3 磅, 1 倍行距