

## 4G-DTU 配置工具使用方法

## 一、使用须知

4G-DTU 有两个型号：XY-A724G 和 XY-A724D。

XY-A724G 型号的 DTU 包含一个 RS232 串口（串口 1）、一个 RS485 串口（串口 2）和一个 TTL 串口（串口 3），其中 RS232 串口的初始波特率为 9600，RS485 串口和 TTL 串口的默认波特率为 115200。

XY-A724D 型号有两路 485 串口，分别对应串口 1 和串口 2，默认波特率均为 115200。

此外还可以通过 AT 端口进行配置。

## 二、配置步骤

## 1、第一步

打开串口。根据 DTU 型号和串口号选择对应波特率。

XiaoYiIoT\_DTU\_Config\_Tool\_v1.0.5

\* 端口号: USB-SERIAL CH340 (COM) \* 波特率: 9600 数据位: 8 校验位: none 停止位: 1 关闭串口

## 2、第二步

点击读取配置按钮，读取配置。配置解析正常后，即可进行自定义配置。

Figure 1 shows the configuration interface of the XiaoYIOT\_DTU\_Config\_Tool v1.0.5 software. The interface includes a top bar with the software name and version, and a main area with various configuration tabs. The '基本参数' (Basic Parameters) tab is selected, showing settings for mode, IMEI, data flow, and other parameters. A red box highlights the '读取配置' (Read Configuration) button in the top right corner. A red arrow points from this button to a secondary window titled '时间戳' (Timestamp), which displays a list of configuration parameters and their values, including flow, passon, warn, adc0, and various pins and delays.

### 3、第三步

在配置完成之后点击“写入配置”，将配置写入 DTU 中，DTU 返回“OK”，随后自动重启，使得配置生效。



注意：上面的“写入配置”按钮只能写入除 **modbus** 配置以外的配置，**modbus** 配置需要在该配置页面内单独进行读取与写入，如下。



### 三、配置参数

#### 1、基本参数

一般保持默认配置即可。如果要修改串口波特率，那么需要修改基本参数中的“串口分帧超时”。例如：用户需要将串口波特率改成 9600，那么建议将“串口分帧超时”修改为 150ms ~ 200ms。

基本参数 串口参数 网络通道参数 预置信息 自动采集任务 GPIO 数据流 预警 任务

模式: ☒ 透传 ☐ 单片机控制

是否加设备识别码IMEI: ☒ 不加 ☐ 加

报文转换(bin -- hex): ☒ 不换 ☐ 转换 提示: 如果启用数据流模板, 这里选择“不换”

首次登录服务器发送注册信息: ☒ 不发 ☐ 发送{csq:rssi,imei:imei,iccid:iccid,ver:Version} ☐ 发送HEX报文"13,12345,12345"

☐ 自定义

参数版本号:  提示: 范围 1~n

每分钟最大串口流量(Byte):  提示: 为0表示不启用

是否启用自动更新: ☒ 否 ☐ 是

串口分帧超时:  提示: (单位: ms 默认25ms, 范围10-2000)

电源模式: ☒ 正常 ☐ 节能

网络通道监听:

网络分帧超时:  提示: 同串口分帧超时, 默认 0

日志输出: ☒ 开启 ☐ 关闭

#### 2、串口配置

初始状态中所有串口都是打开的，RS485 方向控制 GPIO 默认选空，以避免 485 串口不可用。用户修改了串口波特率后，建议去“基本参数”中修改“串口分帧超时”参数，以避免在该波特率下配置返回“JSON ERROR”的问题。

修改了波特率并成功写入配置之后，相应的，打开串口的波特率也要改为修改后的波特率。

#### 4、网络通道参数

网络通道配置，可以让 DTU 接入用户私有服务器或其他云。

以 MQTT 配置为例，根据提示填入参数即可。

(1) 其中“自动任务时间间隔”参数是配合“自动采集任务”使用的，如果未启用“自动采集任务”，那么这里保持默认配置即可。

(2) 配置订阅或发布主题参数时，如果只订阅一个主题，那么直接填入主题即可。如果订阅了多主题，那么格式为：主题 1;QOS;主题 2;QOS;主题 3;QOS。中间用英文分号隔开，结尾无分号。发布主题同理。

MQTT的登录密码:

MQTT的会话保持位: ☐ 持久会话 ☒ 离线自动销毁

订阅消息主题:

发布消息主题:

MQTT的QOS级别: ☒ 0 ☐ 1 ☐ 2

MQTT的publish参数retain: ☒ 0 ☐ 1

(3) 建议通道 1 绑定串口 1, 通道 2 绑定串口 2。

MQTT通道绑定的串口ID: ☒ 1 ☐ 2 ☐ 3

(4) 在“主题添加 imei”参数中, 如果选“是”, 那么会自动在订阅和发布的主题后面加 imei。例如: 假设设备 imei 为“876543211234567”, 用户填入的订阅的主题为“/msg”, 那么最终设备订阅的主题为:“/msg/876543211234567”。如果选择“否”, 设备订阅或发布的主题即为用户输入的主题。

MQTT的QOS级别: ☒ 0 ☐ 1 ☐ 2

MQTT的publish参数retain: ☒ 0 ☐ 1

MQTT通道绑定的串口ID: ☒ 1 ☐ 2 ☐ 3

客户端ID:  提示: 不填系统用IMEI做客户端ID

主题添加IMEI: ☒ 是 ☐ 否 提示: 默认添加 (/自定义主题/IMEI)

transport: ☒ tcp ☐ tcp\_ssl

(5) 在“MQTT 遗嘱”参数中, 如果有需要, 可填入遗嘱的主题, 设备在离线后, 该主题收到的消息为设备的 imei。

MQTT的遗嘱:  提示: 可不填

## 5、自动采集任务

在自采集任务中，“采集等待”指的是两条指令之间的时间间隔，例如设置了采集等待时间为 1000ms，那么在下发了 cmd1 之后，等待 1000ms 再下发第二条指令 cmd2。在下发完一轮指令后，第二轮指令下发的间隔时间由“网络通道参数”配置中的“自动任务间隔时间”决定。下发的指令为 16 进制字符串。

基本参数串口参数网络通道参数预置信息自动采集任务GPIO数据流任务

串口1串口2串口3

启用

不启用

采集等待：

1000

(单位: ms)提示: 1~10000

cmd1:

01 03 00 01 00 04 4A 5C

cmd2:

01 03 00 01 00 04 4A 5C

+ 添加执行指令

## 6、GPIO

目前 DTU 的灯的状态为：

- (1) 通电后，电源灯（绿灯）长亮。
- (2) 网络灯（红灯）：快速闪烁（亮 100ms，灭 100ms），表示正在注册 GMS。  
较慢闪烁（亮 500ms，灭 500ms），表示正在附着 GPRS。  
慢闪（亮 100ms，灭 1900ms），表示成功连接服务器。
- (3) mode 灯（黄灯）：连接服务器后长亮。

为保持上述状态，请将本配置置为“不启用”。如若遇到 led 灯状态异常，可检查是否开启了本配置。

\* 端口号USB-SERIAL CH340 (COM)\* 波特率9600

关闭串口

基本参数串口参数网络通道参数预置信息自动采集任务GPIO数据流任务

启用

不启用

## 7、数据流

可以在这里对上行和下行数据进行处理。数据流模板绑定的通道与网络通道配置的通道一一对应。数据流模板函数需使用 lua 语言编写。数据流模板函数的格式为：

function

--用 str 变量接收上行或下行数据，三个点是固定写法，表示上行或下行数据。

local str = ...

--这里对 str 进行处理，最后 return 处理后的数据

return str

end

(1) 例如，设备采集到的数据为 modbus\_rtu 协议格式的，而服务器要求上传 JSON 格式数据，就可以通过数据流模板进行处理。处理示例：

```
function
  local str = ...
  local idx, crc = pack.unpack(str:sub(-2, -1), "H")
  local tmp = str:sub(1, -3)
  if crc == crypto.crc16("MODBUS", tmp) then
    local idx, addr, fun, length = pack.unpack(tmp:sub(1, 3), ">b3")
    local data = str:sub(4, -3)
    local t = {}
    local values = {}
    t.mei = misc.getmei()
    t.addr = addr --从机地址
    t.fun = fun --功能码
    t.byte = length --数据字节数
    if regAddr ~= nil then
      t.regAddr = regAddr --起始寄存器地址
    end
    if fun == 0x03 or fun == 0x04 then
      for i = 1, length, 2 do
        local idx, val = pack.unpack(data, ">H", i)
        table.insert(values, val)
      end
      t.data = values
      return json.encode(t)
    elseif fun == 0x01 or fun == 0x02 then
      for j = 1, length, 1 do
        local idx, val = pack.unpack(data, "b", j)
        table.insert(values, val)
      end
      t.data = values
      return json.encode(t)
    else
      return str
    end
  end
  return str
end
```

(2) 在配置了 MQTT 多主题发布时，可以通过“发送数据流模板”将消息发布到指定主题。在数据流模板函数最后 return 数据时，在后面添加主题对应的下标。

假设发布主题为：topic1;0;topic2;0;topic3;0

现需要向 topic2 主题发布消息，可以将其视为一个 lua 数组：{ topic1, 0, topic2, 0, topic3, 0 }, 那么其中每个元素的下标依次为 1,2,3,4,5,6，由此可知 topic2 的下标为 3。

那么数据流函数格式为：

function

local str = ...

return str,3

end

这里直接将所有数据发布到 topic2 主题了，用户可以根据自己的需要，在函数中写条件判断，以决定数据发布到对应的主题。

(3) MQTT 订阅了多个主题时，如果想知道接收的消息对应的是哪个订阅的主题，可以在“接收数据流模板”中多解析一个主题字段。如下：

```
function
  local str, topic = ...
  local jsonTable = {}
  jsonTable["msg"] = str
  jsonTable["topic"] = topic
  return json.encode(jsonTable)
end
```

上面的模板函数中的“topic”就是此次消息的主题，这里将消息和主题组成了一个 JSON 字符串返回了，用户可根据自己的需求进行其他处理。

## 8、任务

这里也用 lua 语言写任务函数。但是不能通过 `local str = ...` 的方式接收参数。可以在这里写一些定时任务之类的。例如定时向串口写入数据之类的。

**特别注意，如果编写的代码中有错误，那么 4G-DTU 内的程序将无法正常运行，此时，无法通过配置修复问题，只能重刷固件才能恢复，请谨慎使用此功能。**



The screenshot shows the 'Task' configuration tab. At the top, there are tabs for '基本参数', '串口参数', '网络通道参数', '预置信息', '自动采集任务', 'GPIO', '数据流', and '任务'. The '任务' tab is selected. Below the tabs, there are radio buttons for '启用' (selected) and '不启用'. Underneath, there is a section for '任务1' with a Lua script editor containing the following code:

```
function
while true do
  sys.wait(30000)
  uart.write(1, "Hello World")
end
end
```

At the bottom of the task configuration area, there is a blue button labeled '+ 添加任务'.

## 四、配置保存

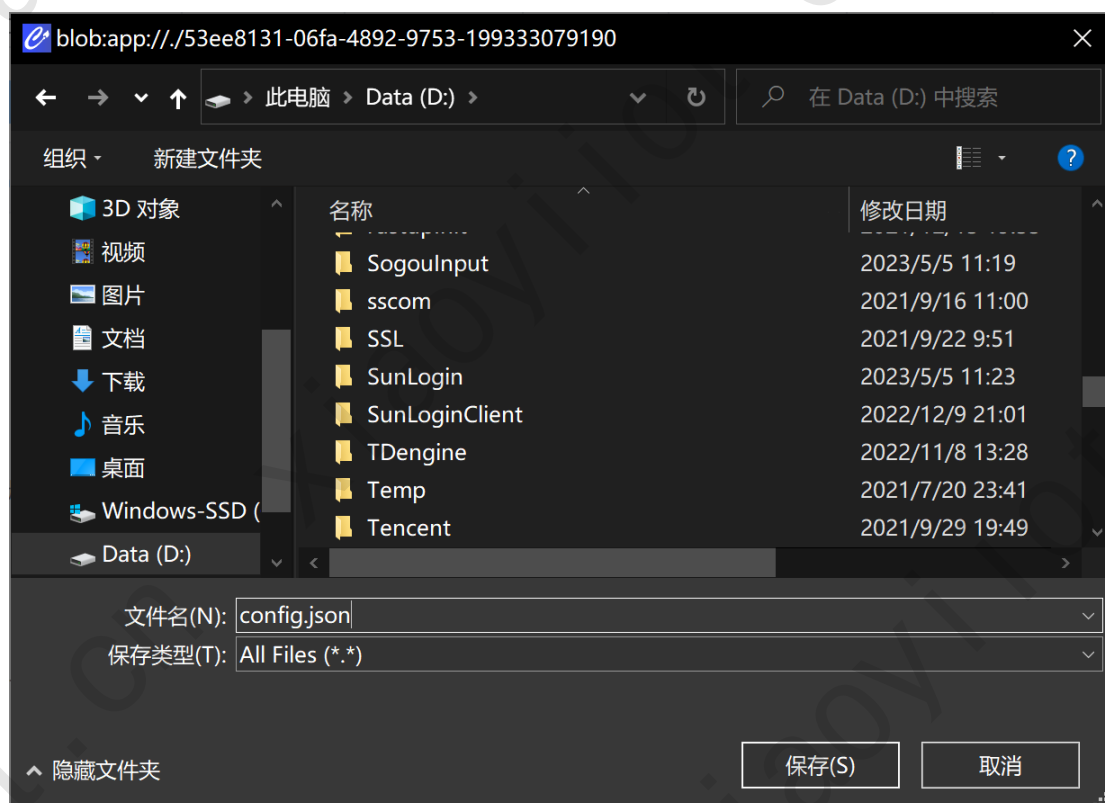
用户进行了自定义配置之后，可以将配置结果保存到文件，以便后续进行配置的时候不用将配置参数重新手动输入一遍。

操作步骤：

(1) 用户在页面中配置完成之后，点击“导出 JSON 按钮”。(导出的 JSON 文件中不包含 modbus 配置)



(2) 保存生成的 JSON 文件。



(3) 需要使用此配置的时候, 点击“导入 JSON 文件”。将前面保存的 JSON 文件导入进来, 可将文件中的配置参数解析到页面中。



注意：“导入 JSON 文件”功能只是将 JSON 文件中的配置参数解析到页面中，并非直接写入到 DTU 中，因此还需要执行“写入配置”才会将配置写入到设备中去。此外，modbus 配置需要在该配置页面中单独进行导入导出操作，如下。



感谢阅读！