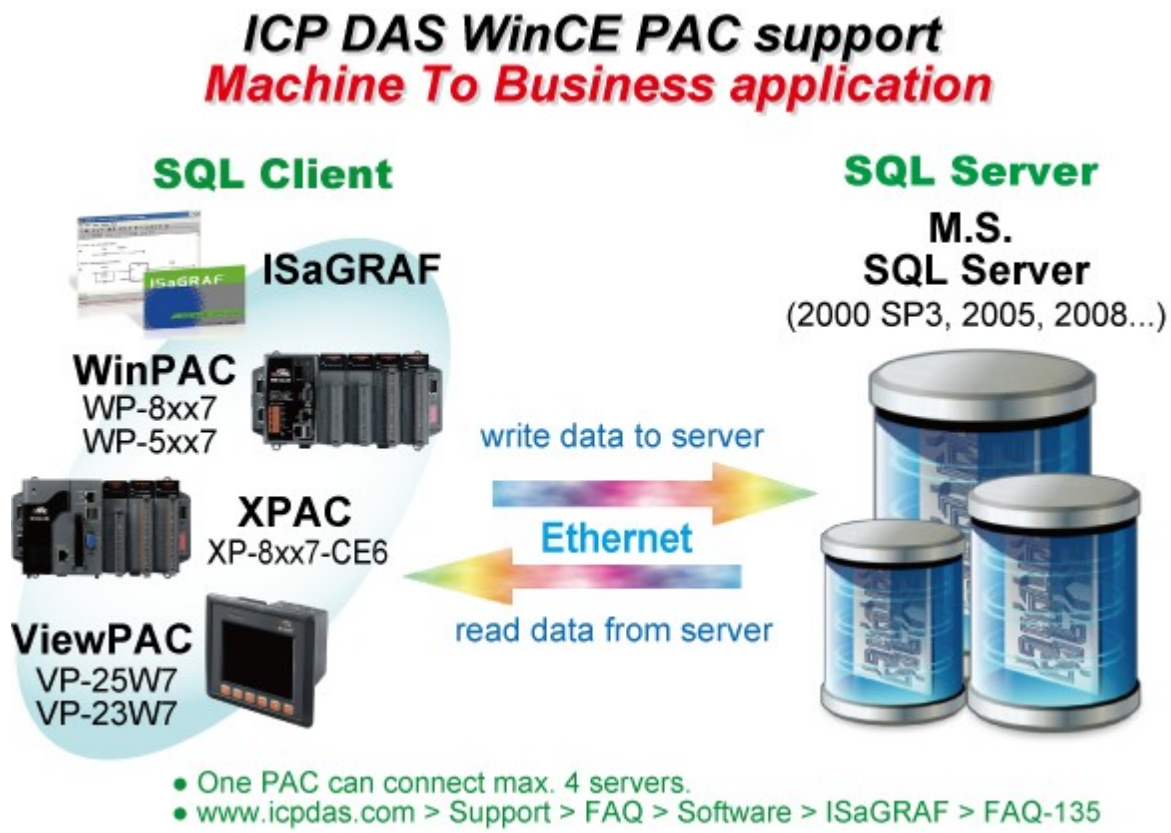
How to program ISaGRAF PAC to support SQL Client to write data to (or read data from) Microsoft SQL server?

The following ISaGRAF PACs support SQL Clinet to write data to the MicroSoft SQL server (2003 SP3 or 2005 or 2008 or compatible version), or read data from MicroSoft SQL server. This feature makes the PAC can communicate to the SQL server directly. It is useful for the Machine to Business application.

XP-8xx7-CE6 (ISaGRAF driver version 1.14 or later version)
WP-8xx7 (ISaGRAF driver version 1.34 or later version)
VP-2xW7 (ISaGRAF driver version 1.26 or later version)
and future released WP-5xx7 PAC.



Visit http://www.icpdas.com/faq/isagraf.htm > FAQ-135 to get this paper and demo programs.

Visit http://www.icpdas.com/products/PAC/i-8000/isagraf-link.htm for new ISaGRAF drivers.

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

Important note : make sure the langauage is same in the SQL server and controller.

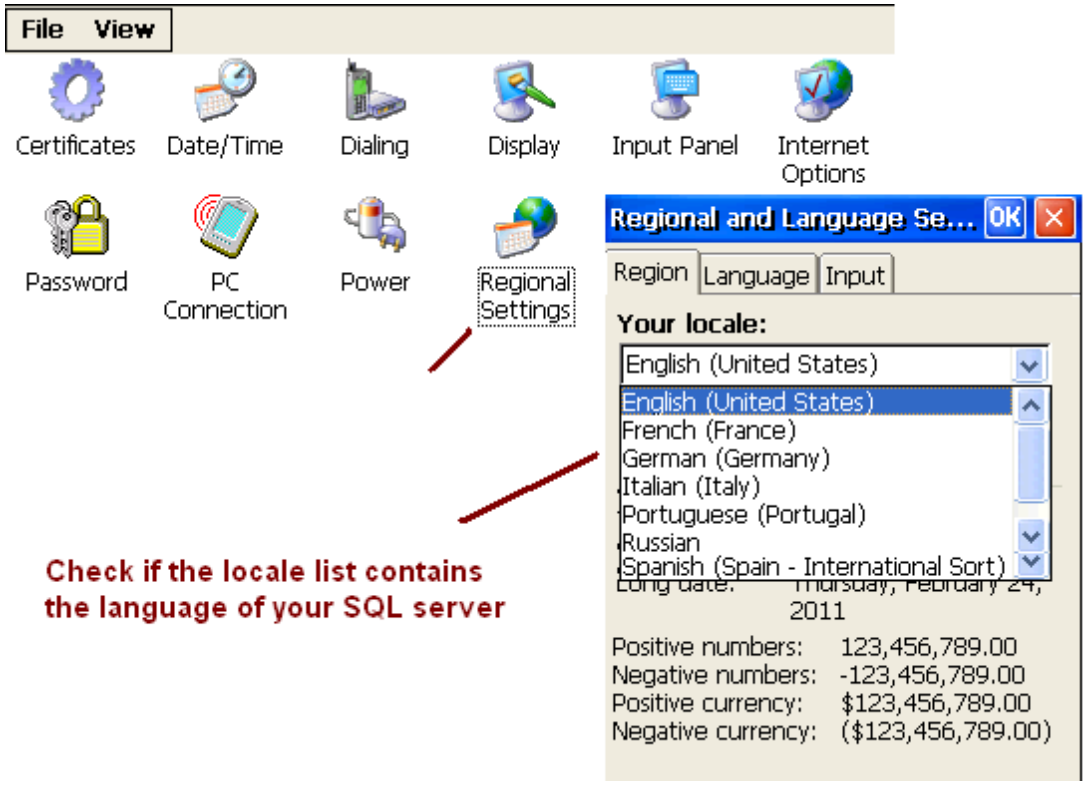
Please make sure the locale list of your controller contains the language of PC where your SQL server installed (“collation” options). For example, if you bought a WP-8847-EN or VP-25W7-EN, you will find the locale list in the “Region Settings” in the “Control Panel” as the following figure. It contains “English” , “French” , “German” , “Italian” , “Portuguese” , “Russian” and “Spanish” . Then if your SQL server is installed in a PC with the language as same as one of the above, the ISaGRAF PAC can communicate with the SQL server well. However if it is not, the PAC will have problem when communicating with the Microsoft SQL server.

If the language of your PC / SQL server is English (or French , German , Italian , Portuguese , Russian or Spanish) , please buy WP-8xx7-EN or VP-25W7-EN or VP-23W7-EN or XP-8xx7-CE6.

If the language of your PC / SQL server is Traditional Chinese, please buy WP-8xx7-TC or VP-25W7-TC or VP-23W7-TC or XP-8xx7-CE6.

If the language of your PC / SQL server is Simplified Chinese, please buy WP-8xx7-SC or VP-25W7-SC or VP-23W7-SC or XP-8xx7-CE6..

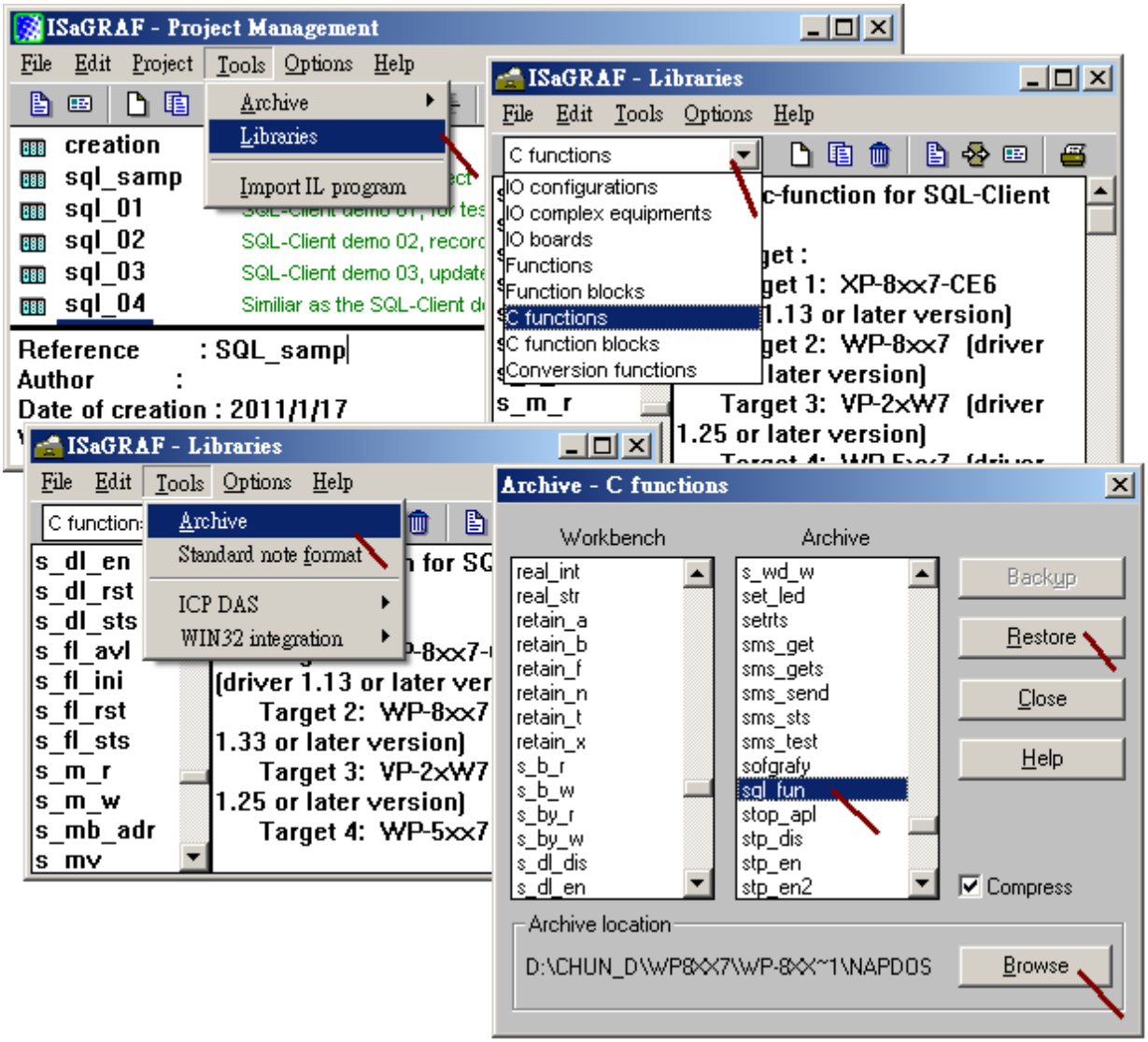
The XP-8xx7-CE6 default contains lots of language (includes all languages listed above).



1.1 : Restore the “SQL_fun” and example programs to the PC / ISaGRAF

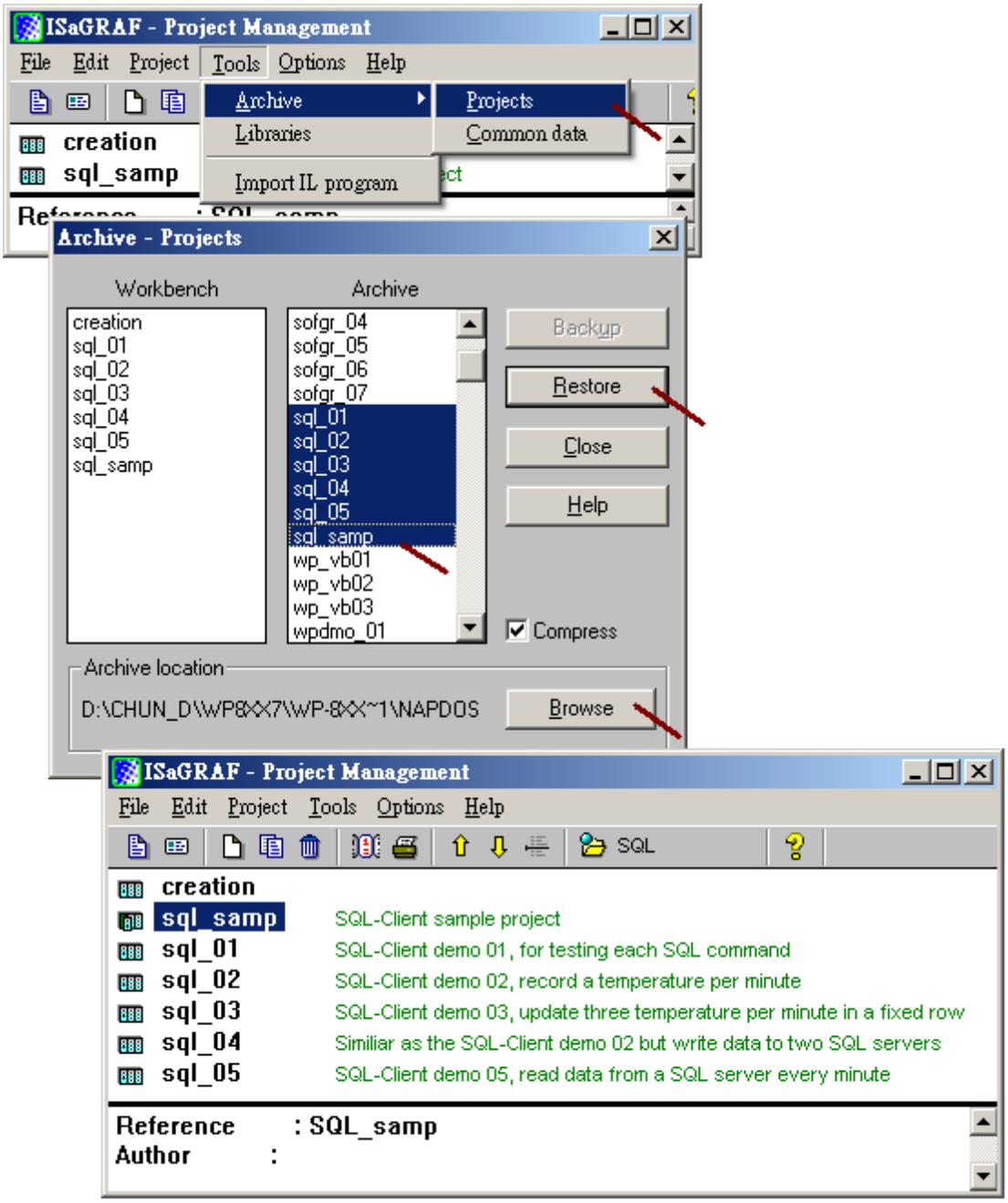
Retore the “SQL_fun.uia” :

To write program to communicate with the MicroSoft SQL Server, first restore the ISaGRAF c-function “SQL_fun.uia” to the PC / ISaGRAF.The “SQL_fun.uia” is inside the “faq135_demo.zip” which is downloaded from http://www.icpdas.com/faq/isagraf.htm > FAQ-135.



Retore example programs:

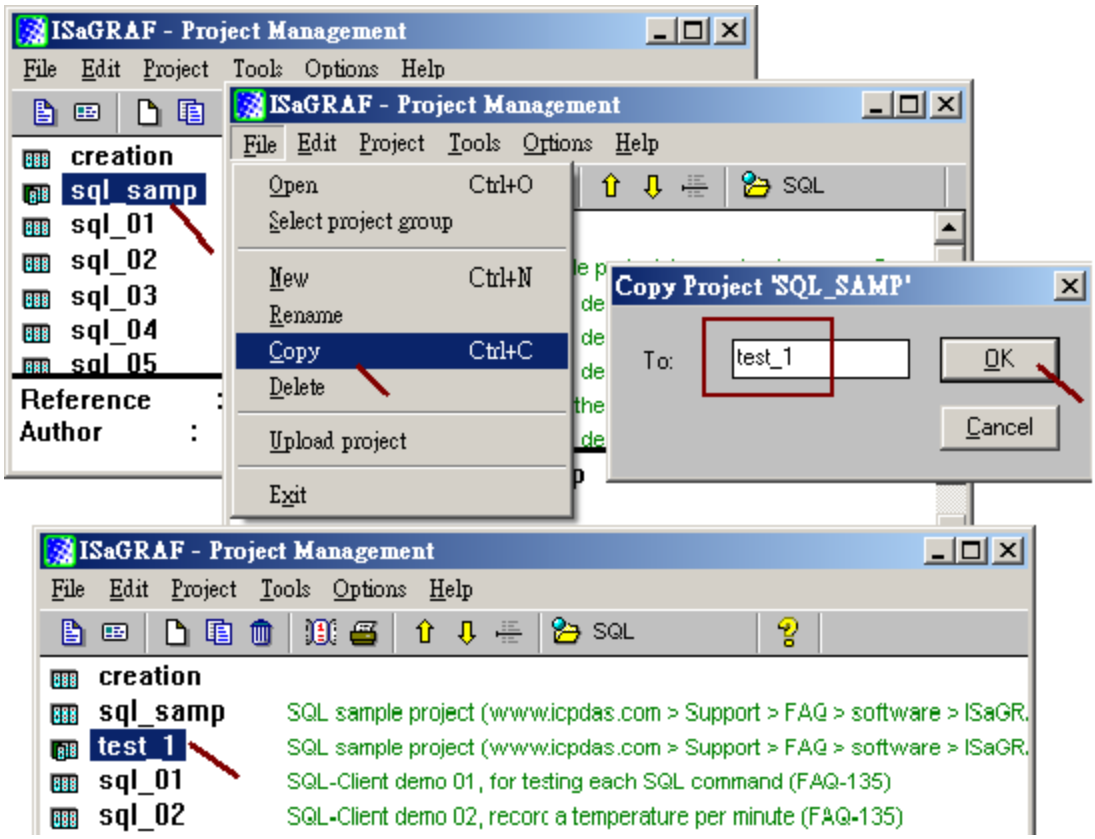
The “faq135_demo.zip” contains one sample project “SQL_samp.pia” and five example projects (“SQL_01.pia” to “SQL_05.pia”). Please restore them to the PC / ISaGRAF. Then user can refer to them easily.



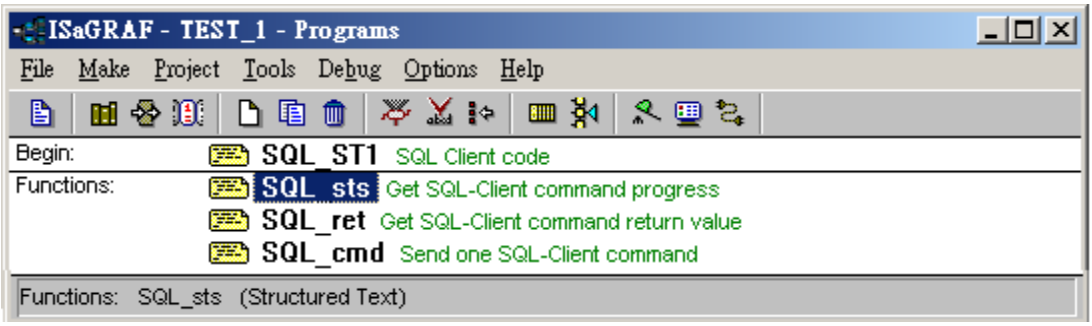
1.2 : Write SQL Client program to communicate with the Microsoft SQL Server

If you are not familiar with the ISaGRAF programming, recommend to refer to the Chapter 2 of the "ISaGRAF User's manual" to study to write a small program. The PDF manual is the "user_manual_i_8xx7.pdf" which can be found in the CD-ROM delivered with the ISaGRAF PAC. Or visit the following web site to download it
http://www.icpdas.com/products/PAC/i-8000/getting_started_manual.htm (about 21MB).

First copy the "SQL_samp" to a new project as the following figure.

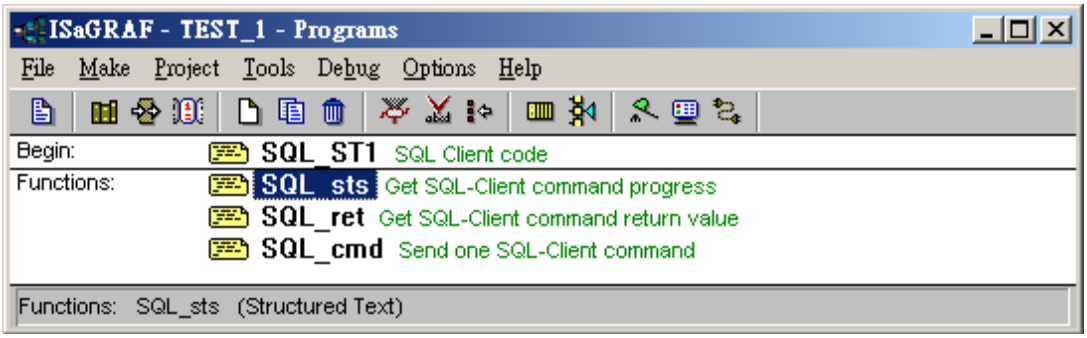


Then open this new copied project, the following windows shows up. It contains three SQL Client functions, the "SQL_sts" , "SQL_ret" and "SQL_cmd" and one "SQL_ST1" program. Please refer to the section 1.3 for detail description of these three SQL Clinet functions. Then user can add codes inside this project to communicate with the SQL server. (refer to the section 1.4 for description of some demo projects).



1.3 : Description of SQL Client functions

There are three SQL Client funstions – SQL_sts , SQL_ret and SQL_cmd .



SQL_sts : get progress of the current executing SQL Clinet command.

Usage : (* “progress1” is decalred as an Internal Integer *)
progress1 := SQL_sts(No) ;

Parameter : No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.

Return :
0 : No command.
1 : Proceed ... (Busy) .
21 : Command completed successfully.
-1 : Command failed.
-9 : Invalid parameters.

SQL_ret : get the SQL Client command result after command completed successfully.
(SQL_ret is only for the “check_table” and “read_data” command)

Usage : (* “SQL_ret_val1” is declared as an Internal Integer *)
SQL_ret_val1 := SQL_ret(No) ;

Must use the SQL_sts to check if command completed successfully before using the SQL_ret.

Parameter : No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.

Return :

-9 : Invalid parameters.

For “check_table” command :
0 : Table doesn't exist.
1 : Table does exist.

For “read_data” command :
0 : No data read (no data found in the table)
1 : One row data read and one row data found in the table.
2 : One row data read and at least two row data found in the table.

SQL_cmd : execute one SQL Client command

Usage : (* do_it is declared as an Internal Boolean, TMP_Val is an Internal integer *)

```
if  do_it  then
   do_it := False ;
   TMP_val := SQL_cmd( No , Cmd , cmd_data1 , cmd_data2 , cmd_data3 ) ;
   return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd : Message, can be one of the following.

'setup' :	setup one SQL Client.
'check_connection' :	to check the connection between the PAC and the SQL Server.
'check_table' :	check if a table exist in the SQL server.
'create_table' :	create a table in the SQL server.
'delete_table' :	delete a table in the SQL server.
'insert_data' :	insert one row data in the table.
'update_data' :	update data of existing rows in the table.
'delete_data' :	delete rows in the table.
'read_data' :	read one row data in the table.

Cmd_data1 ~ cmd_data3 : Message, no use for this command, set as '' (empty message).

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

Note :

1. The 'setup' command can run only in the first PLC scan.
2. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
3. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'setup' command : setup one SQL Client

Usage : (* SQL_INIT1 is decalred as an Internal Boolean and initial value at TRUE,
TMP_Val is an Internal integer. Refer to demo programs - “SQL_01” to “SQL_05” *)

```
if  SQL_INIT1  then
    SQL_INIT1 := False ;
    TMP_val := SQL_cmd( No , 'setup' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
     ...
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the identification information of the SQL Server.
For example, 'server=192.168.1.164\CHUN\SQLEXPRESS,1433; user id=sa; password=ABcd; database=Base1;'
For example, 'server=192.168.1.164,1433; user id=sa; password=ABcd; database=Base1;'
Cmd_data2 : Message, The starting index number of the MsgAry_R( ) which will store the data read when gives the 'read_data' command and read one row successfully. Value can be '1' to '1000' . For ex,if set as '5' and successfully read a row which contains 3 data, these data will be stored in the MsgARY_R(5) , MsgARY_R(6) , MsgARY_R(7) . The 'read_data' command can read max. 10 data in one row at a time.
Cmd_data3 : Message, the language encoding of the SQL Client.
For example, 'UTF-8' means English, 'big5' means Traditional-Chinese, 'gb2312' means Simplified-Chinese or ther language encoding supported by the PAC. Setting as '' (empty message) means English.

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

Note :

1. The 'setup' command can run only in the first PLC scan.
2. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
3. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'check_connection' command : check the connection between PAC and the SQL Server.

Usage : (* to_check_conn1 is declared as an Internal Boolean, TMP_Val is a Internal integer.
Refer to demo programs - “SQL_01” to “SQL_05” *)

```
if to_check_conn1 then
  to_check_conn1 := False ;
  TMP_val := SQL_cmd( No, 'check_connection' , Cmd_data1, Cmd_data2, Cmd_data3 );
  return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 ~ Cmd_data3 : Message, no use for this command, set as '' (empty message).

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'check_connection' command is given,
if the “SQL_sts” returns -1, it means “Can not connect the SQL server”.
However if “SQL_sts” returns 21, it means “Connect to the SQL server well” .
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'check_table' command : check if a table exist in the SQL server.

Usage (* to_check_table1 is declared an Internal Boolean, TMP_Val is an Internal integer . Refer to demo programs “SQL_01” *)

```
if to_check_table1 then
   to_check_table1 := False ;
   TMP_val := SQL_cmd( No , 'check_table' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
   return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 , Cmd_data3 : Message, no use for this command, set as '' (empty message).

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'check_table' command is given,
if the “SQL_sts” returns -1, it means “command failed”.
However if “SQL_sts” returns 21, then please use the “SQL_ret” to get the result.
If “SQL_ret” returns 1 means “table does exist” . However 0 means “table doesn't exist”.
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'create_table' command : create a table in the SQL server

Usage : (* to_create_table1 is declared as an Internal Boolean, TMP_Val is an Internal integer.
Refer to demo programs - “SQL_01” , “SQL_02” and “SQL_04” *)

```
if to_create_table1 then
  to_create_table1 := False ;
  TMP_val := SQL_cmd( No , 'create_table' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
  return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 : Message, defines the table-format.
For example, 'time varchar(8) , temperature real' means there are two columns in each row.
The column identifier are 'time' and 'temperature' respectively.
And the format of each column are 'varchar(8)' and 'real' respectively.
For example, ' id int , value1 real , value2 int , value3 int , value4 varchar(255)' means there are five columns in each row.
The column identifier are 'id' , 'value1' , 'value2' , 'value3' and 'value4' respectively.
And the format of each column are 'int' , 'real' , 'int' , 'int' and 'varchar(255)' respectively.
Cmd_data3 : Message, no use for this command, set as '' (empty message).

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'create_table' command is given,
if the “SQL_sts” returns -1, it means “Command failed”.
However if “SQL_sts” returns 21, it means “Create the table successfully” .
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'delete_table' command : delete a table in the SQL server

Usage : (* to_delete_table1 is declared as an Internal Boolean,TMP_Val is an Internal integer
Refer to demo programs “SQL_01” *)

```
if  to_delete_table1  then
    to_delete_table1 := False ;
   TMP_val := SQL_cmd( No , 'delete_table' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
    return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 ~ Cmd_data3: Message, no use for this command, set as '' (empty message).

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'delete_table' command is given,
if the “SQL_sts” returns -1, it means “Command failed”.
However if “SQL_sts” returns 21, it means “Delete the table successfully” .
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'insert_data' command : insert one row data in the table

Usage (* to_insert_data1 is declared as an Internal Boolean, TMP_Val is an Internal integer.
Refer to demo programs “SQL_01” , “SQL_02” and “SQL_04” *)

```
if to_insert_data1 then
  to_create_data1 := False ;
  TMP_val := SQL_cmd( No , 'insert_data' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
  return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 : Message, the identifier list of columns which need to insert data.
For example, 'id, integer1, real2, msg3' means to insert data to 4 columns – the 'id' , 'integer1' , 'real2' and the 'msg3' column.
Cmd_data3 : Message, value of the above columns.
For example '4 , 56796 , 23.094 , **$'**Chun Tsai**$'** '
Note: If the column format is text, like the 'varchar( )', 'text', then the message value specified in the ISaGRAF program should be included between two **$'** .
for example, the last column value of the above example is Chun Tsai.

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'insert_data' command is given,
if the “SQL_sts” returns -1, it means “Command failed”.
However if “SQL_sts” returns 21, it means “Insert one row successfully” .
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'update_data' command : update data of existing rows in the table.

Usage : (* to_update_data1 is declared as an Internal Boolean,TMP_Val is an Internal integer.
Refer to demo programs - “SQL_01” and “SQL_03” *)

```
if to_update_data1 then
   to_update_data1 := False ;
   TMP_val := SQL_cmd( No , 'update_data' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
   return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 : Message, the identifier and value list of the columns to be updated.
For example, 'integer2=37, real3=7.29' means to update the value of the 'integer2' column as 37 and update the value of the 'real3' column as 7.29 .
Cmd_data3 : Message, defines the search-condition.
For example, 'id=5' will search rows which with 'id' column 's value is 5.
For example, 'name= **$'**Chun Tsai**$'** ' will search rows which with 'name' column 's value is 'Chun Tsai' (<-- its format is text or varchar( ))
Note: If the column format is text, like the 'varchar( )', 'text', then the message value specified in the ISaGRAF program should be included between two **$'** .
for example, the text of the second example is Chun Tsai .

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'update_data' command is given,
if the “SQL_sts” returns -1, it means “Command failed”.
However if “SQL_sts” returns 21, it means “Update data successfully” . All rows which fit the given search-condition will be updated.
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'delete_data' command : delete rows in the table.

Usage : (* to_delete_data1 is declared as an Internal Boolean, TMP_Val is an Internal integer.
Refer to demo programs - “SQL_01” *)

```
if  to_delete_data1  then
   to_delete_data1 := False ;
   TMP_val := SQL_cmd( No , 'delete_data' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
   return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 : Message, defines the search-condition.
For example, 'id=5' will search rows which with 'id' column 's value is 5.
For example, 'name= **$'**Chun Tsai**$'** ' will search rows which with 'name' column 's value is 'Chun Tsai' (<-- its format is text or varchar( ))
Note: If the column format is text, like the 'varchar( )', 'text', then the message value specified in the ISaGRAF program should be included between two **$'** .
for example, the text of the second example is Chun Tsai .
Cmd_data3 : Message, no use for this command, set as '' (empty message).

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'delete_data' command is given,
if the “SQL_sts” returns -1, it means “Command failed”.
However if “SQL_sts” returns 21, it means “Delete data successfully” . All rows which fit the given search-condition will be deleted.
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

'read_data' command : read one row data in the table.

Usage : (* to_read_data1 is declared as an Internal Boolean, TMP_Val is an Internal integer.
Refer to demo programs - “SQL_01” and “SQL_05” *)

```
if to_read_data1 then
   to_read_data1 := False ;
   TMP_val := SQL_cmd( No , 'read_data' , Cmd_data1 , Cmd_data2 , Cmd_data3 ) ;
   return ;
end_if ;
```

Parameters :

No : Integer, value can be 1 to 4. One PAC can setup max, 4 SQL Clients.
Cmd_data1 : Message, the table name. For example, 'Machine1' or 'Tabler_01'
Cmd_data2 : Message, the identifier list of columns which to be read.
For example, 'integer2 , real3' means to read data in the 'integer2' and 'real3' column.
Cmd_data3 : Message, defines the search-condition.
For example, 'id=5' will search rows which with 'id' column 's value is 5.
For example, 'name= **$'**Chun Tsai**$'** ' will search rows which with 'name' column 's value is 'Chun Tsai' (<-- its format is text or varchar( ))
Note: If the column format is text, like the 'varchar( )', 'text', then the message value specified in the ISaGRAF program should be included between two **$'** .
for example, the text of the second example is Chun Tsai .

Only the first row found which fit the search-condition will be read. The data will be stored in the MsgARY_R( ) . Please refer to the description of the 'setup' command.
For example, if parameter “Cmd_data2” of the 'setup' command is set as '5' and successfully read one row which contains 3 data, then these data will be stored in the MsgARY_R(5) , MsgARY_R(6) , MsgARY_R(7) .
The 'read_data' command can read max. 10 data in one row at a time.

Return :

-9 : Invalid parameter.
1 : Parameter Ok.

After the 'read_data' command is given,
if the “SQL_sts” returns -1, it means “command failed”.
However if “SQL_sts” returns 21, then please use the “SQL_ret” to get the result.
If “SQL_ret” returns 0, it means “No data read” .
If “SQL_ret” returns 1, it means “One row data read and found only one row” .
If “SQL_ret” returns 2, it means “One row data read and found at least two rows” .
If the “SQL_sts” returns 1, it means “busy to process the current given command” .
(The PAC 's SQL Client engine will wait max. 45 senconds.Normally if the communication is ok, the “SQL_sts” will retuns 21 or -1 in about 1 to 5 seconds).

Note :

1. DO NOT give other command except the 'setup' when the “SQL_sts” returns 1 (busy).
2. One SQL Client can give only one command at a time. The next command must wait until the return value of the “SQL_sts” is not 1. However the different SQL Client can give its own command at the same time.

1.4 : Description of some SQL Client demo programs

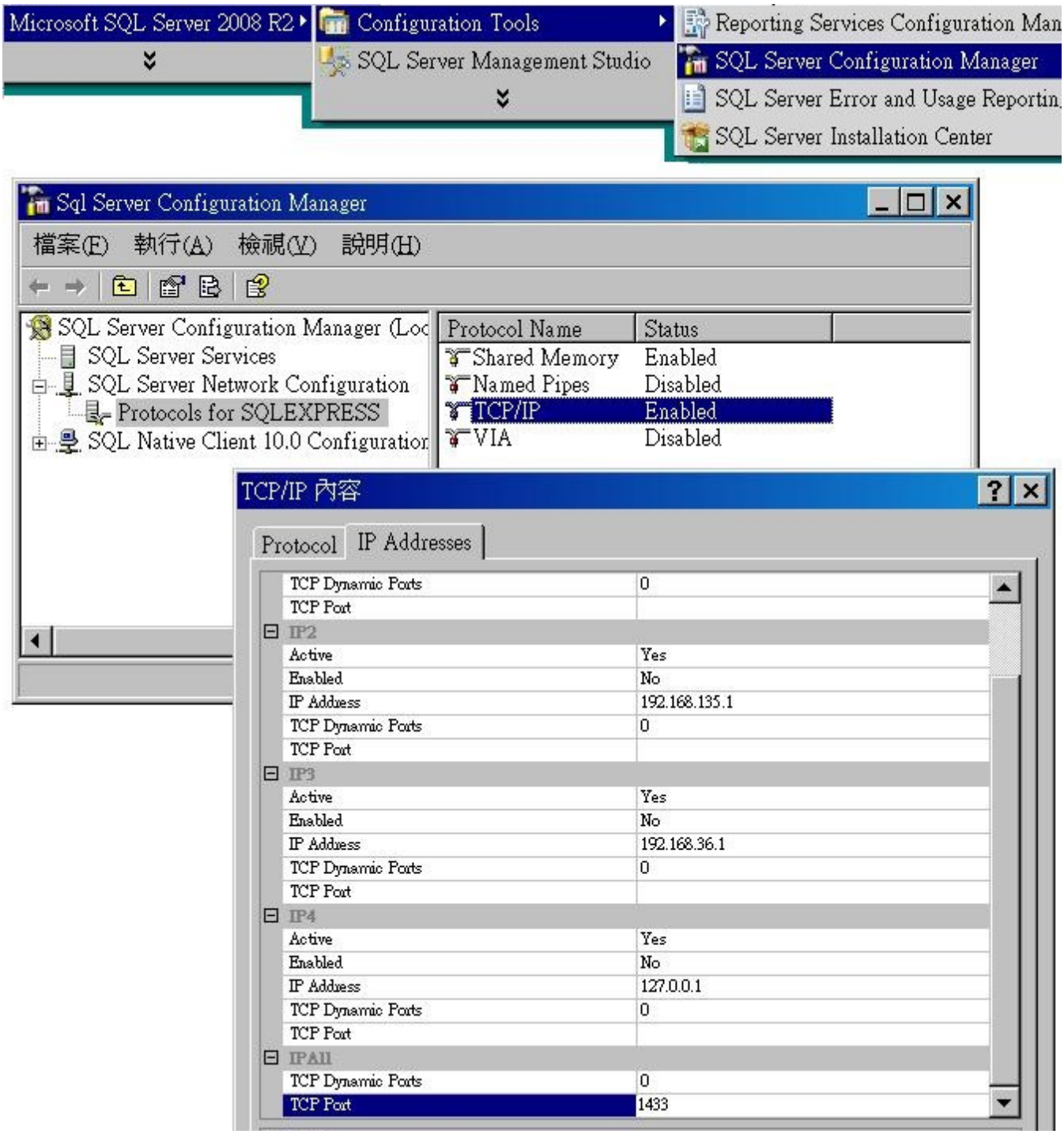
The “faq135_demo.zip” downloaded from http://www.icpdas.com/faq/isagraf.htm > FAQ-135 includes the following demo programs.

SQL_samp	SQL Client sample project. For copy to a new project (refer to section 1.2).
SQL_01	A demo project which contains every SQL Client commands.
SQL_02	To create a new table every day and insert a temperature value every minute.
SQL_03	To update the current value of 3 temperatures per minute to an existing row.
SQL_04	Similar as “SQL_02” however setup two SQL Clients to insert data to two different SQL servers (or to two different database in the same SQL server)
SQL_05	Read data from an existing row in the table.

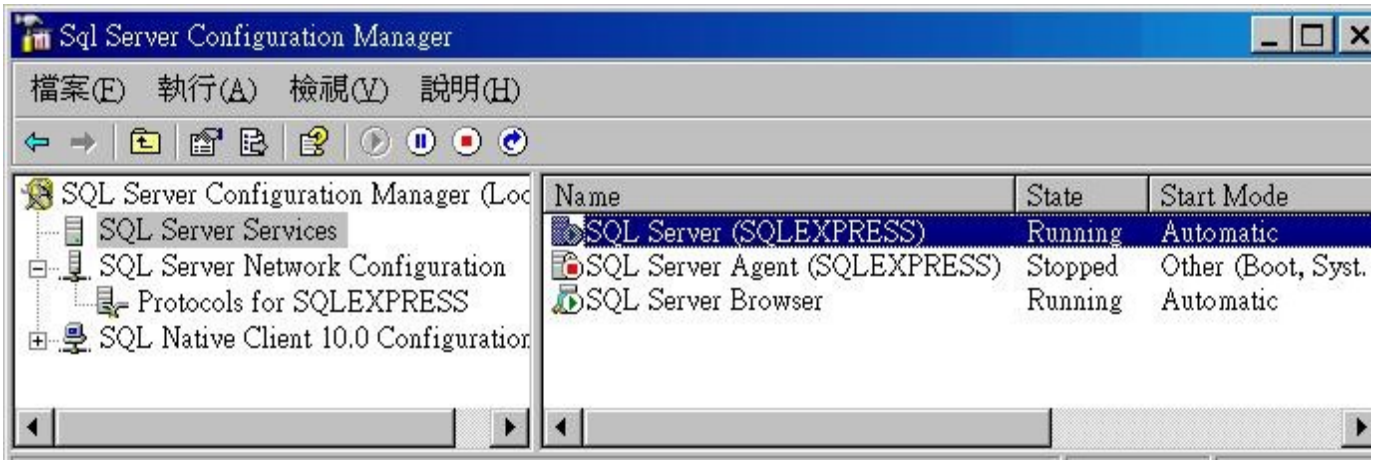
1.4.1 : Prepare one Microsoft SQL server (or setup one) for testing demo programs

Please prepare one Microsoft SQL server for testing. If find no server, user can download the “SQL Server 2008 Trial Software” and install it in a PC.
http://www.microsoft.com/sqlserver/2008/en/us/trial-software.aspx
Recommend to buy one book which is talking about the Microsoft SQL server if user is not familiar with the SQL server. When installing the SQL server and meet the below dialog, select the mode as “Mixed Mode”, enter a password and user must remember this password, (**Note:** this password of the “sa” user-id is necessary in ISaGRAF SQL Client programs)

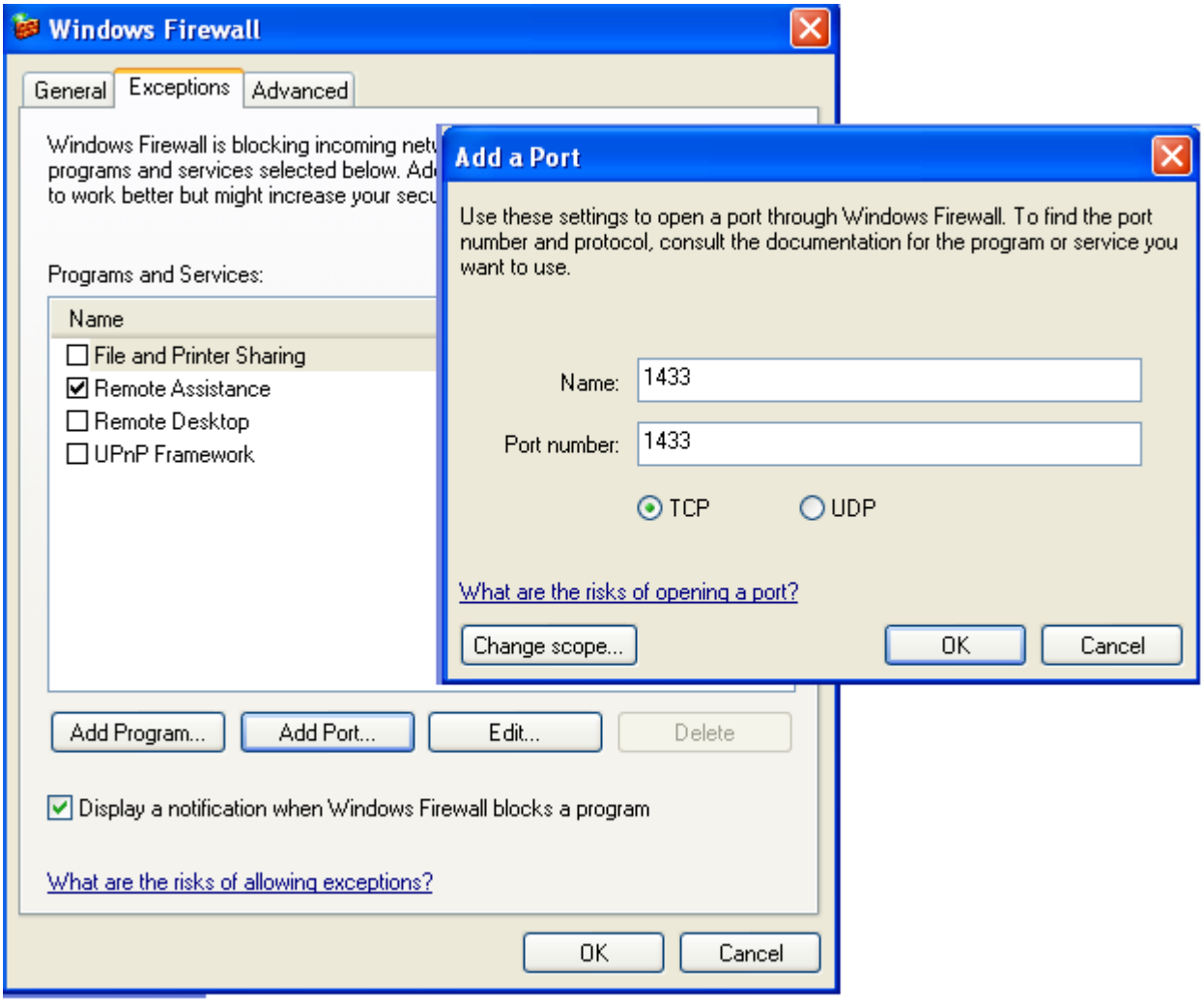
When installation completed successfully, run Configuration Tools > SQL Server Configuration Manager to set the “TCP/IP” as “Enabled” and set the “TCP Dynamic Ports” in the “IP ALL” as 0 , “TCP port” as 1433 (The demo programs “SQL_01” to “SQL_05” all use this 1433 port number. User may use his own port number but remember to modify the demo program.)



Then check if the state of the “SQL Server” is “Automatic Running” from “SQL Server Services” , if it is not, please modify it.



Then do the following setting from the Control Panel > Windows firewall.



Please run “SQL Server Management Studio” to create a new Database in the server (All demo programs “SQL_01” to “SQL_05” use a database name - “**DataBase_01**”). The Server name - “ICPDAS-RD6-CHUN\SQLEXPRESS” on the below window is also used in the demo programs. User has to modify the code in the demo programs to fit the settings of his own SQL server.

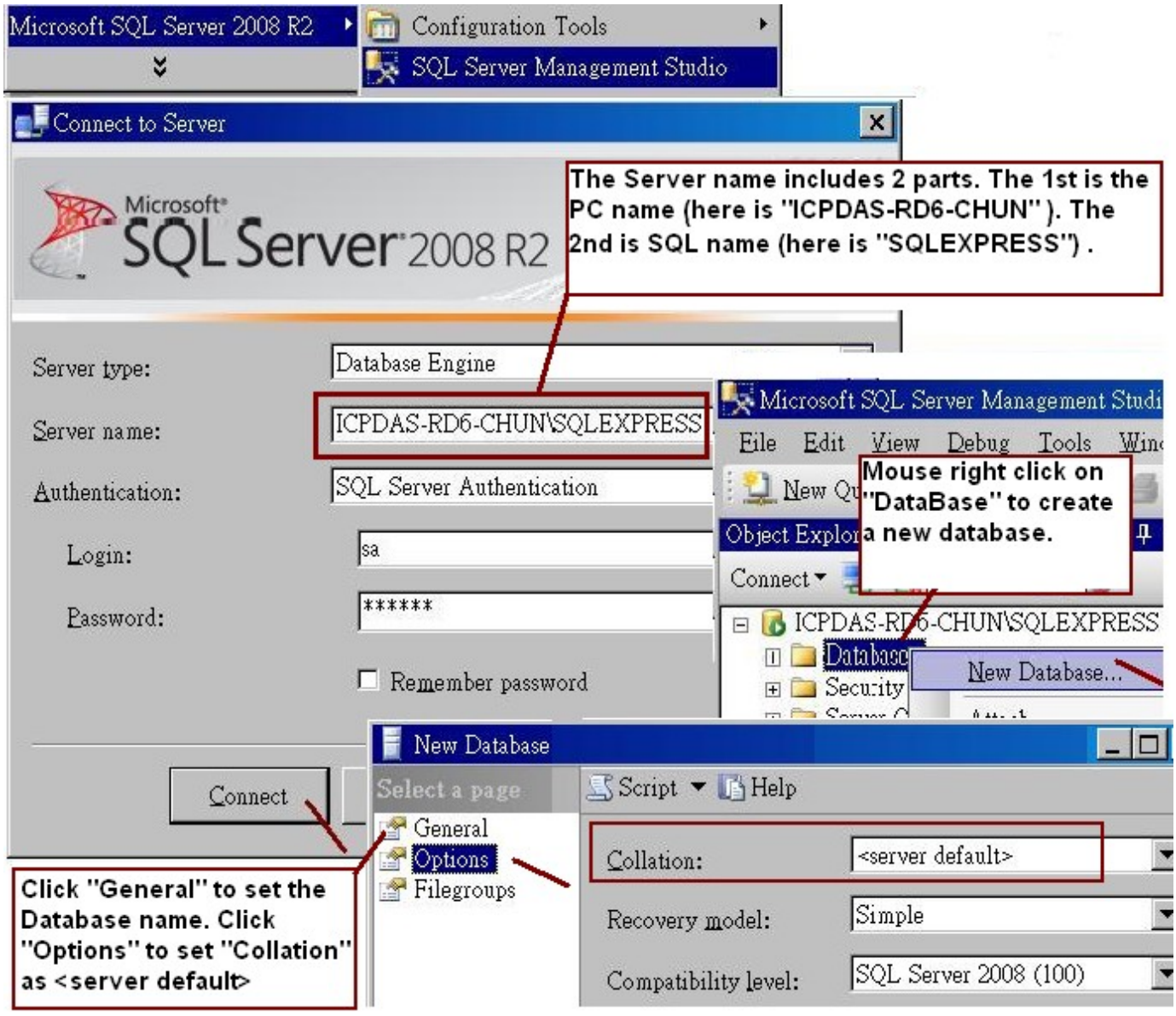
For example,there is one statement in the “SQL_01” > “SQL_ST1” demo program as following.

SQL_server1 := 'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433; user id=sa; password=AABBCC; database=DataBase_01;' ;

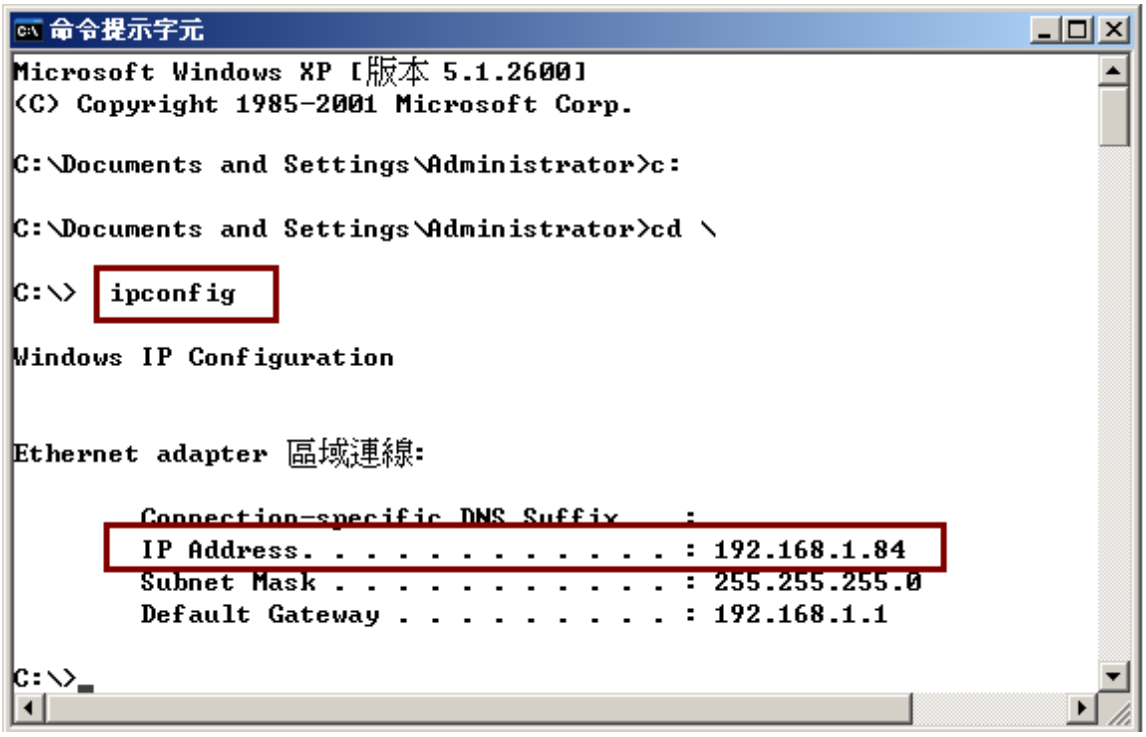
The following server setting is also Ok for some SQL servers (without PC and SQL name).

SQL_server1 := 'server=192.168.1.101,1433; user id=sa; password=AABBCC; database=DataBase_01;' ;

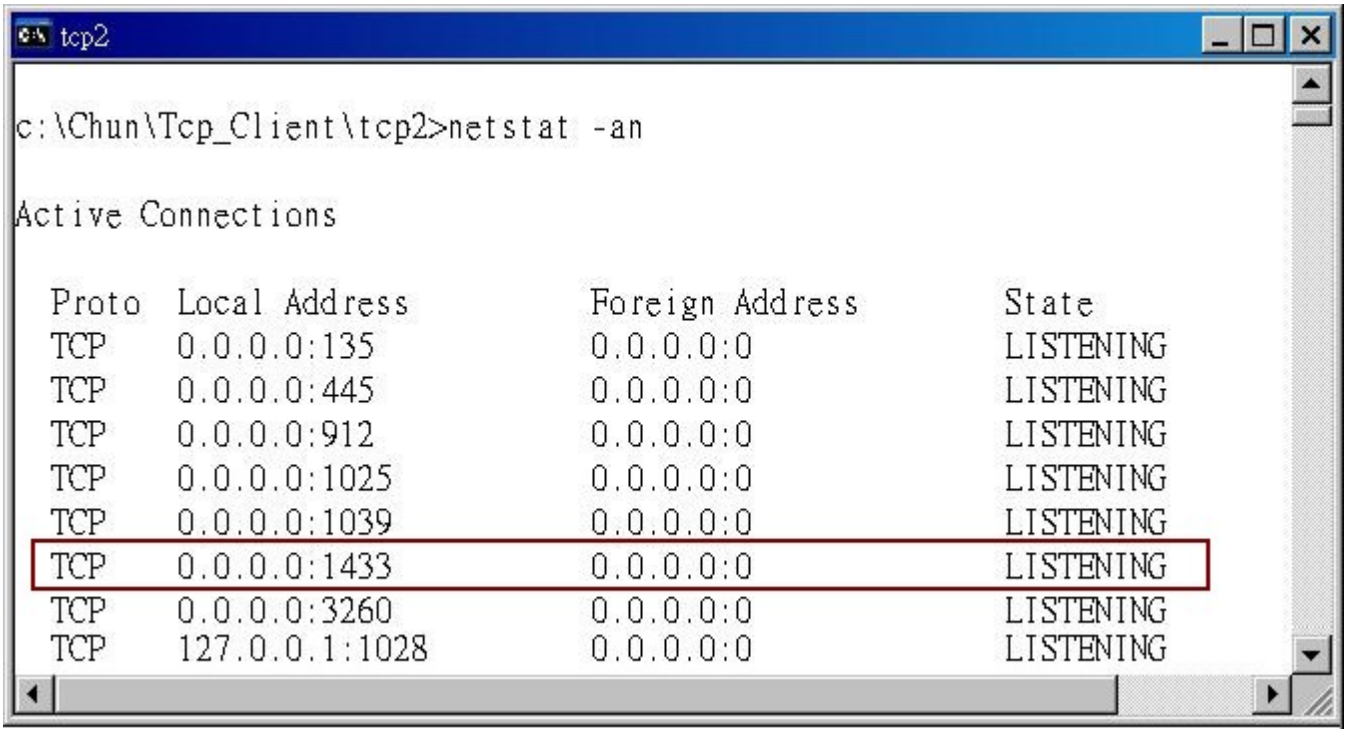
User has to modify the above statement to fit his own SQL server 's IP-address, Server-name, Port-No. , User-id, Password and Database setting.



Remember to check the IP address of the SQL server.

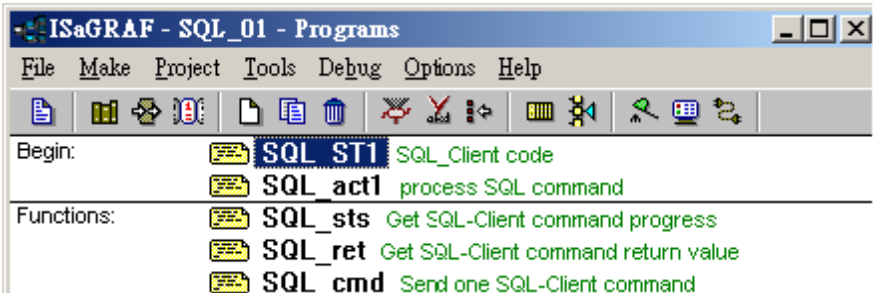


And also use “netstat -an” to check if the SQL server port is listening or not.



Then later we can test demo programs starting from the section 1.4.2.

1.4.2 : SQL_01 demo program - contains every SQL Client command



ISaGRAF Variables :

Name	Type	Description
SQL_INIT1	Boolean	**Must declare a initial value - TRUE**
TMP	Boolean	Temporary used Boolean
to_check_conn1	Boolean	To check the connection once when set as TRUE
to_check_table1	Boolean	To check if the table exist when set as TRUE
to_create_table1	Boolean	To create a new table when set as TRUE
to_delete_table1	Boolean	To delete a table when set as TRUE
to_insert_data1	Boolean	To insert one row when set as TRUE
to_update_data1	Boolean	To update existing row data when set as TRUE
to_delete_data1	Boolean	To delete existing row when set as TRUE
to_read_data1	Boolean	To read one row data when set as TRUE
TMP_Val	Integer	Temporary used Integer
SQL_No1	Integer	SQL Client number, **set initial value as 1**
SQL_status1	Integer	Communication state, 0: dusconnected , 1: connected
SQL_ret_val1	Integer	Result of the command when completed successfully. Only for the “check_table” and “read_data” command
progress1	Integer	Progress of the SQL Client command, 0: no command, 1: busy , -1: command failed , 21: completed successfully
old_progress1	Integer	Command progress of the Old one (former PLC scan)
SQL_server1	Message	The SQL server setting, includes IP, server, port, user-id, password and database. Len=255
SQL_progress1	Message	Command progress (message format). Len=255
SQL_current_cmd1	Message	The current SQL Client command, Len=32
SQL_table_name1	Message	The table name, Len=64
table_format1	Message	Table format. Len=255
insert_format1	Message	Identifier list of colums for insert_data command. Len=128
read_format1	Message	Identifier list of columns for read_data command. Len=128

insert_data1	Message	The data value for insert_data command. Len=255
where1	Message	The search-condition for update_data , delete_data and read_data command. Len=64
update_data1	Message	The identifier and value of columns for update_data command. Len=255
data_msg1	Message	A message for debug and test. Len=255
TMP_msg	Message	Temporary used message. Len=255
Data1 ~ Data3	Message	The data read from the SQL server. Len=255

"SQL_ST1" program:

```
if SQL_INIT1 then  (* operations in the first PLC scan cycle *)
  SQL_INIT1 := False ;
  (* init *)
  SQL_table_name1 := 'Tab_001' ;
  table_format1 := 'id int , integer1 int , real1 real , msg1 varchar(255)' ;
  insert_format1 := 'id , integer1 , real1 , msg1' ;
  read_format1 := 'integer1 , real1 , msg1' ;
  insert_data1 := '1,456,9.0305,$'Hello$'' ;
  SQL_status1 := 0 ;
  SQL_No1 := 1 ;
  (* Set the SQL server setting *)
  SQL_server1 := 'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433; user id=sa;
                  password=AABBCC; database=DataBase_01;' ;
  (* The following server setting is also Ok for some SQL servers.
  SQL_server1 := 'server=192.168.1.101,1433; user id=sa;
                  password=AABBCC; database=DataBase_01;' ;
  *)

  (* setup one SQL Client *)
  TMP_val := SQL_cmd( SQL_No1 , 'setup' , SQL_server1 , '1' , 'UTF-8' ) ;
  to_check_conn1 := True ;  (* set as TRUE to check connection once *)
end_if ;
(* If the connection state is disconnected, exit this "SQL_ST1" program *)
if SQL_status1= 0 then
  return ;
end_if ;
```

"SQL_act1" program :

```
progress1 := SQL_sts(SQL_No1) ;  (* get command progress *)
if old_progress1 <> progress1 then  (* if progress is changed, process it *)
  old_progress1 := progress1 ;
  SQL_progress1 := '' ;  (* first set test / debug message as empty message *)
  data_msg1 := '' ;  (* first set test / debug message as empty message *)
 case progress1 of  (* which progress ? *)
  0 : SQL_progress1 := 'No command.' ;
  1 : SQL_progress1 := 'Busy ...' ;
  21 : SQL_progress1 := 'Command completed successfully.' ;
     SQL_ret_val1 := SQL_ret(SQL_No1) ;  (* command completed successfully, get result *)
     if SQL_current_cmd1 = 'check_connection' then  (* command is check_connection *)
       SQL_status1 := 1 ;  (* set as 1: connected *)
       data_msg1 := 'Server connected.' ;
     elsif SQL_current_cmd1 = 'check_table' then  (* command is check_table *)
       if SQL_ret_val1=1 then  (* 1: table does exist *)
         data_msg1 := SQL_table_name1 + ' does exist.' ;
       else  (* 0: table doesn't exist *)
         data_msg1 := SQL_table_name1 + ' doesn$'t exist !' ;
       end_if ;
    elsif SQL_current_cmd1 = 'create_table' then  (* command is create_table *)
      data_msg1 := SQL_table_name1 + ' created.' ;
    elsif SQL_current_cmd1 = 'delete_table' then  (* command is delete_table *)
      data_msg1 := SQL_table_name1 + ' deleted.' ;
    elsif SQL_current_cmd1 = 'insert_data' then  (* command is insert_data *)
      data_msg1 := 'data inserted.' ;
    elsif SQL_current_cmd1 = 'update_data' then  (* command is update_data *)
      data_msg1 := 'update data ok.' ;
    elsif SQL_current_cmd1 = 'delete_data' then  (* command is delete_data *)
      data_msg1 := 'data deleted.' ;
     elsif SQL_current_cmd1 = 'read_data' then  (* command is read_data *)
       if SQL_ret_val1=0 then  (* 0: read no data *)
         data_msg1 := 'No data read !' ;
       elsif SQL_ret_val1=1 then  (* 1: read one row and one row found*)
         data_msg1 := 'One data read and one data found.' ;
         to_get_data1 := True ;  (* triger to store the read data to the data1 to data3 *)
     elsif SQL_ret_val1 >= 2 then  (* 2: read one row and at least two rows found *)
         data_msg1 := 'One data read and more than one data found.' ;
         to_get_data1 := True ;  (* triger to store the read data to the data1 to data3 *)
       end_if ;
     end_if ;
```

```
  -1 : SQL_progress1 := 'Command failed !' ;   (* -1: command failed *)
       if  SQL_current_cmd1 = 'check_connection'  then   (* the filed command is check_connection  *)
         SQL_status1 := 0 ;   (* set as 0: disconnected *)
        data_msg1 := 'Server disconnected !'  ;
       end_if ;
   -9 : SQL_progress1 := 'Input parameter error !' ;   (* invalid parameter *)
  end_case ;
end_if ;
if  progress1 = 1  then   (* if the command progress is 1: busy, exit this  "SQL_act1"  program *)
  return ;
end_if ;
if  to_check_conn1  then   (* command check_connection given *)
  to_check_conn1 := False ;
  SQL_current_cmd1 := 'check_connection' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , " , " , " ) ;
  return;
end_if ;
if  to_check_table1  then   (* command check_table given *)
  to_check_table1 := False ;
  SQL_current_cmd1 := 'check_table' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , " , " ) ;
  return;
end_if ;
if  to_create_table1  then   (* command create_table given *)
  to_create_table1 := False ;
  SQL_current_cmd1 := 'create_table' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , table_format1 , " ) ;
  return;
end_if ;
if  to_delete_table1  then   (* command delete_table given *)
  to_delete_table1 := False ;
  SQL_current_cmd1 := 'delete_table' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , " , " ) ;
  return;
end_if ;
```

```
if  to_insert_data1  then     (* command insert_data given *)
  to_insert_data1 := False ;
  SQL_current_cmd1 := 'insert_data'  ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , insert_format1 , insert_data1)  ;
  return;
end_if ;

if  to_update_data1  then    (* command update_data given *)
  to_update_data1 := False ;
  SQL_current_cmd1 := 'update_data' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , update_data1 , where1 ) ;
  return;
end_if ;

if  to_delete_data1  then     (* command dalete_data given *)
  to_delete_data1 := False ;
  SQL_current_cmd1 := 'delete_data' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , where1 , '' ) ;
  return;
end_if ;

if  to_read_data1  then      (* command read_data given *)
  to_read_data1 := False ;
  SQL_current_cmd1 := 'read_data' ;
  TMP_val := SQL_cmd( SQL_No1 , SQL_current_cmd1 , SQL_Table_name1 , read_format1 , where1 ) ;
end_if ;

if  to_get_data1  then
  to_get_data1 := False ;
  data1 := MSGARY_R(1) ;    (* store the read data to data1to data3 *)
  data2 := MSGARY_R(2) ;
  data3 := MSGARY_R(3) ;
  return;
end_if ;
```

How to test ?

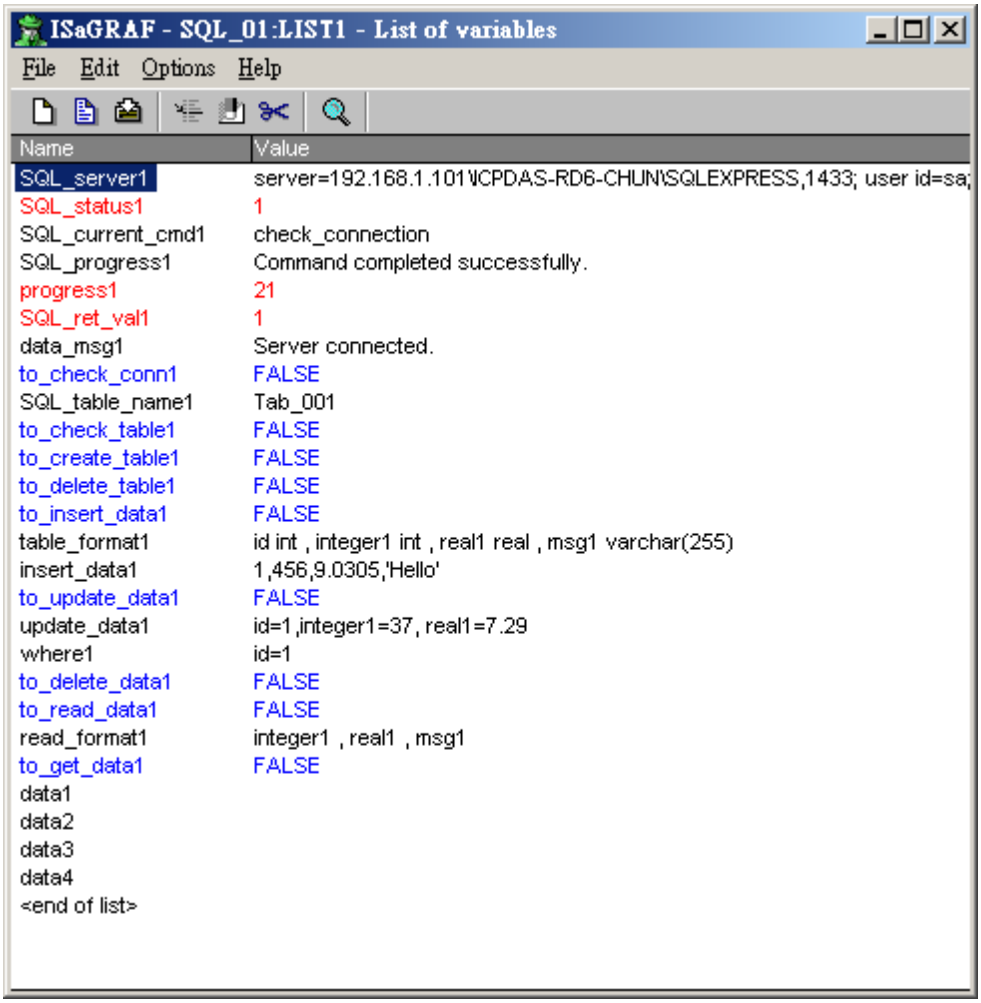
First modify the following statement in the “SQL_ST1” program to fit your own SQL server setting (refer to the section 1.4.1)

```
SQL_server1  :=  'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433;
                 user id=sa; password=AABBCC; database=DataBase_01;'  ;
```

The following server setting is also Ok for some SQL servers (without PC and SQL name).

```
SQL_server1  :=  'server=192.168.1.101,1433; user id=sa; password=AABBCC;
                 database=DataBase_01;'  ;
```

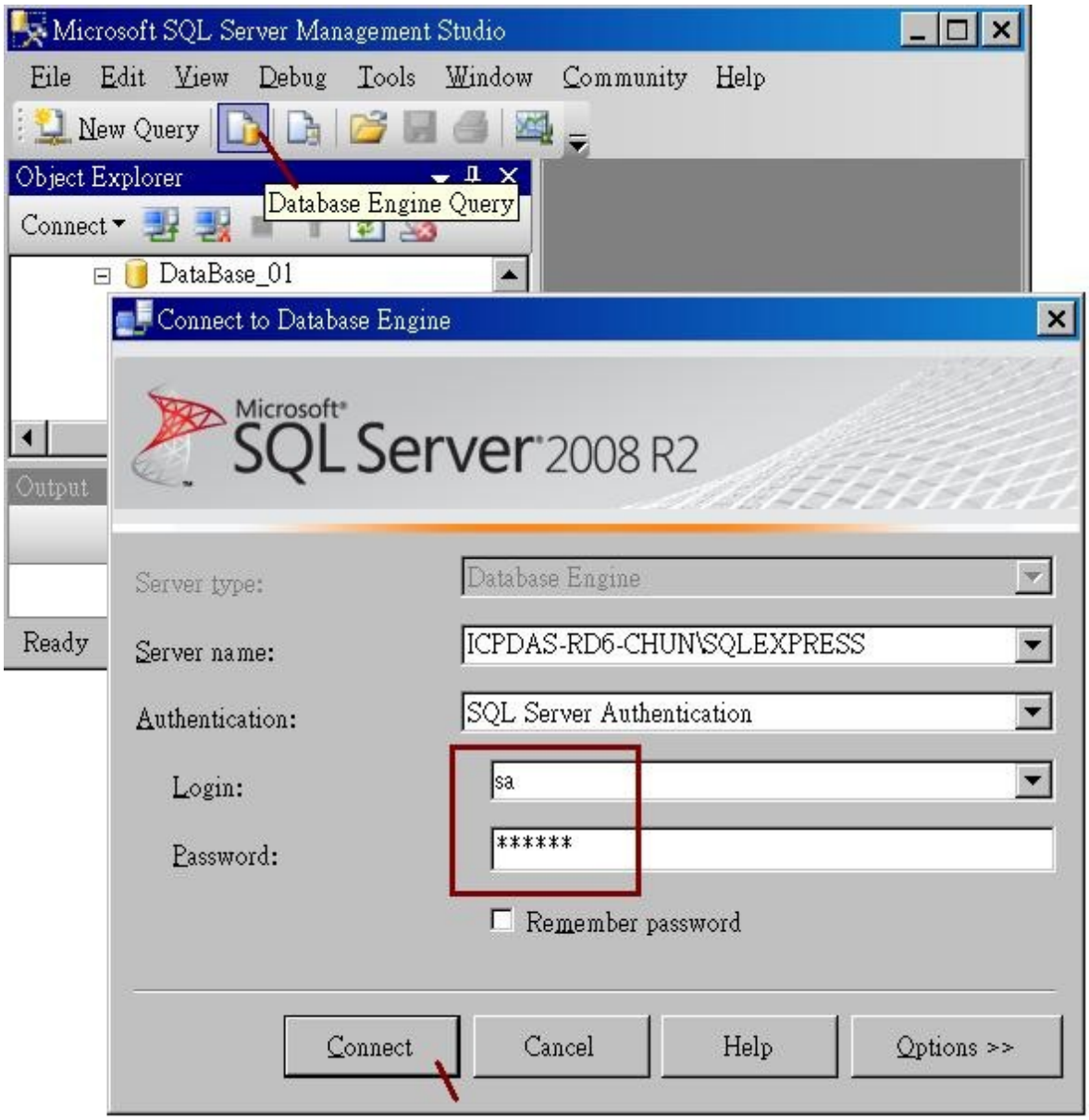
Then re-compile this “SQL_01” project and then download it to your PAC. The below window will show up in the PC / ISaGRAF. First check if the “data_msg1” shows “Server connected” . If not, try mouse-two-click on the “to_check_conn1” to set it as TRUE to check connection once. If it is still disconnected, the problem may be (1) the database “DataBase_01” doesn't exist in the SQL server (refer to the section 1.4.1 to create it) or (2) Ethernet broken-line or (3) the value of the “SQL_server1” variable in the “SQL_ST1” doesn't fit the setting of the SQL server or (4) others.



If server is connected well, then set the “to_check_table1” as True to check the table is existing or not (This example is using the table name “Tab_001”). If it doen't exist, set the “to_create_table1” as TRUE to create this “Tab_001” table.

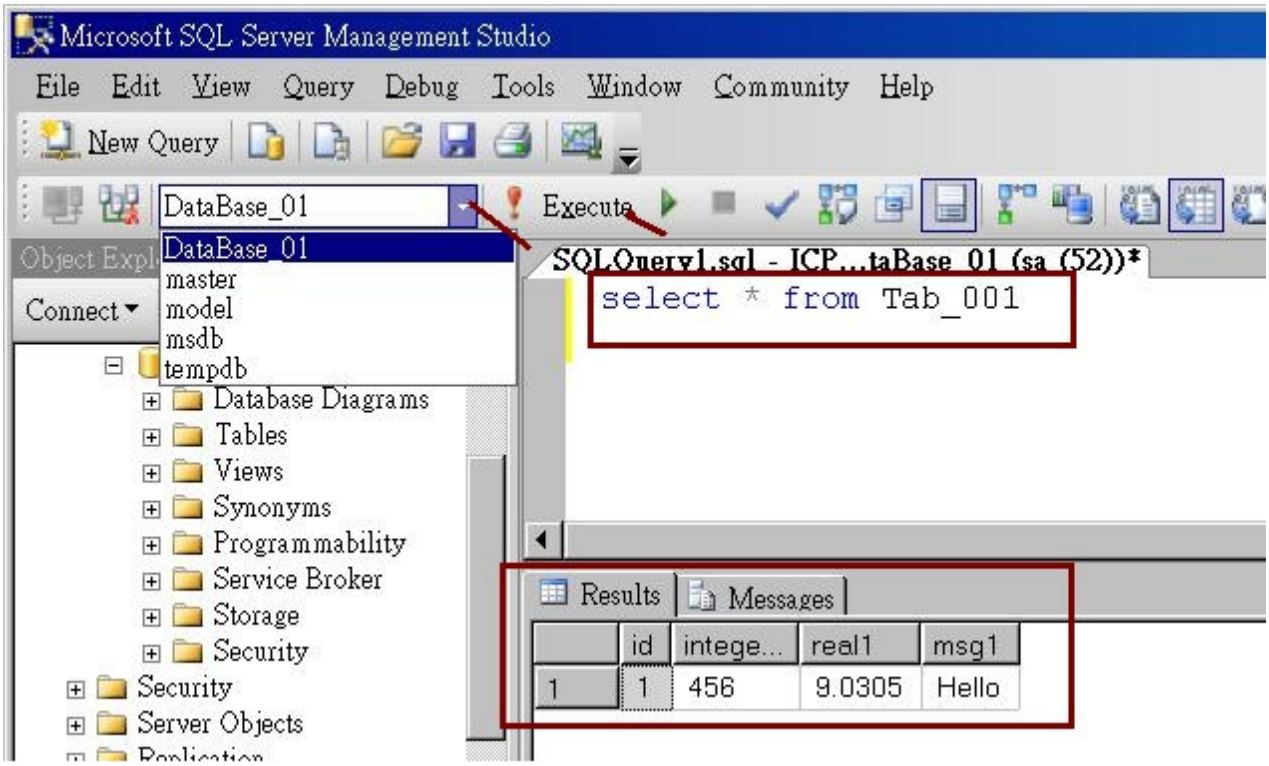
Then set the “to_insert_data1” as TRUE to insert one row data to the table.

Run “SQL Server Management Studio” to see if the row data is inserted in the SQL server.



Select the “DataBase_01” first and then key-in the following SQL server command to request the data in the table.

select * from Tab_001 (remember to click “Execute”)



Then set “to_read_data1” as True to see if the “data1” is 456 , “data2” is 9.0305 and “data3” is 'Hello' .

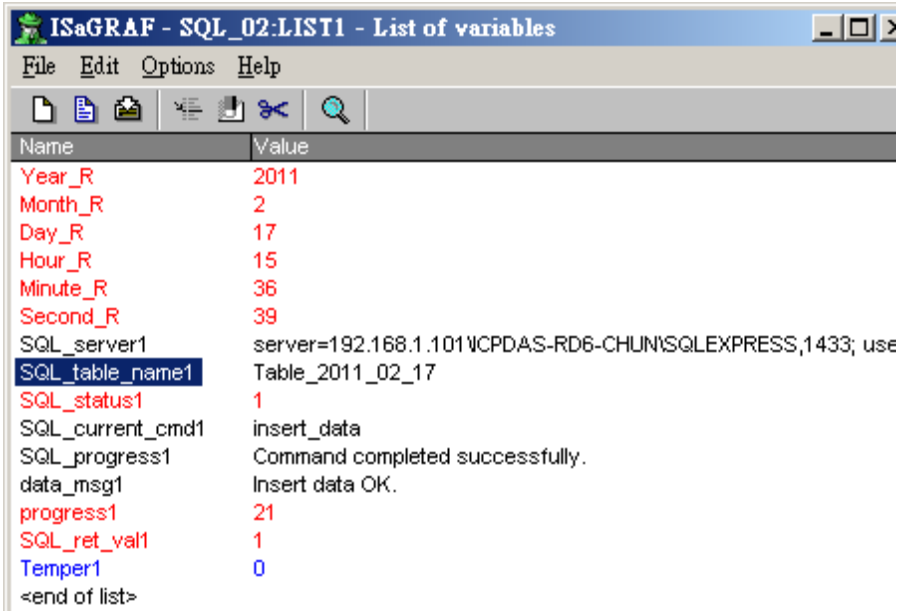
Try to set the “insert_data1” as 2 , 787, 1.023 , $'Tom$' (the ' character in the ISaGRAF message should be entered as $'). Then set “to_insert_data1” as TRUE, then check the data in the SQL Server to see if the second rows inserted (using the “select * from Tabl_001”).

Then set the “to_update_data1” as True and then check the SQL server if the data of the first row is modified.

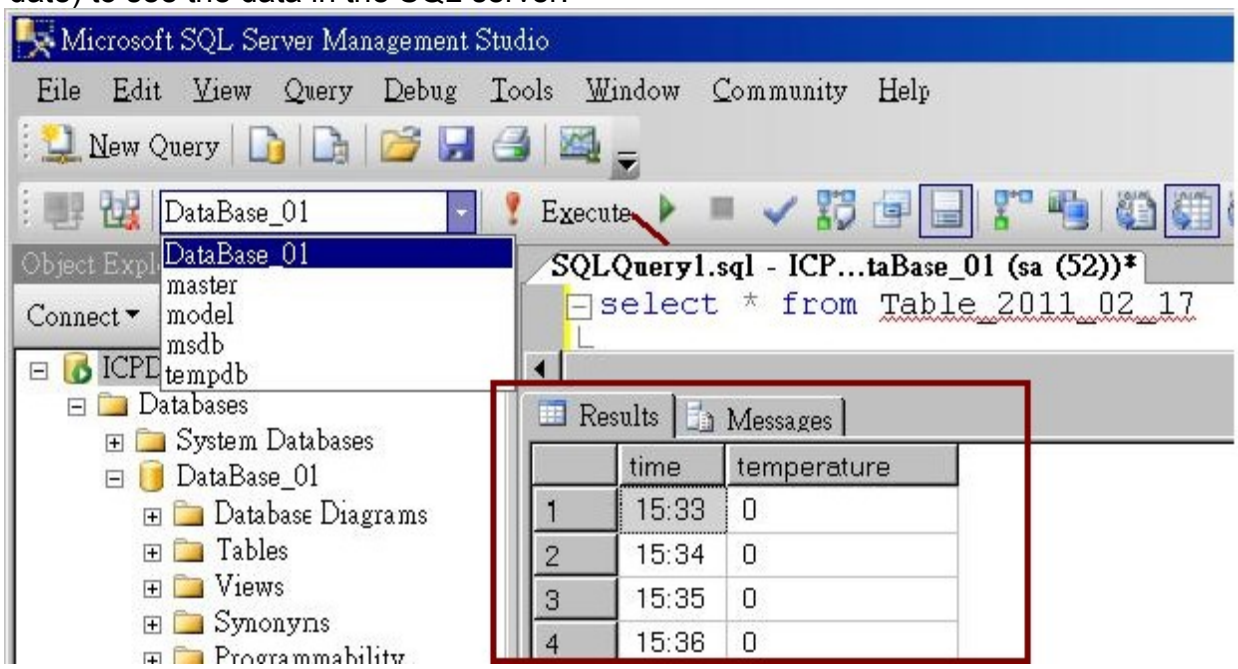
1.4.3 : SQL_02 demo program - Create a table every day and insert one row per minute

The “SQL_02” demo program will write the Slot-1 : I-87018Z 's first channel temperature value (this demo is setting as Thermocouple K type) to the SQL server every minute. The PAC will create one new table automatically when the time pass through 00:00. (If there is no such a table in the SQL server, this “SQL_02” will create one). For example, on the day of Feb.17, 2011, the created table name is “Table_2011_02_17”. First modify the following code in the “SQL_ST1” to fit your server (refer to the section 1.4.1), re-compile it and download.

```
SQL_server1 := 'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433; user id=sa;
               password=AABBCC; database=DataBase_01;' ;
```



Then run “SQL Server Management Studio” to key-in “select * from Table_2011_02_17” (use your date) to see the data in the SQL server.



1.4.4 : SQL_03 demo program - update data in the existing row in the table

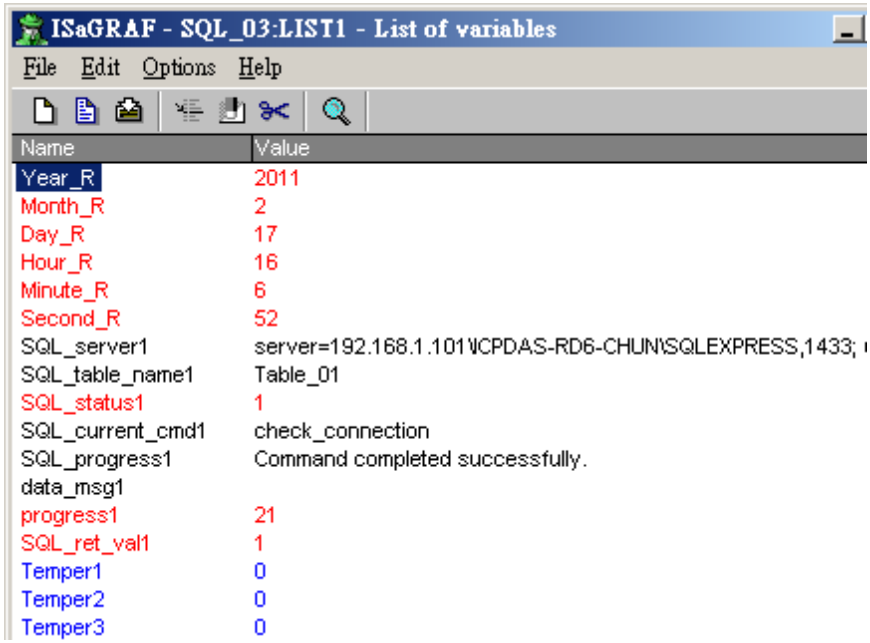
The “SQL_03” will update the Slot-1 : I-87018Z 's channel-1 to channel 3 's temperature value (this demo is setting as Thermocouple K type) to the SQL server every minute.

To well test this “SQL_03” demo , first run “SQL Server Management Studio” to key-in the following SQL server command to create a table “Table_01” and insert one row inside it . Or the PAC will not update the data sucessfully.

```
Create table Table_01( Item varchar(64) , time varchar(8) , temper1 real , temper2 real , temper3 real )
Insert into table_01(Item , time , temper1 , temper2 , temper3)  values( 'Current' , '-1:-1' , '-99' , '-99' , '-99' )
Select  *  from  table_01          (remember to click  “Execute” )
```

First modify the following code in the “SQL_ST1” to fit your server (refer to the section 1.4.1), re-compile it and download it to the PAC.

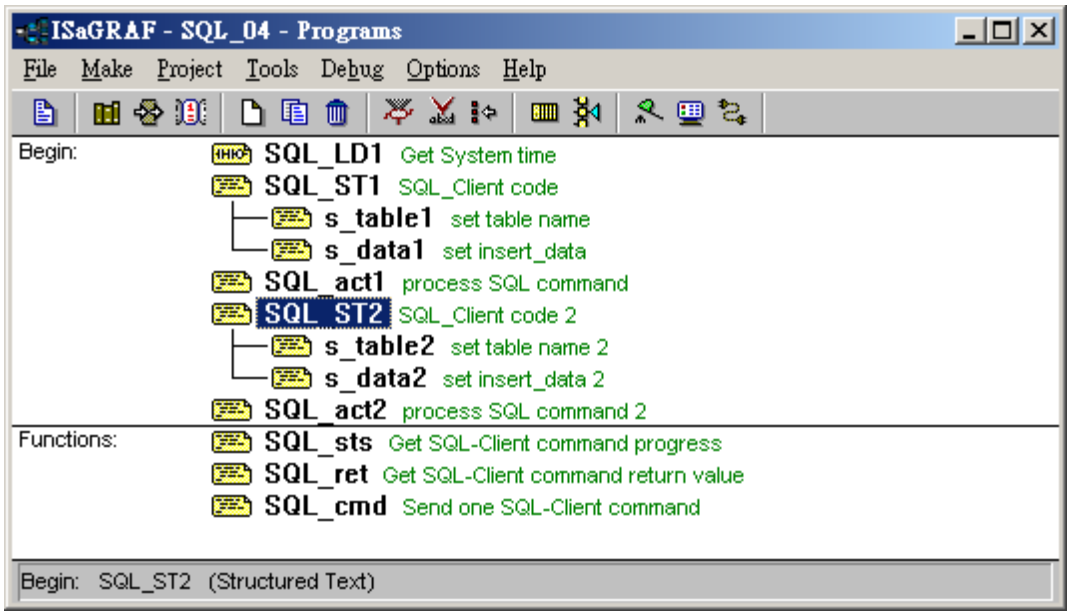
```
SQL_server1  :=  'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433; user id=sa;
                  password=AABBCC; database=DataBase_01;'   ;
```



Then you may check the table about every minute later to see if the data is updated.
(run the “SQL Server Management Studio” and key-in “select * from Table_01”)

1.4.5 : SQL_04 demo program - One PAC setup two SQL Clients to connect two servers

The “SQL_04” is similar as the “SQL_02”, however it write the temperature value to two SQL servers every minute (refer to section 1.4.3). Please prepare two SQL servers to test this demo.



First modify the following code in the “SQL_ST1” to fit your server (refer to the section 1.4.1).

```
SQL_server1 := 'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433; user id=sa;
               password=AABBCC; database=DataBase_01;' ;
```

Then modify the following code in the “SQL_ST2” to fit your server2.

```
SQL_server2 := 'server=192.168.1.164\RAVENLIU\SQLEXPRESS,1433; user id=sa; password=ABCDEF;
               database=testsql;' ;
```

Re-compile it and then download it to your PAC.

Other testing steps are similar as the “SQL_02” (refer to the section 1.4.3).

There is one another way to test this “SQL_04” demo program if there is only one SQL server. User can create one another database “DataBase_02” in the same SQL server. Then modify the value of the “SQL_server2” in the “SQL_ST2” program to point to the “DataBase_02”.

The following server setting is also Ok for some SQL servers (without PC and SQL name).

```
SQL_server1 := 'server=192.168.1.101,1433; user id=sa; password=AABBCC; database=DataBase_01;' ;
SQL_server2 := 'server=192.168.1.164,1433; user id=sa; password=ABCDEF; database=testsql;' ;
```

1.4.6 : SQL_05 demo program – Read data from an existing row in the table

The “SQL_05” reads 4 values form the “DataBase_01” > “Table_02” to the value1 to value4 variables. The ISaGRAF data type is real , integer , integer , message respectively .

To well test the “SQL_05”, first run “SQL Server Management Studio” to key-in the following commands to create one table “Table_02” and insert one row inside it .

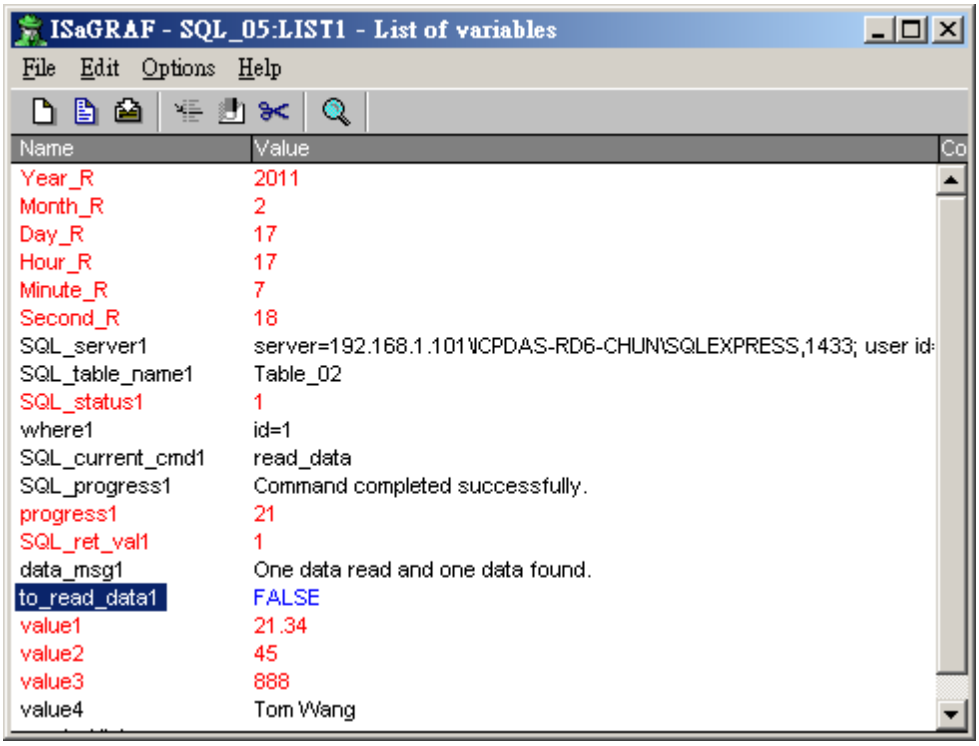
```
Create table Table_02( id int , value1 real , value2 int , value3 int , value4 varchar(255) )
Insert into table_02(id , value1 , value2 , value3 , value4) values( 1 , 21.34 , 45 , 888 , 'Tom Wang' )
Select * from table_02          (Remember to click“Execute”)
```

Then modify the following code in the “SQL_ST1” to fit your server (refer to the section 1.4.1), re-compile it and download it to the PAC.

```
SQL_server1 := 'server=192.168.1.101\ICPDAS-RD6-CHUN\SQLEXPRESS,1433; user id=sa;
               password=AABBCC; database=DataBase_01;' ;
```

The following server setting is also Ok for some SQL servers (without PC and SQL name).

```
SQL_server1 := 'server=192.168.1.101,1433; user id=sa; password=AABBCC;
               database=DataBase_01;' ;
```



This “SQL_05” demo will read the data from the SQL server once automatically when the PAC is powered up. So if everything is fine, the “value1” to “value4” should have correct data.

User can also set the “to_read_data1” as TRUe at anytime to read the data once.