Can I write my own protocol or third-party protocol to apply on ISaGRAF controllers?

ICP DAS ISaGRAF controllers support below Serial COM Port (RS-232/422/485) protocols:

Modbus RTU Slave	Refer to Chapter 4 of the ISaGRAF user's manual & respective getting started manual
I-7000 and I-87xxx RS-485 I/O	Refer to Chapter 6 of the ISaGRAF user's manual
Modbus RTU Master (M-7000)	Refer to Chapter 8 and 21 of the ISaGRAF user's manual
Modbus ASCII Master	Refer to Chapter 8 of the ISaGRAF user's manual
Modem Link	Refer to Chapter 13 of the ISaGRAF user's manual
MMICON	Refer to Chapter 16 of the ISaGRAF user's manual
SMS : Short Message Service	Refer to Chapter 17 of the ISaGRAF user's manual

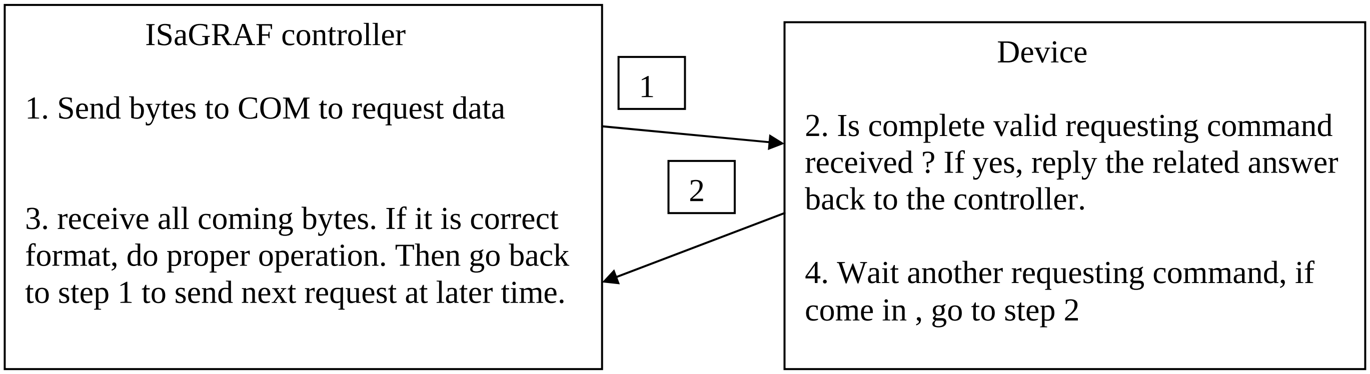
User can appy below COM functions to operate other protocols or 3rd party protocols. (Please refer to Appendix A.4 of the ISaGRAF user's manual for description of these COM functions)

COMOPEN	Open Serial COM Port (without "Flow control" parameter)
COMOPEN2	Open Serial COM Port (with "Flow control" parameter, not for I-8xx7)
COMREADY	Test if any byte come in
COMARY_R	Read all bytes which already come in to a byte array
COMARY_W	Write many bytes in a byte array to COM Port
COMREAD	Read one bytes (Please call "COMREADY" to test first, if there is data, then "COMREAD" can be called)
COMCLEAR	Clear all received bytes in the receiving buffer
COMARY_NW	Write one signed long Integer to COM Port, format is Binary, 4-byte
COMARY_WW	Write one signed Word to COM Port, format is Binary, 2-byte
COMSTR_W	Write one string to COM Port
COMWRITE	Write one byte to COM Port
COMCLOSE	Close Serial COM Port

Note:
1. The default shipping of I-8xx7 controller has set its COM1 and COM2 (COM2:RS-485 is only for I-8417/8817) as Modbus RTU Slave Port. User can choose to switch off the COM1:Modbus RTU Slave function to become a freely used COM port by the above listed COM functions. (Please refer to Appendix C.1). To use i-8xx7 's COM5 to COM20, Please refer to Chapter 1.8 to install I-8112/8114/8142/8144 serial expansion boards.
2. Wincon-8xx7 / 8xx6 ‘s COM2 / COM3 can be switched ON as a Modbus RTU Slave Port. Or Switch Off for freely used. (Refer to Appendix A.2 of its Getting Started manual delivered with the hardware). To use Wincon ‘s COM5 to COM14 at I-8112/8114/8142/8144 serial expansion boards, please refer to Appendix E of the "Getting Started:Wincon ISaGRAF PAC" manual.
3. COM1 of I-7188EG / 7186EG is set as Modbus RTU Slave port when shipping. User may switch it OFF to freely use it by COM port functions. (Please refer to its "Getting Started Manual" delivered with its hardware). However I-7188XG ‘s COM1 can not be switch OFF, it is always Modbus RTU Slave port. If user want to use COM3 to COM8 of I-7188EG/XG and 7186EG, please plug one extra X-5xx expansion I/O board inside it .
http://www.icpdas.com/products/PAC/i-o_expansion/x_list.htm

24.1: Cntroller send 1 request and get 1 reply from device

Below figure lists the most common RS-232 / 422 / 485 application.



ISaGRAF controller

1. Send bytes to COM to request data

3. receive all coming bytes. If it is correct format, do proper operation. Then go back to step 1 to send next request at later time.

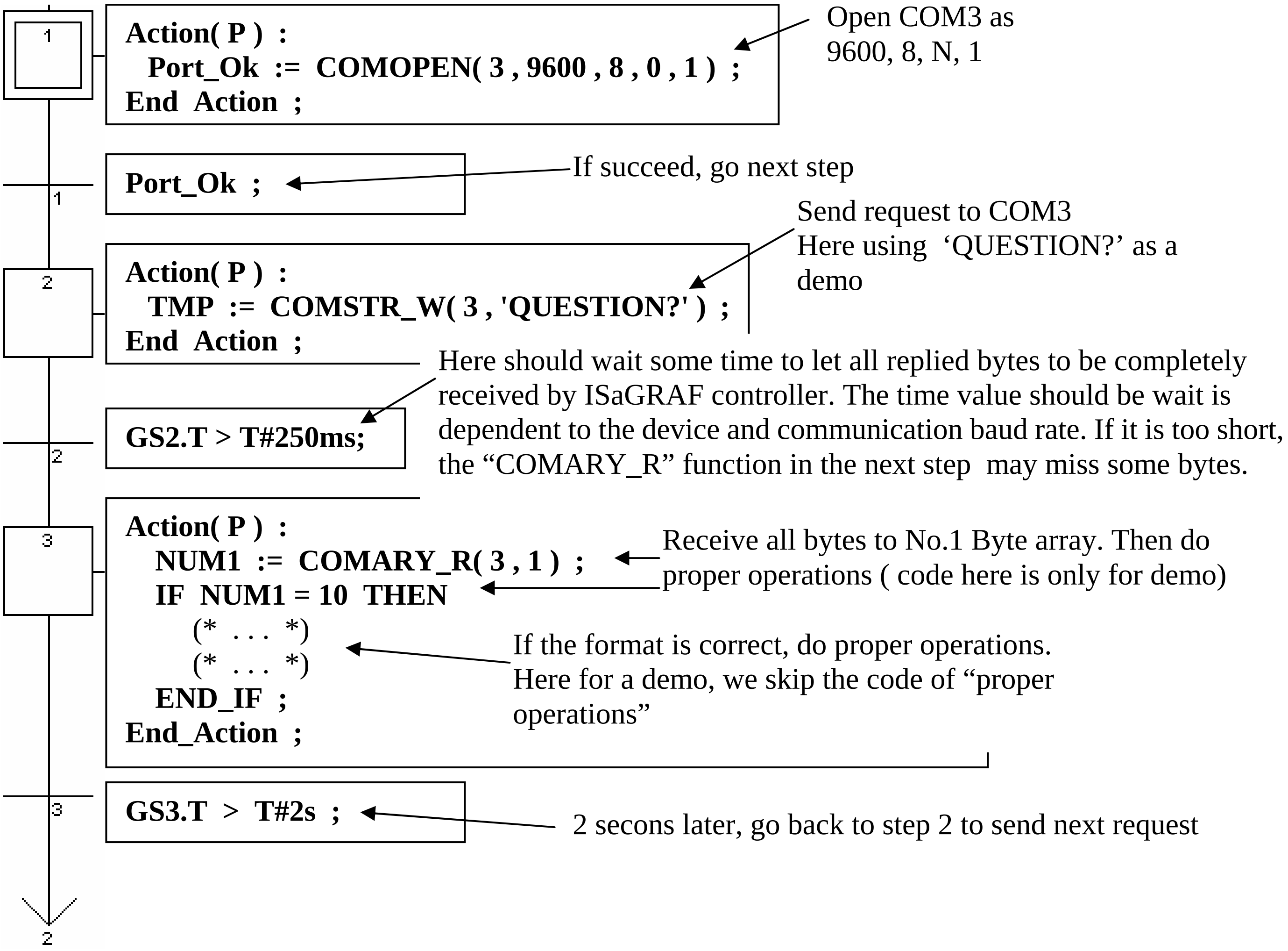
Device

2. Is complete valid requesting command received ? If yes, reply the related answer back to the controller.

4. Wait another requesting command, if come in , go to step 2

User can use the below code or similiar code to do it.
Below example will send a string "QUESTION?" to device via COM3 , then waiting device to reply the related answer. And then 2 seconds later, send next same question to device , …

SFC program: ("Port_OK" & "TMP" is Boolean Internal , "NUM1" is Integer Internal)



```
Action( P ) :
  Port_Ok := COMOPEN( 3 , 9600 , 8 , 0 , 1 ) ;
End Action ;
```

Open COM3 as 9600, 8, N, 1

```
Port_Ok ;
```

If succeed, go next step

```
Action( P ) :
  TMP := COMSTR_W( 3 , 'QUESTION?' ) ;
End Action ;
```

Send request to COM3 Here using 'QUESTION?' as a demo

```
GS2.T > T#250ms;
```

Here should wait some time to let all replied bytes to be completely received by ISaGRAF controller. The time value should be wait is dependent to the device and communication baud rate. If it is too short, the "COMARY_R" function in the next step may miss some bytes.

```
Action( P ) :
   NUM1 := COMARY_R( 3 , 1 ) ;
   IF NUM1 = 10 THEN
       (* . . . *)
       (* . . . *)
   END_IF ;
End_Action ;
```

Receive all bytes to No.1 Byte array. Then do proper operations (code here is only for demo)

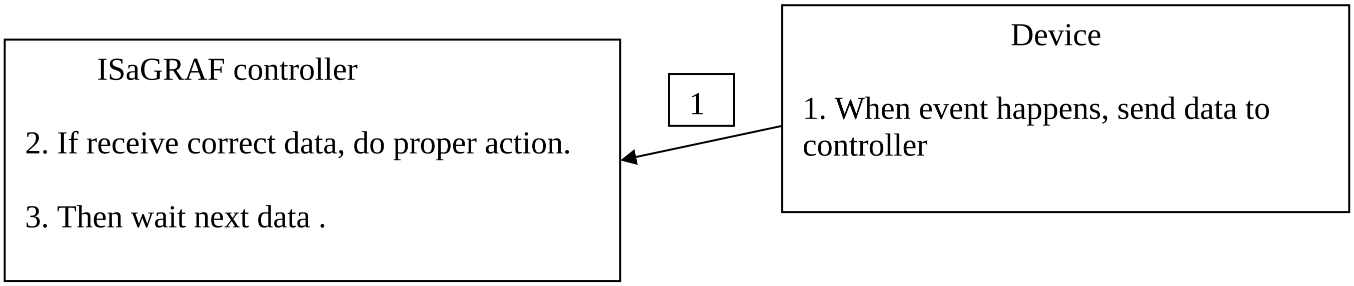
If the format is correct, do proper operations. Here for a demo, we skip the code of "proper operations"

```
GS3.T > T#2s ;
```

2 secons later, go back to step 2 to send next request

24.2: Controller just wait data from the remote device

This kind of application is very common in the store. Like the device of “Bar code reader”, when it reads bar code on the product, it will send the related data to the controller via RS-232 / 422 / 485. The controller just receive it, not necessary to send any byte to device. (Please visit www.icpdas.com – FAQ – Software_ISaGRAF for demo program and more information.)



ISaGRAF controller

2. If receive correct data, do proper action.

3. Then wait next data .

1

Device

1. When event happens, send data to controller

ST program:

```
IF  INIT  THEN
   INIT  :=  FALSE ;
   Port_Ok  :=  COMOPEN( 3 , 9600 , 8 , 0 , 1 ) ;
   T1  :=  T#0s ;
   STEP  :=  0 ;
END_IF  ;
```

Variable declaration
“INIT” is Boolean Internal, init as TRUE
“Port_Ok” & “TMP” is Boolean Internal
“T1” is Timer Internal
“STEP” & “NUM1” is Integer Internal

Open COM3 as 9600, 8, N,1

```
IF  Port_Ok = False  THEN
   Return ;
END_IF ;
```

If open fail, just quit this ST program

```
CASE  STEP  OF

 0 :  IF  COMREADY(3)  THEN
          STEP := 1 ;
          T1 := T#0s ;
          Tstart(T1) ;
      END_IF ;
```

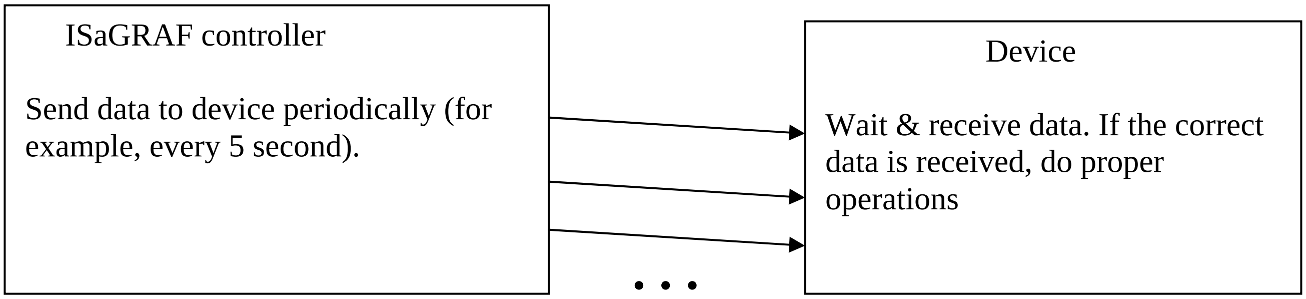
STEP=0 means “waiting state”, it should check if any byte come in. If “COMREADY” returns TRUE, it means there is at least one byte come in. Then set “STEP” to 1 and start timer “T1” to tick.

```
 1 :  IF  T1 > T#250ms  THEN
          Tstop(T1) ;
          T1 := T#0s ;
          STEP := 0 ;
          NUM1 := COMARY_R(3, 1) ;

          IF  NUM1=10  THEN
              (* … *)
          END_IF ;
     END_IF ;

END_CASE ;
```

STEP=1 means “data is coming”, when “T1” tick to the specified waiting time, call “COMARY_R” to receive all bytes to No.1 byte array (this will make sure all bytes from the “device” all well received). This waiting time is dependent with the “Device” and the communication baud rate. Here we use 0.25 second as a demo. (If setting too short than your device should have, some bytes may be missing) Remember to set “STEP” to 0 to wait next data.

Do proper operation if receive correct data.
The code listed here is only for demo. (It is depend with your application, we just assume the correct data has 10 bytes here.)

24.3: Report data to remote device periodically



ISaGRAF controller

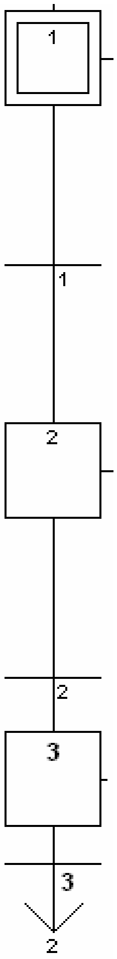
Send data to device periodically (for example, every 5 second).

Device

Wait & receive data. If the correct data is received, do proper operations

• • •

SFC program: (Please declare “TMP” & “Port_Ok” as Boolean Internal)



```
Action( P ) :
  Port_Ok := COMOPEN( 3, 9600, 8, 0, 1 ) ;
End_Action ;
```

Open COM3 as 9600, 8, N, 1

```
Port_Ok ;
```

If succeed, go next step

```
Action( P ) :

   (* Prepare all bytes in No.1 byte array *)
   (* … *)

   TMP := COMARY_W( 3 , 1 , 20 ) ;
End_Action ;
```

The data to be sent depends on your application. Please prepare them and call COMARY_W or COMSTR_W to send it

```
GS2.T > T#5s ;
```

5 seconds later, first jump to an empty step3, then go to step 2 to send next data

(SFC program will operate error if jumping to the same step after it is finished. So we should jump to an empty step first. The ISaGRAF compiler generate no erro if not jumping to an empty step first. But its result will be error.)

Here using “COMARY_W” to send 20 bytes in No.1 byte array to COM3. Or “COMSTR_W” may be used for sending string (string length is max. 255 bytes) , there should be no 0 value in the string. (because 0 is string-end byte, however character ‘0’ is Ok in the string).

24.4: Controller send data when event happens

Please refer to Chapter 11.3.5