# How to connect to remote server and send network package via GPRS with µPAC-5000 series?

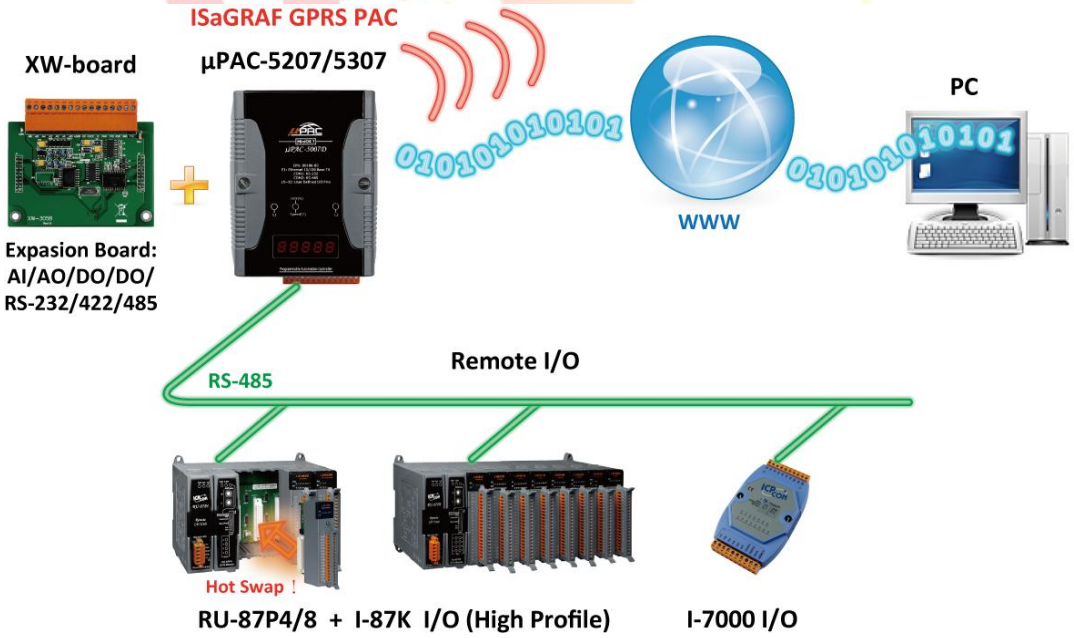
- **Application introduction** :

Program an ISaGRAF project using a built-in GSM controller to send data from the battery backup SRAM via GPRS to a remote server PC and show the data on the PC.

**Note** : **In this demo case, the PC must have a public IP.**

The following versions of the ISaGRAF GSM PACs support to send the network package to the remote server via GPRS.

µPAC-5207: since the first released version
µPAC-5307: since the first released version




**Get this paper and demo programs :**
http://www.icpdas.com/faq/isagraf.htm > FAQ-137

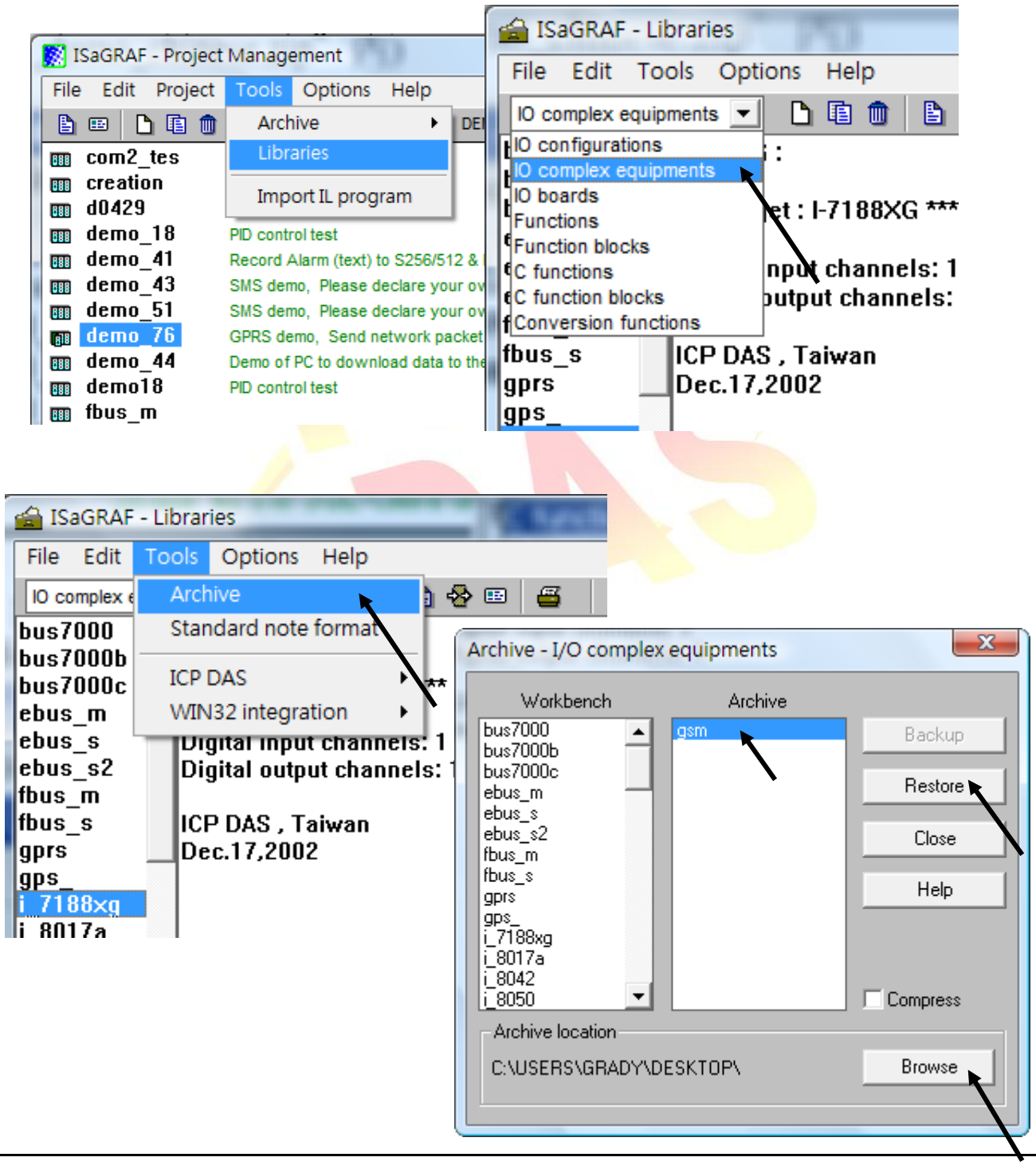
**Get the new ISaGRAF drivers :**
http://www.icpdas.com/products/PAC/i-8000/isagraf-link.htm

**Get Data Sheet :**
http://www.icpdas.com/products/PAC/i-8000/data%20sheet/data%20sheet.htm

## 1.1 Restore the "GSM" and demo programs to the PC / ISaGRAF

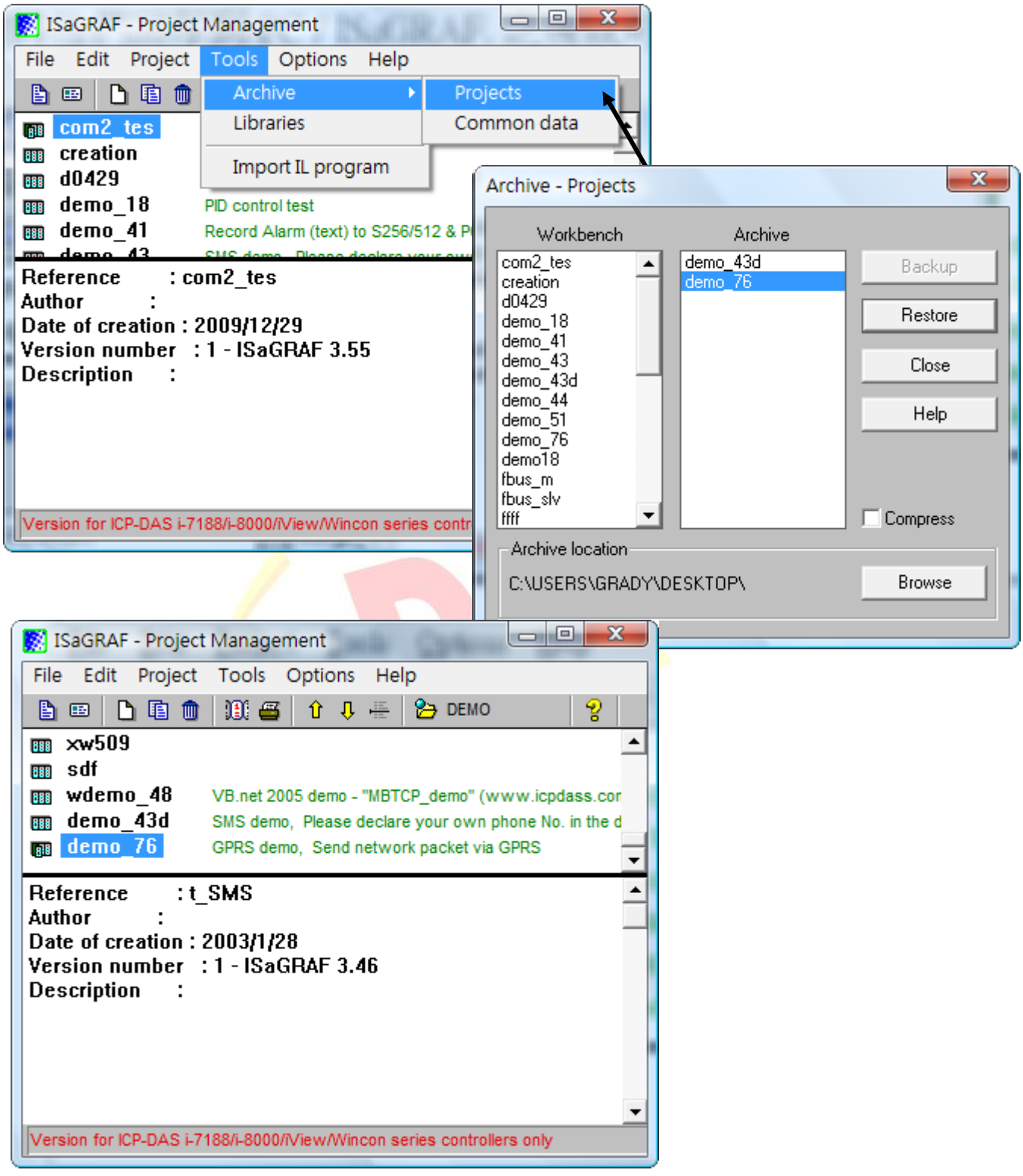
- **Restore the "GSM.xia" :**

To write a program to communicate with the remote server, first restore the ISaGRAF I/O complex equipments "GSM.xia" to the PC/ISaGRAF.
The "GSM.xia" is inside the "faq137_demo.zip" which is downloaded from http://www.icpdas.com/faq/isagraf.htm > FAQ-137.



- **Restore the demo programs** :

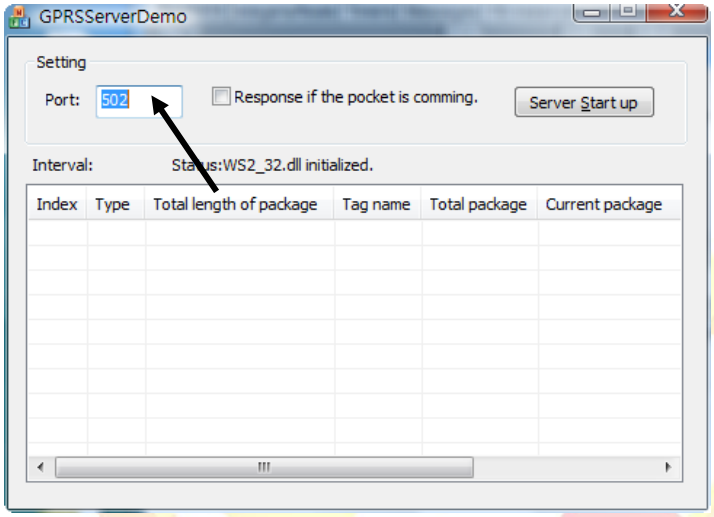
The "faq137_demo.zip" contains the demo project "Demo_76.pia". Please restore it to the PC/ISaGRAF. Then user can refer to it easily.



### 1.2 How to test this demo:

1. This demo contains an ISaGRAF project and a program running on the PC.

2. Execute the demo program "GPRSServerDemo" in the PC. Set up the server port number and click on the "Server start up" button, then this program will show the received network package.




**Note** : **The PC running the program "GPRSServerDemo" must have a public IP, or the µPAC-5000 PAC cannot connect to the PC via GPRS.**

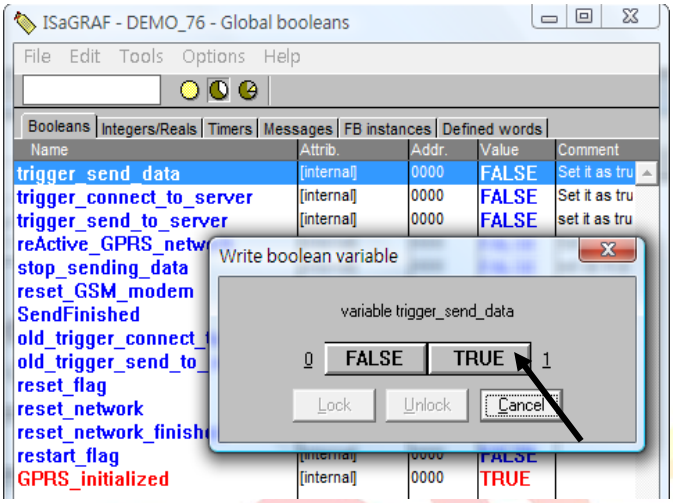
3. Please refer to the section 1.1 to restore "GSM.xia" and ISaGRAF demo project to PC/ISaGRAF.

4. Plug the SIM card in the µPAC-5207 or µPAC-5307 controller.

**Note** : **This demo will use the GPRS function, please contact your TEL network provider to obtain the GPRS or 3G service.**

5. Compile the ISaGRAF project, and download it to µPAC-5207 or µPAC-5307 controller.

6. Set the variable "trigger_send_data" in the dictionary to "true", and wait the built-in GSM module to initialize, then the PAC will send the data from battery backup SRAM to the remote server one by one via GPRS service.




The execution file, "GPRSServerDemo.exe", and the source code can be downloaded from www.icpdas.com > FAQ > Software > ISaGRAF Ver.3(English) > 137

If you want to modify the source code, please make sure the Virtual Studio .net 2008 or 2005 has been installed in your PC. Then double click on the file "GPRSServerDemo.sln" in the folder "GPRSServerDemo" to open it.

## 1.3 GSM I/O board :

There are 5 sub-boards in the GSM I/O board:

1. GSM_set
2. NET_reg
3. IP
4. Sock_set
5. Sock_sts

**Description of the sub-boards:**

**GSM_set: Initialize the connecting GSM module and get the current state**

**Parameters** :

Com_port:
Open the COM port which is connecting with the GSM module.
Usage:
0: Open COM0 (If the PAC is built-in GSM, please open COM0.)
1: Open COM1

PIN_code:
If the SIM card is locked, set the PIN code. If it is not locked, please set as blank.
Usage:
If the PIN code is 0000, please enter 0000

**I/O:**

Ch1: Error code. If there's error for operating the GSM module, it returns an error code.
The error code is as following.
1 : Operation success
2 : Operation failed
3 : No response from the GSM module
4 : SIM card is not inserted
5 : SIM card is locked
6 : Require the PIN code to unlock the SIM card
7 : Incorrect PIN code

Ch2: Signal strength
2 ~ 30: Normal signal strength
99: No signal

Ch3: The state of the SIM card
0: SIM card is not inserted or required the PIN code to unlock.
1: SIM card is inserted and unlocked.

Ch4: The state of the GSM module
0: GSM module is not initialized
1: GSM module is initialized.

**NET_reg** : **Set GPRS/DNS, and get the error code of the GPRS service.**

**Parameter:**

APN_1, APN_2 :

APN: Access Point Name. Please contact your SIM card provider to obtain the APN.
If the APN is less than 15 characters, enter it as following.
Usage: For example, the APN is "internet"
APN_1: internet
APN_2: blank, don't enter any character.

If the APN is more than 15 characters, enter it as following.
Usage: For example, the APN is "internet.mnc012.mcc345.gprs"
APN_1: internet.mnc012 ←enter ONLY 15 characters
APN_2: .mcc345.gprs ←enter from the 16th character

usr_name : User name for log in
pass_word: Password for log in

If controller connects to remote server with the domain name, please enter the DNS:
DNS1_ip : Primary DNS IP
DNS2_ip : Secondary DNS IP

**I/O:**

Ch1 : Error code. It shows what happens during operating GPRS service.
1 : Operation success
9 : Invalid network setting
10 : Fail to register the network
11 : Fail to get IP address
12 : Invalid IP or domain name of the remote server
13 : Connect success
14 : Connect failed
15 : GPRS service not initialized
16 : Invalid socket type
17 : Connection is not established or disconnect
18 : Fail to send package
19 : the length of package is too long
20 : Success to disable GPRS service
21 : Fail to disable GPRS service
22 : Fail to set DNS
23 : Battery Backup SRAM is not found
24 : Internal communication timeout

Ch2 : The state of GPRS service
0: GPRS service is not initialized
1: GPRS service is initialized

**IP :**

**I/O:**

Ch1 : Get the public IP from network provider after the GPRS service initializing.

**Sock_set : the socket setting**

**Parameter:**

Sock_type:

0 : TCP socket

1 : UDP socket

header_tag: Valid only when Protocol_type is 1.

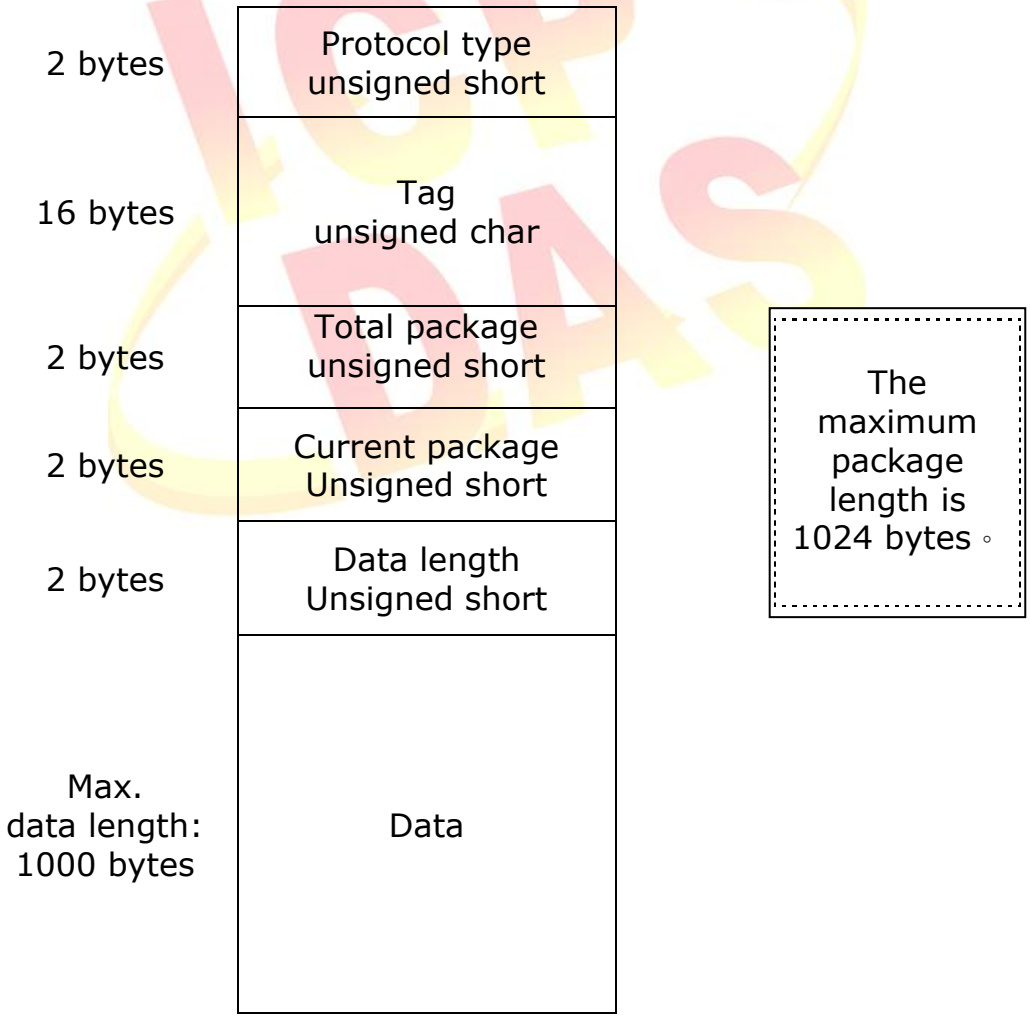
A string less than 15 characters

To identify the package which controller sent in the remote server

Protocol_type :

0 : Raw data

1 : built-in protocol It is defined as following:



ServerIP_Domain1, ServerIP_Domain2 : The IP or domain name of remote server

If you use the IP of remote server, please enter it as following

Usage : For example, the IP of remote server is 110.26.81.1

ServerIP_Domain1 : 110.26.81.1

ServerIP_Domain2 : Blank, don't enter anything

If you use the domain name of remote server, please enter it as following.

Usage : Ex, the domain name of remote server is crocodileci.dyndns.org

ServerIP_Domain1: crocodileci.dyn ←Only enter the first 15 characters

ServerIP_Domain2: dns.org ←enter form 16th character

**Note：If the connection is connected by Domain name, setting DNS is necessary.**

Remote_port: the port number of the remote server

**I/O:**

Ch1: GPRS service command

0 : do nothing

1 : connect to the remote server

2 : send network package to the remote server

3 : disconnect with the remote server

4 : reactive GPRS service

8 : reset GSM module

Ch2: The start address of Battery backup SRAM to send the network package.

The valid value is between 0 and 0x7D000

Ch3: the length of data to send

When Protocol_type is 0, the maximum is 1024

When Protocaol_type is 1, the maximum is 1000

Ch4: the address of battery backup SRAM that to store the received package from the remote server.

The valid value is between 0 and 0x7d000

CH5: the length of received package that you want to store

The maximum is 1024.

**Sock_sts :**

**I/O:**

Ch1: the state of the socket

0 : The socket does not connect to remote server

1 : The socket has connected to remote server

Ch2: the number of packages that want to send

Ch3: the number of packages that have sent

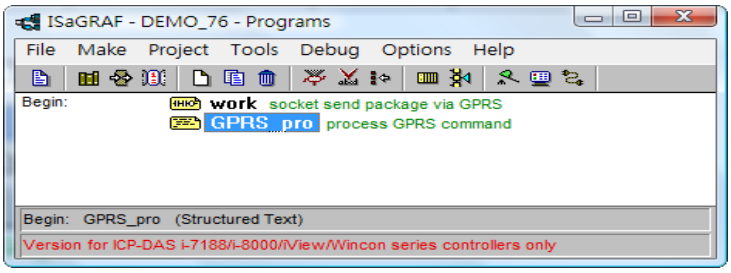
Ch4: the length of the sent package

Ch5: the length of the received package

## 1.4 Demo_73 – Send network package via GPRS service

- ISaGRAF Project Architecture :

  Include one ST program (GPRS_pro) and one LD program(work).



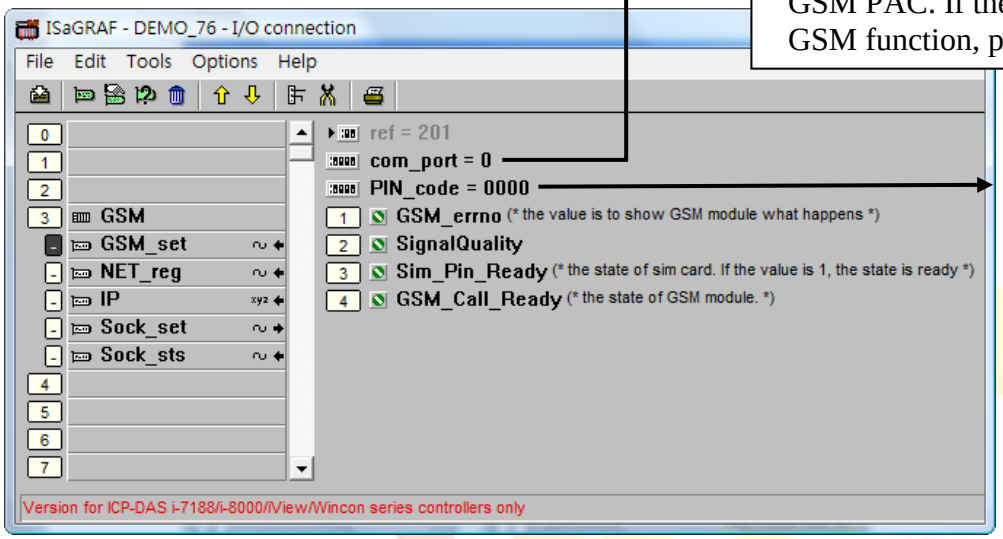

- ISaGRAF Variables :

| Name | Type | Attribute | Description |
|---|---|---|---|
| Step | Integer | Internal | Used to control the flow |
| GPRS_errno | Integer | Input | The error code of GPRS |
| GPRS_Network_Ready | Integer | Input | The state of GPRS. |
| Times | Integer | Internal | The number of packages that have sent during this connection |
| total_times | Integer | Internal | The number of all sent packages |
| Old_GPRS_Network_ready | Integer | Internal | Store the values of cycle, GPRS_Network_ready |
| temp_errno | Integer | Internal | Store the value of GSM_errno for observing |
| GSM_errno | Integer | Input | The error code of GSM module |
| SignalQuality | Integer | Input | Signal strength |
| Sim_Pin_Ready | Integer | Input | The state of SIM card |
| GSM_Call_Ready | Integer | Input | The state of GSM module |
| GPRS_Sock_Command | Integer | Output | GPRS socket command |
| Send_position_from_sram | Integer | Output | Give the SRAM address to send data to remote server |
| Send_data_length | Integer | Output | The length of data |
| Receive_position_to_sram | Integer | Output | Give the SRAM address to store the received package |
| GPRS_Socket_Connected | Integer | Input | The state of socket.<br>1: connect to remote server |

| Name | Type | Attribute | Description |
|---|---|---|---|
| total_send_package_num | Integer | Input | The total number of packages want to send |
| already_sent_package_num | Integer | Intput | The number of sent package |
| Sent_data_length | Integer | Intput | The length of sent package |
| Received_data_length | Integer | Internal | The length of received package |
| trigger_send_data | Boolean | Internal | If “True”, trigger to send package to remote server |
| trigger_connect_to_server | Boolean | Internal | If “true”, trigger to connect to remote server |
| reActive_GPRS_network | Boolean | Internal | Re-initialize the GPRS service |
| stop_sending_data | Boolean | Internal | Stop sending network package |
| reset_GSM_modem | Boolean | Internal | Reset the GSM module |
| SendFinished | Boolean | internal | The flag to recognize whether the package is sent |
| old_trigger_connect_to_server | Boolean | Internal | Store the state of trigger_connect_to_server |
| old_trigger_send_to_server | Boolean | Internal | Store the state of trigger_send_to_server |
| reset_flag | Boolean | Internal | |
| reset_network | Boolean | Internal | |
| reset_network_finished | Boolean | Internal | |
| resetart_flag | Boolean | Internal | |
| Local_IP | Message | internal | Get the IP address from network provider when GPRS service initialized |

- **IO Connection** :

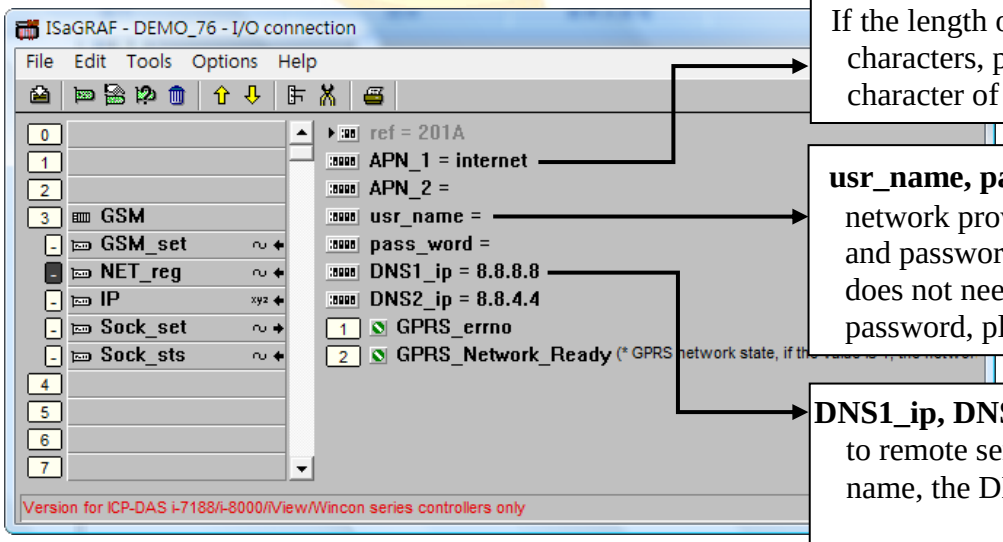
**GSM_set:**




Enter the COM port that connects with GSM PAC. If the PAC is built-in the GSM function, please enter 0.

If SIM card is locked, please enter the PIN code. If SIM card is unlocked, set this column as blank

**NET_set** :

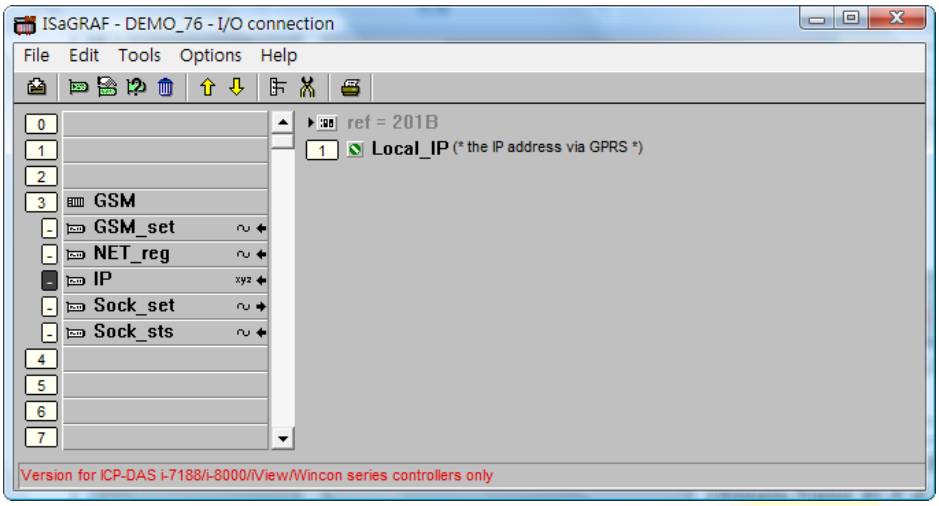



**APN_1, APN2:** Contact your network provider to get the APN,for example, fill in the APN "internet".
If the length of APN is more than 15 characters, please enter from the 16th character of APN in APN2.

**usr_name, pass_word**：Contact your network provider to get the user name and password for GPRS service. If it does not need user name and password, please set them as blank.

**DNS1_ip, DNS:_ip:** If the connection to remote server is using domain name, the DNS setting is necessary.

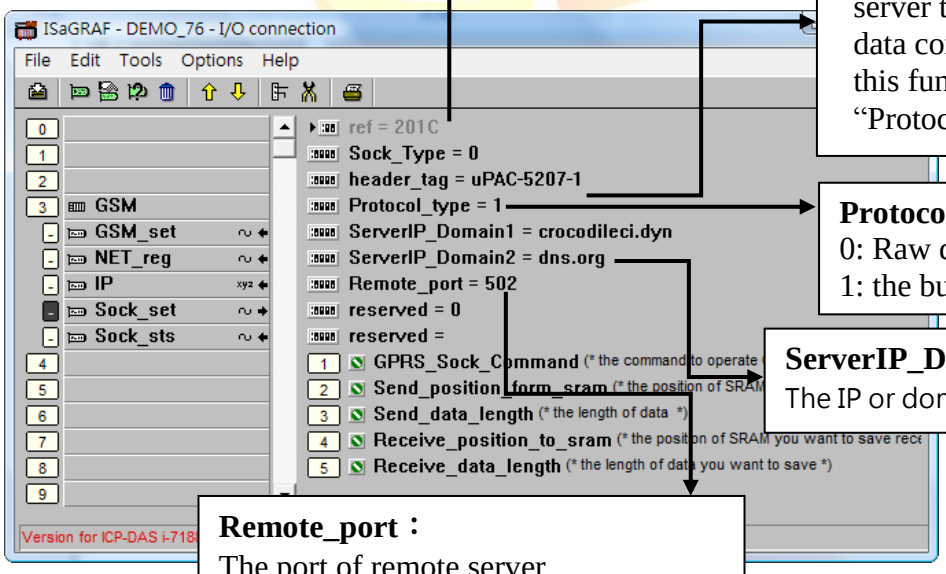
**IP**：




**Sock_set**：

**Sock_type**：
If it is set as 1, use UDP socket.
If it is set as 0, use TCP socket

**header_tag**：The maximum of length is 15 characters. It let the remote server to identify which controller the data come from. If you want to use this function, the variable "Protocol_type" must be set as 1.

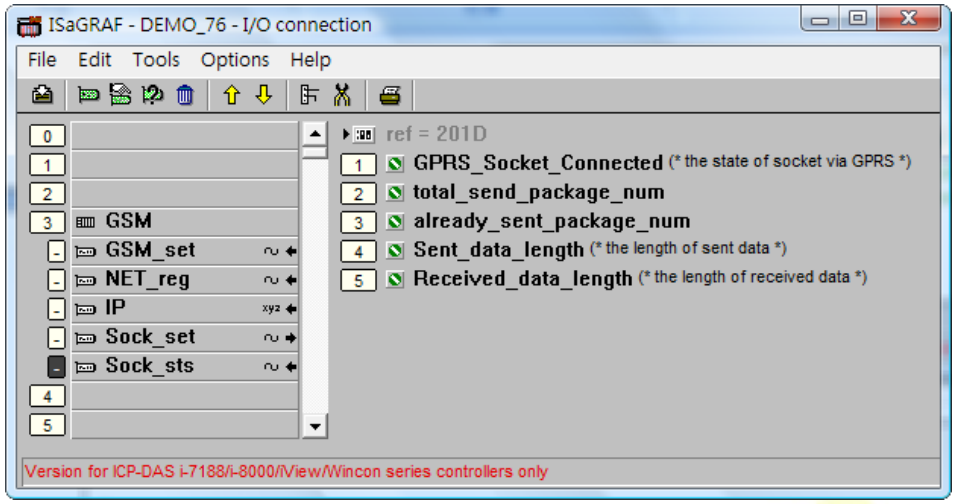



**Protocol_type:**
0: Raw data
1: the built-in protocol with header

**ServerIP_Domain1,ServerIP_Domain2:**
The IP or domain name of remote server

**Remote_port**：
The port of remote server

**Sock_sts:**




**Work Program Description** :

(* trigger to send data. If the value of trigger_send_data is true, it will set trigger_connect_to_server as true *)

[1] trigger_send_data trigger_connect_to_server

(* When the value of trigger_connect_to_server is true, set "trigger_send_data" as false.
set "trigger_send_dat" as true once *)

[2] trigger_connect_to_server trigger_send_data

**GPRS_pro program** :

```
(* if trigger_connect_to_server is true, GPRS module will connect to Remote Server *)
if trigger_connect_to_server then

case step of
   0:
      (* First step: If the socket is already connected, trigger to send *)
      (* package to server, or set GPRS_sock_command as 1 to *)
      (* connect remote server, and then go to the next step *)
      if GPRS_Network_Ready = 1 then
         if GPRS_Socket_Connected = 1 then
            trigger_connect_to_server := false;
            trigger_send_to_server := true;
         else
            GPRS_Sock_Command := 1;
            step := 1;
         end_if;
      else
         (* If GPRS service is not initialzed, set the variable “reactive_GPRS_network” as true *)
         reactive_GPRS_network := true;
         old_trigger_connect_to_server := trigger_connect_to_server;
         old_trigger_send_to_server := trigger_send_to_server;
         trigger_connect_to_server := false;
         trigger_send_to_server := false;
      end_if;
   1:
      (* Second step: Get the value of GPRS_errno to check the result of operation. *)
      (*                  GPRS_errno is always 0 until the operation is finished.          *)
      case GPRS_errno of
         0:
         1:
            (* The operation is success, go to next step *)
            step := 2;
        -15:
            (* GPRS network is not ready *)
            step := 0;
            trigger_connect_to_server := false;
      else
            (* If get the others set GPRS_Sock_Command as 1, try to connect again *)
            GPRS_Sock_Command := 1;
      end_case;
```

```
   2:
      (* The operation is finished and success. *)
      (* Set trigger_send_to_server as true to send package to remote server *)
      if GPRS_Socket_Connected = 1 then
         step := 0;
         GPRS_Sock_Command := 0;
         trigger_connect_to_server := false;
         trigger_send_to_server := true;
         times := 0;
      end_if;
end_case;

(* If the variable "trigger_send_to_server" is set as true *)
(* Do as following to send package *)
if trigger_send_to_server then

   case step of
      0:
         (* The first step: Check the socket is connected or not                    *)
         (*                 If it is not connected, set trigger_connect_to_server as true  *)
         (*                      to try to connect remote server.                   *)
         (*                      If it is connected, set GPRS_Sock_Command as 2 to send *)
         (*                      package to remote server. And go to next step      *)
if GPRS_Socket_Connected = 0 then
            trigger_connect_to_server := true;
            trigger_send_to_server := false;

            times := 0;

         else
            GPRS_Sock_Command := 2;
            step := 1;
         end_if;
      1:
         (* Second step: Get the value of GPRS_errno to check the result of operation. *)
         (*                 GPRS_errno is always 0 until the operation is finished.    *)
         case GPRS_errno of
            0:
            1:
               (* The operation is success, go to next step.                          *)
               (* Check if the data is valid, the valid value is between 0 to 0x7D000 *)
               (* Set the position as the next you want to send                       *)
              step := 2;
              if((Send_position_form_sram + Send_data_length) < 16#7D000) then
                 Send_position_form_sram := Send_position_form_sram + Send_data_length;
              else
```

```
                Send_position_form_sram := 0;
            end_if;
        -17:
            (* Connecttion is not established or disconnected. *)
            (* Go to next step. *)
            step := 2;
        else
            (* If get the others set trigger_connect_to_server as true, *)
            (* try to connect again                                      *)
            trigger_connect_to_server := true;
            trigger_send_to_server := false;
            step := 0;
        end_case;
    2:
        (* The operation is finished and success *)
        step := 0;
        trigger_send_to_server := false;
        (* Set the flag of SendFinished as true to identify that sending package is finished *)
        SendFinished := true;
  end_case;
end_if;

(* Sending package is finished *)
if SendFinished then
  SendFinished := false;

  (* plus one to the counter *)
  total_times := total_times + 1;
  times := times + 1;

  (* If the flag "stop_sending_data" is not true, send the next package *)
  if NOT(stop_sending_data) then
    trigger_send_data := true;
  end_if;
end_if;

(* If the falg "reset_GSM_modem" is true, the GSM module will reset. *)
if reset_GSM_modem then
  reset_gsm_modem := false;

  (* reset all the variable as following *)
  step := 0;
```

```
    (* record the current state of trigger_connect_to_server and trigger_send_to_server *)
    old_trigger_connect_to_server := trigger_connect_to_server;
    old_trigger_send_to_server := trigger_send_to_server;
    trigger_connect_to_server := false;
    trigger_send_to_server := false;

    (* set the variable "GPRS_Sock_Command" as 8 to reset the GSM module *)
    GPRS_Sock_Command := 8;

    (* Set the variable “”reset_flag as true to identify the GSM module is beginning to reset *)
    reset_flag := true;
end_if;

(* Wait the GSM module to reset                                    *)
(* When reset_flag is true and GPRS_Network_Ready is false,        *)
(* the GSM module is resetting.                                    *)
(* Set reset_network as true to identify the GSM module is resetting.  *)
if reset_flag and GPRS_Network_Ready = 0 then
    reset_flag := false;
    reset_network := true;
end_if;

(* Wait the resetting is finished                                  *)
(* If the flag "reset_network" is true and GPRS network is ready   *)
(* Reactive GPRS network service again.                            *)
if reset_network and GPRS_Network_Ready = 1 then
    reset_network := false;
    reactive_GPRS_network := true;
end_if;

(* If the state of GPRS_Network_Ready is from 0 to 1 and resetting_GPRS_network is true, *)
(* set restart_flag as true to identify the operation of the GSM module resetting is success.  *)
if Old_GPRS_Network_Ready = 0 and GPRS_Network_Ready = 1 then
    if resetting_GPRS_network then
        restart_flag := true;
    else
        GPRS_initialized := true;
    end_if;
end_if;
```

```
(* After resetting GSM module, restore the state of trigger_conect_to_server and *)
(* trigger_send_to_server to continue sending package *)
if reset_network_finished and restart_flag then
   reset_network_finished := false;
   restart_flag := false;

   if old_trigger_connect_to_server or old_trigger_send_to_server then
      trigger_send_data := true;
   end_if;
end_if;

(* If the value of reActive_GPRS_network is true,    *)
(* do as following to reactive GPRS network service *)
if reActive_GPRS_network then
   reActive_GPRS_network := false;

   (* reset all the flag about GPRS operation *)
   step := 0;
   trigger_connect_to_server := false;
   trigger_send_to_server := false;

   (* Set GPRS_Sock_Command as 4, then the GPRS network service will reactive *)
   GPRS_Sock_Command := 4;
   reset_network_finished := true;
end_if;

(* Save the value of GPRS_Network_Ready every cycle for comparing *)
Old_GPRS_Network_Ready := GPRS_Network_Ready;

(* Keep the value of GPRS_errno to observe easily*)
if GPRS_errno <> 0 then
         temp_errno := GPRS_errno;
end_if;
```