目录

前 言 3

版权声明 3

版本信息 3

一、YX-RTU自动采集脚本简介 4

二、脚本详解 4

2.1 脚本格式 4

2.2 脚本指令详解 4

2.2.1@C串口输出命令 5

2.2.2 @PRINT 执行打印指令（字符格式） 6

2.2.3 @PRTLN 执行打印指令并换行（字符格式） 6

2.2.4 @D 延时控制指令 7

2.2.5 @DO 继电器控制指令 7

2.2.6 @SET 变量赋值指令，给I1-I10变量赋值 8

2.2.7 @INC 变量值增加 8

2.2.8 @DEC 变量值减小 8

2.2.9 @MUL 变量值乘 9

2.2.10 @DIV 变量值除 9

2.2.11 @IF 条件判断 10

2.2.12 @FOR 循环控制指令 12

前 言

本手册主要介绍有线 RTU的本地脚本编程指令。

适用型号： YX-DIDO-XX V1. 9 . 5及以上版本

版本信息

文档名称： YX-RTU脚本编程手册

版本： 1. 0

修改日期： 2021 年11 月 7 日

 一、RTU自动采集脚本简介

在很多应用中，用户需要灵活简便地自定义本地控制逻辑，需要根据DI，DO以及串口数据命令进行响应动作， RTU 针对这类应用开发了一种脚本语言，用户可以使用脚本语言自定义用户设备的控制逻辑。

 RTU 脚本支持用户自定义控制逻辑，包括DI状态采集、DI状态触发、DO继电器控制以及主动发送数据给上位机，用户也可以自定义串口指令来控制DO继电器动作。 用户只需确定现场控制逻辑，然后通过编写脚本指令即可让 RTU 按照用户的控制逻辑自动运行、以及和上位机数据通信。脚本实现了基本的开关控制、数字量及继电器状态上报、DI变化触发控制、延时控制、指令下发控制、自动循环控制、主动上报控制、自定义控制指令等功能，能满足大部分现场开关控制需求。脚本自动周期性执行，

也可以定义为只执行一次。

此文档定义的脚本指令主要是为了实现 RTU自动控制功能，RTU本身支持标准MODBUS RTU协议，上位机可以通过串口采用标准MODBUS协议进行采集和控制，在某种情况下，用户可能需要上位机通过串口发送非标准的指令进行RTU控制，YRTU 也支持自定义串口控制报文，并指定对应的动作，上位机发送自定义的串口控制报文给 RTU ，RTU 收到后会执行对应动作。

RTU脚本还支持变量及基础的变量运算，以及对变量进行判断。脚本执行和程序一样，从开始顺序执行，有循环指令或判断指令时才会改变执行顺序，直到脚本执行完毕，然后再次循环执行。

默认情况下，RTU支持两套脚本并行运行，可实现更加灵活的功能。（如有需要，还可以联系我们定制更多脚本任务并行执行。)

二、脚本详解

2.1 脚本格式

@cmd=value

@：脚本头，每一条脚本指令都使用@开始。

cmd：指令，为 1 或多个字符、数字组成的字符串，不区分大小写，下面详解每一条脚本指令的含义。 value：指令动作，指定指令需要执行的值。

2.2 脚本指令详解

2.2. 1 @C 执行命令指令（HEX 格式）

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @C | 执行命令 | @C=HEX @C=HEX+V1 @C-HEX+I1 | 定义执行命令，等效于中心下发数据，命令输入采用HEX 格式 |

C： 执行命令，@C 定义的命令 RTU 向串口上发送指定的内容

示例：@C=010203

 向串口发送010203三个字节长度的包

当命令为MODBUS RTU 控制协议时，可以在命令最后使用 V1 让 RTU 自动计算CRC校验并跟在命令的结尾，省去了用户自己计算校验的麻烦。此脚本指令定义的采集命令内容为可见的 HEX 字符。

示例： Modbus 采集指令： 01 03 00 00 00 01 84 0A

脚本表示为： @C=010300000001V1

 报文里面嵌入变量

示例：@C=010203I1040506

如果I1=1，则会向串口发送01020301040506

如果I1=0，则会向串口发送01020300040506

2.2.2 @PRINT 执行打印指令（字符格式）

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @PRINT | 执行命令 | @PRINT=STR@PRINT=Ix | 定义执行命令，往串口发送打印内容，字符格式，功能类似于 @C ，但不支持自动校验。字符内容不能出现@字符 |

PRINT： 执行命令，@PRINT定义的命令 RTU 功能和@C 相似，但@C 命令输入的是 HEX 格式命令，@PRINT输入的命令采用文本格式，适合于字符通讯协议的应用，免去用户转换成 HEX 格式命令的麻烦，也可以用于打印变量I1-I10的内容。

示例：

1. @PRINT=HELLO ,向串口发送HELLO字符串
2. @PRINT=I1，向串口发送变量I1的值。

2.2.3 @PRTLN 执行打印指令并换行（字符格式）

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @PRTLN  | 执行命令 | @PRTLN =STR | 定义执行命令，往串口发送打印内容，字符格式，功能类似于 @C ，但不支持自动校验。字符内容不能出现@字符 |

PRTLN ： 执行命令，@PRINT定义的命令 RTU 功能和@C 相似，但@C 命令输入的是 HEX 格式命令，@PRINT输入的命令采用文本格式，适合于字符通讯协议的应用，免去用户转换成 HEX 格式命令的麻烦，也可以用于打印变量I1-I10的内容。

示例：

1. @PRTLN =HELLO ,向串口发送HELLO字符串以及换行符
2. @PRTLN =I1，向串口发送变量I1的值以及换行符。

2.2.4 @D 延时控制指令

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @D | 延时控制 | @D=X | 在此处 RTU 会等待指令所定义的延时，然后再继续执行 X 表示数值支持的单位：秒 |

D：延时控制，此脚本指令用于控制延时，X为数字的格式，单位为秒 ， 支持小数。RTU遇到此指令后等待相应的延时值再继续执行脚本。最小分辨率为0.001（1毫秒）

示例： @D=10 (等待 10 秒)

@D=0.2 (等待0.2 秒)
 @D=60 (等待60秒)

2.2.5 @DO 继电器控制指令

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @DO | DO控制 | @DOX=1,0,~@DOX=DIX,!DIX@DOX=DOX,!DOX@DOX=(I1<5) | 控制继电器动作 |

示例：

@DO1=1 , 继电器1打开

@DO1=0 , 继电器1关闭

@DO1=~ , 继电器1状态翻转

@DO1=DI1 , 继电器1状态与DI1同步

@DO1=!DI1 , 继电器1状态与DI1相反

@DO2=DO1 , 继电器2状态与继电器1状态同步

@DO2=!DO1 , 继电器2状态与继电器1状态相反

@DO1=(I1<5) 当变量I1小于5时，继电器1导通，否则断开

2.2.6 @SET 变量赋值指令，给I1-I10变量赋值

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @SET  | SET  | @SET =Ix,y | 给把y赋值给变量Ix |

示例：

@SET=I1,1 I1赋值为1

@SET=I1,2 I1赋值为2

说明，变量为I1-I10，整型。

2.2.7 @INC 变量值增加

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @INC | 变量值增加 | @INC=Ix,[y] | 变量Ix的值增加1或者y |

示例：

@INC=I1 I1值加1

@INC=I1,10 I1值加10

2.2.8 @DEC 变量值减小

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @DEC  | 变量值减小 | @DEC=Ix,[y] | 变量Ix的值减小1或者y |

示例：

@DEC=I1 I1值减1
@DEC=I1,10 I1值减10

2.2.9 @MUL 变量值乘

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @MUL  | 变量值乘 | @MUL =Ix,[y] | 变量Ix的值乘以y |

示例：@MUL =I1,2 I1=I1\*2

2.2.10 @DIV 变量值除

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 |
| @DIV  | 变量值除 | @DIV=Ix,[y] | 变量Ix的值除以y |

示例：@DIV =I1,2 I1=I1/2

2.2.11 @IF 条件判断

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 | 默认值 |
| @IF | 条件判断 | @IF @ENDIF@IF@ELSE @ENDIF | 根据条件，判断是否执行代码条件包括：1,0,STARTDI1H,DI1L;DI2H,DI2L,...DIxH,DIxLDO1H,DO1L;DO2H,DO2L,...DOxH,DOxLDI1~,DI2~;...DIx~!DI1~,!DI2~;...!DIx~(RD=NULL);(RD<>NULL)(RD=.....)(I1==X)...(I10==X)或(I10==X)(I1<X)或(I1<=X)...(I10<X)或(I10<=X)(I1>X)或(I1>=X)...(I10>X)或(I10>=X)(I2<X)或(I2<=X)...(I10<X)或(I10<=X)(I2>X)或(I2>=X)...(I10>X)或(I10>=X)(I1<>X);(I2<>X)...(I10<>X)(I1==I2)或(I1=I2)...(Ix==Iy)(I1<I2)或(I1<=I2)...(Ix<Iy)或(Ix<=Iy)(I1>I2)或(I1>=I2)...(Ix>Iy)或(Ix>=Iy)(I1<>I2)...(Ix<>Iy) |  |

示例：

@IF=DI1H@DO1=1@ENDIF@D=0.1 当DI1导通时，DO1导通

@IF=DI1H@DO1=1@ELSE@DO1=0@ENDIF@D=0.1 当DI1导通时，DO1导通,否则DO1断开 （等效于@DO1=DI1）

@IF=(I1<5)@DO1=1@ELSE@DO1=0@ENDIF@D=0.1 当变量I1小于5时，DO1导通，否则断开

@IF=START@DO1=0@DO2=1@ENDIF@D=0.1 当程序启动时，继电器1断开，继电器2导通

说明：START表示程序刚启动运行，一般用于初始化操作。

@IF=(RD=010101)@DO1=1@ENDIF@IF=(RD=00000)@DO1=0@ENDIF@D=0.1 当串口收到010101时，DO1导通，当串口收到000000时，DO1断开

@IF=(RD=NULL)@DO1=1@ENDIF@D=0.1 当串口未收到任何数据包，DO1导通

@IF=(RD<>NULL)@DO1=1@ENDIF@D=0.1 当串口收到任意数据包，DO1导通

@IF=DI1~@DO1=1@ENDIF@D=0.1 当DI1状态发生了变化时，DO1导通

（注：IF可以嵌套使用，最多10层）

例如：

@IF=DI1H@IF=DI2H@DO1=1@ELSE@DO1=0@ENDIF@ELSE@DO1=0@ENDIF
当DI1和DI2都导通时，DO1导通,否则DO1断开

2.2.12 @FOR 循环控制

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | 名称 | 格式 | 含义 | 默认值 |
| @FOR | 条件判断 | @FOR=(条件) @ENDFOR | 根据条件，判断是否执行代码条件包括：1,0,STARTDI1H,DI1L;DI2H,DI2L,...DIxH,DIxLDO1H,DO1L;DO2H,DO2L,...DOxH,DOxLDI1~,DI2~;...DIx~!DI1~,!DI2~;...!DIx~(RD=NULL);(RD<>NULL)(RD=.....)(I1==X)...(I10==X)或(I10==X)(I1<X)或(I1<=X)...(I10<X)或(I10<=X)(I1>X)或(I1>=X)...(I10>X)或(I10>=X)(I2<X)或(I2<=X)...(I10<X)或(I10<=X)(I2>X)或(I2>=X)...(I10>X)或(I10>=X)(I1<>X);(I2<>X)...(I10<>X)(I1==I2)或(I1=I2)...(Ix==Iy)(I1<I2)或(I1<=I2)...(Ix<Iy)或(Ix<=Iy)(I1>I2)或(I1>=I2)...(Ix>Iy)或(Ix>=Iy)(I1<>I2)...(Ix<>Iy) |  |

示例：

@SET=I1,0@FOR=(I1<10)@INC=I1@DO1=~@D=0.1@ENDIFOR

循环执行10次： DO1状态翻转，然后延迟0.1秒

（注：FOR 可以嵌套使用，最多10层。）

例如：

@SET=I1,0@FOR=(I1<10)@INC=I1@SET=I2,0@FOR=(I2<10)@INC=I2@DO1=~@D=0.5@ENDFOR@ENDFOR

效果：DO1翻转，然后延迟0.5秒。 总共执行100次。